Web Technologies-II

DCAP312

Edited by:
Sarabjit Kumar

LOVELY PROFESSIONAL UNIVERSITY

WEB TECHNOLOGIES-II

Edited By

Sarabjit Kumar

Printed by
EXCEL BOOKS PRIVATE LIMITED
A-45, Naraina, Phase-I,
New Delhi-110028
for
Lovely Professional University
Phagwara

SYLLABUS

Web Technologies-II

S. No.	Topics
1.	**Making Sense of .NET & Anatomy of an ASP.NET Page:** The Microsoft .NET Vision, ASP in NET, Introduction to C#, A Simple Web Page, Adding a Web Control, Introduction to In-Line Script, The Page Class.
2.	**Server Controls:** Postback, Data Binding, Web Server Controls.
3.	**Server Controls:** HTML Server Controls, Validation Controls.
4.	**Database Access:** Error Handling, Database Access Using ADO.NET, Connection, Command, DataAdapter, and DataSet, DataReader, Connection Pooling.
5.	**Creating More Advanced ASP.NET Pages:** Communicating with the Browser, Web.Config.
6.	**Creating More Advanced ASP.NET Pages:** Page Sub-classing, User Controls, More Advanced Data Binding.
7.	**Applying What We've Learned So Far:** The Database Model, Creating a Basic Object Model, Creating the User Interface.
8.	**Web Services:** XML Web Services, Uses for Web Services, Web Services in Visual Studio .NET, Creating Web Services, Expanding Web Application with Web Services.
9.	S**ecurity and Membership:** IIS Security, ASP.NET Authentication. **Adding E-Commerce Essentials:** XML Tools, Freight Calculations, Email.
10.	**Debugging and Optimization:** Debugging in an ASP.NET Application, Optimization, Optimizing Using Caching, Optimizing via Performance Profiling.

CONTENT

Unit 1: Making Sense of .NET and Anatomy of an ASP.NET Page

Notes

CONTENTS

Notes

Objectives

After studying this unit, you will be able to:

- Discuss the features of .NET
- Define challenges of building modern applications
- Discuss the .NET philosophy
- Understand the .NET platform and its layers
- Define the various components of the .NET platform
- Describe ASP in .NET
- Discuss the problems with traditional ASP
- Define the advantages of ASP.NET
- Discuss the ASP.NET architecture

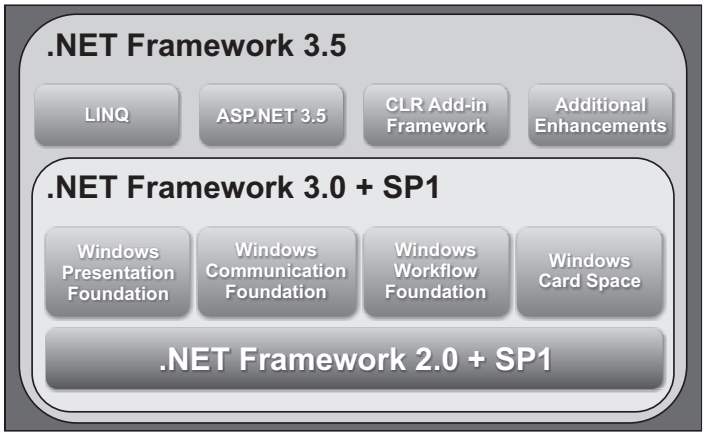
Introduction

Microsoft .Net Framework is a programming communications created by Microsoft for edifice, deploying, and operation applications and services that use .NET technologies, such as desktop applications and Web services. The Microsoft .NET Framework is a software framework that can be installed on computers running Microsoft Windows operating systems. It includes a large library of coded solutions to common programming problems and a virtual machine that manages the execution of programs written specifically for the framework.

The goal in application development is always the same: Create the best possible software in the least amount of time. Yet the bar is continually raised, as demands from customers increase. To meet these demands, the platforms developers build on and the tools they use must get better and better they must evolve.

Consider the illustration in Figure 1.1.

Figure 1.1: .Net Framework



1.1 Features of .NET

Notes

The following features are:

Rich Functionality Out Of the Box

The .NET framework provides rich set of functionality out of the box. It contains hundreds of lessons that provide assortment of functionality ready to use in your applications. This means that as a developer you need not go into low level details of many operations such as file IO, network communication and so on.

Easy Development of Web Applications

The ASP.NET is a technology accessible on .NET stage for developing dynamic and data driven web applications. ASP.NET provides an event driven programming model (similar to Visual Basic 6 that simplify development of web pages (now called as web forms) with complex user interface. ASP.NET server controls provide advanced user interface elements (like calendar and grids) that save lot of coding from programmer's side.

OOPs Support

The advantages of Object Oriented programming are well known. .NET provides a fully object oriented environment. The philosophy of .NET is "Object is mother of all." Languages like Visual Basic.NET now support many of the OO features that were lacking traditionally. Even primitive types like integer and characters can be treated as objects something not available even in OO languages like C++.

Multi-Language Support

Generally enterprises have varying skill sets. For example, a company might have people with skills in Visual Basic, C++, and Java etc. It is an experience that whenever a new language or environment is invented existing skills are outdated. This naturally increases cost of training and learning curve. .NET provides something attractive in this area. It supports multiple languages. This means that if you have skills in C++, you need not throw them but just mould them to suit .NET environment. Currently four languages are available right out of the box namely Visual Basic .NET, C# (pronounced as C-sharp), Jscript.NET and Managed C++ (a dialect of Visual C++). There are many vendors that are working on developing language compilers for other languages (20+ language compilers are already available). The beauty of multi-language support lies in the fact that even though the syntax of each language is different, the basic capabilities of each language remain at par with one another.

Multi-Device Support

Modern life style is increasingly embracing mobile and wireless devices such as PDAs, mobiles and handheld PCs. .NET provides promising platform for programming such devices. .NET Compact Framework and Mobile Internet Toolkit are step ahead in this direction.

Automatic memory management

While developing applications developers had to develop an eye on system resources like memory. Memory leaks were major reason in failure of applications. .NET takes this worry away from developer by handling memory on its own. The garbage collector takes care of freeing unused objects at appropriate intervals.

Compatibility with COM and COM+

Before the preface of .NET, COM was the de-facto standard for componentized software development. Companies have invested lot of money and efforts in developing COM components and controls. The good news is you can still use COM components and ActiveX controls under

Notes

.NET. This allows you to use your existing investment in .NET applications. .NET still relies on COM+ for features like transaction management and object pooling. In fact it provides enhanced declarative support for configuring COM+ application right from your source code. Your COM+ knowledge still remains as a valuable asset.

No more DLL Hell

If you have worked with COM components, you most likely are aware of "DLL hell". DLL conflicts are a common fact in COM world. The main reason behind this was the philosophy of COM "one version of component across machine". Also, COM components require registration in the system registry. .NET ends this DLL hell by allowing applications to use their own copy of dependent DLLs. Also, .NET components do not require any kind of registration in system registry.

Strong XML support

Now days it is hard to find a programmer who is uninformed of XML. XML has gained such a strong industry support that about all the vendors have released some kind of upgrades or patches to their existing software to make it "XML compatible". Currently, .NET is the only platform that has built with XML right into the core framework. .NET tries to harness power of XML in every possible way. In addition to providing support for manipulating and transforming XML documents, .NET provides XML web services that are based on standards like HTTP, XML and SOAP.

Ease of Deployment and Configuration

Deploying windows applications especially that used COM components were always been a tedious task. Since .NET does not require any registration as such, much of the deployment is simplified. This makes XCOPY deployment viable. Configuration is another area where .NET– especially ASP.NET – shines over traditional languages. The configuration is done via special files having special XML vocabulary. Since, most of the configuration is done via configuration files, there is no need to sit in front of actual machine and configure the application manually. This is more important for web applications; simply FTP new configuration file makes necessary changes.

Security

Windows platform was always criticized for poor security mechanisms. Microsoft has taken great efforts to make .NET platform safe and secure for enterprise applications. Features such as type safety, code access security and role based authentication make overall application more robust and secure.


Did u know?

Microsoft started the development on the .NET Framework in the late 1990's originally under the name of Next Generation Windows Services (NGWS). By late 2000 the first beta versions of .NET 1.0 were released.


Task Search more about the history of programming languages.

Self Assessment Questions

Multiple Choice Questions

1. .Net Framework is a programming infrastructure created by.....

   (*a*) Microsoft for building (*b*) Oracle development

   (*c*) Java (*d*) None of these.

Notes

2. The .NET framework provides a ..... of functionality out of the box.
   (*a*) .NET platform (*b*) Rich set
   (*c*) ASP.NET (*d*) All of these.
3. Common language runtime is .....
   (*a*) not interaction with the operating system.
   (*b*) interpreter
   (*c*) Both (*a*) and (*b*)
   (*d*) Compiles the MSIL code in .NET
4. .Net is the …….
   (*a*) Procedural Oriented programming (*b*) sequential oriented programming
   (*c*) Object Oriented programming (*d*) All of these.
5. ……. is handling memory management, garbage collection.
   (*a*) CRT (*b*) CLR
   (*c*) CRM (*d*) CTL.

1.2 The Challenges of Building Modern Applications

In Windows DNA, there are two major choices of user interfaces Win32 clients and browser based clients. During the Internet revolution of the late 90s we saw the emergence of the browser and the Web Server. With the introduction of Internet, information started being available but with limited functionality. With the development of the Windows Distributed Internet Architecture, we started to see Web sites that allowed simple transactions to occur. Clients on browsers could access Web sites that had COM components available to them that allowed them to retrieve information from the database. So now we gained the capability to simulate the environment of the Win32 platform. The client software the browser can access information on a server. But as with the Win32 environment, we are limited in the way in which the information is presented to us. Customization is neither widespread nor broadly developed.

Let us look into limitations of these technologies.

1.2.1 Limitations in Win32 Clients

In a client-server environment visual tool such as Visual Basic, are often used to create a rich user interface. The drawbacks is that such client software is difficult to deploy and maintain, requiring and install on every client and a change to every client when an upgrade is needed. DLL conflicts on the client are frequent because of variations in the version of the operating system and other software installed on the client. Visual Basic is the most common language used to write middle-tier components. This requires high level of expertise in COM. Since these middle-tire components are implemented using Microsoft Transaction Server on Windows NT or COM+ services on Windows 2000.

These components use stateless designs, which can look very different from the stateful designs often used in client-based components. COM components, in the middle tier must work together, Versioning all the components properly so that they understand each other's interfaces can be a challenge. This requires a highly sophisticated skill level and a well - controlled deployment process. COM works well on Microsoft platforms. But it suffers from lack of interoperability with other platforms. One of the most important ways functionality can be reused is for a software component to inherit another component; But COM does not support inheritance.

Notes

Visual Basic is the most popular language for developing applications with the DNA model, this is used in two major roles forms based VB Clients and COM components. This VB6 language has its own limitations it does not have the capability of multithreading, lack of OOPS concepts, Poor error handling ability and poor integration with other languages. Hence it makes it unsuitable for development of object based frameworks. Today's applications need to use the Win32 API for a variety of purposes like monitor widows messages, manipulate controls, reading and writing to INI files and socket programming etc. But these widows API are hard to program for variety of reasons, like it is not object oriented and complex calls to the functions with long lists of arguments, since Win32 API is written in C++ language, getting calling conventions right on data types is messy.

1.2.2 Limitations in DNA Based Internet Development or Browser Based Clients

With DNA-based software development, creating software that is accessed by a user locally is done very differently from development for the Internet. The Visual Basic forms for client-server user interfaces versus the use of Active Server Pages for Internet user interfaces. Even though both situations involve designing and implementing GUI based user interfaces the tools and programming techniques used are quite different. ASP lacks in state management between post backs.

Every time a page is rendered, the programmer must make sure that all the visual controls like text boxes, dropdowns have their information loaded. It is the programmer's responsibility to manage the state in the user interface and to transfer state information between pages. This causes developers to have to write a lot of code for the internet user interfaces that is not relevant to business problem being solved. If the Internet application is going to run on a group of Web Servers, then considerable additional work is necessary to design a state management system that is independent of particular server.

Browser based clients are somewhat more difficult to create, and offer a more limited user interface with fewer controls and less control over layout of the screen and handling of screen events. It is possible to create rich user interfaces using DHTML, but it requires lot of coding and also browser compatibility issues rises, for which a separate coding or two version of the same page have to be maintained, keeping in mind, the browser we are targeting. The Internet has caused server based applications to become much more popular than ever before and has made the connectionless request/response programming model common. But communicating between servers especially among those running on different platforms is difficult, and because most substantial Internet applications are Database Centric, the ability to access a wide variety of data sources easily is more important than ever. As we move on to handheld devices or wireless devices, kiosks or other type of systems, many of which run a different processors and do not use standard operating system. So sharing the data between these devices and communication varies which is not uniform, becomes difficult.

1.3 .NET Philosophy

The driving force behind Microsoft .NET is a shift in focus from individual Web sites or devices to new constellations of computers, devices, and services that work together to deliver broader, richer solutions.

The platform, technology that people use is changing. Since 1992, the client/server environment has been in place, with people running the applications they need on the Win32 platform, for example. Information is supplied by the databases on the servers, and programs that are installed on the client machine determine how that information is presented and processed.

1.3.1 Microsoft.NET Solutions

Notes

- ***Single Programming Model:*** A related goal is to have development for the internet environment look very much like development for other types of software. Likewise, developing user interfaces in Windows Forms is very similar to developing them in Web Forms. There are commonly used controls, such as Labels and Text Boxes, in both, with similar sets of properties and method. The amount of commonality makes it easy to transition between the two types of development and easier for traditional VB developers to start using Web Forms.
- ***Distributed Systems:*** The Vision of Microsoft.NET is globally distributed systems, using XML as the universal glue to allow functions running on different computers across an organization or across the world to come together in a single application. In this vision, systems from servers to Wireless Palmtops, with everything in between, will share the same general platform, with versions of .NET available for all of them, and with each of them able to integrate transparently with the others.
- ***Richer User Interface:*** Web Forms are a giant step towards much richer web-based user interfaces. Their built in intelligence allows rich, browser independent screens to be developed quickly, and to be easily integrated with compiled code. Microsoft has announced an initiative for the future called the Universal Canvas which builds upon the XML standards to transform the internet from a Read only environment into a read/write platform, enabling users to interactively create, browse, edit and analyze information. The universal canvas can bring together multiple sources of information anywhere in the world to enable seamless data access and use. (The universal canvas will log on to the Ms System of servers whenever the new device is turned on) Centrally controlled OS, Office and Visual Studio.
- ***Easy Deployment:*** Executable modules in .NET are self describing. Once the Common Language Runtime (CLR is explained in next sections) knows where a module resides, it can find out everything else it needs to know to run the module, such as the module's object interface and security requirements, from the module itself. That means a module can just be copied to a new environment and immediately executed.
- ***Support for Multiple Languages:*** The CLR executes binary code called MSIL (Microsoft intermediate language), and that code looks the same regardless of the original source language. All .NET enabled languages use the same data types and the same interfacing conventions. This makes possible for all .NET language to interoperate transparently. One language can call another easily, and languages can even inherit classes written in another language and extend them current platform has anywhere near this level of language interoperability.
- ***Extendibility:*** The completely object based approach of .NET is designed to allow base functionality to be extended through inheritance (unlike COM) and the platform's functionality is appropriately partitioned to allow various parts (such as the just-in-time compilers discussed in the next section) to be replaced as new versions are needed. It is likely that, in the future, new ways of interfacing to the outside world will be added to the current trio of windows Form, Web Forms, and Web Services such as universal Canvas.
- ***Portability of Compiled Applications:*** .NET allows the future possibility of moving software to other hardware and operating system platforms. The ultimate goal is that compiled code produced on one implementation of .NET (such as Windows) could be moved to another implementation of .NET on a different operating system merely by copying the compiled code over and running it.

Notes

- ***Integrity with COM:*** .NET integrates very well with COM based software. Any COM component can be treated as a .NET component by other .NET components. The .NET Framework wraps COM components and exposes an interface that .NET components can work with. This is absolutely essential to the quick acceptance of .NET, because it makes .NET interoperable with a tremendous amount of older COM-based software.

1.3.2 Other Benefits of Using .NET Architecture

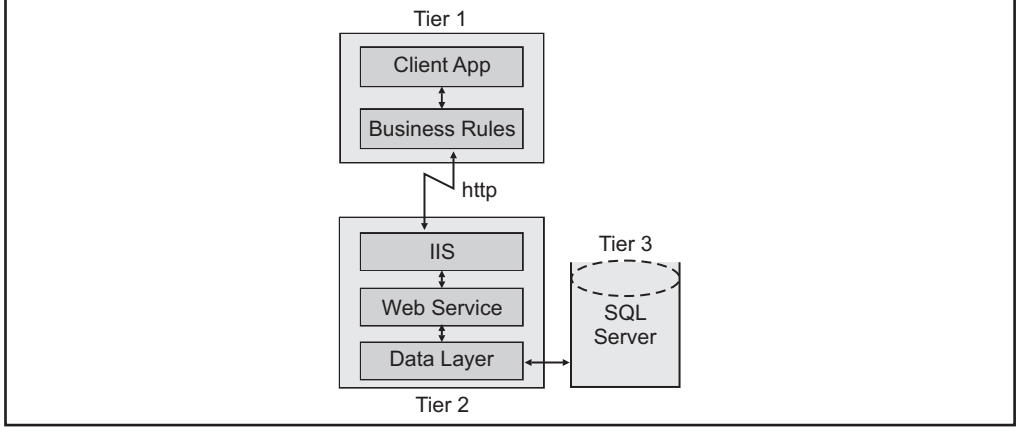
- The Microsoft .NET platform's reliance on XML for data exchange an open standard managed by the World Wide Web Consortium (W3C) and modular XML Web services removes barriers to data sharing and software integration.
- The .NET platform, through the .NET Framework's common language runtime, enables XML Web services to interoperate whatever their source language. Developers can build reusable XML Web services instead of monolithic applications. By making it easy to offer your XML Web services to others.
- The ability to easily find available XML Web services means you can buy pieces of your applications rather than build everything from scratch, focusing your time and money where it makes the most sense.
- Easier to build sophisticated development tools debuggers and profilers can target the Common Language Runtime, and thus become accessible to all .NET enabled languages.
- Potentially better performance in system level code for memory management, garbage collection, and the like have yielded an architecture that should meet or exceed performance of typical COM based applications today. Fewer bugs, as whole classes of bugs should be unknown in .NET. With the CLR handling memory management, garbage collection.
- Faster development using development tool like Visual Studio .NET.

1.3.3 N-tier Architecture with .NET

Applications developed in the .NET Framework will still, in, many cases, use a DNA model to design the appropriate tiers. However, the tiers will be a lot easier to produce in .NET. The presentation tier will benefit from the new interface technologies and especially Web Forms for Internet development. The middle tier will require far less COM related headaches to develop and implement. And richer, more distributed middle tier designs will be possible by using Web Services.

Let us look into how .Net fit into N-tier architecture. When you talk about a true distributed N-tier type of application, you are talking about separating the components of the different tiers on different machines as well as in separate components.

Figure 1.2: N-tier architecture

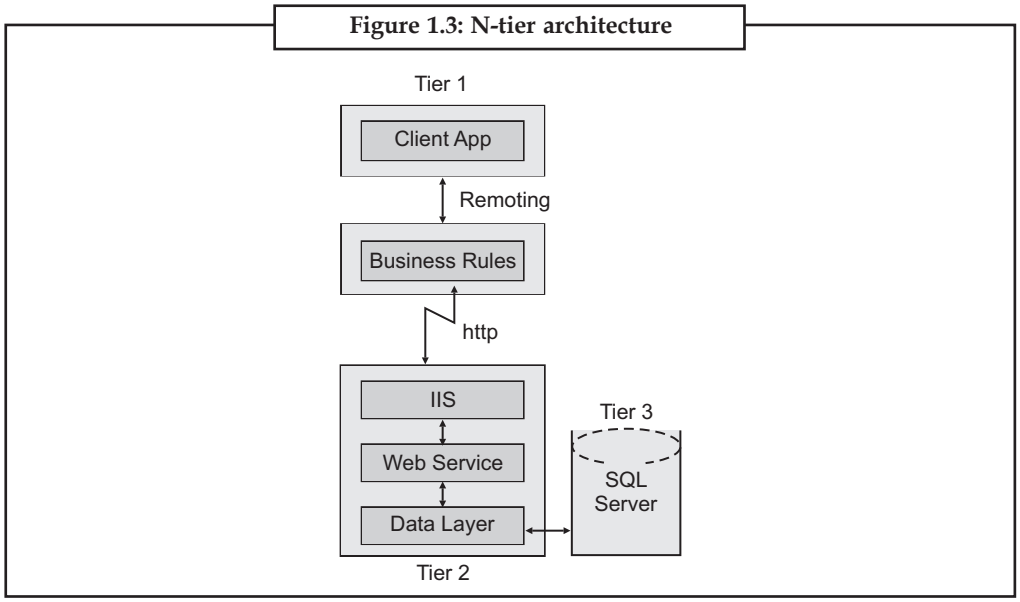


Notes

Figure 1.2 shows a typical example of a N-tier application with multiple components on each machine. There are many different ways you could configure a N-tier application.

For example, the business rules may go on a separate machine and you might use .NET remoting to talk from the client application to the business rule tier as shown in Figure: 1.3

Figure 1.3: N-tier architecture



We may also have a data input validation rule component on the client to check simple rules such as required fields and formatting. These are rules that you do not want to make a trip across the network just to check. You may then also add a business rule layer on the same tier as the data layer component to check complicated business rules that compare the data from one table to another.

These are just a few different configurations that you may utilize. Of course, you could come up with something unique that fits your specific situation. Regardless of how you structure the physical implementation of the components, make sure that the logical structure of the program is broken up into components as shown in the above Figure 1.2 and Figure 1.3.

Task Search more about the history of programming languages.

1.4 Understanding the .NET Platform and its Layers

The .NET Platform is made up of and we will describe its layers. To commence, .NET is a framework that covers all the layers of software progress above the Operating System. It provides the richest level of integration among presentation technologies, component technologies, and data technologies ever seen on Microsoft, or perhaps any, platform. Secondly, the entire architecture has been created to make it easy to develop Internet applications, as it is to develop for the desktop.

1.4.1 Constituents of .NET Platform

The .NET consists of the following three main parts:

- .NET Framework – A completely re-engineered development environment.

Notes

- .NET Products – Applications from MS based on the .NET platform, including Office and Visual Studio.
- .NET Services – Facilitates 3rd party developers to create services on the .NET Platform.

1.4.2 The Common Language Runtime (CLR)

At the base is the CLR. It is considered as the heart of the .NET framework. .NET applications are compiled to a common language known as Microsoft Intermediate Language or "IL". The CLR, then, handles the compiling the IL to machine language, at which point the program is executed.

The CLR environment is also referred to as a managed environment, in which common services, such as garbage collection and security, are automatically provided.

1.4.3 The .NET Class Framework

The next layer up in the framework is called the .NET Class Framework also referred as .NET base class library. The .NET Class Framework consists of several thousand type definitions, where each type exposes some functionality. All in all, the CLR and the .NET Class Framework allow developers to build the following kinds of applications:

- Web Services. Components that can be accessed over the Internet very easily.
- Web Forms. HTML based applications (Web Sites).
- Windows Forms. Rich Windows GUI applications. Windows form applications can take advantage of controls; mouse and keyboard events and can talk directly to the underlying OS.

1.4.4 User Interface

The next layer consists of the user and programming interface that allows .NET to interact with the outside world. The following are the types of interaction interfaces that are supported by the .NET framework:

- Web Forms
- Windows Forms
- Web Services

Now let me tell you about Windows Forms and ASP.NET. WinForms (Windows Forms) is simply the name used to describe the creation of a standard Win32 kind of GUI applications.

The Active Server Pages web development framework has undergone extensive changes in ASP. NET. The programming language of choice is now full blown VB.NET or C# (or any supported .NET language for that matter). Other changes include:

- New support for HTML Server Controls (session state supported on the server).
- It is now possible for the server to process client-side events.
- New control families including enhanced Intrinsic, Rich controls, List controls, DataGrid control, Repeater control, Data list control, and validation controls.
- New support for developing Web Services–application logic programmatically accessible via the Internet that can be integrated into .NET applications using the Simple Object Access Protocol (SOAP).

1.4.5 Languages

The CLR allows objects created in one language be treated as equal citizens by code written in a completely different language. To make this possible, Microsoft has defined a Common

Notes

Language Specification (CLS) that details for compiler vendors the minimum set of features that their compilers must support if they are to target the runtime.

Any language that conforms to the CLS can run on the CLR. In the .NET framework, Microsoft provides Visual Basic, Visual C++, Visual C#, and JScript support.

1.4.6 .NET Products

Microsoft Visual Studio .NET

Microsoft Visual Studio .NET represents the best development environment for the .NET platform.

Integrations is the key in the new VS.NET IDE, thus a single IDE can be used to program in a variety of managed languages from VB.NET to Visual C++ with Managed extensions. Advance features in VS.NET truly propel development in to the highest gear.

1.4.7 .NET Services

XML Web Services

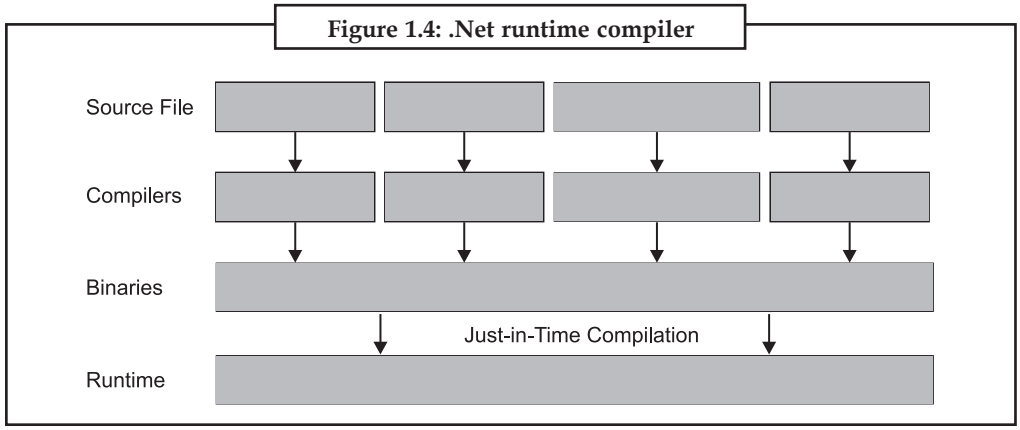
The XML is turning the way we build and use software inside out. The Web revolutionized how users talk to applications. XML is revolutionizing how applications talk to other applications or more broadly, how computers talk to other computers by providing a universal data format that lets data be easily adapted or transformed:

- XML Web services allow applications to share data.
- XML Web services are discrete units of code; each handles a limited set of tasks.
- They are based on XML, the universal language of Internet data exchange, and can be called across platforms and operating systems, regardless of programming language.
- .NET is a set of Microsoft software technologies for connecting your world of information, people, systems, and devices through the use of XML Web services.

1.4.8 .NET Runtime

The .NET Framework provides a run time environment called the Common Language Runtime (See Figure 1.4), which manages the execution of code and provides services that make the development process easier. Compilers and tools expose the runtime's functionality and enable you to write code that benefits from this managed execution environment. Code developed with a language compiler that targets the runtime is called managed code.

Figure 1.4: .Net runtime compiler



To enable the runtime to provide services to managed code, language compilers must emit metadata, which the runtime uses to locate and load classes, lay out instances in memory, resolve

Notes

method invocations, generate native code, enforce security, and set run time context boundaries. The runtime automatically handles objects, releasing them when they are no longer being used. Objects whose lifetimes are managed in this way are called managed data. Automatic memory management eliminates memory leaks as well as many other common programming errors.

The CLR makes it easy to design components and applications whose objects interact across languages. For example, you can define a class and then use a different language to derive a class from your original class, or call a method on the original class. You can also pass an instance of a class to a method on a class written in a different language. This cross language integration is possible because of the common type system defined by the runtime, and they follow the runtime's rules for defining new types, as well as for creating, using, persisting, and binding to types.

Language compilers and tools expose the runtime's functionality in ways that are intended to be useful and intuitive to their developers. This means that some features of the runtime might be more noticeable in one environment than in another. How you experience the runtime depends on which language compilers or tools you use. The following benefits of the runtime might be particularly interesting to you:

- Performance improvements.
- The ability to easily use components developed in other languages.
- Extensible types provided by a class library.
- A broad set of language features.

1.5 Understanding the Various Components of the .NET Platform

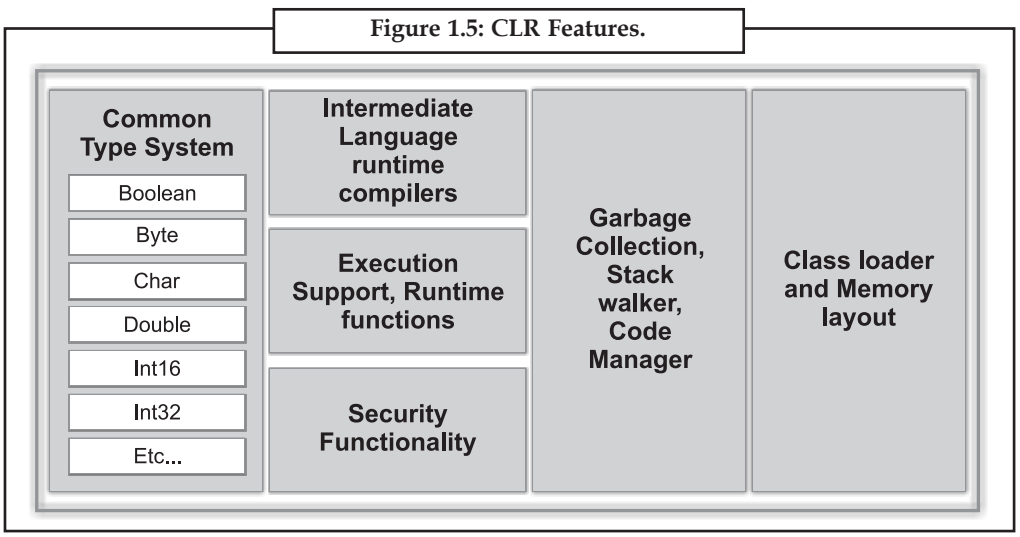
The .NET framework has following comppnents:

1.5.1 Common Language Runtime

At the core of the .NET platform is the Common Language Runtime (CLR). The CLR simplifies application development, provides a robust and secure execution environment, supports multiple languages and simplifies application deployment and management.

The Figure 1.5 below provides more details on the CLR's features:

Figure 1.5: CLR Features.



Notes

Now we discuss some of the more significant features provided to .NET applications by the CLR. These include:

- Memory Management
- Common Type System

Before moving further let us discuss briefly about Common Language Infrastructure (CLI) according to Standardizing Information and Communication Systems (ECMA) specifications. The Microsoft Shared Source CLI Implementation is a file archive containing working source code for the ECMA-334 (C#) and ECMA-335 (Common Language Infrastructure, or CLI) standards. In addition to the CLI implementation and the C# compiler, the Shared Source CLI Implementation from Microsoft called ROTOR contains tools, utilities, additional Framework classes, and samples.

For the benefit of existing codebases, the CLI standard also takes pains to describe in detail how unmanaged software can co-exist safely with managed components, enabling seamless sharing of computing resources and responsibilities. Like the C runtime, the CLI has been designed to exploit the power of diverse platforms, as well as to complement existing tools, languages, and runtimes. Let us look at a few of the likely ways that the Shared Source CLI Implementation might interest you:

- There are significant differences in implementation between this code and the code for Microsoft's commercial CLR implementation, both to facilitate portability and to make the code base more approachable. If you are a developer who is interested in knowing how JIT compilers and garbage collectors work, or of how Microsoft Visual Studio works on your behalf under the covers, this distribution will definitely hold your attention!
- The distribution will help you in creating courseware around interesting topics that can be illustrated by this codebase.
- The distribution will help you in implementing your own version of the CLI and it also helps you in understanding the way the compilers and tools target the CLI.

1.5.2 Automatic Memory Management

A major feature of .NET framework CLR is that the runtime automatically handles the allocation and release of an object's memory resources.

Automatic memory management enhances code quality and developer productivity without negatively impacting expressiveness or performance.

The Garbage Collector (GC) is responsible for collecting the objects no longer referenced by the application. The GC may automatically be invoked by the CLR or the application may explicitly invoke the GC by calling GC.Collect. Objects are not released from memory until the GC is invoked and setting an object reference to nothing does not invoke the GC, a period of time often elapses between when the object is no longer referenced by the application and when the GC collects it.

1.5.3 Common Type System

The Common Type System defines how data types are affirmed, used, and managed in the runtime, and is also a significant part of the runtime's bear for the Cross-Language incorporation. The common type system performs the following functions:

- Establishes a framework that enables cross-language integration, type safety, and high performance code execution.
- Provides an object-oriented model that supports the complete implementation of many programming languages.

Notes

- Defines rules that languages must follow, which helps ensure that objects written in different languages can interact with each other.

The Common Type System can be divided into two general categories of types, Reference type and Value type each of which is further divided into subcategories

1.5.4 Common Type System Architecture

The .NET type system has two unlike kinds of types specifically Value types and situation types.

Value types directly enclose the data, and instances of value types are either billed on the stack or allocated inline in a structure. Value types can be built-in (implemented by the runtime), user-defined, or enumerations.

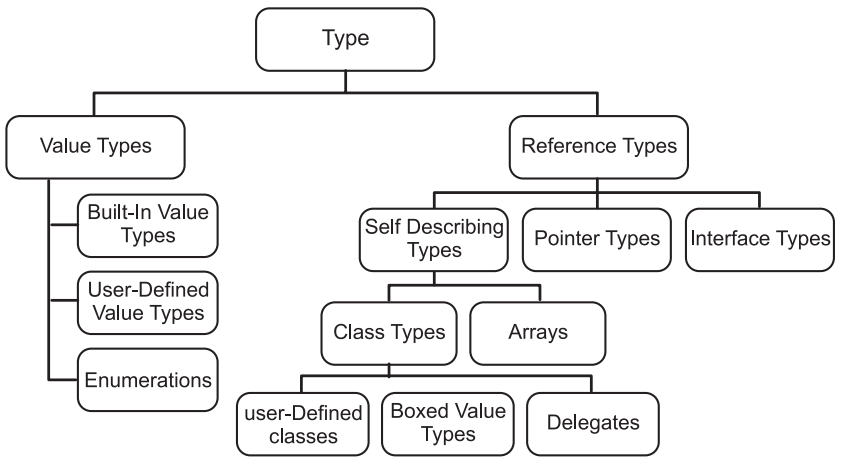
The core value types supported by the .NET platform reside within the root of the System namespace. There types are often referred to as the .NET "Primitive Types".

They include:

- Boolean
- Byte
- Char
- DateTime
- Decimal
- Double
- Guid
- Int16
- Int32
- Int64
- SByte
- Single
- Timespan

Reference types store a reference to the value's memory address, and are allocated on the heap. Reference types can be self describing types, pointer types, or interface types. The type of a reference type can be determined from values of self describing types. Self describing types are further split into arrays and class types.

Figure 1.6: Kind of Types



1.6 ASP in .NET

Notes

This is the engine that hosts the web applications you create with .NET, and supports almost any feature from the .NET class library. ASP.NET also includes a set of web-specific services, like secure authentication and data storage.

ASP.NET takes an object-oriented programming approach to Web page execution. Every element in an ASP.NET page is treated as an object and run on the server. An ASP.NET page gets compiled into an intermediate language by a .NET Common Language Runtime compliant compiler. Then a JIT compiler turns the intermediate code to native machine code, and that machine code is eventually run on the processor. Because the code is run straight from the processor; pages load much faster than classic ASP pages, where embedded VBScript or JScript had to be continuously interpreted and cached. ASP.NET is used to create Web pages and Web services and is an integral part of Microsoft's .NET vision.

1.6.1 Many Faces of ASP.NET

With ASP.NET 3.5, Microsoft aims to persist its success by calming and attractive ASP.NET. The good news is that Microsoft has not removed features, replaced functionality, or reversed direction. Instead, almost all the changes add higher-level features that can make your programming more productive.

All in all, there have been four major releases of ASP.NET:

- *ASP.NET 1.0:* This first release created the core ASP.NET platform and introduced a wide range of essential features.
- *ASP.NET 1.1:* This second release added performance tune ups and bug fixes, but no new features.
- *ASP.NET 2.0:* This third release piled on a huge set of new features, all of which were built on top of the existing ASP.NET plumbing. The overall emphasis was to supply developers with prebuilt goodies that they could use without writing much (if any) code. Some of the new features included built in support for website navigation, a theming feature for standardizing web page design, and an easier way to pull information out of a database.
- *ASP.NET 3.5:* This fourth release keeps the same basic engine as ASP.NET 2.0, but adds a few frills and two more dramatic changes. The most significant enhancement is the ASP.NET AJAX toolkit, which gives web developers better tools for creating highly responsive web pages that incorporate rich effects usually seen in desktop applications (such as drag and drop and auto complete). The other innovation is support for LINQ, a set of language enhancements included with .NET 3.5 that allows you to search in memory data in the same way that you query a database.

Microsoft used the .NET 3.0 names to release a set of new technologies, including Windows Presentation Foundation (WPF), a platform for building slick Windows applications; Windows Workflow Foundation (WF), a platform for modelling application logic using flowchart style diagrams; and Windows Communication Foundation (WCF), a platform for designing services that can be called from other computers. However, .NET 3.0 did not include an updated version of ASP.NET.

Did u know?

ASP.NET 1.0 was released on January 5, 2002 as part of version 1.0 of the .NET Framework.

Notes

1.7 Problems with Traditional ASP

There are many problems with ASP if you think of needs for Today's powerful Web applications:

1. ***Interpreted and Loosely Typed Code***

   ASP scripting code is usually written in languages such as JScript or VBScript. The script execution engine that Active Server Pages relies on interprets code line by line, every time the page is called. In addition, although variables are supported, they are all loosely typed as variants and bound to particular types only when the code is run. Both these factors impede performance, and late binding of types makes it harder to catch errors when you are writing code.

2. ***Mixes Layout (HTML) and Logic (Scripting Code)***

   ASP files frequently combine script code with HTML. This results in ASP scripts that are lengthy, difficult to read, and switch frequently between code and HTML. The interspersion of HTML with ASP code is particularly problematic for larger web applications, where content must be kept separate from business logic.

3. ***Limited Development and Debugging Tools***

   Microsoft Visual InterDev, Macromedia Visual UltraDev, and other tools have attempted to increase the productivity of ASP programmers by providing graphical development environments. However, these tools never achieved the ease of use or the level of acceptance achieved by Microsoft Windows application development tools, such as Visual Basic or Microsoft Access. ASP developers still rely heavily or exclusively on Notepad.

   Debugging is an unavoidable part of any software development process, and the debugging tools for ASP have been minimal. Most ASP programmers resort to embedding temporary Response. Write statements in their code to trace the progress of its execution.

4. ***No Real State Management***

   Session state is only maintained if the client browser supports cookies. Session state information can only be held by using the ASP Session object. And you have to implement additional code if you, for example, want to identify a user.

5. ***Obscure Configuration Settings***

   The configuration information for an ASP web application (such as session state and server timeouts) is stored in the IIS metabase. Because the metabase is stored in a proprietary format, it can only be modified on the server machine with utilities such as the Internet Service Manager. With limited support for programmatically manipulating or extracting these settings, it is often an arduous task to port an ASP application from one server to another.


Caution

If your Web application makes use of components, copying new files to your application should only be done when the Web server is stopped. Otherwise it is like pulling the rug from under your application's feet, because the components may be in use and locked.

1.8 Advantages of ASP.NET

Following are the advantages of the ASP.NET are as:

1. ***Separation of Code from HTML***

   To make a clean sweep, with ASP.NET you have the ability to completely separate layout and business logic. This makes it much easier for teams of programmers and designers to

Notes

collaborate efficiently. This makes it much easier for teams of programmers and designers to collaborate efficiently.

2. ***Support for Compiled Languages***

   Developer can use VB.NET and access features such as strong typing and object-oriented programming. Using compiled languages also means that ASP.NET pages do not suffer the performance penalties associated with interpreted code. ASP.NET pages are precompiled to byte-code and Just In Time (JIT) compiled when first requested. Subsequent requests are directed to the fully compiled code, which is cached until the source changes.

3. ***Use Services Provided By the .NET Framework***

   The .NET Framework provides class libraries that can be used by your application. Some of the key classes help you with Input/Output, access to operating system services, data access, or even debugging. We will go into more detail on some of them in this module.

4. ***Graphical Development Environment***

   Visual Studio .NET provides a very rich development environment for Web developers. You can drag and drop controls and set properties the way you do in Visual Basic 6. And you have full IntelliSense support, not only for your code, but also for HTML and XML.

5. ***State Management***

   To refer to the problems mentioned before, ASP.NET provides solutions for session and application state management. State information can, for example, be kept in memory or stored in a database. It can be shared across Web farms, and state information can be recovered, even if the server fails or the connection breaks down.

6. ***Update Files While the Server is Running***

   Components of your application can be updated while the server is online and clients are connected. The Framework will use the new files as soon as they are copied to the application. Removed or old files that are still in use are kept in memory until the clients have finished.

7. ***XML-Based Configuration Files***

   Configuration settings in ASP.NET are stored in XML files that you can easily read and edit. You can also easily copy these to another server, along with the other files that comprise your application.

Here are some points that give the quick overview of ASP.NET.

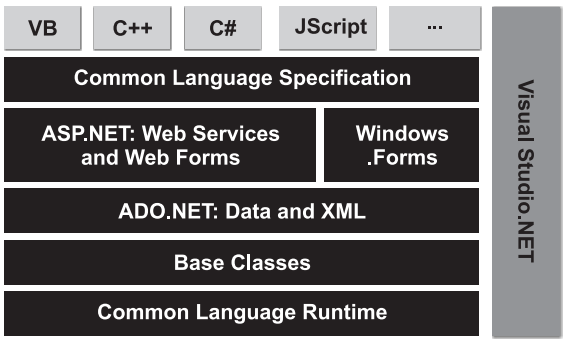
- ASP.NET provides services to allow the creation, deployment, and execution of Web Applications and Web Services.
- Like ASP, ASP.NET is a server side technology.
- Web Applications are built using Web Forms. ASP.NET comes with built in Web Forms controls, which are responsible for generating the user interface. They mirror typical HTML widgets like text boxes or buttons. If these controls do not fit your needs, you are free to create your own user controls.
- Web Forms are designed to make building web based applications as easy as building Visual Basic applications.

Notes

1.9 ASP.NET Architecture

ASP.NET is based on the architecture of .NET Framework. Visual studio provides a uniform way to combine the various features of this Architecture.

Figure 1.7: ASP.NET Architecture



Architecture is explained form underside to top in the following discussion.

1. At the bottom of the Architecture is Common Language Runtime. .NET Framework common language runtime resides on top of the operating system services. The common language runtime loads and executes code that targets the runtime. This code is therefore called managed code. The runtime gives you, for example, the ability for cross language integration.

2. .NET Framework provides a rich set of class libraries. These include base classes, like networking and Input/Output classes, a data class library for data access, and classes for use by programming tools, such as debugging services. All of them are brought together by the Services Framework, which sits on top of the common language runtime.

3. ADO.NET is Microsoft's ActiveX Data Object (ADO) model for the .NET Framework. ADO. NET is not simply the migration of the popular ADO model to the managed environment but a completely new paradigm for data access and manipulation.

   ADO.NET is intended specifically for developing web applications. This is evident from its two major design principles:

   1. Disconnected Datasets In ADO.NET, almost all data manipulation is done outside the context of an open database connection.

   2. Effortless Data Exchange with XML Datasets can converse in the universal data format of the Web, namely XML.

4. The 4th layer of the framework consists of the Windows application model and, in parallel, the Web application model. The Web application model in the slide presented as ASP. NET-includes Web Forms and Web Services.

   ASP.NET comes with built in Web Forms controls, which are responsible for generating the user interface. They mirror typical HTML widgets like text boxes or buttons. If these controls do not fit your needs, you are free to create your own user controls.

   Web Services brings you a model to bind different applications over the Internet. This model is based on existing infrastructure and applications and is therefore standard-based, simple, and adaptable.

Notes

Web Services are software solutions delivered via Internet to any device. Today, that means Web browsers on computers, for the most part, but the device agnostic design of .NET will eliminate this limitation.

5. One of the obvious themes of .NET is unification and interoperability between various programming languages. In order to achieve this; certain rules must be laid and all the languages must follow these rules. In other words we cannot have languages running around creating their own extensions and their own fancy new data types. CLS is the collection of the rules and constraints that every language (that seeks to achieve .NET compatibility) must follow.
6. The CLR and the .NET Frameworks in general, however, are designed in such a way that code written in one language can not only seamlessly be used by another language. Hence ASP.NET can be programmed in any of the .NET compatible language whether it is VB.NET, C#, Managed C++ or JScript.NET.

File Name Extensions

Web applications written with ASP.NET will consist of many files with different file name extensions. The most common are listed here. Native ASP.NET files by default have the extension .aspx (which is, of course, an extension to .asp) or .ascx. Web Services normally have the extension .asmx.

Your file names containing the business logic will depend on the language you use. So, for example, a C# file would have the extension .aspx.cs. You already learned about the configuration file Web.Config.

Another one worth mentioning is the ASP.NET application file Global.asax in the ASP world formerly known as Global.asa. But now there is also a code behind file Global.asax.vb, for example, if the file contains Visual Basic.NET code. Global.asax is an optional file that resides in the root directory of your application, and it contains global logic for your application.

Directives

You can use directives to specify optional settings used by the page compiler when processing ASP.NET files. For each directive you can set different attributes. One example is the language directive at the beginning of a page defining the default programming language.

Code Declaration Blocks

Code declaration blocks are lines of code enclosed in <script> tags. They contain the runat=server attribute, which tells ASP.NET that these controls can be accessed on the server and on the client. Optionally you can specify the language for the block. The code block itself consists of the definition of member variables and methods.

Code Render Blocks

Render blocks contain inline code or inline expressions enclosed by the character sequences shown here. The language used inside those blocks could be specified through a directive like the one shown before.

HTML Control Syntax

You can declare several standard HTML elements as HTML server controls. Use the element as you are familiar with in HTML and add the attribute runat=server. This causes the HTML element to be treated as a server control. It is now programmatically accessible by using a unique ID. HTML server controls must reside within a <form> section that also has the attribute runat=server.

Notes

Custom Control Syntax

There are two different kinds of custom controls. On the one hand there are the controls that ship with .NET, and on the other hand you can create your own custom controls. Using custom server controls is the best way to encapsulate common programmatic functionality.

Just specify elements as you did with HTML elements, but add a tag prefix, which is an alias for the fully qualified namespace of the control. Again you must include the runat=server attribute. If you want to get programmatic access to the control, just add an Id attribute.

You can include properties for each server control to characterize its behavior. For example, you can set the maximum length of a TextBox. Those properties might have sub properties; you know this principle from HTML. Now you have the ability to specify, for example, the size and type of the font you use (font-size and font-type).

The last attribute is dedicated to event binding. This can be used to bind the control to a specific event. If you implement your own method MyClick, this method will be executed when the corresponding button is clicked if you use the server control event binding shown in the slide.

Data Binding Expression

You can create bindings between server controls and data sources. The data binding expression is enclosed by the character sequences <%# and %>. The data binding model provided by ASP.NET is hierarchical. That means you can create bindings between server control properties and superior data sources.

Server-side Object Tags

If you need to create an instance of an object on the server, use server-side object tags. When the page is compiled, an instance of the specified object is created. To specify the object use the identifier attribute. You can declare (and instantiate) .NET objects using class as the identifier, and COM objects using either progid or classid.

Server-side Include Directives

With server-side include directives you can include raw contents of a file anywhere in your ASP.NET file. Specify the type of the path to filename with the pathtype attribute. Use either File, when specifying a relative path, or Virtual, when using a full virtual path.

Server-side Comments

To prevent server code from executing, use these character sequences to comment it out. You can comment out full blocks not just single lines.


Task

Prepare a list to show all the languages used in .NET environment.


Case Study

Online Banking Solution Benefits after ASP.NET MVC Replaces Ruby on Rails, Linux

Jwaala created its MoneyTracker online personal finance management solution using Ruby on Rails development tools for deployment on a Linux operating system, Apache Web server, and MySQL stack. When the company found that its banking and credit union customers, who use MoneyTracker to give users a richer online banking experience, were more comfortable with the Microsoft Application Platform, Jwaala rewrote its application

Contd...

Notes

using ASP.NET MVC for deployment using Microsoft SQL Server 2008 Enterprise (64-bit) database software running on the Windows Server 2008 Enterprise for 64-Bit Systems operating system. While Jwaala developers were fans of Ruby on Rails, they have found that ASP.NET MVC, Microsoft Visual Studio 2008, and other Microsoft tools and technology combine to create a more efficient development environment.

Situation

Jwaala, creator of the MoneyTracker online banking solution, prides itself in helping to change the face of Internet banking with its personal finance management innovations. The company, based in Austin, Texas, chose as its name the Hindi word for passion. A passion for online banking and personal financial management is seen throughout MoneyTracker, which includes integrated account aggregation, natural language searching, personal financial management, consumer remote deposit, secure document repository, widgets, and infinitely customizable alerting options.

Developers at Jwaala liked working with Ruby on Rails because it provides a framework and tools for creating Web-based applications, and because of its use of the Model View Controller (MVC) architecture that simplifies application programming. However, the company found that many of its potential customers in the financial industry had limited experience about deploying an application that required the LAMR stack.

Questions

1. Discuss the initial problem of online banking.
2. What are the benefits of ASP.NET MVC?

Self Assessment Questions

6. Which of the following are correct controls in ASP.NET Source View?

   (*a*) <asp:textbox runat= "server" id= "Msg" text= "Hello, R4R" />

   (*b*) <asp:button runat= "server" id= "Send" text= "Send" onclick= "Send_Click" />

   (*c*) None (*d*) Both.

7. Which is used to add event code (on C# Code) for TextBox control in Source view (ASP. NET).

   (*a*) MsgSent.Text = Msg.Text; (*b*) MsgSent.Text = Msg.Text

   (*c*) None (*d*) Both.

8. Which is code allows the modification of the head tag at runtime?

   (*a*) < head runat= "server"> (*b*) < script javascript= "server">

   (*c*) None (*d*) Both.

9. Correct syntax of ASP.NET directive

   (*a*) <%@ directive [attribute=value] %>.

   (*b*) <% directive [attribute=value] %>.

   (*c*) <@ directive [attribute=value] %>.

   (*d*) <@ directive [attribute=value] >.

Notes

10. The namespaces that are automatically imported into all ASP.NET pages
    1) System,System.Collections, System.IO
    2) System.Web,System. Web.UI
    3) System.Web.UI.HtmlControls,System. Web.UI.WebControls

    (*a*) 1 (*b*) 1, 2 and 3

    (*c*) 2 and 3 (*d*) None.

11. Which is true to bind a method by setting the Master attribute in the <pages>?

    (*a*) < configuration>
    < system.web>
    < pages masterpagefile= "r4r.co.in.MasterPage" />
    </system.web>
    </configuration>

    (*b*) < pages>
    < system.web>
    < pages masterpagefile= "r4r.co.in.MasterPage" />
    </system.web>
    </ pages>

    (*c*) Both (*d*) None

12. Syntax for Register directive specifies

    (*a*) < Register runat= "server" [language= "language"]>
    //code…….
    </ Register >

    (*b*) <%@ Assembly Name= "r4r" %>

    (*c*) 1 and 2 (***d***) None.

13. ASP.NET Supports "Nested Master Page"

    (*a*) Yes (*b*) No

14. Choose correct for Custom Server Controls
    1) Two ways of creating custom server controls.
    2) The pagelet approach, which is easy to do but rather limited in functionality.
    3) Control base class (or UserControl) derivative approach

    (*a*) 1, 2 and 3 (*b*) 1 and 2

    (*c*) 3 (*d*) 1 and 3

15. Choose all true options only

    (*a*) HtmlImage is HTMLControl for image and it used for html tag

    (*b*) HtmlInputHidden Used for hidden just like form fields

    (*c*) Both (*d*) None.

Notes

1.10 Summary

- Microsoft .Net Framework is a programming infrastructure created by Microsoft for building, deploying, and running applications and services that use .NET technologies, such as desktop applications and Web services.
- .NET is a framework that covers all the layers of software development above the Operating System. It provides the richest level of integration among presentation technologies, component technologies, and data technologies ever seen on Microsoft, or perhaps any, platform.
- ASP.NET is based on the fundamental architecture of .NET Framework.
- ASP.NET comes with built-in Web Forms controls, which are responsible for generating the user interface.

1.11 Keywords

Code Declaration Blocks: These are lines of code enclosed in <script> tags. They contain the runat=server attribute, which tells ASP.NET that these controls can be accessed on the server and on the client.

Common Language Runtime: It manages the execution of code and provides services that make the development process easier.

Common Type System: It defines how data types are declared, used, and managed in the runtime, and is also an important part of the runtime's support for the Cross Language Integration.

DLL Hell: This refers to the set of problems caused when multiple applications attempt to share a common component like a dynamic link library (DLL) or a Component Object Model (COM) class.

Lab Exercise

1. Search about the CLR architecture.
2. Prepare a flow chart for debugging a program by CLR.

1.12 Review Questions

1. Explain the .NET architecture.
2. What is impersonation in ASP.NET?
3. Explain the concept of Automatic Memory Management in ASP.NET.
4. Describe the cookies collection in ASP.NET.
5. Explain the life cycle of an ASP .NET page.
6. How is .NET able to support multiple languages?
7. How do you validate the controls in an ASP .NET page?
8. Explain how DLL Hell is solved in .NET.
9. How can we communicate with each server in N-tier Architecture?
10. Explain the life cycle of an ASP .NET page.

Notes

Answers for Self Assessment Questions

1. (*a*)	2. (*b*)	3. (*d*)	4. (*c*)	5. (*b*)
6. (*d*)	7. (*d*)	8. (*a*)	9. (*a*)	10. (*c*)
11. (*a*)	12. (*b*)	13. (*a*)	14. (*a*)	15. (*c*)

1.13 Further Readings

Books *Practical ASP. NET 3.5 Projects for Beginners, by B.M. Harwani*
ASP.NET, by Yashwanth Kanethkar

Online link *http://www.w3schools.com/aspnet/default.asp*

Unit 2: Introduction to C#

Notes

CONTENTS

Objectives

After studying this unit, you will be able to:

- Create a simple Web page
- Discuss the Web controls
- Explain the in-line script
- Define the page class

Introduction

C# is the resident language for the .NET Common Language Runtime. It has been designed to fit seamlessly into the .NET Common Language Runtime. You can inscribe code in either Visual C++ or Visual Basic, but in mainly cases, C# will likely fit your needs better. Because the Common Language Runtime is central to many things in C#.

Although C# is unoriginal from the C programming language, it has type such as garbage anthology that allow beginners to become talented in C# more quickly than in C or C++. Similar to Java, it is object-oriented, comes with an extensive class library, and supports exception handling, multiple types of polymorphism, and separation of interfaces from implementations. Those features, combined with its powerful development tools, multi-platform support, and generics, make C# a good choice for many types of software development projects: rapid application development projects, projects implemented by individuals or large or small teams, Internet applications, and projects with strict reliability requirements. Testing frameworks such as NUnit make C# amenable to test-driven development and thus a good language for use with

Notes

Extreme Programming (XP). Its strong typing helps to prevent many programming errors that are common in weakly typed languages.

A large part of the power of C# (as with other .NET languages), comes with the common .NET Framework API, which provides a large set of classes, including ones for encryption, TCP/IP socket programming, and graphics. Developers can thus write part of an application in C# and another part in another .NET language (e.g. VB .NET), keeping the tools, library, and object-oriented development model while only having to learn the new language syntax.

Because of the similarities between C# and the C family of languages, as well as Java, a developer with a background in object-oriented languages like C++ may find C# structure and syntax intuitive.

2.1 A Simple Web Page

To compile your first C# function, you will need of a .NET Framework SDK installed on your PC.

There are two .NET frameworks accessible: Microsoft's and Mono's

Microsoft

For Windows, the .Net Framework SDK can be downloaded from Microsoft's .NET Framework Developer Center. If the default Windows directory (the directory where Windows or WinNT is installed) is C:\WINDOWS, the .Net Framework SDK installation places the Visual C# .NET Compiler (csc) in the C:\WINDOWS\Microsoft.NET\Framework\v1.0.3705 directory for version 1.0, the C:\WINDOWS\Microsoft.NET\Framework\v1.1.4322 directory for version 1.1, or the C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727 directory for version 2.0.

Mono

For Windows, Linux, or other Operating Systems, an installer can be downloaded from the Mono website.

For Linux, a good compiler is cscc which can be downloaded for free from the DotGNU Portable. Net project page. The compiled programs can then be run with ilrun.

If you are working on Windows it is a good idea to add the path to the folders that contain cs.exe or mcs.exe to the Path environment variable so that you do not need to type the full path each time you want to compile.

It is entirely possible to write C#.NET programs with a simple text editor, but it should be noted that this requires you to compile the code yourself. Microsoft offers a wide range of code editing programs under the Visual Studio line that offer syntax highlighting as well as compiling and debugging capabilities. Currently C#.NET can be compiled in Visual Studio 2002 and 2003 (only supports the .NET Framework version 1.0 and 1.1) and Visual Studio 2005 (supports the .NET Framework 2.0 and earlier versions with some tweaking).

The code below will demonstrate a C# program written in a simple text editor. Start by saving the following code to a text file called hello.cs:

```
using System;
namespace MyConsoleApplication
{
    class MyFirstClass
    {
        static void Main (string[] args)
        {
```

Notes

```
            System.Console.WriteLine ( “Hello,”);
            Console.WriteLine ( “World!”);
            Console.ReadLine ();
        }
    }
}
```

To compile hello.cs, run the following from the command line:

- For standard Microsoft installations of .NET 2.0, run C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\csc.exe hello.cs
- For Mono run mcs hello.cs.
- For users of cscc, compile with “cscc-o <name>.exe <name>.cs”.

Doing so will produce hello.exe. The following command will run hello.exe:

- On Windows, use hello.exe.
- On Linux, use mono hello.exe or “ilrun <name>.exe”.

Alternatively, in Visual C# express, you could just hit F5 or the green play button to run the code, even though that is for debugging.

Running hello.exe will produce the following output:

Hello,

World!

The program will then wait for you to strike ‘enter’ before returning to the command prompt.

Note that the example above includes the System namespace via the using keyword. That inclusion allows direct references to any member of the System namespace without specifying its fully qualified name.

The first call to the WriteLine method of the Console class uses a fully qualified reference.

System.Console.WriteLine (“Hello”);

The second call to that method shortens the reference to the Console class by taking advantage of the fact that the System namespace is included (with using System).

Console.WriteLine (“World”);

C# is a fully object-oriented language. The following sections explain the syntax of the C# language as a beginner’s course for programming in the language. Note that much of the power of the language comes from the classes provided with the .Net framework, which are not part of the C# language syntax.

2.1.1 Syntax

C# syntax looks quite related to the syntax of Java because both accede too much of their syntax from C and C++. The object-oriented natural world of C# requires the high level structure of a C# program to be defined in terms of classes, whose detailed behaviours are defined by their statements.

Statements

The basic unit of carrying out in a C# program is the statement. A statement can announce a variable, define an expression, perform a simple action by vocation a process, control the flow

Notes

of execution of other statements, create an object, or assign a value to a variable, property, or field. Statements are usually terminated by a semicolon.

Statements can be grouped into comma-separated statement lists or brace-enclosed statement blocks.

Examples

```
int sampleVariable; // declaring a variable
sampleVariable = 5; // assigning a value
SampleClass sampleObject = new SampleClass(); // constructing a new object
sampleObject.SampleInstanceMethod(); // calling an instance method
SampleClass.SampleStaticMethod(); // calling a static method
// executing a "for" loop with an embedded "if" statement
for(int i = 0; i < upperLimit; i++)
  {
    if (SampleClass.SampleStaticMethodReturningBoolean(i))
    {
      sum += sampleObject.SampleMethodReturningInteger(i);
    }
  }
```

Statement Blocks

A series of statements surrounded by curly braces form a block of code. Among other purposes, code blocks serve to limit the scope of variables defined within them. Code blocks can be nested and often appear as the bodies of methods, the protected statements of a try block, and the code within a corresponding catch block.

```
private void MyMethod ()
{
// This block of code is the body of "MyMethod()"
CallSomeOtherMethod();
try
{
// Here is a code block protected by the "try" statement.
int variableWithLimitedScope;
// "variableWithLimitedScope" is accessible in this code block.
}
Catch (Exception)
    {
    // Here is yet another code block.
    // "variableWithLimitedScope" is not accessible here.
    }
    // "variableWithLimitedScope" is not accessible here, either.
    CallYetAnotherMethod ();
    // Here ends the code block for the body of "MyMethod ()".
}
```

Notes

Comments

Comments allow in order certification of source code. The C# compiler ignores explanation. Three styles of comments are allowed in C#:

Single-line Comments

The "//" character progression marks the following text as a single-line comment. Single-line comments, as one would expect, end at the first end-of-line following the "//" comment marker.

Multiple-line Comments

Comments can span multiple lines by using the multiple-line comment style. Such comments start with "/*" and end with "*/". The text between those multi-line comment markers is the comment.

```
//This style of a comment is restricted to one line.
/*
This is another style of a comment.
It allows multiple lines.
*/
```

XML Documentation-line Comments

This comment is used to generate XML documentation. Each line of the comment begins with "///".

```
/// <summary> documentation here </summary>
```

This is the most recommended type. Avoid using butterfly style comments. For example:

```
//**************************
// Butterfly style documentation comments like this are not recommended.
//**************************
```

Case Sensitivity

C# is case-sensitive, including its variable and method names.

The variables myInteger and MyInteger below are distinct because C# is case-sensitive:

```
int myInteger = 3;
int MyInteger = 5;
```

The following code will generate a compiler error (unless a custom class or variable named console has a method named writeline()):

```
// Compiler error!
Console.writeline("Hello");
```

The following corrected code compiles as expected because it uses the correct case:

```
Console.WriteLine("Hello");
```

2.1.2 Variables

Variables are used to store values. More officially, a variable binds an object (in the general sense of the term, i.e. a specific value) to an identifier (the variable's name) so that the entity can be accessed later. Variables can, for example, store the value of user input:

```
string name = Console.ReadLine ();
```

Notes

Console.WriteLine ("Good morning, {0}", name);

Each variable is declared with an explicit type. Only values whose types are compatible with the variable's declared type can be bound to (stored in) the variable.

C# supports several program elements corresponding to the general programming concept of variable: fields, parameters, and local variables.

Fields

Fields, from time to time called class-level variables, are variables linked with classes or structures. An instance variable is a field connected with an instance of the class or structure, while a static variable, declared with the static keyword, is a field associated with the type itself. Fields can also be associated with their class by making them constants (const), which requires a declaration assignment of a constant value and prevents subsequent changes to the field.

Each field has a visibility of public, protected, internal, and protected internal, or private (from most visible to least visible).

Local Variables

Like fields, local variables can optionally be constant (const). Constant local variables are stored in the assembly data region, while non-constant local variables are stored (or referenced from) the stack. They thus have both a scope and an extent of the method or statement block that declares them.

Parameters

Parameters are variables connected with a method. An in restriction may either have its value passed in from the callee to the method's environment, so that changes to the parameter by the method do not affect the value of the callee's variable, or passed in by reference, so that changes to the variables will affect the value of the callee's variable. Value types (int, double, string) are passed in "by value" while reference types (objects) are passed in "by reference."

An out parameter does not have its value copied, thus changes to the variable's value within the method's environment directly affect the value from the callee's environment. Such a variable is considered by the compiler to be unbound upon method entry, thus it is illegal to reference an out parameter before assigning it a value. It also must be assigned by the method in each valid (non-exceptional) code path through the method in order for the method to compile.

A reference parameter is similar to an out parameter, except that it is bound before the method call and it need not be assigned by the method. A params parameter represents a variable number of parameters.


Caution

If a method signature includes one, the params argument must be the last argument in the signature.

2.1.3 Types

Each type in C# is either a value type or a situation type. C# has several predefined ("built-in") types and allows for statement of custom value types and reference types.

Integral Types

Because the type system in C# is unified with other languages that are CLI-compliant, each integral C# type is actually an alias for a corresponding type in the .NET framework. Although the names of the aliases vary between .NET languages, the underlying types in the .NET framework remain the same. Thus, objects created in assemblies written in other languages of the .NET Framework can be bound to C# variables of any type to which the value can be converted, per

Notes

the conversion rules below. The following illustrates the cross-language compatibility of types by comparing C# code with the equivalent Visual Basic .NET code:

```
// C#
public void UsingCSharpTypeAlias()
{
int i = 42;
}
public void EquivalentCodeWithoutAlias()
{
System.Int32 i = 42;
}
//Visual Basic .NET
Public Sub UsingVisualBasicTypeAlias ()
Dim i as Integer = 42
End Sub
Public Sub EquivalentCodeWithoutAlias()
Dim i As System.Int32 = 42
End Sub
```

Using the language specific type aliases is often considered more readable than using the fully-qualified .NET Framework type names.

The fact that each C# type corresponds to a type in the unified type system gives each value type a consistent size across platforms and compilers. That consistency is an important distinction from other languages such as C, where, e.g. a long is only guaranteed to be at least as large as an Int, and is implemented with different sizes by different compilers. As reference types, variables of types derived from object (i.e. any class) are exempt from the consistent size requirement. That is, the size of reference types like System.IntPtr, as opposed to value types like System.Int, may vary by platform. Fortunately, there is rarely a need to know the actual size of a reference type.

There are two predefined reference types: Object, an stage name for the System.Object class, from which all previous reference types derive; and string, an alias for the System.String class. C# likewise has several integral value types, each an alias to a corresponding value type in the System namespace of the .NET Framework. The predefined C# type aliases expose the methods of the underlying .NET Framework types. For example, since the .NET Framework's System. Int32 type implements a ToString() method to convert the value of an integer to its string representation, C#'s int type exposes that method:

```
int i = 97;
string s = i.ToString();
// The value of s is now the string "97".
```

Likewise, the System.Int32 type implements the Parse() method, which can therefore be accessed via C#'s int type:

```
string s = "97";
int i = int.Parse(s);
// The value of i is now the integer 97.
```

Notes

The unified type system is enhanced by the ability to convert value types to reference types (boxing) and likewise to convert certain reference types to their corresponding value types (unboxing):

```
object boxedInteger = 97;
int unboxedInteger = (int)boxedInteger;
```

The built in C# type aliases and their equivalent .NET Framework types follow:

Integers

C# Alias	.NET Type	Size (bits)	Range
Sbyte	System.SByte	8	–128 to 127
Byte	System.Byte	8	0 to 255
Short	System.Int16	16	–32,768 to 32,767
Ushort	System.UInt16	16	0 to 65,535
Char	System.Char	16	A unicode character of code 0 to 65,535
Int	System.Int32	32	–2,147,483,648 to 2,147,483,647
Uint	System.UInt32	32	0 to 4,294,967,295
Long	System.Int64	64	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
Ulong	System.UInt64	64	0 to 18,446,744,073,709,551,615

Floating-point

C# Alias	.NET Type	Size (bits)	Precision	Range
Float	System.Single	32	7 digits	$1.5 \times 10\text{-}45$ to 3.4×10^{38}
double	System.Double	64	15-16 digits	$5.0 \times 10$to 1.7×10^{308}
decimal	System.Decimal	128	28-29 decimal places	$1.0 \times 10$to 7.9×10^{28}

Other Predefined Types

C# Alias	.NET Type	Size (bits)	Range
bool	System.Boolean	32	True or false, which are not related to any integer in C#.
object	System.Object	32/64	Platform dependant (a pointer to an object).
string	System.String	16 * length	A unicode string with no special upper bound.


Did u know?

The name "C sharp" was inspired by musical notation where a sharp indicates that the written note should be made a semitone higher in pitch. This is similar to the language name of C++, where "++" indicates that a variable should be incremented by 1.


Task Write a program to show the example of boxing and unboxing.


Caution Using unboxing can be cause of decreasing or loss the value of data.

Notes

2.2 Web Control

Web User organize or the .ascx box file in asp .net is a substitution for the comprise file characteristic in ASP. It actually is a positive development from the asp model. Classic asp had used .inc file and included it wherever we needed it. Though this model is also similar, Web User Controls have a lot of other features added along with them.

Web user controls are imitative from System.Web.UI.UserControl namespace. These controls once created, can be additional to the aspx page either at propose point or programmatically during run time. But they lack the design time support of setting the properties created along with the control. They cannot run on their own and they need to stand on another platform like an aspx page. Even if we try to load it on a web browser, IIS will not serve the files of the type .ascx.

2.2.1 Adding a Web User Control

- Create a new ASP .NET web application using Visual Studio .NET.
- Add a Web User Control to the project. Name it as SampleUserControl.ascx.
- Add a Panel Control on the user control form and name it as pnlSample.
- This Panel control is going to be used to hold our other controls.
- Add a TextBox as txtSearch and a Command button as cmdSearch. These will be used for entering a site search term and doing a search.
- Add a property to the User Control as follows.

```
protected Color backColor;
public Color BackColor
{
    get
    {
    return backColor;
    }
    set
    {
    backColor = value;
    }
    }
```

- The above can be used set the Background color of the control. Add the following code to the Page_Load event of the control.
- private void Page_Load(object sender, System.EventArgs e)

```
{
    // Put user code to initialize the page here
    pnlSample.BackColor = backColor;
}
```

- Now add another web form as TestPage.aspx.
- Drag and drop the SampleUserControl.ascx into the Web form. This will put the following entries in the aspx page.

Notes

- <%@ Register TagPrefix= "uc1" TagName= "SampleUserControl" Src= "/Portals/0/SampleUserControl.ascx" %>. This code registers the control with the current aspx page.
- <uc1:SampleUserControl id= "SampleUserControl1" runat= "server"><uc1:SampleUser Control>. This entry will be found inside the <form> .. </form> tag.
- Build the application.
- Select and view the TestPage.aspx in the browser. This will show the aspx page with the control loaded with it.

Now if we need we can change the background color to Gray, by adding the property in the control as follows.

<uc1:SampleUserControl BackColor= "Gray" id= "SampleUserControl1" runat= "server"></uc1:SampleUserControl>

Now if the application is built, the testpage.aspx will show our user control with a gray background.

2.2.2 Advantages of a Web User Control

The biggest advantage of the Web User controls is that they can be shaped as a site pattern and used all through the site. For example they can be complete to contain the Menu/Link arrangement of a site and can be used at all the previous aspx pages.

This means the following:

- If the website introduces a new site-wide link within the current layout/structure, it is enough if we put it on the user control once. All pages will be updated once if the web user control is used in them.
- If there is any link to be corrected, it can be done once at the server side.
- The .ascx files can either be used as a simple alternative to the plain HTML or they can also be used to respond to events: This means even custom code can be created against them and put in the code behind files.

2.2.3 Drawbacks / Disadvantages

Though the User controls offer a flexibility of having site wide modifications, if the whole structure of the site changes, the HTML/aspx code of all the pages should be modified. But as long as the site maintains a same layout, then Web User controls is the number one choice for maintaining the generic layout of the site.

Another disadvantage is It cannot be just be simply referenced for using in a different project. If we want to use this User Control in a different project, we have to copy the file and modify the namespace to the host namespace name.

Self Assessment Questions

Multiple Choice Questions

1. The .Net framework ..... can be downloaded from Microsoft's .NET framework developer centre.
   (*a*) SDK (b) CLR
   (*c*) CRM (*d*) All of these
2. The object-oriented nature of C# .....
   (*a*) middle-level language (*b*) low level language
   (*c*) high level language (*d*) None of these

Notes

3. The fields is .....
   (*a*) high level language (*b*) class-level variables
   (*c*) declares (*d*) class-level defender
4. The ..... are stored in the assembly data region.
   (*a*) stack (*b*) black
   (*c*) namespace (*d*) constant local variables
5. Web user controls are .....
   (*a*) aspx page (*b*) Black
   (*c*) Derived from System (*d*) local area

2.3 Introduction to In-line Script

The ASP.Net web pages sustain both kinds of coding models: code-behind as well as in sequence code. The difference between them is the in sequence code model enables to write the code statements in the .aspx page beside with HTML source code using <script> tag whereas code-behind model separate the .aspx page and server-side code. You can write the code in a disconnect .cs or .vb code file derived from Page class especially for each .aspx page. The other very important difference is that the inline code is deployed along with the .aspx web page when a compiled web application is published online whereas the code-behind approach compiles the code for all web pages into a .dll file that enables to host the web pages free from any inline server code.

Using this syntax, you can use in sequence code in ASP.NET (.aspx) pages. The server-side code will be mechanically compiled by the .NET framework the first point in time the page is requested on the server. The compiled .dll file is stored in the "Temporary ASP.NET Files" system folder. Changing the code in .aspx files will trigger a new compilation, generating new .dll files. The old .dll files are phased out by the framework and eventually deleted.

```
<%@ Import Namespace="System" %>
<%@ Page Language= "c#"%>
<script runat="server">
       public string ServerSideFunction(string input)
       {
            return "Hello" + input;
       }
</script>
<% string pageVariable = "world"; %>
<! DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns= "http://www.w3.org/1999/xhtml" xml:lang= "en" lang= "en">
<head>
<meta http-equiv= "Content-Type" content="text/html;
charset=windows-1252"/>
<title>ASP.NET inline</title>
</head>
```

Notes

```
<body>
<% =ServerSideFunction(pageVariable) %>
</body>
</html>
```

Task Write an inline code for a "Welcome!" page.

2.4 The Page Class

When an ASP.NET sheet is requested and renders score to a browser, ASP.NET creates an instance of a class that represents your page. That class is unruffled not only of the code that you wrote for the page, but also code that is generated by ASP.NET.

All web forms are really instances of the ASP.NET page class, which is definite in the System. Web.UI namespace. The Page class inherits from the TemplateControl class, which in turn inherits from the Control class. As a result, the Page class provides useful properties and methods that we can use in our code.

2.4.1 Objects

Let us now discuss the various objects of the Page class.

Session

The session object is an example of the System.Web.SessionState.HttpSessionState class. It acts as an international repository to store any type of user detailed data that needs to persist between web-page requests. The session object stores data in the form of name/value pairs which is used to maintain data like user's name, user's ID, a shopping cart and other elements which are discarded when a user logs out or is not accessing any page of the web site. Session sate can be both in process and out process.

Application

The Application object is an example of the System.Web.HttpApplicationState class. Similar to session object, it also provisions data in the form of name/value pairs. However, the scope of this data is global to the entire application.

Cache

Cache object is an example of the System.Web.Caching.Cache class. It also stores worldwide information. It is also a name/value compilation of objects, but we can set modified finishing policies and dependencies for each item so that items are automatically removed when other resources, such as files or database tables, are modified.

Request

The request object is an instance of the System.Web.HttpRequest class. It represents the values and properties of the HTTP request that caused our page to be loaded. It contains all the URL parameters and all other information sent by the client.

Notes

HttpRequest Properties

- *ApplicationPath and PhysicalPath:* ApplicationPath gets the ASP.NET application's virtual application root path on server while PhysicalPath gets the Physical file system path corresponding to the requested URL.
- *AnonymousID:* This uniquely identifies the current user if we have enabled anonymous access.
- *Browser:* This provides a link to the requesting client's browser capabilities object.
- *ClientCertificate:* This is an HttpClientCertificate object that gets the security certificate for the current request.
- *Cookies:* This gets the collection cookies sent with this request.
- *FilePath and CurrentExecutionFilePath:* These return the virtual path of the current request.
- *Form:* This represents the collection of form variables that were posted back to the page. In almost all cases we will retrieve the information from control properties instead of using this collection.
- *ServerVariables:* This Returns a collection of named server variables sent with the request.
- *IsAuthenticated:* This returns true if the user has been successfully authenticated.
- *IsSecureConnection:* This indicates whether the HTTP connection uses secure sockets (that is, HTTPS).
- *Islocal:* This returns true if the user is requesting the page from the current system.
- *QueryString:* This provides the parameters that are passed along with the query string.
- *URLReferrer:* This provides a Url object that represents the current address for the page where the user is coming from.
- *UserAgent:* This is a string representing the browser type.
- *UserHostAddress and UserHostName:* These get the IP address and the DNS name of the remote client.
- *UserLanguages:* This provides a stored string array of client's language preference. This can be useful if we need to create multilingual pages.

