HOLY CROSS HOME SCIENCE COLLEGE, THOOTHUKUDI

B.Sc., - III YEAR

SMCS63- DOT NET TECHNOLOGIES

R.AME RAYAN,
ASSISTANT PROFESSOR,
DEPARTMENT OF COMPUTER SCIENCE

MSU/2016-17/UG-Colleges/Part-III (B.Sc. Computer Science)

Semester-V/ Ppr.no.31/ Core - 8

Dot NET Technologies

Unit I

The .NET Platform and the Web: The Web C;ient/Server Model – Components of ASP.NET and the .NET Framework – Overview of Internet Information Server – Overview of ASP.NET – The .NET Common Language Runtime and Class Library – Managed Components in .NET – Web Services – Language Independence in the .NET Framework – COM+ Component Services and .NET – Direction and plans for .NET. **The VB.NET:** What is VB.NET? – First VB application – Variables, Constants and Operators – Modularizing Code – Functions and Subroutines – Controlling Program Flow – Handling Errors and Exceptions – Object Oriented Programming – Multithread Programming.

Unit II

Working with ASP.NET: The features of ASP.NET – The Anatomy of ASP.NET Pages –Introducing Web Forms – VS.NET Web Applications and other IDE Basics – Separating Content and Code – the Code-Behind Feature – Application Configuration – Using HTML Forms – Using Web Controls – Web Controls for displaying and formatting data –Web Controls for creating buttons – Web control for inputting text – Web controls for selecting choices – Web controls for creating lists – Miscellaneous Basic Controls – Creating a simple ASP.NET Application – ASP.NET Page Directives – ASP.NET Rich Controls – Validation Controls – Data List Controls – User Controls - Saving state with the StateBag Object – ASP.NET Intrinsic Objects.

Unit III

Using the .NET Framework Class Library: Common Features of the .NET Framework Class Library – Using Data Collections – Handling File Input/output and Directories – Watching the File System for Changes – Using the Windows Event Log – Working with Active Directory Services – Using Message Queues – Communicating with Servers on the Internet – Manipulating XML Data – Sending Internet E-mail.

Unit IV

Building .NET Managed Components for COM+: The concept of Managed Code Execution – The Common Language Runtime – COM+ Component Services – Using VB.NET to develop Managed Components – Serviced Components – Building VB.NET Serviced Components. **Building Web Services:** The need for Web Services – Overview of Web Services – Web Service Description Language - Web Service Wire Formats – Web Services Discovery – Creating a simple Web Service – Calling Web Services with Proxy Classes – Creating a Client for a Web Service – Managing State in Web Services – Using Transactions in Web Services.

Unit V

Accessing Data with ADO.NET: Overview of Data Access on the Web – . [illegible] next generation of Data-Access Technology – ADO.NET Programming Objects and Architecture – Displaying Database Data – Programming with the DataList and DataGrid Controls – Working with the DataSet and DataTable Objects – Maintaining Data Integrity with the DataRelation Class – Using Manual Database Transactions – Working with Typed DataSet Objects. **Securing .NET Applications:** Windows Security – IIS Authentication and Authorization Security – A crash course in Cryptography – Implementing Data Encryption – ASD.NET Authentication Security.

Text Book

ASP.NET and VB.NET Web Programming –by Matt J. Crouch, Pearson.

Reference Books

1. Upgrading Microsoft Visual Basic 6.0 to .NET - by d Robinson, Michael Bond, Robert Ian Oliver, WP Publishers

2. Visual Basic.NET - by Shirish Chavan, Pearson

ASP.NET

Chapter 1 : The .Net Platform and the web

R.Ame Rayan, Holy Cross Home Science College, Thoothukudi.
02.07.2020

The web client/server model

AWeb Client : software that requests file, data and services.

Eg : Web browser

AWeb Server : software that fulfils the client's requests for content.

Eg : Microsoft Internet Information Server.

Protocols for Web Client/Server Communication

- The web client and web server sends information back and forth using the Transaction Control Protocol/ Internet Protocol (TCP/IP).
- This protocol is used in many internet services such as e-mail and the FTP
- Its computer language that is used by all the computers that are connected to the internet.
- A digital software conduit or socket is used to connect the computers using TCP/IP.
- A socket is like a line of plumbing established by either the client or the server software that is used to move packets of data back and forth between the client and the server.

- This channel is used to send the request and responses.
- Packet - a fragment of data that include the information about its origination and destination.
- TCP/IP handles the encapsulation of user data into these packets.
- Packets transmitted along sockets arrive at the destination in exactly the order in which it is transmitted.
- Web applications communicates over the socket using higher level protocol – HTTP

- HTTP – is a command response system used for communication on the web.
- Client issues the command and server sends response(data exchange)
- First Version – ten commands in the instruction set.
- The commands sent by the client are request for files(HTML documents and images).
- HTTP– stateless (when the client request is completed, the web server normally closes the socket connection with the browser and forgets the details of the exchange.

- The first version of HTTP (version 0.9) – implemented one client command – the GET method.
- The content was static text only (no multimedia).
- HTTP (version 1.0) – this not only provided commands to retrieve documents but allowed the client to send additional information to the server.
- HTTP (version 1.1) - ability to host more than one web host on the same physical server,
- Some information could include parameters to an external executable program and can send back information to the client.
- This method of encoding data to be sent from web client to web server is called the Common Gateway Interface(CGI)
- Applications and scripts that read CGI is called as CGI scripts or CGI applications

Server side processing with CGI programs

- CGI works by processing the data sent by the web browser.
- The data contained in the form fields are encoded by the browser using x-url encoding system before transmission.
- The data is sent by one of the two methods
- Encoding the data in the body of HTTP POST
- Passing the data as arguments to a script file in the URL.
- Web server receives data and passes it to a CGI application for processing.
- The CGI application then sends the results to the web browser as HTML

Disadvantage of Using CGI

- CGI provides plumbing functionality for web client/server communication, but CGI encoding is difficult.
- Comparison of ASP.NET & CGI

(1) CGI -writing CGI programs is the least cost effective method of web development.

CGI program needs extra code to interpret the encoding. After decoding then we have to process it.

ASP.NET – it is easy to parse the CGI data using built in code – designed for CGI decoding.

It provides a frame work for using many different programming languages(VB.NET,Jscript.NET) to process data.

- (2) CGI programs run separately form the web sever process in a separate application. So it steal the valuable CPU resources.
- ASP.NET - is designed is such a way that it uses minimal system resources by executing the process in the process space of the web server.

Components of ASP.NET & .NET Framework

- Internet Information server – core server software that delivers web content to the user's browser
- ASP.NET pages – HTML pages that host static content and executable code that dynamically generates content . It acts a glue between presentation layer(content) and Application layer(data). Web forms – build web application.
- Common Language Runtime & the .NET Framework class library – provide the runtime environment for managed code(code that executes in CLR). .NET Framework class library -set of components for developing applications with rich functionality.
- Managed components – enable modularity and allow sharing of these modules among many different applications.
- Web Services – component based functionality that can be accessed over the network by any machine using a variety of protocols.

Overview of Internet Information Server

- IIS – s/w for serving contents in web, delivery of web content via HTTP,FTP,SMTP.
- It is extendable using Internet Server Application Programmer’s Interface (ISAPI).
- ISAPI

ISAPI EXTENSIONS
ISAPI FILTERS

- ISAPI EXTENSIONS – receive the URL encoded from the data from HTTP POST & HTTP GET request.

- exists as windows dynamic libraries(compiled code modules which are called by other program during runtime)

- they are multithreaded

-they share the memory

ISAPI FILTERS

-exists as DLLs , can perform many operations at once and run in the memory space of IIS.

-interprets the requests from a web browser and send back the response.

1. The user request a URL– web browser sends appropriate commands to web server.
2. Web server sends to ISAPI filters
3. ISAPI filters process it and sends it to the web browser.

Application

1. Customized logging
2. Advance security system
3. On-the- fly decryption
4. Modification of o/p stream.

Overview of ASP.NET

- ASP.NET provides a framework for building web application.
- Its Uniqueness

(I) It is fully compiled. They use programming languages like VB.NET, C#, and Jscript.NET. This provides flexibility for the programmers during development of applications.

(II) It ships with reusable controls that make programming task very easy. Basic controls like button, text box, drop down list are included. In addition to that advance controls like calendar control and editable grid control is used.

WEB FORMS

- Web forms also programmers to build form-based web application designed around an event driven application mode.
- Its advantages

  (i) They can run in any browser

  (ii) any ,NET compatible language can be used to code ASP.NET application.

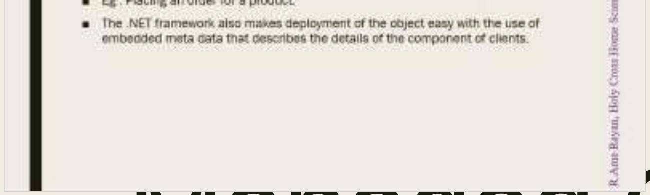
  (iii) web forms execute inside the CLR.

THE .NET COMMON LANGUAGE RUNTIME AND THE CLASS LIBRARY

- The .NET CLR controls the execution of the program.
- Any program written for .NET executes within this environment.
- Code that runs in CLR is known as managed code.
- This enables the programmer to write the code that is interoperable across different environment.
- It provides automatic memory management,
- It enable the language code to interact with other languages at object level.
- It also provides well defined objects.

.NET framework Class Library

- Extensive collection of data types and objects intended to give programmers a start to develop their applications.
- The data type and objects are designed and categorized to make access system functionality such as I/O,graphics ,database access and other system level service easy.
- It also supports basic data type and abstract data type,


- Eg : Placing an order for a product.
- The .NET framework also makes deployment of the object easy with the use of embedded meta data that describes the details of the component of clients.

R.Ame Rayan, Holy Cross Home Scie

Managed Components in .NET

- Acomponent is an independent unit of code that encapsulated a task or critical business rules or logic.
- Eg : Placing an order for a product.
- The .NET framework also makes deployment of the object easy with the use of embedded meta data that describes the details of the component of clients.

Web Services

- It uses HTTP as the network transport.
- It uses XML to facilitate the formatting of request and Reponses from web services.
- XML also provides a mechanism for describing the programming interface.

COM+ component services and .NET

- Manage automatic transactions
- Use declarative role-based security
- Make components use object pooling
- Create queued components

Chapter 2 : The VB.NET Crash Course

Handled By : R.Ame Rayan

Arrays

Definition :

Arrays are collection of data elements that all have the same data type.
Arrays have dimensionality which refers to the number of subscription to access each element in the array.
All arrays have at least one dimension and it can support upto sixty dimensions.
Arrays also have defined minimum and maximum for the subscriptions used to reference the array elements.
Array dimensions usually begin with zero.

It have a maximum value of what an integer data type (Int32) can hold.
Syntax

```
  Dim a(size) as data type
```

Eg:

```
 Dim a(5) as string
 Dim s(10,10) as boolean
```

ReDim:

The redim statement is used to change the size of an array after it has been declared using Dim

Eg:

ReDim ar(15) as string

Note :

when ReDim is used all the data contained in the array are lost however to retain the contents of an array Preserve keyword is used after the ReDim command

Eg:

ReDim Preserve ar(24) as string

Note: This Preserve keyword is used on the last dimension of the array.

Ubound() function : It is used to fund the size of an array.

Asize = Ubound(ar)

To initialize the contents of an array at the time of array declaration

```
Dim a() as string={“ff”,”aa”,”bb”}
Dim a() as Integer = {1,2,3}
```

Converting data types

variables can be converted form one data type to another.VB.NET performs implicit conversion.

To convert integer data type to Long data type.

```
Dim intA as integer = 3

Dim intB as Long
intB = intA
```

Now the variable intB contains the value “3” stored in a long data type.
This conversion is safe conversion.
Explicit conversion can also be used to convert data types that involves special VB.NET functions to convert one data type to another.

Using Constants:

the value of the constants do not change throughout the execution of the programme..
VB.NET provides a number of predefined constants.

Syntax
Const mc=value as typename
Eg :
Const s as string =“Dot”

Date and time constants are enclosed within pound signs (#)
Eg:
Const a as Date =#6/4/2020 3:30 AM#

Arithmetic Operators:

Syntax
variable = numeric expression1+ numeric expression2

^- exponent * - Multiplication /- Divides and returns a floating point result
\ = divides and returns an integer result Mod – divides and returns only the reminder + - sum of two numbers - difference of two numbers.
All are binary operator but – is alone can be used for unary operator.

Assignment operator are used to assign a variable.
Eg:
M= 3

Assignment operators

= - to assign value
^=. *=, /-,\=, +=, &=(concatenate two strings)

Bitwise operators
And,Or,Not, Xor

BitAnd
BitOr
BitNot
BitXor

Logical operators
And,or,not,Xor

Comparison
Operator

Pattern matching characters in VB.NET
? – matches any single character

Eg: “Cow” like “C?w”

*- matches zero or more characters

Eg: “matt” like “m*”

[charlist] – matches any single character in a character list

Eg: “A” like”[A=Z]”

[! charlist] – matches any single character not in a character list

Eg: “A” like”[!A=Z]”

Modularizing your code – functions and subroutines.

Using functions

function is a group of code statements that are executed and give a result back to the code that invoked the function.

```
Function fname() as integer
        'your code
End function

Function area(Byval b as double, byval h as double) as double
        a= 0.5*b*h
End function
```

To invoke a function

```
Dim a as double
A= area(3,8)
Console.Writeline(a)
```

ByVal and ByRef- (used to return more than one value)

Using Subroutines

Subroutines are used when no specific value is needed for executing blocks of code when no specific value is needed as output.

```
Sub area()
  'code
End Sub
```

Controlling Programming flow

1. Conditional processing

   processing tests on expression and executes code based on the value of the expression

If …then …else statements

```
If expression then
   statement1
Elseif (expression2)
  statement2
Else
statement3
End if
Eg:
Dim a as integer =5
If a >0 and a<=10 then
console.writeline("it is within the range",a)
Else
  console.writeline("it is out of the range",a)
End if
```

Select case statement

 Tocheck the value of an expression against several values and execute the code based on each of those choices.

```
Select case test-exp
        case exp
                ‘statement
        case exp1
                ‘statement1
        .
        .
        case else
                ‘statement
end select
```

```
Dim s as string =“dot”

Select case s
        case “dot”
                console.writeline(“dot”)
        case “net”
                 console.writeline(“net”)
        case “Vb.net”
                 console.writeline(“vb.net”)
        case else
                 console.writeline(“who are you”)

end select
```

Flow Control Statements

Definition

This enables to branch to different areas in the code unconditionally.