Response

The Response object is an example of the System.Web.HttpResponse class, and it represents the web server's response to a client request.

httpResponse Members

- *Buffer Output:* When set to true (the default), the page is not sent to the client until the entire page is finished processing.
- *Cache:* It allows us to configure output caching of a Web page.
- *Cookies:* This is the collection of cookies sent with the response.
- *Expires and Expires Absolute:* We can use these properties to cache the rendered HTML for the page for a specified period of time, which helps to improve performance for subsequent requests.

Notes

- *IsClientConnected:* This is a Boolean value which indicates whether the client is still connected to the server.
- *Redirect():* This method transfers the user to another page in the application or to a different web site.
- *ContentType:* The ContentType is a header that tells the browser what type of content it is about to receive. In general, ASP.NET web forms use text or html content type. However, we can create a custom HTTP handler that serves different types of content.
- *OutputStream:* It represents the data we are sending to the browser as a stream of raw bytes.
- *Write():* This method allows us to write text directly to the response stream.
- *BinaryWrite() and WriteFile():* These methods allow us to take binary content from a byte array or form a file and write it directly to the response stream.

Server

The Server object is an example of the System.Web.HttpServerUtility class. It provides dissimilar methods and properties.

MachineName

A property representing the computer name of the computer on which the page is running, that means the server.

CreateObject()

Creates an instance of the COM object that is identified by the object's programmatic identifier (ProgID).

GetLastError

This retrieves the exemption object for the most just encountered error. This is most commonly used in an application event handler that checks for error conditions.


Did u know?

During the development of the .NET Framework, the class libraries were originally written using a managed code compiler system called Simple Managed C (SMC).


Case Study

Turkish Bank Processes Annual Transactions in Just Eight Hours Due to New Infrastructure

Turkey based DenizBank realised the evolution of Web technologies presented both challenges and opportunities. The key challenge was to develop a robust infrastructure that could support all banking services reliably. The organisation required a fully integrated environment delivering customer relationship management, business process management, and a business intelligence layer to maximise efficiency and service quality. DenizBank worked with solution provider Intertech to build the inter. Next infrastructure, which uses Microsoft technologies such as Microsoft SQL Server 2005 data management software and Microsoft BizTalk Server 2006. Today, customer satisfaction rates are high and the system can process a year's worth of transactions in eight hours. After logging on once, personnel can access all applications on the network.

Contd...

Notes

Situation

Over time, it became clear that the UNIX environment could no longer offer the performance and scalability to maintain the bank's market-leading position. A new environment was needed that automated business processes, delivered powerful customer relationship management (CRM) capability with full distribution channel support, and state-of-the-art business intelligence (BI) features. The same environment also required a service-oriented architecture (SOA) to make it easy to integrate with other systems and scale in the future.

Hakan Ateş, President of DenizBank Financial Services Group, says: "We wanted customers to have the same service level across all business channels – be it walking into a branch, using our call centre, or going online. It was also important to provide a single view of the company's performance to executives so we could measure, and manage our business more effectively."

Questions

1. What are the initial situations in DenizBank?

2. Why they use the Web technology in the bank?

Self Assessment Questions

6. Which one is not a type of .NET framework?

   (*a*) Microsoft (*b*) Mono

   (*c*) TK (*d*) None of these.

7. The basic unit of execution in a C# program is the .....

   (*a*) statement (*b*) comma

   (*c*) colon (*d*) None of these.

8. The ASP.NET web pages support both kinds of coding models: one is code-behind and other is:

   (*a*) inline code (*b*) frontend

   (*c*) modern code (*d*) None of these.

9. Web User control or the ..... file in asp .net.

   (*a*) .asx (*b*) .astx

   (*c*) .ascx (*d*) None of these.

10. Which of the following is a type of statement?

    (*a*) Single-line Comments (*b*) Multiple-line Comments

    (*c*) XML Documentation-line Comments

    (*d*) All of these.

11. A class implements two interfaces each containing three methods. The class contains no instance data. Which of the following correctly indicate the size of the object created from this class?

    (*a*) 12 bytes (*b*) 24 bytes

    (*c*) 0 byte (*d*) 8 bytes

Notes

True or False

12. Statements can be grouped into comma-separated statement lists or brace enclosed statement blocks.

    (*a*) True (*b*) False

13. Comments can span multiple lines.

    (*a*) True (*b*) False

14. C# syntax looks quite similar to the syntax of Java.

    (*a*) True (*b*) False

2.5 Summary

- C# is the native language for the .NET Common Language Runtime.
- The object-oriented nature of C# requires the high level structure of a C# program to be defined in terms of classes, whose detailed behaviours are defined by their statements.
- A large part of the power of C# comes with the common .NET Framework API, which provides a large set of classes, including ones for encryption, TCP/IP socket programming, and graphics.
- Web user controls are derived from System.Web.UI.UserControl namespace.
- The Page class inherits from the TemplateControl class, which in turn inherits from the Control class.

2.6 Keywords

Application: it stores data in the form of name/value pairs.

Cache: It is also a name/value collection of objects.

Parameter: Parameters are variables associated with a method.

Request: It represents the values and properties of the HTTP request that caused our page to be loaded.

Response: It represents the web server's response to a client request.

Session: It acts as a global repository to store any type of user specific data that needs to persist between web-page requests.

Statement: The basic unit of execution in a C# program is the statement.

Web User control: It is a replacement for the include file feature in ASP, once created, can be added to the aspx page either at design time or programmatically during run time.

Lab Exercise

1. Write a C# program to show the message?
2. Create a Web page using HTML.

2.7 Review Questions

1. Give a brief description of C#?
2. Write a short note about Web pages in ASP.NET.
3. What is the syntax in C#?

Notes

4. Explain the comments in C#. Give the example also.
5. What are the variables in C#? How many types of variables are defined in C#?
6. What are the types in C#?
7. What are the types of Web controls are defined in the .NET?
8. What are the advantages and disadvantages of Web controls?
9. Differentiate between code behind and inline model.
10. Write a short note on page class.

Answers for Self Assessment Questions

1.	(*a*)	2.	(*c*)	3.	(*b*)	4.	(*d*)	5.	(*c*)
6.	(*c*)	7.	(*a*)	8.	(*a*)	9.	(*c*)	10.	(*d*)
11.	(*b*)	12.	(*a*)	13.	(*d*)	14.	(*a*)		

2.8 Further Readings

Books *Quest - C#.NET Programming,* by Asang Dani

Let Us C# (BPB Publications), by Yashavant Kanetkar

Online link http://www.homeandlearn.co.uk/csharp/csharp.html

Notes

Unit 3: Server Controls Basic

CONTENTS

Objectives

After studying this unit, you will be able to:

- Define postback
- Explain data binding
- Understand web server controls

Introduction

HTML server controls are HTML elements containing attributes to create them noticeable to and programmable on the server. By default, HTML elements on a Web Forms page are not accessible to the server; they are treated as opaque text that is passed through to the browser. However, by converting HTML elements to HTML server controls, we expose them as elements we can program on the server. The object model for HTML server controls maps closely to that of the corresponding elements. For example, HTML attributes are exposed in HTML server controls as properties.

Any HTML element on a page can be converted to an HTML server control. Conversion is a simple process involving just a few attributes. As a minimum, an HTML element is converted to a control by the addition of the attribute RUNAT= "SERVER". This alerts the ASP.NET page framework during parsing that it should create an instance of the control to use during server-side page processing. If we want to reference the control as a member within our code, we should also assign an ID attribute to the control.

Notes

The page framework provides predefined HTML server controls for the HTML elements most commonly used dynamically on a page: forms, the HTML <INPUT> elements (text box, check box, Submit button, and so on), list box (<SELECT>), table, image, and so on. These predefined HTML server controls share the basic properties of the generic control, and in addition, each control typically provides its own set of properties and its own event.

3.1 PostBack

PostBack is the name agreed to the procedure of submitting an ASP.NET page to the server for dispensation. PostBack is done if definite credentials of the page are to be tartan alongside a database (such as verification of username and password). This is something that a customer machine is not able to accomplish and thus these fine points have to be 'posted back' to the server. A simple example to illustrate the usage of PostBack is a login page. After the user has typed his username and password, he clicks on the 'Login' button. Upon the click, the page is sent to the server to check against the database/XML file to check if the user with supplied details is an authenticated user.

Then there arise definite events in an ASP.NET page upon which a PostBack power be needed. Consider the case in which we have 2 Combo Boxes. Let us say the first Combo Box asks us for the Country we reside in and the second one asks us for the State/Province in that country. Based upon the Country we select, the list of States/Provinces must be shown. Thus, in order to fill the values in the second Combo Box, the selection in the first Combo Box must be known to the server. Thus, as and when an item is selected in the first Combo Box, a PostBack must be done and the appropriate list of items must be filled in the second Combo Box.

To handle these situations, we need to enable the 'Auto PostBack' property for those controls whose events are going to trigger some kind of server processing.

PostBack is the name given to the process of submitting an ASP.NET page to the server for processing. PostBack is done if certain credentials of the page are to be checked against some sources (such as verification of username and password using database). This is something that a client machine is not able to accomplish and thus these details have to be 'posted back' to the server.

3.1.1 AutoPostBack Property in ASP.NET

If we make a web Page, which consists of one or more Web Controls that are configured to use AutoPostBack the ASP.Net adds an unusual JavaScript function to the rendered HTML Page. This function is named _doPostBack (). When called, it triggers a PostBack, sending data back to the web Server.

ASP.NET also adds two additional hidden input fields that are used to pass information back to the server. This information consists of ID of the Control that raised the event and any additional information if needed. These fields will empty initially as shown below,

<input type= "hidden" name= "__EVENTTARGET" id= "__EVENTTARGET" value= "" />

<input type= "hidden" name= "__EVENTARGUMENT" id= "__EVENTARGUMENT" value= "" />

The _doPostBack () function has the responsibility for setting these values with the appropriate information about the event and the submitting the form. The _doPostBack () function is shown below:

```
<script language= "text/JavaScript">
Function __doPostBack (event Target, event Argument) {
```

Notes

```
If (! theForm.onsubmit || (theForm.onsubmit ()! = false)) {
theForm.__EVENTTARGET.value = event Target;
theForm.__EVENTARGUMENT.value = event Argument;
theForm.submit ();
}
</script>
```

ASP.NET generates the _doPostBack () function automatically, provided at least one control on the page uses automatic post backs.

Any Control that has its AutoPostBack Property set to true is connected to the _doPostBack () function using the onclick or on change attributes. These attributes indicate what action Browser should take in response to the Client-Side JavaScript events onclick and on change.

In other words, ASP.Net automatically changes a client-side JavaScript event into a server side ASP.Net event, using the _doPostBack () function as an intermediary.

3.1.2 Life Cycle of a Web Page

To work with the ASP.Net Web gearstick events, we need a solid sympathetic of the web page life rotation. The subsequent performance will be taken place when a user changes a control that has the AutoPostBack property set to true:

1. On the client side, the JavaScript _doPostBack function is invoked, and the page is resubmitted to the server.
2. ASP.NET re-creates the Page object using the .aspx file.
3. ASP.NET retrieves state information from the hidden view state field and updates the controls accordingly.
4. The Page. Load event is fired.
5. The appropriate change event is fired for the control. (If more than one control has been changed, the order of change events is undetermined.)
6. The Page.PreRender event fires and the page are rendered (transformed from a set of objects to an HTML page).
7. Finally, the Page. Unload event is fired.
8. The new page is sent to the client.


Did u know?

Microsoft started development on the .NET Framework in the late 1990s originally under the name of Next Generation Windows Services (NGWS). By late 2000 the first beta versions of .NET 1.0 were released.


Caution

Do not set the AutoPostBack property to True. A popup control cannot work properly if it causes postbacks.


Task Create a flow chart for Web page life cycle.

3.2 Data Binding

Notes

ASP.NET 3.5 introduces a new data binding control (Figure 3.1) named the List View. ASP.NET already has a lot of data bind controls; it should be more than 10. But the good news is, List View can literally replace all other data binding controls in ASP.NET. List View control makes data binding easier than previous controls. It has included styling with CSS, flexible pagination, and sorting, inserting, deleting, and updating features.

Figure 3.1: Data binding control

Id	Name		Type	
1	Ashrafur	Rahaman Faisal	Family	Update Delete Cancel
2	Kader Razwan Mishu		Friends	Edit
3	Ajanta Das		Friends	Edit
4	Mahfuzur Rahaman		Family	Edit
5	Raisul Kabir		Business	Edit
	First Name	Last Name	Contact Type	Insert

The List View is a model ambitious control which means that it will not make anything. By default, the developer must completely specify the HTML he/she wants to render in the form of templates. To show data in List View control, we need to take a Layout Template (to define top level of HTML for output rendering). To show data, we need to take Item Template and AlternativeItemTemplate to show alternative row as with different CSS. Here is the example of simple data binding with AlternativeItemTemplate.

```
//very simple databinding in ListView
<LayoutTemplate>
<table border= "0" cellpadding= "1">
<tr style= "background-color:#E5E5FE">
<th align= "left"><asp:LinkButton ID= "lnkId" runat= "server">Id</
asp:LinkButton></th>
<th align="left"><asp:LinkButtonID="lnkName"runat= "server">Name</
asp:LinkButton></th>
<th  align="left"><asp:LinkButton  ID="lnkType"  runat=
"server">Type</asp:LinkButton></th>
<th></th>
</tr>
<tr id= "itemPlaceholder" runat= "server"></tr>
</table>
</LayoutTemplate>
<ItemTemplate>
<tr>
<td><asp:Label  runat=  "server"  ID=  "lblId"><%#Eval("ID")  %></
asp:Label></td>
<td><asp:Label runat= "server" ID= "lblName"><%#Eval("FirstName")+"
  "+Eval("LastName") %></asp:Label></td>
<td><asp:Label  runat=  "server"ID=  "lblType"><%#Eval("ContactTy
pe") %></asp:Label></td>
<td></td>
```

Notes

```
</tr>
</ItemTemplate>
<AlternatingItemTemplate>
<tr style= "background color:#EFEFEF">
<td><asp:Label runat= "server" ID= "lblId"><%#Eval("ID") %></
asp:Label></td>
 <td><asp:Label runat= "server" ID= "lblName"><%#Eval("FirstName")+
""+
  Eval("LastName") %></asp:Label></td>
<td><asp:Label runat="server" ID= "lblType"><%#Eval("ContactTy
pe") %></asp:Label></td>
<td></td>
</tr>
</AlternatingItemTemplate>
```

Here Layout template is making header of the control, and Item Template is showing data taken from table by Binding columns with Label controls, and AlternativeItemTemplate does the same as Item Template just changing CSS for alternative columns.

Data Pager

To add pagination in the list view, we need to add an asp: Data Pager control, better to put this control inside Layout Template at the end of the Layout Template. Data Pager control has many options to show pagination and these are useful too.

```
<asp:DataPager ID= "ItemDataPager" runat= "server" PageSize= "5">
<Fields>
<asp:NumericPagerField ButtonCount= "2" />
 </Fields>
</asp:DataPager>
```

Sorting

It is very simple to sort data in List View. Sort functionality is distinct in the Command disagreement property of the button that contains the columns to be sorted. It occurs when a button with its Command Name property set to Sort is click. And we need to make a call to the Sort method named on sorting="ListView1_Sorting".

```
// to allow sort if click on the header of the table make
//table header controls clickable
//and give commandname and commandargument
<tr style= "background-color:#E5E5FE">
 <th align= "left"><asp:LinkButton ID= "lnkId" runat= "server"
  CommandName= "Sort" CommandArgument= "ID">Id</asp:LinkButton></th>
 <th align= "left"><asp:LinkButton ID= "lnkName" runat= "server"
  CommandName="Sort"  CommandArgument="FirstName">Name</
asp:LinkButton></th>
 <th align= "left"><asp:LinkButton ID= "lnkType" runat= "server"
  CommandName="Sort"CommandArgument="ContactType">Type</
asp:LinkButton></th>
 <th></th>
</tr>
```

Notes

Insert, Update and Delete

To insert data into List View, we need to add a tag in List View named InsertItemTemplate. Add to add inserted code, add code in Item Command.

HTML Code

```
<InsertItemTemplate>
<tr runat= "server">
<td></td>
<td>
<asp:TextBox ID= "txtFname" runat= "server"
  Text='<%#Eval("FirstName") %>' Width= "100px">First Name</
asp:TextBox>
<asp:TextBox ID= "txtLname" runat= "server"
  Text='<%#Eval("LastName") %>'Width= "100px">Last Name</
asp:TextBox>
</td>
<td><asp:TextBox ID= "txtCtype" runat= "server"
  Text='<%#Eval("ContactType")%>'Width="100px">Contact Type</
asp:TextBox></td>
<td><asp:Button ID= "InsertButton" runat= "server"
 CommandName= "Insert" Text= "Insert" /></td>
</tr>
</InsertItemTemplate>
```

CS Code

In the CS file, insert this code in Item Command:

```
if (e.CommandName == "Insert")
{
TextBox txtFname = (TextBox)e.Item.FindControl( "txtFname");
TextBox txtLname = (TextBox)e.Item.FindControl( "txtLname");
TextBox txtCtype = (TextBox)e.Item.FindControl( "txtCtype");
string insertCommand = "Insert into [Contacts]
([FirstName],[LastName],[ContactType]) Values("" + txtFname.Text + "", ""
+ txtLname.Text + "", "" + txtCtype.Text + "");";
 SqlDataSource1.InsertCommand = insertCommand;
}
```

In the same way, code for Update and Delete will be done using EditItemTemlpate. Please check the attached source files to get the complete code.

3.2.1 Properties of Data Binding

The following properties and schemes are used when required the RadTreeView to a data spring:

- *Data Source property*: Set to an instance of our data source. This is compulsory when required the RadTreeView at runtime.
- *DataSourceID property*: Set to the ID of our data source. This is mandatory when binding the RadTreeView declaratively.

Notes

- *Data Member property*: If the data source is a Dataset and Data Member is set, then the RadTreeView is bound to the Data Table with the respective name in the Dataset. If Data Member is not set, the RadTreeView is bound to the first Data Table in the Dataset.
- *DataFieldID property*: This is the field name from the data source used to uniquely identify each row. This field is required when binding to hierarchical data.
- *DataFieldParentID property*: This is the field name from the data source used to identify the row for the parent node. This field is required when binding to hierarchical data.
- *DataTextField property*: This is the field name from the data source that populates each Node's Text property during binding.
- *DataValueField property*: This is the field name from the data source that populates each Node's Value property during binding.
- *DataNavigateUrlField property*: This is the field name from the data source that populates each Node's NavigateUrlField property during binding.
- *Data Bind method*: Call this method after we have set the aforementioned properties when binding at runtime. This method is mandatory for binding at runtime.

Task Write a program for data binding.

Self Assessment Questions

Multiple Choice Questions

1. The ASP.NET is a technology available on .NET platform for developing.....
   (*a*) Trigger
   (*b*) Processing
   (*c*) Dynamic and data driven web applications
   (*d*) Post Back
2. .....will have their own Auto Post Back property
   (*a*) Procedure (*b*) Web controls
   (*c*) controlling (*d*) Net property
3. Microsoft started development on the .NET Framework the first beta versions of .NET
   (*a*) 2000 (*b*) 2003
   (*c*) 1999 (*d*) 1994
4. Dynamic Data provides, provide to the .....for rendering these controls.
   (*a*) Registering (*b*) data layer support
   (*c*) Data model (*d*) None of these
5. The Data-bound control is.....
   (*a*) Object data source or SQL Data Source control
   (*b*) Connect to web
   (*c*) Both (*a*) and (*b*)
   (*d*) None of these

3.3 Web Server Controls

Notes

ASP.NET pages are complete of code, mark-up tags, factual text, and server controls. Based on the demand, the server reins generate the right mark-up language. The ASP.NET runtime combines the output of all controls and serves the client a page to display in a browser. The programming richness of ASP.NET springs from the wide library of server controls that covers the basic tasks of HTML interaction for example, collecting text through input tags—as well as more advanced functionalities such as calendaring, menus, tree views, and grid-based data display.

Key to ASP.NET control programming is the run at characteristic. If a tag in the .aspx basis is declared without the run at quality, it is considered plain text and is output verbatim. Otherwise, the contents of the tag are mapped to a server control and processed during the page life cycle. We identified two main families of server control HTML server controls and Web server controls. HTML controls map to HTML tags and are implemented through server-side classes whose programming interface faithfully represents the standard set of attributes for the corresponding HTML tag. Web controls, in turn, are a more abstract library of controls in which adherence of the proposed API to HTML syntax is much less strict. As a result, Web and HTML controls share a large common subset of functionalities and, in spite of a few exceptions; we could say that Web controls, functionally speaking, are a superset of HTML controls. Web controls also feature a richer development environment with a larger set of methods, properties and events, and they participate more actively in the page life cycle.

As we will see in more detail in the following pages, a second and more thoughtful look at the characteristics of the server controls in ASP.NET reveals the existence of more than just two families of controls. In real world ASP.NET applications, we will end up using controls from at least the following functional categories: HTML controls, core Web controls, validation controls, data-bound controls, user controls, mobile controls, and custom controls. Validation controls are a special subset of Web controls and deserve to be treated in a separate section.

Data-bound controls refer to data binding and therefore to the control's capability of connecting some of its properties to particular data sources. Hence, data-bound controls deserve a section of their own because of the difference in how they are used. User controls are visual aggregates of existing Web and HTML controls that appear as individual, encapsulated, programmable controls to external callers. Mobile controls are used when creating Web applications that target mobile devices. Custom controls refer to server controls we create entirely with code (not visually, as with a user control) that derive from a base control class.

Web server controls are special ASP.NET tags understood by the server. Like HTML server controls, Web server controls are also created on the server and they require a run at= "server" attribute to work. However, Web server controls do not necessarily map to any existing HTML elements and they may represent more complex elements.

The syntax for creating a Web server control is:

```
<asp:control_name id= "some_id" runat= "server" />
```

In the following example we declare a Button server control in an .aspx file. Then we create an event handler for the Click event which changes the text on the button:

```
<script runat= "server">
Sub submit(Source As Object, e As EventArgs)
button1.Text= "You clicked me!"
End Sub
</script>
<html>
```

Notes

```
<body>
<form runat= “server”>
<asp:Button id= “button1” Text= “Click me!”
runat= “server” OnClick= “submit”/>
</form>
</body>
</html>
```

3.3.1 Generalities of ASP.NET Server Controls

All ASP.NET server controls, including HTML and Web controls plus any custom controls we create or download, descend from the Control class. The class is defined in the System.Web. UInamespace; it also is the foundation of all ASP.NET pages.

The Control class is declared as follows:

- Public class Control: IComponent, IDisposable, IParserAccessor,
- IUrlResolutionService, IDataBindingsAccessor,
- IControlBuilderAccessor, IControlDesignerAccessor,
- IExpressionsAccessor

The IComponent interface defines the way in which the control interacts with the other components running in the common language runtime (CLR), whereas IDisposable implements the common pattern for releasing managed objects deterministically.

3.3.2 Identifying a Server Control

The client ID of a control is generated from the value of the UniqueID property the truly server-side identifier that ASP.NET generates for each control. The contents of the ClientID property differ from UniqueID simply in that all occurrences of the dollar symbol ($), if any, are replaced with the underscore (_). Dollar symbols in the UniqueID string are possible only If the control belongs to a naming container different from the page.

ASP.NET generates the value for the UniqueID property based on the value of the ID property that the programmer indicates. If no ID has been specified, ASP.NET auto-generates a name such as _ctlX, where X is a progressive 0-based index. If the control’s naming container is the host page, UniqueID simply takes the value of ID. Otherwise, the value of ID is prefixed with the string representing the naming container and the result is assigned to UniqueID.

3.3.3 Naming Containers

A naming container is chiefly a control that acts as a container for other controls. In doing so, the naming container generates a sort of virtual namespace so that ASP.NET roots the definite ID of contained controls in the ID of the naming container. To fully understand the role and importance of naming containers, consider the following example. Imagine we have a composite control, such as a user control, that includes a child control like a button. Entirely wrapped by the user control, the button is not directly accessible by the page code and cannot be given a distinct and per-instance ID. In the end, the ID of the button is hard-coded in the outermost control that creates it. What happens when two or more instances of the composite control are placed on a page? Are we going to have two button child controls with the same ID? This is exactly what will happen unless we configure the composite control to be a naming container.

The importance of naming containers does not end here. Imagine we have an instance of a composite control named Control1. Imagine also that the embedded button is named Trigger.

Notes

The full name of the child button will be Control1$Trigger. Suppose we click on the button and cause the page to post back. If the name of the posting control contains the $ symbol, the ASP.NET runtime recognizes a known pattern: tokenize the name and locate the postback control correctly, no matter its depth in the page tree.


Caution

If the button is contained in a control not marked to be a naming container, the ID of the clicked button is not prefixed and will simply be, say, Trigger. In this case, the ASP.NET runtime will look for it as a direct child of the form. The search will obviously fail the button is a child of a top-level control and the postback event will pass unnoticed.

3.3.4 Themeable Controls

In the ASP.NET terminology, a theme is a named compilation of possessions settings that can be applied to wheel to make them look steady across pages. We can apply theme setting to an entire Web site, to a page and its controls, or to an individual control. A theme is identified by name and consists of cascading style sheet (CSS) files, images, and control skins. A control skin is a text file that contains predefined values for some control properties. Applied together, these settings contribute to change the look and feel of the control and give the whole site a consistent (and, we hope, appealing) user interface. In addition, because themes are a sort of monolithic attribute, we can easily export that look from one application to the next. With themes enabled, if the developer adds, say, a Data Grid control to a page, the control is rendered with the default appearance defined in the currently selected theme. Server controls can dynamically accept or deny theming through a Boolean property named Enable Heming, set to true by default. As a general rule, themes affect only properties that relate to the control's appearance. Properties that explicitly specify behaviour or imply an action should not be made themeable. Each control has the power to state which properties are themeable and which are not. This happens at compile time through attributes – in particular, the Themeable attribute.

3.3.5 Control State

Some ASP.NET controls need that several states are kept crossways needs. Examples of this type of state information include the present page of a paged control and the present sort order of a sort able data manage. In ASP.NET 1.x, there is only one storage place in which this data can be stored the view state. However, the view state is mainly designed to maintain settings set by the application and, more importantly, it can be turned off. What would happen to control specific state in this case? For this reason, starting with ASP.NET 2.0 Microsoft introduced the notion of the "control state" and managed to keep it separate from the view state. So it is clear that control state is a vital piece of the control infrastructure.

Control state is a collection of critical view state data that controls need to function. Because of its critical role, control state data is contained in separate member variables from normal view state and is not affected when view state is disabled. Unlike view state, control state requires extra implementation steps to use.

For one thing, each control needs to signal to the page that it requires control state. Next, there is no unique container to store data, such as View State; but the data can be retrieved from any object we want – arrays, collections, or a slew of instance variables. Each control persists and loads its control state using a pair of overridable methods, as shown here:

Protected override object SaveControlState ()

Protected override void LoadControlState (object state)

Control state works similarly to view state and is saved and loaded at the same stage of the Pipeline that view state is processed. Ultimately, control state is persisted in the same hidden field as the view state.

Notes


Caution

There are potential security risks when using and developing custom ASP. NET controls.


Case Study

Bringing WinForms Controls to the Web with AJAX

Established in 2000, Atalasoft is a provider of a high-performance, standards-compliant .NET Framework imaging libraries targeted towards authors of document processing and management systems and photographic imaging. Atalasoft offers unparalleled ease of integration through CLR compliant objects, logically laid out object hierarchies, rock solid implementation, and first rate support.

In 2003, Atalasoft decided to make web versions of their WinForms imaging controls. The main issue in the port of the controls to the web was how to get the same interactivity in a web control that you could get in WinForm control. Here is a short list of the major requirements:

Key Web Image Viewer Requirements

- Cross-platform and cross-browser
- Interactivity similar to what you get with the WinForms controls:
  - Scroll bars when a large image was shown in a small viewport
  - Ability to load parts of the image on demand
  - Ability to pan, zoom, and select with mouse gestures
- Tight integration with HTML, CSS and JavaScript on the client and ASP.NET on the server
- No special security settings required
- No plug-ins or anything to install or approve
- Technology could be used for future controls
  - The Web Thumbnail Viewer (released in 2006) required lazy loaded thumbnails and full integration with the Image Viewer
  - The Web Annotation Viewer (released in 2007) required the ability to create, edit, move and resize annotations in the browser

For this project, we chose to use the techniques that would later be called AJAX. The term was coined in 2005 to describe the general technique of using JavaScript to contact a server asynchronously whenever the user needs a server update and using the result of that communication to change the DOM by generating HTML dynamically with JavaScript. The most famous example is Google Maps, but the browser features that enable what we call AJAX were introduced into Internet Explorer in 2000 to implement Outlook Web Access. By the time Atalasoft started, these features were implemented in all major browsers.

<embedsrc="http://www.youtube.com/v/ZCit_ZEUEjc"width="425"height="350" type="application/x-shockwave-flash" wmode= "transparent" /></embed />

AJAX is just one way to get interactivity in a browser. The general term for applications like this at the time were Rich Internet Applications (RIA) and that term is still used to encompass the wide range of technologies that can implement a desktop-like feel in a web

Contd...

Notes

browser. There are several technologies that could be used to implement our feature set: Java Applets, Flash, ActiveX or AJAX.

Java Applets are programs written in Java that run in the sandbox inside the browser. They have been around since 1995, and are a well understood technology. One thing to note is despite their long history, they are still relatively rare on the public web. The following drawbacks:

1. The versions of the Java runtime available in browsers is inconsistent.
2. Not every browser comes with Java by default, and Java is not ubiquitous enough for many applications.
3. Java applets do not integrate well with other browser technologies such as CSS and JavaScript.

Despite that, Java applets are still a viable option for many applications. They are more common in enterprise intranet applications where the desktop machines are more regular and controlled. They are cross-platform and cross browser, they can have high interactivity, and if you program within the confines of the sandbox, there are no special security settings.

The only other problems, from our point of view, are that they do not integrate well into ASP.NET applications. The main reason is that they do not automatically keep and restore state through a post-back. We recommend do not use a post-back to update our controls (we provide better ways), but these controls live on a page with other controls that might need to post-back, so we need to support it. The big advantage we get from this now is full integration with the Microsoft ASP.NET AJAX (formerly Atlas) UpdatePanel, which is based on post-backs.

Second, there is no standard way for Java applets to be controlled from the server-side. Since we want to be able to have an ASP.NET control on the server that behaves like any other server control (which will integrate tightly with Visual Studio's designer), we would have to write this part ourselves.

Flash programs run inside the browser via a plug-in published by Adobe. The player is ubiquitous (an independent assessment says that 97-98% of internet users have it in their browser), and it is possible to create stunning User Interfaces with it. Although it is common to see Flash used for online advertising and games, it is still not widely used to create applications. Flash does have the best cross-browser/cross-platform multimedia support which is one reason why YouTube and Google Video chose it for their respective video services.

Similarly to Java applets, they do not have standard ways to integrate with CSS and JavaScript (although recent Flash players expose some controls to JavaScript), and they would not behave well through a post-back.

The biggest drawback at the time is the programming model for Flash which is very different from the standard ways of writing software. The player evolved from an interactive presentation creation system, and still has many artefacts from that. Essentially, creating a Flash application was modelled after stringing together movie clips or animations, and not based on any developer-friendly GUI methodology.

There are two alternatives for that now. Adobe themselves have released Flex, which among other things adds a mark-up language to describing Flash user interfaces, and OpenLaszlo, which is an open-source alternative that also provides a more standard programming model to delivering programs to the Flash player. At the time we were creating our controls, both of these technologies were in their infancy (and Laszlo was not open until October 2004).

Contd...

Notes

ActiveX and other plug-in technologies like .NET's Web Deploy and even the new XAML Browser Applications have to be considered by any company targeting the Microsoft platform. If you only need to support IE on Windows then they offer superior interactivity and a well-supported programming environment. However, Atalasoft was committed to our client-side working on every major browser and platform without any plug-ins or special security settings, so these were never really an option for us.

AJAX is a term used to describe applications that use JavaScript, Dynamic HTML, and some sort of mechanism to contact the server for updates without needing a post-back. The effect is that the HTML page appears to update in place without causing it to blank out and refresh. AJAX itself is an acronym meaning Asynchronous JavaScript and XML, and describes the technique, not a specific tool or implementation – there is a AJAX player, for instance.

Although the name "AJAX" caught on very quickly, it does not really describe how this technique is actually done, since XML is not always used. The X in AJAX refers to both the format of the server communication (XML) and the browser function used to initiate the connection (XMLHttpRequest). Atalasoft actually uses neither, opting instead for communicating by posting forms in the background via an IFRAME and receiving JavaScript and JSON (JavaScript Object Notation).

- To us, AJAX offers the following benefits over the other technologies:
- Only needs features native to the browser, HTML and JavaScript. All modern browsers support it. No plug-ins required.
- If necessary, you can write it to degrade nicely on older browsers that do not support it.
- It can be styled with CSS and integrates naturally with JavaScript.
- Since it is built on top of HTML and uses the DOM to represent itself, ASP.NET has natural ways for keeping its state via a post-back.
- It uses open technologies well understood by web developers.

Three years later, the technique we chose is gaining great acceptance. In 2005, after Google released Google Maps and Suggest (and GMail starting getting more and more interactive), many developers realized that HTML and JavaScript had come a long way, and were willing to bet on the technology (since it was good enough for Google). Gartner is predicting that by 2010, at least 60% of new application development projects will include Rich Internet Applications (RIA) technology such as AJAX.

The second annual AJAXWorld in NYC was in March this year, and according to the keynote, there are over a hundred AJAX frameworks on the market (many open-source). Google, Yahoo, and Microsoft all have open frameworks and tools available, but there are other popular ones such as Dojo and MochiKit, not to mention many proprietary ones.

<embedsrc="http://www.youtube.com/v/_s1gtVwMMAg"width="425"height="350"type= "application/x-shockwave-flash" wmode= "transparent" /></embed />

For ASP.NET developers, Microsoft publishes ASP.NET AJAX (formerly Atlas), which is a combination of their open-source AJAX JavaScript library and server-side ASP.NET controls to interact with them. Our AJAX implementation is fully compatible with it (since we designed it to be compatible with ASP.NET from the start). One common theme at AJAXWorld is that it was better to adopt AJAX components that operated within a framework. This is because although AJAX is thought of as a client-side technology, its true benefit comes from tight

Contd...

Notes

integration with a server-side proxy for the component. For web components targeted ASP. NET, integrating with the ASP.NET control classes is key to delivering the full benefit of AJAX.

Questions

1. Why it needed to bring WinForms controls to the Web with AJAX?
2. What was the problem of Atalasoft? How did it solved?

Self Assessment Questions

6. PostBack is the name given to the process of submitting …… page to the server for processing.

   (*a*) JSP.NET (*b*) ASP.NET

   (*c*) PHP (*d*) Both a and c

7. …… introduces a new data binding control named the List View. ASP.NET already has a lot of data bind controls; it should be more than 10.

   (*a*) ASP.NET 1.0 (*b*) ASP.NET 2.0

   (*c*) ASP.NET 3.5 (*d*) ASP.NET 4.0

8. CSS stands for….

   (*a*) cascading style sheets (*b*) cascading system sheets

   (*c*) casecade style sheets (*d*) Both b and c

9. HTML stands for…..

   (*a*) Hyper Text Mark-up Language (*b*) Hyper Text Makeup Language

   (*c*) Hyper Text Machine Language (*d*) Both a and b

10. The first Component interface defines the way in which the control interacts with the other components running in the CLR.

    (*a*) CTS (*b*) CLR

    (*c*) SLR (*d*) Both b and c

True or False

11. Data binding is a general technique that binds two data/information sources together and maintains synchronization of data.

    (*a*) True (*b*) False

12. The naming container generates a sort of virtual namespace so that ASP.NET roots the actual ID of contained controls in the ID of the naming container.

    (*a*) True (*b*) False

13. ASP.NET do not retrieves state information from the hidden view state field and updates the controls accordingly.

    (*a*) True (*b*) False

14. Data-bound controls refer to data binding and therefore to the control's capability of connecting some of its properties to particular data sources.

    (*a*) True (*b*) False

Notes

15. CLR stands for common language runtime

(*a*) True (*b*) False

3.4 Summary

- HTML server controls are HTML elements containing attributes that make them visible to and programmable on the server.
- PostBack is the name given to the process of submitting an ASP.NET page to the server for processing. Post Back is done if certain credentials of the page are to be checked against a database.
- ASP.NET pages are made of code, mark-up tags, literal text, and server controls. Based on the request, the server controls generate the right mark-up language. The ASP.NET runtime combines the output of all controls and serves the client a page to display in a browser.
- A naming container is primarily a control that acts as a container for other controls. In doing so, the naming container generates a sort of virtual namespace so that ASP.NET roots the actual ID of contained controls in the ID of the naming container.
- Control state is a collection of critical view state data that controls need to function. Because of its critical role, control state data is contained in separate member variables from normal view state and is not affected when view state is disabled.

3.5 Keywords

ASP.NET: It is a Web application framework developed and marketed by Microsoft to allow programmers to build dynamic Web sites, Web applications and Web services.

Data binding: It is a general technique that binds two data/information sources together and maintains synchronization of data.

Hyper Text Mark-up Language (HTML): It is the main mark-up language for displaying web pages and other information that can be displayed in a web browser.

Namespace: A namespace (sometimes also called a name scope) is an abstract container or environment created to hold a logical grouping of unique identifiers or symbols (i.e., names).

PostBack: It is the name given to the process of submitting an ASP.NET page to the server for processing.

Lab Exercise

1. Write a program to show the use of post back property.
2. Write a program to create a home page.

3.6 Review Questions

1. Define the postback property.
2. What are the advantages of the postback property?
3. What is data binding? Explain with example?
4. What do you understand by web server controls?
5. Discuss the life cycle of a web page.
6. Define Autopostback property in Asp.Net?

Notes

7. What is the difference between the postback and autopostback property?
8. Define the properties of data binding.
9. What are the naming containers?
10. What are the themeble controls?

Answers for Self Assessment Questions

1.	(*c*)	2.	(*b*)	3.	(*a*)	4.	(*b*)	5.	(*a*)
6.	(*b*)	7.	(*c*)	8.	(*a*)	9.	(*a*)	10.	(*b*)
11.	(*a*)	12.	(*a*)	13.	(*b*)	14.	(*a*)	15.	(*a*)

3.7 Further Reading

Books *Pro ASP.Net 3.5 Server Controls and Ajax Components* by Rob Cameron, Dale Michalk

Online link http://books.google.co.in/books?id=5RfcxwmvRXwC&pg=PA1&dq=Server+Controls+Basic&hl=en&sa=X&ei=oJ_6T63NO4L3rQfu7uDABg&sqi=2&ved=0CGAQ6AEwBg#v=onepage&q=Server%20Controls%20Basic&f=false

Notes

Unit 4: Advanced Server Controls

CONTENTS

Objectives

After studying this unit, you will be able to:

- Define HTML server controls
- Explain validation controls

Introduction

Server joystick is an integral aspect of each ASP.NET function we construct. They encapsulate browser appearance and server functionality in a reusable entity. They can be used across multiple pages within a single ASP.NET application as well as across multiple ASP.NET applications. ASP.NET comes with a lot of prebuilt server controls. We have simple controls such as the label and we have complex controls such as the Grid View. We also have the ability to create our own server controls to meet a need not met by one of the existing controls by inheriting from the appropriate base class and overriding its methods as needed.

This model of using server controls to encapsulate browser appearance and server functionality has served our needs well since the inception of ASP.NET 1.0, but our server control needs are changing. A new server control need that has recently surfaced is the ability to incorporate Ajax functionality directly into the server control.

This requires arose because our web applications need to be more receptive and visually interactive than the conventional ASP.NET repaint-the-entire-screen model and therefore the conventional server control supplies. This requirement has emerged because users are using web sites such as Gmail, Live.com, Yahoo! Mail, and others that do not repaint the screen every time they click a button or need to receive fresh data. Rather, they rely on Ajax to fetch fresh data and then update or add to a portion of the screen based upon that data. Because these web sites are heavily used and users really enjoy their experience while using these websites they

Notes

expect other web sites to perform with the same elegance as they do. When a web site does not perform with the same elegance the user will often move onto another web site that does. Those popular applications have raised the bar for what is an acceptably user-friendly web site.

Because our users are demanding a web site experience that essentially uses Ajax and we build our ASP.NET web sites using server controls, we need a way of easily creating server controls that not only encapsulate browser appearance and server functionality, but also include Ajax functionality so that the server control itself is Ajax-enabled.

Taking a step back for a moment, unlike other technologies we might have read books on, ASP. NET AJAX server controls do not provide we with anything that we could not already do. We have always been able to embed Ajax-functionality into server controls it was just a real pain. There were a few different methods we could use to include the JavaScript with our server control such as embedding it as a resource, but we eventually ended up having to do the same three tasks. To make our server control have some serious client capabilities we always had to concatenate strings together to form JavaScript statements and functions, write browser sniffing statements to make sure that the JavaScript was cross-browser compatible, and add attributes or render out Html that attached the JavaScript functionality to the client versions of our server controls. It was not impossible, but it was error-prone and there was always this mingling of server code and JavaScript that was hard to maintain and even harder to read.

Furthermore, if we had multiple server controls that had client capabilities it was difficult (but not impossible) to ensure that the client functions that each server control required did not overwrite each other when rendered on the browser. Tracking down that problem was always a fun hour or so. The difficulty grew exponentially if we wanted to include a mechanism for asynchronously communicating with the server when the user pressed a button embedded in the server control. Even with a helper communication library there were always tricks to getting our control to communicate properly with the server. These hindrances were problematic enough to lead to some bad programming habits and bad code as well as scare programmers away from even attempting to include Ajax-functionality in their server controls. These problems are what Microsoft's ASP.NET AJAX solves.

4.1 HTML Server Controls

The HTML server controls are basically the inventive HTML controls but improved to enable server side processing. The HTML controls like the subtitle tags, anchor tags and contribution elements are not processed by the server but sent to the browser for show.

They are specifically converted to a server control by adding the attribute runat= "server" and adding an id attribute to make them available for server-side processing.

For example, consider the HTML input control:

<input type= "text" size= "40">

It could be converted to a server control, by adding the runat and id attribute:

<input type= "text" id= "testtext" size= "40" runat= "server">

4.1.1 Advantages of Using HTML Server Controls

Though ASP.Net server controls can execute every job accomplished by the HTML server controls, the later controls are useful in the following cases:

- Using static tables for layout purposes.
- Converting a HTML page to run under ASP.NET.

Notes

The following Table 4.1 describes the HTML server controls:

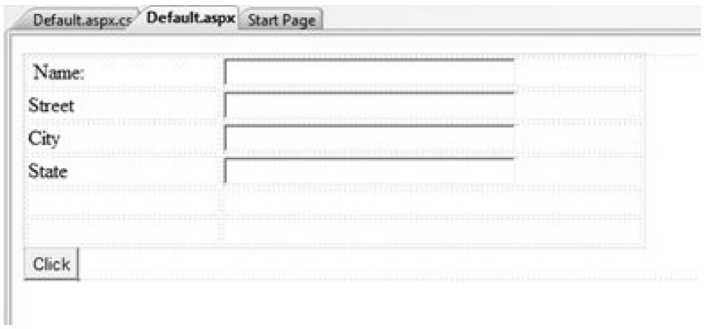
Table 4.1: The HTML server controls

Control Name	HTML tag
HtmlHead	<head>element
HtmlInputButton	<input type=button|submit|reset>
HtmlInputCheckbox	<input type=checkbox>
HtmlInputFile	<input type = file>
HtmlInputHidden	<input type = hidden>
HtmlInputImage	<input type = image>
HtmlInputPassword	<input type = password>
HtmlInputRadioButton	<input type = radio>
HtmlInputReset	<input type = reset>
HtmlText	<input type = text|password>
HtmlImage	<img> element
HtmlLink	<link> element
HtmlAnchor	<a> element
HtmlButton	<button> element
HtmlButton	<button> element
HtmlForm	<form> element
HtmlTable	<table> element
HtmlTableCell	<td> and <th>
HtmlTableRow	<tr> element
HtmlTitle	<title> element
HtmlSelect	<select> element
HtmlGenericControl	All HTML controls not listed

Example: The following instance uses a basic HTML table for design. It uses some text boxes for getting input from the users like, name, address, city, state etc. It also has a button control, which is clicked to get the user data displayed on the last row of the table.

The page should look like this in the design view (See Figure 4.1):

Figure 4.1: Design view for Web page



The code for the content page shows the use of the HTML table element for layout.

Notes

```
<%@ Page Language= "C#" AutoEventWireup= "true"
CodeBehind= "Default.aspx.cs"
Inherits= "htmlserver._Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns= "http://www.w3.org/1999/xhtml" >
<head runat= "server">
<title>Untitled Page</title>
<style type= "text/css">
style1
{
Width: 156px;
}
style2
{
Width: 332px;
}
</style>
</head>
<body>
<form id= "form1" runat= "server">
<div>
<table style= "width: 54%;">
<tr>
<td class= "style1">Name :< /td>
<td class= "style2">
<asp:TextBox ID= "txtname" runat="server" Width= "230px">
</asp:TextBox>
</td>
</tr>
<tr>
<td class= "style1">Street</td>
<td class= "style2">
<asp:TextBox ID= "txtstreet" runat= "server" Width= "230px">
</asp:TextBox>
</td>
</tr>
<tr>
<td class= "style1">City</td>
<td class= "style2">
```

Notes

```
<asp:TextBox ID= "txtcity" runat= "server" Width= "230px">
</asp:TextBox>
</td>
</tr>
<tr>
<td class= "style1">State</td>
<td class= "style2">
<asp: TextBox ID="txtstate" runat="server" Width= "230px">
</asp: TextBox>
</td>
</tr>
<tr>
<td class= "style1"></td>
<td class= "style2"></td>
</tr>
<tr>
<td class= "style1"></td>
<td ID= "displayrow" runat= "server" class= "style2">
</td>
</tr>
</table>
</div>
<asp: Button ID= "Button1" runat= "server"
Onclick= "Button1_Click" Text= "Click"/>
</form>
</body>
</html>
```

The code behind the button control:

```
protected void Button1_Click(object sender, EventArgs e)
{
string str = "";
str += txtname.Text + "<br/>";
str += txtstreet.Text + "<br/>";
str += txtcity.Text + "<br/>";
str += txtstate.Text + "<br/>";
displayrow.InnerHtml = str;
}
```


Caution

Text is not HTML encoded before it is displayed in the LinkButton control. This makes it possible to embed script within HTML tags in the text. If the values for the control come from user input, be sure to validate the values to help prevent security vulnerabilities.

Notes

4.1.2 Difference between Web Server Control and Html Server Control

Web Server Controls are collection of controls imitative straight from the arrangement.Web. UI.WebControls base class. They are reusable components that can execute function as the ordinary HTML controls; the real benefit of web controls is that they are programmable i.e. they can treated and accessed the same way as any other .NET classes. They therefore respond to events, have methods /functions and can basically do all that other .NET classes can do. The main issue to note is that the processing of these controls is done on the server. Consider:

<asp:Button id= “MyButton” runat= “server” Text= “This is Zimcoder’s button”/>

The key attribute is the “runat” which is set to the value “server”, this is their default value Web Server Controls are rendered as standard html to client browsers thus abreacting the functionality. They also make it easy to deal with complex controls such as the Calendar control which would be hell to implement in html!

HTML server controls map directly to html tags. They are defined in System.Web. UI.HtmlControlsnamespace.The base class is System.Web.UI.HtmlControls.HtmlControl. Html tags are converted to Html server controls by including the runat attribute in their declaration and setting it to server. Consider

<input type= “button” id= “mybutton” name= “mybutton” value= “click me” runat= “server”>

Although these two categories may have overlapping functionality and may be even used synonymously they have important differences. Such as

1. HTML s controls offer one to one mapping with html tags no abstraction; Web s controls do not necessarily map to any html tag e.g. calendar control.
2. HTML attributes are not stronly typed to html s controls. Web s controls have strongly typed attributes making for easier access to methods and properties of the base class.
3. HTML s controls do not distinguish between browsers though are not always rendered in a predictable way across browsers. Web s controls were designed with this in mind.
4. Web server controls can be extended as the developer sees fit to come up with custom controls not so easily or intuitively doable with htmls controls.

4.1.3 HTML Server Controls Advantage

1. ASP .NET Server Controls can detect the target browser’s capabilities and render themselves accordingly. No issues for compatibility issues of Browsers.
2. Processing would be done at the server side.
3. ASP .NET Server Controls have an object model different from the traditional HTML and even provide a set of properties and methods that can change the outlook and behaviour of the controls.
4. ASP .NET Server Controls has higher level of abstraction. An output of an ASP .NET server control can be the result of many HTML tags that combine together to produce that control and its events. Example Grandview or Form control.
5. The HTML Server Controls follow the HTML-centric object model. Model similar to HTML.
6. Here the controls can be made to interact with Client side scripting. Processing would be done at client as well as server depending on our code.

4.1.4 Html Server Controls Disadvantage

1. You would need to code for the browser compatibility.

Notes

2. The HTML Server Controls have no mechanism of identifying the capabilities of the client browser accessing the current page.


Did u know?

Version 3.0 of the .NET Framework is included with Windows Server 2008 and Windows Vista. Version 3.5 is included with Windows 7, and can also be installed on Windows XP and the Windows Server 2003 family of operating systems On 12 April 2010, .NET Framework 4 was released alongside Visual Studio 2010.


Task Create a Web page using HTML controls.

Self Assessment Questions

Multiple Choice Questions

1. .....to fetch fresh data and then update or add to a portion of the screen based upon that data.

   (*a*) Ajax (*b*) JavaScript

   (*c*) Regular Expression (*d*) HTML

2. Web server controls are group of controls derived directly from the.....

   (*a*) Registering (*b*) XML

   (*c*) System (*d*) None of these

3. Textbox control requires the user to enter the password again to ensure they did not.....

   (*a*) registering (*b*) mistype the original

   (*c*) data model (*d*) validation control

4. The string data type with the range..... to make sure that the value entered falls within a specific range of characters.

   (*a*) Required Field Validator (*b*) data layer support

   (*c*) Web Server Controls (*d*) Validator server control

True or False

5. ASP.NET has changed this by giving us the capability to use the validation server controls that are provided with the other new controls at our disposal.

   (*a*) True (*b*) False

4.2 Validation Controls

One of the generally universal elements of Web pages is a form in which the users can payment data that is posted back to the server. These forms are made up of different types of HTML elements that are constructed using straight HTML, HTML server controls, or Web server controls. A variety of HTML elements, such as text boxes, check boxes, radio buttons, drop-down lists and more can be used in forms.

Often when our ASP.NET applications are collecting information from a user, we want to ensure that the data that we collect is valid. Some users are not interested in spending enough time to enter the correct information into a form, and in some cases, users might even intentionally enter false information to gain access or get past a certain step in our application's workflow

Notes

process. One of the first steps is to understand what validating data means. Validating, in this case, does not mean that if John Doe types his name into the form field of a text box as Fred Doe the computer sends an alert to inform us that the data is untruthful. No, we still do not have the capability to find out whether a statement is true.

Validation is testing to determine whether the user entered something into the field. After we determine that something was entered, we can then also check to see whether what was entered is a number or a character and if it is in the correct format. From there, we can compare user input in different fields or against values that might be held in other repositories, such as a database. Data collection on the Internet is one of its most important features, so we must make sure that the data we collect has value and meaning. We ensure this by eliminating any chance that the information collected does not abide by the rules we outline.


Caution

When creating a client-side validation function, be sure to also include the functionality of the server-side validation function. If you create a client-side validation function without a corresponding server-side function, it is possible for malicious code to bypass validation.

4.2.1 Understanding the Difference between Server-Side and Client-Side Validation

Many persons new to ASP.NET do not know the dissimilarity between client-side and server-side justification. We must understand these two different ways of validating the data user's contribution into a Web form. After the user enters data into a Web form, clicks the Submit button, and sends the form data to the server as a request, we can perform server-side validation on the data. If the data is incorrect or not valid, we can send back a response stating this. If, however, when the user clicks the Submit button, a scripting language that is part of the overall HTML page is initiated to check the validity of the data before it is sent to the server, this is client-side validation.

It was a lot easier to appreciate the difference between these forms of corroboration when we coded Active Server Pages 3.5 because, as the programmer, we in person performed approximately all data validation. We our self either programmed it to be client-side or server-side.

When we used server-side validation with ASP 3.5, if something the user entered was wrong, we could repost the form and ask the user to correct the information in that particular field of the form. Sometimes, we carried the correct input from the other fields back to the form page, and populated the fields for the users so they did not have to re-enter the same information again. Some sites on the Internet do not carry this inputted information back to the form page, and the user is then required to enter all the information into the form a second time. Obviously, this may cause people to leave our site for another. The bad thing about server-side validation is that it requires trips back and forth to the server. This takes a lot of resources and makes for a slower-paced form for the user. Nothing is more annoying to a user who is on a dial-up connection than clicking the Submit button on the form and then waiting for 20 seconds to find out that they did not enter their password correctly.

The other option for form validation is to put some client-side JavaScript or VBScript at the top of the ASP page that checks if the information in the fields is correct. This takes care of the problem of making unnecessary trips to the server, but it requires another language to learn and manage. JavaScript is a great language, but takes a lot of time to master, and there are always problems getting our JavaScript code to work on different browsers. Example shows us an example of using client-side JavaScript to perform form validation.


Example:

```
Client-Side JavaScript for form Validation
<script language= "javascript">
<!--
```

Notes

```
Function CheckForm(form)
{
for(var intCtr = 0; intCtr <= (form.elements.length - 5);
++intCtr)
{
var temp = form.elements[intCtr];
if(temp.type == "text" && temp.value == "")
{
alert( "Please Enter All Information!");
temp.focus();
return false;
}
}
return true;
}
//-->
</script>
```

This sample piece of JavaScript does some validation, but it does not check for all the information that we might need on the form we are building. This piece of code determines only whether the user entered anything at all in all five fields within the form. It does not determine whether the user entered an actual e-mail address within the e-mail address text box, whether the user entered a number between two given numbers, or whether the password and the confirm password text boxes match. After awhile, we can see that we need many JavaScript functions to obtain the level of form validation required.

4.2.2 .NET to the Rescue

Developers who used classic Active Server Pages 3.5 to develop Web pages might keep in mind that they used to use up a considerable quantity of their time initial validation workings in their page. It was time consuming if we did it right. It was most efficient to do the validation of the form on the client-side to limit the number of requests and responses required to work through an application. However, we were never quite sure if the requesting browser would understand the scripting code that we used for the validation. So, it was usually better, especially for critical Web applications, to bring the validation to the server.

ASP.NET has distorted this by giving us the potential to use the corroboration server controls that are provided with the other new controls at our disposal. What makes these validation server controls effective is that when an ASP.NET page containing these controls is requested, it is the ASP.NET engine that decides whether to perform the validation on the client or on the server depending on the browser that is making the request. Therefore, our page's functionality changes depending on the requesting browser—thus enabling us to make our Web pages the best they can possibly be rather than dummying-down our Web applications for the lowest common denominator. Validation server controls are the only type of ASP.NET server controls that also generate client-side script. All the other controls work with the idea of making post backs to the server (a request to the server to get a response).

Six different validation server controls are available for ASP.NET:

1. RequiredFieldValidator
2. CompareValidator

Notes

3. RangeValidator
4. RegularExpressionValidator
5. CustomValidator
6. Validation Summary

We can also customize validation for our own needs. Then, if there are any errors in the form data, these validation server controls enable us to customize the display of error information on the browser.

We place validation server controls on our page as we would any other type of controls. After the user submits the form, the user's form information is sent to the appropriate validation control, where it is evaluated. If the information does not validate, the control sets a page property that indicates this. After all the form information is sent to all the validation server controls, if one or more of the validation server controls cannot validate the information sent to it, the entire form input is found to be invalid, and the user is notified.(See Table 4.2)

Table 4.2: Available validation server controls

Validation Server Control	Description
RequiredFieldValidator	Ensures that the user does not skip a form entry field
CompareValidator	Allows for comparisons between the user's input and another item using a comparison operator (equals, greater than, less than
RangeValidator	Checks the user's input based upon a lower- and upper-level range of numbers or characters
RegularExpressionValidator	Checks that the user's entry matches a pattern defined by a regular expression. This is a good control to use to check e-mail addresses and phone numbers
CustomValidator	Checks the user's entry using custom-coded validation logic
Validation Summary	Displays all the error messages from the valuators in one specific spot on the page

The RequiredFieldValidator Control

The RequiredFieldValidator server control makes sure that the user enters something into the field that it is associated with in the form. We need to tie the RequiredFieldValidator to each control that is a required field in the form. Although this is the simplest of the validation server controls, we must understand certain things about it.

To see an example of using the RequiredFieldValidator server control, create a Web form that contains a Textbox server control and a Button server control. Next to the button, place a RequiredFieldValidator server control. Our ASP.NET page should look like the code illustrated in Listing 2. (See Figure 4.2)

Example: **Using the RequiredFieldValidator control**

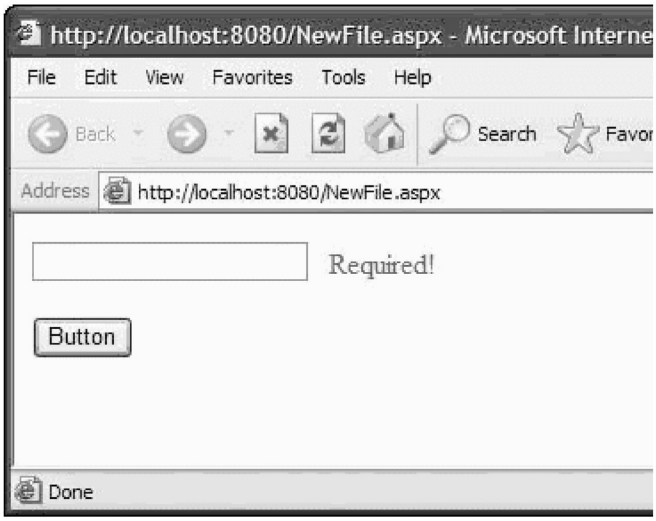
Visual C# .NET

```
<%@ Page Language= "VB" %>
<script runat= "server">
Sub Button1_Click (sender As Object, e As EventArgs)
Label1.Text = "Page is valid!"
```

Notes

```
End Sub
</script>
<html>
<head>
</head>
<body>
<form runat= "server">
<p>
<asp:TextBox id= "TextBox1"
runat= "server"></asp:TextBox>
 
<asp:RequiredFieldValidator
id= "RequiredFieldValidator1" runat= "server"
ErrorMessage= "Required!"
ControlToValidate= "TextBox1">
</asp:RequiredFieldValidator>
</p>
<p>
<asp:Button id= "Button1" onclick= "Button1_Click"
runat= "server" Text= "Button"></asp:Button>
</p>
<p>
<asp:Label id= "Label1" runat= "server"></asp:Label>
</p>
</form>
</body>
</html>
```

Figure 4.2: Causing the RequiredFieldValidator Server control to fire



Notes

There are a few things to be aware of when using the RequiredFieldValidator server control in our ASP.NET pages. First of all, the most important property of this validation control is the ControlToValidate property. The value assigned to this control needs to be the value of the id property for the control to which we want to link the RequiredFieldValidator control. We can link only one RequiredFieldValidator server control to any other server control on the page. For instance, if we have three required Textbox controls on our ASP.NET page, we need three separate RequiredFieldValidator controls to make sure that they are all required.

The second property of this control is the Error Message property. This is the text that appears where the RequiredFieldValidator control is located on the ASP.NET page if the Textbox is left empty. Instead of using the Error Message property, we can also use the Text property:

```
<asp:RequiredFieldValidator id= "RequiredFieldValidator1"
runat= "server" Text= "Required!"
ControlToValidate= "TextBox1">
</asp:RequiredFieldValidator>
Or we can use the following construct:
<asp:RequiredFieldValidator id= "RequiredFieldValidator1"
 runat= "server" ControlToValidate= "TextBox1">
 Required!
</asp:RequiredFieldValidator>
```

The Compare Validator Control

The CompareValidator server control compares the value entered into the form pasture to one more field, a database value, or other value that we identify. When comparing next to data types, we just set the Operator DataTypeCheck. following that is done, we can set the Type attribute to String, Integer, Double, Date, or Currency in the CompareValidator control to make sure that the user's input into the field is the specified type.

Using the CompareValidator server control, we can make comparisons between different controls within a form on our ASP.NET page. For example, if we want to compare what the user enters in the Password field to the entry in the Confirm Password field to see whether they are the same, we can use the CompareValidator server control. Using these two fields is a common practice when a form asks for a password. It ensures that the user does not mistype the password.

Example: **Using the Compare Validator Server Control**

Visual C# .NET

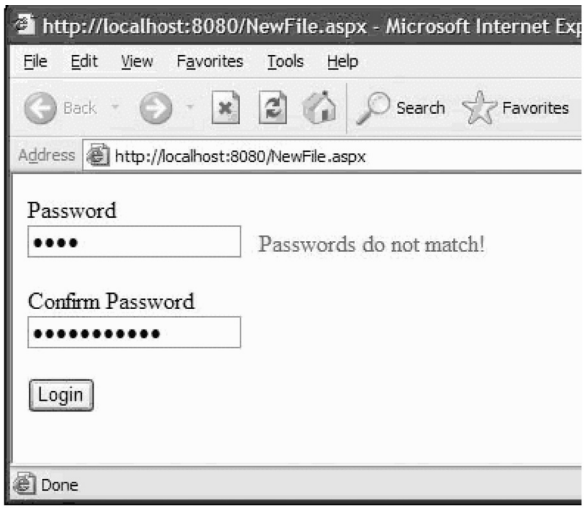
```
<%@ Page Language= "C#" %>
<script runat= "server">
Void Button1_Click(Object sender, EventArgs e) {
Label1.Text = "Passwords match";
}
</script>
<html>
<head>
</head>
<body>
<form runat= "server">
```

Notes

```
<p>
Password<br>
<asp:TextBox id= "TextBox1" runat= "server"
TextMode= "Password"></asp:TextBox>
 
<asp:CompareValidator id= "CompareValidator1"
runat= "server" ErrorMessage= "Passwords do not match!"
ControlToValidate= "TextBox2"
ControlToCompare= "TextBox1"></asp:CompareValidator>
</p>
<p>
Confirm Password<br>
<asp:TextBox id= "TextBox2" runat= "server"
TextMode= "Password"></asp:TextBox>
</p>
<p>
<asp:Button id= "Button1" onclick= "Button1_Click"
runat= "server" Text= "Login"></asp:Button>
</p>
<p>
<asp:Label id= "Label1" runat= "server"></asp:Label>
</p>
</form>
</body>
</html>
```

In this example, the Web form uses two Textbox server controls. One is for the user to enter the password, and the second Textbox control requires the user to enter the password again to ensure they did not mistype the original. By using the CompareValidator server control, we guarantee that they are equal strings. If they are not equal, the page returns an error message (See Figure 4.3). If they are equal, our page submits as valid.

Figure 4.3: The user mistyped the password and got an error message



Notes

Example: Checking to Make Sure Value Entered is of a Specific Data Type

```
Age:
<asp:TextBox id= “TextBox1” runat= “server” MaxLength= “3”>
</asp:TextBox>
 
<asp:CompareValidator id= “CompareValidator1” runat= “server”
ErrorMessage= “You must enter a number”
ControlToValidate= “TextBox1” Type= “Integer”
Operator= “DataTypeCheck”></asp:CompareValidator>
```

In this example, the user must enter an integer in the text box; otherwise, the CompareValidator server control fires and displays an error message on the page. By giving the Type property of the CompareValidator server control a value of Integer, we ensure that the value entered in the text box conforms to this .NET Framework data type.

We also have the option of not only comparing values against specific data types, but also ensuring that values are valid when compared against certain constants.

```
Age:
<asp:TextBox id= “TextBox1” runat= “server”></asp:TextBox>
 
<asp:CompareValidator id= “CompareValidator1” runat= “server”
Operator= “GreaterThan” ValueToCompare= “18”
ControlToValidate= “TextBox1” ErrorMessage= “You must be older than
18 to join” Type= “Integer”></asp:CompareValidator>
```

In this case, a few checks are made next to any value that the user types in the text box. The initial is based on the Type possessions in the CompareValidator server organize. The Type property has the value of digit, therefore any value located in the text box requirements to be conventional to this .NET Framework data type. If the user enters a string into the text box, the form submission is invalid. The next property is the Operator property. This property has a value of Greater Than, meaning that for the form submission to be valid, the value entered in the text box has to be greater than the value that is assigned to the ValueToCompare property. For this CompareValidator server control, the ValueToCompare property has a value of 18. This means that the value entered in the text box on the page must be an integer greater than 18. If it does not meet these criteria, the value is considered invalid, and the CompareValidator server control displays an error message in the browser.

In the circumstances shown in above example, we are not comparing two fields in the form; instead, we are comparing one field against a value that we have particular in cipher. The Operator property can contain one of the following values. The Operator property can contain one of the following values:

- Equal
- Not Equal

Notes

- Greater Than
- GreaterThanEqual
- LessThan
- LessThanEqual
- DataTypeCheck

The Range Validator Control

The RangeValidator server control is analogous to the CompareValidator server control, but the RangeValidator server control compares what is entered into the form field with two values and makes sure that what was entered by the client is between these two specified values.

For instance, imagine that we have a text box where we want end users to enter their ages. Instead of being greater than or less than a specific constant, we want the values entered to be between a specific ranges of numbers. For this, we use the RangeValidator server control, as illustrated in example.

Example: Using the Range Validator Server Control to Work with a Range of Numbers

```
Age:
<asp:TextBox id= “TextBox1” runat= “server”></asp:TextBox>
 
<asp:RangeValidator id= “RangeValidator1” runat= “server”
ControlToValidate= “TextBox1” Type= “Integer”
 ErrorMessage= “You must be between 30 and 40”
 MaximumValue= “40” MinimumValue= “30”></asp:RangeValidator>
```

In this case, the user should enter a value between 30 and 40 in the text box. If some number is entered that is outside of this range, the RangeValidator server control fires an error message and considers the form submission invalid.

The Type property enables us to create comparisons against many changed .NET Framework types, such as String, Integer, Double, Date, and Currency. These choices allow us to do a number of range comparisons. For instance, we can use the Currency value in the Type property to retrieve monetary-value entries that are within a certain range. We can also use the Date value for the Type property to make sure that the entry is between specific date ranges.

Also, just as we can use the String data type in the CompareValidator server control, we can use the String data type with the RangeValidator server control to make sure that the value entered falls within a specific range of characters. For example, if the user is entering her last name, and we want only people with last names starting with M and P to proceed, we can easily do this by using the RangeValidator server control, as illustrated in Example.

Example: Comparing an Entry to a Range of Characters

```
Last name:
<asp:TextBox id= “TextBox1” runat= “server”></asp:TextBox>
 
```

Notes

```
<asp:RangeValidator id= "RangeValidator1" runat= "server"
ControlToValidate= "TextBox1"
ErrorMessage= "Your last name needs to be between M and P"
MaximumValue= "Q" MinimumValue= "M"></asp:RangeValidator>
```

The Regular Expression Validator Control

The Regular Expression Validator server control is a corroboration control that enables us to confirm the user's input based on a prototype defined by a normal expression. This is a great control to check whether the user has entered a valid e-mail address or telephone number. In the past, these kinds of validations took a considerable amount of JavaScript coding. The Regular Expression Validator control with ASP.NET saves coding time.

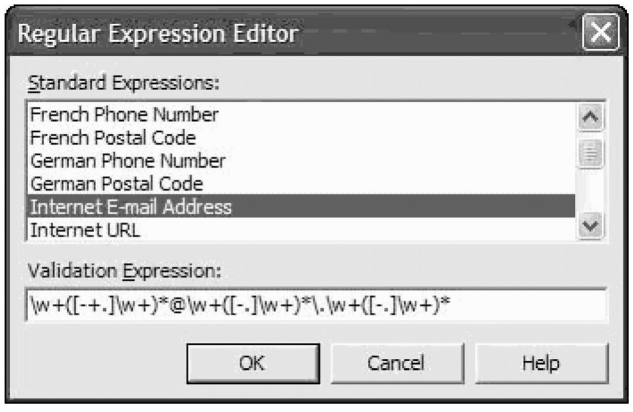
In the Properties window for the Regular Expression Validator server control, click the button in the Validation Expression box, and Visual Studio.NET provides we with a short list of expressions to use in our form via the Regular Expression Editor. However, we are not limited to these regular expressions in our ASP.NET applications. The list of prepared expressions is shown in Figure 4.3.

Example: Validating an e-mail address

```
Email:
<asp:TextBox id= "TextBox1" runat= "server"></asp:TextBox>
 
<asp:RegularExpressionValidator id= "RegularExpressionValidator1"
runat= "server" ControlToValidate= "TextBox1"
ErrorMessage= "You must enter an email address"
ValidationExpression= "\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*">
</asp:RegularExpressionValidator>
```

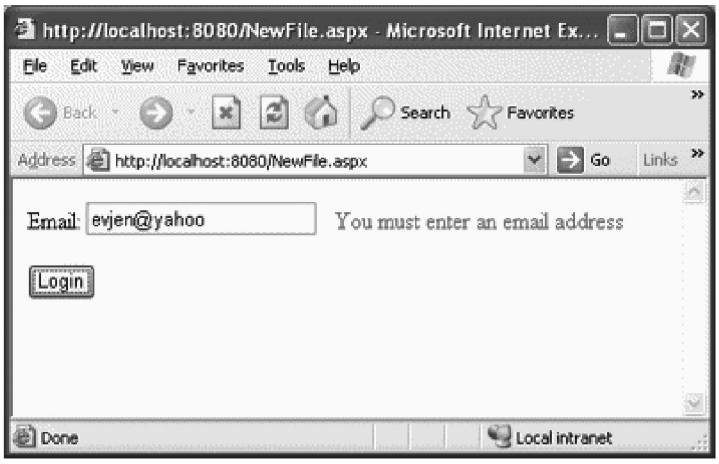
In this example, notice that we place the Internet e-mail addresses regular expression in our Validation Expression property. The great thing is that it is pretty simple, and it takes hardly any coding. Figure 4.5 shows the error message that results if a user enters an incorrect e-mail address in the text box.

Figure 4.4: The regular expression editor



Notes

Figure 4.5: Validating whether an e-mail address is entered in a text box



1. The Custom Validator Control

We are not limited to the validation controls that have been shown thus far in this; we also have the Custom Validator server control. The Custom Validator server control enables us to develop our own custom server-side or client-side validations. At times, we may want to compare the user's input to a value in a database, or to determine whether his input conforms to some arithmetic validation that we are looking for (for instance, if the number is even or odd). We can do all this and more by using this type of validation control.

Client-Side Validation

One of the cool things about using the validation controls, in general, is that they allow for client-side validation of certain HTML server controls. As anyone said, we can create our own JavaScript functions that provide us with a higher level of validation capabilities.

Example: Creating our own Client-Side Validation Functions

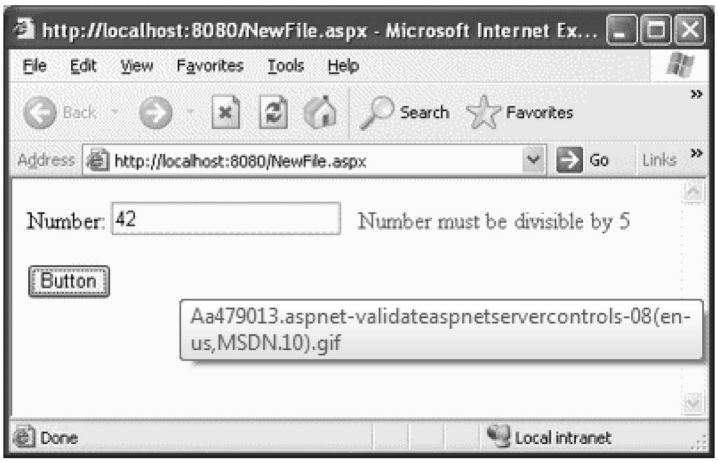
```
<%@ Page Language= "C#" %>
<script runat= "server">

void Button1_Click(Object sender, EventArgs e) {
Label1.Text = "VALID NUMBER!";
}

</script>
<html>
<head>
 <script language= "JavaScript">
function validateNumber(oSrc, args) {
args.IsValid = (args.Value % 5 == 0);
}
</script>
</head>
<body>
<form runat= "server">
```

Notes

```
 <p>
Number:
<asp:TextBox id= "TextBox1"
runat= "server"></asp:TextBox>
 
<asp:CustomValidator id= "CustomValidator1"
runat= "server" ControlToValidate= "TextBox1"
ErrorMessage= "Number must be divisible by 5"
ClientValidationFunction= "validateNumber">
</asp:CustomValidator>
</p>
<p>
<asp:Button id= "Button1" onclick= "Button1_Click"
runat= "server" Text= "Button"></asp:Button>
</p>
<p>
<asp:Label id= "Label1" runat= "server"></asp:Label>
</p>
</form>
</body>
</html>
```

The important part of the Custom Validator server control here is the Client Validation Function property. The value of this property must be the client-side JavaScript function that is in the page in this case, the validate Number function. By simply using this with the Client Validation Function property, we have tied the validation process to the custom client-side function that we created, as illustrated in Figure 4.6.

Figure 4.6: Performing a custom client-side validation using the custom validate server control



Notes

Example: Performing a custom client-side validation using the Custom Validator server control

Visual C# .NET

```
<%@ Page Language= "C#" %>
<script runat= "server">
void Button1_Click(Object sender, EventArgs e) {
if (Page.IsValid) {
Label1.Text = "VALID ENTRY!";
}
}
void ValidateNumber(object source, ServerValidateEventArgs
args)
{
try
{
int num = int.Parse(args.Value);
args.IsValid = ((num%5) == 0);
}
catch(Exception ex)
{
args.IsValid = false;
}
}
</script>
<html>
<head>
</head>
<body>
<form runat= "server">
<p>
Number:
<asp:TextBox id= "TextBox1"
runat= "server "></asp:TextBox>
 
<asp:CustomValidator id= "CustomValidator1"
runat= "server" ControlToValidate= "TextBox1"
ErrorMessage= "Number must be even"
OnServerValidate= "ValidateNumber"></asp:CustomValidator>
</p>
<p>
<asp:Button id= "Button1" onclick= "Button1_Click"
runat= "server" Text= "Button"></asp:Button>
</p>
```

Notes

```
<p>
<asp:Label id= "Label1" runat= "server"></asp:Label>
</p>
</form>
</body>
</html>
```

2. *The Validation Summary Control*

The Validation Summary server control works with all the validation server controls on the page. It takes all the error messages that the other validation controls send back to the page and puts them all in one spot (that we specify) on the page. These error messages can be displayed in a list, bulleted list, or paragraph.

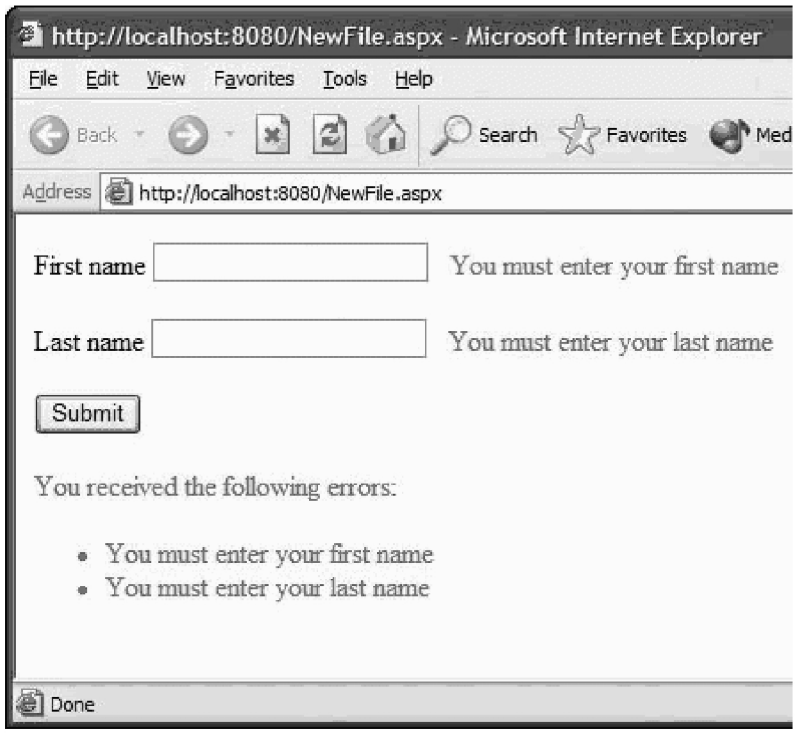
We can use the Validation Summary server control in a number of ways, but the example shows us how to use it in a simple manner. For this example, two text boxes on the page are associated with a RequiredFieldValidator control. When an error is triggered, not only does it display the error next to the text box itself, it also displays it in a summary at the bottom of the ASP.NET page. (See Figure 4.7)

Example: Having a Summary of Errors Appear on the Screen

```
<p>First name
<asp:TextBox id= "TextBox1" runat= "server"></
asp:TextBox> 
<asp:RequiredFieldValidator id= "RequiredFieldValidator1"
runat= "server" ErrorMessage= "You must enter your first
name"
ControlToValidate= "TextBox1"></asp:RequiredFieldValidator>
</p>
<p>Last name
<asp:TextBox id= "TextBox2" runat= "server"></asp:TextBox>
 
<asp:RequiredFieldValidator id= "RequiredFieldValidator2"
runat= "server" ErrorMessage= "You must enter your last
name"
ControlToValidate= "TextBox2"></asp:RequiredFieldValidator>
</p>
<p>
<asp:Button id= "Button1" onclick= "Button1_Click" runat=
"server"
Text= "Submit"></asp:Button>
</p>
<p>
<asp:ValidationSummary id= "ValidationSummary1" runat=
"server"
HeaderText= "You received the following errors:">
</asp:ValidationSummary>
</p>
<p>
<asp:Label id= "Label1" runat= "server"></asp:Label>
</p>
```

Notes

Figure 4.7: Validation errors showed using the validation summary server control



By default, the Validation Summary server control shows the errors in a bulleted list on the page using red text. We have the option to completely alter how output displays in the browser.

To change how the error messages are displayed, we can change the value of the Display Mode property of the Validation Summary control.

The possible values of this control can be set to the following:

- Bullet List
- List
- Single Paragraph

If we use the Bullet List value for the Display Mode property, it appears as shown in (Figure 4.7). If we use List, it appears without bullets. If we use Single Paragraph, the errors appear in a text area all on one line in a single paragraph.

Showing a Dialog Box Containing Error Notifications

One flexible feature of the Validation Summary server control is that we can easily show the ASP.NET page's errors in a pop-up dialog box. We have the option of showing the summary in the browser and the dialog box together, or just in the dialog box.

The property that controls whether the message appears in the browser is the Show Summary property. To turn off the display of validation errors in the browser, set the value of the Show Summary property to False. To show validation errors in a dialog box, set the Validation Summary control's Show Message Box property to true.

Notes

Did u know?

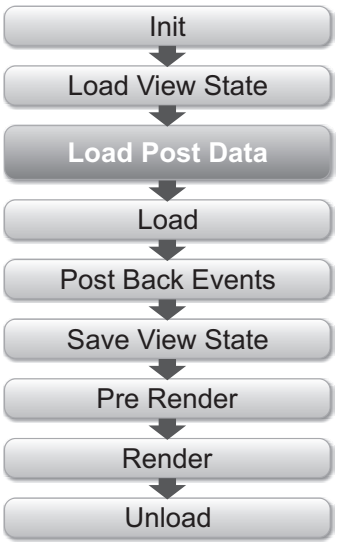
The first versions of Microsoft Windows ran every program in a single address space. Every program was meant to co-operate by yielding the CPU to other programs so that the graphical user interface (GUI) could multitask and be maximally responsive.

Task Create a Web page to show the use of compare validator.

Case Study

Validation Control in ASP.NET

As we all know the ASP.NET Page LifeCycle as:



As you can see, here LoadPostData is a part of PageLife Cycle and comes after LoadViewState. All of the Input control's data dont get wiped off during postback even if viewstate is not enabled for that control. So this is the LoadPostData method that is responsible to get the data from Form collection and assign it to the control. Now asp.net 4.5, validates the input only when you access the data from form collection. That is why if you look the stacktrace then it is visible that the exception is thrown fromLoadPostData method only and page life cycle is the last stage of ASP.NET Request Processing.

Now lets try to have an clarification using a demo. In ASP.NET 4.5, the validation triggers only when the data is accessed and the data is accessed at LoadPostData for input controls. So lets create a custom textbox. For this we will override LoadPostData method and will do nothing in that. It means that the data would not be accessed for the customTextBox at LoadPostData. So even if the ValidationMode is enabled for the CustomtextBox, it would not be fired. Let us see this.

We create a CustomtestBox and overridden the LoadPostData method as:

```
public class CustomTextBox : System.Web.UI.WebControls.TextBox
  {
    protected override bool LoadPostData(string postDataKey, NameValueCollection
postCollection)
```

Contd...

Notes

```
    {
        return false;
    }
}
```

You can see, it has just returned false in the LoadPostData method. Now it has used my CustomTextBox at my aspx page as

```
%@ Register TagPrefix="myControl"Namespace="CustomControl" Assembly="CustomControl" %>
<asp:Content ID="Content2" ContentPlaceHolderID="MainContent" Runat="Server">
    <myControl:CustomTextBox ID="TextBox1" runat="Server" ValidateRequestMode="Enabled">
    </myControl:CustomTextBox><br /?
    <asp:Button ID="Button1" runat="server" Text="Submit" OnClick="Button1_Click" />
</asp:Content>
```

Now set the requestValidationMode as 4.5 in web.config and enter some script tag and submit.

You would not get any error even RequestValidation is enabled. This proves the validation fires only when data is accessed from the form collection.

Questions

1. Explain the page life cycle of a Web page.
2. What is the use of requestValidationMode?

Self Assessment Questions

6. ..... are group of controls derived directly from the System.Web.UI.Web Controls base class.

   (*a*) HTML Server Controls (*b*) Web Server Controls

   (*c*) Client Server Controls (*d*) Both a and b

7. ..... are testing to determine whether the user entered something into the field.

   (*a*) Required Field Validator (*b*) CompareValidator control

   (*c*) Validation (*d*) Both b and c

8. The ..... compares the value of one control to another, or to an explicit value in the control's Value to compare property.

   (*a*) Required Field Validator (*b*) CompareValidator control

   (*c*) Validation (*d*) Both a and c

9. The ..... server control makes sure that the user enters something into the field that it is associated with in the form.

   (*a*) Required Field Validator (*b*) CompareValidator control

   (*c*) Validation (*d*) Both a and b

10. .....is the main mark-up language for displaying web pages and other information that can be displayed in a web browser.

    (*a*) XML (*b*) HTML

    (*c*) JavaScript (*d*) None of these.

Notes

11. A validation control that enables us to check the user's input based on a pattern defined by a regular expression:

    (*a*) RangeValidator server control (*b*) Compare Validation Control

    (*c*) Regular Expression Validator (*d*) None of these.

12. .....compares what is entered into the form field with two values and makes sure that what was entered by the user is between these two specified values.

    (*a*) RangeValidator server control (*b*) Compare Validation Control

    (*c*) Regular Expression Validator (*d*) None of these.

True or False

13. The HTML server controls are basically the original HTML controls but enhanced to enable server side processing.

    (*a*) True (*b*) False

14. A Validation Summary control is displayed when the Is Valid property of the page is false.

    (*a*) True (*b*) False

15. A database is an organized collection of data in digital form.

    (*a*) True (*b*) False

4.3 Summary

- The HTML server controls are basically the original HTML controls but enhanced to enable server side processing.
- ASP .NET Server Controls has higher level of abstraction. An output of an ASP .NET server control can be the result of many HTML tags that combine together to produce that control and its events.
- Web Server Controls are group of controls derived directly from the System.Web. UI.WebControls base class. They are reusable components that can perform function as the ordinary HTML controls; the real advantage of web controls is that they are programmable.
- The RequiredFieldValidator server control makes sure that the user enters something into the field that it is associated with in the form. We need to tie the RequiredFieldValidator to each control that is a required field in the form.
- The CompareValidator server control compares the value entered into the form field to another field, a database value, or other value that we specify.

4.4 Keywords

ASP.NET: It is a Web application framework developed and marketed by Microsoft to allow programmers to build dynamic Web sites, Web applications and Web services.

Compare Validator Control: The CompareValidator control compares the value of one control to another, or to an explicit value in the control's Value to compare property.

Database: A database is an organized collection of data, today typically in digital form. The data are typically organized to model relevant aspects of reality.

Notes

Hypertext Mark-up Language (HTML): It is the main mark-up language for displaying web pages and other information that can be displayed in a web browser.

Validation Summary Control: A Validation Summary control is displayed when the Is Valid property of the page is false.

Lab Exercise

1. Write a program to create a login form using HTML controls.
2. Create a Web page using the required field validator.

4.5 Review Questions

1. Define HTML server controls.
2. Explain validation controls.
3. Discuss the advantages of using HTML server controls.
4. What are the difference between web server control and HTML server control?
5. Explain HTML server controls advantage and disadvantage.
6. What are the differences between server-side and client-side validation?
7. Explain the required field validator with example.
8. Explain the compare validator.
9. Differentiate between compare validator and range validator control.
10. What is the use of validation summary control?

Answer for Self Assessment Questions

1. (*a*)	2. (*c*)	3. (*b*)	4. (*d*)	5. (*a*)
6. (*b*)	7. (*c*)	8. (*b*)	9. (*d*)	10. (*b*)
11. (*a*)	12. (*c*)	13. (*a*)	14. (*a*)	15. (*a*)

4.6 Further Reading

Books *Advanced Asp.Net Ajax Server Controls for .Net Framework 3.5,* by Calderon Adam

Online link http://books.google.co.in/books?id=yp7gUxY4zn4C&printsec=frontcover&dq=Server+Controls+Advance+3.5&hl=en&sa=X&ei=0CD9T7nsOpDPrQeJ8ujRBg&ved=0CDYQ6AEwAA#v=onepage&q&f=false

Unit 5: Database Access

Notes

CONTENTS

Objectives

After studying this unit, you will be able to:

- Define the database access using ADO.NET
- Explain the connection in ADO .NET
- Describe the command in ADO .NET
- Explain the ADO .NET DataAdapter

Notes

- Understand the ADO .NET DataSet
- Define ADO .NET DataReader
- Understand the connection pooling

Introduction

The concept behind concerning to a database has forever been confusing for learner application developers. This post focuses on the dissimilar methods accessible for connecting to databases in C#.

The software support provided by a database seller to access its own creation is identified as a 'Native driver'. This provides the simplest process of connectivity as the data access takes place without any interference by other software layers. But, as the database changes, the data access mechanism also needs to be changed accordingly in this move toward.

So as to standardize data access, Microsoft released a set of general access drivers called OleDb, which stands for Object Linking and Embedding DataBase. It is said to support compound skill, which means that it can either connect to a database using resident drivers or seek the help of other Data Access drivers like ODBC (Open DataBase Connectivity) to access the data store.

Later on, a set of APIs were made to communicate effectively with data in the data store. The common APIs available were

- DAO – Data Access Object
- RDO – Remote Data Object
- ADO – ActiveX Data Object
- ADO.NET – ActiveX Data Object for .NET

The two distinct type of data access models defined are:

- Connected Approach (Connection Oriented)
- ii) Disconnected Approach
- In Connected approach, whenever a query is run, the results are stored in an ActiveX DataSet in the server. A pointer to this ActiveX dataset is returned to the client. This pointer is known as Cursor. Therefore, to access the results, a connection must be consistently open between the server and the client.

In the Disconnected approach, the results of a query and sent back to the client, and stored in a client side cursor. This in turn means that continuous connectivity need not exist between the server and the client.

In .NET, connected approach is accomplished using the following objects.

- Connection
- Command
- DataReader

The Connection is used to send an authority across to the server, which is executed and the consequence is stored in the server side cursor, which in turn is read using the DataReader.

Disconnected approach makes use of the following objects:

- Connection

Notes

- Command
- DataAdapter
- DataSet

A DataAdapter acts as a bridge between a data source and a data class which is disconnected from the data source. It can provide a set of methods to carry out operations like Select, Insert, Update and Delete. A DataSet is the equivalent for client side cursor, which holds a portion of the database after execution of a query. It can be defined as a collection of DataRows and DataColumns. More popularly known as an in-memory database.

There are four methods of executing a query in C#.

- ExecuteReader() – which will return the address of the cursor.
- ExecuteNonQuery() – which will return the number of rows affected.
- ExecuteScalar() – which will return the first [row][column] of the result.
- ExecuteXmlReader() – which will return the instance of an Xml Parser.

5.1 Database Access Using ADO.NET

A database is a collection of information that is prearranged so that it can easily be accessed, managed, and updated. In one view, databases can be confidential according to types of satisfied: bibliographic, full-text, numeric, and imagery.

In computing, databases are sometimes classified according to their organizational approach. The most prevalent approach is the relational database, a tabular database in which data is defined so that it can be reorganized and accessed in a number of different ways. A distributed database is one that can be dispersed or replicated among different points in a network. An object-oriented programming database is one that is congruent with the data defined in object classes and subclasses.

5.1.1 ADO.NET

The ADO.NET is the new database technology of the .NET (DotNet) platform, and it builds on ADO (ActiveX Data Objects). ADO is a language-neutral object representation that is the keystone of Microsoft's Universal Data admission approach. ADO.NET defines DataSet and DataTable substance which are optimized for moving disengaged sets of data across intranets and Internets, including through firewalls. It is also including the traditional Connection and Command objects, as well as an object called a DataReader that resemble a forward-only, read-only ADO record set.

If you create a new application your application require most of the time some form of data access. ADO.NET provides data access services in the Microsoft .NET Platform.

You can use ADO.NET to access data by using the new .NET Framework data providers which are

- Data Provider for SQL Server(System.Data.SqlClient).
- Data Provider for OLEDB(System.Data.OleDb).
- Data Provider for ODBC(System.Data.Odbc).
- Data Provider for Oracle(System.Data.OracleClient).

The ADO.NET is a set of classes that expose data access services to the .NET developer. The ADO.NET classes are found in System.Data.dll and integrated with XML classes in System.Xml.dll.

Notes

There are two central components of ADO.NET classes:

- the *DataSet*, and
- the .NET Framework *Data Provider*.

Data Provider is a set of components including the:

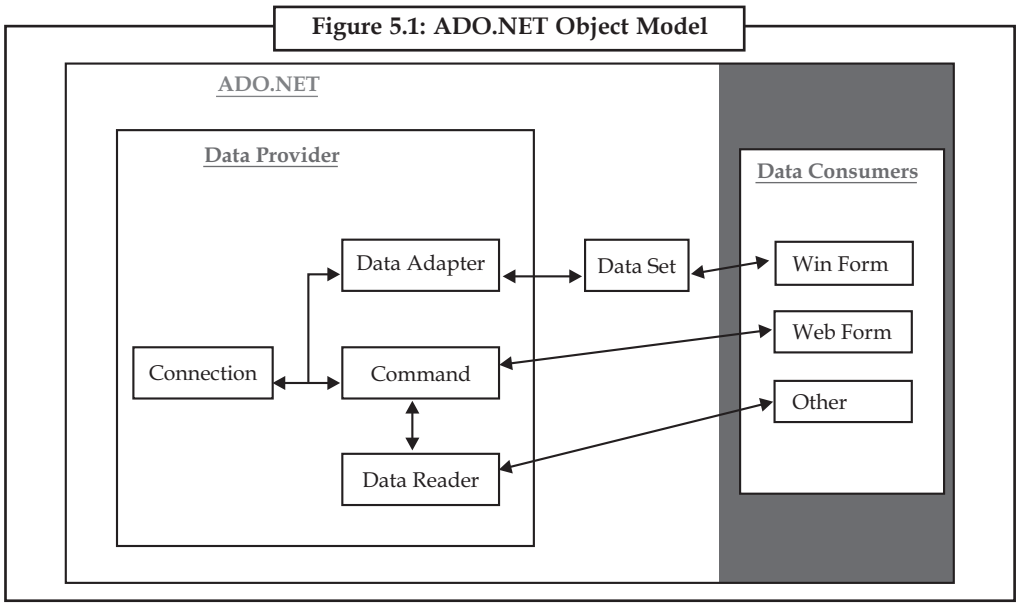
- Connection object(SqlConnection, OleDbConnection, OdbcConnection, OracleConnection),
- Command object(SqlCommand, OleDbCommand, OdbcCommand, OracleCommand),
- DataReader object (SqlDataReader, OleDbDataReader,OdbcDataReader, OracleDataReader),and DataAdapter object(SqlDataAdapter, OleDbDataAdapter, OdbcDataAdapter, OracleDataAdapter).

DataSet object represents a disconnected cache of data which is made up of DataTables and DataRelations that represent the result of the command.

5.1.2 The ADO.NET Object Model

The ADO.NET object model is as following (See Figure 5.1):

Figure 5.1: ADO.NET Object Model



1. Connection to an ADO.NET Database

Before working with a database, you have to add (here) the OleDb .NET Data Provider namespace, by placing the following at the start of your code module:

```
using System.Data.OleDb;
```

Similarly for the SqlClient .NET Data Provider namespace:

```
using System.Data.SqlClient;
```

The using statement should be positioned first in your code.

Now we have to declare a connection string pointing to a MS Access database "PersonDatabase.mdb".

```
publicstringconString=@"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=..\..\PersonDatabase.mdb";
```

Notes

The database should be in the specified path, otherwise you should change the path accordingly. The next step is to create an OleDbConnection object. We pass then the connection string to this OleDbConnection object.

You can code now to create a new ADO.NET Connection object in order to connect to an OLE DB provider database.

```
OleDbConnection con = new OleDbConnection(conString);
```

You can also explicitly reference declared objects if you don?t mind typing a lot.

```
System.Data.OleDb.OleDbConnection con = new System.Data.OleDb.
OleDbConnection(conString);
//1.Connection to a database
//using declaration for OLE DB
using System.Data.OleDb;
//specify the ConnectionString property
publicstringconString=@"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=.\\PersonDatabase.mdb";
//Initializes a new instance of the OleDbConnection
OleDbConnection con = new OleDbConnection     (conString);
```

2. *DataSet*

The DataSet is similar to an array of disconnected Recordset objects. It supports disconnected data access and operations, allowing greater scalability because you no longer have to be connected to the database all the time. DataSet is a copy of a extracted data being downloaded and cached in the client system.

DataSet object is made up of two objects:

- DataTableCollection Object containing null or multiple DataTable objects(Columns, Rows, Constrains)
- DataRelationCollection Object containing null or multiple DataRelation objects

which establishes a parent/child relation between two DataTable objects.

```
//Create a DataSet
DataSet dset = new DataSet();
```

There are two types of DataSets:

Typed Dataset

Typed dataset is derived from the base DataSet class and then uses information in an XML Schema file (.xsd file) in order to generate a new class. Information from the schema (tables, columns, and so on) is generated and compiled into this new dataset class as a set of first-class objects and properties. Typed dataset is easier to read.It?s also supported by IntelliSense in the Visual Studio Code Editor. r /> At compile time, it has type checking so that there are less errors in assigning values to dataset members.

Using Typed dataset has many advantages.

Notes

Example: The following code accesses the CustomerID column in the first row of the Customers table

```
string str;
str=dset.Customers[0].CustomerID;
```

Untyped Dataset

Untyped dataset is not defined by a schema, instead, you have to add tables, columns, and other elements to it yourself, either by setting properties at design time or by adding them at run time. For example : scenario: if you don't know in advance what the stucture of your program is that interacting with a component that returns a dataset. the equivalent code above for Untyped dataset is:

```
string str;
str=(string)dset.Tables[ "Customers"].Row[0].[ "CustomerID"];
```

A dataset is a container; therefore, you have to fill it with data.

You can populate a dataset in a variety of ways;

- By using Fill method.
- By creating DataRow objects and adding them to the table's Rows collection(only at run time).
- Read an XML document or stream into the dataset.
- Merge (copy) the contents of another dataset.

5.1.3 Display Data in a DataGrid

The Windows Forms DataGrid control displays data in a sequence of rows and columns. The Windows Forms DataGrid control provides a user interface to ADO.NET datasets. It displays tabular data and allows for updates to the data source. When you set DataGrid control to a suitable data source, the control will be automatically populated, creating columns and rows based on the shape of the data. You can use the DataGrid control for displaying either a single table or the hierarchical relationships between a set of tables. If you want to work with the DataGrid control, DataGrid should be bound to a data source by using

- the DataSource and
- DataMember properties at design time

  or

- the SetDataBinding method at run time.

Here is the binding to the DataGrid control with DataSet used in this project.

```
this . dataGrid1 . DataSource = datc . dSet . Tables[ "PersonTable"];
```

You can only show one table in the DataGrid at a time. If you define a parent-child relationship between tables, you can navigate between the related tables to select the table you want to display in DataGrid control.

Example:

```
dset.Relations.Add( "CustomerOrders",
dset.Tables[ "customers"].Columns[ "CustomerID"],
dset.Tables[ "orders"].Columns[ "CustomerID"]);
```

Notes

```
//here you can use one of the following
this.dataGrid1.DataSource=dset.Tables[ "customers"];
OR
this.dataGrid1.SetDataBinding(dset, "customers");
customers: Parent table
orders: Child table
```

CustomerID in Orders is a foreign key referring to CustomerID primary key in Customers table.

- If you update the data in the bound DataSet through any mechanism, the DataGrid control reflects the changes. You can update the data in the DataSet through the DataGrid control, if the data grid and its table styles and column styles have the ReadOnly property set to false.

There are four most typical valid data sources for the DataGrid:

- DataTable class
- DataView class
- DataSet class
- DataViewManager class

5.1.4. DataBindings for Textboxes

DataBinding is the ability to bind some elements of a data basis with some graphical elements of a submission. The data in Windows Form is spring by calling DataBindings. Windows Forms allow you to bind easily to almost any structure that contains data. Windows Forms Controls support two types of data binding:

1. Simple Data Binding
2. Complex Data Binding

Simple Data Binding allows you to display a single data element, such as a column value from a dataset table, in a control. You can simple-bind any property of a control to a data value. Simple Data Binding can be performed either at design time using DataBindings property of a control OR dynamically at run time. This is the type of binding typical for controls such as a TextBox control or Label control that displays typically only a single value.

Example: Simple DataBinding for TextBox "textBox1"

```
textBox1.DataBindings.Add( "Text", dataset, "studentTable.studentID");
```

The control "textBox1" is bound to the "studentID" column of a table "studentTable" on the DataSet(dataset) through the BindingContext object.

Complex data binding is the ability of a control to bind to more than one data element, typically more than one record in a database, or to more than one of any other type of bindable data element. DataGrid, ListBox and ErrorProvider controls support Complex data binding.

Here is the method used in this project to bind all TextBoxes:

```
private void fnGetDataBindingForTextBoxes()
{
this.textboxFirstname.DataBindings .Add( "Text", datc.dSet.Tables[
"PersonTable"], "FirstName");
```

Notes

```
this.textboxLastname.DataBindings.Add( "Text", datc.dSet.Tables[
"PersonTable"], "LastName");
this.textboxTitle.DataBindings.Add("Text", datc.dSet.Tables["PersonTable"],
"Title");
this.textboxCity.DataBindings.Add("Text", datc.dSet.Tables["PersonTable"],
"City");
this.textboxCountry.DataBindings.Add("Text",datc.dSet.
Tables["PersonTable"], "Country");
}
```

Did u know? The term Object Database first introduced in 1985.

5.2 Database Connection

Connecting to an Access database requires a different connection object and connection string. Access uses something called OLEDB, so you need to create a database object of this type. Double click your form. Outside of the form load event, type the following:

```
System.Data.OleDb.OleDbConnection con;
```

Inside of the form load event, type this:

```
con = new System.Data.OleDb.OleDbConnection();
```

This sets up a new connection object called con. It is an OLEDB connection object. The connection string can be found using the same technique as for SQL Server Express, *just outlined*. Once the string is pasted over, it will look like this:

```
con.ConnectionString = "PROVIDER=Microsoft.Jet.OLEDB.4.0;
Data Source=C:/Workers.mdb";
```

(Notice that you can use a single forward slash in a file name, but not a single backslash.)

The Provider you use for Access databases is called Microsoft Jet. We are using version 4.0 in the code above. Again, we need to tell C# where the database is. This is done with Data Source. The source above is pointing to a database called Workers.mdb that is on the C drive.

The connection to the database is done with the Open method of your connection object:

```
con.Open();
```

Close the connection in a similar way:

```
con.Close();
```

You can also issue a Dispose command at the end, if you want. This will do the "tidying up" for you:

```
con.Dispose();
```

Add a few message boxes and your coding window should look like this:

```
System.Data.Oledb.OleDbConnection con;
Private void Form1_load(object sender, EventArgs e)
{
    Con=new System.Data.OleDb.OleDbConnection();
Con.ConnectionString= "PROVIDER=Microsoft.Jet.OLEDB.4.0;
```

Notes

```
        Data Source=C:/Workers.mdb";
        Con.Close();
MessageBox.Show( "Database Closed");
Con.Dispose();
}
```

Run your programme and you should see the two message boxes display. The form will display after you click OK on these.


Caution

Do not calls Close or Dispose on a Connection, a DataReader, or any other managed object in the Finalize method of your class. In a finalizer, you should only release unmanaged resources that your class owns directly. If your class does not own any unmanaged resources, do not include a Finalize method in your class definition.

5.3 Database Command

Commands are issued next to databases to receive trial against data stores and to include any declaration that can be issued adjacent to a database. You can use the OleDbCommand or the SqlCommand classes to get a command to your data store, and OleDbCommand can be precise to the data stock up The SqlClient class (to connect to a computer that is running Microsoft SQL Server) and the OleDb class (for any database that has an OLE DB or ODBC driver available) within ADO.NET. However, the code is generally the same for both.

With ADO, you can issue commands through the Command, the Connection, or the Recordset object. In ADO.NET, only the Command objects (SqlCommand or OleDbCommand) run commands.

To run a command, follow these steps:

Follow these steps to create a new console application in Microsoft Visual C# 2005 or in Microsoft Visual C# .NET:

- Start Microsoft Visual Studio 2005 or Microsoft Visual Studio .NET.
- On the File menu, point to New, and then click Project.
- In the New Project dialog box, click Visual C# Projects under Project Types, and then click Console Application under Templates.

Make sure that your project contains a reference to the System.Data namespace, and add a reference if it does not.

Use the using statement on the System and System.Data namespaces so that you do not have to qualify declarations in those namespaces later in your code. You can also include System.Data. SqlClient or System.Data.OleDb, depending on which one you are using.

```
using System;
using System.Data;
using System.Data.SqlClient;
```

Before you can create a connection to a database, you must have a connection string. Connection strings contain all of the information that you need to establish a database connection, including the server name, the database name, the user ID, and the password. For example, the following connection string points to a local computer that is running SQL Server:

Notes

For OleDb connections:

Provider=SQLOLEDB.1;User ID=<UID>;Initial Catalog=pubs;Data Source=(local)

For SqlClient connections:

User ID=<UID>;Initial Catalog=pubs;Data Source=(local)

Visual Studio creates a static class and an empty Main() procedure. Declare a string variable, and store the appropriate connection string for your database in this procedure.

```
class Class1
{
        static void Main(string[] args)
                {
                        string sConnectionString =
"User ID=<UID>;Initial Catalog=pubs;Data Source=(local)";
                }
}
```

Using this connection string, create a new OleDbConnection or SqlConnection object, and call its Open method to establish a connection to your database:

```
SqlConnection objConn = new SqlConnection(sConnectionString);
objConn.Open();
```

Create a SqlCommand or OleDbCommand object, and pass in the command that you want to run and the connection object that you created in the previous step. The following sample code passes in the INSERT statement:

```
string sSQL = "INSERT INTO Employee " +
"(emp_id, fname, minit, lname, job_id, job_lvl, pub_id, hire_date) " +
"VALUES ('MSD12923F', 'Duncan', 'W', 'Mackenzie', 10, 82, '0877','2001-01-01')";
SqlCommand objCmd = new SqlCommand(sSQL,objConn);
```

After you create the SqlCommand or OleDbCommand object, you can call the ExecuteNonQuery method to run the command that it represents. ExecuteNonQuery is designed for commands that do not return any results (such as the DELETE, UPDATE, and INSERT statements). If the statement runs without throwing an exception (see the following code), the command has been executed successfully against the database.

```
objCmd.ExecuteNonQuery();
```

5.3.1 How to Use Parameters

When you run commands against a database (such as the UPDATE, the INSERT, and the DELETE statements or calls to stored procedures), these commands are frequently parameterized. This allows the command to be created one time but executed multiple times with different values that are inserted instead of parameters. Consider the corresponding DELETE statement to the INSERT statement:

```
string sSQL = "DELETE FROM Employee WHERE emp_id = @emp_id"
```

The parameter name ("@emp_id") in this DELETE statement represents a parameter than you can replace with different values each time you run the command.

To use parameters with your command, follow these steps:

- Create your OleDbConnection or SqlConnection object, as you did in the "How to run a command" section.

Notes

- Replace values with placeholders (for example, "@emp_id" or "@fname") so that your command text uses parameters. See the DELETE statement before these steps for an example.
- Create your OleDbCommand or SqlCommand object, and pass in the connection object that you created in the first step and the command text that contains the parameter placeholders.

For each parameter, add a parameter object to the command object's parameters collection. For each parameter, you must specify a name and data type.

```
objCmd.Parameters.Add( "@emp_id", SqlDbType.Char, 9);
```

Stored procedures can have parameters that return values and output parameters. You must also set a value for each input parameter before you can run the query:

```
objCmd.Parameters[ "@emp_id"].Value = "MSD12923F";
```

Run the query as follows:

```
try
{
    objCmd.ExecuteNonQuery();
}
    catch (System.Exception e)
{
    Console.WriteLine(e.Message);
}
Console.WriteLine( "Record Deleted");
```

Task Write a program to insert the data in the data base.

Self Assessment Questions

Multiple Choice Questions

1. Data Adapter to focus the …….on a specific member.
   (*a*) execute Xml reader() (*b*) update operation
   (*c*) server controls (*d*) validator control
2. Set of APIs was made to ……with data in the data store.
   (*a*) connectivity (*b*) connection
   (*c*) communicate effectively (*d*) linking and embedding
3. ……is one that can be dispersed or replicated among different points in a network.
   (*a*) A distributed database (*b*) Data binding
   (*c*) ODBC stands (*d*) Embedding
4. ……, as well as an object called a Data Reader.
   (*a*) Database (*b*) Execute scalar()
   (*c*) the ADO.NET (***d***) Command objects

Notes

5. The data set is similar to an array of disconnected Record set objects.

(*a*) True (*b*) False

5.4 DataAdapter

The ADO.NET DataAdapter and serialisable DataTable are the pivot pins of ADO.NET have disengaged structural design. These classes provide a way to depiction one or more rowsets and an instrument to update the information. This is a new mechanism for data access developer's inscription code for all application architectures.

Note that the DataAdapter is implemented by each of the .NET Data Providers incorporated in the Windows .NET Framework. For example, in the SqlClient .NET Data supplier, the DataAdapter is implemented as the System.Data.SqlClient.SqlDataAdapter. The DataAdapter is not really a new idea born with ADO.NET. Its roots come not from Tao, but from the DOW – the Data Object Wizard first introduced in Visual Basic 6.0 but henceforth abandoned in light of the innovations in Visual Studio.NET. The DOW was used to generate a class that fetches a rowset and defines the UPDATE, DELETE and INSERT SQL needed to manage the rowset. Basically, the Visual Basic.NET DataAdapter does the same thing. Unlike the Data Object Wizard, the DataAdapter permits you to define both the input and action queries with a few more options. It also sports a facility to automatically generate the action queries – but regrettably, only for "home" database challenges.

One of the most significant differences between COM-based ADO "classic" (ADOc) and ADO. NET is how updates are handled. In ADOc, action SQL was generated on the fly controlled by the Update Criteria and other properties. It did not require an extra round trip to generate this SQL, but it did require a bulkier SELECT FOR BROWSE to return schema information needed to construct these action commands. ADO.NET takes another approach – it leaves the generation of the update commands up to the developer. This means that developers have far more flexibility in how updates are posted to the data.

One of the most significant differences between COM-based ADO "classic" (ADOc) and ADO. NET is how updates are handled. In ADOc, action SQL was generated on the fly controlled by the Update Criteria and other properties. It did not require an extra round trip to generate this SQL, but it did require a bulkier SELECT FOR BROWSE to return schema information needed to construct these action commands. ADO.NET takes another approach – it leaves the generation of the update commands up to the developer. This means that developers have far more flexibility in how updates are posted to the data.

5.4.1 Choosing between the DataAdapter and the DataReader

When you first start working with the ADO.NET DataAdapter, you might be puzzled as to how to best use it with existing data access architectures. Developers, yearning for better performance, often lean toward the low-level DataReader approach – rolling their own update code or using the Command methods to execute stored procedures which update the source tables. They are not convinced that the extra features exposed by the DataAdapter and its associated DataSet and DataTable are worth the extra CPU cycles. In addition, once the DataTable is created, the additional functionality exposed by its methods and properties further simplify and stabilise the code – without further loss in the developers or the code's performance.

5.4.2 Getting the Best Performance from the DataAdapter

The disconnected architecture implemented by ADO.NET assumes that your code returns a set of rows to the client and posts changes to the in-memory copy of the data. These changes are not reflected in the database until you submit a "batch" of updates using the DataAdapter

Notes

Update method. No, it is not really a batch in the traditional sense, but that is what Microsoft chose to call it. What really happens is ADO.NET walks the set of rows passed to the Update method and executes either the UpdateCommand, DeleteCommand or InsertCommand based on the row's RowState value. Each execution requires a round trip. It does not help that all too many of the examples you will encounter suggest that you use the DataAdapter to fetch all of the rows (and columns) from a database table and construct appropriate action queries to enable the Update method. While this approach is valid for "toy" databases, it flies in the face of the constraints of scalable, high-performance, real-world systems. If you are working with home databases or those in your office with only a few hundred rows, and you are not concerned with multi-user issues (and never plan to be), whether or not you fetch all of the rows from the table would not make much difference. However, in most business applications, especially those that have to scale up to support dozens to thousands of users and work with thousands to millions of rows, how you fetch and update the data is critical to a successful application.

5.4.3 The Role of the CommandBuilder

One of the most respected produce managers at Microsoft quipped that the CommandBuilder should be renamed the "CommandDonotUseBuilder". The CommandBuilder is a class used to construct the action SQL that changes the data particular in the DataAdapter SelectCommand. That is, the CommandBuilder uses the SelectCommand.CommandText to query the database engine for additional schema information using the undocumented SET NO_BROWSETABLE function (or other mechanisms as dictated by the data provider). If you choose to use the CommandBuilder, you will discover the SQL it generates is pretty crude. In that the approach it takes to concurrency management is very simplistic and inefficient and, more importantly, not functional in a variety of common situations – not to mention the need for a costly additional round trip to the server.

5.4.4 Fetching Data with the DataAdapter

There are several approaches to fetching data using the DataAdapter – only a few of the more common techniques are discussed here. Each of these techniques lends itself to addressing specific issues and data access challenges and opportunities for better performance. The following discussion illuminates these techniques and balances their advantages and challenges.

- Return a rowset generated by an SQL SELECT that references a single-table, view or stored procedure.
- Return two or more rowsets generated by a "batch" SQL query containing more than one SELECT referencing one or more data tables or views.
- Return a rowset generated from a JOIN product that references several related tables.
- Return a rowset generated from a composite query. That is, the data is returned from more than one data source.

On the surface, DataAdapter fetch mechanics are really quite simple. To retrieve the data, set the SelectCommand property with a .NET data provider Command object describing either the appropriate SELECT statement or stored procedure. Next, describe and specify the parameters and values needed to focus the query's WHERE clause and use the Fill method to execute the query. For example:

```
da = New SqlDataAdapter( "SELECT pubid, pubname, _
city, state" & " FROM publishers WHERE _
state = @StateWanted " & " SELECT pubid,isbn, title, _
price FROM titles where pubid in " & _
```

Notes

```
" (SELECT pubid from publishers WHERE state = _@StateWanted)", cn)
da.SelectCommand.Parameters.Add(_ "@StateWanted", SqlDbType.VarChar, 2)
dsPublisherTitle = New DataSet() _
'Create new Dataset
```

The Fill method makes sure the connection is open, executes the query, constructs any needed DataTables, completes rowset population (fetches the rows into the DataTable Rows collection), and closes the connection.

Using the SelectCommand to manage an SQL query and here is an example of the code:

```
da.SelectCommand.Parameters(_"@StateWanted").Value = txtStateWanted.Text
da.Fill(dsPublisherTitle)
DataGrid1.DataSource = dsPublisherTitle.Tables(0)
DataGrid2.DataSource = dsPublisherTitle.Tables(1)
lblTables.Text = dsPublisherTitle.Tables.Count.ToString
lblRows.Text = _
dsPublisherTitle.Tables(0).Rows.Count.ToString & " : " & _
dsPublisherTitle.Tables(1).Rows.Count.ToString
```

Depending on how the DataAdapter is configured, the Fill method either creates a new DataTable for each rowset returned, or cleverly "updates" the DataTable objects already constructed in a selected DataSet. But before we get to that most powerful achievement, let us step through some of the interesting fetch scenarios. The scenario that makes the most sense for you depends on where the data is located and whether or not you need to use the Update method to perform updates on a selected database table.

5.4.5 The Single-Table Fetch Approach

Most of the examples illustrates the use of the DataAdapter by coding the SelectCommand CommandText as "SELECT * FROM Authors". This simple query masks several important issues. First, it does not limit the number of rows or columns returned from the data table. This approach is fine for home databases, but cripples an application dealing with serious amounts of data. This means you will need an example of fetching selected columns focused (and limited) by a WHERE clause for your production application that extracts rows from that 100,000 (or 1,000) row table (assuming you want to build a scalable application). This simplistic approach also assumes that you can return the information you want without benefit of data from other tables. Sure, there are lots of cases where a simple query will do. But SELECT * is evil. It blindly returns all columns from a product whether you need them or not and assumes that changes to the database table post-deployment would not affect your application. This is called optimistic programming, or programming by wishful thinking. Ironically, this approach is the one most likely to work with the DataAdapter's CommandBuilder used to generate the action queries to update the fetched data.

5.4.6 The Multiple Resultset Approach

Assuming the single-table query approach would not work for your application, you should examine the outcome of a "multiple resultset" query. Suppose you avoid the lure of the DataAdapter configuration wizard and create a DataAdapter using the code in our first listing, above.

This query returns two independent rowsets which might be logically related. In this case, ADO.NET constructs two DataTables (or it will if the Fill method is executed) one containing selected Publisher names and the other associated Titles. It is up to my own client-side code to relate these two tables if, let ADO.NET manage and bind controls to show these relationships.

Notes

The code shown in our second listing does just that – it executes the query and returns the two DataTables which are bound to separate DataGrid controls.

This approach assumes that you do not expect to update the source tables, you expect to use the DataAdapter Update method to update only one of the source tables, or you plan to provide your own update routines and not use the Update method. In this case the CommandBuilder will be able to construct the action commands for you – but only for the first data table specified by the initial SELECT.

5.4.7 The JOIN Product Approach

Of course, you can still use the server's ability to JOIN two or more database tables or views together to produce a composite SQL product and the DataAdapter can execute this query and construct a DataTable from the product's rowset. No, ADO.NET does not treat this rowset differently from any other – one DataTable is created for each unique rowset returned by the server. This means that you can return a rowset from a JOIN product and another from a single-table query or another JOIN product and use these to generate multiple DataTables as necessary.

Unlike the previous approach, the CommandBuilder would not be able to construct the action Commands so you will have to fall back on your own ad hoc action commands or stored procedures to make changes to the base tables. For many developers, this is what they have had to do for years anyway.

5.4.8 The Composite Query Approach

One of the more interesting (and powerful) approaches you can use is construction of "composite" queries that return rowsets from more than one data source. The trick here is to use more than one DataAdapter to specify different data sources drawing rowsets from the same or different servers, from different providers or from non-database data sources. Once the data is downloaded into DataTable objects, it is a simple matter to construct relationships between them. Updating is also straightforward in this case, and if the SelectCommand is simple enough the CommandBuilder can be used to generate the action commands. Ideally, you could use stored procedures on each data source system to perform updates to the data table associated with each DataAdapter.

This approach is bound to become one of the most popular as it deals with so many issues quite neatly. Regardless of the SQL product's source, you can define a suitable DataAdapter to focus the update operation on a specific member. That is, you define a DataAdapter UpdateCommand, InsertCommand and DeleteCommand to change a specific row in a specific data table – you define a separate DataAdapter for each database table you wish to update. Just because a DataSet contains DataTables drawn from disparate data sources that is no reason you cannot update any or all of the base database tables as needed.

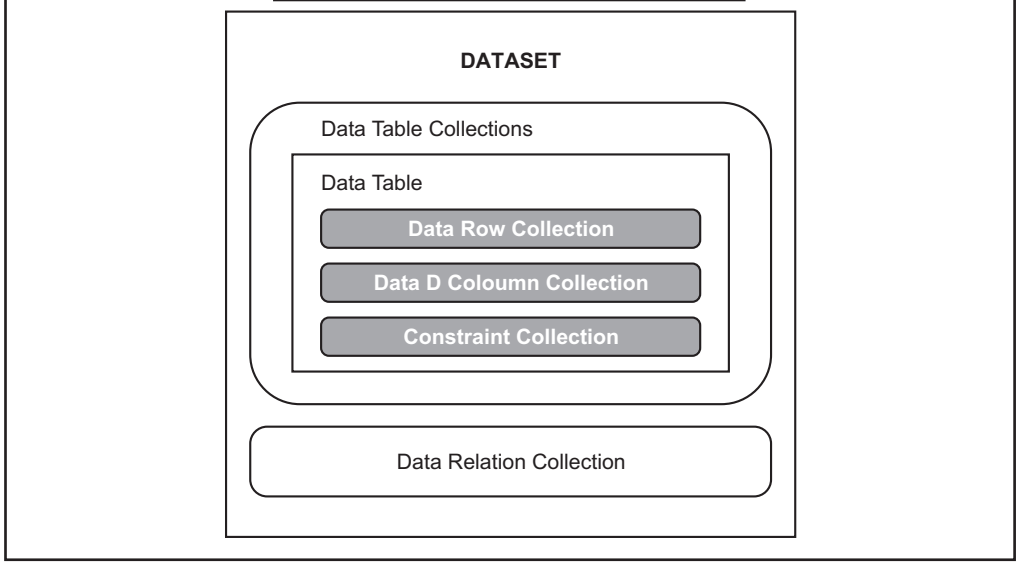
Task Write a program to INSERT, UPDATE and DELETE data from a table.

5.5 ADO.NET DataSet

The ADO.NET DataSet (See Figure 5.2) contains DataTableCollection and their DataRelationCollection. It represents a collection of data retrieve from the Data Source. We can use Dataset in combination with DataAdapter class. The DataSet object offers disconnected data source architecture. The Dataset can work with the data it contains, without knowing the source of the data coming from. That is, the Dataset can work with a disconnected mode from its DataSource. It gives a better advantage over DataReader , because the DataReader is working only with the connection oriented Data Sources.

Notes

Figure 5.2: DataSet Architecture

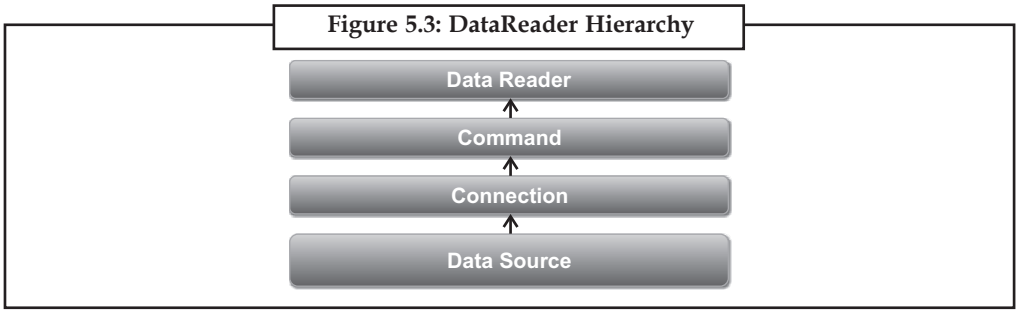


The Dataset contains the copy of the data we requested. The Dataset contains more than one Table at a time. We can set up Data Relations between these tables within the DataSet. The data set may comprise data for one or more members, corresponding to the number of rows. The DataAdapter object allows us to populate DataTables in a DataSet. We can use Fill method of the DataAdapter for populating data in a Dataset. The DataSet can be filled either from a data source or dynamically. A DataSet can be saved to an XML file and then loaded back into memory very easily.

5.6 ADO.NET DataReader

DataReader Object in ADO.NET is a stream-based, forward-only, read-only retrieval of query output from the Data Sources, which do not update the data. The DataReader cannot be shaped directly from code; they can make only by calling the ExecuteReader process of a Command Object.

Figure 5.3: DataReader Hierarchy



SqlDataReader sqlReader = sqlCmd.ExecuteReader();

The DataReader Object provides an association oriented data entrée to the Data Sources. A Connection Object can contain only one DataReader at a time and the connection in the DataReader remains open, also it cannot be used for any other purpose while data is being accessed. When we started to read from a DataReader it should always be open and positioned prior to the first record. The Read() method in the DataReader is used to read the rows from DataReader and it always moves forward to a new valid row, if any row exist.

DataReader.Read();

Notes

Usually we are using two types of DataReader in ADO.NET. They are SqlDataReader and the OleDbDataReader. The System.Data.SqlClient and System.Data.OleDb are containing these DataReaders respectively.


Did u know?

SQL was initially developed at IBM by Donald D. Chamberlin and Raymond F. Boyce in the early 1970s.

5.7 Connection Pooling

Data-driven applications frequently access a database to query data. There are two broad ways to access the data:

1. Open a connection with the database as your application starts and keep it open throughout the life of your application. Fire all the queries through this open connection.
2. Open a connection just before executing a query and close it immediately once the query execution is over.

As you might have guessed, the former way is better as far as performance is concerned. However, it suffers from a serious limitation. Because one connection is held open for a large window, in terms of scalability it is extremely poor. In multiuser scenarios, such as web sites, this will restrict the total number of users who can access your application.

The second way is safe and follows the philosophy—"Open the connection as late as possible and close the connection as early as possible." Although this approach is good in terms of scalability and multiuser scenarios, it has its own disadvantage. The frequent creation, opening, closing, and destroying of the database connections results in performance penalty. This is precisely where database connection pooling come handy.

A database connection pool is a set of database connections that are held open with the database and are kept ready to serve. This way, a set of connections is created once the pool comes into existence. When a user requests a connection, an already created connection is served from the pool. Similarly, when a user closes a connection, it is returned to the pool instead of being destroyed. This can significantly improve the performance of data access.

5.7.1 Connection Pooling in ADO.NET

In ADO.NET, database connection pooling is enabled by default. You can configure various aspects of connection pooling via a connection string. The following Table 5.1 lists some of the important parameters related to connection pooling:

Table 5.1: Parameters Related to Connection Pooling

Connection String Attributes	Description	Default Value
Pooling	Decides whether connection pooling will be enabled. Possible values are true or false.	True
Min Pool Size	Governs the minimum number of connections that are maintained in the pool.	0
Max Pool Size	Governs the maximum number of connections that a pool can contain.	100
Connection Timeout	Indicates a timeout value in seconds for which the data provider tries to connect with the database. Once this time is elapsed, an exception is thrown.	15 seconds

Notes

A connection pool gets created when the first connection is opened with the database.

Some Rules

To get the most out of database connection pooling, it is important to remember the following rules:

1. If you want to pool a set of connections, all of them must have the same connection string.
2. All the connections must have the same security context. If you are using Windows integrated security, the Windows user responsible for starting the application process is used to determine the security context.
3. All the connections must belong to the same process.

5.7.2 Testing Connection Pooling Behavior

Now that you have some idea about what connection pooling is, you can develop a simple application to verify the learned facts.

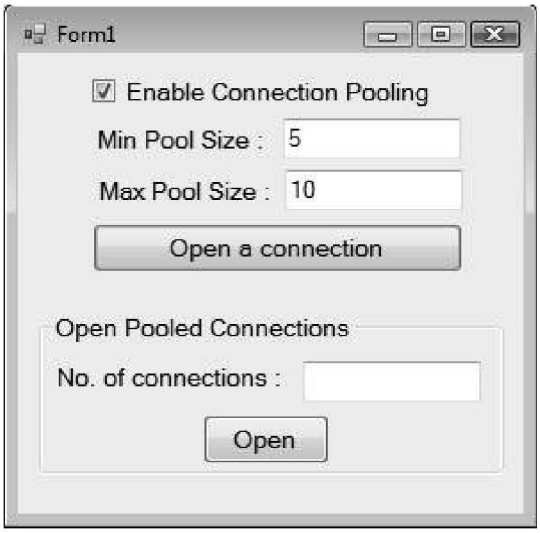
Begin by creating a new Windows Application and design its default form as shown Figure 5.4:

The form contains a checkbox control that governs whether or not connection pooling is enabled. If enabled, you also can specify the minimum pool size and maximum pool size. Clicking on the "Open a connection" button opens a database connection so that a connection pool is created. To test the behavior of MaxPoolSize attribute, you open multiple connections with the database at a time. The number of connections to open can be specified in the relevant textbox.

Now, go in the code window of the form class and add a helper method named GetConnectionString() as shown below:

```
private string GetConnectionString()
{
SqlConnectionStringBuilder builder =
new SqlConnectionStringBuilder();
builder.DataSource = "localhost";
builder.InitialCatalog = "northwind";
```

Figure 5.4: The Default Form



Notes

```
if (checkBox1.Checked)
{
builder.Pooling = true;
builder.MinPoolSize = int.Parse(textBox2.Text);
builder.MaxPoolSize = int.Parse(textBox3.Text);
}
else
{
builder.Pooling = false;
}
builder.UserID = "sa";
builder.Password = "pass@word";
return builder.ConnectionString;
}
```

Case Study

Epicor Software Corporation

Epicor is a global software company that creates enterprise resource planning (ERP) solutions for manufacturing firms and other customers. A growing number of Epicor customers wanted a more flexible version of Epicor 9, the company's flagship ERP solution. Epicor also sought to reduce the time it spent on framework development and wanted to lower the traffic to its Microsoft SQL Server database. The company took advantage of the Microsoft Visual Studio 2010 development system and the Microsoft .NET Framework 4, which includes a new version of the Microsoft ADO.NET Entity Framework. This software, which supports large data models, has greatly eased development for Epicor programmers and reduced SQL Server database requests by 90%. Epicor also saves time and money on development and can deliver products to its customers faster.

Situation

Based in Irvine, California, Epicor is a Microsoft Gold Certified Partner that develops innovative enterprise resource planning (ERP) solutions for small and midsize businesses and Global 1000 companies in more than 150 countries. Most of the organization's 20,000 customers work in the manufacturing distribution, hospitality, retail, and business services industries.

The organization's most widely used ERP solution is Epicor 9, a service-oriented architecture (SOA) product that is based on the Microsoft Visual Studio 2008 development system and the Microsoft .NET Framework. Epicor 9 also uses Microsoft SQL Server 2008 data management software to support its data repository. The product has separate client, business logic, and database layers.

The solution is designed to give employees the data they need to make the right business decisions. "Epicor 9 handles all the critical business processes in an organization, such as ordering, inventory, purchasing, financials, and human resources," says Erik Johnson, Vice President, Product Research, Epicor.

Recently, many Epicor customers expressed a need for additional functionality and flexibility in Epicor 9. For example, customers wanted to be able to perform database queries with

Contd...

Notes

compiler support, so if programmers misspelled a SQL Server table name, the language could still help them write good queries. "For that to work, you need a data model," says Johnson. "Until recently, our customers built their own data models. But there was no application programming interface for the database besides SQL Server itself."

This was problematic for many customers. "People were asking us how to get their own code to mirror the behavior of Epicor 9 with regard to updates," says Johnson. "Their programmers wanted to do things the same way as our application code did things. It became clear to us that we needed to give our customers a certified data model they could use."

Also, Epicor 9 uses Language-Integrated Query (LINQ)—a Microsoft .NET component that adds native querying functionality to .NET languages—which Epicor has provided to its customers for their SQL Server databases. "We do a lot of framework development, but our objective is to avoid customizing the technologies we work with," Johnson says. "We did not want to have to be LINQ providers."

Additionally, Epicor wanted to help reduce traffic to the SQL Server database. "We saw that for each line item of a sale, for example, customers want to verify the department or shipping date, and many of those database queries are the same for each of those items," says Johnson. "It is difficult to do those queries once across all line items, so there was a lot of traffic to SQL Server and that affected scalability."

Epicor also sought to simplify development for its team of 400 developers. "During development, we often had to work with the same table in multiple data models, which was confusing because our developers often have to keep track of which model has changes in it during updates," Johnson says.

Solution

In early 2009, Epicor started searching for a new .NET data access API that its customers could use. However, the company could not find a high-performing solution that would support the large data model size needed for Epicor 9. "Epicor 9 has 1,600 SQL Server tables," says Johnson. "We looked at solutions that just could not handle that database size, and we did not think we could change the solution behavior to fit our needs."

The company considered creating its own solution. However, Epicor changed its mind when it learned about the features in Microsoft Visual Studio 2010 and the Microsoft .NET Framework 4.In particular, the .NET Framework 4 includes the Microsoft ADO.NET Entity Framework 4, which helps developers create data access applications by programming against a conceptual application model. With the ADO.NET Entity Framework 4, developers can reduce the amount of code and maintenance needed for data-oriented applications. The technology also features enhanced performance and better support for large data model sizes.

The Entity Framework 4 includes templates for code generation based on the T4 code-generation technology in Visual Studio 2010. Developers can customize these templates to provide richer functionality, such as implementing custom interfaces.

Also new in the .NET Framework 4 is a programming model for writing multithreaded code, which can simplify application development. Developers can use this model to create scalable parallel code without the need to work with threads or a thread pool. In addition, the .NET Framework 4 provides enhancements in Windows Communication Foundation, such as a simplified configuration model and the inclusion of a routing service with which developers can create a single up-front layer for accessing back-end services.

Epicor implemented beta versions of Visual Studio 2010 and the .NET Framework 4 in late 2009. The company used these products to develop Epicor 9.05, the latest version of its flagship ERP solution.

Contd...

Notes

Benefits

By using Microsoft Visual Studio 2010 and the Microsoft .NET Framework 4, Epicor has support for its large data models and created a new version of its flagship product. The company has also greatly simplified development and reduced traffic to SQL Server by 90%. Additionally, Epicor has lowered its overall development costs and can bring products and features to customers more rapidly.

Supports Large Data Models

Because the ADO.NET Entity Framework 4 can support large data model sizes, Epicor could use the technology to create Epicor 9.05. "The earlier version of the Entity Framework did not handle our data model size very well, and we experienced delays in testing," says Johnson. "Those problems have all been solved with the ADO.NET Entity Framework 4."

As a result, Epicor did not have to create its own in-house data access solution. "Initially, we planned to write our own data access layer because of our data model size and the unique behavior of our application, which we needed to replicate in our data access tier," says Johnson. "The ADO.NET Entity Framework 4 gave us the data model size support and performance we needed, so we did not have to spend time and effort writing our own solution."

Simplifies Development

With Visual Studio 2010 and the .NET Framework 4, Epicor developers took advantage of the flexibility of the T4 code-generation templates when creating Epicor 9.05. "The ADO.NET Entity Framework 4 separates the data model from the code that it generates," says Johnson. "The T4 templates drive what the code looks like, and that made it easier for us to edit those templates so the code turned out exactly the way we wanted it to look."

Epicor developers also made use of the new parallel programming model in the .NET Framework 4 and the new features in Windows Communication Foundation to further simplify development of Epicor 9.05. "Using parallel extensions in the .NET Framework 4, we can do batch processing on the database by reading records and then creating tasks that can be handled in the background," says Johnson. "Our developers no longer have to create and manage threads to do that."

With the simplified configuration made possible by the new version of Windows Communication Foundation, developers can also create Web services more easily. "We have about 1,000 Web services in Epicor 9, and our developers used to create 1,000 entries in a configuration file to create the endpoints," says Johnson. "With Windows Communication Foundation in the .NET Framework 4, only one entry is required for all of those Web services in Epicor 9.05."

Reduces Server Requests by 90%

By using the new T4 code-generation templates, Epicor could modify templates and build a new API into its solution. "We combined the idea of compiling queries and caching query results because of the template flexibility," says Johnson. "Now, we can look in the ADO.NET Entity Framework 4 object state manager to check if a row was already there, or if it was already retrieved using the exact same query with the same parameters."

With these new streamlined development processes, Epicor has reduced the load on SQL Server and increased the overall application responsiveness. "We conducted some performance tests recently with existing code, and we were able to reduce the requests to SQL Server by 90%," says Johnson. "That is because of the ADO.NET Entity Framework 4."

Contd...

Notes

Saves Time and Money

When Epicor realized it did not have to write its own new data access layer, the company knew that it would be able to reduce development costs. "Because of the ADO.NET Entity Framework 4, we no longer have to be in the LINQ provider business," says Johnson. "We can offload framework development, and as a result we do not spend time and money training our developers."

Epicor has also experienced strong compiled query performance, which will lead to further cost savings. "We noticed immediately that the ADO.NET Entity Framework 4 is extremely fast at row materialization," says Johnson. "Overall, the per-row development cost was lower in the Entity Framework 4 than in all of the other technologies we evaluated."

Questions

1. What was the main problem in the company?

2. How the problem was solved and what benefits are found using the ADO.NET?

Self Assessment Questions

6. The term OleDb stands for:
   (*a*) Object Linking and Embedding DataBinding
   (*b*) Object Lightening and Embedding DataBase
   (*c*) Object Linking and Embedding DataBase
   (*d*) None of these.
7. The ODBC stands for:
   (*a*) Object Linking and Embedding DataBinding
   (*b*) Open DataBase Connectivity
   (*c*) Object DataBase Connectivity (*d*) None of these.
8. In .NET, connected approach is accomplished using the …… objects.
   (*a*) connection (*b*) command
   (*c*) datareader (*d*) All of these.
9. Which object is not used in connected approach?
   (*a*) Connection (*b*) Command
   (*c*) DataAdapter (*d*) DataReader
10. Which query will return the address of the cursor?
    (*a*) ExecuteXmlReader() (*b*) ExecuteScalar()
    (*c*) ExecuteNonQuery() (*d*) ExecuteReader()
11. Which query will return the number of rows affected?
    (*a*) ExecuteXmlReader() (*b*) ExecuteScalar()
    (*c*) ExecuteNonQuery() (*d*) ExecuteReader()
12. Which query will return the first [row][column] of the result?
    (*a*) ExecuteXmlReader() (*b*) ExecuteScalar()
    (*c*) ExecuteNonQuery() (*d*) ExecuteReader()

True or False

Notes

13. A DataSet is the equivalent for client side cursor, which holds a portion of the database after execution of a query.

    (*a*) True (*b*) False

14. The ADO.NET classes are found in System.Data.dll and integrated with XML classes in System.Xml.dll.

    (a) True (*b*) False

15. In ADO.NET, database connection pooling is enabled by default.

    (*a*) True (*b*) False

5.8 Summary

- The software support provided by a database vendor to access its own product is known as a 'Native driver'.
- A database is a collection of information that is organized so that it can easily be accessed, managed, and updated.
- A distributed database is one that can be dispersed or replicated among different points in a network.
- An object-oriented programming database is one that is congruent with the data defined in object classes and subclasses.
- The ADO.NET provides data access services in the Microsoft .NET Platform.
- The ADO.NET classes are found in System.Data.dll and integrated with XML classes in System.Xml.dll.
- DataSet is a copy of a extracted data being downloaded and cached in the client system.
- The Windows Forms DataGrid control displays data in a series of rows and columns. The Windows Forms DataGrid control provides a user interface to ADO.NET datasets.
- A DataSet can be saved to an XML file and then loaded back into memory very easily.

5.9 Keywords

ADO.NET: The ADO.NET is the new database technology of the .NET (DotNet) platform, and it builds on ADO (ActiveX Data Objects).

DataAdapter: A DataAdapter acts as a bridge between a data source and a data class which is disconnected from the data source.

Database Connection Pool: A database connection pool is a set of database connections that are held open with the database and are kept ready to serve.

DataBinding: It is the ability to bind some elements of a data source with some graphical elements of an application.

DataSet: A DataSet is the equivalent for client side cursor, which holds a portion of the database after execution of a query.

Lab Exercise

1. Write a program to connect a database with the frontend.
2. Write a program to select and display the records from the table.

Notes

5.10 Review Questions

1. What is the process to access database using ADO.NET?
2. Explain the database connection in ADO .NET.
3. Define database command in ADO .NET.
4. Explain the ADO .NET DataAdapter.
5. Define the ADO .NET DataSet.
6. Explain the use of ADO .NET DataReader
7. What is the connection pooling?
8. Briefly Explain the ADO.NET object model.
9. What is the role of CommandBuilder?
10. Explain the single-table fetch approach.

Answers for Self Assessment Questions

1. (*b*)	2. (*c*)	3. (*a*)	4. (*d*)	5. (*a*)
6. (*c*)	7. (*b*)	8. (*d*)	9. (*c*)	10. (*d*)
11. (*c*)	12. (*b*)	13. (*a*)	14. (*a*)	15. (*a*)

5.11 Further Readings


Books

Practical ASP. NET 3.5 Projects for Beginners, by B.M. Harwani

ASP.NET, by Yashwanth Kanethkar


Online link

http://www.w3schools.com/aspnet/default.asp

Unit 6: Error Handling

Notes

CONTENTS

Objectives

After studying this unit, you will be able to:

- Define the logics of error handling
- Describe all aspects of error handling
- Explain the levels of error handling
- Define the web application error handling in ASP.NET
- Define error handling in ASP.NET web API

Notes

Introduction

One of the main characters of ASP.NET over ASP is its new error handling features. The only way to imprison errors in the ASP/VBScript was using the "On Error Resume Next" declaration and scrutiny each line for an error with "If Err. Number <> 0 Then "statements. Developers who were using JScript instead of the VBScript at the server-side where lucky. They have had access to the "Try. Catch" statements provided by JScript. Well it is all over now. Now all the .NET enabled languages support better error handling statements including VB.NET. Now, VB.NET supports the "Try...Catch" statement. On top of this ASP.NET configuration file also supports ASP.NET level error handling.

We always to create a problem-free function, one that always behaves as expected. In reality, many things can go wrong while a program is running. As an application developer, you can anticipate as much bad behaviour as possible so you can take appropriate actions. To assist you with dealing with errors, the Visual Basic language provides many keywords, operators, and techniques.

6.1 Logics of Error Handling

When an unhandled immunity propagates, the user may well be redirect to an error sheet using different ASP.NET constitution settings. However, such a redirection may be prohibited in the first place by treatment the exceptions that get thrown. Error handling in ASP.NET therefore, may be divided into two separate logics:

- Redirecting the user to an error page when errors go unhandled.
- Handling exceptions when they get thrown.

6.1.1 Redirecting the User to an Error Page

There are two different scopes where we could specify which page the user has to be redirected to, when errors go unhandled:

- Page level (applies to errors that happen within a single page).
- Application level (applies to errors that happen anywhere in the application).

Page Level

Use the error Page attribute in the web form.

This attribute defines the page the user should be redirected to when an unhandled exception occurs in that specific.

The error Page attributes maps to the Page. Error Page property, and hence may be set programmatically. The value may optionally include query string parameters. If no parameters are added, ASP.NET would automatically add one with the name aspx error path. This parameter would hold the value of the relative URL to this page, so that the error page would be able to determine which page caused the error.

If a value is specified in this attribute (or property) and an unhandled exception occurs in the page, the Page class would automatically perform a redirect to the specified page. If a value is not specified, the exception is assumed to be unhandled, wrapped in a new Http Unhandled Exception and then thrown, propagating it to the next higher level.

Application Level

Use the custom Errors section in web.config.

Notes

This section lets you specify the error page to which the user should be redirected to when an unhandled exception propagates in the application level. This section specifies error pages for both default errors as well as the HTTP status code errors.

```
<customErrors mode= “On” defaultRedirect= “/WebTest/ErrorPages/AppError.html”>
<error statusCode= “404” redirect= “/WebTest/ErrorPages/404.html” />
</customErrors>
```

6.1.2 Custom Error Handling in ASP.NET

To customize the default error page, one will have to change the default configuration settings of the application. There are three error modes in which an ASP.Net application can work:

- Off Mode
- On Mode
- Remote Only Mode

The Error mode attribute determines whether or not an ASP.Net error message is displayed. By default, the mode value is set to “Remote only”.

Off Mode

When the error attribute is set to “Off”, ASP.Net uses its default error page for both local and remote users in case of an error.

On Mode

In case of “On” Mode, ASP.Net uses user-defined custom error page instead of its default error page for both local and remote users. If a custom error page is not specified, ASP.Net shows the error page describing how to enable remote viewing of errors.

Remote Only

ASP.Net error page is shown only to local users. Remote requests will first check the configuration settings for the custom error page or finally show an IIS error.

```
<customErrors mode= “On” defaultRedirect= “/WebTest/ErrorPages/AppError.html”>
<error statusCode= “404” redirect= “/WebTest/ErrorPages/404.html” />
</customErrors>
```

The mode attribute specifies whether to show user-defined custom error pages or ASP.NET error pages. Three values are supported for this attribute:

- *Remote Only:* Custom error pages are shown for all remote users. ASP.NET error pages with rich error information are displayed only for local users.
- *On:* Custom error pages are always shown, unless one is not specified. When a custom error page is not defined, an ASP.NET error page will be displayed which describes how to enable remote viewing of errors.
- *Off*: Custom error pages are not shown. Instead, ASP.NET error pages will be displayed always, which will have rich error information.

It is a bad idea to give users more in sequence than what is required. ASP.NET error pages describe technical details that should not be exposed. Ideally, the mode attribute thus should not be set to off.

Notes

The default Redirect attribute specifies the path to a generic error page.

Each error element defines a redirect specific to a particular HTTP status code. For example, if the error is a 404 (File Not Found), then you could set the error page as *FileNotFound.htm*. You could add as many error elements in the custom Errors section as required, each of which specifies a status code and the corresponding error page path. If ASP.NET cannot find any specific error element corresponding to a status code, it would use the value specified in the default Redirect attribute.

Task Write a program to handle an exception using try-cache block.

6.2 Aspects of Error Handling

Following aspects are defined fir error handling:

Tracing: tracing the program execution at page level or application level.

Error handling: handling standard errors or custom errors at page level or application level.

Debugging: stepping through the program, setting break points to analyze the code.

To understand the concepts, create the following sample application. It has a label control, a dropdown list and a link. The dropdown list loads an array list of famous quotes and the selected quote is shown in the label below. It also has a hyperlink which has a nonexistent link.

```
<%@ Page Language= "C#"
AutoEventWireup= "true"
CodeBehind= "Default.aspx.cs"
 Inherits= "errorhandling._Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.
dtd">
<html xmlns= "http://www.w3.org/1999/xhtml" >
<head runat= "server">
<title>Tracing, debugging and error handling</title>
</head>
<body>
<form id= "form1" runat= "server">
 <div>
<asp:Label ID= "lblheading" runat= "server"
Text= "Tracing, Debugging and Error Handling">
</asp:Label>
<br />
 <br />
<asp:DropDownList ID= "ddlquotes"
 runat= "server" AutoPostBack= "True"
  onselectedindexchanged= "ddlquotes_
SelectedIndexChanged">
```

Notes

```
</asp:DropDownList>
 <br />
<br />
<asp:Label ID= "lblquotes" runat= "server">
</asp:Label>
 <br />
<br />
<asp:HyperLink ID= "HyperLink1" runat= "server"
NavigateUrl= "mylink.htm">Link to:</asp:HyperLink>
 </div>
</form>
</body>
</html>
```

The Code behind File

```
public partial class _Default : System.Web.UI.Page
{
protected void Page_Load(object sender, EventArgs e)
{
 if (!IsPostBack)
{
string[,] quotes =
{
{ "Imagination is more important than Knowledge.",
 "Albert Einsten"},
{ "Assume a virtue, if you have it not",
 "Shakespeare"},
{ "A man cannot be comfortable without his own
approval", "Mark Twain"},
{ "Beware the young doctor and the old barber",
 "Benjamin Franklin"},
{ "Whatever begun in anger ends in shame",
 "Benjamin Franklin"}
};
for (int i=0; i<quotes.GetLength(0); i++)
ddlquotes.Items.Add(new ListItem(quotes[i,0],
quotes[i,1]));
}
}
protected void ddlquotes_SelectedIndexChanged(object
sender,
EventArgs e)
{
```

Notes

```
if (ddlquotes.SelectedIndex != -1)
{
lblquotes.Text = String.Format( "{0}, Quote: {1}",
ddlquotes.SelectedItem.Text, ddlquotes.SelectedValue);
}
}
}
```

Tracing

To enable page level tracing, you need to modify the Page directive and add a Trace attribute as:

<%@ Page Language= "C#" AutoEventWireup= "true"
CodeBehind= "Default.aspx.cs"
Inherits= "errorhandling._Default"

Trace = "true" %>Now when you run the file, you get the tracing information:

It provides the following information at the top:

- Session ID
- Status Code
- Time of Request
- Type of Request
- Request and Response Encoding

The status code sent from the server, each time the page is requested shows the name and time of error if any. The following table shows the common HTTP status codes:(See Table 6.1)

Table 6.1: Common HTTP Status Codes

Number	Description
Informational (100 - 199)	
100	Continue
101	Switching protocols
Successful (200 - 299)	
200	OK
204	No content
Redirection (300 - 399)	
301	**Moved permanently**
305	Use proxy
307	Temporary redirect
Client Errors (400 - 499)	
400	Bad request
402	Payment required
404	Not found
408	Request timeout
417	Expectation failed
Server Errors (500 - 599)	
500	Internal server error
503	Service unavailable
505	HTTP version not supported

Under the top level information is the Trace log, which provides details of page life cycle. It provides elapsed time in seconds since the page was initialized.(See Figure 6.1)

Notes

Figure 6.1: Trace Information

Trace Information

Category	
aspx.page	Begin PreInit
aspx.page	End PreInit
aspx.page	Begin Init
aspx.page	End Init
aspx.page	Begin InitComplete
aspx.page	End InitComplet
aspx.page	Begin LoadState
aspx.page	End LoadState
aspx.page	Begin ProcessPostData
aspx.page	End ProcessPostData
aspx.page	Begin PreLoad
aspx.page	End PreLoad
aspx.page	Begin Load
aspx.page	End Load
aspx.page	Begin ProcessPostData Second Try
aspx.page	End ProsessPostData Second Try
aspx.page	Begin Raise ChangedEvents
aspx.page	End Raise ChangedEvents

The control tree, which lists all controls on the page in a hierarchical manner:

Figure 6.2: Control tree

Control Tree

Control UniqueID	Type	Render size
__Page	ASP.default_aspx	2850
ctl02	System.Web.UI.LiteralControl	175
ctl00	System.Web.UI.HtmlControls.HtmlHead	70
ctl01	System.Web.UI.HtmlControls.HtmlTitle	57
ctl03	System.Web.UI.LiteralControl	14
form1	System.Web.UI.HtmlControls.HtmlForm	2571
ctl04	System.Web.UI.LiteralControl	27
lblheading	System.Web.UI.WebControl.Label	65
ctl05	System.Web.UI.LiteralControl	42
ddlquotes	System.Web.UI.WebControl.DropDownlist	570
ctl06	System.Web.UI.LiteralControl	42
lblquotes	System.Web.UI.WebControl.Label	96
ctl07	System.Web.UI.LiteralControl	42
HyperLink1	System.Web.UI.WebControl.HyperLink	49
ctl08	System.Web.UI.LiteralControl	24
ctl09	System.Web.UI.LiteralControl	20

Last in the Session and Application state summaries, cookies and headers collections, followed by list of all server variables.

The Trace object allows you to add custom information to the trace output. It has two methods to accomplish this: the Write method and the Warn method.

Change the Page_Load event handler to check the Write method:

```
protected void Page_Load(object sender, EventArgs e)
        {
        Trace.Write( “Page Load”);
        if (!IsPostBack)
        {
        Trace.Write( “Not Post Back, Page Load”);
        string[,] quotes
```

Notes

Run to observe the effects:

aspx.page	Begin PreLoad
aspx.page	End PreLoad
aspx.page	Begin Load
	Page Load
	Not Post Back, Page Load
aspx.page	End Load
aspx.page	Begin LoadComplete
aspx.page	End LoadComplet

To check the Warn method, let us forcibly enter some erroneous code in the selected index changed event handler:

```
try
{
int a = 0;
int b = 9 / a;
}
catch (Exception e)
{
Trace.Warn( "UserAction", "processing 9/a", e);
}
```

Try-Catch is a C# programming construct. The try block holds any code that may or may not produce error and the catch block catches the error. When the program is run, it sends the warning in the trace log.

Aspx.page Begin Raise changedEvents

Processing 9/a

Attempted to divide by zero

UserAction

At errorhandling_Default.ddlquotes_SelectedIndexChanged(Object sender,EventArgs e) in)

Application level tracing applies to all the pages in the web site. It is implemented by putting the following code lines in the web.config file:

```
<system.web>
<trace enabled= "true" />
</system.web>Error Handling
```

Although ASP.Net can detect all runtime errors, still some subtle errors may still be there. Observing the errors by tracing is meant for the developers, not for the users.

Hence, to intercept such occurrence, you can add error handing settings in the web.config file of the application. It is application wide error handling. For example, you can add the following lines in the web.config file:

```
<configuration>
<system.web>
<customErrors mode= "RemoteOnly"
defaultRedirect= "GenericErrorPage.htm">
<error statusCode= "403" redirect= "NoAccess.htm" />
<error statusCode= "404" redirect= "FileNotFound.htm" />
```

Notes

```
</customErrors>
</system.web>
<configuration>
```

The <customErrors> section has the possible attributes:

Mode:

It enables or disables custom error pages. It has the three possible values:

On: displays the custom pages.

Off: displays ASP.NET error pages (yellow pages).

Remote Only: displays custom errors to client, display ASP.NET errors locally.

DefaultRedirect:

It contains the URL of the page to be displayed in case of unhandled errors.

To put different custom error pages for different type of errors, the <error> sub tags are used, where different error pages are specified, based of the status code of the errors.

To implement page level error handling, the Page directive could be modified:

```
<%@ Page Language= "C#" AutoEventWireup= "true"
CodeBehind= "Default.aspx.cs"
Inherits= "errorhandling._Default"
Trace = "true"
ErrorPage= "PageError.htm" %>
```

6.3 Levels of Error Handling

There are different levels where you could handle exceptions.

- Locally (method level), where exceptions could be thrown.
- Page level by handling thePage.Error event.
- Application level by handling the HttpApplication.Error event.
- HTTP Module level by handling the HttpApplication.Error event.