1. Exit statement:
   - ❖ Exit is used to leave a function, subroutine or loop.
   - ❖ It takes the following keywords : Do, For,Sub, Function,Property,Try.

```
Eg:
  Sub cp(Byval p as string)
     if p <> "123" then
         Exit sub
     else
        console.writeline("Welcome")
     End if
  End sub
```

Goto Statements:

It transfers execution to a labeled line of code.

```
Sub eg()
         console.writeline(“Welcome”)
         goto h
         console.writeline(“first”)
 h :
        console.writeline(“Second”)
         End if
    End sub
```

Loops

Definition :

Loops execute a block of code a predetermined number of time or until a particular test condition bus been met.

Do ……..Loop loops

It can be executed in two ways :

(i) it executes the statement until test expression is true (Do until). This type of loop is called as top driven loop.

Eg:

```
Do While i >30
  console.writeline(i)
loop
```

(ii)it executes the statement as long as test expression is true (Do while). This type of loop is called as bottom driven loop.

Eg:

```
Do
  console.writeline(i)
Loop While i >30
```

For …..Next

Definition :

the for…next loop is used to execute a group of statements a specified number of times. It has a counter variable that increments or decrements with each pass through the loop. The statements inside the loop are executed until the designated value of the counter is reached.

```
Sub demo()

Dim i as integer

For I = 1 to 5
    console.wrtiteline(“Hello”)
next
End sub
```

•There is a optional keyword for the For…Next statement called step, which controls how the counter variable is incremented or decremented..

•If no counter variable is specified then VB.NET assumes that it should be incremented by 1.

•By using the step clause, the loop is controlled in both in direction and the amount the

Eg:

```
Sub demo()

Dim i as integer

For I = 1 to 10 step2
     console.write(i)
next
End sub
```

o/p

```
0    2    4    6    8    10
```

```
Dim i as integer

For I = 10 to 0 step-1
     console.write(i)
next
End sub
```

o/p

```
10   9   8   7   6   5  4  3  2   1   0
```

For Each ….Next Loops

Definition :

this is used to step or count through a collection of items such as the elements in an array.

```
Dim a() as string("Red","Blue","orange","green"}

For Each I in a
   console.writeline(a)
Next
```

While ….End While Loops

Definition:

it is also a top driven loop. The set of statements are executed until the given test conditions are satisfied.

```
Eg:
 while demo(i<4)
   console.writeline(i)

End while
```

Handling Errors and Exceptions

Error are noticed during the execution of the program.

There are two types of error handling

(1) unstructured error handling.

(2) structured exception handling.

Unstructured error handling:

this is a simple type of error handling where a function is called and the return value if checked. If the return value corresponds to a defined error result code, then an error message is displayed.

VB.NET provides a statement called on error Goto for handling errors.

Eg:

```
Sub demo()
On Error Goto Err_des

Dim a as integer = 1
Dim b as integer = 2
Dim c as integer = 0
Dim r as integer

  r = b/a
r=a/c
Done :
    exit sub
Err_des:
    console.writeline(Err.Description)
    Goto Done
End sub
```

Eg:

```
Dim i as integer = 1000

Try
   if i >=1000 then
       Throw New Exception(" x is out of range")
   End if
Catch
    console.writeline("Error!)
Finally
    console.writeline("excecuting finally block")
End Try.
```

Structured Exception Handling

•Astructured error handler begins with Try keyword known as Try block.

•The Try block is followed by one or two catch blocks.

•Each catch has an associated exception type and its declaration follows the Catch keyword.

•The error handling code is placed within the catch block.

Eg: DivideByZeroException

•The error handling code includes the display of error message, resetting the variables , exiting the function etc.,

•After all the exceptions have been tested and caught, code in the Finally block is executed.

•The code if the Finally block can contain a success message or another terminating message.

•VB.NET provides the ability to throw an exception.

•

ASP.NET

Chapter 3 : WORKING WITH ASP.NET

Unit - II

R.Ame Rayan, Holy Cross Home Science College, Thoothukudi.
10.08.2020

THE FEATURES OF ASP.NET

1. Compactivity with any .Net languages

The primary design goals ASP.NET was to make web applications as rich as features as their desktop counterparts.

2. Compiled Code

When the user request a web page with ASP.NET code, ASP.NET compiles the code embedded in that web page. This not only provides great speed enhancement but allows other features such as strong variable typing and early binding of objects. ASP.NET can cache data for faster delivery through special API's.

3. Easy Application Development

Many web applications are tightly integrated with the web server since it makes the web deployment and maintenance of the application difficult. ASP.NET solves the problem by swapping out new components and web pages for pending request.

4. Simple Application configuration

ASP.NET store the application setting in human readable XML configuration files. These XML files are kept in the common directories making the administrators easy to locate. New settings can be added to the XML file and the settings are stored in hierarchical fashion for easy reference.

Advanced session and Application state management

ASP.NET had several offering for maintaining application statem and some are even compatible with older versions of ASP. ASP.NET offers advanced state management that improves on the legacy methods of state management.

Integrated security features :

ASP.NET provides various security measures. Users are authenticated using traditional methods provided by IIS like BASIC and Windows and newer methods like Microsoft passport. Protecting site files can be done by assigning user and group access to the individual files and directories.

Direct Low-Level HTTP support :

Modern web applications usually consist of HTML and executable codes integrated within the HTML. Web programs are sometimes supplemented by low level programs that interact with the web server through tight integration.

Backward Compatibility with ASP

Applications written in classic ASP can run along side ASP.NET applications without any ill effects.

THE ANATOMY OF ASP.NET PAGES

The code structure of ASP.NET

* it begins with the plain text file that has .aspx extension.

- The contents in the .aspx file are called the pages.
- ASP.NET has the ability to store application code in a separate file rather than grouping it with the HTML. This concept is called code behind.
- when a ASP.NET class is compiled , a new class that derives from the page class is automatically generated.
- The .aspx file contains three new items
  - (i) ASP.NET page directives
  - (ii) the runat = “server” attribute
  - (iii) web controls.

EXECUTION STAGES AND STATE MANAGEMENT

The web applications use request-response model of communication : the web browser request a document from the web server and the web server responds with the data from that document.

The web application also contain HTML forms, which allow for user input.

The web browser must package up the data entered in the input form before sending it to the server.

- The data in the form can then be used as input for an ASP.NET program,
- The program is executed using the form data as input, and the results are returned to the browser as HTML. This sequence of steps is referred to as round – trip.
- This round trip is required when the web application expects a meaningful response to a query issued from an HTML form.
- When needed the form provides feedback for the input.
- This the data validation that is done in the client side scripts.
- Data validation include checking for form completion, numerical ranges and correct data formatting
- Functions that are searching a data from the database require a round – tri to the server as the data is found in the server.
- The web page must be regenerated for each round – trip.
- The ASP.NET web controls and HTML controls can maintain their associated values between round – trips without any coding by the user. This is called as view state.
- View state is maintained using the _VIEWSTATE hidden form field, which is automatically generated by ASP.NET for every .aspx file that contains a server-side form.
- The encoded text string contains information about the state (values) of the server-side controls on the form.
- When the form is posted back to the server the _VIEWSTATE will be posted along with the other form values.
- Form values that haven’t been changed by the user has the values encoded from the previous post.

THE EVENT MODEL FOR THE PAGE CLASS

The Page class contains events that fire when
(i) the page loads into browser
(ii)when the server code execution ends and page rendering has finished,
These events are Page Load and Page Unload respectively.

The page load event is fired when the page and all its control are rendered.
The page unload event is fired when the page is about to be discarded.

INTRODUCING WEB FORMS

Web forms are used to provide dynamic web pages with ASP.NET.

Web forms are used to write program in an object oriented manner.

Advantages :
(i) They provide event based programming model. Eg: click event in a button.
(ii) It allows a separation of application logic from content.
(iii)Its Integrated Development environment provide a easy design time environment.
(iv) The benefits of Rapid Application Development (RAD) combines event based programming with form design tools.

VS.NET WEB APPLICATIONS AND OTHER IDE BASICS

- VS.NET offers an easy way to create new ASP.NET web application projects.
- File->new ->projects from the menu bar and clicking the ASP.NET web application button.
- To add new web form to the application select add - >add web form.
- Then select the web form icon in the template section of the screen and give your new web form a file name.
- The new web form is saved and become a part of the current solution.
- Whenever the user wants to view the code or when the user wants to place a web control on the web form for the first time, a new code module is added to the solution that serves as the code module for the web form.
- The code – behind module defines a new class for the web form and created plumbing code.

SEPARTING CONTENT AND CODE – THE CODE BEHIND FEATURE

- CODE behind came about when design and organizational issues arose from mixing application code with HTML.
- Many programmers combined the two elements within the same file.
- The result was a code base that was difficult to maintain and understand due to the mixture of HTML and script.
- The code – behind feature removes the problem by completely separating HTML from the program code; they are in separate files.
- The two files are linked together using a special page directive, Codebehind, which is defined in the .aspx

file.
- Codebehind, specifies the pathname to a code module file.
- If the pathname is omitted, the class is assumed to be in the /bin directory of the web application's virtual web directory.

```
 <% page language = “vb”
      AutoEventsWireup = “False”
      Codebehind=“Sample.vb”
      Inherits = “WebApplication1, Sample”
%>
```

- The src attribute , tells ASP.NET where to the class file is located.
- If this attribute is omitted, ASP.NET assumes that the class is located in the /bin directory of the web application.

```
 <% page language = “vb”
       AutoEventsWireup = “False”
       Codebehind=“Sample.vb”
       Inherits = “WebApplication1, Sample”
%>
```

APPLICATION CONFIGURATION

- Applications in ASP.NET are comprised of a virtual web and pages, files and assemblies contained within the directory and subdirectories.

Structures and Configuration of the Global.asax file.

- **There** is exact one Global.asax file per application and it is optional.
- The file is located in the root level of the virtual directory of the ASP.NET web application.
- This file is like the other web application directory that the user may request.
- The Global.asax file may contain code that compromise the security of the web server.
- When VS.NET creates a new web application, a Global.asax file is created automatically.
- The Global.asax file contains a code-behind file named Global.asax.vb

- The Global.asax.vb file defines a class called Global, which inherits from System.Web.HttpApplication. This class defines all the methods, properties and events that all application objects use within an ASP.NET application.

- VS.NET overrides the event of Http Application class.

- Those events are associated with the beginning and ending of the user sessions and ASP.NET application.

USING HTML CONTROLS

Most controls in ASP.NET is server based controls.
These are server side code in a web pages.
The code outputs standard HTML 3.2 for compatibility with all web browsers.
ASP.NET comes in two varieties (i) HTML controls and (ii)web controls
HTML controls correspond to standard HTML 3.2 elements.

HTML Controls can be referenced using object.member.notation
HTML controls must contain an ID attribute. This is used to identify the control.

(i) HTMLForm control

Creates a server-side control that maps to the **<form>** HTML element and allows you to create a container for elements in a Web page.
The method attribute is set to POST.ASP.NET, so that the values are posted back to the server in a hidden form variable for control state management.

```
<form id="programmaticID"
    method=POST | GET
    action="srcpageURL"
    runat="server" >
  Other controls, input forms, and so on.
</form>
```

(ii) The HTMLAnchor control

Creates a server-side control that maps to the **<a>** HTML element and allows you link to another Web page.
The href, name, targer and title properties can be set at runtime.
The OnServerClick procedure is fired when the anchor link is clicked.

```
<a id="programmaticID"
   href="linkurl"
   name="bookmarkname"
   OnServerClick="onserverclickhandler"
   target="linkedcontentframeorwindow"
   title="titledisplayedbybrowser"
   runat="server" >
linktext
</a>
```

To display the new Web page by using the **Target** property. **Target** values must begin with a letter in the range from a to z (case insensitive), except the following special values that begin with an underscore: _**blank**, _**self**, _**parent**, and _**top**.

The HTML title attribute text serves as the floating tool tip test for the anchor.

(iii) **The HTMLButton control**

Creates a server-side control that maps to the **<button>** HTML element and allows you create push buttons.

```
<button
       id="programmaticID"
       OnServerClick="onserverclickhandler"
       runat="server" >
   buttontext, image, or control
</button>
```

Cascading style sheets (CSS) styles changes the appearance of the button.

```
<button runat="server">
  style="font: 8pt verdana;
background-color:lightgreen;
border-color:black;
height=30;
width:100"
</button>
```

(iv) **The HTMLGenericControl control**

Some controls don’t have the visual representation by themselves.
The HTML tags <span>, <div>,<body> and <font> are the examples of this.

```
<span | body | div | font | others
      id="programmaticID"
      runat="server" >
   Content between tags
</span | body | div | font | others>
```

(v) **HTMLImage control**

Creates a server-side control that maps to the **<img>** HTML element and allows you to display an image.

```
<img id="programmaticID"
     alt="alttext"
     align= top | middle | bottom | left | right
     border="borderwidth"
     height="imageheight"
     src="imageURL"
     width="imagewidth"
```

This control allows you to dynamically set and retrieve the image's source, width, height, border width, alternate text, and alignment by using the **Src**, **Width**, **Height**, **Border**, **Alt**, and **Align** properties, respectively.

Note This control does not require a closing tag.

(vi) The HTMLInputButton control

Creates a server-side control that maps to the **<input type=button>**, **<input type=submit>**, and **<input type=reset>** HTML elements and allows you to create a command button, submit button, or reset button, respectively.

When a user clicks an **HtmlInputButton** control, input from the form that the control is embedded on is posted to the server and processed. A response is then sent back to the requesting browser.
By providing a custom event handler for the ServerClick event, you can perform a specific set of instructions when the control is clicked.

Note A **reset** button does not support the **ServerClick** event.
When a **reset** button is clicked, all input controls on the page are not necessarily cleared.
Instead, they are returned to their original state when the page was loaded.

For example, if a text box originally contained the value "JohnDoe",
clicking on the **reset** button would return the text box to this value.

When used in conjunction with the **HtmlInputText** and **HtmlTextArea** controls,
you can create user input or authentication pages that can be processed on the server.

Note This control does not require a closing tag.

(vii) The HTMLInputCheckBox Contol

Creates a server-side control that maps to the **<input type=checkbox>** HTML element and allows you to a create check box control that lets the user select a **true** or **false** state.

```
<input type=checkbox
      id="programmaticID"
      checked
      runat="server" >
```

The **HtmlInputCheckBox** control does not post back to the server when it is clicked. The state of the check box is sent to the server for processing when you use a control that posts back the server, such as the **HtmlInputButton** control. To determine whether the check box is selected, test the **Checked** property of the control.

Note This control does not require a closing tag.

(viii) The HTMLInputFiile Control

Creates a server-side control that maps to the **<input type=file>** HTML element and allows you to upload a file to the server.