6.3.1 Local Error Handling

Drape code that might fling exceptions in a try-catch-finally block. If you can get better from the exemption, then handle it in the catch block. If the exception cannot be improved from locally, let the exception propagate to higher levels by throwing it. If the exception cannot be recovered from locally, but additional information can be provided, then wrap the exception with the new information and throw the new exception. This method is used when you use custom exceptions. Place the clean up code in the finally block.

6.3.2 Page Level

Attach a handler to the Page.Error event. In C#, you will have to write the event wire up code yourself in the Page Load method.

When an exception goes unhandled in a page, the Error event of the Page group of pupils gets triggered.

Notes

Typically, the first action you would achieve in this handler would be to obtain the exemption thrown, by using theServer.GetLastError method. This method would return a reference to the last Exception object that was thrown.

After you get the exemption object, you will want to readdress the user to a mistake page. We could make ASP.NET do the redirection by using the error Page attribute of the Page (design time) or by using the Page.ErrorPageproperty (runtime). Obviously, the choice here would be to programmatically set the value using thePage.ErrorPage property in the event handler.

If you do not specify an error page, the exception gets wrapped inside an HttpUnhandledException object and propagates. If you do not want the exception to be wrapped, then simply throw the last exception, which would force immediate propagation escaping any intervention. However, this would prevent ASP.NET from redirecting the user to a page specific page either. In other words, if you are going to throw the last error (or any exception for that matter), setting the error page will have no effect.

```
private void WebForm1_Error(object sender, EventArgs e)
{
// Get the last exception thrown
Exception ex = Server.GetLastError();

// Do something with the exception like logging etc.
// Set the error page
this.ErrorPage = "/ErrorHandling/ErrorPages/BaseError.html";
}
```

To reduce redundant code, you could define a base web form page which defines the Page.Error event handler and then wire up code in the constructor, and then make all your Web Form pages derive from this base page. This would save you the effort of writing the error handler in each web form.

6.3.3 Application Level

Attach an event handler to the Application.Error event.

When an unhandled exception leaves a page, it gets propagated to the application level, which would trigger this event.

There are two things you would want to do in an application error handler.

- Get the last exception thrown using Server.GetLastError.
- Clear the error using Server.ClearError, to inform ASP.NET that you have handled the error.

However, since there is not any higher scope where the exception could be caught, ASP.NET is forced to handle it. The way ASP.NET handles the exception depends upon the settings specified in the customErrors section we saw before. If no settings are defined, ASP.NET would use the defaults and display the infamous 'yellow' error page.


Caution

If you do not clear the error, the exception would propagate. It becomes difficult to handle the exception.

6.3.4 HTTP Module Level

As an alternative of handling application errors in worldwide.asax, exceptions may also be handled by attaching an HTTP Module which would have a handler attached to the submission

Notes

Error event. This method would be triggered previous to the corresponding application handler would be invoked. Such an implementation would be beneficial if you have multiple projects with the same global error handling implementation. In such a scenario, you could create a module and attach it to each web application you have.


Did u know?

In 1996 the Ariane V rocket exploded due in part to the Ada programming language exception handling policy of aborting computation on arithmetic error - a floating point to integer conversion overflow - which would not have occurred if IEEE 754 exception-handling policy had been used.

Self Assessment Questions

Multiple Choice Questions

True or False

1. The Visual Basic language provides many keywords.

   (*a*) True (*b*) False

2. ASP.NET would not automatically add one with the name aspx error path.

   (*a*) True (*b*) False

3. ASP.Net uses its default error page for both local and remote users in .....

   (*a*) Page level (*b*) Execute

   (*c*) case of an error (*d*) Debugging

4. ..... execution at page level or application level.

   (*a*) Transfer Protocol (*b*) Application level

   (*c*) error (*d*) Tracing the program

5. Trace object allows you to add custom information to the .....

   (*a*) trace output. (*b*) Compile error

   (*c*) Runtime (*d*) Logic error

6.4 Web Application Error Handling in ASP.NET

All applications have to error handling. This we all know. We cannot always be notified of an unhandled error (and usually are not) when one occurs on a client's machine. The benefit we have on the Web is that we can always be notified when an unhandled error occurs. With the advent of ASP.NET, there are various enormous new ways to handle errors. There are some differences in .NET in not only how to handle the error, but how the information is provided to you. For example, classic ASP uses Server.GetLastError to return an ASPError object. You can and should still use Server.GetLastError in .NET, but this now returns a type System.Exception.

6.4.1 The Problem

Errors will occur in our applications. We try to trap for most errors using try-catch blocks; however, we usually do not cover every possible exception. What happens when an unhandled error occurs? Usually the user is brought to IIS's default error pages (usually located in c:\winnt\help\iishelp\common). The downsides are you have no idea when this occurs and the page does not have your site's look and feel.

Errors are a development fact, but we strive to eliminate or handle them gracefully. With this in mind, we need to know:

Notes

1. When an error occurs
2. Where it occurred
3. What the error is

Having a central location such as the event log, database or some other log file to log errors is essential for debugging this problem later.

IIS provides great error-handling capabilities. There are some problems with these though. Sometimes we know an error will occur, and we cannot always trap for it in a nice way without overriding the site's default error redirection page. For example, upon access to a resource that requires authentication, we may need to redirect to an application's login page. Also, a very common problem exists with Web hosts. If you have a hosted Web site, you usually have no control over its IIS configuration. Thus, setting up custom error pages can be next to impossible in traditional ASP. This is eliminated with ASP.NET, as you will learn as you read on.

6.4.2 The Solution

For such a list of problems, the solution is actually pretty simple. There are three places in ASP.NET to define what happens to these unhandled errors.

1. In the web.config file's customErrors section.
2. In the global.asax file's Application_Error sub.
3. On the aspx or associated codebehind page in the Page_Error sub.

The actual order of error handling events is as follows:

1. On the Page itself, in the Page_Error sub (this is default, you can name it anything because it specificed Handles MyBase.Error)
2. The global.asax Application_Error sub
3. The web.config file

When an exception occurs in your application, it should be an object inherited from type System. Exception, and as such will have the following public members:

Table 6.2: An Object Inherited From Type System. Exception

HelpLink	Gets or sets a link to the help file associated with this exception.
InnerException	Gets the Exception instance that caused the current exception.
Message	Gets a message that describes the current exception
Source	Gets or sets the name of the application or the object that causes the error.
StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown.
TargetSite	Gets the method that throws the current exception.

6.4.3 Using the Page Error or on Error Sub

The first line of defense in error handling happens at the page level. You can override the MyBase.Error sub as such: (Visual Studio will complete the code if you click either the Overrides or Base Class events in the editor). The two functions you can use (one or the other, both will not work as only one will get called)

Private Sub Page_Error(ByVal sender As Object, ByVal e As System.EventArgs) **Notes**

Handles MyBase.Error

End Sub

Or you can use this one:

Protected Overrides Sub OnError(ByVal e As System.EventArgs)

End Sub

Handling errors in these subs is a simple process. Just call Server.GetLastError to return the error. If you want to redirect to a specific page here you can just call Response.Redirect ("HandleError. aspx") or whatever your page may be. This method of handling errors is good for several reasons.

1. If you need to override the Application_Error or the customErrors setup in the web.config file.
2. If each page must implement its own error handling. If you need to log specific information and then carry on, just code for your logging or whatever here, and that is all. If you need to cancel the error processing here (so it does not go to the Application_Error or customErrors) simply call Server.ClearError in this sub.

6.4.4 Using the Global.Asax File

The global.asax file contains the next line of defense against errors. When an error occurs, the Application_Error sub is called. This location happens to be my favorite place to log errors because it is the most functional. For most of my applications in .NET, we do not handle too many custom errors at the page level. The only two locations that actually give you access to Server.GetLastError are in the Page_Error and Application_Error subs.

After the Page_Error is called, the Application_Error sub is called. Here you can also log the error and redirect to another page. We will not explain anything else about it because it is basically the same as the Page_Error but happens to be at the application level rather than the page level.

6.4.5 Using the Web.Config File

The customErrors element of the Web.config file is the last line of defense against an unhandled error. If you have other error handlers in place, like the Application_Error of Page_Error subs, these will get called first. Provided they do not do a Response.Redirect or a Server.ClearError, you should be brought to the page(s) defined in the web.config. In the web.config file, you can handle specific error codes (500, 404, etc), or you can use one page to handle all errors. This is a major difference between this method and the others (although you can emulate this by doing various Response. Redirects using the other methods). Open up your web.config file. The customErrors section uses this format:

```
<customErrors defaultRedirect= "url" mode= "On|Off|RemoteOnly">
 <error statusCode= "status code" redirect= "url"/>
</customErrors>
```

Here is some important information about the "mode" attribute:

"*On*" specifies that custom errors are enabled. If no defaultRedirect is specified, users see a generic error.

"*Off*" specifies that custom errors are disabled. This allows display of detailed errors.

"*Remote Only*" specifies that custom errors are shown only to remote clients, and ASP.NET errors are shown to the local host. This is the default.

Notes

By default, the section looks like this when you create a Web application.

<customErrors mode= “Remote Only” />

This will show a generic page to users. To redirect it to one of your own pages, you would change it to this:

<customErrors mode= “On” defaultRedirect= “error.htm” />

Now all errors that occur will be brought to the error.htm page.

To handle specific errors, and redirect to the error page for everything else you can specify the error code you want specially handled like so:

<customErrors mode= “On” defaultRedirect= “error.htm”>
<error statusCode= “500” redirect= “error500.aspx?code=500”/>
<error statusCode= “404” redirect= “filenotfound.aspx”/>
<error statusCode= “403” redirect= “authorizationfailed.aspx”/>
</customErrors>

There is a problem here with this solution. Once the redirect is done, your error information is no longer available on the redirected page. This is because IIS (via the .net framework) performs a plain old GET request to the error page and does not do a “Server. Transfer” likes the built-in IIS error handling does.

The only information available to you at this time is the URL that caused this error to be raised. This is located on the query string as“aspxerrorpath”: http://localhost/ErrorHandling/error500.aspx?aspxerrorpath=/ErrorHandling/WebForm1.aspx. The only places this information is available is the two methods described above.

Another interesting point about the above customErrors element is that you can specify different error pages for different subdirectories.

For this example, let us say you have a directory named “Customers” off of your root directory that contains a branding specific for logged in customers but is not in its own application. As such you want to define a different set of pages for errors. Please note that these pages specified in the “redirect” attribute are relative to the “Customers” subdirectory and not the root path of your site. It also included a security rule which says only MYDOMAIN\Customers can access these files. You can define rules for these errors in the web.config file:

```
<Configuration>
<system.web>
</system.web>
<!-- Configuration for the “Customers” subdirectory.
-->
<location path= “Customers”>
<system.web>
<customErrors mode= “On” defaultRedirect= “error.htm”>
<error statusCode= “500” redirect= “CustomerError500.
aspx”/>
<error statusCode= “401”
redirect= “CustomerAccessDenied.aspx”/>
<error statusCode= “404”
redirect= “CustomerPageNotFound.htm”/>
<error statusCode= “403” redirect= “noaccessallowed.
htm”/>
</customErrors>
```

Notes

```
<Authorization>
<allow roles= "MYDOMAIN\Customers" />
<deny users= "*" />
</authorization>
</system.web>
</location>
```

6.5 Error Handling in ASP.NET Web API

Web API is a brand new scaffold that makes it easy to build HTTP military. As such, it provides several descriptions that make it simple to send back helpful and enlightening error messages in a variety of cases we will go over several of these capabilities.

Now that's out of the way, let us take a look at what a typical WebAPI error's HTTP content looks like:

```
{
 "Message": "No HTTP resource was found that matches the request
URI 'http: //localhost/Foo'.",
 "MessageDetail": "No type was found that matches the controller
named 'Foo'."
}
```

So the error is actually just a collection of key-value pairs that provide information about what went wrong. This collection of key-value pairs is then sent back to the client in whatever content type it asked for through HTTP content negotiation. In the example above, the content is actually JSON.

But if you had "text/xml" in your request's Accept header for example, you might get this response instead:

```
<Error>
<Message>No HTTP resource was found that matches the request URI
'http://localhost/Foo'.</Message>
<MessageDetail>No type was found that matches the controller named
'Foo'.</MessageDetail>
< /Error>
```

This simple format makes it easy for clients to understand what went wrong or to extract relevant error information for error logging or reporting.

6.5.1 HttpError

What you just saw above is actually an instance of an HttpError being serialized to the wire by Json.NET and DataContractSerializer respectively.

The HttpError type defines a consistent and extensible way for representing error information across the framework. Effectively, it is just a Dictionary<string, object> that provides some helpers for creating errors that contain error messages, exceptions, or invalid model states. HttpError defines the following public constructors:

```
public HttpError();
public HttpError(string message);
public HttpError(Exception exception, bool includeErrorDetail);
public HttpError(ModelStateDictionary modelState, bool includeErrorDetail);
```

Notes

6.5.2 HttpResponseException

Now you may be wondering: how do we return an error if the action returns a different type instead of HttpResponseMessage? This is where HttpResponseException comes in. HttpResponseException is an exception type defined for WebAPI that serves two purposes:

It allows you to return a specific HTTP response from actions with return types that are not HttpResponseMessage.

It allows you to short-circuit the WebAPI pipeline pretty much anywhere and return a response immediately.

Technically, HttpResponseException can be used for any kind of HTTP response, but it is especially useful in error cases. It allows us to implement actions like this:

```
public Person Get(int id)
{
if (!_contacts.ContainsKey(id))
{
throw new HttpResponseException(Request.CreateErrorResponse(Htt
pStatusCode.NotFound, String.Format( "Contact {0} not found.", id)));
}
return _contacts[id];
}
```

Task Write the code to http error handling.

6.5.3 Error Detail

You may have noticed the "includeErrorDetail" parameter earlier in this post on the HttpError constructor. This is because typically, servers want to send different error information depending on whether the client is just consuming the service or whether the client needs additional debugging information to understand what went wrong on the server. By default, WebAPI will not send error details to remote clients and will provide these additional error details to clients on the local machine. Returning to our first example, whereas a local client would see this:

```
{
 "Message": "No HTTP resource was found that matches the request
URI 'http: //localhost/Foo'.",
 "Message Detail": "No type was found that matches the controller
named 'Foo'."
}
```

A remote client would only see this:

```
{
 "Message": "No HTTP resource was found that matches the request
URI 'http: //localhost/Foo'."
}
```

The difference between Message and Message Detail above is that MessageDetail contains error information that is WebAPI-specific that remote clients should not have to see in most cases. The exact meaning of error detail depends on the case. For exceptions, error detail includes the

Notes

exception message, exception type, the stack trace, and the inner exceptions. Only a vague "An error has occurred." message is sent back to remote clients for exceptions by default. For model states, any model error messages are sent to remote clients. Model error exceptions, however, are considered detail and will not get sent to remote clients by default.

You can choose to override when the error detail gets sent back to clients directly on your HttpConfiguration object:

```
config.IncludeErrorDetailPolicy = IncludeErrorDetailPolicy.LocalOnly;
config.IncludeErrorDetailPolicy = IncludeErrorDetailPolicy.Always;
config.IncludeErrorDetailPolicy = IncludeErrorDetailPolicy.Never;\
```

6.5.4 Applying Error Handling to Handle Invalid Model States

One especially common use of WebAPI's error handling is to immediately return an error if the model state is invalid. This can be done fairly easily by implementing the following action filter:

```
public class ValidationFilterAttribute : ActionFilterAttribute
{
public override void OnActionExecuting(HttpActionContext
actionContext)
{
if (!actionContext.ModelState.IsValid)
{
actionContext.Response=actionContext.Request.CreateErrorResponse(
HttpStatusCode.BadRequest, actionContext.ModelState);
}
}
}
```


Caution

Attempting to use the JScript Error object in an ASP.NET page may product unintended results. This results from the potential ambiguity between the JScript Error object and the Error event of the ASP.NET page. Use the System. Exception class instead of the Error object for error handling in ASP.NET pages.


Case Study

ASP.NET 2.0 Crash Case: Unhandled Exceptions

Problem Description

Once in a while ASP.NET crashes and we see events in the system event log like this one

Event Type:	Warning
Event Source:	W3SVC
Event Category:	None
Event ID:	1009
Date:	2006-04-25
Time:	09:41:22
PM User:	N/A

Contd...

Notes

Computer: SUBSPACE1

Description:

A process serving application pool 'ASP.NET V2.0' terminated unexpectedly. The process id was '1732'. The process exit code was '0xe0434f4d'.

Or this one

Event Type: Warning

Event Source: W3SVC

Event Category: None

Event ID: 1011

Date: 2006-04-25

Time: 09:41:22

User: N/A

Computer: SUBSPACE1

Description:

A process serving application pool 'ASP.NET V2.0' suffered a fatal communication error with the World Wide Web Publishing Service. The process id was '6256'. The data field contains the error number.

And in the application event log we get a pretty cryptic error message like this one

Event Type: Error

Event Source: .NET Runtime 2.0 Error Reporting

Event Category: None

Event ID: 5000

Date: 2006-04-25

Time: 09:41:20

User: N/A

Computer: SUBSPACE1

Description:

EventType clr20r3, P1 w3wp.exe, P2 6.0.3790.1830, P3 42435be1, P4 app_code.pn5mfdcr, P5 0.0.0.0, P6 444dcf44, P7 5, P8 5, P9 system.dividebyzeroexception, P10 NIL.

Initial Thoughts

Now what do we know about the issue? We know that asp.net terminated unexpectedly, and that right before this we got a System.DivideByZeroException…. We also know that the process exit code was 0xe0434f4d.

Usually when you get a stopped unexpectedly error message the exit code will be the type of exception that caused the crash. For example a 0xC0000005 means you got a second chance access violation, 0x800703e9 means you suffered a StackOverflowException but what about 0xe0434f4d?

Contd...

Notes

0xe0434f4d is the exception code for CLR (.net) exceptions, so any managed exception like a NullReferenceException or InvalidOperationException or SQLException… basically all managed exception are natively referred to as 0xe0434f4d. In this case, if we look closer at the application event log entry we can see that it is in fact a System.DivideByZero exception.

Trivia: Just a piece of info of no particular value that you might want to pull out of pocket on your next dateJ 0xe0434f4d or at least 43 4f 4d are the ASCII values for the letters COM.

But now should a .net exception cause the asp.net process to crash??? If you divide by zero in your page and do not have a try catch block around it, surely you will get one of those "nice " white and yellow error pages saying that an exception occurred on your page, but the process does not just exit.

The answer is yes, you will get one of those pages because the asp.net global error handler will eventually catch your exception, format it for you and print it out on the screen. But what happens if it is not on an asp.net request, so there is no-one to feedback the exception to? It is the old paradox: If a tree falls in the forest and nobody is there, does it still make a sound?

In 1.0 and 1.1 it did not. For example if you throw an exception in a piece of code called on a timer, or use QueueUserWorkItem and throw an exception in code executing there, or otherwise throw exceptions in code that is not running inside the context of an asp.net request, the framework will swallow the exception and continue. Or rather it will stop that thread of execution but it would not die.

Does not sound all that bad right?

That thread could have been doing anything, and we will never be the wiser that it died. It could have been holding a lock of some sort, or it could have been in the middle of cleaning up resources, or really a number of different things that will now never happen, but that may immediately or eventually have really bad side effects like hangs or crashes or memory issues, but the exception will be long gone so we cannot figure out what it was.

The policy for unhandled exceptions was changed in ASP.NET 2.0 to the default for .net which is a process exit. This can be changed back by adding the following to the aspnet. config in the frameworks directory, but it would not recommend it without putting in some preventive measures to take care of potential unhandled exceptions on non ASP.NET threads.

```
<configuration>
    <runtime>
        <legacyUnhandledExceptionPolicy enabled= "true " />
    </runtime>
</configuration>
```

Questions

1. What is the problem of ASP.NET crash?

2. What were the initial thoughts about this exception?

Self Assessment Questions

6. Which is not a type of error programmers look for?

   (*a*) Logic (*b*) Runtime

   (*b*) Superficial (*d*) Syntax

Notes

7. Which action will raise an exception?
   (*a*) Dividing by zero
   (*b*) Assigning the string "Hi" to an integer variable.
   (*c*) Accessing an empty CD drive.
   (*d*) All of these.
8. An Exception is another name for a:
   (*a*) compile error (*b*) logic error
   (*c*) runtime error (*d*) superficial error
9. ..... applies to errors that happen within a single page.
   (*a*) Page level (*b*) Application level
   (*c*) Both (*a*) and (*b*) (*d*) None of these.
10. …… applies to errors that happen anywhere in the application.
   (*a*) Page level (*b*) Application level
   (*c*) Both (*a*) and (*b*) (*d*) None of these.
11. Which is not an error mode in an ASP.Net application……..?
   (*a*) on mode (*b*) off mode
   (*c*) remote mode (*d*) logic
12. HTTP stands for .....
   (*a*) Hypertext Transfer Protocol (*b*) Hypertool Transfer Protocol
   (*c*) Hypertext.Transport.Protocol (*d*) None of these.
13. Which one is not an aspects of error handling
   (*a*) tracing (*b*) error handling
   (*d*) debugging (*d*) All of these.
14. IIS stands for …….
   (*a*) International Information Services
   (*b*) Internet Information Services
   (*c*) International Integrated Services
   (*d*) None of these.
15. Web API is a brand new framework that makes it easy to build HTTP services.
   (*a*) True (*b*) False

6.6 Summary

- When the error attribute is set to "Off ", ASP.Net uses its default error page for both local and remote users in case of an error.
- Try-catch is a C# programming construct. The try block holds any code that may or may not produce error and the catch block catches the error.

Notes

- The custom Errors element of the Web.config file is the last line of defence against an unhandled error.
- The difference between Message and Message Detail above is that MessageDetail contains error information that is WebAPI-specific that remote clients should not have to see in most cases.
- Web API is a brand new framework that makes it easy to build HTTP services. As such, it provides several features that make it easy to send back useful and informative error messages in a variety of cases.

6.7 Keywords

ASP.NET: It is a Web application framework developed and marketed by Microsoft to allow programmers to build dynamic Web sites, Web applications and Web services.

Exception Handling: It is the process of responding to the occurrence, during computation, of exceptions anomalous or exceptional situations requiring special processing often changing the normal flow of program execution.

Hypertext Transfer Protocol: The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems.

Web API: A Web API (Application Programming Interface) is typically a defined set of HTTP request messages along with a definition of the structure of response messages, typically expressed in JSON or XML.

Web.Config: Web.config is the main settings and configuration file for an ASP.NET web application.

Lab Exercise

1. Create a Web.config file in Visualstudio.
2. Prepare a flow chart to handle an exception.

6.8 Review Questions

1. Define the logics of error handling.
2. What are the aspects of error handling?
3. Explain the levels of error handling.
4. Define the web application error handling in ASP.NET.
5. Explain error handling in ASP.NET web API.
6. Define error handling using the page error or on error sub.
7. What is the custom error handling in ASP.NET?
8. Define web config file.
9. Explain Http response exception.
10. Explain global.Asax File?

Answers for Self Assessment Questions

1.	(*a*)	2.	(*b*)	3.	(*c*)	4.	(*d*)	5.	(a)
6.	(*b*)	7.	(*d*)	8.	(*c*)	9.	(*a*)	10.	(*b*)
11.	(*d*)	12.	(*a*)	13.	(*d*)	14.	(*b*)	15.	(*a*)

Notes

6.9 Further Readings


Books

Robust ASP.Net Exception Handling, by Lee Dumond


Online link

http://books.google.co.in/books?id=g6q4Cd66MdoC&printsec=frontcover&dq=Error+Handling+in+asp.net&hl=en&sa=X&ei=kRcAUJaFMIe3rAeW4dGaBg&ved=0CDYQ6AEwAA#v=onepage&q=Error%20Handling%20in%20asp.net&f=false

Unit 7: Advanced ASP.NET

Notes

CONTENTS

Objectives

After studying this unit, you will be able to:

- Explain about communicate with browser
- Define web.configuration
- Define the characteristic of ASP.NET
- Explain the new features of ASP.NET

Introduction

ASP.NET is a set of Web development equipment offered by Microsoft. Programs like Visual Studio .NET and Visual Web Developer allow Web developers to produce dynamic websites by means of a visual interface. Of course, programmers can write their own code and scripts and incorporate it into ASP.NET websites as well. Though, it often seen as a successor to Microsoft's ASP programming technology, ASP.NET also supports Visual Basic .NET, Script .NET and open-source languages like Python and Perl.

ASP.NET is built on the .NET framework, which provides an application program interface (API) for software programmers. The .NET advance tools can be used to create applications for both the Windows operational system and the Web. Programs similar to Visual Studio .NET provide a visual interface for developers to create their applications, which makes .NET a reasonable choice for designing Web-based interfaces as well.

In order for an ASP.NET website to function correctly, it must be published to a Web server that supports ASP.NET applications. Microsoft's Internet Information Services (IIS) Web server

Notes

is by far the most common platform for ASP.NET websites. While there are some open-source options available for Linux-based systems, these alternatives often provide less than full support for ASP.NET applications.

7.1 Communicating with the Browser

There are numerous scenarios where it is helpful to integrate a desktop function with the web browser. Given that most users spend the widely held of their time today surfing the web in their browser of choice, it can make sense to provide some sort of incorporation with our desktop application. Often, this will be as simple as long as a way to export the current URL or a selected block of text to our application. For this, we have created a very simple application that uses text to speech to speak out loud the currently selected block of text in our web browser. Internet Explorer provides many hooks for integrating application logic into the browser, the most popular being support for adding custom toolbars. Chrome plug-in architecture which uses XML for the UI layout and JavaScript for the application logic.

What about the other browsers, Google Chrome, Safari, and Opera? Given there is no common plug-in architecture used by all browsers, we can see a huge development effort is required to provide a separate toolbar implementation for each browser.

It would be nice to be able to write one toolbar that could be used across all browsers. This is not possible today, but we can achieve almost the same effect using Bookmarklet.

7.1.1 Bookmarklet

A bookmarklet is special URL that will run a JavaScript application when clicked. The JavaScript will execute in the context of the current page. Like any other URL, it can be bookmarked and added to our Favourites menu or placed on a Favourites toolbar. Here is a simple example of a bookmarklet:

```
javascript:alert('Hello World.');
```

Here is a slightly more complex example that will display the currently selected text in a message box:

```
<a href= "javascript:var q = ''; if (window.getSelection)
q = window.getSelection().toString(); else if (document.getSelection)
q = document.getSelection(); else if (document.selection)
q = document.selection.createRange().text; if(q.length > 0)
alert(q); else alert('we must select some text first');">Show Selected
Text</a>
```

Now, drag the bookmarklet and drop it on our Favourites toolbar (for IE, we need to right click, select Add to Favourites, and then create it in the Favourites Bar). Navigate to a new page, select a block of text, and click the Show Selected Text button. Once again, the selected text will be shown in a message box. We can see the potential of bookmarklets. We can create a bookmarklet for each command of our application and display them on a web page. The user can then select the commands they want to use and add them to their Favourites (either the toolbar or menu). The downside is it is slightly more effort to install than a single toolbar, but on the upside, it gives the user a lot of flexibility. They need only choose the commands they are interested in and can choose whether they want them accessible from a toolbar or menu. From a developer's perspective, bookmarklets are great as they are supported by all the major browsers. The only thing we need to worry about is making sure our JavaScript code handles differences in browser implementations, something that is well documented and understood these days.

Notes

7.1.2 Communicating with a Desktop Application from JavaScript

Bookmark lets allow we to execute an arbitrary block of JavaScript code at the click of a button, but how do we use this to communicate with a desktop application? The answer is to build a web server into our desktop application and issue commands from our JavaScript code using HTTP requests.

Now, before we baulk at the idea of building a web server into our application, it is actually very simple. We do not need a complete web server implementation. We just need to be able to process simple HTTP GET requests. A basic implementation is as follows.

1. Listen for a socket connection on a specific port (80 is the default for the HTTP protocol, but we should use a different port for our application).
2. Accept a connection and read the HTTP GET request.
3. At the URL from the GET header and execute the associated command in our application.
4. Send an HTTP Response back to the browser.
5. Close the connection.

The .NET framework 2.0 has an HttpListener class and associated HttpListenerRequest and HttpListenerResponse classes that allow us to implement the above in a few lines of code. On the browser, we need a way of issuing HTTP requests from our JavaScript code. There are a number of ways of issuing HTTP requests from JavaScript.

The simplest is to write a new URL into the document.location property. This will cause the browser to navigate to the new location. However, this is not what we want. We do not want to direct the user to a new page when they click one of our bookmark lets. Instead, we just want to issue a command to our application while remaining on the same page.

This sounds like a job for AJAX and the HttpXml Request. AJAX provides a convenient means of issuing requests to a web server in the background without affecting the current page. However, there is one important restriction placed on the HttpXmlRequest called the same domain origin policy. Browsers restrict HttpXmlRequests to the same domain as that used to serve the current page. For example, if we are viewing a page from codeproject.com, we can only issue HttpXmlRequests to codeproject.com. A request to another domain (e.g., google.com) will be blocked by the browser. This is an important security measure that ensures malicious scripts cannot send information to a completely different server behind the scenes without our knowledge.

This restriction means that we cannot use an HttpXmlRequest to communicate with our desktop application. Remember that JavaScript bookmark lets are executed in the context of the current page. We need to be able to send a request from any domain (e.g. codeproject.com) to our desktop application which will be in the local host domain. Google needs to be able to do precisely this in order to gather analytics information for a site. If we are not familiar with Google analytics, it can be used to gather a multitude of information about the visitors to our web site, such as the number of visitors, where they came from, and the pages on our site they visit. All this information is collected, transferred back to Google, and appears in various reports in our analytics account.

To add tracking to our site, we simply add a call to the Google analytics JavaScript at the bottom of every page of our site. Whenever a visitor lands on our page, the JavaScript runs and the visitor details are sent back to our Google analytics account. The question is how does Google do this? Surely, they cannot use an HttpXmlRequest as it would break the same domain origin policy? They do not. Instead, they use what can only be described as a very clever technique.

Notes

7.1.3 Using the JavaScript Image Class to Make Asynchronous Cross-Domain Requests

The JavaScript Image class is a very simple class that can be used to asynchronously load an image. To request an image, we simply set the source property to the URL of the image. If the image loads successfully, the unload() method is called. If an error occurs, the onerror() method is called. Unlike the HttpXmlRequest, there is no same domain origin policy. The source image can be located on any server. It does not need to be hosted on the same site as the current page.

We can use this behaviour to send arbitrary requests to our desktop application (or any domain for that matter) if we realize the source URL can contain any information, including a query string. The only requirement is that it returns an image. Here is an example URL:

http://localhost:60024/speaktext/dummy.gif?text=Hello%20world

We can easily map this URL to the following command in our application.

Public void speak text(string text);

In order to ensure the request completes without error, a 1×1 pixel GIF image is returned. This image is never actually shown to the user. A tiny image is used to minimize the number of bytes being transmitted. The most important point to realize is all communication is one way, from the browser to the desktop application. There is no way of sending information from the desktop application back to the browser. However, for many applications, this is not a problem. Google uses the JavaScript Image technique to send visitor information to our pages (hosted on our domain.com) back to our Google analytics account (hosted on google.com).

Maximum URL Length

We need to be aware that URLs have a maximum length that varies from browser to browser (around 2K - check). This restricts the amount of information we can send in a single request. If we need to send a large amount of information, we will need to break it up into smaller chunks and send multiple requests. The sample application, Browser Speak, uses this technique to speak arbitrarily large blocks of text.

Text Encoding

JavaScript will automatically encode a URL we pass to Image.src as UTF-8. However, when passing arbitrary text as part of a URL, we will need to escape the '' and '=' characters. These characters are used to delimit the name/value pairs (or arguments) that are passed in the query string portion of the URL. This can be done using the JavaScript escape() function.

Avoiding the Cache

Web browsers will cache images (as well as many other resources) locally to avoid making multiple requests back to the server for the same resource. This behaviour is disastrous for our application. The first command will make it through to our desktop application, and the browser will cache dummy.gif locally. Subsequent requests will never reach our desktop application as they can be satisfied from the local cache.

There are a couple of solutions to this problem. One answer is to set the cache expiry directives in the HTTP response to instruct the browser never to cache the result.


Did u know?

The Uniform Resource Locator was created in 1994 by Tim Berners-Lee and the URI working group of the Internet Engineering Task Force (IETF) as an outcome of collaboration started at the IETF Living Documents "Birds of a Feather" session in 1992.

Notes

7.1.4 Browser Speak: A Concrete Example

It has some real world use. Now time to put all this theory into practice and create a sample application that Browser Speak is a C# application that will speak the text on a web page out loud. It can be used when we are tired of reading large passages of text from the screen. It uses the System Speech Synthesis component found in the .NET Framework 3.0 for the text to speech functionality. Browser Speak provides the following commands, available through its web interface and through its UI.

- Speak Text
- Stop Speaking
- Pause Speaking
- Resume Speaking

It also provides a Buffer Text command available from the web interface. This command is used to send a block of text from the web browser to the desktop application. It splits the text into 1500 byte chunks so it is not limited by the maximum size of a URL. It is used by the Speak Selected bookmark let to transfer the selected text to the BrowserSpeak application prior to speaking.

7.2 Web.Configuration

Web.Config acts as a central position for storing the information to be accessed by web pages. This information could be an association String stored at a federal location so that it can be accessed in a data-driven page. If the connection string changes it is just a matter of changing it at one place.

In classic ASP such global information was typically stored as an application variable. In the sample we will read the information from web configuration using ASP.NET and ASP as there could be a possibility of project having ASP and ASP.NET

Web. Config

```
<?xml version="1.0 " encoding= "utf-8"?>
<configuration>
<appSettings>
<add key= "ConnectionString1" value= "server=localhost;uid=sa;pwd
=;database=northwind" />
<add key= "ConnectionString2" value= "server=localhost;uid=sa;pwd
=;database=pubs" />
</appSettings>
</configuration
```

In ASP.NET

We can just using Configuration .App Settins (<key>) gives the Value.

(Name space: System. Configuration)

VB.NET

```
Dim star Connection as String
Star Connection = Configuration Settings .App Settings( "ConnectionString1")
Response.Write (star Connection)
```

Notes

C#

```
String star Connection;
STR Connection = Configuration Settings .App Settings[
“ConnectionString1”];
Response .Write (star Connection);
```

In ASP

We need to iterate through the nodes in web .comfit.

VBscript

```
set xml doc=server .Create Object( “Microsoft. XMLDOM”)
set xml app Settings=server .Create Object( “Microsoft .XMLDOM”)
set xml add=server. Create Object( “Microsoft .XMLDOM”)
xml Doc .a sync= “false”
xml Doc .load(server Map Path ( “web comfit”))
Set xml app Settings = xml doc .Get Elements By Tag Name( “app
Settings”).Item(0)
Set xml add = xml app Settings. Get Elements by Tag Name(“add”)
For each x in xml add
‘Check for the Attribute Value
If x .get Attribute(“key”) = “ConnectionString1” then
Response.write(x .get Attribute (“value”))
End if
Next
```

7.2 ASP.NET Web.Configuration File

ASP.Net Applications of XML have been integrated into such an extent that XML format for the exchange of data, it is also used to store

1. A Web application can contain more than one file. The settings in a file apply to the directory in which it is located, and all child directories. Web .comfit files in child directories take precedence over the settings that are specified in parent directories.
2. Web .comfit files are protected by IIS, so clients cannot get to them. If we try to retrieve an existing http://?Com/Web.config file, we will be presented with an “Access denied” error message.
3. IIS monitors the Web .comfit files for changes and caches the contents for performance reasons. There is no need to restart the Web server after we modify a Web. Configuration file.

Configuration settings for any of our ASP.NET Web applications can be stored in a simple text file. Presented in an easily understandable XML format, this file, called Web. Configuration can contain application-wide data such as database connection strings, custom error messages, and culture settings. The Web .comfit is an XML file, it can consist of any valid XML tags, but the root element should always be <configuration>. Nested within this tag we can include various other tags to describe our settings. Since a Web. Comfit file comes as a standard when we start to build a new Web application.The default XML file generated by Visual Studio .NET:

```
<?xml version= “1.0” encoding= “utf-8” ?>
```

Notes

```
<configuration>
<system.web>
<compilation default Language= "c#" debug= "true"/>
<custom Errors mode= "Remote Only"/>
<authentication mode= "Windows" />
<Authorization>
<allow users= "*" />
</authorization>
<trace
enabled= "false"
request Limit= "10"
page Output= "false"
trace Mode= "Sort By Time"
local Only= "true"
/>
<sessionState
mode= "InProc"
stateConnectionString= "tcpip=127.0.0.1:42424"
sqlConnectionString= "data source=127.0.0.1;Trusted_
Connection=yes"
cookieless= "false"
timeout= "20"
/>
<globalization
requestEncoding= "utf-8"
responseEncoding= "utf-8"
/>
</system.web>
</configuration>
```

The <configuration> tag has only one child tag, which we call section group, the <system.web> tag. A section group typically contains the setting sections, such as: compilation, customErrors, authentication, authorization, etc. The way this works is pretty straightforward: we simply include our settings in the appropriate setting sections. We want to use a different authentication mode for our Web application; we had changed that setting in the authentication section.

Apart from the standard system.web settings, we can define our own specific application settings, such as a database connection string, using the <app Settings> tag. Consequently, our most common Web.config outline would be:

```
<configuration>
<system.web>
<! — sections-->
</system.web>
<appSettings>
<! — sections -->
</appSettings >
</configuration>
```

Notes

Let us discuss the details of both section groups now.

The system.web Section Group

Now we will include configuration settings that, we did have set up somewhere in the IIS administration console. At Microsoft's MSDN Library, we can find an overview of all the tags that the system.web section group understands, but, depending on the complexity of our site, we may not ever use even half of those options.

<authentication>

The authentication section controls the type of authentication used within our Web application, as contained in the attribute mode. We will enter the value "None" if anyone may access our application. If authentication is required, we will use "Windows", "Forms" or "Passport" to define the type of authentication. For example:

<authentication mode= "Windows"/>

<authorization>

To allow or deny access to our web application to certain users or roles, use <allow> or <deny> child tags.

<authorization>

<allow roles= "Administrators, Users" />

<deny users= "*" />

</authorization>

It is important to understand that ASP.NET's authorization module iterates through the sections, applying the first rule that corresponds to the current user. In this example, users carrying the role Administrators or Users will be allowed access, while all others (indicated by the * wildcard) will encounter the second rule and will subsequently be denied access.

<compilation>

Here, we can configure the compiler settings for ASP.NET. We can use loads of attributes here, of which the most common are debug and default Language. Set debug to "true" only if we want the browser to display debugging information. Since turning on this option reduces performance, we had normally wanted to set it to "false". The default Language attribute tells ASP.NET which language compiler to use, since we could use either Visual Basic .NET or C# for instance. It has value by default.

<custom Errors>

To provide our end users with custom, user-friendly error messages, we can set the mode attribute of this section to on. If we set it to Remote Only, custom errors will be shown only to remote clients, while local host users will see the ugly but useful ASP.NET errors -- clearly, this is helpful when debugging. Setting the mode attribute to off will show ASP.NET errors to all users.

If we supply a relative or absolute address in the default Redirect attribute, the application will be automatically redirected to this address in case of an error. Note that the relative address is relative to the location of the Web. Configuration file, not the page in which the error takes place. In addition we can use <error> tags to provide a status Code and a redirect attributes:

<custom Errors mode= "Remote Only" default Redirect= "/error.html ">

<error status Code="403" redirect= "/accessdenied.html" />

<error status Code= "404" redirect= "/pagenotfound.html" />

</custom Errors>

<globalization>

The globalization section is useful when we want to change the encoding or the culture of our application. Globalization is such an extensive subject could be dedicated to the matter. In short,

Notes

this section allows we to define which character set the server should use to send data to the client (for instance UTF-8, which is the default), and which settings the server should use to interpret and displaying culturally specific strings, such as numbers and dates.

<globalization request Encoding= "utf-8" response Encoding= "utf-8"

Culture= "nl-NL" />

Encoding is done through the attributes request Encoding and response Encoding. The values should be equal in all one-server environments. In this example, the application culture is set to Dutch. If we do not supply a culture, the application will use the server's regional settings.

<Http Runtime>

We can use the http Runtime section to configure a number of general runtime settings, two of which are particularly convenient.

<http Runtime app Request Queue Limit= "100" execution Timeout= "600" />

The first attribute specifies the number of requests the server may queue in memory at heavy-traffic times. In the example, if there are already 100 requests waiting to be processed, the next request will result in a 503 error ("Server too busy").The execution Timeout attribute indicates the number of seconds for which ASP.NET may process a request before it is timed out.

<session State>

Web. Configuration file, we tell ASP.NET where to store the session state. The default is in the process self:

<session State mode= "In Proc" />

Session variables are very powerful, but they have a few downsides. Information is lost when the ASP.NET process crashes, and sessions are generally useless in the case of a Web farm (multiple Web servers). In that instance, a shared session server can solve our issues.

<trace>

Our application's trace log is located in the application root folder, under the name trace.axd. We can change the display of tracing information in the trace section.

The attributes we will look for initially are enabled: local only, and page Output.

<trace enabled= "true" local only= "true" page Output= "false" />

Set local only to "false" to access the trace log from any client. If we set the value of page Output to "true", tracing information will be added to the bottom of each Web page.

Apart from the Website configuration settings we have been talking about in the preceding paragraphs, we will know that a programmer frequently likes to use custom application-wide constants to store information over multiple pages. The most appealing example of such a custom constant is a database connection string, but we can probably think of dozens more from our own experience.

The common denominator of these constants is that we want to retrieve their values programmatically from our code. The web. configuration file provides the possibility to do so, but as a security measure, these constants have to be included in the <app Settings> section group. Just like <system.web>, <app Settings> is a direct child tag of the Web. Configuration's configuration root.

A typical custom section group would look something like this:

<app Settings>

<add key= "sql Conn" value= "Server=myPc;Database=Northwind" />

<add key= " smtp Server" value= "smtp.mydomain.com" />

</appSettings>

Notes

Task Add a Logging Database Connection String to the Web.config File.

Self Assessment Questions

Multiple Choice Questions

True or False

1. .NET is not provides a visual interface for developers to create their applications, which makes.
   (*a*) True (*b*) False
2. JavaScript code using HTTP requests.
   (*a*) True (*b*) False
3. Http Xml Request as it would not break the same domain origin policy.
   (*a*) True (*b*) False
4. .....as contained in the attribute mode.
   (*a*) Liberalization (*b*) Privatization
   (*c*) Web application (*d*) None of these
5. The Uniform Resource Locator was created in.....
   (*a*) 1989 (*b*) 1999
   (*c*) 1972 (*d*) 1994

7.3 Characteristics of ASP.Net

ASPX file format ASPX is a text file arrangement used to produce Web form pages; in programming terminology, the ASPX file typically contain static HTML or XHTML mark-up, as well as chalk up defining Web Controls and Web User Controls where the developers place all the required static and dynamic content for the web page. Additionally, dynamic code which runs on the server can be placed in a page within a block which is similar to other web development technologies such as PHP, JSP, and ASP, but this practice is generally frowned upon by Microsoft except for the purposes of data binding since it requires more calls when rendering the page. The method recommended by Microsoft for dealing with dynamic program code is to use the code-behind model, which places this code in a separate file or in a specially designated script tag. Code-behind files are typically named something to the effect of MyPage.aspx.cs or MyPage. aspx.vb based on the ASPX file name (this practice is automatic in Microsoft Visual Studio and other IDEs). When using this style of programming, the developer writes code to respond to different events, like the page being loaded, or a control being clicked, rather than a procedural walk through the document. Rendering technique ASP.NET uses a visited composites rendering technique. During compilation the template (.aspx) file is compiled into initialization code which will build a control tree (the composite) representing the original (static) template. Literal text goes into instances of the Literal control class, server controls are represented by instances of a specific control class. The initialization code is combined with user-written code (usually by the assembly of multiple partial classes) and results in a class specific for the page. The page doubles as the root of the control tree. Actual requests for the page are processed through a number of steps. First, during the initialization steps, an instance of the page class is created and the initialization code is executed. This produces the initial control tree which is now typically manipulated by the methods of the page in the following steps. As each node in the tree is a control represented as an instance of a class, the code may change the tree structure as well as manipulate the properties/methods of the individual nodes. Finally, during the rendering step a visitor is used to visit every node in the tree, asking each node to render itself using the

Notes

methods of the visitor. The resulting HTML code is sent to the client. After the request has been processed, the instance of the page class is discarded and with it the entire control tree. Other files Other file extensions associated with different versions of ASP.NET include: asax Global.asax, used for application-level logic and event handling ascx Web UserControls: custom controls to be placed onto web pages. ashx custom HTTP handlers asmx web service pages. axd when enabled in web.config requesting trace.axd outputs application-level tracing. Also used for the special webresource axd handler which allows control/component developers to package a component/control complete with images, script, css etc. for deployment in a single file (an 'assembly') browser browser capabilities files stored in XML format; introduced in version 3.0 ASP.NET 2 includes many of these by default, to support common web browsers. These specify which browsers have which capabilities, so that ASP.NET 2 can automatically customize and optimize its output accordingly. Special .browser files are available for free download to handle, for instance, the W3C Validator, so that it properly shows standards-compliant pages as being standards-compliant. Replaces the harder-to-use BrowserCaps section that was in machine.config and could be overridden in web.config in ASP.NET 1.x. config web.config is the only file in a specific Web application to use this extension by default (machine.config similarly affects the entire Web server and all applications on it), however ASP.NET provides facilities to create and consume other config files. These are stored in XML format, so as to allow configuration changes to be made with simplicity. cs/vb In ASP.NET 2 any cs/vb files placed inside the App_Code folder are dynamically compiled and available to the whole application. master Master Pages; introduced in version 2.0 sitemap sitemap configuration files skin theme skin files. resx resource files for internationalization and localization. Resource files can be global (e.g. messages) or "local" which means specific for a single aspx or ascx or file.


Task

Write a Transform to Change the Environment and Logging Connection Strings in the Staging Web.config.


Caution

Although ASP.NET applications are automatically updated to use the installing version of ASP.NET if the preceding conditions are met, custom configuration settings in the current Machine.config file are not transferred to the installing Machine.config file. If your application uses custom configuration settings, be sure to either manually update the new Machine.config file or use the ASP.NET IIS Registration tool (Aspnet_regiis.exe) to remap the application to the previous version of ASP.NET.

7.4 New Features in ASP.NET 3.5

Following features are added in ASP.NET 3.5:

ASP.NET AJAX

In ASP.NET 2.0, ASP.NET AJAX was used as an extension to it. We had to download the extensions and install it. However in ASP.NET 3.5, ASP.NET AJAX is integrated into the .NET Framework, thereby making the process of building cool user interfaces easier and intuitive.

The integration between web parts and the update panel is much smoother. Another noticeable feature is that we can now add ASP.NET AJAX Control Extenders to the toolbox in VS2008. Even though this is an IDE specific feature, if it deserves a mention over here for developers, who had to add extenders using source view earlier. It is also worth noting that Windows Communication Foundation (WCF) now supports JSON along with other standard protocols like SOAP, RSS and POX.

New Controls

The List View and Data Pager are new controls added along with a new data source control called the LinqDataSource.

Notes

List View

The List View control is quiet flexible and contains features of the Gridview, Data grid, Repeater and similar list controls available in ASP.NET 2.0. It provides the ability to insert, delete, page (using Data Pager), sort and edit data. However one feature of the List View control that stands apart is that it gives us a great amount of flexibility over the mark-up generated. So we have a complete control on how the data is to be displayed. We can now render our data without using the tag. We also get a rich set of templates with the List View control.

Data Pager

DataPager provides paging support to the ListView control. The best advantage is that we need not have to keep it 'tied' with the control on which the paging is being done. We can keep it anywhere on the page.

DataPager gives us a consistent way of paging with the controls that support it. Currently only ListView supports it as it implements the IPageableItemContainer. However support is likely to be added to other List controls as well.

LINQ

LINQ (Language Integrated Query) adds native data querying capability to C# and VB.NET along with the compiler and Intelligence support. LINQ is a component of .NET 3.5. LINQ defines operators that allow us to code our query in a consistent manner over databases, objects and XML. The ASP.NET LinqDataSource control allows us to use LINQ to filter, order and group data before binding to the List controls.

ASP.NET Merge Tool

ASP.NET 3.5 includes a new merge tool (aspnet_merge.exe). This tool lets us combine and manage assemblies created by aspnet_compiler.exe.

Case Study

ASP.NET Social Networking Website with PHP Blog

This client is an Italy based company operating a social network website. This web site comprised an outmoded blog module written in ASP.NET which was difficult to maintain. Therefore Nova helped the client to develop a web control for the function of sending batches of emails, replace the old blog system with WordPress which is a famous blog system, and customize it to eliminate the weakness in uploading files by WordPress.

Situation

This client is an Italy based company operating a social network website. This website comprised an outmoded blog module which was difficult to maintain. Then the client was thinking of replacing the old blog with WordPress. However he was also unsatisfied with WordPress' weak function for uploading files. Therefore he decided to engage our expert to develop a web control for the function of sending batches of emails. In addition to further enrichment of the site functions, he also looked to customize the WordPress. WordPress is the most famous blog system now.

Requirements

The project was divided into three sub-projects: File Manager Control, Mass Mailing Control and Blogs Integration.

File Manager Control

We were required to use Fancy Uploader to realize the file upload function and make all the interface can be customized by amending relevant XSLT template.

Contd...

Notes

Mass Mailing Control

The sub-project should realize the function of Email collection and filtration, as well as the procedure of sending Email through Widows Service of the background.

Blogs Integration

The part should integrate the blog of WPMU to the current system.

Solutions

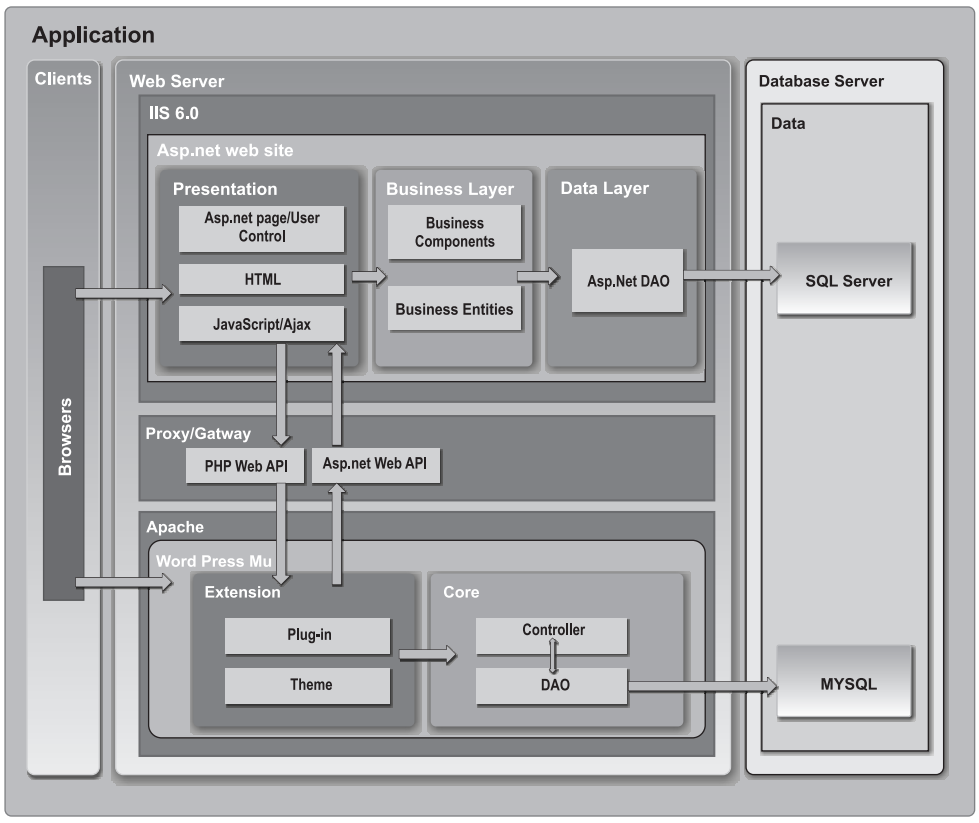
Technology Required

- XML/XSLT
- JavaScript/Prototypes
- Windows Service
- Asp.net/Custom User Control
- PHP
- Custom Word Press Mu Plug-in
- SWF File Uploader

System Architecture

We adopted a multi-user version called WordPress Mu(sometimes WPMu for short) because this version had some functions that were closer to the client's requirements.

But the website was wrote in ASP.NET, so we added the layer Proxy/Gateway to communicate between the website and WPMU



Contd...

Notes

The ASP.NET Website

This was a social network website of the client which we integrated to WPMU.

Proxy/Gateway

As we knew WPMU was a PHP project using the database MySql while the client's original website was an Asp.net project using the database SQLServer2005. So we decided to develop the layer to communicate between PHP and ASP. We reserved the user's information in the ASP.Net and used PHP to deal with identity authentication information. At the same time Asp.Net also could call the custom network API written in PHP to access the data in WPMU.

WPMU Layer

WordPress is a PHP open source project that has multiple versions and supporting Plugin and Theme. To realize accounts synchronization, we implemented account authentication by intercepting API function through plug-in.

Challenges of the Development Process

- In the development of the sub-project "File Manager Control", we were required to use Fancy Uploader to realize the file upload function. However there was a stubborn problem: the Fancy Uploader control also used a JavaScript framework named MooTools which would conflict with prototype.js.

  As the time was very tight, we advised that temporarily rename the related method in MooTools for the purpose of resolving the conflict.
- WPMU supports two types of deployment, one is deploying through visiting the directory and the other is deploying through supporting subdomain. We adopted the client's suggestion to deploy using subdomain. But the subdomain was often parsed to wrong websites.

  We found that IIS filter plug-in did not support the subdomain well. So we gave up deploying on IIS and switched to use Apache deployment on the Linux platform

Benefits to the Client

- The completed code has greatly enhanced the expansibility, stability and maintainability.
- The original website has added more members after the system integrated the Word Press Mu blog to the Asp.net.
- The new system is easier to update for the client and brings the latest user experience.
- We have helped our client reduce their development cost.

Questions

1. What are the challenges for the development process of the Website?
2. What benefits the client found?

Self Assessment Questions

6. URLs have a maximum length that varies from browser to browser like

   (*a*) Around 3k (*b*) Around 2k

   (*c*) Around 4k (*d*) Around 1k

7. URL stands for .....

   (*a*) Universal Resource Locator (*b*) United Resource Locator

   (*c*) Both (*a*) and (*b*) (*d*) None of these.

Notes

8. It is the process of international integration, required due to the increasing connectivity and interdependence of the world's markets and businesses

   (*a*) Liberalisation (*b*) Privatization

   (*c*) Globalization (*d*) Both b and c

9. LINQ stands for ....

   (*a*) Language Integrated Query (*b*) Language Integrated Queue

   (*c*) Line Integrated Query (*d*) None of these.

10. ..... provides paging support to the ListView control

    (*a*) List View (*b*) Grid view

    (*c*) DataPager (*d*) Both a and c

11. <app Settings> is used for.....

    (*a*) database connection (*b*) coding

    (*b*) testing (*d*) None of these.

12. A bookmarklet is special URL that will run a JavaScript application when clicked.

    (*a*) True (*b*) False

13. Web.config does not act as a central location for storing the information to be accessed by web pages.

    (*a*) True (*b*) False

14. ASP.NET is built on the .NET framework, which provides an application program interface (API) for software programmers.

    (*a*) True (*b*) False

15. ASP.Net Applications of XML have been integrated into such an extent that XML format for the exchange of data.

    (*a*) True (*b*) False

7.5 Summary

- ASP.NET is a set of Web development tools offered by Microsoft. Programs like Visual Studio .NET and Visual Web Developer allow Web developers to create dynamic websites using a visual interface.
- ASP.NET is more than the next version of Active Server Pages (ASP); it provides a unified Web development model that includes the services necessary for developers to build enterprise-class Web applications.
- A bookmarklet is special URL that will run a JavaScript application when clicked. The JavaScript will execute in the context of the current page.
- The Web .comfit is an XML file, it can consist of any valid XML tags, but the root element should always be <configuration>.
- NET provides a visual interface for developers to create their applications, which makes .NET a reasonable choice for designing Web-based interfaces as well.

Notes

7.6 Keywords

Authentication: It is the act of confirming the truth of an attribute of a datum or entity.

Authorization: It is the function of specifying access rights to resources, which is related to information security and computer security in general and to access control in particular.

Bookmarklet: A bookmarklet is unobtrusive JavaScript stored as the URL of a bookmark in a web browser or as a hyperlink on a web page.

Globalization: It is the process of international integration, required due to the increasing connectivity and interdependence of the world's markets and businesses.

Web.config: It is the main settings and configuration file for an ASP.NET web application. The file is an XML document that defines configuration information regarding the web application.

Lab Exercise

1. Add Code to Populate Text Value of EnvName from a Value in the Web. config.
2. Transforming a Web.Config File for Deployment.

7.7 Review Questions

1. What is the advance ASP.NET?
2. What are the general principles of web browser?
3. Describe the Web.config file.
4. Explain the classifications of Advanced ASP.NET
5. How can we communicate with a desktop application from JavaScript
6. What is app setting?
7. What is bookmarklet?
8. Define the characteristic of ASP.NET.
9. Explain the new features of ASP.NET.
10. What is data pager?

Answer for Self Assessment Questions

1.	(*b*)	2.	(*a*)	3.	(*b*)	4.	(*c*)	5.	(*d*)
6.	(*b*)	7.	(*a*)	8.	(*c*)	9.	(*a*)	10.	(*c*)
11.	(*a*)	12.	(*a*)	13.	(*b*)	14.	(*a*)	15.	(*a*)

7.8 Further Readings

Books *ASP.Net 3.5 Website Programming*: *Problem Design Solution*, by Chris Love

Online link http://books.google.co.in/books?id=av--5iDeXo4C&printsec=frontcover&dq=Advanced+ASP.NET+3.5&hl=en&sa=X&ei=5bb-T6DhJYXRrQeK6b3cBg&ved=0CFoQ6AEwBQ#v=onepage&q=Advanced%20ASP.NET%203.5&f=false

Unit 8: Creating More Advanced ASP.NET

Notes

CONTENTS

Objectives

After studying this unit, you will be able to:

- Define page sub-classing
- Explain user control
- Describe data binding
- Understand two-way data binding

Notes

Introduction

ASP.NET is above the subsequently version of Active Server Pages (ASP); it provides a combined Web development model that includes the services required for developers to build enterprise-class Web application. While ASP.NET is mostly syntax like-minded with ASP, it also provides a new programming replica and transportation for more scalable and stable applications that help provide greater protection. You can feel free to augment your existing ASP applications by incrementally adding ASP.NET functionality to them.ASP.NET is a compiled, .NET-based environment; you can author applications in any .NET compatible language, including Visual Basic .NET, C#, and JScript .NET. Additionally, the entire .NET Framework is available to any ASP.NET application. Developers can easily access the benefits of these technologies, which include the managed common language runtime environment, type safety, inheritance, and so on.ASP.NET has been designed to work seamlessly with WYSIWYG HTML editors and other programming tools, including Microsoft Visual Studio .NET. Not only does this make Web development easier, but it also provides all the benefits that these tools have to offer, including a GUI that developers can use to drop server controls onto a Web page and fully integrated debugging support. Developers can use Web Forms or XML Web services when creating an ASP.NET application, or combine these in any way they see fit. Each is supported by the same infrastructure that allows you to use authentication schemes; cache frequently used data, or customizes your application's configuration, to name only a few possibilities.

- Web Forms allow you to build authoritative forms-based Web pages. When structure these pages, you can use ASP.NET wine waiter controls to create common UI elements, and program them for common tasks. These controls allow you to rapidly build a Web Form out of reusable built-in or custom components, simplifying the code of a page.
- An XML Web service provides the means to access server functionality remotely. Using XML Web services, businesses can expose programmatic interfaces to their data or business logic, which in turn can be obtained and manipulated by client and server applications. XML Web services enable the exchange of data in client-server or server-server scenarios, using standards like HTTP and XML messaging to move data across firewalls. XML Web services are not tied to a particular component technology or object-calling convention. As a result, programs written in any language, using any component model, and running on any operating system can access XML Web services.
- Each of these models can take full advantage of all ASP.NET features, as well as the power of the .NET Framework and .NET Framework common language runtime. These features and how you can use them are outlined as follows:

If you have ASP development skills, the new ASP.NET programming model will seem very familiar to you. However, the ASP.NET object model has changed significantly from ASP, making it more structured and object-oriented. Unfortunately this means that ASP.NET is not fully backward compatible; almost all existing ASP pages will have to be modified to some extent in order to run under ASP.NET. In addition, major changes to Visual Basic .NET means that existing ASP pages written with Visual Basic Scripting Edition typically will not port directly to ASP.NET. In most cases, though, the necessary changes will involve only a few lines of code.

Accessing databases from ASP.NET applications is an often-used technique for displaying data to Web site visitors. ASP.NET makes it easier than ever to access databases for this purpose. It also allows you to manage the database from your code.

ASP.NET provides a simple model that enables Web developers to write logic that runs at the application level. Developers can write this code in the Global.asax text file or in a compiled class deployed as an assembly. This logic can include application-level events, but developers can easily extend this model to suit the needs of their Web application.

ASP.NET provides easy-to-use application and session-state facilities that are familiar to ASP developers and are readily compatible with all other .NET Framework APIs.

Notes

- For advanced developers who want to use APIs as powerful as the ISAPI programming interfaces that were included with previous versions of ASP, ASP.NET offers the IHttpHandler and IHttpModule interfaces. Implementing the IHttpHandler interface gives you a means of interacting with the low-level request and response services of the IIS Web server and provides functionality much like ISAPI extensions, but with a simpler programming model. Implementing the IHttpModule interface allows you to include custom events that participate in every request made to your application.
- ASP.NET takes advantage of performance enhancements found in the .NET Framework and common language runtime. Additionally, it has been designed to offer significant performance improvements over ASP and other Web development platforms. All ASP.NET code is compiled, rather than interpreted, which allows early binding, strong typing, and just-in-time (JIT) compilation to native code, to name only a few of its benefits. ASP.NET is also easily factorable, meaning that developers can remove modules (a session module, for instance) that are not relevant to the application they are developing. ASP.NET also provides extensive caching services (both built-in services and caching APIs). ASP.NET also ships with performance counters that developers and system administrators can monitor to test new applications and gather metrics on existing applications.
- Writing custom debug statements to your Web page can help immensely in troubleshooting your application's code. However, they can cause embarrassment if they are not removed. The problem is that removing the debug statements from your pages when your application is ready to be ported to a production server can require significant effort. ASP.NET offers the TraceContext class, which allows you to write custom debug statements to your pages as you develop them. They appear only when you have enabled tracing for a page or entire application. Enabling tracing also appends details about a request to the page, or, if you so specify, to a custom trace viewer that is stored in the root directory of your application.
- The .NET Framework and ASP.NET provide default authorization and authentication schemes for Web applications. You can easily remove, add to, or replace these schemes, depending upon the needs of your application.
- ASP.NET configuration settings are stored in XML-based files, which are human readable and writable. Each of your applications can have a distinct configuration file and you can extend the configuration scheme to suit your requirements.
- Applications are said to be running side by side when they are installed on the same computer but use different versions of the .NET Framework. To learn how to use different versions of ASP.NET for separate applications on your server, see Side-by-Side Support in ASP.NET.

8.1 Page Sub-classing

A subclass, heir class, or child class is a modular, derivative class that inherits one or extra properties from a different class. The properties in query vary from language to language, other than commonly include class data variables, properties, and methods or functions. Some languages support the inheritance of other properties as well. For example, in Eiffel, contracts which define the specification of a class are also inherited by heirs. The superclass establishes a common interface and foundational functionality, which specialized subclasses can inherit, modify, and supplement. The software inherited by a subclass is considered reused in the subclass.

In some cases, a subclass may customize or redefine a method inherited from the superclass. A superclass method which can be redefined in this way is called a virtual method.

In the RoR application, we are not making use of the model layer because the model is implemented through a C++ server and my RoR application communicates with that server using a middleware product.

Relay nice about my C++ server is that it holds very rich meta information describing the business objects it returns in great details. This allows me to implement a single controller/view that can render any object type in a very generic way.

Notes

While this model works perfectly, there is a downside. For some object types, the generic representation is not quite sufficient. We might have to override or augment the default behavior.

In a truly object oriented environment we would simply subclass my ObjectController. The sub-classed controller could neatly alter the behaviour of the ancestor class. The problem is, we have no idea how we can direct RoR to instantiate the sub-classed controller. Is this even possible?

The following controller hierarchy:

ActionController::Base

ApplicationController

VehicleController

Vehicle::CarController

Vehicle::BikeController

While this goes some ways to address my design issue, one of the things it does not seem to do "out-of-the-box" is to allow me to override views.

Consider my VehicleController implements the method show. In app/views/vehicle we have a file show.rhtml. We never instantiate a VehicleController object though: We either instantiate a Vehicle::CarController or a Vehicle::BikeController. Either of these two controllers inherits the show method from VehicleController.

However, there is now an issue with selecting the views. Consider send the show message to a Vehicle::CarController. Ruby correctly invokes the VehicleController.show method which might call render(:action => "show"). The preferred behaviour is now that if a show.rhtml was found in app/views/vehicles/car, rails should render that template. If none was found there, it should try and pick the show.rhtml from app/views/vehicle and if that did not exist ether, it should fail.

Example: ParentPage.cs

```
public class ParentPage : System.Web.UI.Page
{
protected String _UserName;
public ParentPage()
{
this.Load += new System.EventHandler(this.Page_Load);
}

protected void Page_Load(object sender, EventArgs e)
{
UserName = Session[ "UserName"].ToString();
}
}
page1.aspx.cs
public partial class Page1 : ParentPage
{
protected void Page_Load(object sender, EventArgs e)
{
}
}
page2.aspx.cs
public partial class Page1:ParentPage
{
protected void Page_Load(object sender, EventArgs e)
```

Notes

```
{
}
}
```

Note that we need to add the Page_Load method in the Parent Page.cs to the loading event handler so that the method will get called when the page loads.

Now we just need to use the User Name in either of our pages, even though it is a protected member of Parent Page, the pages that subclass from that page can reach the member. However, other objects can modify User Name. This we get from inheritance.

Example:

```
ParentPage.cs
public class ParentPage : System.Web.UI.Page
{
protected String _UserName;
protected override void OnInit(EventArgs e)
{
//WWB: Allow The Base Class To Intialize First
base.OnInit(e);
UserName = Session[ "UserName"].ToString();
}
}
page1.aspx.cs
public partial class Page1 : ParentPage
{
 protected void Page_Load(object sender, EventArgs e)
{
Response.Write(_UserName);
}
}
{
```

Example:

```
ParentMasterPage.cs
public class ParentMasterPage : System.Web.UI.MasterPage
{
protected String _title;
public ParentMasterPage()
{
}
public void SetTitle(String title)
{
title = title;
}
}
ParentPage.cs
public class ParentPage : System.Web.UI.Page
{
protected String _UserName;
```

Notes

```
public String UserName
{
get { return (_UserName);
}
protected override void OnInit(EventArgs e)
{
// WWB: Allow The Base Class To Initialize First
base.OnInit(e);
UserName = Request[ "UserName"].ToString();
this.Load += new System.EventHandler(this.Page_Load);
}
protected void Page_Load(object sender, EventArgs e)
{
((ParentMasterPage)Master).SetTitle(_UserName);
}
}
MasterPage1.master.cs
public partial class MasterPage1 : ParentMasterPage
protected void Page_Load(object sender, EventArgs e)
{
lblTitle.Text =title;
}
}
```

The IIS 6.0 uses a new process model called worker process isolation mode, which is different from the process model used in previous versions of IIS. ASP.NET uses this process model by default when running on Windows Server 2003.

8.2 User Control

Server controls are one of the possessions that create developing with ASP.NET so easy and influential at the similar time. We have discuss HTML and web server controls and have showed you how to use them in your ASP.NET pages, but what if present is not a control that do exactly what you want to do? Like most all in ASP.NET there is really no magic concerned with server controls. In fact, you can build your own controls and use them on your pages just like you use the ones that ship with .NET. Controls that you build are called user controls and they are the topic of this lesson.

User Control Structure

- ASP.NET User Controls Overview
- How to: Convert Web Forms Pages into ASP.NET User Controls
- How to: Include a User Control in an ASP.NET Web Page
- How to: Create Instances of ASP.NET User Controls Programmatically
- How to: Create Templated ASP.NET User Controls
- How to: Create ASP.NET User Controls
- How to: Include ASP.NET User Controls in Web Pages
- Walkthrough: Creating Reusable Elements with ASP.NET User Controls

Notes

8.2.1 Overview

An ASP.NET Web user control is similar to a complete ASP.NET Web page (.aspx file), with both a user interface page and code. You create the user control in much the same way you create an ASP.NET page and then add the markup and child controls that you need. A user control can include code to manipulate its contents like a page can, including performing tasks such as data binding.

A user controls differs from an ASP.NET Web page in these ways:

- The file name extension for the user control is .ascx.
- Instead of a @ Page directive, the user control contains an @ Control directive that defines configuration and other properties.
- User controls cannot run as stand-alone files. Instead, you must add them to ASP.NET pages, as you would any control.
- The user control does not have html, body, or form elements in it. These elements must be in the hosting page.
- You can use the same HTML elements (except the html, body, or form elements) and Web controls on a user control that you do on an ASP.NET Web page. For example, if you are creating a user control to use as a toolbar, you can put a series of Button Web server controls onto the control and create event handlers for the buttons.

The following example shows a user control that implements a spinner control where users can click up and down buttons to rotate through a series of choices in a text box.

Example: <% @ Control Language= "C#" ClassName= "UserControl1" %>

```
<script runat= "server">
protected int currentColorIndex;
protected String[] colors = { "Red", "Blue", "Green", "Yellow"};
protected void Page_Load(object sender, EventArgs e)
{
if (IsPostBack)
{
currentColorIndex =
Int16.Parse(ViewState[ "currentColorIndex"].ToString());
}
else
{
currentColorIndex = 0;
DisplayColor();
}
}
protected void DisplayColor()
{
textColor.Text = colors[currentColorIndex];
```

Notes

```
ViewState[ "currentColorIndex"] = currentColorIndex.ToString();
}
protected void buttonUp_Click(object sender, EventArgs e)
{
if(currentColorIndex == 0)
{
currentColorIndex = colors.Length - 1;
}
else
{
currentColorIndex -= 1;
}
DisplayColor();
}
protected void buttonDown_Click(object sender, EventArgs e)
{
if(currentColorIndex == (colors.Length - 1))
{
currentColorIndex = 0;
}
else
{
currentColorIndex += 1;
}
DisplayColor();
}
</script>
<asp:TextBox ID= "textColor" runat= "server"
ReadOnly= "True" />
<asp:Button Font-Bold= "True" ID= "buttonUp" runat= "server"
Text= "^" OnClick= "buttonUp_Click" />
<asp:Button Font-Bold= "True" ID= "buttonDown" runat= "server"
Text= "v" OnClick= "buttonDown_Click" />
```

8.2.2 How to Convert Web Form Pages into ASP.NET User Control

If you have developed an ASP.NET Web page and would like to access its functionality throughout your application, you can make some minor alterations to the page to change it to a user control.

Notes

To convert a single-file ASP.NET Web page into a user control

1. Rename the control so the file name extension is .ascx.
2. Remove the html, body, and form elements from the page.
3. Change the @ Page directive to a @ Control directive.
4. Remove all attributes of the @ Control directive except Language, AutoEventWireup (if present), CodeFile, and Inherits.
5. Include a className attribute in the @ Control directive. This allows the user control to be strongly typed when it is added to a page.

To convert a code-behind ASP.NET Web page into a user control

1. Rename the .aspx file so the file name extension is .ascx.
2. Rename the code-behind file to have the file name extension .ascx.vb or .ascx.cs, depending on what programming language the code-behind file is in.
3. Open the code-behind file and change the class from which it inherits from Page to User Control.
4. In the .aspx file, do the following:
   - Remove the html, body, and form elements from the page.
   - Change the @ Page directive to an @ Control directive.
   - Remove all attributes of the @ Control directive except Language, AutoEventWireup (if present), CodeFile, and Inherits.
   - In the @ Control directive, change the CodeFile attribute to point to the renamed code-behind file.
5. Include a className attribute in the @ Control directive. This allows the user control to be strongly typed when it is added to a page.

8.2.3 How to Include a User Control in an ASP.NET Web Page

To use a user control, you include it in an ASP.NET Web page. When a request arrives for a page and that page contains a user control, the user control goes through all of the processing stages that any ASP.NET server control performs.

To include a user control in a Web Forms page

1. In the containing ASP.NET Web page, create an @ Register directive that includes:
   - A Tag Prefix attribute, which associates a prefix with the user control. This prefix will be included in opening tag of the user control element.
   - A Tag Name attribute, which associates a name with the user control. This name will be included in the opening tag of the user control element.
   - A Src attribute, which defines the virtual path to the user control file that you are including.
2. In the body of the Web page, declare the user control element inside the form element.
3. Optionally, if the user control exposes public properties, set the properties declaratively.

Notes

8.2.4 How to Create Instances of ASP.NET User Controls Programmatically

To create an instance of a user control programmatically

In the user control, be sure that the @ Control directive contains a ClassName attribute that assigns a class to the user control. The following example sets the ClassName attribute to strongly type a user control.

<%@ Control class Name= "MyUserControl" %>

In the page where you want to work with the user control, create a reference to the user control with the @ Reference directive.

When you create the user control programmatically, the strong type for your user control is available to the ASP.NET Web page only after you have created a reference to it. For example, the following code creates a reference to a user control created in the MyUserControl.ascx file.

Create an instance variable for the user control, using the control's class name. The class will be part of the ASP namespace.

Example: If you want to create an instance of the user control declared as class Spinner, you use syntax such as the following:

VB

Protected Spinner1 As ASP.Spinner

C#

Protected ASP.Spinner Spinner1

- Create an instance of the user control in code by calling the LoadControl method.
- Assign property values as necessary, and then add the control to the ControlCollection collection of a container on the page, such as a PlaceHolder control.

8.2.5 How to Create Templated ASP.NET User Controls

You can create user controls that implement templates, an ASP.NET feature that allows the separation of control data from its presentation. A templated control does not provide a user interface. Instead, it is written to implement a naming container and to include a class whose properties and methods are accessible to the host page.

The user interface for the user control is supplied by a page developer at design time. The developer creates templates of the type defined by the user control, and can then adds controls and markup to the templates.

To create a template user control

1. In the .ascx file, add an ASP.NET PlaceHolder control where you want the template to appear.
2. In the user control's code, implement a property of type ITemplate.
3. Define a server control class that implements the INamingContainer interface as a container in which to create an instance of the template. This is called the template's naming container.
4. Apply the TemplateContainerAttribute to the property that implements ITemplate and pass the type of the template's naming container as the argument to the attribute's constructor.
5. In the control's Init method, repeat the following steps one or more times:
   - Create an instance of the naming container class.
   - Create an instance of the template in the naming container.
   - Add the naming container instance to the Controls property of the PlaceHolder server control.

Notes

8.2.6 How to Create ASP.NET User Controls

You create ASP.NET user controls in much the same way that you design ASP.NET Web pages. You can use the same HTML elements and controls on a user control that you do on a standard ASP.NET page. However, the user control does not have html, body, and form elements; and the file name extension must be .ascx.

To create an ASP.NET user control

- Open the Web site project to which you want to add user controls. If you do not already have a Website project, you can create one.
- On the Website menu, click Add New Item.
- The Add New Item dialog box appears.
- In the Add New Item dialog box, under Visual Studio installed templates, click Web User Control.
- In the Name box, type a name for the control.
- By default, the .ascx file name extension is appended to the control name that you type.
- From the Language list, select the programming language that you want to use.
- Optionally, if you want to keep any code for the user control in a separate file, select the Place code in separate file check box.
- Click Add.