The three step process for uploading a file is

1. The control allow the user to select a file to upload using a file browsing dialog box
2. Once the user selects the file, it is encoded for transmission and sent along with other form data with the server post back.
3. The file is saved to disk on the server

```
<input type=file
      id="programmaticID"
      accept="MIMEencodings"
      maxlength="maxfilepathlength"
      size="widthoffilepathtextbox"
      postedfile="uploadedfile"
      runat="server" >
```

The **HtmlInputFile** control can be used to easily design a page that allows users to upload binary or text files from a browser to a directory that you designate on your Web server.
File upload is enabled in all HTML 3.2 and later Web browsers.

Note   You must set the **enctype** attribute of the form to "multipart/form-data".

Web controls for displaying and formatting data

(i) Label Control

Displays static text on a Web Forms page and allows you to manipulate it programmatically.

```
<asp:Label id="Label1"
     Text="label"
     runat="server"/>
or
<asp:Label id="Label1"
     runat="server">
   Text
</asp:Label>
```

Id – to reference the control programmatically.
When the label control onto a web web form, the source code is automatically in the .aspx.

First syntax – text attribute that specifies the text to display.
Second syntax – it has the closing tag and any text text can be placed inside the tags.

ADJUSTING THE SHARED PROPERTIES FOR WEB CONTROLS

Web control properties are from te system.drawing and system.web.UI.webcontrols namespaces.

Eg:
 borderstyle, unit.

Panel control :
  web pages can contain many different elements and controls and they can be contained in one group. Provides a container for other controls. This control is rendered as an HTML **<div>** element.

```
<asp:Panel id="Panel1"
     BackImageUrl="url"
     HorizontalAlign="Center|Justify|Left|NotSet|Right"
     Wrap="True|False"
     runat="server">

   (Other controls declared here)

</asp:Panel>
```

The **Panel** control is a container for other controls. It is especially useful for generating controls programmatically and displaying and hiding groups of controls.

You can display an image in the background of the **Panel** control by setting the **BackImageUrl** property.

Using the **HorizontalAlignment** property, you can specify the horizontal alignment of the items contained in the control.

The **Wrap** property lets you determine whether items in the control automatically continue on the next line when a line is longer than the width of the panel.

(iii) **TABLE, TABLEROW AND TABLE CELL CONTROLS**

Declares a table and allows you to manipulate it programmatically.

```
<asp:Table id="Table1"
     BackImageUrl="url"
     CellSpacing="cellspacing"
     CellPadding="cellpadding"
     GridLines="None|Horizontal|Vertical|Both"
     HorizontalAlign="Center|Justify|Left|NotSet|Right"
     runat="server">

   <asp:TableRow>

      <asp:TableCell>
         Cell text
      </asp:TableCell>

   </asp:TableRow>

</asp:Table>
```

The **Table** class allows you to build an HTML table and specify its characteristics. A table can be built at design time with static content, but the **Table** control is often built programmatically with dynamic contents.

Each **Table** control is made up of rows (represented by instances of the **TableRow** class) stored in the **Rows** collection of the control.

Each row is made up of cells (represented by instances of the **TableCell** class) stored in the **Cells** collection of the each **TableRow**.

To display an image in the background of the **Table** control by setting the **BackImageUrl** property.
By default, the horizontal alignment of the items in the table is not set. If you want to specify the horizontal alignment, set the **HorizontalAlignment** property.

The spacing between individual cells is controlled by the **CellSpacing** property. You can specify the amount of space between the contents of a cell and the cell's border by setting the **CellPadding** property.

To display the cell borders, set the **GridLines** property. You can display the horizontal lines, vertical lines, or both horizontal and vertical lines.

WEB CONTROLS FOR CREATING BUTTONS

All the buttons in ASP.NET cause a post-back to the server.
When the server needs to respond to the user click, use the button controls.

The Button Control

The button control renders the regular push button, the most common type of button seen is the HTML button.

```
<asp:Button id="MyButton"
    Text="label"
    Command="command"
    CommandArgument="commandargument"
    CausesValidation="true | false"
    OnClick="OnClickMethod"
    runat="server"/>
```

Unique properties:

- The text properties specifies the text ot display on the button itself, such as “Submit Form”.
- The “On-click” property sets the procedure name to call when the button is clicked.
- Command – The event fired by the child control event can be bubbled up to the parent the message contained in the command property.
- Command Arguments : an optional command parameter is passed in the CommandArgument property.

ImageButton Control :

- ImageButton Control allows to use graphical image as a button for causing a post-back.
- It is used to create clickable icons as well as image maps.

```
<asp:ImageButton id="ImageButton1"
    ImageUrl="string"
    Command="Command"
    CommandArgument="CommandArgument"
    CausesValidation="true | false"
    OnClick="OnClickMethod"
    runat="server"/>
```

- The **ImageButton** control is used to display an image that responds to mouse clicks.
- Specify the image to display in the control by setting the **ImageUrl** property.
- Both the **Click** and **Command** events are raised when the **ImageButton** control is clicked.

The LinkButton Control

The LinkButton Control appears like a hyperlink to the user, for navigating to another URL. The LinkButton control can cause a post-back to the server just like the Button and ImageButton controls.

```
<asp:LinkButton id="LinkButton1"
    Text="label"
    Command="Command"
    CommandArgument="CommandArgument"
    CausesValidation="true | false"
    OnClick="OnClickMethod"
    runat="server"/>
or
<asp:LinkButton id="LinkButton1"
    Command="Command"
    CommandArgument="CommandArgument"
    CausesValidation="true | false"
    OnClick="OnClickMethod"
    runat="server"/>
  Text
</asp:LinkButton>
```

Specify the text to display in the **LinkButton** control by either setting the **Text** property or placing the text between the opening and closing tags of the **LinkButton** control.

Web Controls for Inputting Text

The TextBox Control

The TextBox control's actual rendering is determined by the TextMode Property.

```
<asp:TextBox id="value"
    AutoPostBack="True|False"
    Columns="characters"
    MaxLength="characters"
    Rows="rows"
    Text="text"
    TextMode="SingleLine | MultiLine | Password"
    Wrap="True|False"
    OnTextChanged="OnTextChangedMethod"
    runat="server"/>
```

- The **TextMode** property is set to **SingleLine**, which creates a text box with only one line.
- Set the property to **MultiLine** or **Password**. **MultiLine** creates a text box with more than one line.

- **Password** creates a single-line text box that masks the value entered by the user.
- The display width of the text box is determined by its **Columns** property.
- If the text box is a multiline text box, the display height is determined by the **Rows** property.
- The number of characters that is entered can be limited in the control by setting the **MaxLength** property.
- Set the **Wrap** property to **true** to specify that the contents of the cell should automatically continue on the next line when the end of the text box is reached.

The Textbox responds through an OnTextChanged event when the content is changed.
OnTextChanged – causes post back to the server.

Web Controls for selecting Choices

Selection control includes Radio Button and CheckBoxes, which allow users to select from a series of predefined values.
These selection controls are grouped together to represent choices in the same property.
Radio Buttons, like checkboxes, allow users to select from the list of options, one choice in a category is selected.

The CheckBox Control

Each checkbox can have a label, which appear to the right or left of the checkbox, according to the TextAlign property.

The default state of the checkbox and its current state are indicated by the Checked property.

```
<asp:CheckBox id="CheckBox1"
    AutoPostBack="True|False"
    Text="Label"
    TextAlign="Right|Left"
    Checked="True|False"
    OnCheckedChanged="OnCheckedChangedMethod"
    runat="server"/>
```

The checkbox respond to the change made to itself.
A change event can be raised and handled by an event-handling procedure passed into OnCheckedChanged.

RadioButton control

It works same as the checkbox control.
Creates an individual radio button on the page.
Multiple radio buttons can be grouped together to provide a mutually exclusive set of choices.

```
<asp:RadioButton id="RadioButton1"
    AutoPostBack="True|False"
    Checked="True|False"
    GroupName="GroupName"
    Text="label"
    TextAlign="Right|Left"
    OnCheckedChanged="OnCheckedChangedMethod"
    runat="server"/>
```

The GroupName property to assign a name to a collection of radio buttons.

The CheckList and RadioButtonList Controls:

These controls are used when web controls are generated dynamically from the data source. The data source may contain fairly large amount of data.

```
<asp:CheckBoxList id="CheckBoxList1"
     AutoPostBack="True|False"
     CellPadding="Pixels"
     DataSource='<% databindingexpression %>'
     DataTextField="DataSourceField"
     DataValueField="DataSourceField"
     RepeatColumns="ColumnCount"
     RepeatDirection="Vertical|Horizontal"
     RepeatLayout="Flow|Table"
     TextAlign="Right|Left"

OnSelectedIndexChanged="OnSelectedIndexChangedMethod
"
     runat="server">

   <asp:ListItem value="value"
        selected="True|False">
      Text
   </asp:ListItem>

</asp:CheckBoxList>
```

Properties of CheckBoxList and RadioButton Controls

Checkbox List:

Creates a multiselection check box group. This control supports binding to a data source.

```
<asp:CheckBoxList id="CheckBoxList1"
     AutoPostBack="True|False"
     CellPadding="Pixels"
     DataSource='<% databindingexpression %>'
     DataTextField="DataSourceField"
     DataValueField="DataSourceField"
     RepeatColumns="ColumnCount"
     RepeatDirection="Vertical|Horizontal"
     RepeatLayout="Flow|Table"
     TextAlign="Right|Left"

OnSelectedIndexChanged="OnSelectedIndexChangedMethod"
     runat="server">

   <asp:ListItem value="value"
      selected="True|False">
    Text
   </asp:ListItem>

</asp:CheckBoxList>
```

The **CheckBoxList** control creates a multiselection check box group that can be dynamically generated using data binding.

To specify items that you want to appear in the **CheckBoxList** control, place a **ListItem** element for each entry between the opening and closing tags of the **CheckBoxList** control.

The **CheckBoxList** control also supports data binding. To bind the control to a data source, first create a data source, such as a **System.Collections.ArrayList** object, that contains the items to display in the control.

Next, use the **Control.DataBind** method to bind the data source to the **CheckBoxList** control.

Use the **DataTextField** and **DataValueField** properties to specify which field in the data source to bind to the **Text** and **Value** properties of each list item in the control, respectively.

The **CheckBoxList** control will now display the information from the data source.

To determine the selected items in the **CheckBoxList** control, iterate through the Items collection and test the Selected property of each item in the collection.

You can specify the way the list is displayed by using the RepeatLayout and RepeatDirection properties.

If **RepeatLayout** is set to **RepeatLayout.Table** (the default setting), the list is rendered within a table.

If it is set to **RepeatLayout.Flow**, the list is rendered without any table structure.

By default, **RepeatDirection** is set to **RepeatDirection.Vertical**.

Setting this property to **RepeatDirection.Horizontal** renders the list horizontally.

Radiobuttonlist Control

Creates a group of radio buttons. This control supports binding to a data source.

```
<asp:RadioButtonList id="RadioButtonList1"
    AutoPostBack="True|False"
    CellPadding="Pixels"
    DataSource="<% databindingexpression %>"
    DataTextField="DataSourceField"
    DataValueField="DataSourceField"
    RepeatColumns="ColumnCount"
    RepeatDirection="Vertical|Horizontal"
    RepeatLayout="Flow|Table"
    TextAlign="Right|Left"
    OnSelectedIndexChanged="OnSelectedIndexChangedMethod"
    runat="server">

  <asp:ListItem Text="label"
      Value="value"
      Selected="True|False" />

</asp:RadioButtonList>
```

The **RadioButtonList** control allows you to create a single-selection radio button group that can be dynamically generated by binding to a data source. To specify the items that you want to appear in the **RadioButtonList** control, place a **ListItem** element for each entry between the opening and closing tags of the **RadioButtonList** control.

The **RadioButtonList** control also supports data binding. To bind the control to a data source, first create a data source, such as a **System.Collections.ArrayList** object, that contains the items to display in the control. Next, use the **Control.DataBind** method to bind the data source to the **RadioButtonList** control.

Use the **DataTextField** and **DataValueField** properties to specify which field in the data source to bind to the **Text** and **Value** properties, respectively, of each list item in the control. The **RadioButtonList** control will now display the information from the data source.

To determine the selected items in the **RadioButtonList** control, iterate through the **Items** collection and test the **Selected** property of each item in the collection.

You can specify the rendering of the list with the **RepeatLayout** and **RepeatDirection** properties. If **RepeatLayout** is set to **RepeatLayout.Table** (the default setting), the list will be rendered within a table. If it is set to **RepeatLayout.Flow**, the list will be rendered without any tabular structure. By default, **RepeatDirection** is set to **RepeatDirection.Vertical**. Setting this property to **RepeatDirection.Horizontal** will render the list horizontally.

ListBox Control

Creates a single-selection or multiselection list box.

```
<asp:ListBox id="Listbox1"
    DataSource="<% databindingexpression %>"
    DataTextField="DataSourceField"
    DataValueField="DataSourceField"
    AutoPostBack="True|False"
    Rows="rowcount"
    SelectionMode="Single|Multiple"
    OnSelectedIndexChanged="OnSelectedIndexChangedMethod"
    runat="server">

  <asp:ListItem value="value" selected="True|False">
    Text
  </asp:ListItem>

</asp:ListBox>
```

Use the **ListBox** control to create a list control that allows single or multiple item selection. Use the **Rows** property to specify the height of the control. To enable multiple item selection, set the **SelectionMode** property to **ListSelectionMode.Multiple**.

To specify the items that you want to appear in the **ListBox** control, place a **ListItem** element for each entry between the opening and closing tags of the **ListBox** control.

The **ListBox** control also supports data binding. To bind the control to a data source, first create a data source, such as a **System.Collections.ArrayList**, that contains the items to display in the control. Next, use the **Control.DataBind** method to bind the data source to the **ListBox** control.

Use the **DataTextField** and **DataValueField** properties to specify which field in the data source to bind to the **Text** and **Value** properties, respectively, of each list item in the control. The **ListBox** control will now display the information from the data source.

If the **SelectionMode** property is set to **ListSelectionMode.Multiple**, determine the selected items in the **ListBox** control by iterating through the **Items** collection and testing the **Selected** property of each item in the collection.

If the **SelectionMode** property is set to **ListSelectionMode.Single**, you can use the **SelectedIndex** property to determine the index of the selected item.

The index can then be used to retrieve the item from the **Items** collection.

Image Control

Displays a Web-compatible image on the Web Forms page.

```
<asp:Image id="Image1" runat="server"
     ImageUrl="string"
     AlternateText="string"
     ImageAlign="NotSet|AbsBottom|AbsMiddle|BaseLine|
                Bottom|Left|Middle|Right|TextTop|Top"/>
```

Use the **Image** control to display an image on the Web Forms page.

Setting the **ImageUrl** property specifies the path to the displayed image. You can specify the text to display in place of the image when the image is not available by setting the **AlternateText** property.

The **ImageAlign** property specifies the alignment of the image in relation to other elements on the Web Forms page.

ASP .NET PAGE DIRECTIVES

Page Directives Control settings for the code compiler like page settings.
There are Six types of page directives

(i) The @ Page and Control Directives

The Page directive defines the attributes specific to the page file for the page parser and the compiler.

The basic syntax of Page directive is:

```
<%@ Page Language="C#" AutoEventWireup="true"
 CodeFile="Default.aspx.cs"             Inherits="_Default"
Trace="true" %>
```

The attributes of the Page directive are:

Attributes	Description
AutoEventWireup	The Boolean value that enables or disables page events that are being automatically bound to methods; for example, Page_Load.
Buffer	The Boolean value that enables or disables HTTP response buffering.
ClassName	The class name for the page.
ClientTarget	The browser for which the server controls should render content.
CodeFile	The name of the code behind file.
Debug	The Boolean value that enables or disables compilation with debug symbols.
Description	The text description of the page, ignored by the parser.
EnableSessionState	It enables, disables, or makes session state read-only.
EnableViewState	The Boolean value that enables or disables view state across page requests.
ErrorPage	URL for redirection if an unhandled page exception occurs.
Inherits	The name of the code behind or other class.
Language	The programming language for code.
Src	The file name of the code behind class.
Trace	It enables or disables tracing.
TraceMode	It indicates how trace messages are displayed, and sorted by time or category.
Transaction	It indicates if transactions are supported.
ValidateRequest	The Boolean value that indicates whether all input data is validated against a hardcoded list of values

The @ import Directive

- To add a reference to a namespace. To import multiple namespace

- <%@ import namespace =“System.Drawing”>
- <%@ import namespace =“System.Net.Sockets”>

All the pages automatically includes

System.Collections, System.IO,System.Web,System.Web.UI, Web,System.Web.UI.HTMLControls, Web,System.Web.UI.WebControls

The @Register Directive

- Allows to refer the namespaces and classes by an alias.
- <% Register Tagprefix =“tp” Namespace=“ns”> %>
- <% Register Tagprefix =“tp” tagname=“ns” src=“pathname”> %>
- Tagprefix – sets alias for a namespace
- Tagname – sets alias for a class
- Namespace –namespace associated with tag prefix

@Assembly

The Assembly directive links an assembly to the page or the application at parse time.

This could appear either in the global.asax file for application-wide linking, in the page file, a user control file for linking to a page or user control.
The basic syntax of Assembly directive is:

```
<%@ Assembly Name ="myassembly" %>
```

The attributes of the Assembly directive are:

Attributes	Description
Name	The name of the assembly to be linked.
Src	The path to the source file to be linked and compiled dynamically.

@outputCache

The OutputCache directive controls the output caching policies of a web page or a user control.
The basic syntax of OutputCache directive is:

```
<%@ OutputCache Duration="15" VaryByParam="None" %>
```

ASP.NET RICH CONTROLS

1. Calendar control

- is used to get input of dates.
- it allows the user to select the dates graphically.
- move to the next or previous month.
- It displays the dates in one month view just like a traditional calendar.

Syntax

```
    <asp:Calendar id="Calendar1"
runat="server">
```

Property	Description	Style class
DayHeaderStyle	The style for the section of the calendar where the names of the days of the week appear.	TableItemStyle
DayStyle	The style for the individual days in the displayed month.**Note** Weekends, the current date, and the selected day can have different styles by setting the **WeekendDayStyle**, **TodayDayStyle**, and **SelectedDayStyle** properties, respectively.	TableItemStyle
NextPrevStyle	The style for the sections at the left and right ends of the title bar where the month navigation **LinkButton** controls are located.	TableItemStyle
OtherMonthDayStyle	The style for the days from the previous and next month that appear on the current month view.	TableItemStyle
SelectedDayStyle	The style for the selected date.**Note** If this property is not set, the style specified by the **DayStyle** property is used to display the selected date.	TableItemStyle
SelectorStyle	The style for the column on the left of the **Calendar** control containing links for selecting a week or the entire month.	TableItemStyle
TitleStyle	The style for the title bar at the top of the calendar containing the month name and month navigation links.**Note** If **NextPrevStyle** is set, it overrides the style for the next and previous month navigation controls located at the ends of the title bar.	TableItemStyle
TodayDayStyle	The style for the current date.**Note** If this property is not set, the style specified by the **DayStyle** property is used to display the current date.	TableItemStyle
WeekendDayStyle	The style for the weekend days.**Note** If this property is not set, the style specified by the **DayStyle** property is used to display the weekend dates.	TableItemStyle

TABLE 2	
Property	Description
ShowDayHeader	Displays or hides the section that displays the days of the week.
ShowGridLines	Displays or hides the grid lines between the days of the month.
ShowNextPrevMonth	Displays or hides the navigation controls to the next or previous month.
ShowTitle	Displays or hides the title section.

The AdRotator Control

Displays an advertisement banner on a Web Forms page.

```
<asp:AdRotator
    id="Value"
    AdvertisementFile="AdvertisementFile"
    KeyWordFilter="KeyWord"
    Target="Target"
    OnAdCreated="OnAdCreatedMethod"
    runat="server"/>
```

- The **AdRotator** control uses a separate XML advertisement file to store the advertisement information, such as the location of the image to display and the URL of the page to link to.

- The **AdvertisementFile** property of the **AdRotator** control specifies the path to this file.

When creating the advertisement file, opening and closing **<Advertisements>** tags mark the beginning and the end of the file, respectively.

- Opening and closing **<Ad>** tags delimit each advertisement. All advertisements are nested between the opening and closing **<Advertisements>** tags.

- If the file contains multiple **<Advertisements>** tags, only the first set of **<Advertisements>** tags in the file will be parsed by the **AdRotator** control.
- All other **<Advertisements>** tags will be ignored.
- The data elements for each advertisement are nested between the opening and closing **<Ad>** tags.
- Although certain data elements are predefined (such as **ImageUrl** and **NavigateURL**), you can place custom elements between the **<Ad>** tags.
- These elements will be read by the **AdRotator** control when it parses the file. The information is then passed to the **AdCreated** event in the **AdProperties** dictionary property.

Property	Description
Advertisement File	sets the relative paths to the advertisement data file (in Xml)
Keyword Filter	Designates a keyword to use that filters the advertisements. If specified, only banner ads that have a keyword that maches the value in keywordfilter are eligible for display.
Target	specifies the name of the browser window or frame in which to display the advertisement. Keywords such as _TOP, _NEW,_CHILD,_SELF,_PARENT,_BLANK can be used.

The impression feature allows to make a particular banner ads appear more often than others.
The values for Impression are relative higher values receive more rotation.

It defines an error called OnAdcreated. This event is raised upon the next post-back to the server whenever the advertisement changes.
It is useful for adjusting the content of the rest of the page based on which based on the add the Adrotator control is picked.

This event has these members

1. e.Adproperties – Idictionary collection of all propertied for the selected banner ad.)
2. e.AlternateText – it corresponds to the alternatetext in the XML file.
3. e. ImageUrl-corresponds to the ImageURL in the XML file.
4. e.NavigateURI – corresponds to the TargetURL in the XML file.

Validation control

ASP.NET validation controls validate the user input data to ensure that useless, unauthenticated, or contradictory data don't get stored.

ASP.NET provides the following validation controls:

•RequiredFieldValidator
•RangeValidator
•CompareValidator
•RegularExpressionValidator
•CustomValidator
•ValidationSummary

BaseValidator Class

The validation control classes are inherited from the BaseValidator class hence they inherit its properties and methods. Therefore, it would help to take a look at the properties and the methods of this base class, which are common for all the validation controls:

Members	Description
ControlToValidate	Indicates the input control to validate.
Display	Indicates how the error message is shown.
EnableClientScript	Indicates whether client side validation will take.
Enabled	Enables or disables the validator.
ErrorMessage	Indicates error string.
Text	Error text to be shown if validation fails.
IsValid	Indicates whether the value of the control is valid.
SetFocusOnError	It indicates whether in case of an invalid control, the focus should switch to the related input control.
ValidationGroup	The logical group of multiple validators, where this control belongs.
Validate()	This method revalidates the control and updates the IsValid property.

RequiredFieldValidator Control

The RequiredFieldValidator control ensures that the required field is not empty.

It is generally tied to a text box to force input into the text box.

The syntax of the control is as given:

```
<asp:RequiredFieldValidator ID="rfvcandidate" runat="server" ControlToValidate ="ddlcandidate"

ErrorMessage="Please choose a candidate" InitialValue="Please choose a candidate">

</asp:RequiredFieldValidator>
```

RangeValidator Control

The RangeValidator control verifies that the input value falls within a predetermined range.
It has three specific properties:

The syntax of the control is as given:

```
<asp:RangeValidator ID="rvclass" runat="server"
ControlToValidate="txtclass" ErrorMessage="Enter your class (6 - 12)"
MaximumValue="12" MinimumValue="6" Type="Integer"> </asp:RangeValidator>
```

Properties	Description
Type	It defines the type of the data. The available values are: Currency, Date, Double, Integer, and String.
MinimumValue	It specifies the minimum value of the range.
MaximumValue	It specifies the maximum value of the range.

CompareValidator Control

The CompareValidator control compares a value in one control with a fixed value or a value in another control. It has the following specific properties:

The basic syntax of the control is as follows:

```
<asp:CompareValidator ID="CompareValidator1" runat="server" ErrorMessage="CompareValidator">

 </asp:CompareValidator>
```

Properties	Description
Type	It specifies the data type.
ControlToCompare	It specifies the value of the input control to compare with.
ValueToCompare	It specifies the constant value to compare with.
Operator	It specifies the comparison operator, the available values are: Equal, NotEqual, GreaterThan, GreaterThanEqual, LessThan, LessThanEqual, and DataTypeCheck.

RegularExpressionValidator

The RegularExpressionValidator allows validating the input text by matching against a pattern of a regular expression. The regular expression is set in the ValidationExpression property.

The following table summarizes the commonly used syntax constructs for regular expressions:

Apart from single character match, a class of characters could be specified that can be matched, called the metacharacters.

Character Escapes	Description
\b	Matches a backspace.
\t	Matches a tab.
\r	Matches a carriage return.
\v	Matches a vertical tab.
\f	Matches a form feed.
\n	Matches a new line.
\	Escape character.

The syntax of the control is as given:

```
<asp:RegularExpressionValidator ID="string" runat="server" ErrorMessage="string"
ValidationExpression="string" ValidationGroup="string">

</asp:RegularExpressionValidator>
```

CustomValidator

The CustomValidator control allows writing application specific custom validation routines for both the client side and the server side validation.

The client side validation is accomplished through the ClientValidationFunction property.
The client side validation routine should be written in a scripting language, such as JavaScript or VBScript, which the browser can understand.

The server side validation routine must be called from the control's ServerValidate event handler. The server side validation routine should be written in any .Net language, like C# or VB.Net.
The basic syntax for the control is as given:

```
<asp:CustomValidator ID="CustomValidator1" runat="server" ClientValidationFunction=.cvf_func.
 ErrorMessage="CustomValidator">

</asp:CustomValidator>
```

Data List Control

The Web controls that produce list are called List Controls.

The data for the list controls can come from a variety of sources they are:
Flat files, XML, relational databases.

The three types of list controls are :

(i) Repeater Control
(ii) Data Grid
(iii) Data List

(iv) Repeater Control
It is the simplest of the List Controls.
It allows for the display of data items by rendering each row in a data source as a row in the web page.
The Repeater control renders the rows in te list according to an HTML template that is specified.

Using this template the total appearance of the list is controlled.

A list header, list item and list footer can each have different styles.

The Repeater Control uses the following syntax :

```
<asp :Repeater id: “Repeater1” runat =“server” DataSource=“<% databindindingexpression %>”>

<HeaderTemplate>
    Header template HTML
</HeaderTemplate>

<ItemTemplate>
     Item template HTML
</ItemTemplate>

<AlternatingItemTemplate>
    Alternating item template HTML
</AlternatingItemTemplate>

<SeparatorTemplate>
     Separator template HTML
</SeparatorTemplate>
<FooterTemplate>
    FooterTemplate HTML
</FooterTemplate>
<asp : Repeater>
```

HTML is specified for rows, headers and footers for a particular list.

Templates used in the Repeater Control

Template	Description
AlternatingItemTemplate	HTML to render for alternating rows in the list
FooterTemplate	HTML to render at the end of all the rows in the list
HeaderTemplate	HTML to render before any rows in the list are rendered
ItemTemplate	HTML to render for each row to the list
SeparatorTemplate	HTML to render between rows in the list.

User Control

- The ASP.NET controls allows to build most web applications with ease.
- Web applications use custom code that is specific to an organization.
- This code is likely to be reused many times and in many different projects.
- A user control is a reusable piece of ASP.NET code.
- When ASP.NET code is packaged up as a user control, it makes reuse of the code as another web form as easy as supplying a new page directive at the top of the Web form and placing the User Control in the page.
- User controls are used within a Web Form similar to other web controls are used.

Authoring a User Control

- User Controls are simply a collection of other Web Controls and their behaviors.
- User Controls are exposed with own properties.

Saving state with the StateBag object

ASP.NET provides an option to save state across page requests without saving any information on the server.

ViewState is a static object of type StateBag that can be used to store arbitrary values across page requests.

The properties and method of the StateBag class are:

Properties

Count – gets the number of StateItem objects in the StateBag object
Item – Gets or sets the value of an item stored in the StateBag object
Keys - Gets a collection of keys representing the items in the StateBag object.
Values – Gets a collection of view state values stored in the StateBag object.

Methods:

Add – adds a new StateItem object to the StateBag object. If the item already exists in the StateBag object

Add updates the value of the item.

Clear – removes all items from the current StateBag object.

GetEnmerator – returns an enumerator that iterates over all the key-value pairs of the StateItem ibjects stored in the StateBag object.

IsItemDirty –checks StateItem object stored in the StateBag object to evaluvate whether it has been modified since the call to Control.TrackViewState.

Remove – removes the specified StateItem object from the StateBag Object.

Every webform has access to the ViewState object, which is a static object that is not declared explicitly.
Any value can be stored in the ViewState Collection.

Eg:

Writing to viewstate

ViewState(“a”) = “Test”

Retrieving from ViewState
S = ViewState(“a”)

ASP.NET Intrinsic Objects

Def : It is a set of objects in the ASP.NET that provides low level access to web application protocols and frameworks.

We can directly work with CGI,HTTP stream and session management capabilities of ASP.NET.

It also provides compatibility with the classic ASP, this is the only to provide webserver functionality through program code.

The HTTPRequest Object :

This is responsible for retrieving information sent in an HTML form. The HTML form data can be sent using two methods : HTTP GET and HTTP POST.

The CGI encoding scheme is used to encode data as name-value pair, where name is the name of the HTML form and value is the data associated with the form element.

Properties and methods for HTTPRequest objects

ContentEncoding – gets the character set of the entity body

ContentType - gets the MIME content type of the incoming request.