8.2.7 How to Include ASP.NET User Controls in Web Pages

Adding an ASP.NET user control to a Web page is similar to adding other server controls to the page. However, you must be sure to follow the procedure below so that all of the necessary elements are added to the page.

To add an ASP.NET user control to a Web page

1. In Visual Web Developer, open the Web page to which you want to add the ASP.NET user control.
2. Switch to Design view.
3. Select your custom user control file in Solution Explorer, and drag it onto the page.

8.2.8 Creating Reusable Elements with ASP.NET User Controls

ASP.NET user controls let you encapsulate the functionality of multiple server controls in a unit. User controls are made up of one or more ASP.NET server controls—Button controls, TextBox controls, and so on—along with any code that is required for the controls to perform the function that you want to accomplish. The user control can also include custom properties or methods that reveal the functionality of the user control to a container, which is typically an ASP.NET page. In this walkthrough, you will create an ASP.NET user control that acts as a picker control. The picker control has two lists, with a set of choices in one list (the source). Users can select items in the SourceList list, and then add the items to the Target List.

It has three parts, as follows:

- In the first part, you will create the basic user control, adding controls and code.
- In the second part, you will create a new ASP.NET page (the host page), and then add the user control to the ASP.NET page.

Notes

- In the third part, you will add custom properties and methods to the user control so that you can interact with the control from the host page.

Tasks illustrated in this include the following:

- Creating a user control and adding ASP.NET server controls to the user control.
- Creating properties and a method in the user control.
- Adding a user control to a host page.
- Adding code to the host page to handle the user control.

Creating the User Control

Creating a user control is similar to creating an ASP.NET Web page. In fact, a user control is effectively a subset of an ASP.NET page and it can include most of the types of elements that you put on an ASP.NET page.

To create a user control

1. In Solution Explorer, right-click the name of the Website, and then click Add New Item.
2. In the Add New Item <Path> dialog box, under Visual Studio installed templates, click Web User Control.
3. In the Name box, type ListPicker.

   The user control file will have an .ascx extension, which is added automatically to ListPicker.
4. In the Language list, select the programming language that you prefer to work in.
5. Click Add.

The new control is created and is opened in the designer. The markup for the control looks similar to that for a page, with one important exception: there is no Page directive at the top of the page. Instead, there is an Control directive, which identifies the file to ASP.NET as a user control.

Adding Server Controls to the User Control

In this part of the walkthrough, you will add the controls that make up the user interface for the user control.

To add server controls

1. Switch to Design view.(If you are working with a code-behind page, switch to the ListPicker.ascx control and then switch to Design view.)
2. On the Table menu, click Insert Table.
3. Use the Insert Table dialog box to create a table with one row and three columns, and then click OK.

   You are creating the table to hold the controls; that is, a layout table.
4. In the table, in the left column, type Available, and then press ENTER to create a new line.
5. In the right column, type Selected, and then press ENTER to create a new line.
6. From the Standard group in the Toolbox, drag the following controls onto the table and set their properties as indicated.

Notes

Limitations of User Controls in ASP.NET?

As the user controls are not compiled into assemblies, they have the following limitations:

1. A copy of the control must exist in each Web application project in which the control is used.
2. User controls cannot be loaded in the Visual Studio .NET Toolbox; instead, you must create them by dragging the control from Solution Explorer to the Web form.
3. User control code is initialized after the Web form loads, which means that user control property values are not updated until after the Web form's Load event.

Advantages of Using User Controls in ASP.NET:

1. User controls give more functionality than the ordinary controls since you can combine the other controls to create user controls. User controls can be created that are specific to the application that you create.
2. So you can add unique functionalities to the control. These user controls are easy to add to any web page once you created. So there is no need to code again for the same functionality if it is used in another page of the web application. It is also possible to convert the whole web pages into a user control with some minor modifications and use it anywhere.
3. By using the user control for some functionality you can separate the design work and the back-end work so that you can engage your design and coding team separately and efficiently.
4. If you use user control the development is modularized which means you have different functionalities available by using just a simple tag in your web page. This speeds up the development process.


Caution

When you develop a custom server control, you must include it in a namespace. If you do not, it will not be accessible from an ASP.NET page.

Self Assessment Questions

Multiple Choice Questions

1. …….. are one of the things that make developing with ASP.NET.
   (*a*) Privatization (*b*) Page level
   (*c*) Server controls (*d*) None of these.
2. We may use data binding to fill lists with selectable items from an ……
   (*a*) program interface (*b*) paging
   (*c*) XML program (*d*) imported data source
3. Each selectable item in a List Box control is defined by a……..
   (*a*) listItem element (*b*) markup language
   (*c*) encapsulation (*d*) synchronization
4. …….requirements as close as possible in a data model.
   (*a*) ordered collection (*b*) real-world data
   (*c*) collaborate (*d*) Infrastructure

True or False

5. User controls can be loaded in the Visual Studio.
   (*a*) True (*b*) False

Notes

8.3 Data Binding

Data binding is merely a program's ability to bind some members of an aim module to the members of a data source. We may use statistics binding to fill lists with selectable substance from an imported data cause, like a database, an XML file, or a script.

The following controls are list controls which support data binding:

- asp: RadioButtonList
- asp: CheckBoxList
- asp: DropDownList
- asp: Listbox

The selectable items in each of the above controls are usually defined by one or more asp:ListItem controls, like this:

```
<html>
<body>
<form runat= "server">
<asp:RadioButtonList id= "countrylist" runat= "server">
<asp:ListItem value= "N" text= "Norway" />
<asp:ListItem value= "S" text= "Sweden" />
<asp:ListItem value= "F" text= "France" />
<asp:ListItem value= "I" text= "Italy" />
</asp:RadioButtonList>
</form>
</body>
</html>
```

However, with data binding we may use a separate source, like a database, an XML file, or a script to fill the list with selectable items.

By using an imported source, the data is separated from the HTML, and any changes to the items are made in the separate data source.

8.3.1 RadioButtonList

The RadioButtonList control is used to create a group of radio buttons.

Each selectable item in a RadioButtonList control is defined by a ListItem element!

Example:

```
<script runat= "server">
Sub submit(sender As Object, e As EventArgs)
label1.Text= "You selected " & radiolist1.SelectedItem.Text
End Sub
</script>
<!DOCTYPE html>
<html>
<body>
<form runat= "server">
```

Notes

```
<asp:RadioButtonList id= "radiolist1" runat= "server">
<asp:ListItem selected= "true">Item 1</asp:ListItem>
<asp:ListItem>Item 2</asp:ListItem>
<asp:ListItem>Item 3</asp:ListItem>
<asp:ListItem>Item 4</asp:ListItem>
</asp:RadioButtonList>
<br />
<asp:Button text= "Submit" OnClick= "submit" runat= "server"/>
<p><asp:Label id= "Label1" runat= "server"/></p>
</form>
</body>
</html>
```

8.3.2 CheckBoxList

The CheckBoxList control is used to create a multi-selection check box group.

Each selectable item in a CheckBoxList control is defined by a ListItem element!

Example:

```
<script runat= "server">
Sub Check(sender As Object, e As EventArgs)
dim i
mess.Text= "<p>Selected Item(s) :</p>"
for i=0 to check1.Items.Count-1
if check1.Items(i).Selected then
mess.Text+=check1.Items(i).Text + "<br/>"
end if
next
End Sub
</script>
<!DOCTYPE html>
<html>
<body>
<form runat= "server">
<asp:CheckBoxList id= "check1" AutoPostBack= "True"
TextAlign= "Right" OnSelectedIndexChanged= "Check"
runat= "server">
<asp:ListItem>Item 1</asp:ListItem>
<asp:ListItem>Item 2</asp:ListItem>
<asp:ListItem>Item 3</asp:ListItem>
<asp:ListItem>Item 4</asp:ListItem>
<asp:ListItem>Item 5</asp:ListItem>
<asp:ListItem>Item 6</asp:ListItem>
</asp:CheckBoxList>
```

Notes

```
<br/>
<asp:label id= “mess” runat= “server”/>
</form>
</body>
</html>
```

8.3.3 Dropdown List

The DropDownList control is used to create a drop-down list.

Each selectable item in a DropDownList control is defined by a ListItem element!

Example:

```
<script runat= “server”>
Sub submit(sender As Object, e As EventArgs)
mess.Text= “You selected “ & drop1.SelectedItem.Text
End Sub
</script>
<!DOCTYPE html>
<html>
<body>
<form runat= “server”>
<asp:DropDownList id= “drop1” runat= “server”>
<asp:ListItem>Item 1</asp:ListItem>
<asp:ListItem>Item 2</asp:ListItem>
<asp:ListItem>Item 3</asp:ListItem>
<asp:ListItem>Item 4</asp:ListItem>
<asp:ListItem>Item 5</asp:ListItem>
<asp:ListItem>Item 6</asp:ListItem>
</asp:DropDownList>
<asp:Button Text= “Submit” OnClick= “submit” runat= “server”/>
<p><asp:label id= “mess” runat= “server”/></p>
</form>
</body>
</html>
```

8.3.4 Listbox

The ListBox control is used to create a single- or multi-selection drop-down list.

Each selectable item in a ListBox control is defined by a ListItem element!

Example:

```
<script runat= “server”>
Sub submit(Sender As Object,e As EventArgs)
mess.Text= “You selected“ & drop1.SelectedItem.Text
End Sub
</script>
<!DOCTYPE html>
```

Notes

```
<html>
<body>
<form runat= "server">
<asp:ListBox id= "drop1" rows= "3" runat= "server">
<asp:ListItem selected= "true">Item 1</asp:ListItem>
<asp:ListItem>Item 2</asp:ListItem>
<asp:ListItem>Item 3</asp:ListItem>
<asp:ListItem>Item 4</asp:ListItem>
<asp:ListItem>Item 5</asp:ListItem>
<asp:ListItem>Item 6</asp:ListItem>
</asp:ListBox>
<asp:Button Text= "Submit" OnClick= "submit" runat= "server"/>
<p><asp:label id= "mess" runat= "server"/></p>
</form>
</body>
</html>
```

Task Write a program for code binding.

8.4 Understanding Two-Way Data Binding

One common example of two-way data binding is the DataSet/DataAdapter grouping. You can update the data in the DataSet, and then update the underlying data cause using the DataAdapter, or, if changes occur to the underlying data, you can refresh the DataSet using the Fill() method.

With ASP.NET 2.0's bi-directional data binding, you can eliminate the code required to update a data store by binding the data controls in your presentation layer to the data sources. Because the binding is bi-directional, when users modify the data in the controls the underlying data sources update when users post the web form back to the server. Typically, you have had to use code similar to the following to manage displayed data binding:

```
//Populate the data controls from a business object
txtCode.Text = emp.Code;
txtName.Text = emp.Name;
//Populate a business object from the data controls
emp.Code = txtCode.Text;
emp.Name = txtName.Text;
<%# Bind( "FieldName") %>
```

8.4.1 Sample Two-Way Data Binding Application

To test the framework, you need to build an application that uses it. As a simple example, suppose you have an Employee entity, represented by an EmployeeDO class. This class contains three private fields and corresponding exposed public properties. It implements the IBusinessEntity interface and extends the Component class. IBusinessEntity is a base interface that all entities in this application extend, but this sample application deals exclusively with the Employee entity.

Notes

Here is the code for the IBusinessEntity interface:

```
public interface IBusinessEntity
{
Object Code
{
get;
set;
}
}
```

The IBusinessEntity interface exposes one property called Code, which functions as a unique ID. Using the IBusinessEntity interface as the base interface or contract for all entities ensures that you can reuse the data binding framework code with any business object. Here is the code for the EmployeeDO class:

```
public class EmployeeDO : Component, IBusinessEntity
{
private object code;
private string empName;
private string empAddress;
public Object Code
{
get {return code; }
set {code = value; }
}
public String EmpName
{
get {return empName; }
set {empName = value; }
}
public String EmpAddress
{
get {return empAddress; }
set {empAddress = value; }
}
}
```

Create a Container

Next, you need a container in which to store the data binding information. You will use this information to bind the controls to the appropriate property of the business object at runtime. This DataBindingRegister class serves that purpose. It holds the data binding information for each control along with the corresponding property of the business object to which the control will be bound.

Here is the source code for the DataBindingRegister class:

```
Public class DataBindingRegister
{
```

Notes

```
#region Fields
protected string businessObj;
protected string objPropertyName;
protected Control ctrlObj;
protected string ctrlPropertyName;
#endregion
#region Properties
public string BusinessObject {
get { return businessObj; }
set { businessObj = value; }
}
public string BusinessObjectPropertyName
{
get { return objPropertyName; }
set { objPropertyName = value; }
}
public Control ControlObject
{
get { return ctrlObj; }
set { ctrlObj = value; }
}
public string ControlObjectPropertyName {
get { return ctrlPropertyName; }
set { ctrlPropertyName = value; }
}
#endregion
}
```

Build the UI

Finally, you need to build the user interface, which contains a Label that displays the employee's code, two TextBoxes that display the employee's name and address respectively, and a Button control.

The following code snippet shows the controls used in the .aspx file for the sample application:

```
<asp:label id=lblEmpCode runat= "server"
Text='<%# DataBinder.Eval(employeeDO, "Code") %>'>
</asp:label>
<asp:textbox id=txtEmpName runat= "server"
Text='<%# DataBinder.Eval(employeeDO, "EmpName") %>'>
</asp:textbox>
<asp:textbox id=txtEmpAddress runat= "server"
Text='<%# DataBinder.Eval(employeeDO, "EmpAddress") %>'>
</asp:textbox>
```

Notes

```
<asp:Button ID= "Button1" runat= "server"
OnClick= "Button1_Click" Text= "Button" />
```

8.4.2 Using the Data Binding Framework

To use this data binding framework you create an array of DataBindingRegister instances and populate them with the necessary data binding information in the Page_Load event of the web form. This array is then added to an instance of the DataBindingRegisterCollection class that holds a collection of DataBindingRegister instances:

```
void Page_Load(object sender, EventArgs e) {
DataBindingRegister [] dataBindingRegister = new
DataBindingRegister[2];
// You would normally store the following binding information
// in an XML file
dataBindingRegister[0] = new DataBindingRegister();
dataBindingRegister[0].ControlObject = this.txtEmpName;
dataBindingRegister[0].ControlObjectPropertyName = "Text";
dataBindingRegister[0].BusinessObject= "employee";
dataBindingRegister[0].BusinessObjectPropertyName = "EmpName";
dataBindingRegister[1] = new DataBindingRegister();
dataBindingRegister[1].ControlObject = this.txtEmpAddress;
dataBindingRegister[1].ControlObjectPropertyName = "Text";
dataBindingRegister[1].BusinessObject = "employee";
dataBindingRegister[1].BusinessObjectPropertyName =
"EmpAddress";
//Now, add the data binding information to the
// Data Binding Collection Register
this.bindingManager.DataBindings.Add(dataBindingRegister);
if (!IsPostBack)
{
employeeDO = ReadData();
this.bindingManager.BusinessEntity = employeeDO;
this.bindingManager.Direction =
BindingDirection.FromUIToBusinessObject;
this.bindingManager.Bind(this);
}
}
```

After storing the data binding information you can bind data from the controls in the user's interface to the corresponding properties of the business object and vice-versa using the BindingManager.Bind() method. This method accepts an instance of the Page class as a parameter, iterates through the collection of databound control instances and then binds or un-binds data—depending on the binding direction set earlier using the BindingDirection enumeration. The Bind() method returns an IBusinessEntity instance.

```
IBusinessEntity IBindingManager.Bind(Page form)
{
```

Notes

```
foreach (DataBindingRegister dbR in
this.dataBindingCollection)
{
Control control = dbR.ControlObject;
if (control.ID != null) {
if (this.Direction ==
BindingDirection.FromUIToBusinessObject)
{
control.DataBind();
}
else if (bindingDirection ==
BindingDirection.FromBusinessObjectToUI)
{
PropertyInfo propertyInfo = control.GetType().
GetProperty(dbR.ControlObjectPropertyName,
BINDING_FLAGS);
object obj = propertyInfo.GetValue(control, null);
propertyInfo = this.BusinessEntity.GetType().
GetProperty(dbR.BusinessObjectPropertyName,
BINDING_FLAGS);
propertyInfo.SetValue(
this.BusinessEntity, obj, null);
}
}
}
return this.BusinessEntity;
}
```

The preceding example calls the Bind() method twice: once in the Page_Load event and once in the Button Click event. The Binding Direction enum will hold one of two values: either From Business Object To UI or From UI To Business Object.

That completes the application. Upon execution, it displays the default data bound to the controls in the Page_Load event. And if you enter new values in the text boxes and then click on the Button control, the form displays the name and address entered in the text boxes after a postback. In both cases, the data binding takes place using the Bind() method, which executes once in each direction.

Data binding in ASP.NET 1.x was unidirectional, i.e., the control could in no way update the underlying data source to which it was bound. While ASP.NET 2.0 introduced two-way data binding, not all the ASP.NET controls supported i; you need to write custom data binding logic such as that discussed in this topic to make two-way data binding available for all the controls.

Task Create a two way binding application.

Notes

8.5 Data-Bound Controls

ASP.NET provides a rich set of data-bound server controls. These controls are easy to use and provide a Rapid Application Development (RAD) Web development. You can categorize these controls in two groups: single-item data-bound control and multi-item data-bound controls.

You use the single-item data-bound controls to display the value of a single item of a database table. These controls do not provide direct binding with the data source. The highlighted part of given table shows an item of a database table. You use the Text, Caption, or Value property of these controls to show the data a field.

Table 8.1: Data-bound Controls

AMAR	234, G. Road	12398
SUNIL	5443, NY	89433
AJAY	P. Rd	54323

8.5.1 An Item

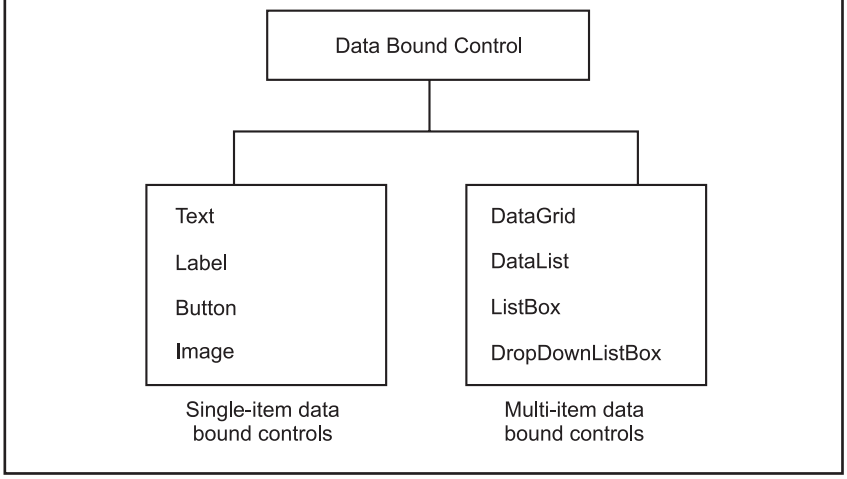
We use the multi-item data bound controls to display the entire or a partial table. These controls provide direct binding to the data source. We use the Data Source property of these controls to bind a database table to these controls (Table 8.2).

Table 8.2: Multiple-item Controls

AMAR	234, G. Road	12398
SUNIL	5443, NY	89433
AJAY	P.RD	54323

Some examples of multi-item data-bound controls are DataGrid, ListBox, DataList, DropDownList, and so on some of the data bound controls (See Figure 8.1).

Figure 8.1: Data-bound Control



In ASP.NET, you create these controls using a <asp:controlName> tag.

describes some common data-bound server-side controls

8.5.2 ASP .NET Data-bound Controls

Notes

Table 8.3: Data-bound Controls

CONTROL	ASP.NET CODE	DESCRIPTION
DataGrid	<asp:DataGrid>	Display a database (through ADO.NET) in a scrollable grid format and supports selection, adding, updating, sorting, and paging
DataList	<asp:DataList>	Displays data in templates and style format
ListBox	<asp:ListBox>	Displays list box, which can be associated to ADO.NET data field to display data in a list format
DropDownList	<asp:DropDownList>	Displays DropDownList control, which can be used to display ADO.NET data source data in a drop down combo box format
CheckBox	<asp:CheckBox>	Display single check box, which can be connected to an item of the ADO.NET data source
CheckBoxList	<asp:CheckBoxList>	Display list of check boxes that can be connection to the to the ADO.NET data source
Repeater	<asp:Repeater>	Displays a templated data-bound list
TextBox	<asp:TextBox>	Displays a text box, which can be used to display ADO.NET using its Text property

Case Study

Advanced Telemetry

Advanced Telemetry began by manufacturing and marketing its flagship remote energy-monitoring software—and paying for the servers required for its customers' data. The relationship between adding customers and paying for more rack space was a serious barrier to expanding market share. Advanced Telemetry then moved its IT infrastructure to Windows Azure, using Microsoft SQL Azure and Windows Azure. It also switched to a new business model whereby OEMs can license an instance of its telemetry software in Windows Azure, and then offer the product in new markets. This move reduced IT infrastructure expenses by 75%, introduced new revenue from customization, reduced marketing costs by 80% and turned the company into an agile, profitable operation that can respond quickly to customers in any market.

Situation

Tom Naylor, Founder and Chief Technology Officer of Advanced Telemetry, brought together a team of developers in San Diego, California, to explore innovations in telemetry—technology that enables remote measurement and reporting of information—for transferring data over the Internet. The company received venture capital funding to launch in 2007, and Advanced Telemetry went to market with its flagship product, the EcoView Energy Management System.

Targeted toward restaurants and convenience stores, the system includes an on-premises touch panel that communicates wirelessly with intelligent thermostats and power meters, and transmits energy consumption, temperature, and other environmental data over the Internet

Contd...

Notes

to the company's web tier, middleware tier, and data storage servers. The middleware tier, which Advanced Telemetry built using Microsoft ASP.NET and Microsoft SQL Server 2005 data management software (since upgraded to Microsoft SQL Server 2008), is a complex and extensible set of business rules that can be applied to the data to determine appropriate responses and actions. For example, a customer could specify that if a temperature reaches a certain threshold, the air conditioners will activate at a specific setting.

The touch panel displays real-time and historical usage data and stores the customer's unique business rules. Building managers can use the touch panel to set energy schedules and manage energy consumption within a building, or regional managers can monitor and control many buildings using the system's web portal, called EcoView Web. Alternatively, customers can have Advanced Telemetry perform the remote monitoring service on their behalf.

Low cost and easy to install, EcoView brings the benefits of sophisticated energy management to medium and small buildings to dramatically reduce resource consumption. The product was well received when it launched, and business grew steadily. Between October 2009 and November 2010, Advanced Telemetry doubled its customer base. Today, it serves 1,000 customers with a total of 2,000 installed sites that transmit approximately 150,000 unique points of data every hour. This success, however, came with a price.

Breaking Ties to an IT Infrastructure

"Like most startups, we began with a single product and a local infrastructure, delivering an end-to-end solution under a brand name, EcoView," says Naylor. "In this model, we were the manufacturer and did the marketing, channel development, and customer support for ourselves."

It quickly became clear that this business model, which entailed managing a local infrastructure, was not a scalable or practical way for Advanced Telemetry to grow its business. The EcoView Energy Management System generates a tremendous amount of data, which it records to the middleware and data storage tiers. The bulk of the data is historical information that consists of unique data points recorded at each customer site, such as an energy demand change. The rest of the data is relational in nature, such as customer configurations and metadata. Even though the historical data is more suited to table storage, it ended up in the company's SQL Server databases anyway because that was the only database solution Advanced Telemetry had.

"We were forced to cull the data from the SQL Server databases because they were getting so unwieldy, and we did not want to keep paying for more servers," says Naylor. "We were only able to store about six months of information. Naturally, we did not like to lose information that could be useful for our customers in some way in the future."

Advanced Telemetry came to see its reliance on a physical infrastructure as a major impediment towards profitability, growth, agility, and even customer service. The company moved its burgeoning amount of servers to a rack-mounted infrastructure hosted at a collocation vendor, but even then, it was still concerned about the unavoidable link between adding customers and paying for more rack space. Also, it was time consuming to set up a customer on the system, a process that could take weeks of effort. Some EcoView customers were unhappy about sharing their data servers with others, but Advanced Telemetry could not afford to set up a single instance of its middleware and data storage for individual customers.

"It was too expensive and difficult for customers to have separate banks of servers that we would remotely support," says Naylor. "Looking forward, all we could see was the increasing cost of maintaining an IT infrastructure eating into the profitability of our business. The overhead that we were carrying to drive sales and provide support was not going to get us around the corner as quickly as our investors wanted. We had to rethink our business model."

Contd...

Notes

Changing the Business Model

Naylor had always preferred the idea of an original equipment manufacturer (OEM) model, one that would showcase the company's uniquely adaptable and extensible middleware. "We wanted to leverage our core telemetry infrastructure for a number of applications," he explains. "However, like a lot of startup companies, we needed immediate access to the market, so we developed EcoView. We just got stuck there. I wanted to back out of our direct market approach and deliver our product to new verticals through OEM licenses."

The OEM model would absolve the company of the burden of directly marketing and supporting its product to customers, freeing up time and resources to build more features into its telemetry software. "We tried to build maximum extensibility into our middleware, but there's always room for more flexible configurations to accommodate as many requests as we can from customers," says Naylor. "It would have been nice to have more time to devote to that sort of development work."

Also, the new business model would give the company an opportunity to enter untapped vertical markets that it simply did not have the resources to acquire on its own. "Our core technology is designed to accommodate any remote monitoring scenario," says Naylor. "The OEM model, whereby licensees could take our product to market in new verticals using our core technology, was so compelling that we knew we had to make the change."

But there was a significant, and by now all too familiar, stumbling block to moving ahead with this key business strategy: the company's reliance on a physical IT infrastructure. Even if an OEM was interested in licensing Advanced Telemetry software, it was obvious that it would quickly come up against the same issues that plagued the startup from the beginning. "We had to move our infrastructure to a cloud-computing model where our application and data could reside on remotely hosted servers for which the OEMs would not be responsible," says Naylor. "So along with a new business model, we were suddenly contemplating a new computing paradigm."

Solution

Advanced Telemetry first considered storing its historical data on BigTable from Google, but that still left its middleware and web tiers residing on collocated servers. The company wanted a complete solution for running its Windows-based application and storing its data in the cloud. Naylor found what the company needed with Windows Azure from Microsoft. Moving its entire IT environment to Windows Azure would mean that Advanced Telemetry could take advantage of the following components offering integrated cloud computing services:

- Windows Azure: Provides a Windows-based environment for running applications and storing data on servers in Microsoft data centers. Advanced Telemetry could use Windows Azure cloud compute services to run the company's web presentation tier and remote monitoring and control middleware. And it could use Windows Azure Storage to store its customers' nonrelational historical data.
- Microsoft SQL Azure: Provides a relational database service in the cloud based on SQL Server 2008. Advanced Telemetry could use SQL Azure to store customers' configurations and metadata.

"With Windows Azure, I realized right away that this was more than a new computing environment, it was a key business enabler," says Naylor. "It was easy to see that basing our operations on an Internet connection instead of racks of servers running in our facility would literally transform our business." Advanced Telemetry also liked the fact that Windows

Contd...

Notes

Azure offers a computing environment already familiar to its developers who work with the Microsoft Visual Studio 2008 Professional development system.

In August 2009, Advanced Telemetry began migrating its web-based presentation layer to Windows Azure. The company also upgraded its web management portal for the middleware using Microsoft Silverlight 2.0, a cross-browser implementation of the Microsoft .NET Framework for building and delivering rich, interactive web applications.

Then Advanced Telemetry proceeded with the more complex middleware. In Windows Azure, each application can have multiple instances. Each instance is assigned a virtual machine (VM) that can run part or all of the application's code. The company's developers can divide the application's code into web roles (where it is necessary for the software to accept and process Intelligent Terminal Transfer Protocol requests using Internet Information Services, or IIS—a component of the Windows Server 2008 operating system) and worker roles (which do not use IIS). The developers simply tell Windows Azure how many instances of each role to run, and Windows Azure creates the VMs and provides built-in load balancing to equalize requests coming into the application's web role instances. "This capability is important, because it allows us to do upgrades to various parts of the middleware without touching the other parts," says Naylor.

Questions

1. What is the moral of the above case study?
2. Which technology was used in advanced telemetry?

Self Assessment Questions

6. ASP.Net use for ……

   (*a*) Web development (*b*) Window Development

   (*c*) Both a and b (*d*) None of these.

7. The file name of user control extension is …….

   (*a*) aspx (*b*) asjx

   (*c*) .ascx (*d*) None of these.

8. Which one is not compatible language of ASP.NET

   (*a*) Visual Basic.Net (*b*) C#

   (*c*) Wamp server (*d*) None of these.

9. Developers can use ……when creating an ASP.NET application.

   (*a*) Window forms or XML Web services

   (*b*) C#

   (*c*) Web Forms or XML Web services

   (*d*) None of these.

10. ASP.NET offers the IHttpHandler and ……

    (*a*) HttpHandler (*b*) IHttpModule interfaces

    (*c*) IIHttpModule interfaces (*d*) None of these.

Notes

11. …….establishes a common interface and foundational functionality, which Specialized subclasses can inherit, modify, and supplement.

(*a*) Super class (*b*) Master class

(*c*) Both (*d*) None of these.

12. You can create user controls that implement……. an ASP.NET feature that allows the separation of control data from its presentation.

(*a*) Templates (*b*) Master class

(*c*) Both (*d*) None of these.

13. ASP.NET user controls let you encapsulate the functionality of …….controls in a unit.

(*a*) Server (*b*) Multiple Server

(*c*) Both (*d*) None of these.

14. Which one support data controls list controls binding……..

(*a*) Dropdown List (*b*) RadioButtonList

(*c*) Checkbox List (*d*) All of these.

15. The List Box control is used to create a single- or multi-selection drop-down list.

(*a*) single or multi (*b*) Multi

(*c*) Single (*d*) None of these.

8.6 Summary

- ASP.NET provides easy-to-use application and session-state facilities that are familiar to ASP developers and are readily compatible with all other .NET Framework APIs.
- Server controls are one of the things that make developing with ASP.NET so simple and powerful at the same time.
- An ASP.NET Web user control is similar to a complete ASP.NET Web page (.aspx file), with both a user interface page and code.
- Data binding is simply a program's ability to bind some members of a target component to the members of a data source.
- To use this data binding framework you create an array of DataBindingRegister instances and populate them with the necessary data binding information in the Page_Load event of the web form.

8.7 Keywords

Active Server Pages (ASP): ASP known as Classic ASP or ASP Classic, was Microsoft's first server-side script engine for dynamically generated web pages.

API: It stands for Application Program Interface. It is a way through which using computers becomes quite easy.

Data Binding: It is general technique that binds two data/information sources together and maintains synchronization of data.

Extensible Markup Language (XML): It is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

Notes

User Controls: User controls are one of ASP.NET methods to increase reusability of code, implement encapsulation and reduce maintenance.

Lab Exercise

1. Create a custom user control in ASP.NET.
2. Write an XML program to create a DTD for the college.

8.8 Review Questions

1. Explain about page sub-classing with example.
2. What is the objective of user control? Explain with example.
3. How we can include a User Control in an ASP.NET Web Page?
4. How can create user control and add server control to the user control?
5. What are the limitation and advantage of user control?
6. What do you know about data bound control explain with example?
7. How can use the Data Binding Framework?
8. Write a program of data binding using dropdown list.
9. Write a program by using sub class.
10. How to create template of User Controls in ASP.NET?

Answers for Self Assessment Questions

1. (*c*)	2. (*d*)	3. (*a*)	4. (*b*)	5. (*b*)
6. (*a*)	7. (*c*)	8. (*c*)	9. (*c*)	10. (*b*)
11. (*a*)	12. (*a*)	13. (*b*)	14. (*d*)	15. (*a*)

8.9 Further Readings

Books

Professional ASP.NET by Jon Galloway, Phil Haack, and Brad Wilson

Net: Asp the Complete Reference, by Matthew MacDonald

Online link

www.telerik.com/products/aspnet-mvc.aspx

Unit 9: The Database Model

Notes

CONTENTS

Notes

Objectives

After studying this unit, you will be able to:

- Describe the concept of database model
- Discuss the hierarchical database model
- Explain the network database model
- Understand the relational database model
- Define the types of databases
- Explain that how create a basic object model
- Describe the creating the user interface

Introduction

We have the benefit of database systems and established how they are superior to the earlier file oriented data systems. We caught a glimpse of the features of database systems that produce several benefits. Database systems reduce data redundancy, integrate corporate data, and enable information sharing among the various groups in the organization. Now you are ready for an initial, formal definition of a database.

A data model represents the data requirements of an organization. You can diagrammatically show a data model with symbols and figures. Data for an organization reside in a database. Therefore, when designing a database, you first create a data model. The model would represent the real-world data requirements. It would show the arrangement of the data structures.

Database software has evolved to support different types of data models. As we try to represent real-world data requirements as close as possible in a data model, we come up with a replica of the real-world information requirements. It turns out that we can look at data requirements and create data models in a few different ways. At this stage, let us survey a few leading data models. Over time, different vendors have developed commercial database management systems to support each of these common data models.

Let us examine the following definition:

A database is an ordered collection of related data elements intended to meet the information needs of an organization and designed to be shared by multiple users.

The key terms in the definition:

Ordered Collection: A database is a collection of data elements. Not just a random assembly of data structures, but a collection of data elements put together deliberately with proper order. The various data elements are linked together in the most logical manner.

Related Data Elements: The data elements in a database are not disjointed structures without any relationships among them. These are related among themselves and also pertinent to the particular organization.

Information Needs: The collection of data elements in a database is there for a specific purpose. That purpose is to satisfy and meet the information needs of the organization. In a database for a bank, you will find data elements that are pertinent to the bank's business. You will find customer's bank balances and ATM transactions. You will not find data elements relating to a student's major and examination grades that belong in a database for a university. You will not find a patient's medical history that really belongs in a database for a medical centre.

Notes

Shared: All authorized users in an organization can share the information stored in its database. Integrated information is kept in the database for the purpose of sharing so that all user groups may collaborate and accomplish the organization's objectives.

9.1 Concept of Database Model

You are now opening to be grateful for the significance of the database move toward. You are discerning the major benefits of mounting and using applications in a database environment. Before proceeding further, allow us evaluation a few fundamental concepts and become recognizable with some key expressions.

9.1.1 Data Repository

All data in the database reside in a data repository. This is the data storage space part where physical data files are kept. The data depository contains the physical data. Mostly, it is a central place of storage for the data contented.

9.1.2 Data Dictionary

The data repository contains the authentic data. Allow us say that you desire to keep data about the customers of your business in your database. The arrangement of a customer's data could include fields such as customer name, customer address, city, state, zip code, phone number, and so on. Data about a particular customer could be as follows in the respective fields: Jane Smith/1234 Main Street/Piscataway/NJ/08820. There are two aspects of the data about customers. One aspect is the structure of the data consisting of the field names, field sizes, data types, and so on. This part is the structure of the data for customers. The other part is the actual data for each customer consisting of the actual data values in the various fields.

The first part relating to the structure resides separately in storage, and this is called the data dictionary or data catalog. A data dictionary contains the structures of the various data elements in the database. It also contains the relationships among data elements. The other part relating to the actual data about individual customers resides in the data repository. The data dictionary and the data repository work together to provide information to users.

9.1.3 Database Software

Are Oracle and Informix databases? Oracle and Informix are really the software that manages data. These are database software or database management systems. Database software supports the storing, retrieving, and updating of data in a database. Database software is not the database itself. The software helps you store, manage, and protect the data in a database.

9.1.4 Data Abstraction

Consider the example of customer data again. Data about each customer consist of several fields such as customer name, street address, city, state, zip code, credit status, and so on. We can look at customer data at three levels. The customer service representative can look at the customer from his or her point of view as consisting of only the fields that are of interest to the representative. This may be just customer name, phone number, and credit status. This is one level. The next level is the structure of the complete set of fields in customer data. This level is of interest to the database designer and application programmer. Another level is of interest to the database administrator, who is responsible for designing the physical layout for storing the data in files on disk storage.

Now go through the three levels. The customer service representative is just interested in what he or she needs from customer data, not the entire set of fields or how the data is physically stored on disk storage. The complexities of the other two levels may be hidden from the customer

Notes

service representative. Similarly, the physical level of how the data is stored on disk storage may be hidden from the application programmer. Only the database administrator is interested in all three levels. This concept is the abstraction of data the ability to hide the complexities of data design at the levels where they are not required. The database approach provides for data abstraction.

9.1.5 Data Access

The database approach includes the fundamental operations that can be applied to data. Every database management system provides for the following basic operations:

- READ data contained in the database
- ADD data to the database
- UPDATE individual parts of the data in the database
- DELETE portions of the data in the database

Database practitioners refer to these operations by the acronym CRUD:

- C—Create or add data
- R—Read data
- U—Update data
- D—Delete data

Transaction Support Imagine the business functions of entering an order from a customer into the computer system. The order entry clerk types in the customer number, the product code, and the quantity ordered. The order entry program reads the customer data and allows the clerk to sight verify the customer data, reads product data and displays the product description, reads inventory data, and finally updates inventory or creates a back order if inventory is insufficient. All these tasks performed by the order entry program to enter a single order comprise a single order entry transaction. When a transaction is initiated it should complete all the tasks and leave the data in the database in a consistent state. That is, if the initial stock is 1000 units and the order is for 25 units, the stock value stored in the database after the transaction is completed must be 975 units. How can this be a problem? See what can happen in the execution of the transaction. First, the transaction may not be able to perform all its tasks because of some malfunction preventing its completion. Second, numerous transactions from different order entry clerks may be simultaneously looking for inventory of the same product. Database technology enables a transaction to complete a task in its entirety or back out intermediary data updates in case of malfunctions preventing completion.

9.2 Hierarchical Database Model

Allow us examine the data necessities for a typical developed company. Typically in manufacturing, you have major assemblies, with each main congress consisting of subassemblies, each subassembly consisting of parts, each part consisting of subparts, and so on. In your database for the developed company, you require to stay put data for the assemblies, subassemblies, parts, and subparts. And the data model for manufacturing operations must represent these data necessities.

Think about this data model. This model should show that an assembly contains subassemblies, a subassembly contains parts, and a part contains subparts. Immediately you can observe that this data model must be hierarchical in nature, diagramming the assembly at the top with subassembly, part, and subpart at successive lower levels.

Notes

In the business world, many data structures are hierarchical in nature. You can notice a hierarchy in department, product category, product subcategory, product line, and product. You can trace a hierarchy in division, subdivision, department, and employee. Figure 9.1 illustrates one such model showing the hierarchy of customer, order, and order line item. A customer may have one or more orders, and an order may have one or more line items, perhaps one line item for each product ordered.

Let us review the key features of the hierarchical model by referring to Figure 9.1 levels. Each data structure representing a business object is at one of the hierarchical levels.

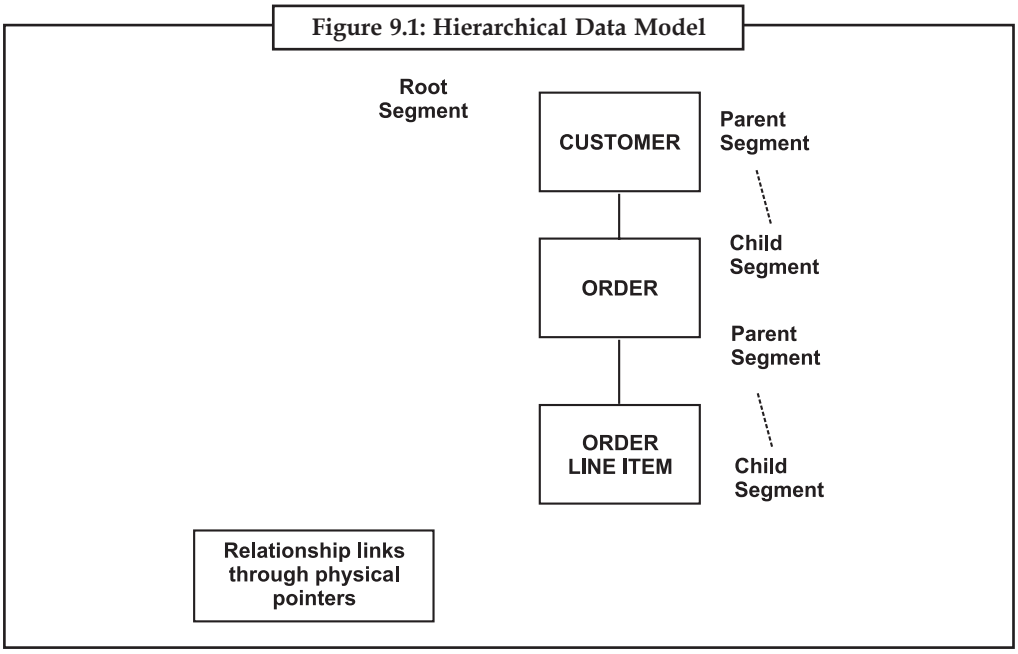
9.2.1 Parent-child Relationships

The relationship between each pair of data structures at levels after that to each other is a parent-child relationship. CUSTOMER is a parent data piece whose child is the ORDER data segment. In this arrangement, a child segment can have only one parent segment but one parent segment may have multiple child segments. You may want to separate orders into phone orders and mail orders. In that case, CUSTOMER may have PHONE ORDER and MAIL ORDER as two child segments.

9.2.2 Root Segment

The data segment at the top level of the hierarchy is known as the root data segment (as in an inverted tree).

Figure 9.1: Hierarchical Data Model



9.2.3 Physical Pointers

How are the orders of a particular customer linked in the implementation of the hierarchical data model? These linkages are by means of physical pointers or physical storage addresses embedded within physical records in the database. Physical pointers link records of the parent segments to those of the child segments by means of parent-child forward or backward pointers. Similarly, forward and backward physical pointers link records of the same segment type.

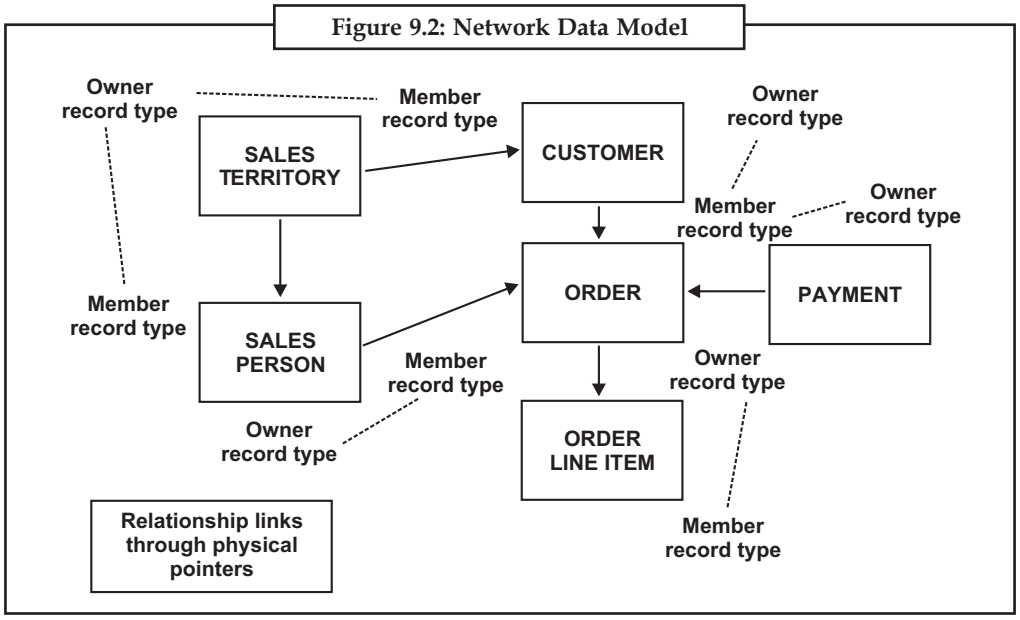
Notes

9.3 Network Database Model

The hierarchical data model represent well any business data that intrinsically contains levels one below the extra. We have just discussed how the developed application deals with hierarchical levels of plant record with assemblies broken down into buried components. The hierarchical data model suits this application well. However, in the real world, most data structures do not conform to a hierarchical arrangement. The levels of data structures do not fall into nice dependencies one below another as in a hierarchy. In the hierarchical data model, you have noticed that each data segment at any level can have only one parent at the next higher level. In practice, many sets of related elements may not be subjected to such restrictions.

Let us consider a common set of related data elements in a typical business. The data elements pertain to customers placing orders and making payments, salespersons being assigned, and salespersons being part of sales territories. All of these data elements cannot be arranged in a hierarchy. The relationships cross over among the data elements as though they form a network. Refer to Figure 9.2 and note how it represents a network arrangement and not a hierarchical arrangement. Observe the six data elements of sales territory, salesperson, customer, order, order line item, and payment as nodes in a network arrangement.

Figure 9.2: Network Data Model



The network data model overcomes some of the limitations of the hierarchical data model. The network data model is more representative of real-world information requirements than the hierarchical model. The network data model can represent most business information.

Let us go over the key features of the network model by referring to Figure 9.2.

9.3.1 Levels

As in most real-world situations, no hierarchical levels exist in the network model. The lines in a network data model simply connect the appropriate data structures wherever necessary without the restriction of connecting only successive levels as in the hierarchical model. Note the lines connecting the various data structures with no restrictions.

9.3.2 Record Types

In the network data model, each data structure is identified as an evidence type. For example, the CUSTOMER record type represents the data content of all customers. The ORDER record type represents the data content of all orders.

Notes

9.3.3 Relationships

The network data model expresses relationships between two record types by designating one as the owner record type and the other as the member record type. For each occurrence of an owner record type, there are one or more occurrences of the member record type. The owner record type may be reckoned as the parent and the member record type as the child. In a sense, the owner record type "owns" the corresponding member record type. Each member type with its corresponding owner record type is known as a set. A set represents the relationship between an owner and a member record type.

9.3.4 Multiple Parents

Look at the ORDER member record type. For ORDER there are two parents or owner records, namely, CUSTOMER and PAYMENT. In other words, for one occurrence of CUSTOMER, one or more occurrences of ORDER exist. Similarly, for one occurrence of PAYMENT there are one or more occurrences of ORDER. By definition, a hierarchical data model cannot represent this kind of data arrangement with two parents for one child data structure.

9.3.5 Physical Pointers

Just as in the case of the hierarchical data model, related occurrences of two different record types in a network model are connected by physical pointers or physical storage addresses embedded within physical record in the database. Physical pointers link occurrences of an owner record type with the corresponding occurrences of the member record type. Within each record type itself the individual occurrences may be linked to one another by means of forward and backward pointers.

9.4 Relational Database Model

This data model is greater to the earlier models. This stage, though, we want to begin the relational model as a greater data model that addresses the confines of the earlier data model.

The earlier hierarchical data model is appropriate for data structures that are naturally hierarchical, with each data structure placed at a certain level in the ladder. However, in the business arena, many of the data structures and their relationships cannot be readily placed in a hierarchical arrangement. The network data model evolved to dispense with the arbitrary restriction of the hierarchical model. Nevertheless, in both of these models, you need physical pointers to connect related data occurrences. This is a serious drawback because you have rewrite the physical addresses in the data records every time you reorganize the data, move the data to a different storage area, or change over to another storage medium. The relational model establishes the connections between related data occurrences by means of logical links implemented through foreign keys. Figure 9.3 illustrates the relational data model.

Let us note the key features of the relational data model by referring to Figure 9.3

9.4.1 Levels

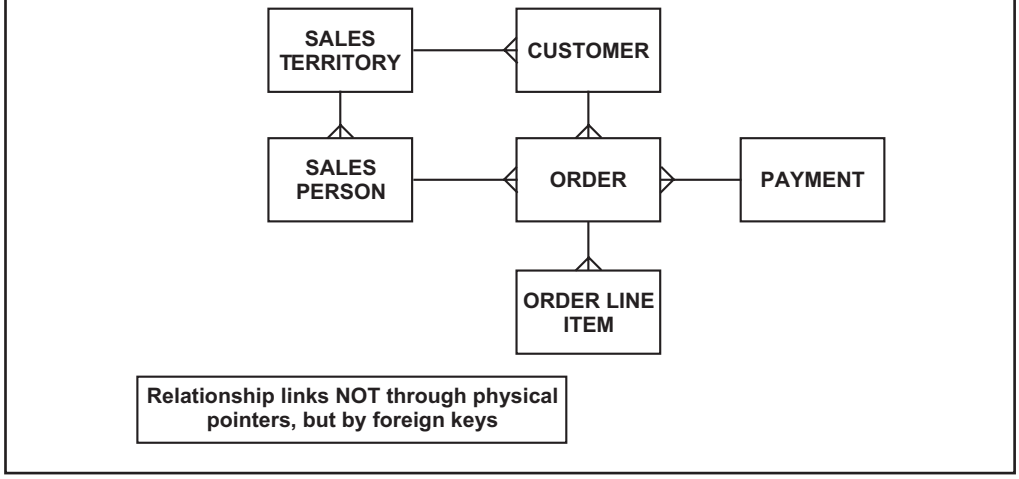
Just like the network data model, no hierarchical levels are present in the relational model. The lines in a relational data model simply indicate the relationships between the appropriate data structures wherever necessary without the restriction of connecting only successive levels as in the hierarchical model. As in the network model, note the lines connecting the various data structures with no restrictions.

9.4.2 Relations or Tables

The relational model consists of relations. A relation is a two-dimensional table of data observing relational rules. For example, the CUSTOMER relation represents the data content of all customers. The ORDER relation represents the data content of all orders.

Notes

Figure 9.3: Relational Data Model



9.4.3 Relationships

Consider the relationship between CUSTOMER and ORDER. For each customer one or more orders may exist. So this customer occurrence must be connected to all the related order occurrences. In the relational model, physical pointers do not establish these connections. Instead, a foreign key field is included in the ORDER data structure. In each of the order occurrences relating to a certain customer, the foreign key contains the identification of that customer. When you look for all the orders for a particular customer, you search through the foreign key field of ORDER and find those order occurrences with identification of that customer in the foreign key field.

9.4.4 No Physical Pointer

Unlike the hierarchical or the network data models, the relational model establishes relationships between data structures by means of foreign keys and not by physical pointers.

Did u know?

Dr. E. F. Codd is the father of the relational model, stipulated the rules and put this model on a solid mathematical foundation.

Self Assessment Questions

1. There are two aspects of the data about customers.
   (*a*) True (*b*) False
2. Oracle and Informix are not really the software that manages data.
   (*a*) True (*b*) False
3. The data repository contains the actual data.
   (*a*) True (*b*) False
4. …….. occurrences of an owner record type with the corresponding occurrences of the member record type.
   (*a*) ListItem element (*b*) Markup Language
   (*c*) Encapsulation (*d*) Physical pointers link
5. Physical pointers link records of the parent segments to those of the child segments by means of parent-child forward or ……..
   (*a*) NET provides (*b*) components
   (*c*) Backward pointers (*d*) Compatibility

Notes

9.5 Types of Databases

Through now you are influenced of the meaning of information for an association. You know that information is a key business asset and that it has to be managed, protected, and used like any other major asset. The corporate database that holds an organization's data is the original foundation for corporate information. Organizations are also faced with questions regarding how and where to hold the corporate data.

Where should an enterprise hold its data? Should all the corporate data be kept centrally in one place? If so, what are the advantages and disadvantages? What are the implications of this arrangement?

Organizations primarily adopt one of two approaches. If the entire database is kept in one centralized location, this type of database is a centralized database. On the other hand, if fragments of the database are physically placed at various locations, this type of database is a distributed database. Each type has its own benefits and shortcomings. Again, whether an enterprise adopts a centralized or a distribute approach depends on the organizational setup and the information requirements. Let us review the two types.

Caution

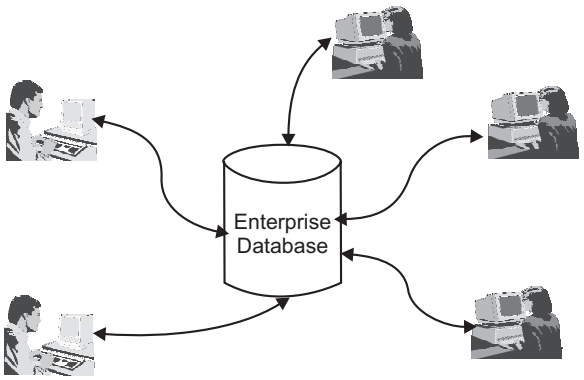
The corporate data should be divided into suitable fragments and the pieces kept at different locations

9.5.1 Centralized Database

Figure 9.4 illustrates a central database. Personalized databases are always centralized in one position. If your company has a centralized computer system, then the database must exist in in that central location. In the client/server building, the database resides on a server machine. The entire database may be kept on a single server machine and placed in a central location.

When all corporate data is in one place in a centralized database, companies find it easier to manage and administer the database. You can control concurrent accesses to the same data in the database easily in a centralized database. You can maintain security controls easily. However, if your company's operations are spread across remote locations, these locations must access the centralized database through communication links. Here, data availability depends on the capacity and dependability of the communication links.

Figure 9.4: Centralized Database



All data at a single site.

Data access from remote sites through communication links.

Easy to administer.

Uncertain data availability.

Common Examples:

Personal Database

Central Computer Database

Client/Server Database

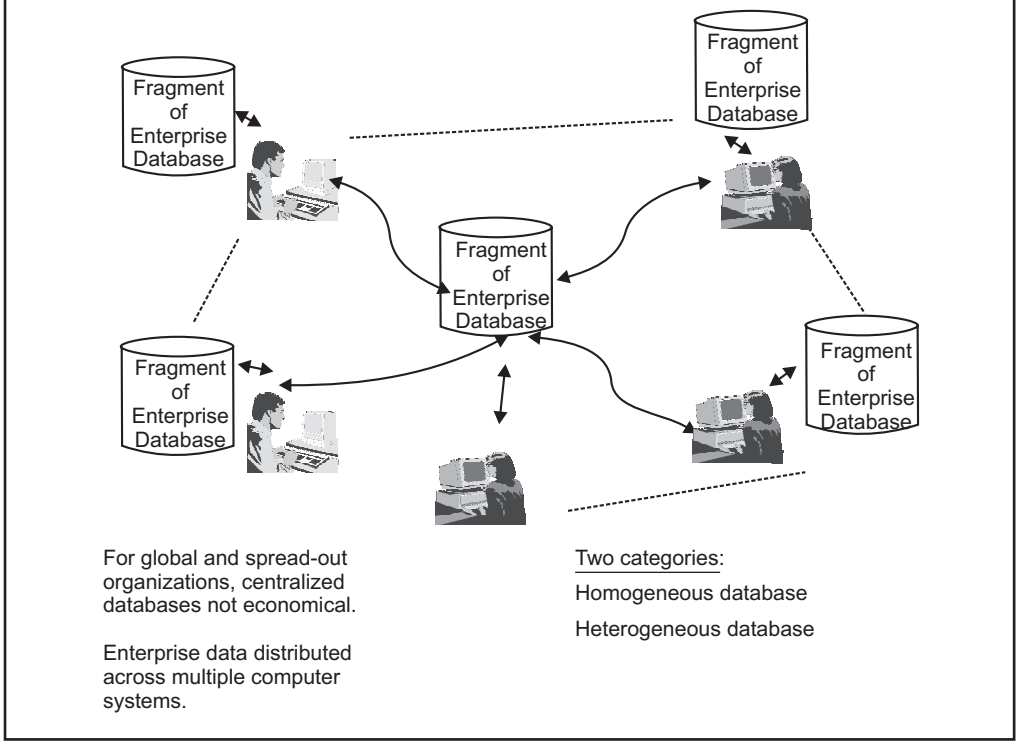
Notes

9.5.2 Distributed Database

Figure 9.5 shows how fragments of a corporate database are spread across remote locations. Global organizations or enterprises with widespread domestic operations can benefit from distributed databases. In such organizations computer development is also disseminated, with processing done locally at each location. A distributed database gets fragmented into smaller data sets. Normally, you would divide the database into data sets on the basis of usage. If a fragment contains data that are most relevant to one location, then that data set is kept at that location. At each location, a fragment of the enterprise data is placed based on the usage.

Each fragment of data at each position may be managed with the identical type of database organization system. For example, you may run Oracle DBMS at every location. In that case, you run your distributed database as a homogenous database. On the other hand, if you elect to manage the data fragments at different locations with different DBMSs, then you run your distributed database as a anthology of heterogeneous database systems. Assorted arrangement provides extra flexibility. However, heterogeneous distribution is difficult to coordinate and administer.

Figure 9.5: Distributed Database



9.6 Creating a Basic Object Model

The object model of Smalltalk is simple and uniform, everything is an object. However, this uniformity can still be a source of problem for programmers used to other languages.

The Rules of the Model

The design of the object model is based on a set of simple rules that are applied *uniformly*.

The rules are the following ones:

Rule 1: Everything is an object that has some *private data*.

Notes

Rule 2: Every object is instance of a class.

Rule 3: A class defines the behavior via *public* methods and the structure of its instances via instance variables which are *private* to the instances.

Rule 4: Each class is inheriting its behavior and structure description from a single superclass.

Rule 5: Objects *only* communicate via message passing (i.e., method invocation). When an object receives a message, the corresponding method is looked up in the *class of the receiver*, then if not found on this class continues in the class's superclasses.

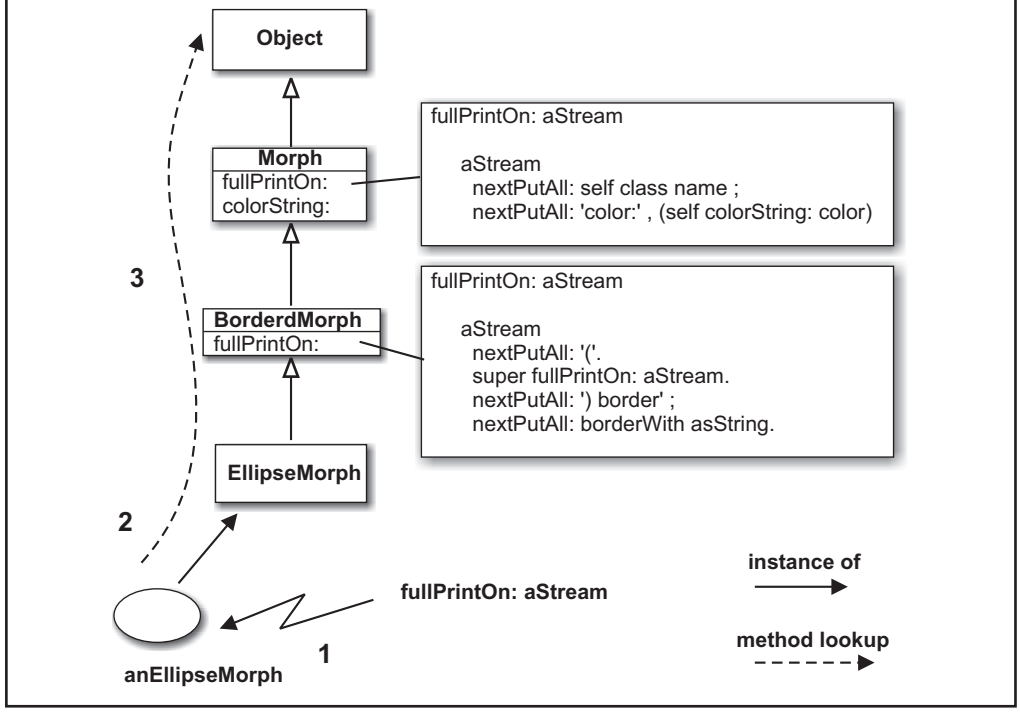
Rule 6: The class *Object* is the root of the inheritance tree (in Squeak this is *ProtoObject* the class that represents objects understanding the smallest set of messages).

Rule 7: Classes are instances too. They are instances of other classes called *metaclasses*.

9.6.1 Model Precisions

In Smalltalk we have only objects that are instance of classes. Class defines the structure of the instance in terms of instance variables, and methods.

Figure 9.6: Lookup of Messages Sent to Self Starts in the Class of the Receiver



9.6.2 Instance Variables

Instance variables are private to the instance itself, contrary to Java or C++. Even instances of the same class cannot access the instance variables of an object if this one did not define accessors methods. Instance variables are accessible by all the methods of the class and subclasses. Therefore there are protected in the C++ jargon. However, we prefer to say that they are private because this is bad style to access directly instance variable from a subclass.

9.6.3 Methods

All the methods are public. A method always precedes a value using the construct. When not specified explicitly using the construct, the return value of the method is the receiver of the

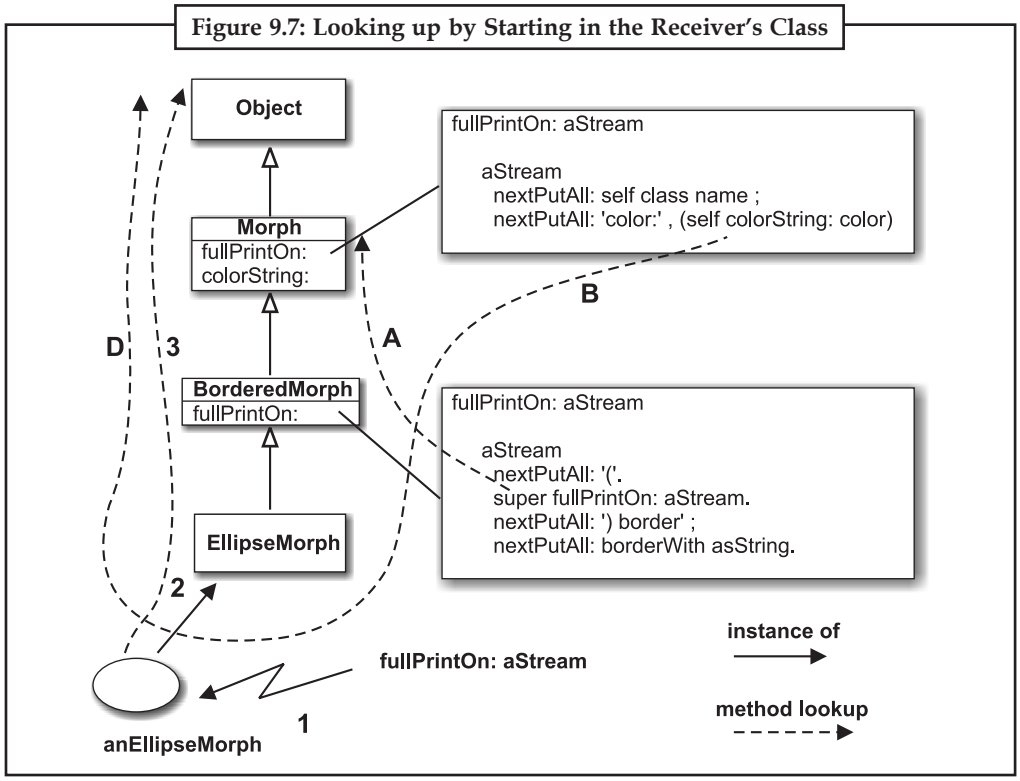
Notes

message i.e., self. Self (equivalent to this in Java) represents the receiver of the message. The lookup of messages sent to self starts in the class of the receiver (as shown in the Figure 9.6 When we define a method in a subclass it can hide method in superclasses. To access such hidden methods of a superclass messages should be sent to super and not self. Super represents also the receiver of the message but the lookup of messages start in the superclass of the class of the method which issued the super invocation. In Figure 9.6, the method *fullPrintOn*: is looked up in the class of the receiver *EllipseMorph* which does not define it, therefore the lookup continues *inBorderedMorph* the superclass of *EllipseMorph*. This class defines the method which then gets executed. The expression super fullPrintOn, aStream is then executed. The lookup then starts in the superclass of *BorderedMorph* – note that the receiver class is not involved to determine where to start the lookup. This method is defined in the superclass so it is executed.

Self is said to be dynamic in the sense that it always represents the receiver of the message. This means that all the messages sent to self are looking up by starting in the receiver's class. For example in Figure 9.7 the message *fullPrintOn*, is sent to anEllipseMorph therefore the lookup of the method *colorString*, invoked in the class Morph in the expression self *colorString*, *aStream* (B) starts in the class of the receiver: EllipseMorph (C).

Note that the model we presented is conceptual in the sense that the virtual machine implementers use all kind of tricks and optimizations to speed up the method lookup. The main point here is to understand what the semantics of self and super.

Figure 9.7: Looking up by Starting in the Receiver's Class



Caution To access hidden methods of a superclass messages should be sent to super and not self.

9.6.4 Abstractness

To finish with the precisions, a class can be abstract. However, there is no dedicated construct for that. A class is considered abstract if one of its method contains the expression self

Notes

subclassResponsibility stating that subclasses have the responsibility to define the method. When such a method is executed an exception is raised.

Method

aMethodDeclaredAbstract

"This is the responsibility of my subclasses to define this method"

self subclassResponsibility

Note that nothing prevents you to create instance of the class having abstract methods. This will work until an abstract method will be invoked.

9.7 Creating the User Interface

Traditional application development involves the propose of data model/ERD representation, develop the database, implement the industry layer, and finally the presentation layer. Some advanced methodologies like RAD, let developers to develop their applications faster by facilitate advanced and special tools. Another standard shift in application development seems happening based on developing smarter components at different layers of software. Consequently, we can expect faster and easier application design/development. This process focuses on the higher level of development or the user interface design. To accomplish the task, we need a set of tools to support this methodology. In other words, smart components would develop the back end of the applications from the GUI, by setting all necessary constraints, validation, business rules and other components. We call the proposed method "User Interface Oriented Application Development, UIOAD". Previous related works based on similar concepts have been brain-stormed and user interface driven system design has been proposed. But in this approach, since the user interface (UI) forms are essential in capturing data requirements, UI prototype can be used by the developer as a source for developing the classes and objects. The existence of a field on a dialog, a web page, or a report means that the data must either be an attribute of an object, the result of some operation on an object or series of objects, or be calculated from some other object(s) attributes. When the same data exists within different objects on the same user interface, it means that those objects are related to each other and results into an association between classes in the class diagram. Initial repetitions for such associations might be detected by the occurrences of the related objects.

Another recent attempt to create a UI centric development is XUL. XUL is designed specifically for building portable user interfaces. The supporting software architecture to create a platform independent information system from the GUI. In addition, a sample commercial application based on this concept will be introduced.

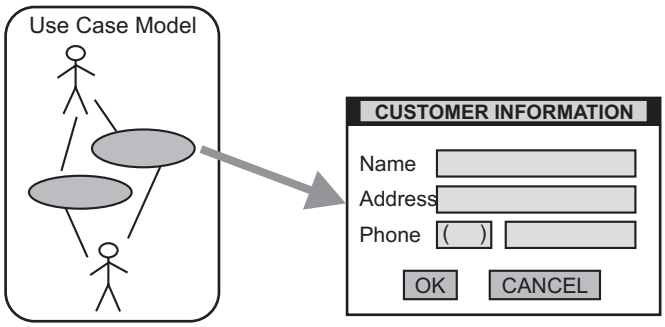
The Database concept has evolved since the 1960s to ease increasing difficulties in designing, building, and maintaining complex information systems (typically with many concurrent end-users, and with a large amount of diverse data).

9.7.1 New System Design Process

In this methodology, we start the design of a new system by first designing the user interfaces or forms of the desired application. To accomplish this, use cases are going to be determined first followed by designing the interfaces based on the design of the user interfaces as shown in Figure 9.8.

Notes

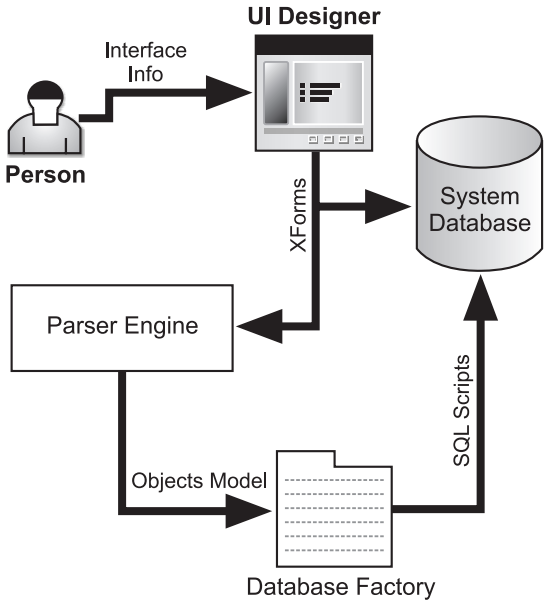
Figure 9.8: The Design Process Starts by Designing the use Cases and the Interfaces Based on the Use Cases



Once the use cases and user interfaces are designed, the development environment will build the supporting structure for the new application including classes, objects, and eventually the database and its corresponding objects, transparently. To accomplish that, this methodology includes three main phases: "User Interface Design", "Parsing Mechanism", and "Database Generation". Therefore, we can assume that a system shown in Figure 9.8 can implement this methodology. In other words, such system requires the following building blocks: "User Interface (UI) Designer", "Parser Engine", and "Database Factory" as shown in Figure 9.9.

Each of the shown phases is explained in more details in the following subsections.

Figure 9.9: Entire Process of Designing a New System



User Interface (UI) Design Phase

This phase includes the following parts:

- *Determination of Form role in System (Systematicviewpoint)*

  In this part, the flow of information from each interface will be determined. This is done by determining each form's role in the system and the type of users or actors that use each interface. Issues such as access control and security of the users are also addressed here. Knowing the role of the form in the system means that we try to find out how this

Notes

piece of the puzzle affects the entire picture. Use cases interoperations and interactions are to be determined and the information flow for each case is specified.

- *Form Detail Determination*

  This stage includes declaring the form details. This high level definition will help us not only to decrease our dependency to different platforms but also to make it possible for high level users to design a form, so that they could describe their forms in an abstract form. There are some well known innovated tools/standards that are suitable for this methodology such as XForms standard. It will bring a distinct part in conjugation with XML standards to design a form. Another example is XUL. Generally our output can be one of the HTML, XForms, XUL, MS Word, or Pdf formats.

- *Interface Functional Definition*

  In this totally conceptual stage, an interface is defined by defining its functionality. This is done by answering a series of questions as shown below:

  - What kind or type of information does the form get?
  - What kind of necessary constraints does a particular field in the form have?
  - How do the form objects relate to the information other forms? (For example contents/ items in the combo boxes, option lists, …)

In this part another important aspect of XForms is discovered. This, in conjugations with other XML standards, provides capabilities to declare the form model absolutely independent of the logic behind the interface.

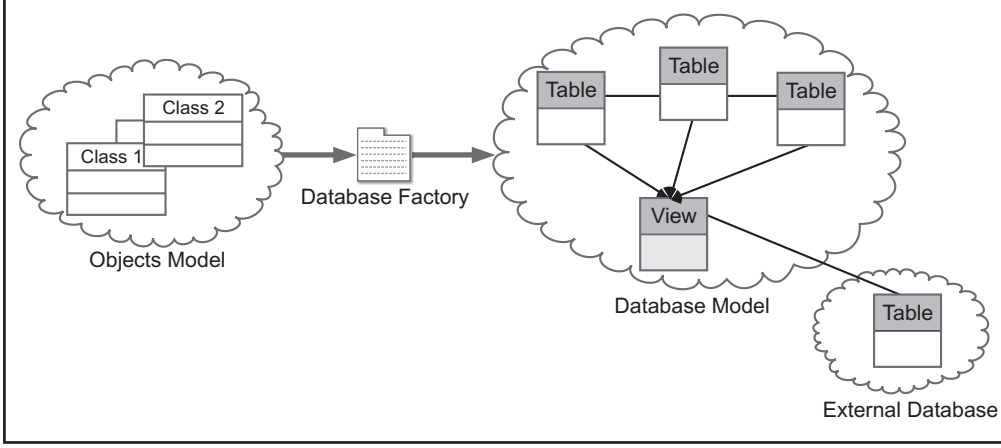
Analysis and Object Production Phase

The output of the previous phase is an XForms document, which includes the design and logic of the form. All the designed items are coming from a standard template. The XForm document maintains the design and the logic of the form as a standard relation between Design and Development phase. This is the main phase in our proposed method. It receives the structured input in the special format of XForm, and transforms it to a well-defined format of the programming languages. In fact this part will parse text into objects as declared. This part will generate the data model as a data structure, considering the input fields of the form. Also, proper communication means to other classes and objects are considered in the class definition. The methods are defined in a very abstract and general form since the classes are made dynamically.

Database Generations

After generating the classes and the relational model, the database should be developed. Database Factory is responsible for transformation of generated structures to the equivalent model in a database as shown in Figure 9.10.

Figure 9.10: Extracting the ER Model From Designed Forms



Notes

Transforming of the OO Model (Which has been done by parser engine) to an ER model is a challenging process and will be done by database factory. In this phase, inputs and outputs are the OO structures and relational structures, respectively.

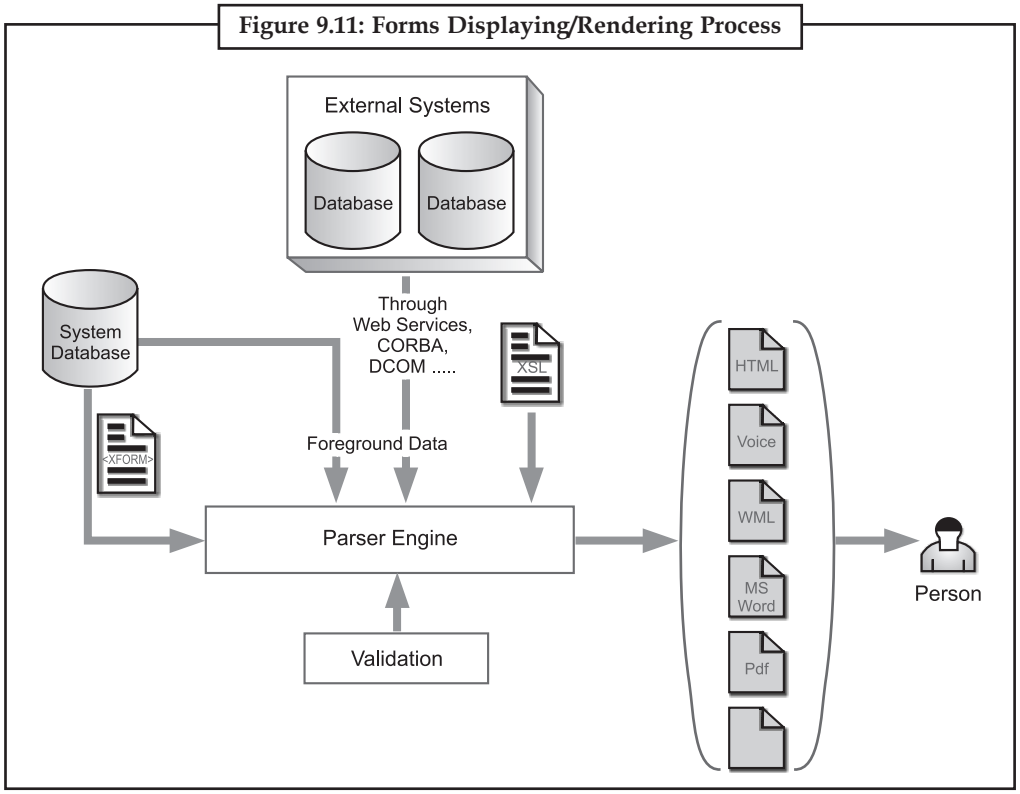
9.7.2 Methodology in Action

Once the system is built, the system information will be stored in the system database. The first step to use the system is to retrieve the form information and rebuild it for the user. The next step is to get the data from the user and store the information in the database.

Form Displaying/Rendering Process

This process includes presenting the form to the user independent from the platform depending on the environment (See Figure 9.11).

Figure 9.11: Forms Displaying/Rendering Process



One of the important outputs of this process is an XForm document that has the exact declaration of the original use cases in the design phase. This declaration is abstract and platform independent. As we saw in design phase, parser engine is responsible for converting declarative model of designed form to OO model. The other responsibility of parser engine is to generate the actual user interface from the stored information and according to end user's platform.