FilePath – gets the virtual path of the current request.

Headers – Gets the collection of HTTP headers

Path – gets the virtual path of the current request

UserHostName – Gets the DNS name of the remote client.

Methods of the HttpRequest

SaveAs- save an HTTP request to disk
BinaryRead – performs a binary read of a specified number of bytes from the current input stream.
MapPath – maps the virtual path in the requested URL to the physical path on the server for the current request.
MapImageCoordinates – maps an incoming image field form parameter to appropriate x,y coordinate values.

HttpResponse Object :

The HttpResponse object is to send data back to the user’s browser and control the output of the HTTP stream in a programmatic way.

- Content can be inserted dynamically into the HTTP stream.
- Cookies can be set as a part of the script.

The HttpResponse object methods programmatically control the HTTP output stream.

Using HttpResponse

The flow of the HTTP output is controlled
Write to the output stream
Redirect the browser and append custom data to the web server logs.

The HttpResponse object contains one collection called Cookies.

This allows to place cookies on the client machine.

The HttpServerUtility object

The HttpServerUtility object contains some miscellaneous utility functions and accesses properties on the web server

With the HttpServerServer object,performs HTML/URL encoding, resolve physical pathnames on the server and create instances of COM software components.

The HttpApplicationState object

It is used to store application – persistent data available for all users of the web application to access.
The HttpApplicationState Object stores all of its application variables in collection.

To store an application variable,

Application(“a”)=value

The value assigned to the application variable can be of any type, including string, numeric and other variable.

To retrieve a value of an application variable
Variable = Application(“variable name”)

The HttpSessionState Object

HttpSessionState class van be used to declare variables that have session scope.

They can be used in any .asp file within the application.

This can be also used to find the left time of the user

The session is considered ended if that timeout period has elapsed.

HttpSessionState variables are destroyed after the time out period.

It has a method called Abandon which immediately terminates the session.

The ObjectContext Object

It is used to control the transactions for COM+.
It can abort or complete the COM+ transaction.

Unit III

4.1 Common features of.net framework class library

The .net framework class library contains an assortment of classes and data types that can be used to develop.net applications

Feature of .net class.

- **Data collection.**
- **File I/O**
- **Event logging**
- **Message queuing**
- **Text Handling**
- **Internet communications**
- **XML data handling**
- **Internet email**

4.2 Data collections (System.collections)

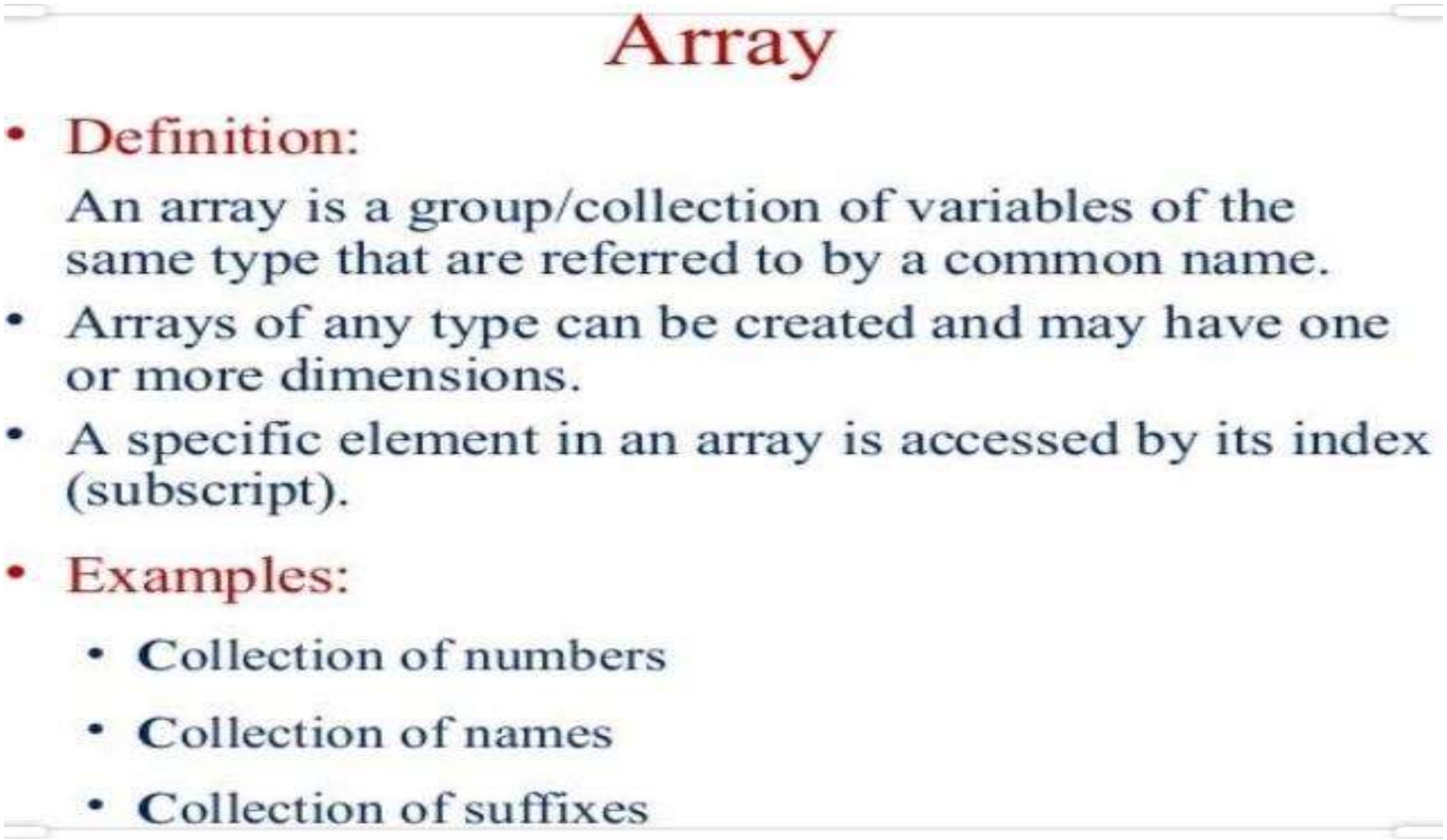
1.A collection is a set of similar type of objects that are grouped together.

2.System. Collections namespace contains specialized classes for storing and accessing the data.

3.Each collection class defined in . NET has a unique feature.

Different types of sets:

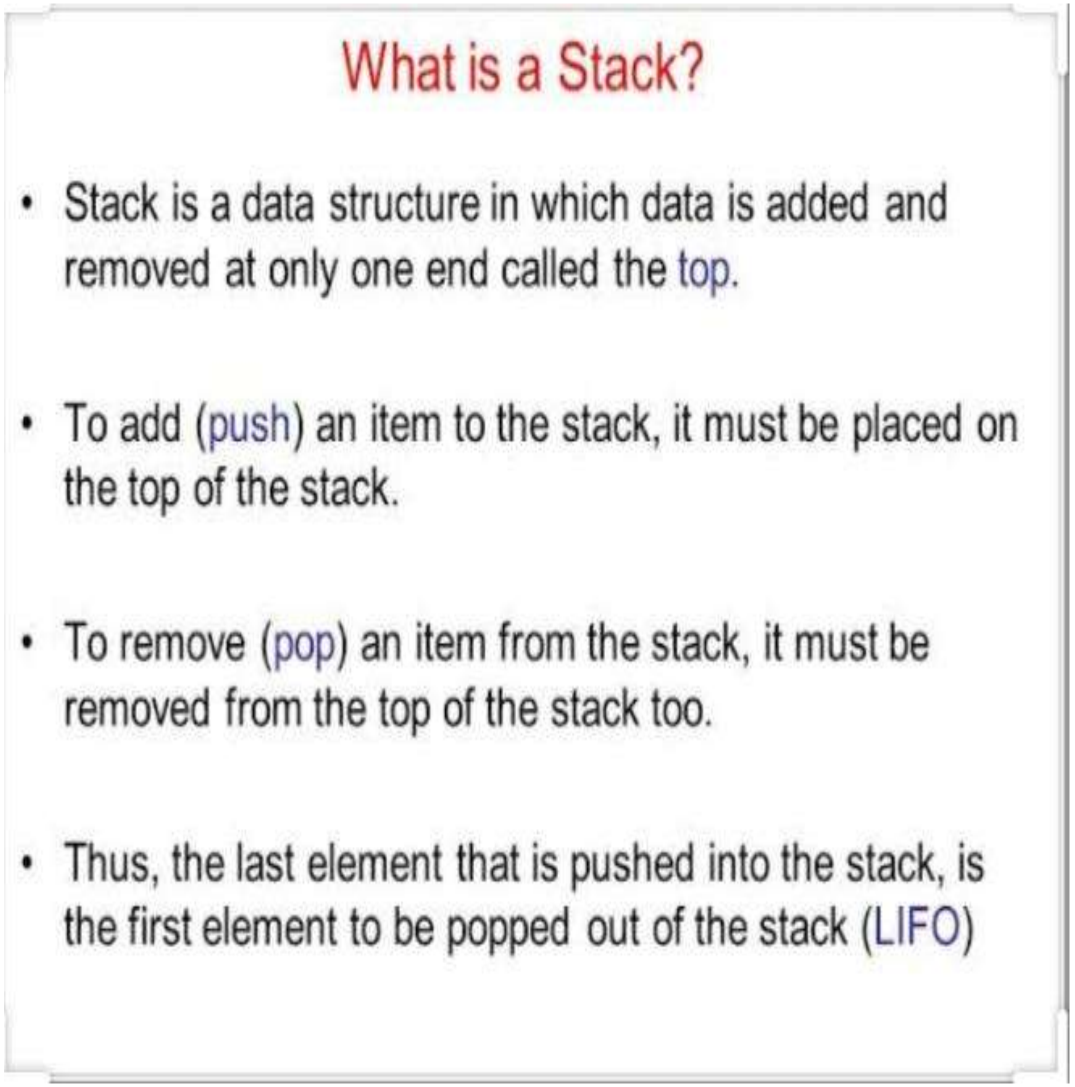
- Array
- Stacks
- Queues
- Hashtables



Array

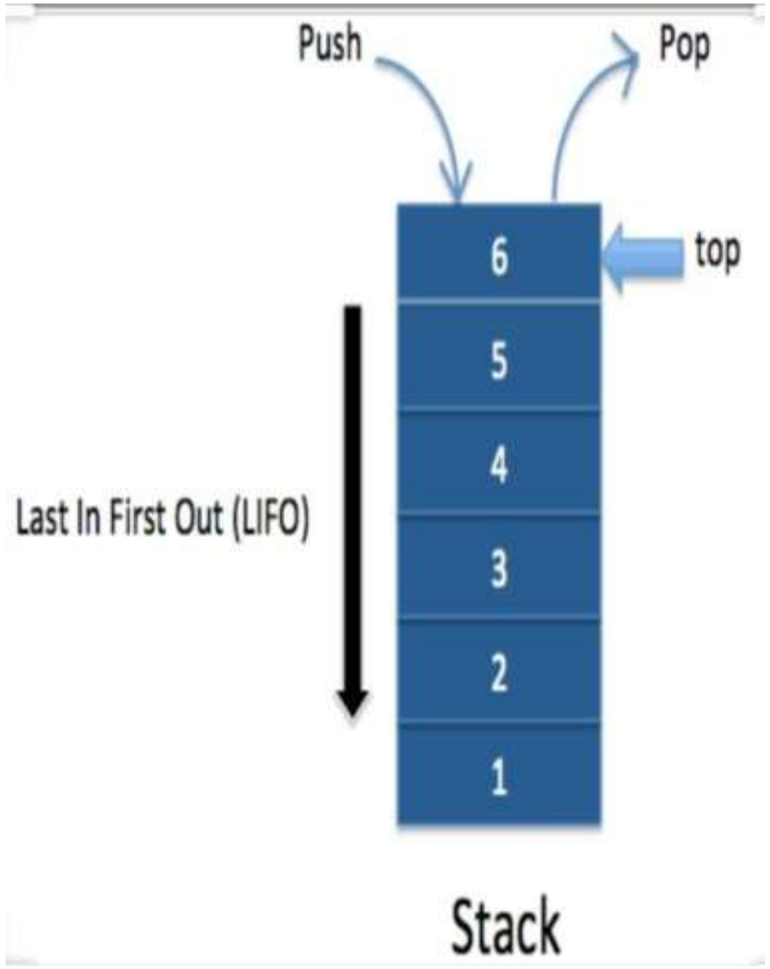
- Definition:

  An array is a group/collection of variables of the same type that are referred to by a common name.
- Arrays of any type can be created and may have one or more dimensions.
- A specific element in an array is accessed by its index (subscript).
- Examples:
  - Collection of numbers
  - Collection of names
  - Collection of suffixes

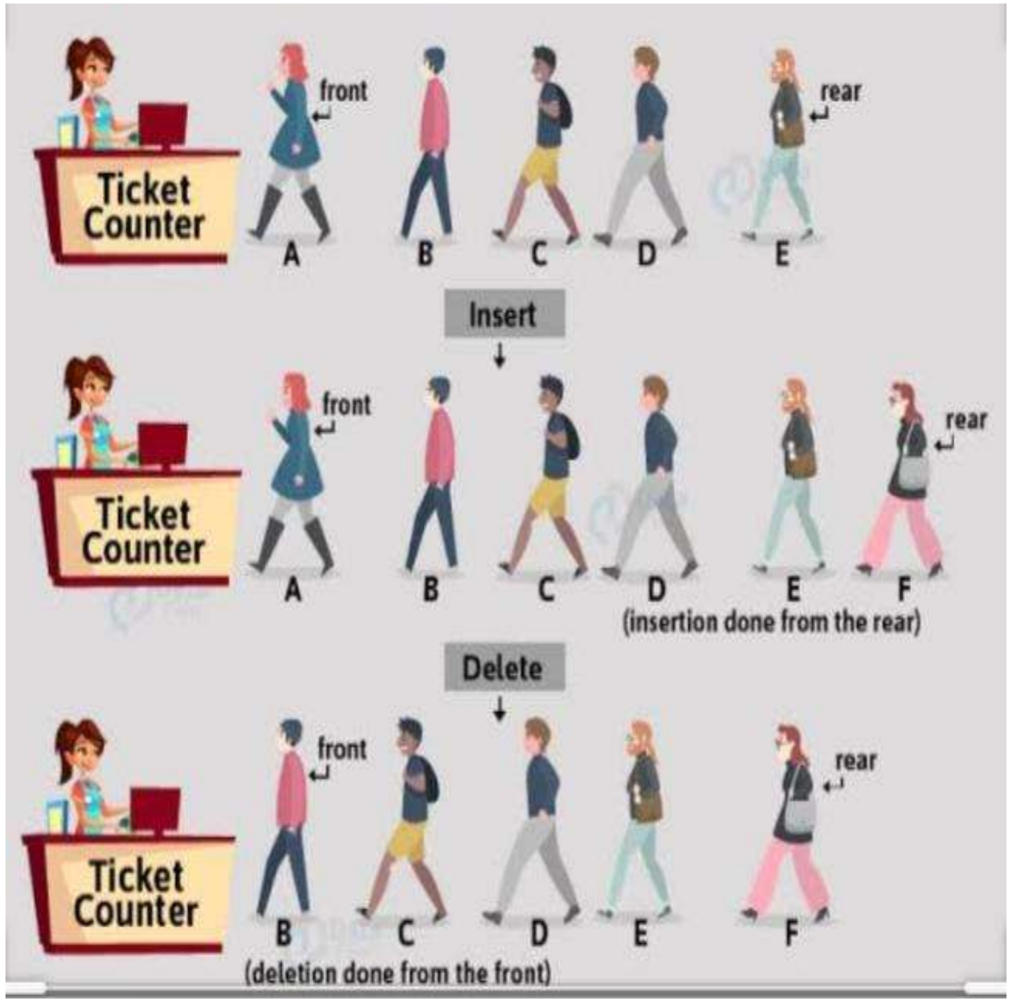


What is a Stack?