The parser receives form's declarative definition in XForms format and then regenerates the user interface for the target platform from the logic and the user interface design information. For example for regular web browsers it generates HTML codes while for PDAs it may generate WML codes or a windows form for PC desktop applications. To perform this task, parser uses the following parts of the system or information:

XSL files and XSLT files

These files have the main role in generating the interface and the look of the form for different platforms.

Notes

Database

The parser may need to establish a connection to the database to get some information about the sub forms or sibling forms and use them to generate the code.

Validation

In addition to the user interface code, some auxiliary-codes maybe needed to control/validate the input data like by injecting javascript control codes into the HTML forms.

Web Services

If a form requires to be connected to another distributed system, then web services is the access method to such system.

In this way, the form will be displayed and is ready for the next stage.

Processing and Storing Data

The user fills the displayed form and returns it to the system. Figure 9.12 shows the steps clearly (it assumed the form is an HTML one).

After filling the form by the user, the data will be delivered to parser engine box. One of the other responsibilities of this box is converting user's data to the proper format that would be tangible by the system (it has different responsibilities in "designing form" and "displaying form" phases). It should recognize the relations between sent data and form's fields by referring to the form's definition document (in XForms format). This is the first step in importing user's data to system as a structured one. The next step, which is so important, is data validation. In "displaying form" process, the validation was done in partial mode but the main activity would be done after receiving user's data. In this step, user's data should be validated against the form's logic. In other words the declarative statements that would be supplied in form's logic definition would be checked against the user's data to verify if they are consistent with them or not. But as we mentioned earlier, it may be impossible to validate the user's data at the same place where we display the form to the user. Then it is required to connect to the other part of same system (or even other systems) to verify correctness of user's data. So as you see in the Figure 9.12, the validation box may connect to database or use some web services (to connect to other systems in distributed solutions) to see if the user's data are consistent with other data or not. Note that we can connect to other systems with different solutions like using RPC (Remote Procedure Call), CORBA (Common Object Request Broker Architecture). DCOM (Distributed COM) and so on that are not mentioned in this diagram.

Here we have a consistency problem between distributed databases that should be resolved by the DBAs (Database Administrators). After user's data is verified and relations between data and form's components are found, the related objects would be created from the classifiers that were made in "designing form" phase. In other words, the user's data would be stored in some objects (in OO concepts). Therefore the user's data are capable to circulate through the system easily and being processed wherever is needed because its format is recognizable for the program generator system. Briefly, the objects are the real instances of form's data in the format of the classes that present the form's schema.

Finally these generated objects should be stored in database. This process would be done by "Data Access Methods" part. This part receives the user's data in object format and then stores them in database as output result. This part is responsible for mapping these received objects to such database objects that were created in "design form" phase and store received objects data into these database objects. Note that the reason for this mapping in current phase is because of input objects of this box are OO objects whereas the equivalent objects in database are Database

Notes

Objects (like tables, views, stored procedures and so on).The received objects would be mapped to related ones in database system as a result. Moreover, "Data Access Methods" routine should store related user's data information to make it available for future access and also for specifying the state of user's data in system. This information usually consists of one unique identification number, date of filling form, related actors and so on.

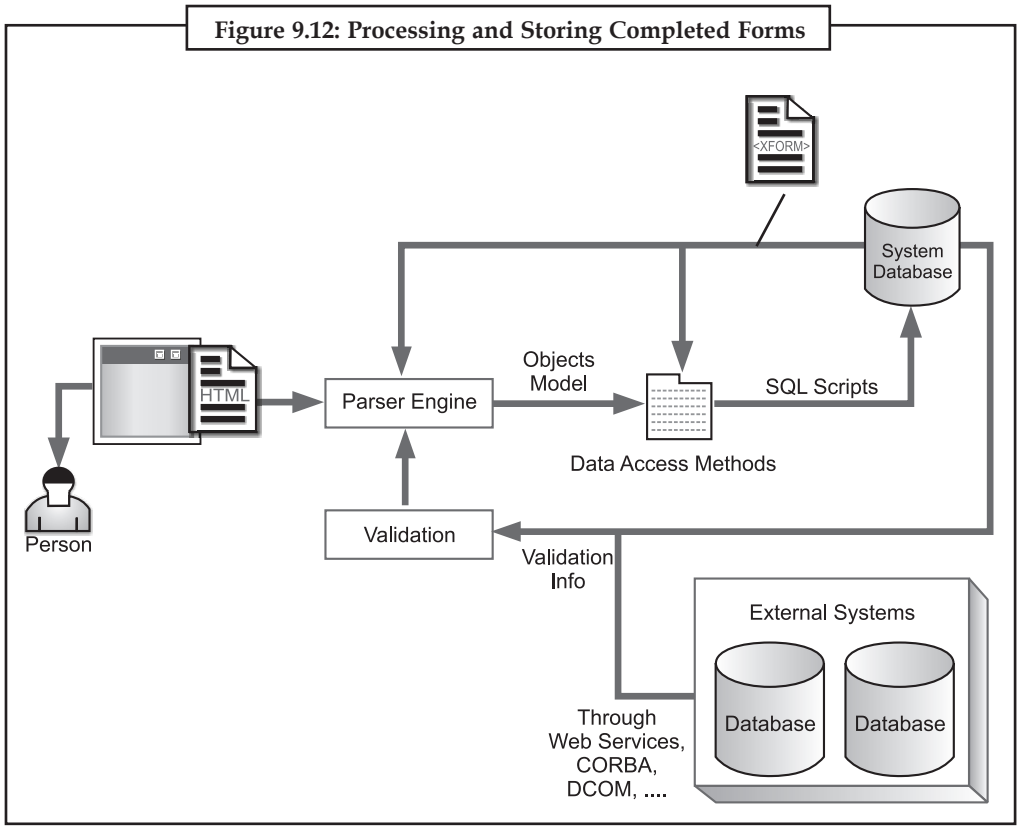
Representing the Filled Forms

This stage consists of representation of a designed form with data samples. In other words, it is the final stage of a designed form including data. As mentioned before we can divide this stage into two major steps:

- Form Representation
- Injecting form's data to the rendered form

At the first step, design part of the form in XForms document (declarative, platform dependent) is converted into as implemented model in target platform. Obviously in this conversion would be done by XSLT. Also for preparing required form data which are dependent on other parts of the system such as other forms or even other software systems, we use web services. The second part includes the following steps: based on XForms, related entities in database are recognized. Data Access Component does this task, which has been used in data store step, too. This component gets the XForms and complementary information and retrieves the filled forms data from the database.

Figure 9.12: Processing and Storing Completed Forms



Major parts of the retrieved data in previous step are unprocessed. So it is necessary to retrieve more data from database or web services to prepare the rendered form's data.

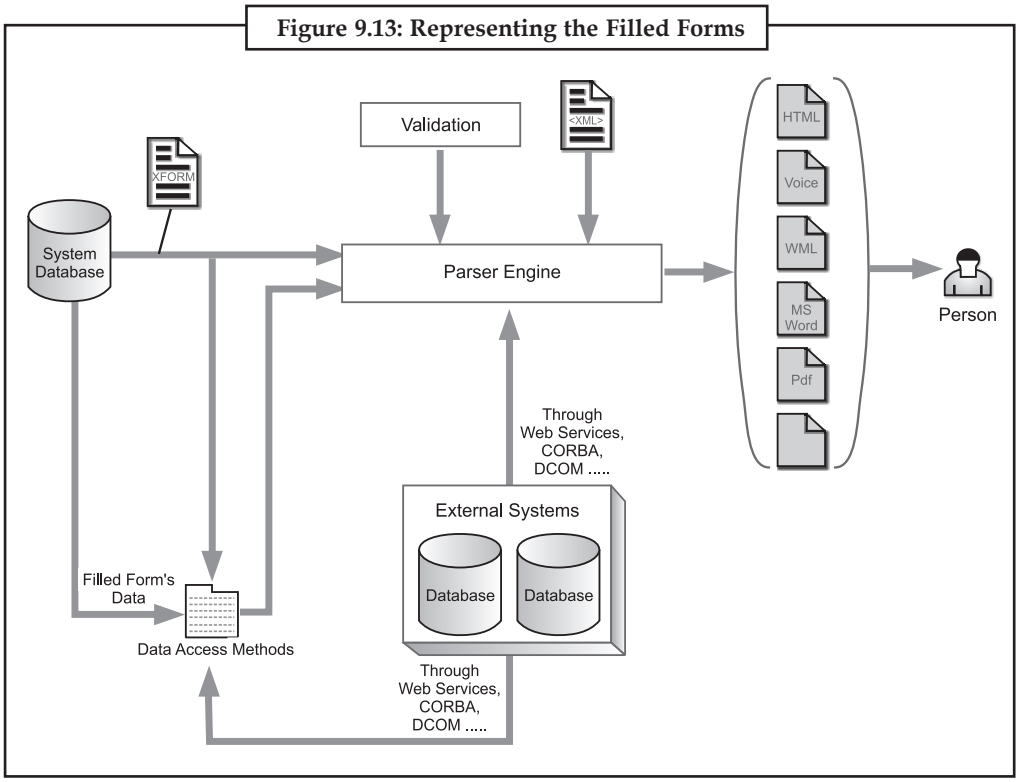
The DAC, creates the OO structure for rendered data by using XForms structure. After this stage the form's structure would be available in system as a set of controls/components dependent on platform.

Notes

The rendered controls would be transfer to the parser engine. This engine:

- Renders the form for platform.
- Maps the prepared data to controls.

Figure 9.13: Representing the Filled Forms



Case Study

RDF Representation and Access to Master Data

Master data, as the core business entities a company uses, refers to lists or hierarchies of customers, suppliers, accounts, products, or organizational units. In this scope, Product and Customer Information play a very important role since their accurate management is becoming critical for modern enterprises. They enable companies to centralize, manage and synchronize all product and customer information with heterogeneous systems and trading partners. The most critical challenge is the need to build a common master model flexible enough to deal with business changes, and expressive enough to represent the semantics of master data.

To enhance IBM master data management (MDM) solutions (Websphere Product Center and Websphere Customer Center), we developed semantic technologies for Product Information Management (PIM) and Customer Data Integration (CDI), respectively. Here, we would like to highlight the value of semantic web technologies for MDM and brief completed and ongoing work. As introduced in our previous work, the advantages of OWL ontologies for product information include followings:

On RDF, OWL uses the concept of Universal Resources Identifiers (URIs) as Web-based identification scheme. It firstly allows one to refer to industry specific or external ontologies;

Contd...

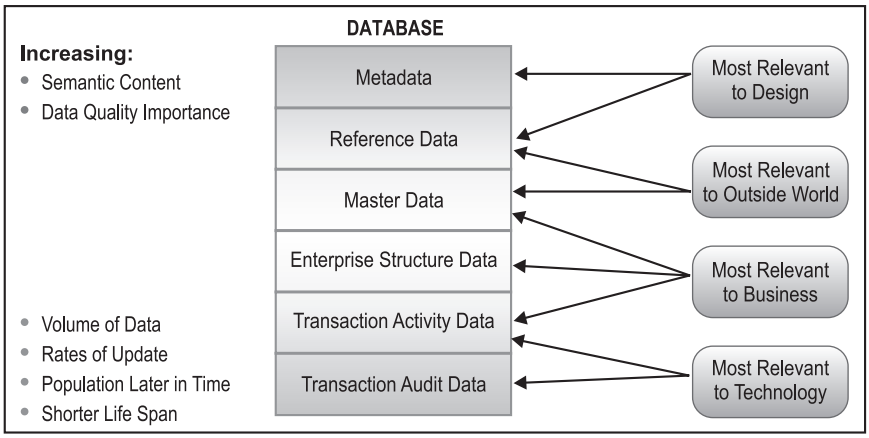
Notes

and on the other hand it allows synchronization of product information management utilities to other core business entities, such as those in customer data integration (CDI).

OWL allows the definition of richer properties and relationships. Object Properties can be defined as symmetric, functional, inverse functional, or transitive. Object Properties are then suitable to describe complex relationships among products and between products and other entities in product information. The expressivity of OWL allows the definition of logical classes (intersection, union and complement operators), which enables automatic classification for product items. For instance, new product categories can be defined as the intersection of two others: smart phones products, which gather characteristics of both PDA and phones, are a good example. Any product which is simultaneously a PDA and a phone is then a smart phone. OWL restrictions can define dynamic categories which do not exist in the pre-designed category hierarchy and are specified by users at query time. It can represent complex and potentially evolving categories. For example, using Minimum cardinality restriction, it is possible to define an "outdated products" category which gathers all products replaced by at least one other product. Items of dynamic categories can be retrieved using OWL ontology reasoning. Details of RDF representation of PIM can be found in. Since IBM PIM system currently uses technologies similar to the triple store for storage (item-property-value), we support SPARQL queries over PIM storage easily by reusing

SPARQL2SQL query translation technologies developed in. The query rewriting method translates a SPARQL query into a single SQL statement, utilizing well-developed SQL engines in a most effective manner.

Figure: Conceptual Architecture for SPARQL Queries over the CDI System



The advantages of OWL ontologies for customer information are similar to those for product information. Representing and discovering various relationships among customers has a very high value for the CDI, which is enabled by ontology and rule reasoning. Different from the PIM system, IBM CDI system makes use of object-oriented database schema for storage. Each entity of the CDI model owns a separate table to store corresponding instances. So, we need a mapping to link the CDI data with the OWL ontology generated and enriched from the CDI logical model. We proposed the following conceptual architecture to develop a POC system for RDF Access to the IBM CDI system. Note that the ontology view in the bottom right of Figure is in fact a virtual representation of the CDI data in SOR's schema form, over which IBM SHER engine for ontological reasoning can work directly. In the future, we will support SHER engine over a D2RQ like mapping, and thus replace the ontology

Contd...

Notes

views component. Besides the mapping between relational DB and ontology, it is critical to construct an appropriate RDF representation (ontology) for relational data.

The challenge in using domain knowledge in ontologies effectively is in crafting "integrating" ontologies that tie the domain knowledge in ontologies with other ontologies that may be used to model the relational data in the database. For example, in figure a crucial piece in answering the query is the ontology in top right, which ties the data in the relational DB with the domain ontologies describing regions and businesses. We would need to come up with guidelines and best practices for developing these ontologies IBM's Semantic Web Tools and Systems Here, we briefly introduce some IBM's ontology tools and systems related to RDF Access to relational data. IODT is a toolkit for ontology-driven development, including EMF Ontology Definition Metamodel (EODM) and an OWL Ontology Repository (named SOR). EODM is derived from the OMG's Ontology Definition Metamodel (ODM) and implemented in Eclipse Modelling Framework (EMF). It is the run-time library that allows the application to put in and put out RDFS/OWL ontology in RDF/XML format; manipulate an ontology using Java objects; call an inference engine and access inference results; and transform between ontology and other models. SOR is an OWL ontology storage and query system on the relational DBMS. It supports Description Logic Program (DLP), a subset of OWL DL, and SPARQL query language. SHER reasoned uses a novel method that allows for efficient querying of SHIN ontologies with large ABoxes stored in databases. Currently, this method focuses on instance retrieval that queries all individuals of a given class in the ABox. It is well known that all queries over DL ontologies can be reduced to consistency check, which is usually checked by a tableau algorithm. SHER groups individuals which are instances of the same class into a single individual to generate a summary ABox of a small size. Then, consistency check can be done on the dramatically simplified summary ABox, instead of the original ABox. It is reported in that SHER can process ABox queries with up to 7.4 million assertions efficiently, whereas the state of art reasoners could not scale to this size. As described in the example shown in figure to enable semantic queries over existing data sources, we need to store and leverage ontologies representing domain knowledge. SOR could be used to manage such ontologies. Similarly, in the CDI case, we need an ontology repository to cache and materialize some inference results for performance improvement. In general, an RDF store, such as SOR, could be used to store domain knowledge or part of reasoning results for RDF access to relational databases. Obviously, SHER engine could be used for scalable ontological reasoning for SPARQL queries over relational databases. The system described in takes an ETL (Extract-Transform-Load) approach, where the relational data in the database is extracted, transformed into RDF triples based on a set of domain ontologies and mapping rules, and loaded into SOR. This system also provides mechanisms to handle updates to the relational database as well as to the ontologies.

Question

1. Explain Universal Resources Identifiers (URIs).

2. What is Customer Data Integration (CDI)?

Self Assessment Questions

6. All authorized users in an organization can share the ……. stored in its database.

   (*a*) information (*b*) method

   (*c*) processor (*d*) None of these

7. The …… of data elements in a database are there for a specific purpose.

   (*a*) single (*b*) only related

   (*c*) collection (*d*) None of these

Notes

8. All data in the database reside in a …….
   (*a*) abstract data (*b*) data repository
   (*c*) dictionary (*d*) relational data
9. The relationship between each pair of data structures at levels next to each other is a …….. relationship.
   (*a*) customer (*b*) client server
   (*c*) parent-child (*d*) None of these
10. A relation is a …….. table of data observing relational rules.
   (*a*) two-dimensional (*b*) single-dimensional
   (*c*) multi-dimensional (*d*) None of these
11. The relational model establishes relationships between data structures by means of …….. and not by physical pointers.
   (*a*) primary keys (*b*) foreign keys
   (*c*) derived key (*d*) secondary key
12. Each fragment of data at every location may be managed with the ……..type of database management system.
   (*a*) same (*b*) different
   (*c*) all (*d*) related
13. To finish with the precisions, a class can be …….
   (*a*) subclassResponsibility (*b*) *aMethodDeclaredAbstract*
   (*c*) method (*d*) abstract
14. The XForm document maintains the design and the logic of the form as a standard relatin between …….. and Development phase.
   (*a*) Design (*b*) Documentation
   (*c*) Testing (*d*) Maintenance
15. The parser receives form's declarative definition in ……. format and then regenerates the user interface.
   (*a*) XForms (*b*) Web Services
   (*c*) XSL files and XSLT files (*d*) Database

9.8 Summary

- A data model represents the data requirements of an organization. You can diagrammatically show a data model with symbols and figures.
- All data in the database reside in a data repository.
- The data repository contains the actual data.
- The database approach provides for data abstraction.
- Hierarchy model should show that an assembly contains subassemblies, a subassembly contains parts, and a part contains subparts. Immediately you can observe that this

Notes

data model must be hierarchical in nature, diagramming the assembly at the top with subassembly, part, and subpart at successive lower levels.

- Hierarchical data model represents well any business data that inherently contains levels one below the other.
- Hierarchical or the network data models, the relational model establishes relationships between data structures by means of foreign keys and not by physical pointers.
- User interface (UI) forms are essential in capturing data requirements; UI prototype can be used by the developer as a source for developing the classes and objects.
- Once the system is built, the system information will be stored in the system database.
- Database Factory is responsible for transformation of generated structures to the equivalent model in a database
- The XForm document maintains the design and the logic of the form as a standard relation between Design and Development phase

9.9 Keywords

Database: It is an ordered collection of related data elements intended to meet the information needs of an organization and designed to be shared by multiple users.

Hierarchical Data Model: Represents well any business data that inherently contains levels one below the other.

Network Data Model: It is more representative of real-world information requirements than the hierarchical model.

Parser: May need to establish a connection to the database to get some information about the sub forms or sibling forms and use them to generate the code.

Physical Pointers Link: It occurrences of an owner record type with the corresponding occurrences of the member record type.

Related Data Elements: The data elements in a database are not disjointed structures without any relationships among them.

Relation: It is a two-dimensional table of data observing relational rules.

User Interface (UI): Forms are essential in capturing data requirements, UI prototype can be used by the developer as a source for developing the classes and objects.

User Interface Oriented Application Development, UIOAD: Previous related works based on similar concepts have been brain-stormed and user interface driven system design has been proposed.

Validation: In addition to the user interface code, some auxiliary-codes maybe needed to control/ validate the input data like by injecting javascript control codes into the HTML forms.

XForms: It format and then regenerates the user interface for the target platform from the logic and the user interface design information.

Lab Exercise

1. Construct Lookup of messages sent to self starts in the class of the receiver.
2. Create a basic object model which follows all the rules.

Notes

9.10 Review Questions

1. What is your understanding of the term *database*? Describe it in two or three sentences.
2. List three areas of technology growth that had direct positive effects on data systems. Briefly explain the positive impact.
3. Name and briefly describe the major driving forces for database systems.
4. List and explain any three advantages with centralized database.
5. List and describe any three benefits of database systems in comparison with file-oriented systems.
6. What is a data dictionary? Describe its purpose.
7. Briefly describe the features of a network data model. How are relationships represented in a network data model?
8. Is the relational data model better than the earlier data models? Give your reasons.
9. Write down the differences between the hierarchical database model and the relational database model.
10. How create a basic object model and also describe the rules for creating the user interface?

Answer for the Self Assessment Questions

1. (*c*)	2. (*b*)	3. (*a*)	4. (*d*)	5. (*c*)
6. (*a*)	7. (*c*)	8. (*b*)	9. (*c*)	10. (*a*)
11. (*b*)	12. (*a*)	13. (*d*)	14. (*a*)	15. (*a*)

9.11 Further Readings

Books Creating user interfaces by *demonstration,* by Brad A. Myers.

Online link *http://www.iam.unibe.ch/ _ducasse/FreeBooks.html.*

Unit 10: Web Services

Notes

CONTENTS

Objectives

After studying this unit, you will be able to:

- Define the XML Web Services
- Explain the Uses for Web Services

Introduction

We can now use ASP.NET to create Web Services based on industrial standards including XML, SOAP and WSDL. A Web Service is a software program that uses XML to exchange information with other software via common internet protocols. In a simple sense, Web Services are a way for interacting with this chapter provides an overview of web service concepts and technologies supported by Net Beans IDE.

Web services are distributed application components that are externally available. You can use them to integrate computer applications that are written in different languages and run on different platforms. Web services are language and platform independent because vendors have agreed on common web service standards. Oracle is developing a java.net project called Metro. Metro is a complete web services stack, covering all of a developer's needs from simple "Hello, World!" demonstrations to reliable, secured, and transacted web services.

Metro includes Web Services Interoperability Technologies (WSIT). WSIT supports enterprise features such as security, reliability, and message optimization. WSIT ensures that Metro services with these features are interoperable with Microsoft .NET services. Within Metro, Project Tango develops and evolves the code base for WSIT. Several programming models are available to web service developers. These models fall into two categories, both supported by the IDE:

- ***REST-base***: Representational State Transfer is a new way to create and communicate with web services. In REST, resources have URIs and are manipulated through HTTP header operations.

Notes

- ***SOAP/WSDL-based***: In traditional web service models, web service interfaces are exposed through WSDL documents (a type of XML), which have URLs. Subsequent message exchange is in SOAP, another type of XML document.
- ***RESTful Web Services***:
  1. REST-based ("RESTful") web services are collections of web resources identified by URIs. Every document and every process is modeled as a web resource with a unique URI. These web resources are manipulated by the actions that can be specified in an HTTP header. Neither SOAP, nor WSDL, nor WS-standards are used. Instead, message exchange can be conducted in any format—XML, JSON, HTML, etc. In many cases a web browser can serve as the client.
  2. REST is a suitable technology for applications that do not require security beyond what is available in the HTTP infrastructure and where HTTP is the appropriate protocol. REST services can still deliver sophisticated functionality. Flickr, Google Maps and Amazon all provide RESTful web services. NetBeans IDE Software as a Service (SaaS) functionality lets you use Facebook, Zillow, and other third-party-provided services in your own applications.
- ***SOAP-based Web Services***: In SOAP-based web services, Java utilities create a WSDL file based on the Java code in the web service. The WSDL is exposed on the net. Parties interested in using the web service create a Java client based on the WSDL. Messages are exchanged in SOAP format. The range of operations that can be passed in SOAP is much broader than what is available in REST, especially in security.

10.1 Web Services

A Web Service is a component written in any language, deployed on any platform, which has a standard wrapping layer based on XML. Other services have to be able to discover the Web Service and invoke it dynamically.

Web Services are applications with two characteristics.

First, a web service publishes, and is defined by, an application programming interface (API) for the functionality it makes available to external callers.

Second, a web service is accessed over a network by using the hypertext transfer protocol (HTTP).Web services enable interoperability between software systems, and are the foundation for modern service oriented architecture (SOA).

XML Web Services using the .NET platform and ASP.NET. It begins with an overview of the .NET framework and an explanation of the process behind XML Web Services, and then quickly delves into building and consuming Web Services. Included will be discussions of proxies, classes, SOAP, Global XML Web Services Architecture, WSDL, UDDI, and Disco.

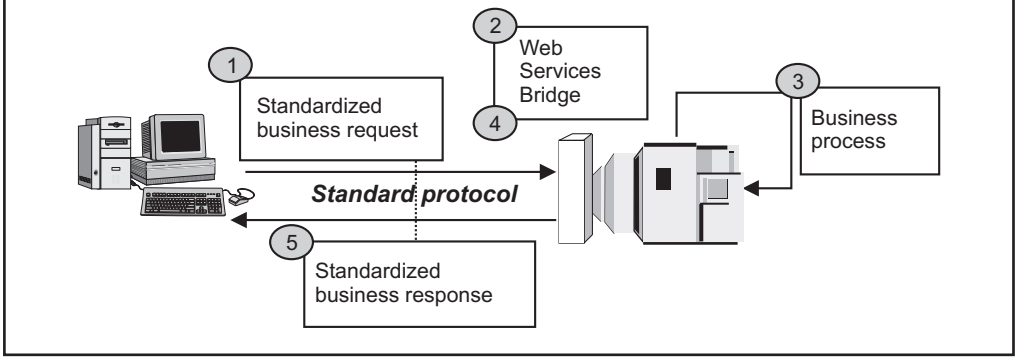
The general workflow is shown below.

This new technology, initiated by IBM and Microsoft, then partially standardized under the aegis of the W3C, is unanimously accepted throughout the computing industry. It is this; above all, that makes Web Services a revolutionary technology.

The purely technical aspects are not fundamentally innovative. What makes the architecture of Web Services so attractive, in much the same way as XML, is its simplicity, readability and that it is built on existing standards.

Notes

Figure10.1: Web Services



The concepts used in Web Services hinge on the three following acronyms:

- SOAP (Simple Object Access Protocol) is a protocol based on the language, XML. It allows exchanges between applications, whatever their platform. A SOAP service call is a stream of ASCII text framed by XML tags and transported in the HTTP protocol.
- Web Services (Description Language) describes the Web Services in XML format, specifying the methods that can be called, their signatures, and their access points (URL, port, etc.). To some extent, it is the equivalent of CORBA's IDL language for distributed programming.
- UDDI (Universal Discovery Description and Integration) standardizes the use of distributed directories for Web Services, allowing both publishing and discovery. UDDI behaves like a Web Service itself, its methods being called using the SOAP protocol.

A significant advantage of Web Services, relative to other distributed architectures, is the support of firewalls. By using the HTTP protocol on port 80, which is generally not secured, the Web Services can get into the enterprise without being blocked by firewalls.

However, this creates other safety concerns because the default use of these characteristics is too permissive. It is necessary to take this into account at the protocol levels.

A Web Service Example

In the following example we will use ASP.NET to create a simple Web Service that converts the temperature from Fahrenheit to Celsius, and vice versa:

```
<%@ WebService Language= "VBScript" Class= "TempConvert" %>
Imports System
Imports System.Web.Services
Public Class TempConvert :Inherits WebService
<WebMethod()> Public Function FahrenheitToCelsius
(ByVal Fahrenheit As String) As String
 dim fahr
fahr=trim(replace(Fahrenheit, "", "", "."))
if fahr= "" or IsNumeric(fahr)=false then return "Error"
return ((((fahr) - 32) / 9) * 5)
End function
<WebMethod()> Public Function CelsiusToFahrenheit
(ByVal Celsius As String) As String
dim cel
```

Notes

```
cel=trim(replace(Celsius, ",", "."))
 if cel= "" or IsNumeric(cel)=false then return "Error "
return ((((cel) * 9) / 5) + 32)
End function
End class
```

This document is saved as an .asmx file. This is the ASP.NET file extension for XML Web Services.

The term web service can mean a number of different things:

1. In the broadest sense a web service is any service available via the web.
2. More commonly a web service is any service which is based on web technologies, which is intended for use by computer programs rather than people.
3. In some cases web service is specifically used to refer to services which use specific web services technologies (e.g. SOAP or REST).

Throughout this guide the term Web Service will be used to refer to services intended for use by programs.

While many bioinformatics resources are available on the web, they are usually only available via web interfaces which are targeted directly at users. This limits their utility in applications which require systematic access to a resource, where the local resources are not available to install and maintain the data and software required. Or in cases where a data resource or analytical tool needs to be integrated into a web portal or workbench application. In these cases Web Services offer a method for accessing the resource remotely. This has the advantage of delegating the maintenance costs for the resource to the service provider, rather than having to absorb these costs locally, and significantly reduces the development and deployment costs for applications which need to consume data or results from the resource.

Client/Server Model

Web Services use the traditional client server model: a, possibly remote, server provides resources which are requested and consumed by a client whom the user interacts with. A simple example of this is browsing the web where the web browser is a client which displays pages and allows the user to intact with the pages, and the web server(s) provide the data for the pages (HTML, images, etc.) when it is requested by the client.

Synchronous and Asynchronous Access

While many resources will return a result almost immediately and thus are suitable for synchronous requests, where the client makes the request and waits for the server to send the response containing the result, some types of analysis take longer. In these cases a synchronous request will have issues with timeouts and may make the client unresponsive. To address these issues services which provide access to such resources provide a mechanism for making asynchronous requests. This usually takes the form of a collection of methods which allow the client to:

1. Submit a job and get a job identifier.
2. Get the status of a job. Typically return a status code indicating if the job is pending, running, finished or gave an error.
3. Get the results of a finished job.

Notes

The nomenclature for these methods and the job status codes depends on the service and additional methods may also be provided giving more control over the life cycle of the job.

Dispatcher

The Dispatcher based services at EMBL-EBI (e.g. WSFasta, WSInterProScan, WSNCBIBlast and WSWUBlast) provide four methods for interacting with a job:

1. A "run " method to submit the job, this is named after the service (e.g. RunFasta for WSFasta)
2. Get the status of the job (check Status). Returns a status code indicating the status of the job, possible values are:
   - **RUNNING**: The job is currently being processed.
   - **PENDING**: The job is in a queue waiting processing.
   - **NOT_FOUND**: The job cannot be found, commonly seen when the job is no longer available.
   - **ERROR**: The job failed.
   - **DONE**: Job has finished, and the results can then be retrieved.
3. Discover the available result types for a finished job (get Results).
4. Get the result of the specified type for the job (poll).

Soap Lab

Soap lab services provide many methods for interacting with a job, to run a job asynchronously there are three relevant methods:

1. To submit the job (create AndRun for generic service or run for typed service)
2. Get the status of the job (get Status). Returns a status code indicating the status of the job, possible values are:
   - **CREATED**: Job has been created but not yet executed.
   - **RUNNING**: Job is currently being processed.
   - **COMPLETED**: Job has finished, its results can be retrieved.
   - **TERMINATED_BY_ERROR**: Job was terminated due to an error.
   - **TERMINATED_BY_REQUEST**: Job was terminated either by user request or by the Soap lab job manager.
3. Get the all the results of job (get Results)

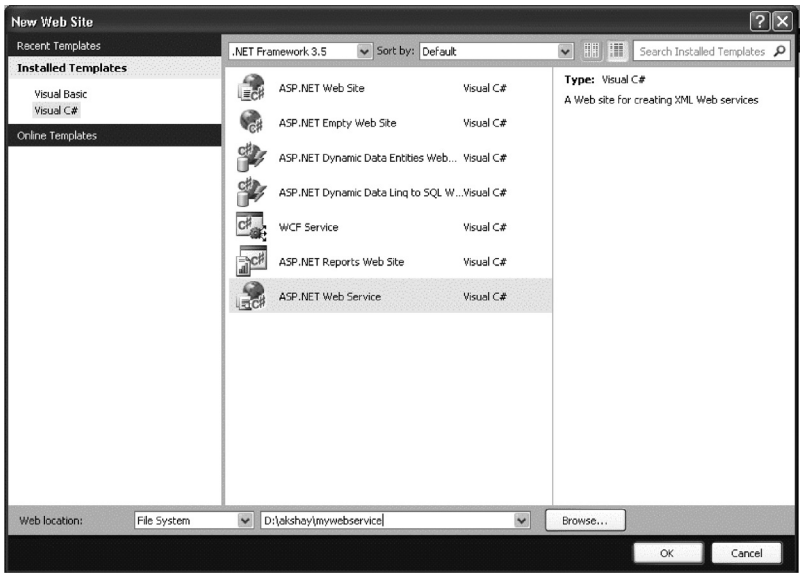
Example of Creating Web Service in .NET

Here are samples codes which use to create and consume ASP.NET Web Service:

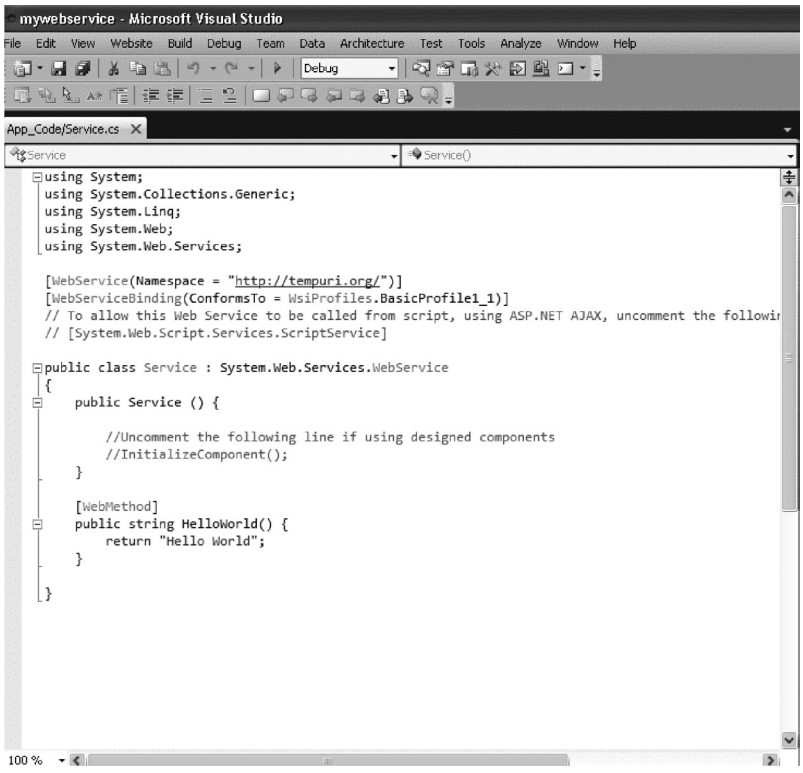
Step 1: Create the ASP.NET Web Service Source File

Open Visual Studio 2010 and create a new web site.->Select .Net Framework 3.5. ->Select ASP. NET Web Service page -> Then, you have to give the name of your service. In this example it is given name "mywebservice ". Then Click the ok Button. A screen-shot of this activity is given below.

Notes



Step 2: Click on the "ok " button; you will see the following window.



Here (in the above figure), you will note that there is predefined method "HelloWorld " which returns the string "Hello World ". You can use your own method and can perform various operations. ***Service.cs***

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;
```

Notes

```
[WebService(Namespace= "http://tempuri.org/ ")]
[WebServiceBinding(Conformist= WsiProfiles.BasicProfile1_1)]
// To allow this Web Service to be called from script, using
ASP.NET AJAX, uncomment the followingline.
//[System.Web.Script.Services.ScriptService]
public class Service :System.Web.Services.WebService
{
[WebMethod]
 public int Multiplication(int a,int b)
{
return (a*b);
 }
}
```

Before Debugging the above Web Service see some important term:

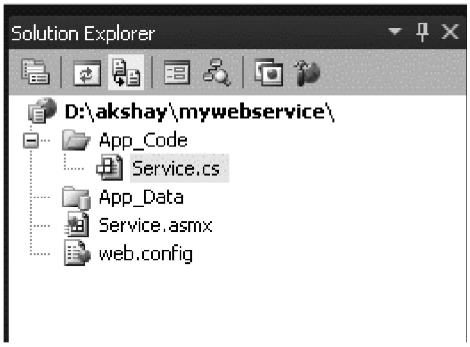
Using System.Web.Services

This directive allows you to refer to objects in the System.Web. Services namespace without having to fully qualify the request. This statement is optional, but if it is not included, every reference to an object in this namespace must be fully qualified. An example is the next line, which is our class declaration. With the using statement, it looks as follows in C#:

The [Web Method] attribute

The Service class exposes a single method, the public method Multiplication, which takes two integer arguments and returns the multiplication of two number as integer. To expose a method as a part of a web service, you must decorate it with the Web Method attribute, which tells the compiler to treat it as such. Any method marked with the Web Method attribute must be defined as public. Class methods exposed as web services follow the same object-oriented rules as any other class, and therefore methods marked private, protected, or internal are not accessible and will return an error if you attempt to expose them using the Web Method attribute.

In the Solution Explorer you will see



Service.asmx- which contains the following code:

<%@ WebService Language="C# " CodeBehind="~/App_Code/Service.
cs " Class= "Service" %>

The page directive WebService is required and the class is the name of the .NET Class to expose the WebService, each method exposes as WebService Class Method need to have a declarative attribute statement.

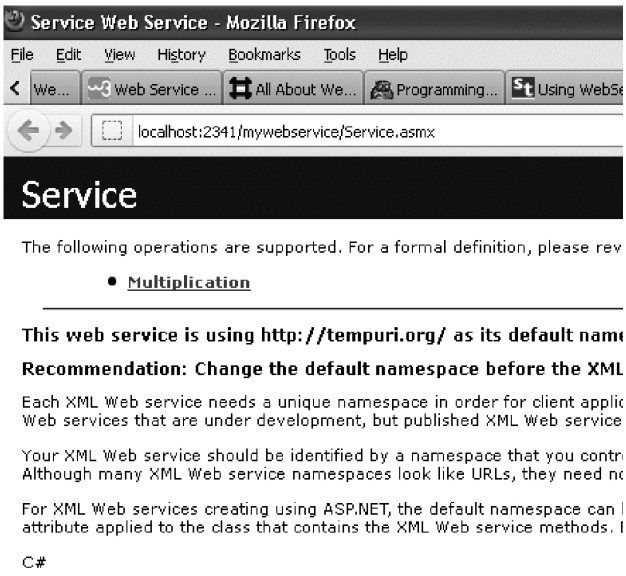
Notes

The WebService directive is similar to the Page directive that begins most .aspx pages. For the Multiplication web service to work, you must assign values to two WebService directive attributes: Language and Class.

The required Language attribute lets .NET know which programming language the class has been written in. As you might guess, the acceptable values for the language attribute are currently C#, VB, and JS for JScript.NET.

The Class attribute, also required, tells ASP.NET the name of the class to expose as a web service. Because a web service application can comprise multiple classes, some of which may not be web services, you must tell .NET which class to expose, a step analogous to declaring a Main() method to indicate the entry point of a .NET console application or component. Note that even if your web service contains only one class, setting this attribute is required. Now back to our web service.

Step 3: Build Web Service and Run the Web Service for testing by pressing F5 function key.

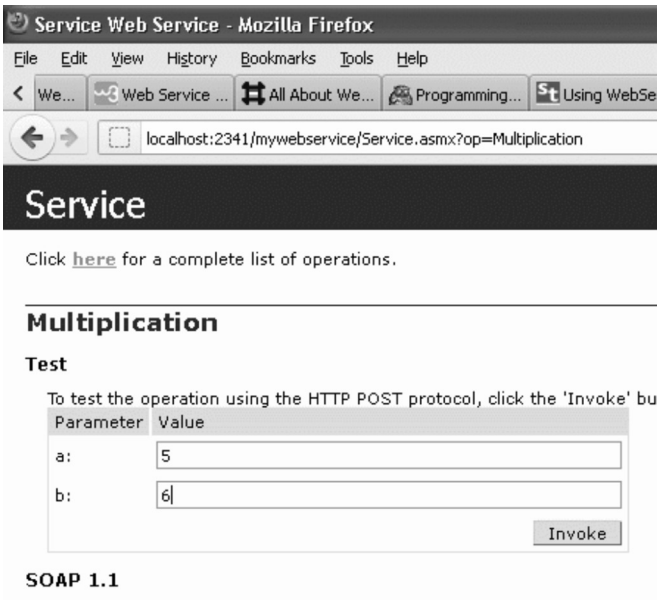


Copy the URL of this web service for further use

Click on the Multiplications button to test the web service.

Enter the value of a and b

Notes



By pressing the "Invoke" button a XML file is generated.

Now our web service is ready to use; we just need to create a new web site to consume the web service.

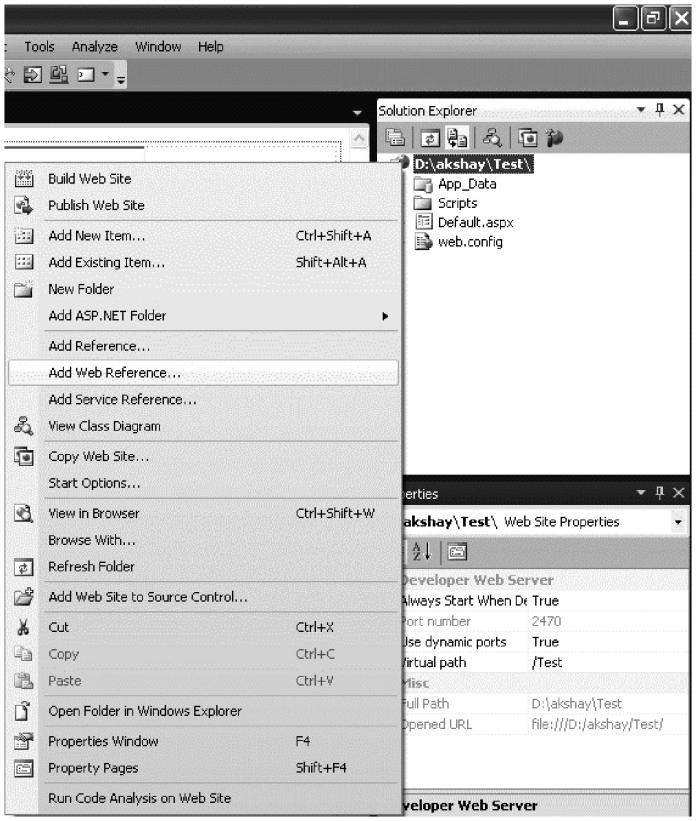
Notes

Example of Testing Web Service

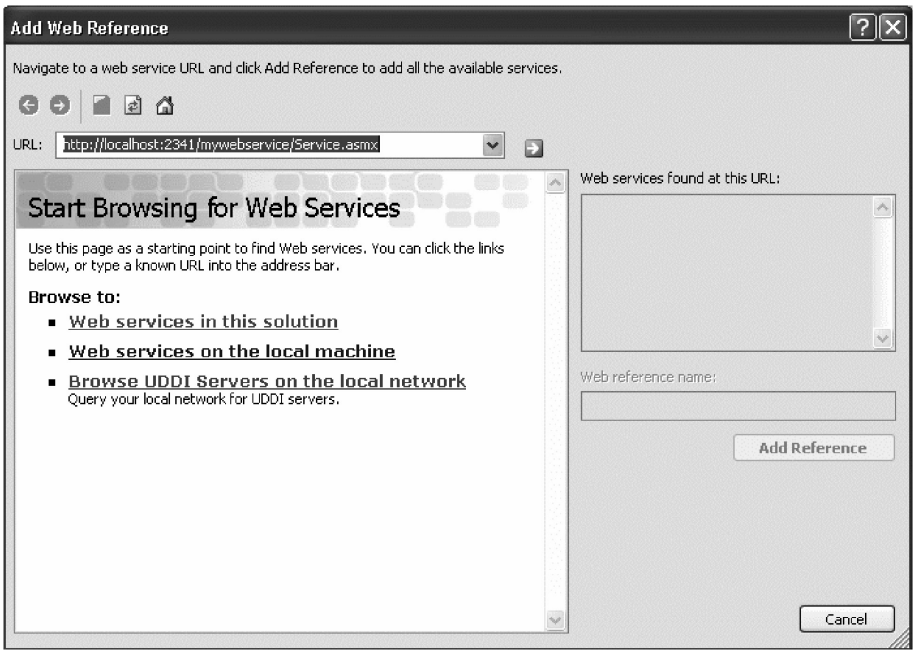
Step 5: Create a Test Web Site by File > New > Web Site > ASP.NET Web Site

Name the web site, for example here the name "Test" is choosen.

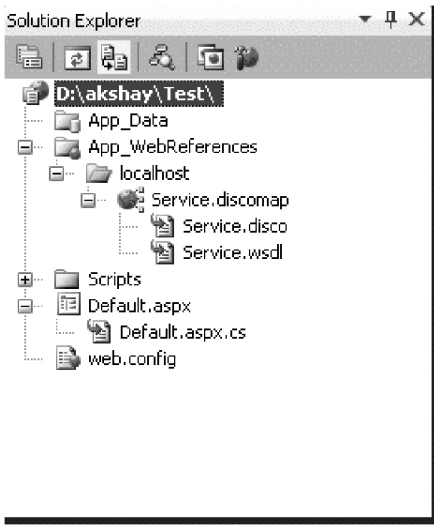
Step 6: Right-click Solution Explorer and choose "Add Web Reference".



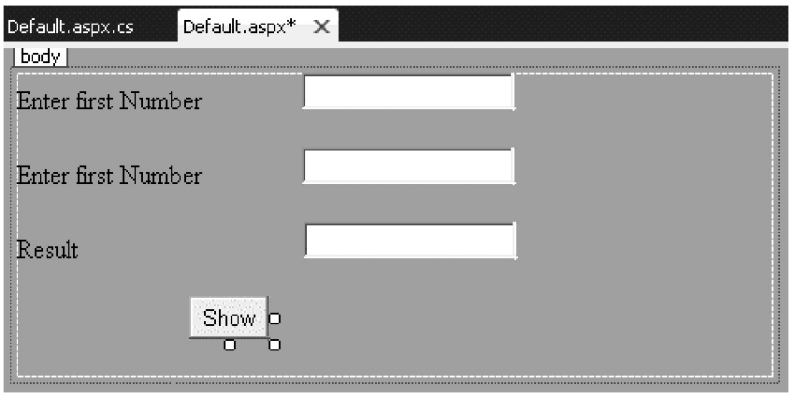
Step 7: Paste the URL of the web service and click on 'Go' button and then 'Add reference'. **Notes**



Step 8: Now your web service is ready to use in the Solution Explorer you will see:



Step 9: Go to the design of the Default.aspx page; drag and drop three Textboxes and one button.

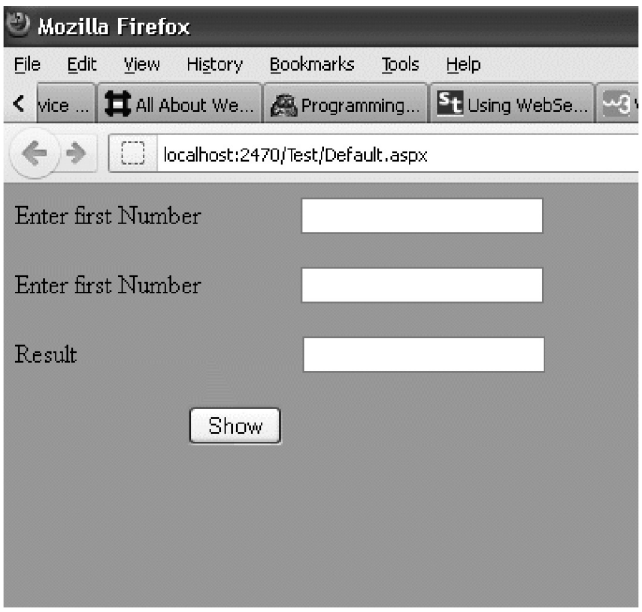


Notes

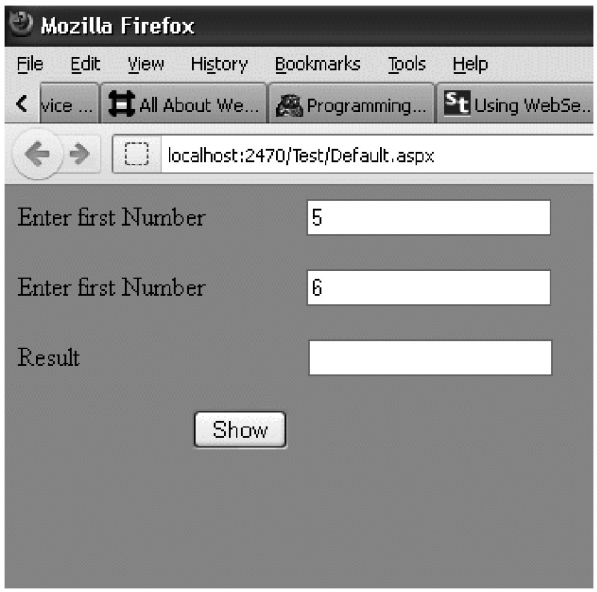
Step 10: Go to Default.cs page and on the button click event use the following code.

```
Protected void Button1_Click(object sender, EventArgs ex
{
localhost.Service my = new localhost.Service(), // you need to create the object of the web service
inta=Convert.ToInt32(TextBox1.Text);
intb=Convert.ToInt32(TextBox2.Text);
intc=mys.Multiplication(a,b);
TextBox3.Text=c.ToString();
}
```

Step 11: After pressing the F5 function key to run the website, you will see:

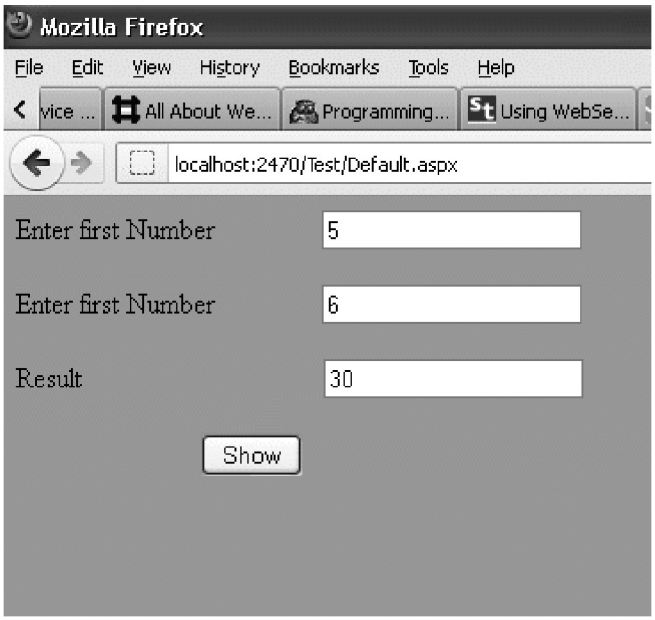


Enter the number.



Press the show button.

Notes



10.2 XML Web Services

XML Web services are the fundamental building blocks in the move to distributed computing on the Internet. Open standards and the focus on communication and collaboration among people and applications have created an environment where XML Web services are becoming the platform for application integration. Applications are constructed using multiple XML Web services from various sources that work together regardless of where they reside or how they were implemented.

There are probably as many definitions of XML Web Service as there are companies building them, but almost all definitions have these things in common:

- XML Web Services expose useful functionality to Web users through a standard Web protocol. In most cases, the protocol used is SOAP.
- XML Web services provide a way to describe their interfaces in enough detail to allow a user to build a client application to talk to them. This description is usually provided in an XML document called a Web Services Description Language (WSDL) document.
- XML Web services are registered so that potential users can find them easily. This is done with Universal Discovery Description and Integration (UDDI).

One of the primary advantages of the XML Web services architecture is that it allows programs written in different languages on different platforms to communicate with each other in a standards-based way. Those of you who have been around the industry a while are now saying, How is this any different?" The first difference is that SOAP is significantly less complex than earlier approaches, so the barrier to entry for a standards-compliant SOAP implementation is significantly lower which at last count contained 79 entries. You will find SOAP implementations from most of the big software companies, as you would expect, but you will also find many implementations that are built and maintained by a single developer. The other significant advantage that XML Web services have over previous efforts is that they work with standard Web protocols—XML, HTTP and TCP/IP. A significant number of companies already have a Web infrastructure, and people with knowledge and experience in maintaining it, so again, the cost of entry for XML Web services is significantly less than previous technologies.

Notes

The next logical question is. "What can we do with XML Web services?" The first XML Web services tended to be information sources that you could easily incorporate into applications—stock quotes, weather forecasts, sports scores etc. It is easy to imagine a whole class of applications that could be built to analyze and aggregate the information you care about and present it to you in a variety of ways; for example, you might have a Microsoft® Excel spreadsheet that summarizes your whole financial picture—stocks, 401K, bank accounts, loans, etc. If this information is available through XML Web services, Excel can update it continuously. Some of this information will be free and some might require a subscription to the service. Most of this information is available now on the Web, but XML Web services will make programmatic access to it easier and more reliable.

Exposing existing applications as XML Web services will allow users to build new, more powerful applications that use XML Web services as building blocks. For example, a user might develop a purchasing application to automatically obtain price information from a variety of vendors, allow the user to select a vendor, submit the order and then track the shipment until it is received. The vendor application, in addition to exposing its services on the Web, might in turn use XML Web services to check the customer's credit, charge the customer's account and set up the shipment with a shipping company.

In the future, some of the most interesting XML Web services will support applications that use the Web to do things that cannot be done today. For example, one of the services that XML Web Services would make possible is a calendar service. If your dentist and mechanic exposed their calendars through this XML Web service, you could schedule appointments with them on line or they could schedule appointments for cleaning and routine maintenance directly in your calendar if you like. With a little imagination, you can envision hundreds of applications that can be built once you have the ability to program the Web.

10.2.1 SOAP

SOAP is the communications protocol for XML Web services. When SOAP is described as a communications protocol, most people think of DCOM or CORBA and start asking things like, "How does SOAP do object activation?" or "What naming service does SOAP use?" While a SOAP implementation will probably include these things, the SOAP standard does not specify them. SOAP is a specification that defines the XML format for messages—and that's about it for the required parts of the spec. If you have a well-formed XML fragment enclosed in a couple of SOAP elements, you have a SOAP message. Simple is not it?

There are other parts of the SOAP specification that describe how to represent program data as XML and how to use SOAP to do Remote Procedure Calls. These optional parts of the specification are used to implement RPC-style applications where a SOAP message containing a callable function, and the parameters to pass to the function, is sent from the client, and the server returns a message with the results of the executed function. Most current implementations of SOAP support RPC applications because programmers who are used to doing COM or CORBA applications understand the RPC style. SOAP also supports document style applications where the SOAP message is just a wrapper around an XML document. Document-style SOAP applications are very flexible and many new XML Web services take advantage of this flexibility to build services that would be difficult to implement using RPC.

The last optional part of the SOAP specification defines what an HTTP message that contains a SOAP message looks like. This HTTP binding is important because HTTP is supported by almost all current OS's (and many not-so-current OS's). The HTTP binding is optional, but almost all SOAP implementations support it because it is the only standardized protocol for SOAP. For this reason, there are a common misconception that SOAP requires HTTP. Some implementations support MSMQ, MQ Series, SMTP, or TCP/IP transports, but almost all current XML Web

Notes

services use HTTP because it is ubiquitous. Since HTTP is a core protocol of the Web, most organizations have a network infrastructure that supports HTTP and people who understand how to manage it already. The security, monitoring, and load-balancing infrastructure for HTTP are readily available today.

10.2.2 WSDL

WSDL (often pronounced whiz-dull) stands for Web Services Description Language. For our purposes, we can say that a WSDL file is an XML document that describes a set of SOAP messages and how the messages are exchanged. In other words, WSDL is to SOAP what IDL is to CORBA or COM. Since WSDL is XML, it is readable and editable but in most cases, it is generated and consumed by software.

To see the value of WSDL, imagine you want to start calling a SOAP method provided by one of your business partners. You could ask him for some sample SOAP messages and write your application to produce and consume messages that look like the samples, but this can be error-prone. For example, you might see a customer ID of 2837 and assume is an integer when in fact it is a string. WSDL specifies what a request message must contain and what the response message will look like in unambiguous notation.

The notation that a WSDL file uses to describe message formats is based on the XML Schema standard which means it is both programming-language neutral and standards-based which makes it suitable for describing XML Web services interfaces that are accessible from a wide variety of platforms and programming languages. In addition to describing message contents, WSDL defines where the service is available and what communications protocol is used to talk to the service. This means that the WSDL file defines everything required to write a program to work with an XML Web service. There are several tools available to read a WSDL file and generate the code required to communicate with an XML Web service. Some of the most capable of these tools are in Microsoft Visual Studio .NET.

10.2.3 UDDI

Universal Discovery Description and Integration (UDDI) is the yellow pages of Web services. You can, of course, offer a Web service without registering it in UDDI, just as you can open a business in your basement and rely on word-of-mouth advertising but if you want to reach a significant market, you need UDDI so your customers can find you.

A UDDI directory entry is an XML file that describes a business and the services it offers. There are three parts to an entry in the UDDI directory. The "white pages" describe the company offering the service: name, address, contacts, etc. The "yellow pages" include industrial categories based on standard taxonomies such as the North American Industry Classification System and the Standard Industrial Classification. The "green pages" describe the interface to the service in enough detail for someone to write an application to use the Web service. The way services are defined is through a UDDI document called a Type Model or t-Model. In many cases, the t-Model contains a WSDL file that describes a SOAP interface to an XML Web service, but the t-model is flexible enough to describe almost any kind of service.

10.2.4 General Principles of Web Services XML

The general principles are given as:

Principle of Core Content

If you consider the information in question to be part of the essential material that is being expressed or communicated in the XML, put it in an element. For human-readable documents this generally means the core content that is being communicated to the reader. For machine-oriented records formats this generally means the data that comes directly from the problem domain.

Notes

If you consider the information to be peripheral or incidental to the main communication, or purely intended to help applications process the main communication, use attributes. This avoids cluttering up the core content with auxiliary material. For machine-oriented records formats, this generally means application-specific notations on the main data from the problem-domain.

Principle of Structured Information

If the information is expressed in a structured form, especially if the structure may be extensible, use elements. On the other hand: If the information is expressed as an atomic token, use attributes. Elements are the extensible engine for expressing structure in XML. Almost all XML processing tools are designed around this fact, and if you break down structured information properly into elements, you will find that your processing tools complement your design, and that you thereby gain productivity and maintainability. Attributes are designed for expressing simple properties of the information represented in an element. If you work against the basic architecture of XML by shoehorning structured information into attributes you may gain some specious terseness and convenience, but you will probably pay in maintenance costs.

Dates are a good example: A date has fixed structure and generally acts as a single token, so it makes sense as an attribute (preferably expressed in ISO-8601). A personal name has surprisingly variable structure (in some cultures you can cause confusion or offense by omitting honorifics or assuming an order of parts of names). A personal name is also rarely an atomic token. As an example, sometimes you may want to search or sort by a forename and sometimes by a surname. Thus:

```
<customer>
 <name>Gabriel Okara</name>
 <occupation>Poet</occupation>
</customer>
```

is not much better than:

```
<customer name= “Gabriel Okara”>
 <occupation>Poet</occupation>
</customer>
```

Principle of Readability

If the information is intended to be read and understood by a person, use elements. In general this guideline places prose in element content. If the information is most readily understood and digested by a machine, use attributes. In general this guideline means that information tokens that are not natural language go in attributes.

In some cases, people can decipher the information being represented but need a machine to use it properly. URLs are a great example: People have learned to read URLs through exposure in Web browsers and e-mail messages, but a URL is usually not much use without the computer to retrieve the referenced resource. Some database identifiers are also quite readable (although established database management best practice discourages using IDs that could have business meaning), but such IDs are usually props for machine processing.

Principle of Element/Attribute Binding

Use an element if you need its value to be modified by another attribute. XML establishes a very strong conceptual bond between an attribute and the element in which it appears. An attribute provides some property or modification of that particular element. Processing tools for XML tend to follow this concept and it is almost always a terrible idea to have one attribute modify another. For example, if you are designing a format for a restaurant menu and you include the portion sizes of items on the menu, you may decide that this is not really important to the

Notes

typical reader of the menu format so you apply the Principle of core content and make it an attribute. The first attempt is:

```
<menu>
    <menu-item portion= “250 mL”>
        <name>Small soft drink</name>
    </menu-item>
    <menu-item portion= “500 g”>
        <name>Sirloin steak</name>
    </menu-item>
</menu>
```

Following the Principle of structured information you decide not to shoehorn the portion measurement and units into a single attribute, but instead of using an element, you opt for:

```
<menu>
    <menu-item portion-size= “250” portion-unit= “mL”>
        <name>Small soft drink</name>
    </menu-item>
    <menu-item portion-size= “500” portion-unit= “g”>
        <name>Sirloin steak</name>
    </menu-item>
</menu>
```

The attribute portion-unit now modifies portion-size, which as mentioned is a bad idea. An attribute on the element menu-item should modify that element, and nothing else. The solution is to give in and use an element:

```
<menu>
    <menu-item>
        <portion unit= “mL”>250</portion>
        <name>Small soft drink</name>
    </menu-item>
    <menu-item>
        <portion unit= “g”>500</portion>
        <name>Sirloin steak</name>
    </menu-item>
</menu>
```

This is one of those cases that are less cut and dried, and other schemes might be as suitable as mine. The solution also involves contradicting the original decision to put the portion size into an attribute based on the Principle of core content. This illustrates that sometimes the principles will lead to conflicting conclusions where you will have to use your own judgment to decide on each specific matter.

Web Services Classification

Classification of XML Appliances

Although the term XML appliance is the most general term to describe these devices, most vendors use alternative terminologies that describe more specific functionality of these devices.

Notes

The following are alternative names used for XML Appliances:

XML accelerators: Are devices that typically use custom hardware or software built on standards-based hardware to accelerate XPath processing. This hardware typically provides a performance boost between 10 and 100 times in the number of messages per second that can be processed.

Integration appliance: (also known as application routers) are devices that are designed to make the integration of computer systems easier.

XML firewalls are classes of XML appliances focused on identity and message security. They typically implement WS-Security message standards along with standards like SAML, WS-I BSP, WS-Policy and so forth.

Message: Oriented middleware appliances - are hardware devices supporting the sending and receiving of messages between distributed systems.

- SOA Gateways are commonly used to govern SOA traffic.
- API proxy are commonly used to manage Web API's.

Self Assessment Questions

1. …… are distributed application components that are externally available.

   (*a*) NET provides (*b*) Integration

   (*c*) Pointers (*d*) Web services

2. A significant advantage of web services, relative to other ……, is the support of firewalls.

   (*a*) WSDL (*b*) Access Protocol

   (*c*) distributed architectures (*d*) SOAP

3. Elements are the extensible engine for expressing structure in …….

   (*a*) Java (*b*) PHP

   (*c*) UDDI (*d*) XML

4. Oracle is developing a java.net project called Metro.

   (*a*) True (*b*) False

True or False

5. Integration appliance is devices that are not designed to make the integration of computer systems easier.

   (*a*) True (*b*) False

10.3 Using for Web Services

A Web service is a system of communiqué between two electronic devices over the Web (Internet).

The W3C defines a "Web service" as "a software system intended to support interoperable machine-to-machine interaction over a network". It has an interface described in a machine-processable format (specifically Web armed forces Description words, known by the acronym WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards."

Notes

The W3C also states, "We can identify two major classes of Web services, REST-compliant Web services, in which the primary purpose of the service is to manipulate XML representations of Web resources using a uniform set of "stateless" operations; and arbitrary Web services, in which the service may expose an arbitrary set of operations

SOAP Webservices provide a simple way of providing public information in websites. Directories, maps, translations, searches and zip code/postcode lookup all suddenly become quick and easy to implement, especially in .NET. Windows Live Webservice has now joined the market with a whole range of services which are likely to change the way we use the Web.

A simple XML Web service using ASP.NET is relatively easy and is covered in Building XML Web Services Using ASP.NET Basics. However, the true power of XML Web services is realized when you look at the infrastructure. XML Web services are built on top of the .NET Framework and the common language runtime. An XML Web service can take advantage of these technologies. For instance, the performance, state management, and authentication supported by ASP.NET can all be taken advantage of by building XML Web services using ASP.NET.

The infrastructure for XML Web services is built to conform to industry standards such as SOAP, XML, and WSDL, and this allows clients from other platforms to interoperate with XML Web services. As long as a client can send standards-compliant SOAP messages, formatted according to a service description, that client can call an XML Web service created using ASP.NET (regardless of the platform on which the client resides). When you build an XML Web service using ASP.NET, it automatically supports clients communicating using the SOAP, HTTP-GET, and HTTP-POST protocols. Since HTTP-GET and HTTP-POST support passing messages in URL-encoded name-value pairs, the data type support for these two protocols is not as rich as that supported for SOAP. In SOAP, which passes data to and from the XML Web service using XML, you can define complex data types using XSD schemas, which support a richer set of data types. Developers building an XML Web service using ASP.NET do not have to explicitly define complex data types they expect using an XSD schema. Rather, they can simply build a managed class. ASP.NET handles mapping class definitions to an XSD schema and mapping object instances to XML data in order to pass it back and forth across a network.

Why use Web Services

The major reasons for using Web services are to gain:

1. Interoperability among distributed applications that span diverse hardware and software platforms.
2. Accessibility of applications through firewalls using Web protocols.
3. Across-platform, cross-language data model (XML) that facilitates developing heterogeneous distributed applications.

Because Web services are accessed using standard Web protocols, such as XML and HTTP, the diverse and heterogeneous applications on the Web (which typically already understand XML and HTTP) can automatically access Web services, solving the ever-present problem of how different systems communicate with each other. These different systems might be Microsoft SOAP Toolkits clients, J2EE applications, legacy applications, and so on. These systems might be written in a variety of programming languages, such as Java, C++, or Perl. As long as the application that provides the functionality is packaged as a Web service each of these systems can communicate with any other.

Web Services represent a new standard-based and simplified model for creating and connecting distributed applications across the web in the form of services. Web Services are built on the top of existing and widely adopted Internet protocols such as HTTP, XML, TCP/IP , HTML , Java and XML . This means the base foundation for building the Web Services is already in place.

Notes

From a more technical perspective, Web Services are XML depictions of objects, messages, and documents designed to interact over the web to enable application integration. Web Services applications can be published found or invoked as atom like services any where ion the internet thus creating service grid of dynamic business components.

By providing a common interoperability layer between disparate platform, Web Services allow companies to connect with each other based on business considerations as opposed to underlying infrastructure requirements. The benefit will come in the form of enhanced user experience as a much wider variety of services are offered to customers. As business begin to adopt the Web Services scenario, whereby vendors strictly focus on their core competencies and aggregate supplementary services. Further more one of the most attractive aspects of the Web Services is that there is now a significant amount of additional technological investment in the application server technology , so these companies can begin taking advantage of the Web Services today.

The advantages of using these web services include:

- A full blown SOAP Webservice allowing you to take control of the data
- The ability to search multiple websites from your company and/or its partners
- Not having to worry about indexing the content of your company's global website(s)
- Harnessing the power of Live Search as advanced search features in your queries
- Using Virtual Earth to map out streets in your local area.

The limitations include:

A limit of 10,000 queries a day to use the service for free. (Note: Normally each page in the Grid view Control will fire a new query even though the search query is the same. This demo makes use of sessions to prevent this from happening.– Optional in web.config)

Case Study

AWS Case Study: Harvard Medical School

The Laboratory for Personalized Medicine (LPM), of the Center for Biomedical Informatics at Harvard Medical School, run by Dr. Peter Tonellato, took the power of high throughput sequencing and biomedical data collection technologies and the flexibility of Amazon Web Services (AWS) to develop innovative whole genome analysis testing models in record time. "The combination of our approach to biomedical computing and AWS allowed us to focus our time and energy on simulation development, rather than technology, to get results quickly," said Tonellato. "Without the benefits of AWS, we certainly would not be as far along as we are."

Tonellato's lab focuses on personalized medicine—preventive healthcare for individuals based on their genetic characteristics—by creating models and simulations to assess the clinical value of new genetic tests.

Other projects include simulating large patient populations to aid in clinical trial simulations and predictions. To overcome the difficulty of finding enough real patient data for modeling, LPM creates patient avatars—literally "virtual" patients. The lab can create different sets of avatars for different genetic tests and then replicate huge numbers of them based on the characteristics of hospital populations. Tonellato needed to find an efficient way to manipulate many avatars, sometimes as many as 100 million at a time.

In 2006, Tonellato turned to cloud computing to address the complex and highly variable computational need. "I evaluated several alternatives but found nothing as flexible and robust

Contd...

Notes

as Amazon Web Services," he said. Having built datacenters previously, Tonellato could not afford the time he knew would be required to set up servers and then write code. Instead, he decided to conduct a test to see how fast his team could put together a series of custom Amazon Machine Images (AMIs) that would reflect the optimal development environment for researchers' web applications.

Now, Tonellato's lab has extended their efforts to integrate Spot instances into their workflows so that they could stretch their grant money even further. According to Tonellato, "We leverage Spot instances when running EC2 clusters to analyze entire genomes. We have the potential to run even more worker nodes at less cost when using Spot, so it is a huge saving in both time and cost for us. To take advantage of these savings, it just took us a day of engineering, and saw roughly 50% savings in cost." Tonellato's lab leverages MIT's StarCluster tools, which has built-in capabilities to manage an Oracle Grid Engine Cluster on Spot Instances. Erik Gafni, a programmer in Tonellato's lab, performed the integration of StarCluster into our workflow. According to Gafni, "Using StarCluster, it was incredibly easy to configure, launch, and start using a running Spot Cluster in less than 10 minutes."

In addition the LPM recognized the need for published resources about how to effectively use cloud computing in an academic environment and published an educational primer in PLoS Computational Biology to address this need. "We believe this topic clearly shows how an academic lab can effectively use AWS to manage their computing needs. It also demonstrates how to think about computational problems in relation to AWS costs and computing resources," says Vincent Fusaro, lead author and senior research fellow in the LPM.

"The AWS solution is stable, robust, flexible, and low cost," Tonellato commented. "It has everything to recommend it."

Tonellato runs his simulations on Amazon Elastic Compute Cloud (Amazon EC2), which provides customers with scalable compute capacity in the cloud. Designed to make web-scale computing easier for developers, Amazon EC2 makes it possible to create and provision compute capacity in the cloud within minutes.

Tonellato's lab is thrilled with their AWS solution. "The number of genetic tests available to doctors and hospitals is constantly increasing," Tonellato explained, "and they can be very expensive. We're interested in determining which tests will result in better patient care and better results." He added, "We believe our models may dramatically reduce the time it usually takes to identify the tests, protocols, and trials that are worth pursuing aggressively for both FDA approval and clinical use."

Questions

1. What are the advantages of Amazon Web services?
2. How did it provide the service to the school?

Self Assessment Questions

6. Web Service is a …….. that uses XML to exchange information with other software via common internet protocols.

   (*a*) software program (*b*) hardware program

   (*c*) Both of these (*d*) None of these.

7. XML Web services interfaces that are accessible from a ……. variety of platforms and programming languages.

   (*a*) short (*b*) wide

   (*c*) Both of these (*d*) None of these.

Notes

8. …….. and Integration is the yellow pages of Web services.

   (*a*) FDDI (*b*) UDDI

   (*c*) WSDL (*d*) None of these.

9. XML establishes a very …….. conceptual bond between an attribute and the element in which it appears. An attribute

   (*a*) Weak (*b*) Normal

   (*c*) Strut (*d*) None of these.

10. A Web service is a method of communication between …….. Over the Web.

    (*a*) Two electronic device (*b*) Three electronic device

    (*c*) Four electronic device (*d*) None of these.

11. The infrastructure for XML Web services is built to conform to industry standards such as ……., and this allows clients from other platforms to interoperate with XML Web services.

    (*a*) XML (*b*) WSDL

    (*c*) SOAP (*d*) All of these.

12. …….. is the communications protocol for XML Web services

    (*a*) WSDL (*b*) SOAP

    (*c*) UDDI (*d*) None of these.

13. SOAP is a

    (*a*) Simple Object Access Protocol (*b*) Standard Object Access Protocol

    (*c*) Simple Oriented Access Protocol (*d*) None of these.

14. Which of the following object is not an ASP component?

    (*a*) LinkCounter (*b*) Counter

    (*c*) AdRotator (*d*) FileAccess

15. Web services using the HTTP protocol on port 80, which is generally not secured.

    (*a*) True (*b*) False

10.4 Summary

- The W3C defines a "Web service" as "a software system designed to support interoperable machine-to-machine interaction over a network.
- The infrastructure for XML Web services is built to conform to industry standards such as SOAP, XML, and WSDL, and this allows clients from other platforms to interoperate with XML Web services.
- When you build an XML Web service using ASP.NET, it automatically supports clients communicating using the SOAP, HTTP-GET, and HTTP-POST protocols.
- Web services are distributed application components that are externally available. You can use them to integrate computer applications that are written in different languages and run on different platforms.

Notes

- This directive allows you to refer to objects in the System.Web.Services namespace without having to fully qualify the request.
- Open standards and the focus on communication and collaboration among people and applications have created an environment where XML Web services are becoming the platform for application integration.
- SOAP is the communications protocol for XML Web services.
- WSDL specifies what a request message must contain and what the response message will look like in unambiguous notation.
- WSDL file and generate the code required to communicate with an XML Web service.

10.5 Keywords

Application Programming Interface (API): The word to really pay attention to is "Interface ". If you have any experience at all with programming, all kinds of abstractions and contracts must be coming to your mind when you hear the word "interface" but we are more interested in the classical meaning of the term.

CORBA: The acronym for Common Object Request Broker Architecture, is a widely used communications model for building distributed (multi-tier) applications that connect both cross-platform and cross-language clients to server-based services.

MSMQ: It is a message queuing framework used for applications that use messaging infrastructure. MSMQ is a tool for sending and receiving messages.

RPC-style Applications: XML-RPC.NET is a library for implementing XML-RPC Services and clients in the .NET environment, supporting versions 3.5 and upwards of the .NET runtime.

Service Oriented Architecture (SOA): A service-oriented architecture can be defined as a group of services, which communicate with each other. The process of communication involves either simple data passing or it could involve two or more services coordinating some activity. Some means of connecting services to each other is needed.

SOA Gateways: The term "SOA Governance" was used to describe everything from design time management of test and design artifacts, monitoring, enforcement, UDDI registry storage.

Stateless: That is, information that is generated on one page is not normally accessible by any other page

10.6 Review Questions

1. What is the Web Services and also described Web Service Example?
2. Example of Creating Web Service in .NET.
3. Explain the XML Web Services.
4. Write short notes on the following.

   (1) SOAP (2) WSDL (3) UDDI
5. Explain the principles of Web Services XML.
6. Described Web Services Classification.
7. The concept of Using for Web Services.

Notes

8. Described the advantages of using web services include and the limitations include.
9. Why use Web Services?
10. What is the SOAP? Define use of SOAP in the Web services.

Answers for Self Assessment Questions

1. (*d*)	2. (*c*)	3. (*d*)	4. (*a*)	5. (*b*)
6. (*a*)	7. (*b*)	8. (*b*)	9. (*a*)	10. (*b*)
11. (*b*)	12. (*b*)	13. (*a*)	14. (*a*)	15. (*c*)

10.7 Further Readings

Books *XML Web Services with ASP.NET*, by Bill Evjen

Online link http://dotnetguts.blogspot.in/2007/09/all-about-web-service-in-net.html

Unit 11: Web Services in Visual Studio .NET

Notes

CONTENTS

Objectives

After studying this unit, you will be able to:

- Explain about creating web services
- Expanding web application with web services

Introduction

A Web Service is programmable application logic available by normal Web protocols. One of these Web protocols is the Simple Object Access Protocol (SOAP) that uses standards based technologies (XML for data description and HTTP for transport) to inculcate and spread application information.

Consumers of a Web Service do not require knowing anything about the stage, object model, or indoctrination language used to implement the repair; they only need to recognize how to send and be given SOAP messages (HTTP and XML).

Web Services are simple and easy to understand. It is possible, in fact, to author a simple application that surfaces data as XML conforming to the SOAP specification. It would also be relatively straightforward to build an application capable of receiving SOAP messages over HTTP and deriving meaningful value out of it. For those of you familiar with PERL, this could simply be a matter of using RegEx to parse the value out of the XML result; it is just another string.

However, just as we use frameworks such as ASP and ASP.NET to build Web applications, we would much rather use a framework for building Web Services. The reasoning is quite logical. We do not need to reinvent the plumbing—that is, at a high level, the capability to serialize our data as XML, transport the data using HTTP, and de-serialize the XML back to meaningful data. Instead, we want a framework that makes building Web Services easy, allowing us to focus on the application logic not the plumbing. ASP.NET provides this framework for us.

Notes

From a developer's point of view, if you have ever written application logic, you have the required skills to author ASP.NET Web Services. More importantly, if you are at all familiar with ASP or ASP.NET application services, (application state memory, and so on) you can also leverage these skills when you build ASP.NET Web Services.

A Web Service in .Net consists of an .ASMX page that either contains a class that provides the Web Service functionality or references a specific external class that handles the logic in an external class file. Classes are standard .Net classes and the only difference is that every method that you want to expose to the Web is prefixed with a [Web Method] attribute. Once the .ASMX page has been created the Web Service is ready for accessing over the Web. .Net provides a very useful information page about your Web Service showing all the methods and parameters along with information on how to access the Web Service over the Web. You can also use this page to test basic operation of your Web Service without calling the Web Service with a 'real' client.

The .Net Web Services that run over HTTP can be called in three different ways:

- ***HTTP GET Operation:*** You can pass parameters to a Web Service by calling the ASMX page with query string parameters for the method to call and the values of simple parameters to pass.
- ***HTTP POST Operation:*** Works the same as GET Operation except that the parameters are passed as standard URL encoded form variables. If you use a client such as wwIPStuff you can use AddPostKey () to add each parameter in the proper parameter order.
- ***SOAP:*** this is the proper way to call a Web Service in .Net and it is also the way that .Net uses internally to call Web Services.

The GET and POST operations are useful if you need to call a Web Service quickly and no SOAP client is readily available. For example, in a browser based client application it may be easier to use GET and POST instead of constructing and parsing the more complex SOAP headers that are passed back and forth in a SOAP request. But with a proper SOAP client in place SOAP provides the full flexibility of the protocol, where GET and POST operations have to stick to simple inputs and outputs. Among other things that you can do with SOAP is pass complex objects and data over the wire and for these operations to work you need to use SOAP.

Behind the scenes there are three major components that make up a Web Service:

- The Web Service on the Server side
- The client application calling the Web Service via a Web Reference
- A WSDL Web Service description that describes the functionality of the Web Service.

11.1 Creating Web Services

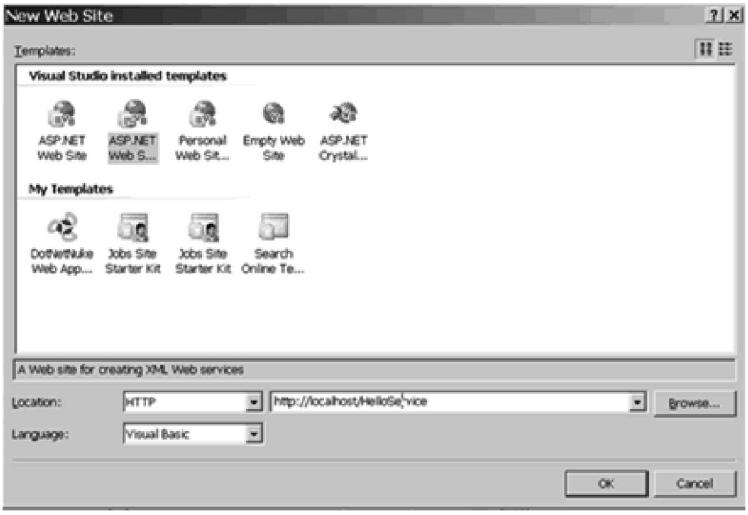
Creating a Web service in Visual Studio .NET is very easy. ASP.NET server side knowledge allows developers to create web services and the .NET framework provides a sophisticated position of tools and rules to devour web services.

Let us start by creating the Hello world web service. We will create the web service using asp. net and visual basic language. The IDE we will use is Visual Studio. NET. We can also create an ASP.NET Web Service in C# programming language using Visual Studio .NET.

To create the web service, open Visual Studio and select File-> New Web site from the menu. Make sure Visual Basic is selected in the language box and HTTP in the location box. Select ASP.NET Web service from the template list. Enter the name as http://localhost/HelloService as shown in Figure 11.1 and click OK.

Notes

Figure 11.1: Creating new Web Service



A new virtual directory - Hello Service - will be created on your computer's Web server. This action also creates two files called Service.asmx and service.vb. Please note that Visual Studio creates the entry point file with .asmx extension and the other code behind file with .vb extension. These files are automatically added to the project, which we can see in a solution explorer. The service.asmx amd service.vb files represent a single web service. Web Services do not have a visible interface and so, the design interface is empty. When we open the Service.asmx file, only the following code is seen.

<%@ WebService Language= "vb" CodeBehind= "~/App_Code/Service.
vb" Class= "Service" %>

The service.asmx file serves as the entry point into the web service. The other file, service.vb, is the code-behind file for the web service. We can write the functionality of the web service in this file. The code-behind file imports the System.Web.Services namespace. This namespace contains the classes that are required to build and use Web services. We can view this file by switching to the code view for the service.asmx file.

Note that a single project can contain multiple Web Services, each web service implemented by a different class. If we open the service.vb code-behind page, we can see the default code which is created by vs.NET automatically, as listed below.

```
Import System. Web
Import System.Web.Services
Import System.Web.Services. Protocols
<WebService (Namespace:= "http://tempuri.org/")>
<WebServiceBinding (Conforms To: =WsiProfiles.BasicProfile1_1)>
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerted ()>_
Public Class Service
Inherits System.Web.Services. WebService
<WebMethod ()>
Public Function Hello World () As String
Return "Hello World"
End Function
End Class
```

Notes

From the above code, we see that service is the class that contains all the code that implements the web service methods. This is actually the name of the WebService; it is named after the class and not after the project. The .vb and .asmx files together form the Code model for the web services.

As we have known a web service is basically a class that sits on the server. Building a Web Service with Visual Studio is as simple as writing a class. The class inherits from System.Web. Services and this base class includes all that is necessary to make the functions (methods) of the class available on the Web. The Web Service class belongs to the System.Web.Services namespace. This class provides direct access to built-in ASP.NET objects, such as Application and Session. If an application does not require these objects, we can create a Web service without deriving it from the Web Service class. So, Note that deriving from a web service base class is optional.

Examining the Web Service Files

Service.asmx file.

<%@ WebService Language= "vb" CodeBehind= "~/App_Code/Service. vb" Class= "Service" %>

In the above line of code, a WebService is the web processing directive that specifies a class name; here service for a web service and the language used to create the web service is visual basic. Service.vb file. Explanation of the WebService attributes. The WebService attribute is an optional attribute that can be used to provide additional information about the Web service. We can also use the WebService attribute to provide a unique namespace for the web service. This namespace can be used by client applications to resolve conflicts that may arise due to different Web services having methods with the same name. The Webservice attribute can be provided before the Web service class declarations as shown below.

```
<WebService (Namespace:=http://vkinfotek.com/services,
Description: = "This Web service contains a method for displaying Hello world string")>
Public Class Service
Inherits System.Web.Services.WebService
End class
```

The description of the Web service appears in the browser window when we try to access the web service directly by providing the path. The WebMethod Attribute. The methods that we want to expose to web service are marked with WebMethod attribute. The WebMethod attribute enables us to call remotely, the web service methods over the internet. The syntax for applying the WebMethod attribute to a method is given below. Syntax for applying the WebMethod attribute to a method is given below.

```
<WebMethod ()> Public Function functionname (Parameter list) As ReturnType
```

The methods of this class do not return an HTML page; instead, they return one or more values, packaged as XML documents. Basically, if you can write a function, you can write a web method; it is that simple. The difference between a class and a WebService class is that the members of the WebService can be called remotely. VB.NET uses less than and greater than symbols to the attribute to appear on the line before the function definition.

From the above code, we note that class name is Service and method is Hello World. If we want to change its name, we must rename the Service item in the Solution Explorer's window, as well as the name of the class in the code. Note that we can also create methods without preceding them with the WebMethod attribute. However, these methods cannot be used to expose some functionality to applications. They can only be used by the other methods within the Web service class.

A Web Service Example

Notes

In the following example we will use ASP.NET to create a simple Web Service that converts the temperature from Fahrenheit to Celsius, and vice versa:

```
<%@ WebService Language= "VBScript" Class= "TempConvert" %>
Imports System
Imports System.Web.Services
Public Class TempConvert: Inherits WebService
<WebMethod ()> Public Function FahrenheitToCelsius
As String
dim fahr
fahr=trim(replace(Fahrenheit, ",", "."))
if fahr= "" or IsNumeric(fahr)=false then return "Error"
return ((((fahr) - 32) / 9) * 5)
End function
<WebMethod()> Public Function CelsiusToFahrenheit
As String
dim cel
cel=trim(replace(Celsius, ",", "."))
if cel= "" or IsNumeric(cel)=false then return "Error"
return ((((cel) * 9) / 5) + 32)
End function
End class
```

This document is saved as an .asmx file. This is the ASP.NET file extension for XML Web Services.

Task Create a Web service.

Did u know?

The SOAP was designed as an object-access protocol in 1998 by Dave Winer, Don Box, Bob Atkinson, and Mohsen Al-Ghosein for Microsoft, where Atkinson and Al-Ghosein were working at the time.

Self Assessment Questions

1. ASP.NET server side technology not allows developers to create web services.

   (*a*) True (*b*) False

2. Web browsers are software applications that allow users to retrieve data and interact with content located on web pages within a website.

   (*a*) True (*b*) False

3. ……..is one form of client side script that permits dynamic elements on each page.

   (*a*) Java (*b*) PHP

   (*c*) JavaScript (*d*) XML

Notes

4. A Web Service is programmable application logic accessible via standard……...

   (*a*) Web service (*b*) Web protocols

   (*c*) Web defender (*d*) None of these

5. .NET framework provides a ……..and code to consume web services.

   (*a*) Collection technologies (*b*) HTTP protocols

   (*c*) Protocols (*d*) Sophisticated set of tools

11.2 Expanding Web Application with Web Services

Over the past decade or so, the web has been embraced by millions of businesses as an inexpensive channel to exchange a few words and exchange information with forecast and transactions with clientele.

In particular, the web provides a way for marketers to get to know the public visiting their sites and create communicating with them. One way of doing this is asking Web Company to subscribe to newsletters, to submit an application form when requesting information on products or provide details to customize their browsing experience when next visiting a particular website.

The web is also an excellent sales channel for a myriad of organizations, large or small: with over 1 billion Internet users today. All this data must be somehow captured, stored, processed and transmitted to be used immediately or at a later date. Web applications, in the form of submit fields, enquiry and login forms, shopping carts, and content management systems, are those website widgets that allow this to happen.

They are, therefore, fundamental to businesses for leveraging their online presence thus creating long-lasting and profitable relationships with prospects and customers.

No wonder web applications have become such a ubiquitous phenomenon. However, due to their highly technical and complex nature, web applications are a widely unknown and a grossly misunderstood fixture in our everyday cyber-life.

11.2.1 Web Applications Defined

From a technical view-point, the web is a highly programmable environment that allows mass customization through the immediate deployment of a large and diverse range of applications, to millions of global users. Two important components of a modern website are flexible web browsers and web applications; both available to all and sundry at no expense.

Web browsers are software applications that allow users to retrieve data and interact with content located on web pages within a website.

Today's websites are a far cry from the static text and graphics showcases of the early and mid-nineties: Modern web pages allow personalized dynamic content to be pulled down by users according to individual preferences and settings. Furthermore, web pages may also run client-side scripts that "change" the Internet browser into an interface for such applications as web mail and interactive mapping software (e.g., Yahoo Mail and Google Maps).

Most importantly, modern web sites allow the capture, processing, storage and transmission of sensitive customer data (e.g., personal details, credit card numbers, social security information, etc.) for immediate and recurrent use. And, this is done through web applications. Such features as webmail, login pages, support and product request forms, shopping carts and content management systems, shape modern websites and provide businesses with the means necessary to communicate with prospects and customers. These are all common examples of web applications.

Notes

Web applications are, therefore, computer programs allowing website visitors to submit and retrieve data to/from a database over the Internet using their preferred web browser. The data is then presented to the user within their browser as information is generated dynamically (in a specific format, e.g. in HTML using CSS) by the web application through a web server.

For the more technically oriented, Web applications query the content server (essentially a content repository database) and dynamically generate web documents to serve to the client (people surfing the website). The documents are generated in a standard format to allow support by all browsers (e.g., HTML or XHTML). JavaScript is one form of client side script that permits dynamic elements on each page (e.g., an image changes once the user hovers over it with a mouse). The web browser is key - it interprets and runs all scripts etc. while displaying the requested pages and content.

Another significant advantage of building and maintaining web applications is that they perform their function irrespective of the operating system and browsers running client side. Web applications are quickly deployed anywhere at no cost and without any installation requirements (almost) at the user's end.

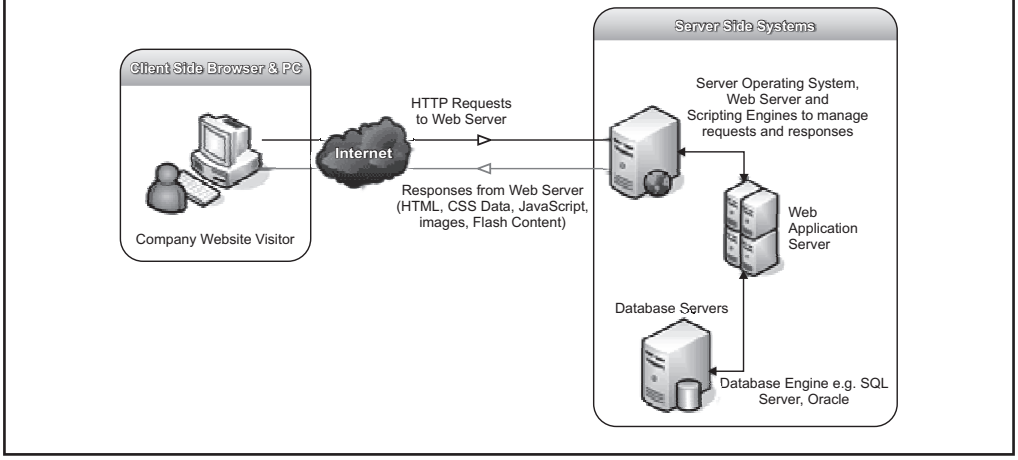
As the number of businesses embracing the benefits of doing business over the web increases, so will the use of web applications and other related technologies continue to grow. Moreover, since the increasing adoption of intranets and extranets, web applications become greatly entrenched in any organization's communication infrastructures, further broadening their scope and possibility of technological complexity and prowess.

Web applications may either be purchased off-the-shelf or created in-house.

11.2.2 How do Web Applications Work?

The Figure 11.2 details the three-layered web function model. The first layer is usually a web browser or the user interface; the second layer is the active content production technology tool such as Java servlets (JSP) or Active Server Pages (ASP), and the third layer is the database containing content (e.g., news) and customer data (e.g., usernames and passwords, social sanctuary information and credit card details).

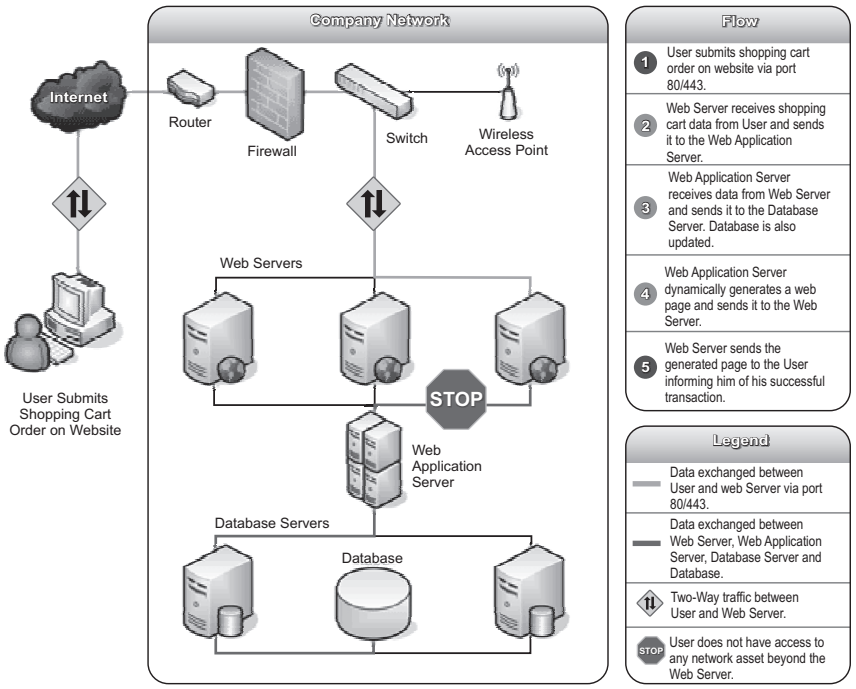
Figure 11.2: Three-layered Web Application Model



The Figure 11.3 shows how the initial demand is triggered by the user through the browser in excess of the Internet to the web function server. The web application accesses the databases servers to perform the requested mission updating and retrieving the in order lying within the database. The web application then presents the information to the user through the browser.

Notes

Figure 11.3: Initial Request is Triggered by the User Through the Browser Over the Internet to the Web Application Server



11.2.3 Types of Web Applications

There are three main types of web applications:

Customer-facing: Applications are known as ecommerce or B2C sites and use the internet. These typically present a customer with choices of products or services to buy using a shopping cart and payment method. Examples are travel reservations.

Employee-facing: Applications use the intranet in a company. One example is an accounting application of a company. Another might be employee expense reporting. A third might be the ERP (Enterprise Requirements Planning) system.

These applications previously operated on an internal client-server network. They are now web enabled to make them easier to use and deploy. Disparate applications, such as an ERP and CRM (Customer Relationship Management) systems are now being integrated using XML and web services.

Customer-supplier Facing: Applications are known as B2B (Business to Business) sites and use the extranet, (an extension of an intranet that allows outside companies to work in a password-protected space). B2B sites provide a secure means for sharing selected information. One example is supply chain software that allows all suppliers to see demand and inventory in the supply chain. Another example is procurement software that allows a customer to send RFQs and receive quotes over the web. A third example is collaboration software that allows companies to share product development and project management information.

Task Prepare a Web application?

Notes

11.2.4 Technologies used to Build Web Applications

Originally, the Internet was designed to serve "static" pages. A rudimentary technology based on CGI was developed to allow information to be passed back to a web server. During the last ten years, four main technologies have emerged to replace CGI and the basic CGI technology has been further refined, using Perl as the primary programming language. This has lead to 5 competing technology stacks that differ in the following attributes:

- Programming languages (Lang)
- Operating system (OS). This can be Linux (L), Unix (U) or Windows (W).
- Web server (Server)
- Database support (DB)
- Sponsoring companies (Sponsors)

The following is a more detailed explanation of each of these technology stacks:

CGI/Perl: CGI is the granddaddy of interfaces for passing data from a submitted web page to a web server. Perl is an open-source language optimized for writing server-side applications. Together, CGI and Perl make it easy to connect to a variety of databases. Apache tends to be the web server used because it runs on all major operating systems and is highly reliable. Other open-source languages such as C and Python can also be used. For high-end applications, especially e-commerce sites like amazon.com this technology is used because it is so powerful. However other technology stack can be implemented more easily and quickly.

Macromedia: Sells a collection of products that make it easy to build small and medium-sized web applications. The primary tools provided by Macromedia are ColdFusion, which is an engine that lets one program in CFML (Cold Fusion Markup Language) and Dreamweaver, which is a development tool for making web applications. Because Macromedia is a smaller player, they have focused on trying to make their products compatible with components from other technology stacks. Macromedia also sells Flash and has tools for using this in web applications.

Java/J2EE: Is a robust, well-developed method for creating medium to large web applications. It has support from a number of large industry players. Sun Microsystems provides the Java. IBM (Web sphere) and BEA Systems (Web logic) are two major suppliers of web application servers and associated software to make it easy to create and manage these applications. There is a large body of Java programmers available to write the code. This technology stack works with a variety of databases and is particularly well-tuned to mainstream commercial databases like Oracle and DB2. IBM has developed a development environment called Eclipse that is making it easier to write applications, but in general, Java is associated with powerful applications built by capable programmers.

LAMP (Linux, Apache, MySql, PHP): Is a relatively new technology stack for building web applications that has been adopted for many small and medium-size web tasks because: (a) the entire technology stack is available through open-source; (b) it works well; (c) it is easy to learn; (d) it allows one to build a web application quickly; and (e) there are many open source code samples that can be bolted together to make a full solution. LAMP relies on CGI for data exchange between the server and browser, but the CGI commands are hidden from the developer. LAMP does not have all of the capabilities of J2EE, but it gains ground every year. Sites that use PHP can be seen by the ".php" as part of the page name in the URL. LAMP has become especially popular with ISVs (Independent Software Vendors) because they can create an application and sell it without having to pay for the underlying (open source) software.

Microsoft .NET: Microsoft is using their .NET strategy to take over the server market the way in which Windows, Office, and Internet Explorer have taken over the desktop. The stack comprises

Notes

a web server (ASP.NET) and two programming languages (VisualBasic.NET and C#.NET) that compete against PHP and Java respectively. They also have a database (SQL Server). Microsoft has done an excellent job making their products easy to use so a business analyst can create a web application without needing a programmer.

Did u know?

Perl was originally developed by Larry Wall in 1987 as a general-purpose Unix scripting language to make report processing easier. Perl gained widespread popularity in the late 1990s as a CGI scripting language, in part due to its parsing abilities.

Case Study

SMS: Case Study of a Web Services Deployment

Lucin PLC is a small software consultancy headquarterd in Newport, Wales, in the United Kingdom. Mike Clark is its founder and managing director. It has specialized in transaction-oriented client-server applications for over a decade. In February 2001, three engineers began work on a small demonstration of a web services-to-SMS gateway. A few weeks later, in March, it opened the service to the public.

"Short Messaging Service" or "Short Message Sending" is widely supported in wireless telephones in most European and Asia-Pacific countries, although rarely in the USA. Wildly popular in Japan and elsewhere, SMS allows telephone users to compose short textual messages using the telephone handset, and transmit them asynchronously. Think of it as instant messaging for cellphones.

It is natural to bind together the pertinent telephony and computing protocols so that computers can originate and perhaps receive such messages. That is Lucin's achievement in its web services-to-SMS gateway; it is the first of several such for-fee gateways it plans for the months ahead:

- Web service to pager
- Web service to fax
- Web service to speech
- Web Service to TAPI (telephony application programming interface)

In Lucin's plan, it will be only a short time, probably the final quarter of 2001, before it has the capability to dial a conventional telephone, and read out an arbitrary message in a synthesized voice.

The starting point, though, is the integration of web services and SMS.

Construction of a Web Service

The SMS gateway is a small software project. Much of the media publicity around web services envisions integrating enterprise applications on large supply chain projects. Clark insists that this misses the point of the "instant gratification services" already moving into production for modest, well-specified projects. He deliberately chose SMS as a model. As Clark indicates, "[SMS] fits this [model] perfectly as a web service that's great to use," but can afford occasional outages or downtime.

Current infrastructure enforces this kind of tolerance. Networks simply are not 100% reliable or ubiquitous yet.

Contd...

Notes

Some web services vendors are hawking "revolution" with the intelligence in their web services tools. However well this software works in the laboratory, it appears few organizations have enough well-connected customers to layer so much weight over existing networks. It does not matter how clever the error-correction and transactional algorithms one programs: Until basic network connectivity gains an order of magnitude in price and reliability, there's little point to building elaborate web services. Clark summarizes, "Do not expect your users to be happy with response times. It is currently out of our hands; maybe in two years time it will change."

The inadequacy of the current networking infrastructure impels Clark to manage Lucin conservatively: "We have deliberately budgeted for a three year cycle, meaning we can pretty much last three years with no significant customer base. This is because my belief is that it is going to be a long trek before web services are thought as reliable and trustworthy enough for businesses to use in the long term."

In the meantime, the kinds of service that do satisfy users might be even simpler than you expect. Web services developers think of the typical high-level design as one that pulls in data or objects across the network, performs calculations on them, then uses web services protocols to deliver results back across the network. Lucin sells a number of web services products that fit this model and has several internal tools to support such development.

For the SMS gateway, though, Lucin eventually settled on an even more primitive approach. Clark explains that ". . . if you know the interface then you simply need to construct a simple string variable and substitute the values that change at the time you send the data to the remote web service. This saves time and the overhead of having to reference [a SOAP toolkit or module]."

Lucin does this simple string substitution, and the other light-duty calculations involved in the production version of the SMS gateway, with Visual Basic, hosted on an NT Server 4.0. This supplies more than enough performance for the several hundred messages transacted each day (most often between 300 and 800).

The gateway's simplicity extends to its authentication: A user-password combination sent in plain text by way of Hyper Text Transport Protocol (HTTP). Clark contends, "You would not expect to use HTTPS [secure HTTP] for signing onto a web site, so why use HTTPS to get users to pass down username and passwords so that you can validate them for using your web service." Pressed about the frequency of HTTPS use, Clark criticized its run-time performance, and concluded, "we may be using HTTPS ability at some stage, but it will almost certainly be by customer request."

"One of the beauties of web service development," according to Clark, "is the flexibility that comes from use of simple, robust parts." The gateway connects on the back end to a metered SMS interface provided by a telecommunications vendor. Twice already, Lucin has switched providers for this back end, but without any interruption in service to its customers. Users of the gateway just receive acknowledgment of receipt of their requests; they do not have to know or care about the back end.

Authentication, along with interfaces to validate telephonic country code and local number, make the Web Services Description Language (WSDL) instance for the gateway one of the most widely-used publicly-visible services. It defines four operations: SendTextMessage, SendMessage, ValidPhoneNumber, and GetCountryCodes. Part of the definition for the second of these, for example, is shown in Listing 1.

Contd...

Notes

Table 1: Process.xml

```
<message name= "SendMessageRequest">
<part name= "PhoneNumber" type= "xsd:string"/>
<part name= "Message" type= "xsd:string"/>
<part name= "SenderName" type= "xsd:string"/>
<part name= "SenderPassKey" type= "xsd:string"/>
</message>
```

Lucin largely recycled its expertise in Visual Basic and allied technologies for this project. Clark tells, "the web service is simply a layer to DCOM components."

The next changes for the gateway, according to Clark, are "to push it as a pay to use service," at 4.5 British pennies per call, and to contract with telecomm provisioners for development of the pager, FAX, and other gateways.

Manageability Conclusions

As a software project, the SMS gateway is small: Under a dozen engineer-weeks went into its design, implementation, and deployment. Clark is reluctant to offer specifics about the coding, for competitive reasons. Part of the argument that he does advance publicly, though, is that web services provides the advantages of component-based software development. Requirement sets apt for web services often decompose naturally into simple designs to which even junior coders can contribute.

Other public web services also appear to be small in engineering scope. XMethods.net has a gateway to PacBell SMS (in the western United States). It has an even simpler public interface (see Resources), transacting only a single WSDL operation: sendMessage. The most elaborate public service interface is Lucin's demonstration of a mailing list manager. While this defines twenty six distinct operations, all the data transacted are of simple types such as string.

So, what is special about web services development? We are all really not sure yet. None of the companies doing large-scale web services-based development are disclosing much about their experiences. At small scales, the technology "feels" like other light-duty networking development. It is a definite advantage that humans can read XML, and that so many tools are available for its management.

Clark's view is that "The problem in developing web services is not the development cycle, but trying to create a web service that people will pay for." With the web itself now starting its second decade, the basic technologies are quite well understood, but there's still plenty of dissatisfaction and experiment in getting the right "business model" for human-readable web sites. That looks to be the biggest challenge for external web services – the fourth category named in the introduction – as well.

The big change in web services development is likely to come when its "component market" has matured in the fashion of the OCX or VBX practices. The technology already emphasizes work with building blocks, and that's likely to intensify as more and more services come online. One difference, perhaps, will be the influence of open source. While OCX and VBX had proprietary definitions and components licensed for fee have always dominated that field, web services are based entirely on public standards, and many of the toolkits for this technology are open-sourced.

Questions

1. How Lucin PLC prepare the SMS project?

2. What were the challenges fronts of the team?

Self Assessment Questions

Notes

6. ……. is used to encode and transmit application data.
   (*a*) HTTP (*b*) LAMP
   (*c*) SOAP (*d*) SMTP
7. Soap messages include standard based technologies such as
   (*a*) XML (*b*) HTTP
   (*c*) Both (*a*) and (*b*) (*d*) None of these.
8. Hyper text transfer protocol is a …….. layer protocol of OSI Reference model.
   (*a*) Network (*b*) Transport
   (*c*) Application (*d*) Data link
9. ASMX page contains …….. that provides the Web Service functionality.
   (*a*) Class (*b*) Object
   (*c*) Entity (*d*) Attributes
10. ……. operations are used to call web services quickly.
    (*a*) Call in (*b*) Get
    (*c*) post (*d*) Both (*b*) and (*d*)
11. A …… Web Service description that describes the functionality of the Web Service.
    (*a*) CSS (*b*) WSDL
    (*c*) ASMX (*d*) CGI
12. The service.vb, is the ……. file for the web service.
    (*a*) code-behind (*b*) entry point
    (*c*) exe (*d*) None of these.
13. ……. contains the classes that are required to build and use Web services.
    (*a*) CFML (*b*) Namespace
    (*c*) Dreamweaver (*d*) Macromedia

True of False

14. It is not possible to build an application without using any form.
    (*a*) True (*b*) False
15. ABS () function will generate a whole value when used with a number with fraction part (ex: 122.12345).
    (*a*) True (*b*) False

11.3 Summary

- A Web Service is programmable application logic accessible via standard Web protocols.
- Web Services are simple and easy to understand.
- A Web Service in .NET consists of an .asmx page that either contains a class that provides the Web Service functionality or references a specific external class that handles the logic in an external class file.
- Web Service is the web processing directive that specifies a class name; here service for a web service and the language used to create the web service is visual basic.
- Web application allowing website visitors to submit and retrieve data to/from a database over the Internet.

Notes

11.4 Keywords

Cascading Style Sheets (CSS): It is a style sheet language used for describing the presentation semantics (the look and formatting) of a document written in a markup language.

Customer Relationship Management (CRM): It is a widely implemented model for managing an interaction of a company with customers, clients, and sales prospects.

Dreamweaver: It is the perfect tool for web designers, coders, and application developers at all levels.

Enterprise Resource Planning (ERP): systems integrate internal and external management information across an entire organization, embracing finance/accounting, manufacturing, sales and service, customer relationship management, and so on.

Perl: It is a high-level, general-purpose, interpreted, dynamic programming language.

Lab Exercise

1. Create a flow chart to prepare a Web services.
2. Prepare a list of companies which provide online Web services.

11.5 Review Questions

1. Define web services in terms of visual studio .NET?
2. Define the uses of SOAP and Xml terminologies.
3. Write the main operations of calling the web services.
4. Define Get and Post operation with their advantages.
5. Create a web service using Asp.NET
6. What are the important features of VB?
7. What is the Advantage of VB?
8. Define the web application with web services.
9. Explain types of web application?
10. Define the technologies used to build web application.

Answers for Self Assessment Questions

1. (*b*)	2. (*a*)	3. (*c*)	4. (*b*)	5. (*d*)
6. (*c*)	7. (*c*)	8. (*b*)	9. (*a*)	10. (*e*)
11. (*b*)	12. (*a*)	13. (*b*)	14. (*b*)	15. (*a*)

11.6 Further Readings

Books *Practical ASP. NET 3.5 Projects for Beginners*, by B.M. Harwani
ASP.NET, by Yashwanth Kanethkar

Online link http://www.w3schools.com/aspnet/default.asp

Unit 12: Security and Membership

Notes

CONTENTS

Objectives

After studying this unit, you will be able to:

- Understand the IIS security
- Discuss the ASP.NET authentication

Introduction

Forms authentication uses a substantiation ticket that is created when a client logs on to a location, and then it tracks the user throughout the site. The forms authentication ticket is usually contained inside a cookie. However, ASP.NET version 3.5 supports cookie less forms authentication, which results in the ticket being passed in a query string.

If the user requests a page that requires authenticated access and that user has not previously logged on to the site, then the user is redirected to a configured logon page. The logon page prompts the user to supply credentials, typically a user name and password. These credentials are then passed to the server and validated against a user store, such as a SQL Server database. In ASP.NET3.5, user-store access can be handled by a membership provider. After the user's credentials are authenticated, the user is redirected to the originally requested page.

Forms authentication processing is handled by the **Forms Authentication Module** class, which is an HTTP module that participates in the regular ASP.NET page-processing cycle. This document explains how forms authentication works in ASP.NET3.5.

This How To shows how to use the membership feature in ASP.NET version 3.5 applications. It shows us how to use two different membership providers: the **Active Directory Membership Provider** and the **SqlMembership Provider**. The membership feature greatly reduces the amount of code we have to write to authenticate users at Web site. The **ActiveDirectory Membership** provider uses Microsoft Active Directory directory service to maintain user information, while the **SqlMembership Provider** stores user details in a SQL Server database.

Notes

The ASP.NET version 3.5 membership feature provides secure credential storage for application users. It also provides a membership API that simplifies the task of validating user credentials when used with forms authentication. Membership providers abstract the underlying store used to maintain user credentials. ASP.NET 3.5 includes the following providers:

- ActiveDirectory Membership Provider. This uses either an Active Directory or Active Directory Application Mode (ADAM) user store.
- SqlMembership Provider. This uses a SQL Server user store.

With the pluggable membership architecture, we can also add support for own user stores. For example, we can add support for other Lightweight Directory Access Protocol (LDAP) directories or other existing corporate identity stores. To do so, we create a custom provider that uses the Membership Provider abstract class.

In most cases, the user store contains user credentials such as user names and passwords, and in some cases, personalization information. Avoid mixing personalization with authentication. If we only need to identify users for personalization reasons, a simple user name inside a cookie is sufficient. However, if we want to restrict and control access to different areas and functions of Web site and if we need to audit operations attributed to different users, then we must use authenticated access and forms authenticat.

We use the **SQLMembership Provider** with forms authentication if user information is stored in SQL Server. In most cases, this occurs when we have an intranet and user information is application-specific or when the application is Internet facing and the users do not have Active Directory accounts.

When we install ASP.NET, the Machine.config file for our server includes configuration elements that specify SQL Server membership providers. By default, the SQL provider is configured to connect to the local instance of SQL Server.

12.1 ASP.NET IIS Security

Most Web sites need to selectively restrict access to some portions of the site. We can think of a Web site as somewhat equivalent to an art colonnade. The gallery is open for the public to come in and browse, but there are confident parts of the ability, such as the business offices, that are accessible only to people with certain credentials, such as employees. When a Web site stores its customers' credit card information in a database, for example, ASP.NET helps defend the database from public access. ASP.NET security addresses this and many other security issues.

ASP.NET, in conjunction with Internet Information Services (IIS), can authenticate user credentials such as names and passwords using any of the following authentication methods:

- Windows: Basic, digest, or Integrated Windows Authentication (NTLM or Kerberos).
- Microsoft Passport authentication
- Forms authentication
- Client Certificate authentication.

ASP.NET helps control access to site information by comparing authenticated credentials, or representations of them, to NTFS file system permissions or to an XML file that lists authorized users, authorized roles (groups), or authorized HTTP verbs.

12.1.1 Provides an of ASP.NET Security

Websites against unauthorized right of entry are a critical, complex issue for Web developers. A successful system requires careful planning, and Web site administrators and programmers must have a clear understanding of the options for securing their site.

Notes

ASP.NET works in concert with the Microsoft .NET Framework and Internet Information Services (IIS) to help provide Web application protection. To help protect an ASP.NET application, you must perform the two fundamental functions described in the Table 12.1.

Table 12.1: Security Function

Security function	Description
Authentication in ASP.NET	Assures that the user is, in fact, who the user claims to be. The application obtains credentials (various forms of identification, such as name and password) from a user and validates those credentials against some authority. If the credentials are valid, the entity that submitted the credentials is considered an authenticated identity.
Authorization in ASP.NET	Limits access rights by granting or denying specific permissions to an authenticated identity.

IIS can also grant or deny access based on a user's host name or IP address. Any further access authorization is performed by NTFS file access permission's URL authorization.

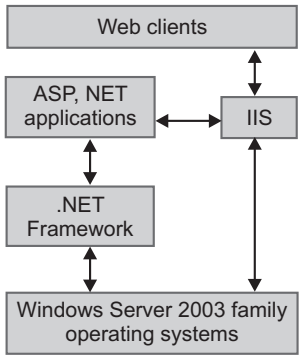
It is helpful to understand how all the various security subsystems interact. Since ASP.NET is built on the Microsoft .NET Framework, the ASP.NET application developer also has access to all the built-in security features of the .NET Framework, such as code access security and role-based user-access security. For details about the security capabilities of ASP.NET, see ASP. NET code access security.

Task Write the steps to activate IIS server.

12.1.2 ASP.NET Infrastructure and Subsystem Relationships, as Related to Security

This is provides an overview of the ASP.NET communications and subsystem relationships, as they relate to the subject of security. The following Figure 12.1 shows the relationships among the security systems in ASP.NET.

Figure 12.1: Security Systems in ASP.NET



As the Figure 12.1 shows, all Web clients communicate with ASP.NET applications through Internet Information Services (IIS). IIS deciphers and optionally authenticates the request. If Allow Anonymous is set to true, no authentication occurs. IIS also finds the requested resource (such as an ASP.NET application), and, if the client is authorized, returns the appropriate resource.`

Notes

In addition to the built-in ASP.NET features, an ASP.NET application can use the low-level security features of the .NET Framework. For more information, see the "Key Security Concepts" topic in .NET Framework Help.

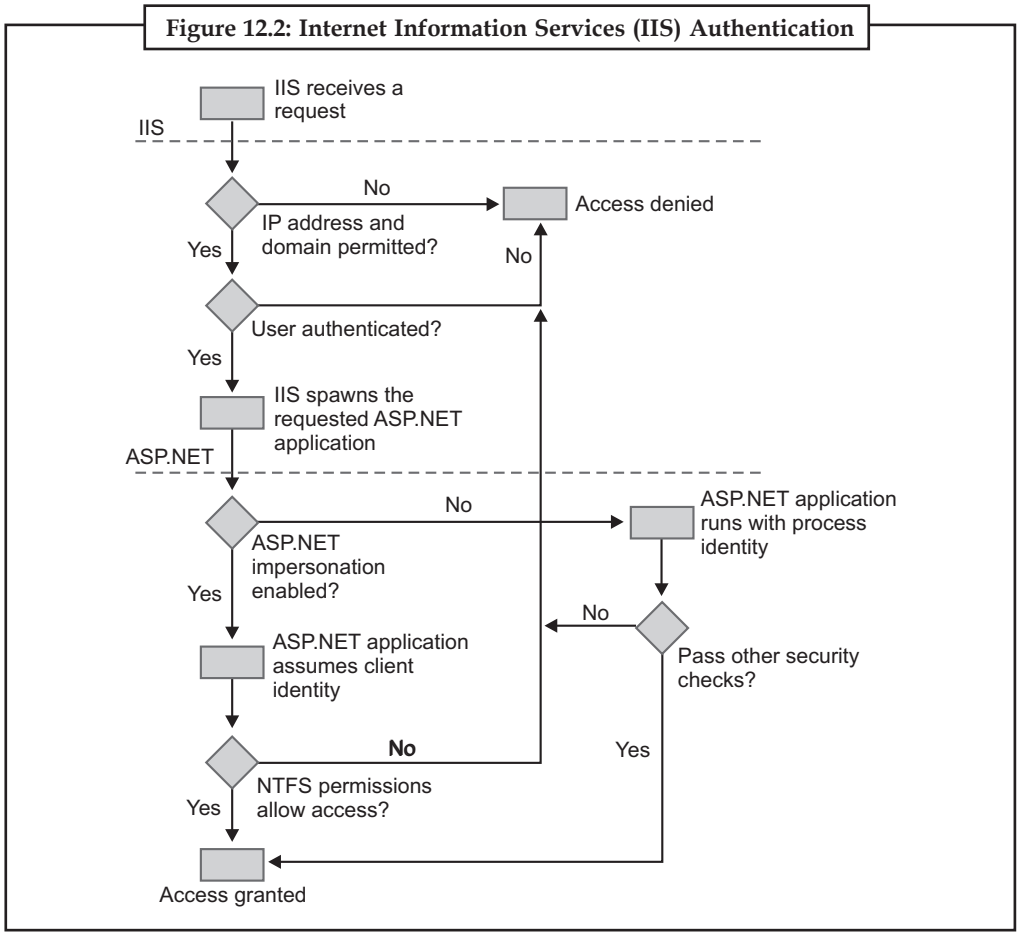
The Security Data Flow

There are a number of different ways to design security into ASP.NET applications. This is describes the data flow for two common scenarios: impersonation and forms authentication using cookies.

Scenario 1: Impersonation

This scenario relies on Internet Information Services (IIS) authentication and Microsoft Windows NT file access security to minimize security programming in the ASP.NET application itself. The data flow is shown in the following illustration.

Figure 12.2: Internet Information Services (IIS) Authentication



The Figure 12.2 shows the following sequence of events:

1. A request for access comes to IIS from a network client.
2. IIS authenticates the client using basic, digest, or Integrated Windows Authentication (NTLM or Kerberos).
3. If the client is authenticated, IIS hands the authenticated request over to ASP.NET.
4. The ASP.NET application impersonates the requesting client using the access token passed from IIS, and relies on NTFS file permissions for granting access. The ASP.NET application

Notes

needs only to verify that in the ASP.NET configuration file, the impersonation-enable directive is set to true; no ASP.NET security code needs to be written.

Notice that if impersonation is not enabled, the application runs with the IIS process identity. For Microsoft Windows 2000 Server and Windows XP, the default identity is a User account named ASP.NET that is created automatically when ASP.NET is installed. For products in the Microsoft Windows Server 2003 family, the default identity is the Network Service account. If we want to restrict access, we must use some other means of authorization, such as URL authorization.

For more details about using impersonation in ASP.NET applications, see Impersonation in ASP.NET and Using IIS Authentication with ASP.NET Impersonation.

5. If access is granted, the ASP.NET application returns the requested page through IIS.

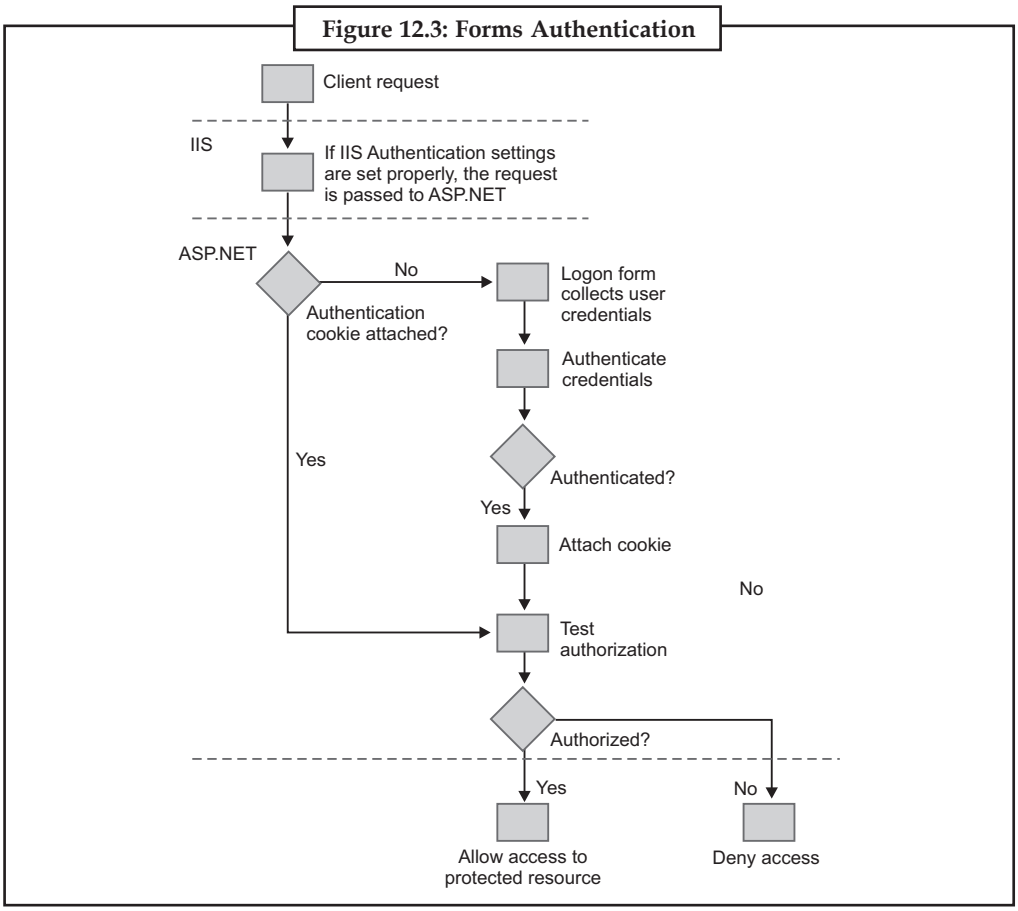
Scenario 2 - Forms Authentication

In this scenario an application uses ASP.NET forms authentication, a process that enables the application to collect credentials such as name and password directly from the client requestor and make its own determination about their authenticity. IIS authentication is not used by the application, but IIS authentication settings are important to the ASP.NET forms authentication process. Unless we decide to reject all requests that do not meet the criteria for the enabled method of IIS authentication, we must enable the IIS Anonymous Access setting.

If we do not enable anonymous access in IIS, requests not meeting the criteria for IIS authentication will be rejected and never reach the ASP.NET application.

The data flow in this scenario is shown in the following Figure 12.3.

Figure 12.3: Forms Authentication



Notes

This Figure 12.3 shows the following sequence of events:

1. A client generates a request for a protected resource.
2. IIS receives the request, and if the requestor is authenticated by IIS, or if IIS anonymous access is enabled, the request gets passed on to the ASP.NET application. Because the authentication mode in the ASP.NET application is set to forms in this case, IIS authentication is not used.
3. If there is no cookie attached to the request, ASP.NET redirects the request to a logon page, the path of which resides in the application's configuration file. On the logon page, the client user enters the required credentials (usually a name and password).
4. The application code checks the credentials to confirm their authenticity, usually in an event handler. If the credentials are authenticated, the application code attaches a ticket (as a cookie) containing the user name, but not the password. If authentication fails, the request is usually returned with an Access Denied message or the logon form is presented again.
5. After a ticket is issued by the application, ASP.NET just checks the ticket for validity using a message authentication check. Applications do not need the credentials in the .config files. In fact, ASP.NET does not check them after the cookie is issued, even if they are present.
6. If the user is authenticated, ASP.NET checks authorization and can either allow access to the originally requested, protected resource or redirect the request to some other page, depending on the design of the application. It can also direct the request to a custom authorization module where the credentials are tested for authorization to access the protected resource. If authorization fails, ASP.NET always redirects to the logon page.
7. If the user is authorized, access is granted to the protected resource; or the application might require an additional test of the credentials before authorizing access to the protected resource, depending on the design of the application.


Did u know?

The first Microsoft web server was a research project at the European Microsoft Windows NT Academic Centre (EMWAC), part of the University of Edinburgh in Scotland, and was distributed as freeware.


Task Create a flow chart of ASP.NET authentication.

Self Assessment Questions

1. …… uses a SQL Server user store.
   (*a*) ActiveDirectory Membership Provider
   (*b*) Directory Application Mode
   (*c*) SqlMembership Provider
   (*d*) None of these
2. A successful system requires careful planning, and web site ……must have a clear understanding of the options for securing their site.
   (*a*) administrators (*b*) programmers
   (*c*) administrators and programmers (*d*) None of these
3. …… are a number of different ways to design security into ASP.NET applications.
   (*a*) Internet Information Services (*b*) Security data flow
   (*c*) Data flow (*d*) Data control flow

True or False **Notes**

4. ASP.NET helps protect the database from public access.

   (*a*) True (*b*) False

5. IIS authentication is not used by the application, but IIS authentication settings are important to the ASP.NET forms authentication process.

   (*a*) True (*b*) False

12.2 ASP.NET Authentication

Authentication is the process of obtaining identification credentials such as name and password from a user and validating those credentials against some authority. If the credentials are valid, the entity that submitted the credentials is considered an authenticated identity. Once an identity has been authenticated, the authorization process determines whether that identity has access to a given resource.

Authentication is the process by which we obtain identification credentials such as the user's name and password and validate those credentials against some authority.

ASP.NET gives we more control to implement security for application. ASP.NET security works in conjunction with Microsoft Internet Information Services (IIS) security and includes authentication and authorization services to implement the ASP.NET security model. ASP.NET also includes a role-based security feature that we can implement for both Microsoft Windows and non-Windows user accounts.

ASP.NET implements authentication through authentication providers, the code modules that contain the code necessary to authenticate the requestor's credentials. ASP.NET supports the authentication providers described in the following Table 12.2.

Table 12.2: ASP.NET Authentication Provider

ASP.NET authentication provider	Description
Forms authentication	A system by which unauthenticated requests are redirected to an HTML form using HTTP client-side redirection. The user provides credentials and submits the form. If the application authenticates the request, the system issues a cookie that contains the credentials or a key for reacquiring the identity. Subsequent requests are issued with the cookie in the request headers; they are authenticated and authorized by an ASP. NET event handler using whatever validation method the application developer specifies.
Passport authentication	Centralized authentication service provided by Microsoft that offers a single logon and core profile services for member sites.
Windows authentication	ASP.NET uses Windows authentication in conjunction with Microsoft Internet Information Services (IIS) authentication. Authentication is performed by IIS in one of three ways: Basic, digest, or Integrated Windows Authentication. When IIS authentication is complete, ASP.NET uses the authenticated identity to authorize access.

To enable an authentication provider for an ASP.NET application, we only need to create an entry for the application configuration file as follows:

```
// Web.config file
<authentication mode= "[Windows|Forms|Passport|None]"/>
```

Notes

The mode is set to one of the authentication modes: Windows, Forms, Passport, or None. The default is Windows. If the mode is None, ASP.NET does not apply any additional authentication to the request - this can be useful when we want to implement a custom authentication scheme, or if we are solely using anonymous authentication and want the highest possible level of performance.

The authentication mode cannot be set at a level below the application root directory. As is the case with other ASP.NET modules, subdirectories in the URL space inherit authentication modules unless explicitly overridden.

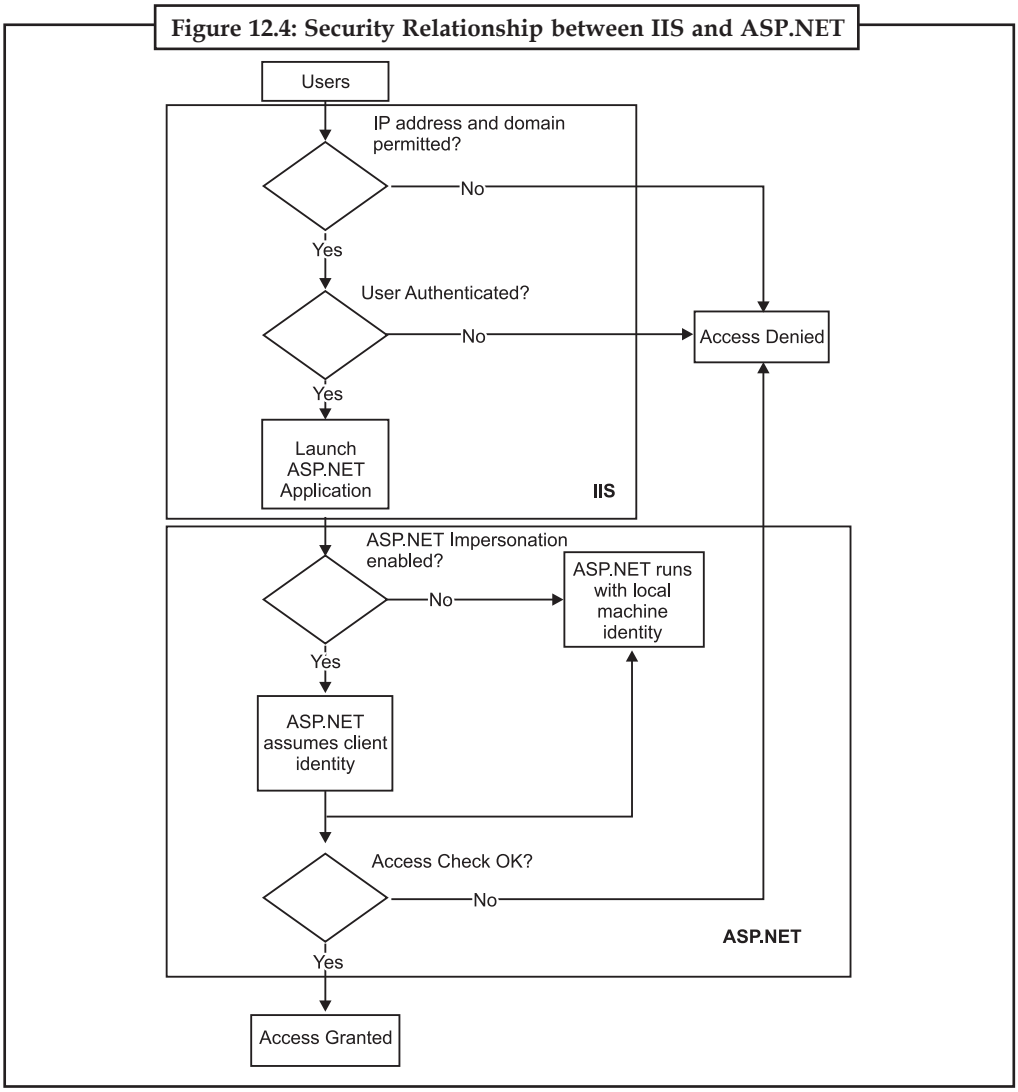

Did u know?

Web Site Administration tool was first introduced with ASP.NET 2.0 along with ASP.NET Microsoft Management Console (MMC) Snap-in.

12.2.1 Security Relationship between IIS and ASP.NET

IIS maintains security-related configuration settings in the IIS metabase. However, ASP.NET maintains security (and other) configuration settings in XML configuration files. While this generally simplifies the deployment of your application from a security standpoint, the security model adopted by your application will necessitate the correct configuration of both the IIS metabase and your ASP.NET application via its configuration file (Web.config). (See Figure 12.4)

Figure 12.4: Security Relationship between IIS and ASP.NET



Notes

When the user requests a specific resource on the system, that request will come to IIS. IIS authenticates the user requesting the resource and then hands off the request and the security token for the authenticating user to ASP.NET worker process.

ASP.NET worker process will decide whether to impersonate the authenticated user supplied by IIS or not. If impersonation is enabled in the configuration setting in Web.config file, then ASP.NET worker process impersonates the authenticated user. Otherwise, the thread will run under the ASP.NET worker process identity.

ASP.NET checks whether the authenticated user is authorized to access these resources. If they are allowed, ASP.NET serves the request; otherwise it sends an "access-denied" error message back to the user. ASP.NET application security configuration and IIS security configuration are completely independent and can be used independently or in conjunction with each other and also IIS maintains security related configuration settings in the IIS metabase. However, ASP. NET maintains security (and other) configuration settings in XML configuration files.

Factors in choosing an authentication method:

- Server and client operating systems
- Client browser type
- Number of users, location and type of user name and password database
- Deployment considerations (Internet vs. intranet and firewalls)
- Application type (interactive Web site or non-interactive Web service)
- Sensitivity of data being protected
- Performance and scalability factors.

Windows authentication for ASP.NET application, need to configure authentication within IIS. This is because IIS provides Windows authentication.

IIS provides four different authentication methods:

1. Anonymous
2. Basic
3. Digest and
4. Windows integrated

    - IIS does not perform any authentication for the anonymous authentication ie it allows anyone to access the ASP.NET application.
    - In the basic authentication, users must provide a windows user name and password to connect. However this information is sent over the network in clear text, which makes basic authentication very much insecure over the internet.
    - In the digest authentication, users must still provide a windows user name and password to connect. However the password is hashed before it is sent across the network.
    - Digest authentication requires that all users be running Internet Explorer5 or later and that windows accounts to stored in active directory.

Windows Integrated Authentication

In Windows authentication, IIS performs the authentication, and the authenticated token is forwarded to the ASP.NET worker process. The advantage of using Windows authentication is

Notes

that it requires minimal coding. We may want to use Windows authentication to impersonate the Windows user account that IIS authenticates before we hand off the request to ASP.NET.

The WindowsAuthenticationModule Provider Following:

- In windows integrated authentication, passwords never cross the network.
- Users must still have a username and password, but the application uses either the Kerberos or challenge/response protocols authenticate the user.
- Windows-integrated authentication requires that all users be running internet explorer 3.01 or later Kerberos is a network authentication protocol.
- It is designed to provide strong authentication for client/server applications by using secret-key cryptography.
- Kerberos is a solution to network security problems. It provides the tools of authentication and strong cryptography over the network to help to secure information in systems across entire enterprise.

Passport Authentication

Passport authentication is a centralized authentication service, which Microsoft provides, that offers a single log on and core profile services for member sites. Typically, Passport authentication is used when we need single log on capability across multiple domains.

The Passport Authentication Provider or following:

- Passport authentication makes use of Microsoft's passport service to authenticate users of application.
- If users have signed up with passport and we configure the authentication mode of the application to the passport authentication, all authentication duties are off-loaded to the passport servers.
- Passport uses an encrypted cookie mechanism to indicate authenticated users. If users have already signed into passport when they visit our site, they will be considered authenticated by ASP.NET. Otherwise they will be redirected to the passport servers to log in. When they are successfully log in, they will be redirected back to your site.
- To use passport authentication we have to download the Passport Software Development Kit (SDK) and install it on our server. The SDK can be found at.

Default Authentication

Default authentication is used when we do not want any security on Web application; anonymous access is required for this security provider. Among all authentication providers, Default authentication provides maximum performance for your application. This authentication provider is also used when we use your own custom security module.

Forms Authentication

Forms authentication refers to a system in which unauthenticated requests are redirected to a Hypertext Markup Language (HTML) form in which users type their credentials. After the user provides credentials and submits the form, the application authenticates the request, and the system issues an authorization ticket in the form of a cookie. This cookie contains the credentials or a key to reacquire the identity. Subsequent requests from the browser automatically include the cookie.

The Forms Authentication Provider Following:

Notes

- Forms authentication provides us with a way to handle authentication using our own custom logic with in an ASP.NET application.

The following applies if we choose forms authentication:

- When a user requests a page for the application, ASP.NET checks for the presence of a special session cookie. If the cookie is present, ASP.NET assumes the user is authenticated and processes the request.
- If the cookie is not present, ASP.NET redirects the user to a web form we provider and we may choose any authentication method to check the form.
- When the user is authenticated, we indicate this to ASP.NET by setting a property, which creates the special cookie to handle subsequent requests.

Forms Authentication Configuration

The default attribute values for forms authentication are shown in the following configuration-file fragment.