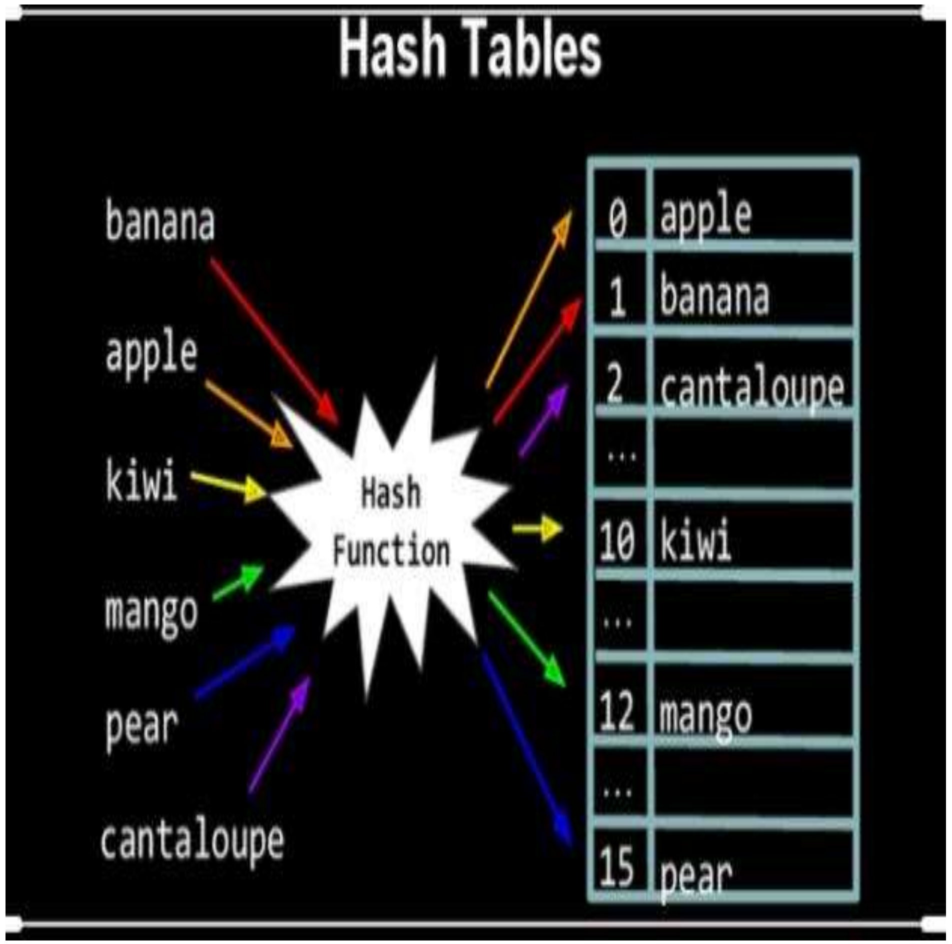
- Stack is a data structure in which data is added and removed at only one end called the top.
- To add (push) an item to the stack, it must be placed on the top of the stack.
- To remove (pop) an item from the stack, it must be removed from the top of the stack too.
- Thus, the last element that is pushed into the stack, is the first element to be popped out of the stack (LIFO)



A **queue** is an ordered collection of items where the addition of new items happens at one end, called the “rear,” and the removal of existing items occurs at the other end, commonly called the “front.”



Hash table (hash map) is a data structure that implements an associative array abstract data type, a structure that can map keys to values. A hash table uses a hash function to compute an index, also called a hash code, into an array of buckets or slots, from which the desired value can be found.



THE STACK CLASS

- The System.Collections.Stack class implements a stack data structure.Stacks have many different uses and can hold any System.Object type.
- Some programs offers a multiple undo feature.
- A stack can a be used to track that state information.

Example:

```
Imports System
Imports System.Collections
Imports Microsoft.VisualBasic

Module  Module1

    Sub  Main()

            Dim stkUndoStack As New Stack()
```

```
                    Do While (AddPhrase(stkUndoStack))
                    Loop

                    Console.WriteLine("Final sentence is:")
                    DisplayStackContents(stkUndoStack)
            End Sub
            Function AddPhrase(By Ref UndoStack As Stack)  As
        Boolean
                    Dim strWordOrPhrase As String
                     Dim strPhrase As String
                    AddPhrase = True

                    Console.Write("Add a word or phrase: ")
                    strWordOrPhrase = Console.ReadLine()

                    Select Case strWordOrPhrase
                        Case ","
                                AddPhrase = false
                    Exit Function
            Case “#”
                    If UndoStack.Count >  0  Then
                         Console.WriteLine(                “Undo phrase: “&
                                                      UndoStack.Peek())
                                   strPhrase = UndoStack.Pop()
                    AddPhrase = True
            Else
                    Exit Function
            End If
        Case Else
                    UndoStack.Push(strWordOrPhrase)
    End Select

    DisplayStackContents(UndoStack)

End Function
```

```
Sub DisplayStackContents(By Ref UndoStack As Stack)
      Dim strphrase As String
      Dim aryStandardArray As Object()
      Dim intIndex As Integer

      Console.Write(“Current contents:”)
              aryStandardArray = UndoStack.ToArray

      for intIndex = UBound(aryStandardArray) To 0 Step -1
               Console.Write (aryStandardArray (intIndex) & “ “
      Next
      Console.Write(Control Chars.Crlf)

End Sub

End Module
```

- The AddPhrase() function handles pushing and popping.
- Peek()method:allows the user to look at the value before it is popped off the stack.
- Pop()function:it removes from the top of the pop.
- Push()method:to add an element.
- ToArray()method:to obtain a string array representation of the data in the stack.

THE QUEUE CLASS

- Queues are used to model data that is stored in FIFO.

  Eg:Documents being submitted to the printer,appropriately called a print queue.

Example:

```
Imports System
Imports System.Collections
Imports Microsoft.VisualBasic

Module  Module1
      Public Class DocRecord
              Public DocTitle As String
               Public DocOwner As String
               Public DateTimeSubmitted As Date
              Sub New(Optional  ByVa1 DTitle = “ ”,
                              Optional ByVa1 Downer =“”)
              DocTitle = DTitle
              DocOwner = Downer
      End Sub
End Class
```

```
Function InitializeQueue() As Queue
        Dim objDocQueue As New Queue()
         Dim objDoc As DocRecord
         Dim intIndex As Integer

        For intIndex = 1 To 5
                 objDoc = New DocRecord()

                With objDoc
                        .DocOwner = “joe”
                        .DocTitle = “Document #” &
                                intIndex.ToString()
                End With

                        objDocQueue.Enqueue(objDoc)
          Next

        InitializeQueue = objDocQueue

End Function
```

```
Sub AddDocToQueue(ByVa1 DocQueue As Queue,
            ByVa1 Doc As DocRecord)
     DocQueue.Enqueue(Doc)
     Console.WriteLine(“{0} submitted to queue”,
            Doc.DocTitle)
End Sub

Sub PrintDoc(ByVa1 DocQueue As Queue)
     Dim objDoc As DocRecord
            objDoc = DocQueue.Dequeue()
     Console.WriteLine(“{0} submitted to queue”,
            objDoc.DocTitle)
End Sub
```

- The program adds the newly created object to the queue using the Enqueue() method.
- The program that displays the current contents of the queue.
- Removing it from the queue which isolate by using the Dequeue() method.

THE HASHTABLE CLASS

- The System.Collections.Hashtable class implements an assosiative name value pair array of objects.
- Hashtables are highly used data structures.

Example:

```
Imports System
Imports System.Collections
Imports Microsoft.VisualBasic

Module  Module1

      Public Class ContactEntry
              Public m- strLastName As String
               Public m- strFirstName As String
               Public m- intAge As Integer

              Sub New(Optional ByVa1 Lname =“”,
                              Optional ByVa1 Fname =“”,
                              Optional ByVa1 Age = 0)
              m-strLastName = LName
              m-strFirstName = Fname
      End Sub

End Class
```

```
If Not objContact Is Nothing Then
        Console.WriteLine(
                “Contact info: “ & ControlChars.Crlf &
                “Last Name: {0}” & ControlChars.Crlf &
                “First Name: {1}” & ControlChars.Crlf &
                “Age: {2}” & ControlChars.Crlf &
                objContact.m-strLastName,
                objContact.m-strFirstName,
                objContact.m-intAge. ToString())
        Else
                Console.WriteLine(“Contact not found”)
        End If

End Sub

Sub ListContacts(By Ref ContactTable As Hashtable)

                Dim objContact As ContactEntry
                For Each objContact In ContactTable.Values
                                Console.WriteLine(“{0}” & ControlChars &
                                “{1}” & ControlChars.Tab &
                                 “{2}” & ControlChars.Tab &
                                objContact.m-strLastName,
                                objContact.m-strFirstName,
                                objContact.m- intAge. ToString())
                Next

End Sub

Sub Main
      Dim objMyContacts As New Hashtable()

      AddContact(objMyContacts)
       AddContact(objMyContacts)
       AddContact(objMyContacts)
       ListContacts(objMyContacts)
       QueryContacts(objMyContacts)

End Sub

End Module
```

The Item() method is used to search an object in the hashtable.
A unique key value must be used to insert a new object into a hashtable.
The key value for each contact is randomly generated.

Handling file input/output :

File input/output plays key role in all application. The file class encapsulates various operation that can be performed on a single file such as, **creating, copying, deleting, moving, opening**.

The methods of the file class return special classes that are used to read & write, this is collectively known as stream classes.

Reading text files :

Text file contain only character data that is ASCII and unicode text. These methods of the file class is used to read these files easily.

Program

```
Imports system

Imports system .IO

Module  modul=1

       Sub main()

            Dim  objstreamReader  As   streamReader

                Try
                ObjstreamReader =file .Opentext(“test.txt”)

     Console. Write(objstreamReader .ReadToEnd())
```

```
ObjstreamReader  .close()
 Catch  E  as exception
    Console .writeLine(“error occurred: “&  E. Message)
  End try
 End sub
End module
```

Writing text files:

We can write a text file using the streamWritter class. The streamWritter class can write data to the stream in number of ways they are , **writing a single character , a character array , an object, a subset of a character array, simple data types(** boolean, decimal, integer, long, double)

Reading binary files:

The MP3ID3VI class contains member variable for each of the IB3VI segment that occur at different offsets in the file

The overloaded new () constructor calls the function that reads the song information

The work of reading that ID3V1 block begins with opening the file by creating a new filestream object in line.

writing Binary files:

This involves building a string containing the ID3V1 string converting the string to a byte array and writing the byte array to the file .

Properties of the StreamReader class:

- **Property**
- Base stream
- Current encoding

- **Description**
- Gets the character encoding
- Returns the stream object

Properties of StreamWritter class:

- **Property**
- Aut of Tusb
- BaseSteam
- Encoding
- Format provider (Textwritter)
- New line

- **Description**
- All output bound for the stream to be witten immediately when the value is true
- Returns the underlying stream object
- Gets the encoding in which the output is written
- Gets an object that contains formatting
- Gets or sets the line string used by the current Textwritter class

Performing file operations :

The file class helps you whenever you wish to change characteristics manage files

The open Text() method to return a filestream object for writing text files

Object factory for stream object the class can be used to rename, delete, move snd copy files

The files class methods are static so it has no constructor and it's available at any time in the program

The file Info class offers instance method of the same functionally as the file class

Copying, moving & Renaming files :

To make a copy of a file and rename it using the copy To () method and the move To () method

(Refer program in the book)

Deleting files:

We delete a file using the staticdelete () method of the file class

(Refer program in the book)

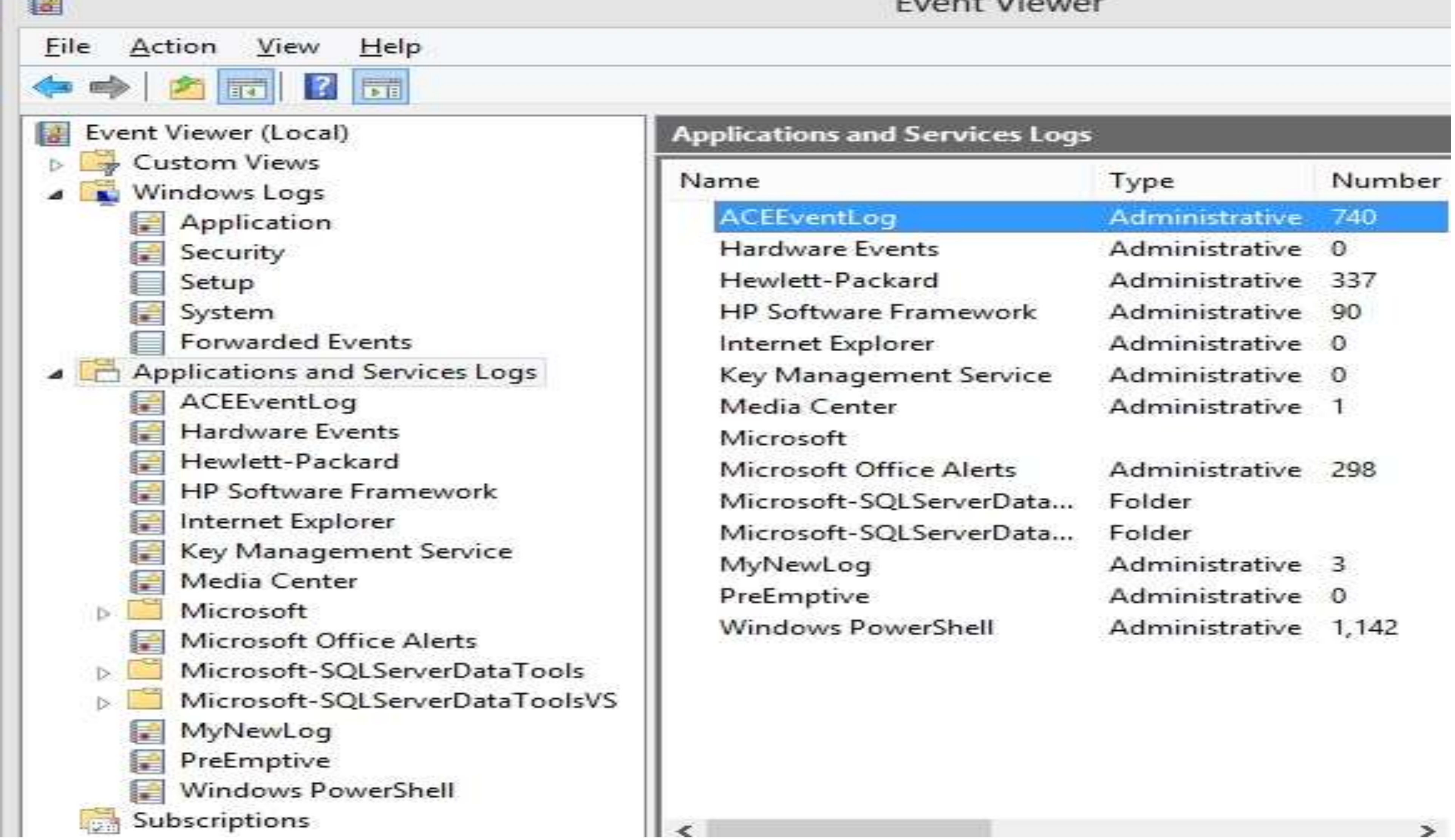
Using binary file I/O with the filestream object:

Graphic images, contain binary data that is the bytes in the files extend beyond what can be represented using text encoding method such as ASCII

The stream Reader & StreamWritter classes are designed to handle only text data for reading binary files the filestream and binary reader objects should be used for writting binary file use the binarywritter object together with the filestream object.