```
<system.web>
<authentication mode= “Forms”>
<forms loginUrl= “Login.aspx”
    protection= “All”
    timeout= “30”
    name= “.ASPXAUTH”
    path= “/”
    requireSSL= “false”
    slidingExpiration= “true”
    defaultUrl= “default.aspx”
    cookieless= “UseDeviceProfile”
    enableCrossAppRedirects= “false” />
    </authentication>
</system.web>
```

The default attribute values are described below:

- **loginUrl** points to application’s custom logon page. We should place the logon page in a folder that requires Secure Sockets Layer (SSL). This helps ensure the integrity of the credentials when they are passed from the browser to the Web server.
- **Protection** is set to **All** to specify privacy and integrity for the forms authentication ticket. This causes the authentication ticket to be encrypted using the algorithm specified on the **machine Key** element, and to be signed using the hashing algorithm that is also specified on the **machine Key** element.
- **Timeout** is used to specify a limited lifetime for the forms authentication session. The default value is 30 minutes. If a persistent forms authentication cookie is issued, the **timeout** attribute is also used to set the lifetime of the persistent cookie.
- **Name** and **path** are set to the values defined in the application’s configuration file.
- **requireSSL** is set to **false**. This configuration means that authentication cookies can be transmitted over channels that are not SSL-encrypted. If we are concerned about session hijacking, we should consider setting **requireSSL** to **true**.

Notes

- **slidingExpiration** is set to **true** to enforce a sliding session lifetime. This means that the session timeout is periodically reset as long as a user stays active on the site.
- **defaultUrl** is set to the Default.aspx page for the application.
- **Cookieless** is set to **UseDeviceProfile** to specify that the application use cookies for all browsers that support cookies. If a browser that does not support cookies accesses the site, then forms authentication packages the authentication ticket on the URL.
- **Enablecrossappredirects** is set to **false** to indicate that forms authentication does not support automatic processing of tickets that are passed between applications on the query string or as part of a form POST.


Caution

No Authentication mode is not secure. If you enable No Authentication mode, debugging leaves your computer vulnerable to any user on the network. A hostile user can connect to your computer, launch applications on your computer, access data on your computer, and perform other mischievous or destructive actions by using a debugger.


Case Study

Advanced Payment Solutions (APS)

In September 2005, London-based start up Advanced Payment Solutions (APS) launched the first chip and pinenabled prepaid payment card into the United Kingdom. Its first product, the cashplus prepaid MasterCard, is a reloadable, full utility, highly secure MasterCard that is accepted at over 24 million merchants worldwide. cashplus allows cardholders to perform ATM cash withdrawals, point of sale payments and online and telephone purchase transactions.

Though common in the US, open-loop prepaid payment cards such as cashplus are not yet widespread in the UK, despite the fact that about half of the adult population in the UK does not have a credit card and over two million people do not have a bank account. APS' products will provide the "unbanked" (people without bank accounts) and the "underbanked" (people with only basic bank accounts) with access to a secure cash storage and payment facility previously unavailable to them.

Project Drivers

APS required a highly secure system to issue, reload and redeem cards, which could be accessed by customer service representatives, consumers, and merchants. A highly secure infrastructure was a "must-have" for APS, since its system will be used to capture identity details, process card sales, make payments, reload funds and check account balances.

APS originally considered using a one-time password token-based authentication system to verify the online identity of users with varying roles, rights and privileges, such as administrators and merchants. However, the cost of this system was prohibitive and limited the number of people to whom the tokens could be issued. In order to more cost effectively match authentication strength to the risk level across these different groups, APS partnered with TriCipher, which offers a wide variety of authentication options and allowed APS to centrally manage different authentication strengths across its diverse user base.

Scope

The project enables APS to cover the whole of Europe. It is using the TACS system to provide strong authentication for multiple retail users issuing and redeeming cards, consumers and internal customer service reps. Retail outlets access APS software securely over the Internet

Contd...

Notes

to liven an instant use ATM card at the point of sale. A personally embossed MasterCard is then mailed to the customer. Once activated and loaded, cardholders can use their cashplus MasterCard to make purchases anywhere on the global MasterCard merchant network. The APS chip and pin technology tracks customer behavior and APS software tracks how much money the consumer is spending, retailer commissions by channel and outlet, and reconciles all transactions across the system.

APS is using the TriCipher Armored Credential System (TACS) to provide Device 2 factor authentication for retailers and staff. This provides strong, affordable security, authenticating both the user and their location without requiring provisioning of tokens or other hardware devices because the retailers can use their existing PCs as the second factor. Consumers use TriCipher's Armored Passwords, which enable them to use simple passwords while protecting against password theft. The TACS patented architecture makes it extremely difficult to steal a user's credentials, providing critical protection against phishing attacks.

Why APS Chose the TriCipher Armored Credential System (TACS) APS partnered with TriCipher because TACS was the only system to meet all of APS' business and IT security requirements:

- Authentication integrated with a customizable, secure data storage vault.
- The ability to match authentication strength to business risk for different users.
- The ability to manage multiple levels of authentication from a single infrastructure.
- The ability to use the PC as an affordable 2nd factor, avoiding the cost of managing and deploying separate hardware tokens.
- Easily manageable user roaming capabilities.
- Reliable high availability architecture.
- Certified by appropriate security authorities – TACS is both Identrus and FIPS certified.
- Scalable enough to grow with the business and extensible enough to integrate with new security technologies.

Deployment

Deployment of the TACS authentication infrastructure for the APS system is complete, and is currently serving multiple retail users issuing and redeeming cards, consumers and internal customer service reps. APS has launched and is actively selling its cashplus prepaid MasterCard, the first of its kind in the United Kingdom.

Questions

1. Explain the advanced payment solutions.

2. Describes the TriCipher Armored Credential System (TACS).

Self Assessment Questions

6. Authentication is the process of obtaining identification credentials such as …… from a user and validating.

   (*a*) name and password (*b*) name

   (*c*) password (*d*) None of these.

Notes

7. **Name** and **path** are set to the values defined in the application's Authorization Configuration file.

   (*a*) Name (*b*) Path

   (*c*) Name and path (*d*) None of these.

8. IIS can also grant or deny access based on a user's …….

   (*a*) . host name or ip address (*b*) host name

   (*c*) . ip address (*d*) None of these.

9. Timeout is used to specify a limited lifetime for the forms authentication session. The default value is ……..

   (*a*) 60 minutes (*b*) 90 minutes

   (*c*) 30 minutes (*d*) None of these.

10. It is designed to provide strong authentication for client/server applications by using……...

    (*a*) secret-key cryptography. (*b*) secret-key encryptography.

    (*c*) cryptography. (*d*) encryptography.

True or False

11. IIS deciphers and optionally authenticates the request. It Allow Anonymous is set to False, no authentication occurs.

    (*a*) True (*b*) False

12. The processes communicate through events.

    (*a*) True (*b*) False

13. Cookies are small text files that the server and browser exchange on each page request.

    (*a*) True (*b*) False

14. Cookies never changed in Webpages.

    (*a*) True (*b*) False

15. Passport authentication is a centralized authentication service, which Microsoft provides, that offers a single log on and core profile services for member sites.

    (*a*) True (*b*) False

12.3 Summary

- The user store contains user credentials such as user names and passwords, and in some cases, personalization information.
- Authentication is the process of obtaining identification credentials such as name and password from a user and validating those credentials against some authority.
- A system by which unauthenticated requests are redirected to an HTML form using HTTP client-side redirection. The user provides credentials and submits the form.
- Forms authentication provides us with a way to handle authentication using our own custom logic with in an ASP.NET application.

Notes

- The ASP.NET helps control access to site information by comparing authenticated credentials, or representations of them, to NTFS file system permissions or to an XML file that lists authorized users, authorized roles (groups), or authorized HTTP verbs.
- When we install ASP.NET, the Machine.config file for server includes configuration elements that specify SQL Server membership providers.
- The membership feature greatly reduces the amount of code we have to write to authenticate users at Web site.
- The logon page prompts the user to supply credentials, typically a user name and password.
- Avoid mixing personalization with authentication. If we only need to identify users for personalization reasons, a simple user name inside a cookie is sufficient.
- ASP.NET implements authentication through authentication providers, the code modules that contain the code necessary to authenticate the requestor's credentials.

12.4 Keywords

Access-denied: If a request to ASP.NET application returns the error, "Denied Access to Directory Name directory. Failed to start monitoring directory changes.

Cookie: The topics in this topic describe how to create cookies in ASP.NET Web applications. Cookies are small text files that the server and browser exchange on each page request, and that we can use to store information that can help we customize your application for each user.

Cryptography: Means and methods for the transformation of data in order to hide its information content, prevent its undetected modification, or prevent its unauthorized use.

Event Handle: Event is an action or occurrence like mouse click, key press, mouse movements, or any system generated notification. The processes communicate through events.

Internet Information Services (IIS): IIS (Internet Information Server) is a group of Internet servers (including a Web or Hypertext Transfer Protocol server and a File Transfer Protocol server) with additional capabilities for Microsoft's Windows NT and Windows 2000 Server operating systems.

Lightweight: In information technology, the term lightweight is sometimes applied to a program, protocol, device, or anything that is relatively simpler or faster or that has fewer parts than something else.

Lab Exercise

1. Write the steps to authenticate a web form.
2. Search about the authentication modes.

12.5 Review Questions

1. What is the ASP.NET IIS Security?
2. Simply Provides a ASP.NET security.
3. Provides an ASP.NET infrastructure and subsystem relationships, as related to security.
4. What is the ASP.NET Authentication?
5. Give the Security Relationship Between IIS and ASP.NET.
6. Discuss the Factors in Choosing an Authentication Method.
7. Explain Authentication types by ASP.NET.

Notes

8. Discuss the Passport authentication.
9. Example of the Forms Authentication Configuration
10. Describes the security data flow for two common scenarios.

Answers for Self Assessment Questions

1. (*c*)	2. (*c*)	3. (*b*)	4. (*a*)	5. (*a*)
6. (*a*)	7. (*c*)	8. (*a*)	9. (*c*)	10. (*a*)
11. (*b*)	12. (*a*)	13. (*a*)	14. (*b*)	15. (*a*)

12.6 Further Readings


Books *Professional ASP.Net 3.5*, by Bill Evjen


Online link http://www.informit.com/store/product.aspx?isbn=0321159659

Unit 13: Adding E-Commerce Essentials

Notes

CONTENTS

Objectives

After studying this unit, you will be able to:

- Discuss the XML tools
- Define freight calculations
- Discuss the e-mail

Introduction

Contribution business information, maintain business associations and conduct business communication using computers associated to a telecommunication network is called E-Commerce.

There are mainly 7 Essentials for Supporting E-Commerce:

Inventory Visibility and Synchronization

Accurate inventory levels reflected in your customer's E-Commerce shopping cart software is central for the highest level of customer contentment and for an aerodynamic order completion process. A 3PL storehouse administration system which has the aptitude to update your clientele shopping cart software with current inventory levels will significantly differentiate you from other 3PL competitors.

Besides E-Commerce inventory is commonly depleted for shipments from a variety of sources. There are many other ways shipments might be initiated in addition to shopping cart orders, those include: Salesperson samples, Retail store orders, and Wholesale/distribution channel orders

Notes

to name a few. Since these other order sources could quickly exhaust stock, it is important to have an automated and regularly scheduled inventory update from the WMS to shopping cart.

Typically inventory levels are updated once a day or once per week depending on the popularity of the products. This ensures that items which are out of stock are not be displayed (or they will be listed as out of stock) to consumers visiting your customer's online store. By properly reflecting out of stock status, you can better manage backorder issues and customer expectations.

Real-time Communication of Order Details Between Shopping Cart Software and WMS

The continued growth of online retail sales have been a boom for fulfilment houses year round. From mega-sites like Amazon.com and e-Bay to hosting companies that service one person operations, online retailers have streamlined the buying process by implementing virtual "shopping carts " with a convenient "checkout " button to pay for and complete a transaction. Shopping cart software saves the buyer and seller time and expense by automating the buying process. Your warehouse can also take advantage of this automation by tapping into these shopping carts for order processing and status updates.

Shopping cart software and warehouses primarily communicate through three methods:

1. Email Alerts
2. Batch Processing by Manual Import/Export
3. Automated Integrations (APIs and Web Service Calls) between systems.

The first and most basic method is through e-mail alerts sent to the fulfilment house. This one-way notification is fine for drop shipments and partners who do not ship regularly but is not a guaranteed message delivery method.

The second method is integration through batch processing which allows the fulfilment house to process multiple transactions by manually importing and exporting data files between systems. Batch processing requires user interaction to import and export flat files to move data between systems. This method is relatively inexpensive and is familiar to most experienced computer users.

Flat file imports and exports can also be implemented very quickly. Unfortunately, batch processing creates delays in the information pipeline which is inefficient be definition.

Handling Larger Order Volumes

Operational inefficiencies can be magnified as order volumes increase. This is especially true when seasonal workers are brought in that may not be as familiar with the warehouse or inventory setup as your regular staff. A solid Standard Operating Procedure (SOP) for pulling orders is a great place to start, but enforcement can be difficult in times of high activity. Right when you need it most! As your order volumes increase, let your employees focus on accurate fulfilment while your WMS optimizes your workflow and enforces your picking rules.

Limiting movement and repetition in the warehouse is key to improving productivity. Pick line locations and pick location priority settings can be used to direct your staff to the most logical items to pick based on shortest distance, least touches, package configuration or a number of other criteria. When set up correctly, your WMS can automatically allocate product from these optimized locations. Mobile scanning features can enforce the pick another way to limit movement and touches in the warehouse are to utilize batch or wave picking methods. Wave picking works by reviewing a large batch of orders and "plotting" the best course through the warehouse to pick product from each location. This means that if all multiple orders require items from the same location, the operator will pick all items at the same time. Once all items have been picked, the operator will segregate the product into the correct orders during the packaging process. Batch picking eliminates redundant location picking and greatly reduces distance travelled in the warehouse.

Notes

Increased Pick Accuracy (Barcode Scanning)

Picking accuracy is an essential component of ensuring that the product ordered is the product delivered. One way to ensure accuracy is to leverage barcode scanning technology at the time of pick and pack. A WMS system that has barcode scanning functionality will both direct a warehouse picker to the correct inventory location as well as verify the picked product. It is critical to have the capability to immediately verify that the correct SKU was selected at the time the product is picked. This enables the picker to quickly move to the next line item to be picked or stop and resolve the reason why the item does not match the item specified by the WMS system. Having this real-time verification based on the Item Number or UPC barcode provides immediate feedback to the picker. Barcode scanning is one of the best ways to increase your shipment accuracy resulting in happier customers and consumers. Barcode scanning also helps to reduce customer service costs.

Custom Branded Packing Slips

One of the largest challenges in handling E-Commerce fulfillment as a 3PL warehouse is supporting the customized requirements each of your customers would like the consumer to have when they open the carton. It is understandable that each customer wants to not only deliver the products the consumer ordered quickly and accurately, but they also want to give the customer a feeling of being special. One way to create this brand experiences is to place a customized packing slip in the carton.

Placing a printed packing slip in the carton is easy to do - the challenge is printing different looking packing slips for each of your customers. One customer wants their logo included, while another wants the 3PL's address as the return address instead of the companies' own address. Having a WMS system that can support these types of customizations, by customer, will enable you to grow your E-Commerce fulfillment customer base. It is also common to have more than one custom packing slip per customer – this adds another layer of complexity. So, having a system that can not only customize packing slips per customer - but also have multiple different versions depending on the website where the order originates, will truly set you apart. One example is when fulfilling orders from Amazon. In Amazon's case, the packing slip will look different than the orders generated by your customer's own online store.

The largest benefit to you as a 3PL is that with each packing slip you customize, it becomes more difficult for a customer to switch to another fulfillment provider. Thus, custom branded packing slips help make your solution more relevant and valuable to your customers.

Integration with Shipping Systems

A great way to make your E-Commerce fulfillment more profitable is to integrate with small parcel carriers such as UPS World Ship, FedEx Ship Manager and Endicia and eliminate retyping and associated costs.

Order fulfillment is not complete until the packages are off your dock. With the peak season approaching, the demands on small parcel carriers and the United States Postal Service are in overdrive. This means drivers are less likely to wait while you create labels and generate manifests, and customers are less forgiving if their packages do not arrive on time. Time and accuracy are of the essence while pressure and workload are on the rise.

Automatic Email Notifications to Customers

In order to handle higher volumes of orders which often come from E-Commerce fulfillment, you must streamline your processes and reduce the need for additional customer service. With increased order volume, often comes increased customer support needs – the trick is to increase your volume without any additional back office resources. And this is possible with a robust reporting and email notification system as part of your 3PL WMS offering.

Notes

As a 3PL operator you know the availability of real-time, accurate information is as important as any core function. Customers demand immediate details, but they do not all expect it to be delivered identically. Some want it "pushed" to them via automatic e-mail updates, while others wish to "pull" it themselves via their own access to online reporting. Some want it both ways. A robust WMS enables the 3PL operator to provide flexible communication options to its varied customer base, and a great WMS does so while rolling that functionality into the day-to-day workflow. This streamlines this critical task operationally while eliminating missed customer updates.

13.1 The XML Tools

Extensible Markup Language (XML) is an easy, very flexible text layout derived from SGML (ISO 8879). Originally designed to meet the challenges of important electronic publishing, XML is also playing an increasingly important role in the exchange of a extensive variety of data on the Web and elsewhere.

This page describes the work organism done at W3C within the XML movement, and how it is structured. Work at W3C takes place in Working Groups. The Working Groups within the XML Activity are listed below, together with links to their individual web pages.

Each document also contains email addresses you can use to send comments or questions, for example if you have been writing software to implement them and have found problems or errors.

There are various tools of XML are given below:

Xerces

- Validating XML parser
- SAX support
- DOM support
- XML Schema support
- JAXP support

Xalan

- XSLT engine
- XPath support

JDOM

- Java interface to SAX or DOM based parsers

Other popular XML Processing Tools:

- Saxon – popular XSLT engine

Java XML Data Binding Tools

- Castor – open source, although lacks complete schema support
- JAXB – Java Community Process based, provides schema support

DTD Validation

The WFS Specification mandates the use of XML Schema as a minimum for all operations. Other output formats for Get Feature, such as DTD, are possible if specified by DescribeFeatureType and GetCapabilities.

DTD Validation Tools:

Notes

- XSDValid – contains dtdvalid DTD validation tool


Did u know?

The XSD 1.0 specification was originally published in 2001, with a second edition following in 2004 to correct large numbers of errors. XSD 1.1 became a W3C Recommendation in April 2012.


Caution

When specified, the generate XML command writes the resulting XML data to a file otherwise any existing file will be overwritten without warning.


Task Write an XML program to validating DTD.

13.2 Freight Calculations

A charge paid for wagon or transport of goods by air, land, or sea. Goods may be on cloud nine on freight-prepaid or freight-collect basis:

1. If the freight is paid by the consignor (as under C&F and CIF terms) the goods remain the consignor's property until their delivery is taken by the consignee upon their arrival at the destination, and payment of the consignor's invoice.
2. If freight is paid by the consignee (as under FOB terms) the goods become the consignee's property when handed over to the carrier against a bill of lading. It may be charged on the weight or volume of the shipment (depending upon its nature or density) and also varies according to the mode of shipment, such as bulk, break bulk, and containerized. It is also called freightage.

A freight forwarder, forwarder, or forwarding agent is a person or company that organizes shipments for individuals or corporations to get large orders from the manufacturer or producer to market or final point of distribution. Forwarders will contract with a carrier to facilitate the movement of goods. A forwarder is not typically a carrier, but is an expert in supply chain management. In other words, a freight forwarder is a "travel agent, " for the cargo industry, or a third-party (non-asset-based) logistics provider. A forwarder will contract with asset-based carriers to move cargo ranging from raw agricultural products to manufactured goods. Freight can be booked on a variety of carrier types, including ships, airplanes, trucks, and railroads. It is not unusual for a shipment to move along its route on multiple carrier types.

International freight forwarders typically arrange cargo movement to an international destination. International freight forwarders, have the expertise that allows them to prepare and process the documentation and perform related activities pertaining to international shipments. Some of the typical information reviewed by a freight forwarder is the commercial invoice, shipper's export declaration, bill of lading, and other documents required by the carrier or country of export, import, or transshipment. Much of this information is now processed in a paperless environment.

13.2.1 Air Freight Calculation

Airlines that are members of the worldwide Air Transport Association (IATA) are bound by their relationship to comply with tariff issued by IATA. Though since 11th September 2002, airfreight rates are now tremendously negotiable. Airfreight rates cover transportation from the airport of loading to the airport of discharge.

These rates do not include the following:

- Collection of air cargo from the consignor's/exporters premises

Notes

- Delivery of cargo from the airport of destination to the consignee's premises
- Storage of cargo before or after loading
- Customs clearance in the country of destination
- Any duties and taxes that may have to be paid
- Insurance

Chargeable/volumetric Weight

Airline freight rates are based on a "chargeable weight", because the volume or weight that can be loaded into an aircraft is limited. The chargeable weight of a shipment will be either the "actual gross mass " or the "volumetric weight", whichever is the highest. The chargeable weight is calculated as follows: 1 metric ton = 6 cubic metres. In order to establish if the cargo will be a weight or volumetric based shipment.

Step 1: Measure the parcel/cargo along the greatest length, width and height of that parcel For example; 100 cm (L) × 100 cm (W) × 100 cm (H) = 1 000 000 cm3. Next, weigh the parcel; assume it weighs 150kg.

Step 2: Now divide the 1 000 000 cm3 by 6,000 = 166,66 kg. You have now converted the centimeters (cm) into kilograms (kg).

Step 3: Now compare the weight to the volume. If the weight is 150 kg then the airline would base the freight on the higher amount being: 166, 66 kg.

Calculations

The airline calculates freight based on weight or volume, which ever yields the greatest amount. Airlines quote freight rates based on the following rate structures:

- A basic minimum charge per shipment.
- General cargo rates quoted for per kilogram. This rate applies without reference to the nature or description of the parcel, which is to be freighted.
- Specific commodity rates apply to certain goods of specific descriptions, such as fresh produce. These rates are lower than the general cargo rate, and they provide breakpoints at which the level of the rate reduces further.

Example: 0 - 50 Kg @ R22.00/per kg
50 - 100 Kg @ R19.00 per kg
100 - 150 Kg @ R17.00 per kg

Unit Load Device Charges

These rates are charged per container/ULD without reference to the commodity loaded therein. Calculation of freight rates:

Let us assume the following figures:

The freight rate is R18.00 per kg

The weight of the parcel is 300 kg

The dimensions are: 114,6 cm × 120,4cm × 132,5 cm (round the cm's up or down)

Therefore: 115 cm × 120 × 133 cm = 1 835 400 divide by 6 000 = 305.9 kg (having converted cm's to kg's now round up the kg's to the next half a kilogram = 306 kg.)

Notes

As the freight rate quoted by the airline is R18.00 per kg, we calculate the price as follows:

306 kg × R18/kg = R5 508.00

The freight rate will not be calculated on the actual mass 300 kg × R18.00 = R5 400.00 as the airline will always use the greater amount either the kg, or volumetric weight.

Consolidation

Consolidation is an economical method of moving cargo by employing a consolidator. The consolidator receives cargo from a number of suppliers/shippers and then combines these cargoes into one consignment by packing the goods into a Unit Load Device. The consolidator then books the Unit Load Device with an airline. The supplier/shipper would have a contract of carriage with the consolidator of the cargo and in turn the airline would have a contract of carriage with the consolidator. The airline would issue an air waybill to the consolidator when accepting the Unit Load Device and in turn the consolidator would issue the supplier/shipper with a house air waybill.

The Air Waybill

The air waybill, unlike the ocean bill of lading is not a document of title to the goods described therein, however it does perform several similar functions these are:

- It is a receipt for the goods
- It is evidence of the contract of carriage between the exporter and the carrier
- It incorporates full details of the consignor/shipper, the consignee/receiver and the consignment/goods
- It is an invoice showing the full freight amount
- It must be produced, be it in an electronic format, at the airport of discharge for clearing purposes.

All copies of the air waybill, together with the commercial invoice, packing list, certificate of origin and any other document which may be necessary for clearing the goods through customs, these documents are carried in the flight captain's bag.

13.2.2 Sea Freight Calculations

Seafreight calculations can broadly be divided into two main components; breakbulk and containerised. In this section we deal with how you should calculate the freight costs of both of these two types of seafreight.

Break Bulk Cargo Calculations

Break bulk cargo, is cargo that is unitised, palletised or strapped. This cargo is measured along the greatest length, width and height of the entire shipment. The cargo is also weighed. Shipping lines quote break bulk cargo per "freight ton", which is either 1 metric ton or 1 cubic metre, which ever yields the greatest revenue.

Example: A case has a gross mass of 2 Mt.
The dimensions of the cargo are:
2.5 × 1 × 2 metres

The tariff rate quoted by the shipping line is: INR 110.00 weight or measure (freight ton)

Step 1: Multiply the metres 2.5 × 1 × 2 = 5 metres Compare to the mass = 2 Mt.

Notes

Step 2: Calculate the freight with the greater amount either the mass or the dimension. 5 × INR 110.00 = INR 550.00

Freight would be paid on the measurement and not the weight. All shipping lines carrying cargo in a break-bulk form insist on payment based on a minimum freight charge which is equivalent to one freight ton, one cubic metre or one metric ton.

Full Container Load Calculations and Surcharges

Freight rates for containers are based on the container as a unit of freight irrespective of the commodity or commodities loaded therein, (FAK) Freight All Kinds. The shipping lines quote per box (container) either a six or twelve metre container. From time to time, abnormal or exceptional costs arise in respect of which no provision has been made in the tariffs. For example a shipping line cannot predict the movement of the US Dollar or the sudden increase of the international oil price. These increases have to be taken into account by the shipping line in order to ensure that the shipping line continues to operate at a profit. These increases are called surcharges. All shipping lines accordingly retain the right to impose an adjustment factor upon their rates taking into account these fluctuations. All surcharges are expressed as a percentage of the basic freight rate. Surcharges are regularly reviewed in the light of unforeseen circumstances, which may arise and bring cause for a surcharge increase.

Bunker Adjustment Factor (BAF)

"Bunkers" is the generic name given to fuels and lubricants that provide energy to power ships. The cost of bunker oil fluctuates continually and with comparatively little warning.

Example: Freight rate: Port Elizabeth to Singapore

Freight rate: INR: 1 250.00 per 6-M container

+ BAF 5.2%

INR 1 250.00 × 5.2% = INR 65.00

Add the two amounts together

Freight rate: INR 1 315.00

Currency Adjustment Factor (CAF)

The currency adjustment factor is a mechanism for taking into account fluctuations in exchange rates, these fluctuations occur when expenses are paid in one currency and monies earned in another by a shipping company. The currency adjustment factor is a mechanism for taking into account these exchange rate fluctuations. It is always expressed as a percentage of the basic freight and is subject to regular review.

Self Assessment Questions

1. ..... is commonly depleted for shipments from a variety of sources.
   (*a*) Warehouse management system
   (*b*) Telecommunication network
   (*c*) E-Commerce inventory
   (*d*) None of these
2. Shopping cart software and warehouses primarily communicate through methods:
   (*a*) Batch Processing by Manual Import/Export
   (*b*) Handling larger order volumes

Notes

(*c*) Increased pick accuracy

(*d*) Custom branded packing slips

3. Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML.

(*a*) SGML (*b*) HTML

(*c*) W3C (*d*) XML

4. Order fulfillment is not complete until the packages are off your dock.

(*a*) True (*b*) False

5. Airlines that are not members of the International Air Transport Association are bound by their membership to comply with tariffs issued by IATA.

(*a*) True (*b*) False

13.3 E-mail

E-mail (electronic mail) is the switch over of computer-stored mail by telecommunication. E-mail mail is usually programmed in ASCII transcript. However, you can too mail non-text files, such as graphic imagery and sound files, as attachment sent in binary streams. E-mail was one of the first uses of the Internet and is still the most popular use. A large fraction of the total traffic over the Internet is e-mail. E-mail can also be exchanged between online service provider users and in networks other than the Internet, both public and private. E-mail can be distributed to lists of people as well as to persons. A shared distribution list can be managed by using an e-mail reflector. Some mailing lists allow you to subscribe by sending a request to the mail list administrator. A mailing list that is administered automatically is called a list server. E-mail is one of the protocols included with the Transport Control Protocol/Internet Protocol (TCP/IP) suite of protocols. A popular protocol for sending e-mail is Simple Mail Transfer Protocol and a popular protocol for receiving it is POP3. Both Netscape and Microsoft include an e-mail utility with their Web browsers.

13.3.1 Types of E-mail

- Web-Based E-mail (Webmail)
- POP3 E-mail Services
- IMAP E-mail Servers
- MAPI E-mail Servers

13.3.2 How does E-mail Work?

Billions of electronic mail (e-mail) mail move across the Internet every year. distribution electronic letters, pictures and data files, either across a structure or across the sphere, has grown so popular that it has started to replace some postal mail and telephone calls. This worldwide medium is no longer restricted to exchange of simple text messages and is now regularly used to deliver voice mail, facsimiles and documents that may include images, sound and video.

Typically, a message becomes available to the recipient within seconds after it is sent—one reason why Internet mail has transformed the way that we are able to communicate.

Message Sender: Uses mail software, called a client, to compose a document, possibly including attachments such as tables, photographs or even a voice or video recording. System software, called Transmission Control Protocol (TCP), divides the message into packets and adds

Notes

information about how each packet should be handled-for instance, in what order packets were transmitted from the sender. Packets are sent to a mail submission server, a computer on the internal network of a company or an Internet service provider.

Internet Mail Addresses: Attached to each message are in the form "mailbox@domainname" – one specific example being "webmaster@seniorindian.com." The multipart domain name in the above example denotes a top-level domain (".com") following the second-level domain ("seniorindian"). A message is delivered to an individual or a group by the mailbox name ("webmaster").

Mail Submission Server: Converts the domain name of the recipient's mail address into a numeric Internet Protocol (IP) address. It does this by querying domain name servers interspersed throughout the Internet. For example, the mail submission server can first request from the "root" name server the where about of other servers that store information about ".com" domains. It can then interrogate the ".com" name server for the location of the specific "sciam.com" name server. A final request to the "sciam.com" name server provides the IP address for the computer that receives the mail for sciam.com, which is then attached to each message packet.

Routers: Dispersed throughout the Internet read the IP address on a packet and relay it toward its destination by the most efficient path. (Because of fluctuating traffic over data lines, trying to transmit a packet directly to its destination is not always the fastest way.) The packets of a single message may travel along different routes, shuttling through 10 or so routers before their journey's end.

13.3.3 Advantages of E-mail

- E-mails are easy to use. You can organize your daily **correspondence**, send and receive electronic messages and save them on computers.
- E-mails are fast. They are **delivered** at once around the world. No other form of written communication is as fast as an email.
- The language used in emails is simple and **informal**.
- When you **reply** to an email you can **attach** the original message so that when you answer the **recipient** knows what you are talking about. This is important if you get hundreds of emails a day.
- It is possible to send **automated** emails with a certain text. In such a way it is possible to tell the sender that you are on vacation. These emails are called auto responders.
- E-mails do not use paper. They are **environment** friendly and save a lot of trees from being cut down.
- E-mails can also have pictures in them. You can send birthday cards or newsletters as emails.
- Products can be **advertised** with emails. Companies can **reach** a lot of people and inform them in a short time.

13.3.4 Disadvantages of E-mails

- E-mails may carry viruses. These are small programs that harm your computer system. They can read out your e-mail address book and send themselves to a number of people around the world.
- Many people send unwanted emails to others. These are called spam mails. It takes a lot of time to filter out the unwanted emails from those that are really important.

Notes

- E-mails cannot really be used for official business documents. They may be lost and you cannot sign them.
- Your mailbox may get flooded with emails after a certain time so you have to empty it from time to time.


Did u know?

In 1971 the first ARPANET e-mail was sent, and through RFC 561, RFC 680, RFC 724 – and finally 1977's RFC 733, became a standardized working system.


Task Prepare a flow chart to sending process of an e-mail.


Case Study

Oxygen XML Case Study

Oxygen XML offers a complete cross platform XML editor providing the tools for XML authoring, conversion, Schema, DTD, Relax NG and Schematron development, XPath, XSLT, XQuery debugging, SOAP and WSDL testing. To get an idea about the features in the suite, one can download free trial licenses from the company's website. Intel and IBM are just some of the big names that have incorporated Oxygen XML's solutions for web services and service oriented architectures.

The Challenge

In early 2007, Oxygen XML was searching for a way to optimize its business, especially by increasing the online sales volumes products were already well-known worldwide and the company wanted to monetize that in an accessible and profitable way for both them and their clients. From having a customized solution to building a strong cash flow, the company continuously seeks for techniques to better run the business.

The main things to be considered were: Cutting costs in selling online, having a strong cash flow and working with an online services provider that would offer customized services to fit the software's particularities. Avangate came to meet all these requirements and more, becoming one of Oxygen XML's strategic partners.

Avangate was required to securely process online orders on the Oxygen XML website, as well as insure a strong cash flow for the company and a customized solution, all in a cost effective manner.

Avangate – A Real Partner

By working together to improve Oxygen XML's online solutions, the two companies built a strong partnership that will continue to produce great results in the future. Avangate's experience and know-how in selling software online proved to be decisive.

First of all, the personal approach that Avangate uses insures true and efficient client-provider communication, so that both parties involved get to know each other better and faster. This leads to faster and customized results, saving time and offering Oxygen XML the requested deliverables.

The client-oriented attitude of the Avangate team is the most important asset from the client's point of view. Every one of the customer's demands was thoroughly analyzed and a solution was found, that would fit both Oxygen XML's necessities and Avangate's development timeline.

Contd...

Notes

The Avangate Solution

The online payment platform was the main part of the solution proposed by Avangate. It contains all the necessary modules for Oxygen XML plus additional requested customizations that Avangate successfully implemented, especially for financial reporting.

Given the fact that the client needed a way to protect itself against fraud attempts without being too obtrusive to the customers and, possibly, losing well-intended clients, Avangate customized the antifraud filter by changing the way each request is treated.

Avangate also customized the support services for Oxygen XML by manually analyzing the lost customers and following-up to those that fit the company's profile, dependent on the order and other specific parameters.

Last but not least, Avangate also presented Oxygen XML a lower total cost of ownership (TCO) for the solution, in order to increase business efficiency. Also, by offering 4 payments a month, it gives Oxygen XML the opportunity to have a strong cash flow at hand, minimizing the chances for unpredictable situations.

Implementation

The whole implementation phase of the project was finished after only 3 months; during this time, Avangate developed and successfully tested the custom required features the client needed. Currently, over 80% of their total software sales are being processed online, turning Avangate into a key partner for Oxygen XML.

The custom-developed features are the most important aspect of the implementation phase. They were discussed together with the client and were solved with maximum priority in the shortest possible time. This way, Oxygen XML's time-to-market was dramatically improved.

Benefits

"Improved efficiency" is the expression to use when it comes to describe the benefits of the Avangate solution for Oxygen XML:

- worldwide reach to the company's products via an internationally recognized E-Commerce solution;
- more high quality customers;
- strong cash flow;
- business-efficient TCO.

Question

1. What are the challenges of Oxygen XML?
2. What are the solutions provided by Avantage solutions?

Self Assessment Questions

6. SKU stands for:

   (*a*) Store kool unit. (*b*) Store kare unit

   (*c*) Store keeping unit (*d*) None of these

7. …….. Tool is found for XML Schema validation:

   (*a*) WBS (*b*) SPY

   (*c*) SQC (*d*) None of these

Notes

8. Which is not an XML tool?
   (*a*) Xerces (*b*) Serax
   (*c*) SQC (*d*) Xalan
9. Which one is the Java XML Data Binding Tool?
   (*a*) Castor (*b*) JAXP
   (*c*) XSLT (*d*) Saxon
10. E-mail messages are usually encoded in …… text.
   (*a*) ASCII (*b*) unicode
   (*c*) alphabetical (*d*) binary
11. …… protocol is used for sending E-mail.
   (*a*) POP (*b*) SNMP
   (*c*) Telnet (*d*) SMTP
12. ……. protocol is used for receiving E-mail.
   (*a*) ICMP (*b*) POP3
   (*c*) FTP (*d*) RIP
13. …… relays most efficient path to the packet.
   (*a*) Bridge (*b*) Gateway
   (*c*) Switch (*d*) Router
14. Which is used to translating IP address to Domain name?
   (*a*) IP translator (*b*) DNS server
   (*c*) HTTPS (*d*) HTTP
15. …….. protocol divides the message into packets.
   (*a*) TCP (*b*) IP
   (*c*) NFS (*d*) UDP

13.4 Summary

- Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.
- The WFS Specification mandates the use of XML Schema as a minimum for all operations.
- E-mail (electronic mail) is the exchange of computer-stored messages by telecommunication. E-mail messages are usually encoded in ASCII text.
- XML spy is an XML development environment for designing and editing XML, XML Schema, XSL/XSLT, SOAP, WSDL and Web Service technologies.

13.5 Keywords

Post Office Protocol (POP): It is an application-layer Internet standard protocol used by local e-mail clients to retrieve e-mail from a remote server over a TCP/IP connection.

Notes

SKU: It refers to a Stock-Keeping Unit, a unique identifier for each distinct product and service that can be purchased in business.

Standard Operating Procedures (SOP): There are detailed explanations of how a policy is to be implemented.

Third Party Logistics (3PL): It is the activity of outsourcing activities related to Logistics and Distribution.

Universal Product Code (UPC): It is a barcode symbology (i.e., a specific type of barcode), that is widely used in North America, and in countries including the UK, Australia, and New Zealand for tracking trade items in stores.

Warehouse Management System: It is a key part of the supply chain and primarily aims to control the movement and storage of materials within a warehouse and process the associated transactions, including shipping, receiving, put away and picking.

Lab Exercise

1. Write a program to calculate freight of air.
2. Write a program to create an XML DTD.

13.6 Review Questions

1. What is XML Tool?
2. Define DTD (Document Type Definition).
3. What is XML Namespace?
4. What is data binding?
5. What is the difference between DTD and Schema?
6. How do you parse the XML document?
7. Why is XML extensible?
8. Describe the logical structure of XML.
9. Define CSS and XSL.
10. What is Freight calculation?

Answers for Self Assessment Questions

1.	(*c*)	2.	(*a*)	3.	(*d*)	4.	(*a*)	5.	(*b*)
6.	(*a*)	7.	(*c*)	8.	(*b*)	9.	(*a*)	10.	(*a*)
11.	(*d*)	12.	(*b*)	13.	(*d*)	14.	(*b*)	15.	(*a*)

13.7 Further Readings

Books *Essentials of E-Commerce Technology*, by Rajaraman

Online link http://books.google.co.in/books?id=-8BsyuXuEOMC&pg=PA3&dq=Adding+E-Commerce+Essentials&hl=en&sa=X&ei=J5j-T43bKIfQr Qfm5sjPBg&ved=0CEQQ6AEwAg#v=onepage&q=Adding%20E-Commerce%20Essentials&f=false

Unit 14: Debugging and Optimization

Notes

CONTENTS

Objectives

After studying this unit, you will be able to:

- Explain debugging and optimization application
- Define optimizing using caching
- Discuss the optimizing via performance profiling

Introduction

Debugging is the process of finding and correcting the errors in a program. In normal Active Server Pages (ASP) file Response. Write statement was the only primary method to debug the errors and even Script Debugger was not sophisticated to debug, in comparison to the debuggers that Windows developers had for Visual Basic and C++ code. All these deficiencies have been changed in .Net Framework because the Common Language Runtime provides integrated debugging for all the applications. The primary objectives for debugging in ASP.NET include both in-process and out-of-process debugging, remote process debugging, managed and unmanaged code debugging, edit and continue, and mixed language support.

Notes

We can enable debugging in ASP.NET page by writing the <%@Page debug= "True" %> page directive at the top of the page. This statement asks the compiler to exclude debugging symbols into the compiled page, and allows the debugger to attach to the running program. But before using the debugger, we should also need to view the web page in the browser so that the debugging symbols are loaded for the page.

The SDK debugger that comes along with .NET SDK can be found in the Guidebook directory. This debugger has all the features such as viewing contents of variables, setting breakpoints on specific exceptions, expressions, or specific lines, stepping into, out of, and over code, and viewing the call stack, threads and modules as that of Visual Studio debugger except that SDK debugger does not support remote debugging or Edit and Continue features. Therefore it can be presumed that SDK debugger is a read-only debugger that does not allow us to edit code inline. We can just edit the code externally and to view the new changes made in the application we have to rerun the application.

Just-In-Time debugging is another technique to debug a program that is started outside of Visual Studio. If we have enabled Just-In-Time debugging, we can view a dialog box when a crash occurs. Debugging a traditional ASP application generally involves placing Response. Write statements throughout our code to track variable values and execution paths. If we fail to remove debugging statements before we deploy our application, the statements are visible to users. ASP.NET makes Web application debugging much simpler to use. We can use tracing statements to debug our application in a way that is similar to using Response. Write statements. However, we can leave the tracing statements in our application, even after deployment.

We can configure the Web application to allow tracing at either the page level or the application level, and we can easily turn it on and off. We can also control the destination of the tracing Output to allow only certain users to see the debugging Output.

ASP.NET incorporates a variety of features and tools that allow us to design and implement high-performance Web applications. These features include:

- An improved process model.
- Compilation of requested pages and automatic storage on the server.
- ASP.NET-specific performance counters.
- Web application testing tools.

ASP.NET also gives us the ability to create Web applications that can handle the demands of processing large numbers of requests simultaneously.

14.1 Debugging in an ASP.NET Application

ASP.NET is a web application framework developed by Microsoft to allow programmers to build dynamic web sites and web applications. ASP.NET supports compiling applications in a special debug mode that facilitates developer troubleshooting. Debug mode causes ASP.NET to compile applications with extra information that enables a debugger to closely monitor and control the execution of an application. Applications that are compiled in debug mode execute as expected. However, the performance of the application is affected. To avoid the effect on performance, it is a good idea to enable debugging only when a developer is doing interactive troubleshooting. By default, debugging is disabled, and although debugging is frequently enabled to troubleshoot a problem, it is also frequently not disabled again after the problem is resolved.

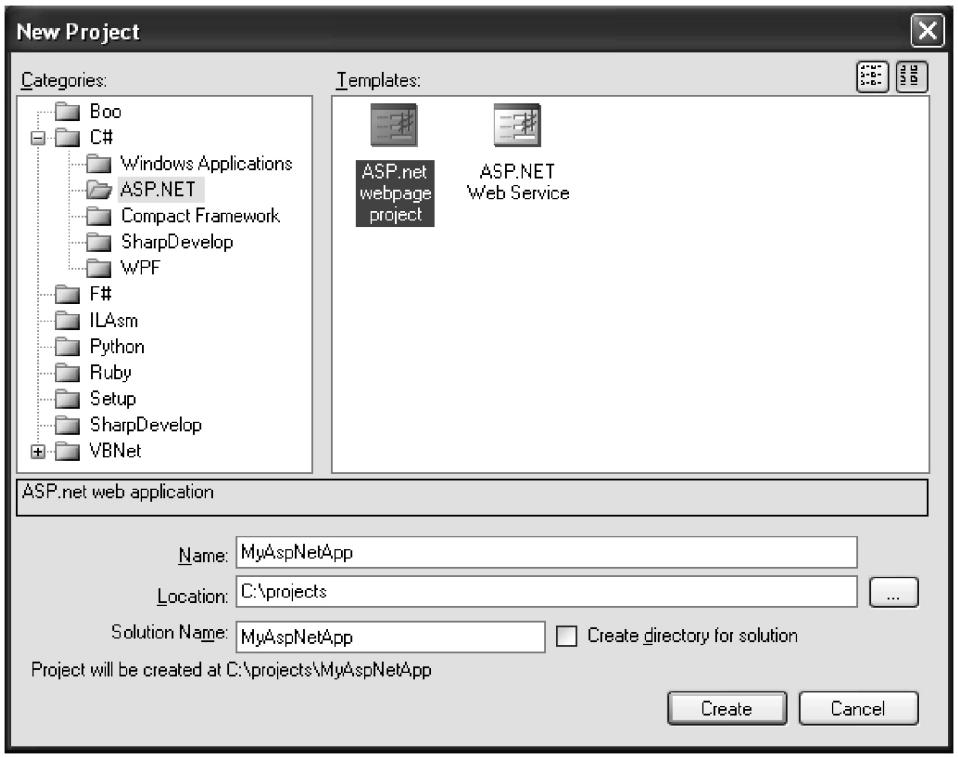
14.1.1 Creating an ASP.NET Application

Before we begin it should be noted that SharpDevelop does not have great ASP.NET support. There is no web forms designer, no intellisense for ASP.NET pages (.aspx) and no support for

Notes

the new style ASP.NET Web Site projects introduced in Visual Studio 2005. We can however create an ASP.NET Web Application project or an ASP.NET Web Service in SharpDevelop. If we are looking for a better ASP.NET experience then we should use However if still want to use SharpDevelop then here is how to debug our ASP.NET application. First let us create an ASP.NET Web Application. From the File menu select New, then Solution, to open up the New Project dialog (See Figure 14.1).

Figure 14.1: Creating new ASP.NET Application



On the left hand side of this dialog select ASP.NET category underneath the C# category. On the right hand side select the ASP.NET web page project icon. Give project a name and choose where we want the project created and click the Create button. SharpDevelop will then create a basic ASP.NET.

Task Create login page for any institute using an ASP.NET Application.

14.1.2 Download Cassini

In order to be able to debug the ASP.NET application we will be using Cassini. Cassini is a lightweight open source web server. It was originally released as a sample by Microsoft. Dmitry Robsman then updated the sample to run under .NET 3.5. After that different versions based on Dmitry's original work were released on codeplex. Any of these versions of Cassini, in the list below, will work with SharpDevelop. The version of Cassini provided by Dmitry requires us to compile the source code. The other versions do not. Cassini++.

1. Dmitry Robsman - Cassini v3.5 or Cassini v3.5.0.2 with friendly URL routing support.
2. Cassini 3.5 Developers Edition.

Notes

Download Cassini++ from the link provided and extract the files to a directory on the same machine that we are running Sharp Develop on.

14.1.3 Debugging an ASP.NET Application

First let us set a breakpoint in our application. Find the Default.aspx.cs file in our project and open it in the text editor. Locate the Page Load method and set a breakpoint inside this method on the Response.Write line (See Figure 14.2).

Figure 14.2: Debugging an ASP.NET Application

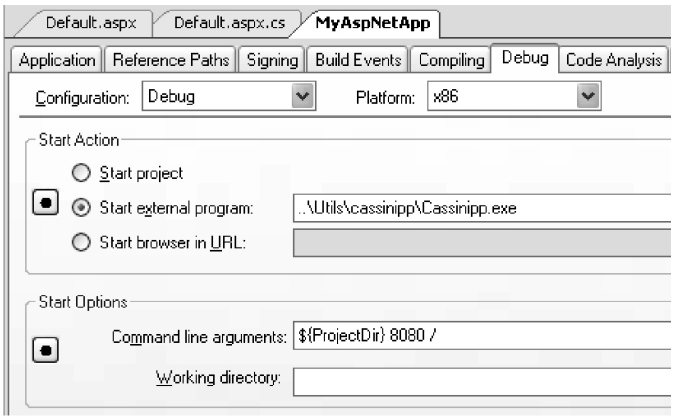
```
25  public class Default : Page
26  {
27      //<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
28      Data
34      //<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
35      Page Init & Exit (Open/Close DB connections here...)
46      //<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
47      #region Page Load
48      private void Page_Load(object sender, EventArgs e)
49      {
50          Response.Write(@"Hello #Develop<br>");
51          //-----------------------------------------------------
52          if(IsPostBack)
53          {
54          }
55          //-----------------------------------------------------
56      }
57      #endregion
```

Now we need to configure the project to run Cassini when we debug our application. From the Project menu select Project Options. In the options dialog select the Debug tab. Set the Start Action to Start external program and enter the path to the Cassini executable. In the Start Options enter the command line that will be passed to Cassini. The Cassini command line is of the form:

<Physical-path> <port> <virtual-path>

In the Figure 14.3 we can see an example command line passed to Cassini. In this case Cassini will listen for requests on port 8080, the virtual path will be set to "/" and the physical path to our project is specified by a using SharpDevelop property ${ProjectDir} which will be replaced when we start debugging. We can put the full path to our project directory here instead of using the ProjectDir property.

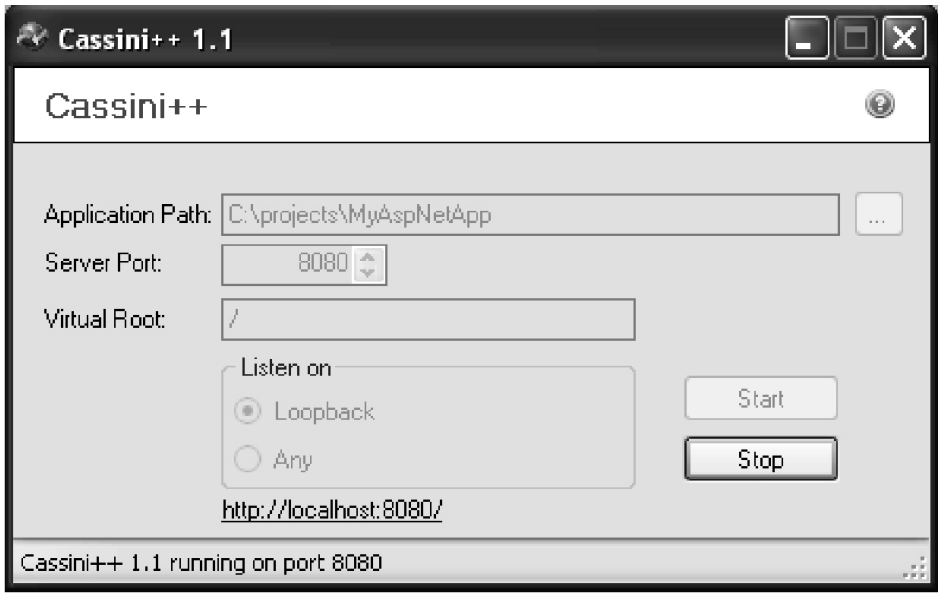
Figure 14.3: Command Line Passed to Cassini



Notes

Save the changes we made to the project options. Then select Run from the Debug menu. Cassini should then start (See Figure 14.4).

Figure 14.4: Start the Cassini



Click the http://localhost:8080/ hyperlink at the bottom of the Cassini window to visit our application's web page in the default browser. When the page is loaded in the browser the breakpoint in Page Load should then be hit and we can start debugging (See Figure 14.5).

Figure 14.5: Breakpoint in Page Load

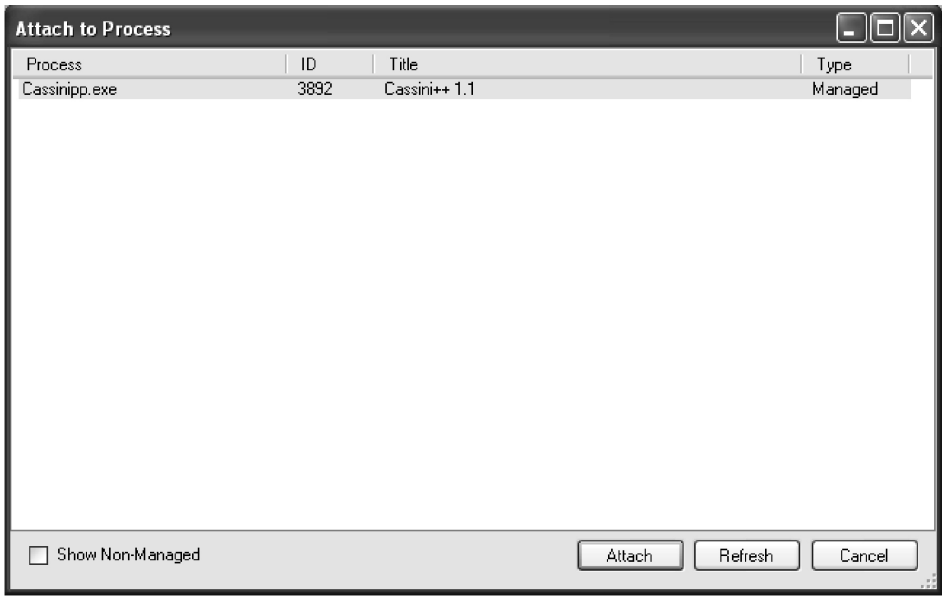
14.1.4 Attach to Process

As an alternative to configuring SharpDevelop to start Cassini every time we debug our ASP.NET application we can leave Cassini running and attach to its process when we need to.

First on our desktop create a shortcut to Cassini and specify the appropriate Cassini command line as explained earlier. Note that the ${ProjectDir} property cannot be used here so instead we will need to put the full path to our project. Start Cassini by double clicking the shortcut and use a browser to display our web page. Now in SharpDevelop select Attach to Process from the Debug menu. Select the Cassini process and click the Attach button.

Notes

Figure 14.6: Attach to Process



Now when we press Ctrl+F5 in the browser showing our web page to refresh the page the breakpoint in the Page Load method should be hit and we can start debugging.

14.1.5 Disable Debugging for an ASP.NET Application

To enable debugging, modify either the Web.config file or the Machine.config file, as detailed in the following steps.

Modify the Web.config File

To enable debugging, add the compilation element to the Web.config file of the application. The Web.config file is located in the application directory. To do this, follow these steps:

1. Open the Web.config file in a text editor such as Notepad.exe. Web.config file is typically located in the application directory.
2. In the Web.config file, locate the compilation element. Debugging is enabled when the debug attribute in the compilation element is set to true.
3. Modify the debug attribute to false, and then save the Web.config file to disable debugging for that application.

The following code sample shows the compilation element with debug set to false:

```
<compilation
   debug= “false”
/>
```

Save the Web.config file. The ASP.NET application automatically restarts.

Modify the Machine.config File

We can also enable debugging for all applications on a system by modifying the Machine.config file. To confirm that debugging has not been enabled in the Machine.config file, follow these steps.

1. Open the Machine.config file in a text editor such as Notepad.exe. The Machine.config file is typically located in the following folder:

   %SystemRoot%\Microsoft.NET\Framework\%VersionNumber%\CONFIG\

Notes

2. In the Machine.config file, locate the compilation element. Debugging is enabled when the debug attribute in the compilation element is set to true.
3. If the debug attribute is true, change the debug attribute to false.
4. The following code sample shows the compilation element with debug set to false:

   <compilation

   debug= "false"

   />

   Save the Machine.config file.

14.2 Optimizing Using Caching

ASP.NET 3.5 intrinsic cache functionality has long been one of my favourite aspects of ASP. NET. With a little bit of planning and organization, we can use it to create highly scalable applications with robust fidelity. Optimization is the act of improving a method or design. In programming, optimization usually takes the form of increasing the speed of an algorithm, or reducing the resources it requires.

Best ways are to optimize our use of the ASP.NET cache and what some of the tips are in regards to how to determine what should and should not go in the cache. Also, are there any rules of thumb for determining how long something should say in the cache?

14.2.1 ASP.NET Caching Features

The ASP.NET 3.5 Cache API was a revolutionary feature that provided capabilities such as declarative output caching, programmatic output caching, and invalidation of cached items when the contents of an XML file or another cached item change. Even though all these increased the performance of Web applications, ASP.NET 3.5 did not provide a mechanism for invalidating the data in a cache object when the data in a database changes. This much sought after feature will finally ship with ASP.NET 3.5 Apart from this, ASP.NET 3.5 will also provide functionalities to cache the output of a SqlDataSource control, which enable us to take advantage of caching without writing a single line of code.

In ASP.NET 3.5, caching has improved in a couple of notable ways. Probably the most interesting improvement is the introduction of database-triggered cache invalidation. In ASP.NET 3.5, we can invalidate a cached item based on some pre-defined conditions, such as change in an XML file, change in another cache item, and so on. By using this feature, we can remove or invalidate an item from the cache when the data or another cached item changes. However, the ASP.NET 3.5 Cache API does not allow us to invalidate an item in the cache when data in a SQL Server database changes, even though most applications require this capability. ASP.NET 3.5 addresses this by providing the database-triggered cache invalidation capability that allows us to ensure that items in the cache are kept up to date with the changes in the database.

Another important caching feature in ASP.NET 3.5 is the ability to enable caching at the SqlDataSource level. The SqlDataSource control is designed to work with SQL Server, OLE DB, ODBC, and Oracle databases. As the name suggests, this control enables us to select, update, delete, and insert data using SQL commands. With the ability to set caching attributes at the SqlDataSource control level, we now have a finer level of control over the cached data.

The ASP.NET 3.5 also provides a new control named Substitution, which we can use to inject dynamic content in an otherwise cached Web page. If we have a page with output-caching but still want to display dynamic content (that needs to be generated every time the page is requested), consider using the Substitution control.

The following sections provide examples of the above features.

Notes

Time-Based Cache Invalidation in a SqlDataSource Control

Caching in ASP.NET is a powerful feature that can increase the performance of a Web application. In fact, the most dramatic way to improve the performance of a database-driven Web application is through caching. Retrieving data from a database is one of the slowest Web site operations that we can perform. However, if we can cache the database data in memory and avoid accessing the database with every page request, we can dramatically increase the performance of our application.

ASP.NET 3.5 provides a number of enhancements to the caching feature set in ASP.NET 3.5 One new feature is the ability to specify the caching attributes as part of the data source control declarations. The new data source controls in ASP.NET 3.5 work seamlessly with the new caching features of ASP.NET 3.5, enabling us to set the caching attributes as part of the SqlDataSource control declaration.

We can set the following two properties in the SqlDataSource control to enable caching:

Enable Caching: By setting this attribute to true, we enable caching in a SqlDataSource control.

Cache Duration: This property allows us to set or get the duration of the cached data in the SqlDataSource control. This attribute is specified in terms of seconds.

For the purposes of this example, consider a categories and products table in the Northwind database. It displays all the categories in a DropDownList and the products that belong to a specific category in a GridView control. Start by creating a new Web site named Caching in Visual Studio 2005. Next, add a Web page named TimeBasedCaching.aspx to it. Modify the code in the TimeBasedCaching.aspx file to look like the following:

```
<%@ Page Language= "C#" %>
  <html>
  <head>
<title>SqlDataSource Control Caching And Parameters</title>
  </head>
  <body>
  <form id= "form1" runat= "server">
  <asp:DropDownList DataValueField= "CategoryID"
  DataTextField= "CategoryName"
  DataSourceID= "CategoriesDataSource"
  ID= "DropDownList1" Runat= "server" AutoPostBack= "True">
  </asp:DropDownList>
    <br/><br/>
  <asp:GridView ID= "GridView1" Runat= "server"
  DataSourceID= "ProductsDataSource"
  DataKeyNames= "ProductID" AutoGenerateColumns= "False">
  <Columns>
  <asp:BoundField HeaderText= "ProductID" DataField= "ProductID"
SortExpression= "ProductID"/>
  <asp:BoundField HeaderText= "Timestamp" DataField= "Timestamp"
  < SortExpression= "Timestamp"/>
  <asp:BoundField HeaderText= "ProductName"
```

Notes

```
DataField= "ProductName"
SortExpression= "ProductName"/>
<asp:BoundField HeaderText= "QuantityPerUnit"
DataField= "QuantityPerUnit"
SortExpression= "QuantityPerUnit"/>
<asp:BoundField HeaderText= "UnitPrice" DataField= "UnitPrice"
SortExpression= "UnitPrice"/>
</Columns>
</asp:GridView>
<asp:SqlDataSource ID= "ProductsDataSource" Runat= "server"
    SelectCommand= "SELECT DatePart(second, GetDate())
    As Timestamp, *
    FROM [Products] where CategoryID = @CategoryID"
    ConnectionString= "<%$ ConnectionStrings:Northwind %>"
    EnableCaching= "True" CacheDuration= "10">
    <SelectParameters>
    <asp:ControlParameter Name= "CategoryID"
    ControlID= "DropDownList1"
    PropertyName= "SelectedValue"/>
    </SelectParameters>
    </asp:SqlDataSource>
    <asp:SqlDataSource ID= "CategoriesDataSource" Runat= "server"
    SelectCommand= "SELECT * FROM [Categories]"
    ConnectionString= "<%$ConnectionStrings:Northwind %>"
    EnableCaching= "True" CacheDuration= "10"/>
    </form>
    </body>
    </html>
```

In the preceding code, the connection string to the database is retrieved from the web.config file. The web.config file contains the following connectionStrings element:

```
<connectionStrings>
      <add name= "Pubs"
connectionString= "server=localhost;database=Pubs;
         trusted_connection=true"/>
       <add name= "Northwind"
         connectionString= "server=localhost;database=Pubs;
         trusted_connection=true"/>
</connectionStrings>
```

Now that the required connection string is defined in the web.config file, the SqlDataSource control can use that connection string by using the following declaration:

1. <%$ ConnectionStrings:Northwind %>

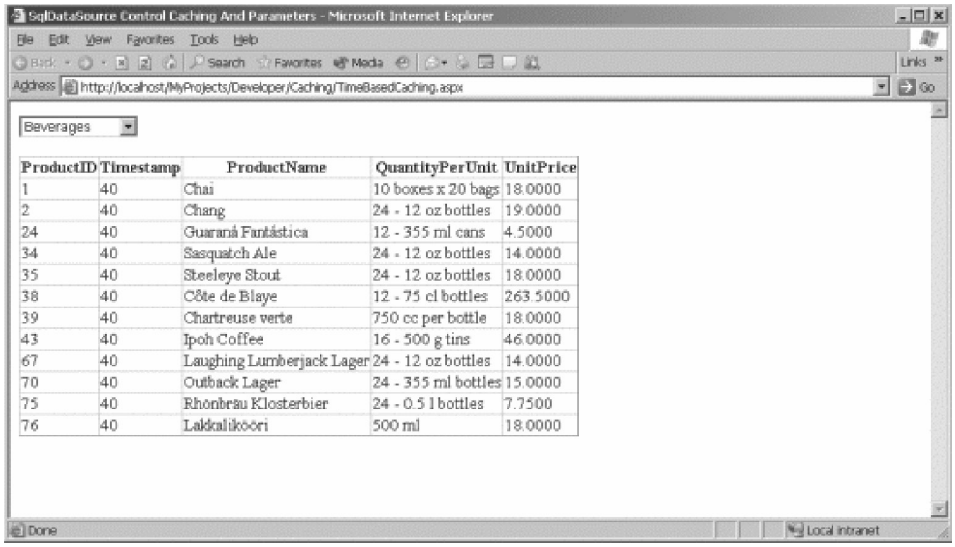
Notes

The above code retrieves the connection string value defined in the connectionString attribute of the Northwind connection string element.

The SqlDataSource control also has the EnableCaching property set to true, which results in the SqlDataSource automatically caching the data retrieved by the SelectCommand. The CacheDuration property enables us to specify (in seconds) how long the data should be cached before it is refreshed from the database. By default, the SqlDataSource will cache data using an absolute expiration policy, meaning that the data will be refreshed every so many seconds as specified in the CacheDuration property.

We also have the option of configuring the SqlDataSource to use a sliding expiration policy, by which the data is not dropped as long as it continues to be accessed. Employing a sliding expiration policy is useful whenever we have a large number of items that need to be cached, because the policy enables us to keep only the most frequently accessed items in memory. In the above example, we cached the results of the SQL query for 10 seconds by setting the EnableCaching and CacheDuration attributes to True and 10, respectively (see the output Figure 14.7).

Figure 14.7: CacheDuration Attributes



As we can see, the values in the timestamp column remain the same for 10 seconds. After that they will be refreshed with the new timestamp values from the database.

One of the most important factors in building high-performance, scalable Web applications is the ability to store items, whether data objects, pages, or parts of a page, in memory the initial time they are requested. We can store these items on the Web server or other software in the request stream, such as the proxy server or browser. This allows us to avoid recreating information that satisfied a previous request, particularly information that demands significant processor time or other resources. Known as caching, it allows us to use a number of techniques to store page output or application data across HTTP requests and reuse it. Thus, the server does not have to recreate information, saving time and resources.

ASP.NET provides two types of caching that we can use to create high-performance Web applications. The first is called output caching, which allows us to store dynamic page and user control responses on any HTTP 1.1 cache-capable device in the output stream, from the originating server to the requesting browser. On subsequent requests, the page or user control code is not executed; the cached output is used to satisfy the request. The second type of caching

Notes

is traditional application data caching, which we can use to programmatically store arbitrary objects, such as data sets, to server memory so that our application can save the time and resources it takes to recreate them.


Did u know?

Microsoft C 1.0, based on Lattice C, was Microsoft's first C product in 1983.

14.2.2 The Upside of Using the Cache

ASP.NET's cache can be used in a variety of ways-too many ways to cover in which can be problematic in some cases due to how difficult this makes it to unit test repositories or factories. Another caveat is that it is possible to shoot our self in the foot by allowing cached items to trump actual, tangible, data as it should exist within our application.

For the sake of simplicity, suppose that tracks products that can be sold online. Furthermore, assume that while the underlying data storage mechanism for all of these products is a SQL Server database, the only way that products can be updated or added to the system is through my application. But let us also assume that this little application gets tons of traffic, and that products are fairly infrequently added or updated. And, when they are update, it is usually just to reflect something critical, such as a change in inventory status.

Assume LINQ to SQL to pull these products in and out of the database and into my application. With such a scenario we could easily create a repository that would put all of my products into memory on the web server when requested like so:

```
Public List<Product> GetAllProducts ()
        {
            List<Product> output =
                HttpContext.Current.Cache [“AllProducts”] as List<Product>
            if (output == null)
            {
            output = db.Products
                .Select()
                .ToList();
            HttpContext.Current.Cache.Add(
                “AllProducts”, output,null,Cache.NoAbsoluteExpiration,
                Cache.NoSlidingExpiration, CacheItemPriority.AboveNormal, null);
            }
            return output;
        }
```

With this approach, we are then free to slice and dice my products for consumption by various other methods within my repository, such as the ability to get all products from a certain product family:

```
        public List<Products> GetProductsByProdLine(string prodLine)
        {
            return this.GetAllProducts()
                .Where(prod => prod.ProductLine == prodLine).ToList();
        }
```

And, as we can see, we are able to do that all from within memory using LINQ to Objects.

Notes

Assuming that we do not want to iterate over my product list each time we need a specific product, we can also store individual requests to products within the cache as well:

```
public Product GetProductById(int id)
        {
            Product output =
                HttpContext.Current.Cache[“Product” + id] as Product;
            if (output == null)
            {
            output = this.GetAllProducts()
                .Where(p => p.ProductId == id)
                .SingleOrDefault();
            HttpContext.Current.Cache.Add(
                “Product” + id, output, null, Cache.NoAbsoluteExpiration,
            Cache.NoSlidingExpiration, CacheItemPriority.BelowNormal, null);
            }
            return output;
        }
```

Doing so can help make data retrieval much faster, though it will take up a tiny bit of RAM. Most CLR objects are tiny, so they do not take up much RAM. If our objects are huge, then we might want to rethink this pattern of usage, if we do not have gobs of RAM at hand. In most cases though, storing business objects in RAM is usually preferable to recreating them when needed.

And, in cases where caching a single entity that was pulled from a collection of cached objects does not make sense, just remember that using the pattern of caching a single object by ID can provide a huge boost in performance if that object is being requested on a regular basis.


Caution

Do not create MemoryCache instances unless it is required. If you create cache instances in client and Web applications, the MemoryCache instances should be created early in the application life cycle.

14.2.3 The Downside of Extensive Caching

Taking this approach also adds two big problems. First, implementing unit tests on top of a repository that takes advantage of cache can be a total beast because we have thrown a big set of dependencies, which cannot be easily tested, into the mix.

Second, it is pretty easy to let cached items “trump” real data. This is because we now have a number of cached objects that need to be removed should we change any of our underlying data. Common approaches to dealing with this include setting sliding or absolute expiration times that are good enough or using cache invalidation callbacks from SQL Server. Setting expiration times can provide a great boost even in highly volatile environments by setting cache timeouts for just a few seconds and accepting that data can/will be a few seconds old in some cases. Using cache invalidation callbacks is also a great way to keep volatile data up-to-date while enjoying the benefits of cache. But it requires a bit of extra work to set up.

14.2.4 Standardizing Cache Access

The final benefit of this approach is that it let me standardize on the creation of cache keys. If we look at the examples above, we just are creating keys for any cached objects on the fly. And, since these keys are strings, IntelliSense would not help call out typos or other errors that would cause stupid caching problems.

Notes

So, with our interface, or cache proxy, and also added a couple of additional helper methods. One helps us generate cache keys in a standardize format that takes, more or less, a hierarchical approach to naming (more on that in a second). This particular approach is highly coupled to the model hierarchy within the site/solution we are currently working on, but the pattern is pretty simple:

```
public string GetCacheKey(string repoName, ObjectType type,
string? id)
{
    return string.Format("{0}::{1}->{2}",
      new object[] {repoName, type, id });
}
```

First, we just create a simple ObjectType Enum to list the three (in this case) different object types "Single" objects (where the id is the id of the object itself), "Sets" of objects (where the id is the id of the parent), and "All" where what's being cached represents an entire collection of objects, as is the case with the GetAllProducts() method, above.

The use of a constant to designate the repository name, along with wrapper methods to handle cache Input and Output (as well as to make a cache provider neutral mapping for cache retention priorities), yields an approach like this:

```
CacheProvider CacheProxy;
const string REPO_NAME = "Products";
public ProductRepository(ICacheProvider provider)
{
  this.CacheProxy = provider;
}
public List<Product> GetAllProducts()
{
  string cacheKey =
    CacheProxy.GetCacheKey(REPO_NAME, ObjectType.All, "");
List<Product> output =
  CacheProxy.Get(cacheKey) as List<Product>;
if (output == null)
{
  output = db.Products
    .Select()
    .ToList();
CacheProxy.Add(cacheKey, output, CacheType.Keep);
}
  return output;
}
```

Where standardized cache key entries get really cool though, is when it comes to updating, removing, or adding new products or objects. Because my cache keys are standardized, if add a new Product, the method that handles that addition can simply create a new cache key for the repo-name with an Object Type of All and kick out just the collection of All Products. Individual Products, however, remain cached because they're all using different Cache Keys. Likewise, however, they can be removed as needed by simply generating cache keys matching their keys, and they will be dropped. So, for example, in a Repository method that Updates a

Notes

given product, the approach would be to handle the update as normal, then create a cache key for all products, and one for just the product ID in question. Then use those keys to kick both objects out of cache.

This approach does not require much more coding than normal, but it allows much more robust caching without the need to accept a certain threshold of stale data, nor does it require incurring the overhead of SQL Server cache invalidation.

Finally, since this approach to creating cache keys results in hierarchical names, it becomes fairly easy to remove entire blocks of cache key entries by creating a partial cache key and then removing any entry within the native cache providers' set of keys that .StartsWith(partialKey) as it were.

The point is that a bit of planning and effort can help take an already powerful component of ASP.NET and make it even more robust in terms of enabling unit testing, mocking, and more powerful and robust access—all through the use of a simple wrapper.


Did u know?

Visual C++ 2010 (known also as Visual C++ 10.0) was released on April 12, 2010, and it is currently the latest stable release.

Self Assessment Questions

1. …… is the process of finding and correcting the errors in a program.
   (*a*) Compiler (*b*) Debugging
   (*c*) Interpreter (*d*) None of these
2. ASP.NET is a web application framework developed by .....
   (*a*) Oracle (*b*) Sun Java
   (*c*) Microsoft (*d*) All of these
3. Cassini is a lightweight …… source web server.
   (*a*) open (*b*) close
   (*c*) both of (*a*) and (*b*) (*d*) curve
4. Performance profiling is a ..... process that requires developers to run through repeated iterations of profiling and optimization.
   (*a*) strategic (*b*) easy
   (*c*) raw (*d*) complicated

True or False

5. Optimization is the act of improving a method or design.
   (*a*) True (*b*) False

14.3 Optimizing via Performance Profiling

Performance profiling is a complicated process that requires developers to run through repeated iterations of profiling and optimization. There are a number of factors affecting performance that operate independently and so must be tackled independently, often by different groups of people. Developers must tackle application coding issues by identifying the offending code blocks and then rewriting or optimizing them. System administrators must tackle server resource problems by examining the full set of tasks that the server is handling. Performance profiling has to be a cooperative task between different groups of people because at the end of the day, a user who experiences slow performance will be unhappy with the experience, regardless of whose piece of the puzzle is causing the issue. Performance profiling affects everyone on the technical team, and so it must engage everyone as well.

Notes

Performance profiling is a time-dependent activity because application performance changes over time. Performance may change periodically throughout the day or as the load on the application fluctuates. Alternatively, application performance may experience degradation over a long period of time, particularly as the database grows. Performance profiling must begin with a baseline, which is a set of metrics that define the performance at a specific time, under a specific set of conditions. The baseline is the measure against which future performance will be compared. A baseline includes a description of both hardware and software, and it typically includes a range of performance numbers that were derived without changing the hardware configuration.

This is an example of a baseline description:

- The Web application runs on a single, dual-processor 400MHz server with 512MB of RAM. The application was subjected to an increasing load of 10 to 100 concurrent users with five threads. The application serves between 18 and 45 requests per second, diminishing with an increasing number of concurrent users. Response times for 10 concurrent users varied between 0.8 and 3.5 seconds, depending on the page. The Home page, which is cached, responds in 0.8 seconds, and more intensives careens take longer to deliver to the client.

Figure 14.8 shows the baseline application performance's throughput, and Figure 14.9 shows the baseline application performance's response time.

Figure 14.8: Baseline Application Performance Throughput

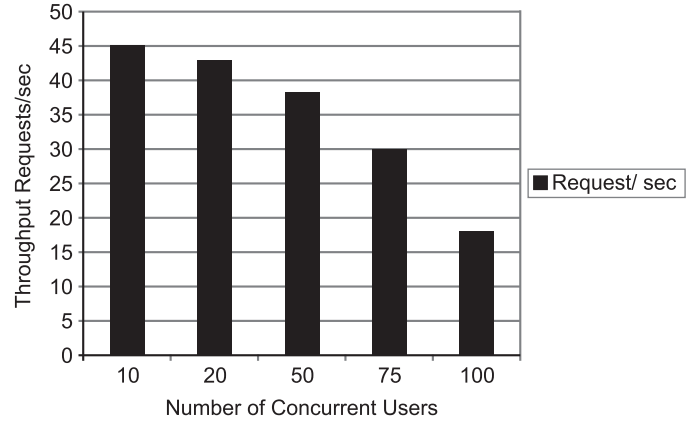
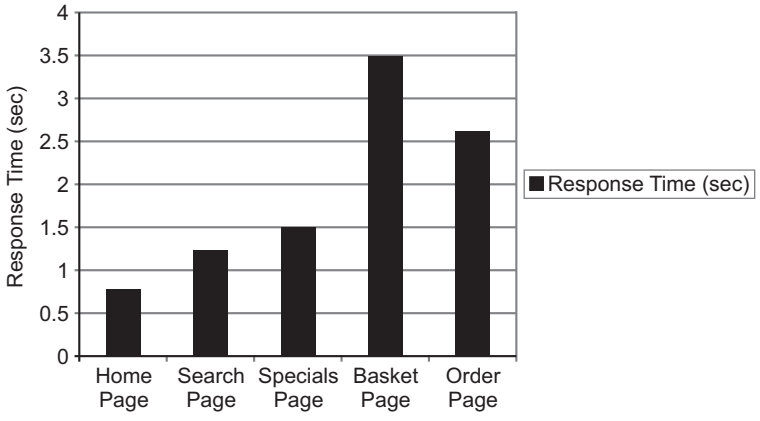


Figure 14.9: Baseline Application Performance Response Time



Notes

These graphs focus on just two metrics namely, throughput and response time. A complete baseline should contain data for several metrics but should always include basic metrics such as throughput and response time because these most directly affect the user's experience with the application.

Task Search about the caching process. Differentiate between cache memory and RAM.

14.3.1 ASP.NET Performance

Creating Web applications that respond to user requests quickly, even when a large number of requests are being processed on the server, has been a challenge for developers and IT personnel since the Internet began. Monitoring Web site performance is something that all Internet and intranet developers need to be able to do. ASP.NET was designed with this in mind.

The ASP.NET model provides a number of built-in performance enhancements that are not included in earlier versions of ASP. Particularly, there are two enhancements involved in processing HTTP requests. First, when an ASP.NET page is requested for the first time, an instance of the Page class is dynamically compiled. (In earlier versions of ASP, page code was interpreted for requests in the order that they appeared on the page.) The common language runtime just-in-time (JIT) compiles ASP.NET managed page code to the native code of the processing server at run time. Second, when the Page instance has been compiled for the first request, it is cached on the server. For each subsequent request for that page, the cached instance of the class is executed. After the initial request, the Page class is recompiled only when the original source for the page or one of its dependencies has changed.

In addition, ASP.NET caches internal objects, such as server variables, to speed user code access. As a part of the .NET Framework, ASP.NET benefits from the performance enhancements provided by the common language runtime, including the previously mentioned JIT compiling, a fine-tuned common language runtime for both single and multiprocessor computers, and so on.

These enhancements, unfortunately, cannot protect us from writing code that does not perform well when a large number of HTTP requests are processed by our application simultaneously. We must test our application to ensure that it meets the demands of its users. There are four common performance measures that we can test to ensure that our application is performing well.

Execution Time

The time it takes to process a request, usually measured between the first byte and the last byte returned to the client from the server. Execution time directly affects the throughput calculation.

Response Time

The length of time between when a request is requested and when the first byte is returned to the client from the server. This is often the most perceptible aspect of performance to the client user. If an application takes a long time to respond, the user can become impatient and go to another site. The response time of an application can vary independently of (even inversely to) the rate of throughput.

Scalability

The measurement of an application's ability to perform better as more resources (memory, processors, or computers) are allocated to it. Often, it is a measurement of the rate of change of throughput with respect to the number of processors.

Throughput

Notes

The number of requests a Web application can serve per unit of time often measured in requests per second. Throughput can vary, depending on the load (number of client threads) applied to the server. This is usually considered the most important performance metric to optimize.

To write applications that perform well, it is important to maintain a balance of these metrics. No single measurement can characterize how our application will behave under varying circumstances, but several measurements taken together can show how well our application is performing. For more information about taking measurements of this kind, as well as information about the performance counters provided with ASP.NET.

ASP.NET has many secrets, which when revealed can give us big performance and scalability boost. For instance, there are secret bottlenecks in Membership and Profile provider which can be solved easily to make authentication and authorization faster. Furthermore, ASP.NET 3.5 HTTP pipeline can be tweaked to avoid executing unnecessary code that gets hit on each and every request. Not only that, ASP.NET Worker Process can be pushed to its limit to squeeze out every drop of performance out of it. Page fragment output caching on the browser (not on the server) can save significant amount of download time on repeated visits. On demand UI loading can give our site a fast and smooth feeling. Finally, Content Delivery Networks (CDN) and proper use of HTTP Cache headers can make our website screaming fast when implemented properly.

The following techniques are:

- ASP.NET pipeline optimization
- ASP.NET process configuration optimization
- Things we must do for ASP.NET before going live
- Content Delivery Network
- Caching AJAX calls on browser
- Making best use of Browser Cache
- On demand progressive UI loading for fast smooth experience
- Optimize ASP.NET Profile provider
- How to query ASP.NET Membership tables without bringing down the site
- Prevent Denial of Service (DOS) attack

The above techniques can be implemented on any ASP.NET website, especially those who use ASP.NET Membership and Profile provider.

14.3.2 ASP.NET Pipeline Optimization

There is several ASP.NET default Http Modules which sit in the request pipeline and intercept each and every request. For example, SessionStateModule intercepts each request, parses the session cookie and then loads the proper session in the Http Context. Not all of these modules are always necessary. For example, if we are not using Membership and Profile provider, we do not need FormsAuthentication module. If we are not using Windows Authentication for our users, we do not need WindowsAuthentication. These modules are just sitting in the pipeline, executing some unnecessary code for each and every request.

The default modules are defined in machine.config file (located in the$WINDOWS$\Microsoft.NET\Framework\$VERSION$\CONFIG directory).

<httpModules> <add name= “OutputCache” type= “System.Web.

Notes

```
Caching.OutputCacheModule"/>
 <add name= "Session" type= "System.Web.SessionState.
SessionStateModule"/>
 <add name= "WindowsAuthentication" type= "Sys
tem.Web.Security.WindowsAuthenticationModule"/>
 <add name= "FormsAuthentication"
 type= "System.Web.Security.FormsAuthenticationModule"/>
 <add name= "PassportAuthentication"
 type= "System.Web.Security.PassportAuthenticationModule"/>
 <add name= "UrlAuthorization" type= "System.Web.Security.
UrlAuthorizationModule"/>
 <add name= "FileAuthorization" type= "System.Web.Security.
FileAuthorizationModule"/>
 <add name= "ErrorHandlerModule" type= "System.Web.Mobile.
ErrorHandlerModule,
 System.Web.Mobile, Version=1.0.5000.0,
 Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"/>
</httpModules>
```

We can remove these default modules from our Web application by adding <remove> nodes in our site's web.config. For example:

```
<httpModules>
 <!-- Remove unnecessary Http Modules for faster pipeline -->
 <remove name= "Session"/>
 <remove name= "WindowsAuthentication"/>
 <remove name= "PassportAuthentication"/>
 <remove name= "AnonymousIdentification"/>
 <remove name= "UrlAuthorization"/>
 <remove name= "FileAuthorization"/>
</httpModules>
```

The above configuration is suitable for websites that use database based Forms Authentication and do not need any Session support. So, all these modules can safely be removed.

14.3.3 ASP.NET Process Configuration Optimization

ASP.NET Process Model configuration defines some process level properties like how many number of threads ASP.NET uses, how long it blocks a thread before timing out, how many requests to keep waiting for IO works to complete and so on. The default is in many cases too limiting. Nowadays hardware has become quite cheap and dual core with gigabyte RAM servers have become a very common choice. So, the process model configuration can be tweaked to make ASP.NET process consume more system resources and provide better scalability from each server.

A regular ASP.NET installation will create machine.config with the following configuration:

```
<system.web>
 <processModel autoConfig= "true"/>
```

Notes

We need to tweak this auto configuration and use some specific values for different attributes in order to customize the way ASP.NET worker process works.

For example:

```
<processModel
 enable= "true"
 Timeout= "Infinite"
 idleTimeout= "Infinite"
 shutdownTimeout= "00:00:05"
 requestLimit= "Infinite"
 requestQueueLimit= "5000"
 restartQueueLimit= "10"
 memoryLimit= "60"
 webGarden= "false"
 cpuMask= "0xffffffff"
 userName= "machine"
 password= "AutoGenerate"
 logLevel= "Errors"
 clientConnectedCheck= "00:00:05 "
 comAuthenticationLevel= "Connect"
 comImpersonationLevel= "Impersonate"
 responseDeadlockInterval= "00:03:00"
 responseRestartDeadlockInterval= "00:03:00"
 AutoConfig= "false"
 MaxWorkerThreads= "100"
 maxIoThreads= "100"
 minWorkerThreads= "40"
 minIoThreads="30"
 serverErrorMessageFile=""
 pingFrequency="Infinite"
 pingTimeout= "Infinite"
 asyncOption= "20"
 maxAppDomains= "2000"
/>
```

Here all the values are default values except for the following ones:

maxWorkerThreads

This is default to 20 per process. On a dual core computer, there will be 40 threads allocated for ASP.NET. This means at a time ASP.NET can process 40 requests in parallel on a dual core server. We have increased it to 100 in order to give ASP.NET more threads per process. If we have an application which is not that CPU intensive and can easily take in more requests per second, then we can increase this value. Especially if our Web application uses a lot of Web service calls or downloads/uploads a lot of data which does not put pressure on the CPU. When ASP.NET runs out of worker threads, it stops processing more requests that come in. Requests

Notes

get into a queue and keep waiting until a worker thread is freed. This generally happens when site starts receiving much more hits than we originally planned. In that case, if we have CPU to spare, increase the worker threads count per process.

maxIOThreads

This is default to 20 per process. On a dual core computer, there will be 40 threads allocated for ASP.NET for I/O operations. This means at a time ASP.NET can process 40 I/O requests in parallel on a dual core server. I/O requests can be file read/write, database operations, web service calls, and HTTP requests generated from within the Web application and so on. So, we can set this to 100 if our server has enough system resource to do more of these I/O requests. Especially when our Web application downloads/uploads data, calls many external web services in parallel.

minWorkerThreads

When a number of free ASP.NET worker threads fall below this number, ASP.NET starts putting incoming requests into a queue. So, we can set this value to a low number in order to increase the number of concurrent requests. However, do not set this to a very low number because Web application code might need to do some background processing and parallel processing for which it will need some free worker threads.

minIOThreads

Same as minWorkerThreads but this is for the I/O threads. However, we can set this to a lower value than min Worker Threads because there is no issue of parallel processing in case of I/O threads.

memoryLimit

Specifies the maximum allowed memory size, as a percentage of total system memory, that the worker process can consume before ASP.NET launches a new process and reassigns existing requests. If we have only our Web application running in a dedicated box and there is no other process that needs RAM, we can set a high value like 80. However, if we have a leaky application that continuously leaks memory, then it is better to set it to a lower value so that the leaky process is recycled pretty soon before it becomes a memory hog and thus keep our site healthy. Especially when we are using.

Case Study **Reduce the size of downloads**

The problem

WebPagetest is accounting for 1,425KB of data transferred. That is quite a lot.

The worst offender in GitHub's case seems to be the Gravatar 3rd party service. Gravatar is a popular service that allows showing a personal avatar for users of the interweb based on their e-mail address. In the case of that GitHub page, Gravatar is loading 34 images for a total of 500KB. That is more than a third of the payload.

The second worst offender is the host called camo.githubapp.com. It seems to be serving 2 images in our example. At first, was a bit puzzled about that host as I could not quite figure out why GitHub was serving directly 2 images worth 500KB (that is another third of our total payload) instead of using their CDN. After a bit of research I found the Camo project. GitHub is enforcing SSL for every page. When GitHub users write comments and

Contd...

insert funny cat pictures to illustrate their point, the said cat pictures might be served from a non SSL host. This will cause a nagging mixed content warning from the browser. Camo is basically an SSL image proxy that solves the issue. See more on this blog post. Note that one of the 2 images is taking 7 seconds to load.

Solutions

We have about 1,000KB of images. A common practice is to compress and optimize images. There are a lot of tools out there for that purpose. WebPagetest performance review tab estimates that some of the images can be compressed and save about 40KB. Note that it does not include one of the largest images, which is a animated gif proxied through Camo. I sent the gif to Smush.it and it came back 50KB lighter. Bottom line: potential savings would be around 10%. Compressing images might not be easily achievable. I do not think it would be a good idea to compress them on the fly (it would take some extra CPU resources to do that through Camo for instance). A better approach might be to have the image compressed at the source (in the case of Gravatar, when a user uploads an avatar). That would probably require a significant effort and I am not sure it will be worth the gain.

An alternative could be to implement asynchronous image loading. When we first load the GitHub page, none of that content is visible (all the images are below the fold). Knowing that, the HTML page could be generated with a placeholder for the image which would be loaded later on via javascript. The javascript could be listening for scroll events and monitor the visibility of the images in order to decide when to load them. Pamela Fox wrote about that not long ago. We can also see the technique in action on Meetup's website (for instance visit this meetup page and look at the sidebar with the list of attendees while we scroll). Note that for users browsing with javascript disabled, they would miss the images. I am not sure that would be a deal breaker as I am assuming most of GitHub users have javascript turned on.

Questions

1. How do reduce the size downloads?

2. Describe the javascript and html.

Self Assessment Questions

6. The SDK …… that comes along with .NET SDK can be found in the Guidebook directory.

   (*a*) compiler (*b*) debugger

   (*c*) interpreter (*d*) None of these.

7. ……. debugging is another technique to debug a program that is started outside of Visual Studio.

   (*a*) Just-In-Time (*b*) Just-on-Time

   (*c*) Just-out-Time (*d*) None of these.

8. ASP.NET is a web application framework developed by Microsoft to allow programmers to build ……. web sites and web applications.

   (*a*) static (*b*) static and dynamic both

   (*c*) dynamic (*d*) None of these.

9. The ASP.NET model provides a number of …… formance enhancements that are not included in earlier versions of ASP.

   (*a*) built-in (*b*) built-out

   (*c*) functions (*d*) None of these.

Notes

10. The SqlDataSource control also has the EnableCaching property set to true, which results in the SqlDataSource automatically caching the data retrieved by the …….
    (*a*) query command. (*b*) select command.
    (*c*) insert command (*d*) None of these.
11. ASP.NET provides two types of caching that we can use to create …….. Web applications.
    (*a*) low-performance (*b*) high-performance
    (*c*) static (*d*) None of these.
12. Performance profiling is a …….. activity because application performance changes over time.
    (*a*) space-dependent (*b*) space and time-dependent
    (*c*) time-dependent (*d*) None of these.
13. The default Http Modules sit in the request pipeline and …….. each and every request.
    (a) compile (*b*) debug.
    (*c*) intercept (*d*) None of these.

True or False

14. Caching in ASP.NET is a powerful feature that can decrease the performance of a Web application.
    (*a*) True (*b*) False
15. Performance profiling is a complicated process that requires developers to run through repeated iterations of profiling and optimization.
    (*a*) True (*b*) False

14.4 Summary

- We can enable debugging in ASP.NET page by writing the <%@Page debug= "True" %> page directive at the top of the page.
- The SDK debugger that comes along with .NET SDK can be found in the Guidebook directory.
- Just-In-Time debugging is another technique to debug a program that is started outside of Visual Studio. If we have enabled Just-In-Time debugging, we can view a dialog box when a crash occurs. Debugging a traditional ASP application generally involves placing Response.
- ASP.NET is a web application framework developed by Microsoft to allow programmers to build dynamic web sites and web applications. ASP.NET supports compiling applications in a special debug mode that facilitates developer troubleshooting.
- The ASP.NET 3.5 Cache API was a revolutionary feature that provided capabilities such as declarative output caching, programmatic output caching, and invalidation of cached items when the contents of an XML file or another cached item change.
- ASP.NET 3.5 also provides a new control named Substitution, which we can use to inject dynamic content in an otherwise cached Web page.
- The SqlDataSource control also has the EnableCaching property set to true, which results in the SqlDataSource automatically caching the data retrieved by the SelectCommand.

LOVELY PROFESSIONAL UNIVERSITY
Jalandhar-Delhi G.T. Road (NH-1)
Phagwara, Punjab (India)-144411
For Enquiry: +91-1824-521360
Fax.: +91-1824-506111
Email: odl@lpu.co.in

14.5 Keywords

Notes

Cache Keys: The cache key is the unique key that can be used to identify objects in the ICM server cache.

Pipeline: Computer processors can handle millions of instructions each second. Once one instruction is processed, the next one in line is processed, and so on. A pipeline allows multiple instructions to be processed at the same time.

Scalability Boot: Remove performance bottlenecks related to our data storage and databases and scale our .NET and Java applications to extreme transaction processing (XTP) with NCache.

SDK Debugger: This section details the software and hardware supported and required by the PayPal SDK, installation, and post-installation tasks. These samples can be installed in Microsoft Internet Information Server (IIS).

SqlDataSource Level: The SqlDataSource control enables us to use a Web server control to access data that is located in a relational database. This can include Microsoft SQL Server and Oracle databases.

Trump: Trump means to get the better of someone using a hidden resource.

Lab Exercise

1. Write C# program using SqlDataSource.
2. Write a C# program to create a data base connectivity.

14.6 Review Questions

1. What is the debugging in an ASP.NET application?
2. Briefly explain optimizing using caching and ASP.NET caching features.
3. What is the time-based cache invalidation in a SqlDataSource control?
4. What do you understand the downside of extensive caching?
5. Explain the optimizing via performance profiling.
6. Explain ASP.NET performance.
7. Define Asp.Net Process Configuration Optimization.
8. Give the concept of the asp.net pipeline optimization.
9. What is the process of modify the machine.config file?
10. How to modify the Web.config File?

Answers for Self Assessment Questions

1.	(*b*)	2.	(*c*)	3.	(*a*)	4.	(*d*)	5.	(*a*)
6.	(*b*)	7.	(*a*)	8.	(*c*)	9.	(*a*)	10.	(*b*)
11.	(*b*)	12.	(*c*)	13.	(*c*)	14.	(*b*)	15.	(*a*)

14.7 Further Readings

Books *Performance tuning and optimizing ASP.NET applications*, by Jeffrey Hasan, and Kenneth Tu

Online link http://www.wintrac.com/courses/dotnetTuning.asp