4.5 USING THE WINDOWS EVENT LOG (System.diagnostics)

- The first version of windows NT , a comman event log has been used as a tool to record application errors and events.
- Windows event log and it’s common repository that all applications use to record application events.
- The contents of the windows event log are viewed using the event viewer application.
- To launch the event viewer in windows 2000 , right click on the my computer icon and select manage.
- ***Computer Management → System Tools → Event Viewer.***



Event Viewer

```
Event Viewer (Local)
  Custom Views
  Windows Logs
    Application
    Security
    Setup
    System
    Forwarded Events
  Applications and Services Logs
    ACEEventLog
    Hardware Events
    Hewlett-Packard
    HP Software Framework
    Internet Explorer
    Key Management Service
    Media Center
    Microsoft
    Microsoft Office Alerts
    Microsoft-SQLServerDataTools
    Microsoft-SQLServerDataToolsVS
    MyNewLog
    PreEmptive
    Windows PowerShell
  Subscriptions
```

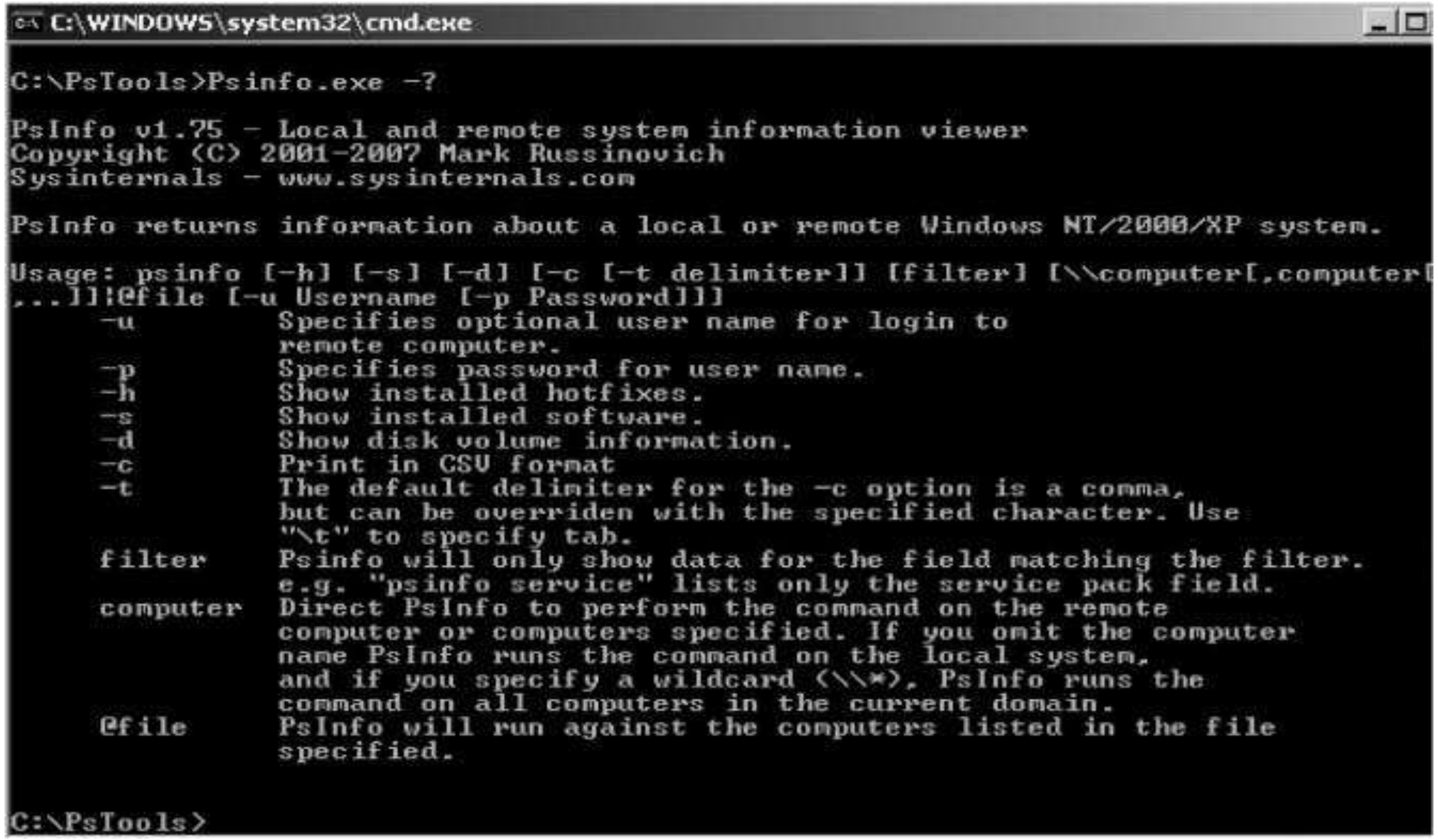
Applications and Services Logs

Name	Type	Number
ACEEventLog	Administrative	740
Hardware Events	Administrative	0
Hewlett-Packard	Administrative	337
HP Software Framework	Administrative	90
Internet Explorer	Administrative	0
Key Management Service	Administrative	0
Media Center	Administrative	1
Microsoft		
Microsoft Office Alerts	Administrative	298
Microsoft-SQLServerData...	Folder	
Microsoft-SQLServerData...	Folder	
MyNewLog	Administrative	3
PreEmptive	Administrative	0
Windows PowerShell	Administrative	1,142

- The event viewer is comprised of 3 different logs : 1. Application log 2. Security log 3. System log.
- These are used to record program events , security events and audits , and low level device and system events respectively.
- The type column describes the type of the event.
- This is usually information warning are error.
- The date and time columns indicate the date and time respectively , that the event was posted to the log.
- The EventLog object is very versatile and provides facilities to read , write and query events
- This programs writes a string to the application log and read the current entries from the same log.

SAMPLE PROGRAM:-

```
Imports system
Imports system.Diagnostics
Import Microsoft.VisualBasic
Module Module1
 Sub main()
   Dim strLogstr As String
   Console.Write(“Enter a string”)
   strLogStr = Console.ReadLine()
   WriteToLog(strLogStr)
   ReadApplicationLog()
 End Sub
End Module
```



```
C:\PsTools>Psinfo.exe -?

PsInfo v1.75 - Local and remote system information viewer
Copyright (C) 2001-2007 Mark Russinovich
Sysinternals - www.sysinternals.com

PsInfo returns information about a local or remote Windows NT/2000/XP system.

Usage: psinfo [-h] [-s] [-d] [-c [-t delimiter]] [filter] [\\computer[,computer[
,...]]|@file [-u Username [-p Password]]]
     -u         Specifies optional user name for login to
                remote computer.
     -p         Specifies password for user name.
     -h         Show installed hotfixes.
     -s         Show installed software.
     -d         Show disk volume information.
     -c         Print in CSV format
     -t         The default delimiter for the -c option is a comma,
                but can be overriden with the specified character. Use
                "\t" to specify tab.
     filter     Psinfo will only show data for the field matching the filter.
                e.g. "psinfo service" lists only the service pack field.
     computer   Direct PsInfo to perform the command on the remote
                computer or computers specified. If you omit the computer
                name PsInfo runs the command on the local system,
                and if you specify a wildcard (\\*), PsInfo runs the
                command on all computers in the current domain.
     @file      PsInfo will run against the computers listed in the file
                specified.


C:\PsTools>
```

- We proceed to line where we create a new EventLog object.
- We set the log and MachineName properties with the values “Application” and “.” , respectively.
- With those properties set we are ready to retrieve information about the log.
- We start a For ….Each loop to iterate the collection of EventLogEntry object in entries.

4.6 Working with Active Directory Services (System.DirectoryServices)

- The term directory, as it relates to Active Directory Services is a general one.
- It does not specifically refer to directories on your file systems(folders) but instead to any data stored in a hierarchical manner.
- Examples of directories include a name and address book, an employee roster, and a store of account information on multiuser computer system.
- Each of these directories is referred to as a *namespace*.
- A name space is a root node in the directory service.
- The children of these nodes are called *container objects*.
- Container object is can contain other container objects and also leaf objects.

The following released providers can be used with Active Directory Services:

- **Windows NT 4.0 Directory Services:** This is the most widely used directory service. Windows NT 4.0 Directory Services allow you to completely administer a Windows 2000/NT server or domain. You can control Windows 2000/NT services, including starting, stopping, and setting the start up status.
- **Lightweight Directory Access Protocol (LDAP) and Exchange Server:** LDAP is an Internet standard protocol for communicating with a wide variety of directory services. One application for LDAP is to provide access to data on Microsoft Exchange Server. Active Directory Services allows access to items such as message stores and recipient data.

- **Internet Information Server (IIS) :** With the IIS provider, we can perform many administrative tasks relating to IIS. For example, you can manage virtual directories in the Web server filespace, set up other Internet services like FTP, manage log files, and configure settings for the servers.
- **Novell Directory Services (NDS):** Active Directory Services also allows access to NDS.

Displaying Active Directory Services Contents:

we will explore an example of how to query for information about a computer and a user account on that computer.

```
Imports System
Imports System.IO
Imports System.DirectoryServices
Imports Microsoft.VisualBasic

Module Modulel
Public Sub GetADInfo(ByVal ADPath As String)
    Dim strKey As String
    Dim objValue As Object
    Dim objDirEnt As DirectoryEntry

    Try
        objDirEnt = New DirectoryEntry(ADPath)

        Console.WriteLine (“Name = “ & _
                       objDirEnt.Name)
        Console.WriteLine (“path = “ & _
                       objDirEnt.path)
        Console.WriteLine (“SchemeClassName = “ & _
                        objDirEnt.SchemaClassName)
        Console.WriteLine(“ “)
        Console.WriteLine(“Properties:”)
```

```
      For Each strKey In _
              objDirEnt.Properties.PropertyNames
         Console.write(ControlChars.Tab & _
              strKey & " = ")
         Console.WriteLine("")
         For Each objValue In_
                  objDirEnt.Properties(strKey)
         Console.WriteLine(ControlChars.Tab & _
              ControlChars.Tab & objValue.ToString())
         Next
      Next
    Catch E As Exception
      Console.WriteLine("Error: " E.Message)
    End Try

  End Sub
```

```
   Sub Main()
        GetADInfo(“WinNT://mdevbiz01”)
        Console.WriteLine(“----------------------------”)
        GetADInfo(“WinNT://rmdevbiz01/matt”)
   End Sub
End Module
```

In the example above, we use the name of a computer on the network,RMDEVBIZ01, to construct new DirectoryEntry object in line. After the constructor executes and the node is located, the properties of the DirectoryEntry object are populated in the code block beginning in line.

Nodes in Active Directory Services can contain any number of properties. These properties are housed in a PropertyCollection object. Ther propertyNames and value properties of the propertyCollection object are of particular interest. They are both collections that contain the names and values, respectively, of the node. Each value in the ProptertyNames collection can contain one or more values objects. So, with that in mind, we use two nested For…Each loops to visit each value in the PropertyCollectilon object. The outer loop that begins in line. Obtains all the names of the properties. Each Property name is then used as an index into the Properties collection to obtain individual values for the property in the inner loop that starts in line.

OUTPUT:

```
Name = RMDEVBIZ01
Path = WinNT://rmdevbiz01
SchemaClassName = Computer

Properties:
        OperatingSystem =
                windows NT
        OperatingSystemVersion =
                5.0
        Owner =
                SampleOwner
        Division =
                SampleDivision
        ProcessorCount =
                Uniprocessor Free
        Processor =
                x86 Family 6 Model Stepping 2
```

Searching Active Directory Services Contents:

- Sometimes we need to locate a particular item in Active Directory Services by using a searching mechanism.
- Often personnel directories are kept in an LDAP store made publically available.
- LDAP is also a very popular way to locate individuals on the Internet.

Example

```
Public Sub SearchAd(ByVal LastName As String)

    Dim objDirEnt As DirectoryEntry
    Dim objADSearchDResult As SearchResult
    Dim objSearchRSColl As SearchResultCollection
    Dim objTmpDirEnt As DirectoryEntry
    Dim objADSearch As DirectorySearcher
```

```
Try
     objDirEnt = New DirectoryEntry( _
          “LDAP://ldap.yourserverhere.com”)
     objADSearch = New DirectorySearcher(objDirEnt, _
          “(sn=“ & LastName & “)”)

     objSearchRScoll = objADSearch.FindAll()
  For Each objADSearchResult In objSearchRScoll
    objTmpDirEnt = _
               objADSearchResult.GetDirectoryEntry()
    Console.WriteLine(objTempDirEnt.Name)
  Next
  Catch E As Exception
    Console.WriteLine(“ERROR: “ & E.Message)
  End Try

End Sub
```

In the DirectoryEntry object’s constructor, we supply a URL to the LDAP SERVER In the form LDAP://yourserverhere . Com.

The code then moves on to next line, where the DirectorySearcher object is created.

Next Line executes the actual search. Since more than one user in the LDAP directory might match the query, the results are returned in the form of a SearchResultCollection object that implemtns the Icollection interface.

Inside the For…Each loop, notice jthe call to the GetDirectoryEntry() method in line. This method of the SearchResult class is used to obtain a DirectoryEntry object for each SearchResult object in the objSearchRSColl SearchResultCollection.

After this DirectoryEntry object is retrieved, we can display information about it, such as its Name property. This returns the LDAP common name (cn) for the object, which usually contains the person’s full name.

Properties of the SearchResult Class

Property	Description
Path	Gets the path for this SearchResult object
Properties	Gets a ResultPropertyCollection of properties set on this object.

Method of the SearchResult Class:

Method	Description
GetDirectoryEntry	Gets the DirectoryEntry object that matches with the SearchResult object.

Properties of the SearchResultsCollection Class

Property	**Description**
Count	Counts the number of SearchResult objects in the collection.
Handle	Gets the handle returned by IDirectorySearch::ExecuteSearch (called by DirectorySearcher)
Item	Gets the SearchResult object in the collection at the specified index
PropertiesLoaded	Gets the DirectorySearcher properties that were specified before the search was executed.

Methods of the SearchResultsCollection Class

Method	Description	Syntax
Contains	Indicates whether the specified SearchResult object is in the collection	Contains (ByVal result As SearchResult) As Boolean
CopyTo	Copies SearchResult objects in the collection to an array beginning with the specified index	CopyTo(ByVal results() As SearchResult, ByVal index As Integer)
IndexOf	Retrieves the index f a specified SearchResult object in this collection	IndexOf(ByVal result As SearchResult) As Integer

Modifying Active Directory Services Contents

The previous LDAP examples showed how you can display the contents of nodes and entries in Active Directory Services and how to search an LDAP store for particular items.

Let’s return to the example of retrieving the account information of a Windows user. We will modify the example to change the description property of the account. The code change is minimal; it requires setting one of the properties of a DirectoryEntry object with a new value, as shown in boldface text in the following code snippet.

```
Public Sub ChangeAcctDescription(ByVal Acctpath As String, _
                    ByVal DescValue As String)

    Dim objDirEnt As DirectoryEntry
    Try
```

```
        objDirEnt = New DirectoryEntry(Acctpath)
          objDirEnt .Properties(“Description”)(0) = DescValue
          objDirEnt .CommitChanges*)
    Catch E As Exception
          Console.WriteLine(“ERROR: “& E.Message)
    End Try
End sub
```

All that’s involved in changing the property is obtaining the key name of a property in theproperties collection (a propertycollection class) and specifying the item index (in this case, 0).

Then we can assign a new value.

Changes aren’t permanent until the Commitchanges() method is called and the application has sufficient permissions to modify the directory.

4.7 Using Message Queues (system.messaging)

- MICROSOFT MESSAGE QUEUES (MSMQ) is an integral part of many enterprise applications.
- It plays a key role in the scalability and robustness of transaction intensive system.
- MSMQ facilitates communication between disparate components or system by passing messages.
- It robustness comes from the fact that sender and receiver don’t need a currently active network connection to communicate.

- Message queuing works by using a store-and-forward technique for communication between system across the enterprise.
- Consider your e-mail system. It is a method of communication between two parties, but unlike “live” forms of communication, like a telephone conversation that relies on the fact that both parties are actively connected.
- E-mail does not require you to be present to receive a message.
- An e-mail message to you is received by the mail server and is stored there until you retrieve it. Hence, the name *store-and-forward communication.*

You can create four different types of queues in MSMQ

1. *Public queues* are published in **ACTIVE DIRECTORY SERVICES,** which enables any computer within the **ACTIVE DIRECTORY SERVICES** forest to access the public queues.
2. *Private queues* can be accessed only on the local computer on which they were created. *System queues* are a type of private queue reserved by the operating system for storage of administrative messages.
3. *Dead-letter queues* are reserved for storing messages that cannot be delivered to their destination.
4. *Transactional queues* receive messages within a transaction boundary.

CREATING A QUEUE:

- We will create a new public queue and a function called CreatePublicQueue (), that returns a MessageQueue object.
- The MessageQueue object provides the functionality needed to access message queues.

```
Public Function CreatePublicQueue (
            Byval Qname As String ) As MessageQueue
     Dim objQueue As MessaeQueue

     Try
       If Not MessageQueue. Exists (“.\” & Qname ) Then
            objQueue = MessageQueue. Create (“.\” & Qname )
        Else
                objQueue =  New MessageQueue (“.\” & Qname )
        End If
```

```
CreatePublicQueue = objQueue

Catch E As Exception
       Console.Writing (“Error: “& E.Message)
End Try

End Function
```

SENDING A MESSAGE TO A QUEUE:

- Sending a message to queue is simple.
- The following example sends a **String** object to the queue, you can send any object type (System.Object) to the queue using the Send () method.

```
Public Sub SendMessageToQueue(
        Byval Q As MessageQueue, Byval Msg As String )
    Q. Send( Msg)
End Sub
```

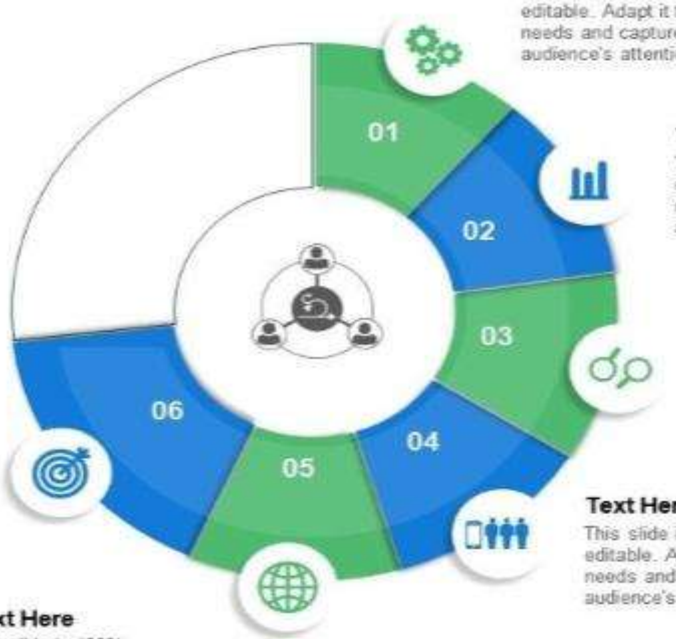
DEQUEUING A MESSAGE:

- Dequeuing a message is done by calling the Receive() method.
- This method returns a Message object, which contains the body of the message queue plus many different message characteristics.

```
Public Function Dequeue  (ByVal Q As MessageQueue )
    As String
    Dim objQMsg As Message
    objQMsg = Q.Receive()
    Dequeue = objQMsg.Body
End Function
```

Communication Client Server System

This slide is 100% editable. Adapt it to your need and capture your audience's attention.



Communication Client Server System
This slide is 100% editable. Adapt it to your needs and capture your audience's attention.

Text Here
This slide is 100% editable. Adapt it to your needs and capture your audience's attention.

Text Here
This slide is 100% editable. Adapt it to your needs and capture your audience's attention.

Text Here
This slide is 100% editable. Adapt it to your needs and capture your audience's attention.

Text Here
This slide is 100% editable. Adapt it to your needs and capture your audience's attention.

Text Here
This slide is 100% editable. Adapt it to your needs and capture your audience's attention.

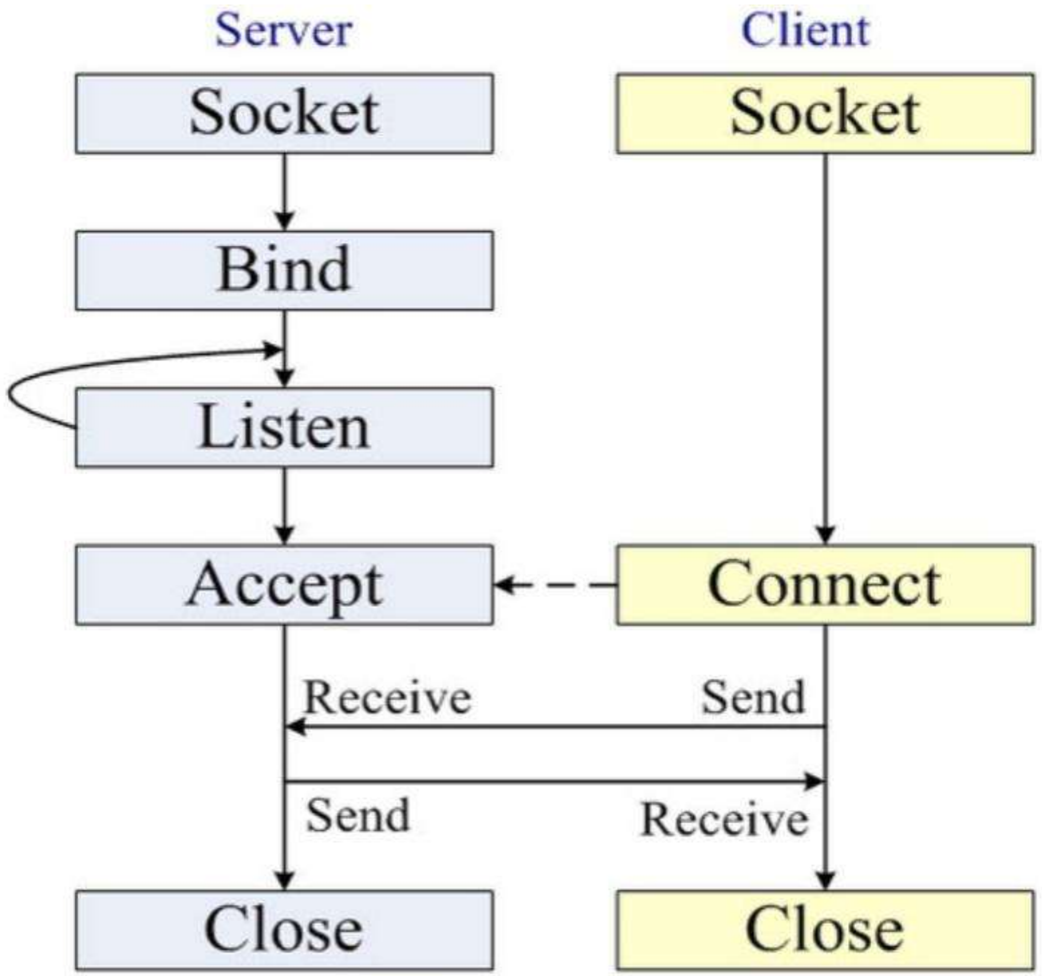
Communicating with server on the internet

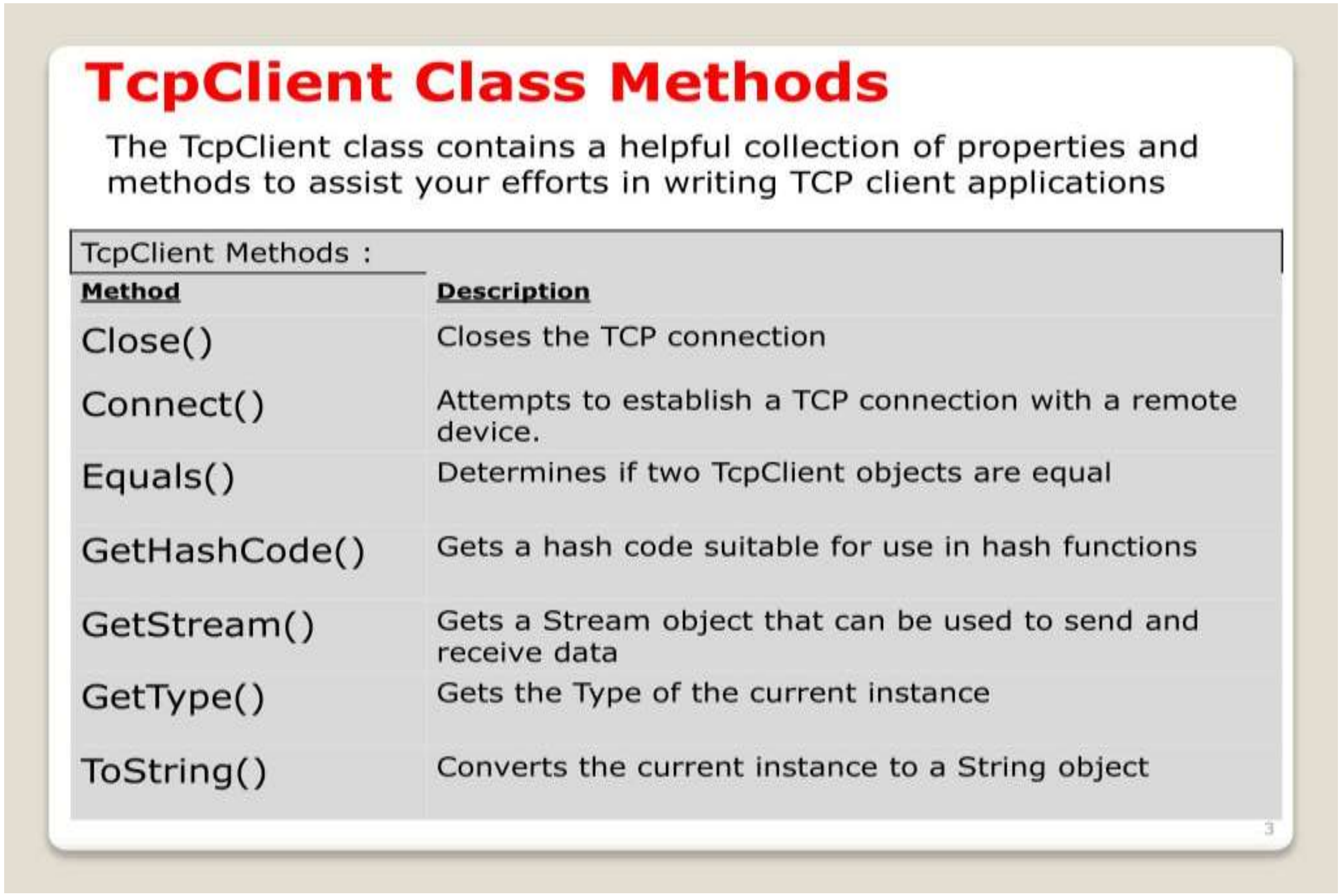
Communicate with servers or peers on the Internet using standard TCP/IP the same network protocol used by many other Internet service, including E-MAIL.

- The system.NET . Sockets namespace contain classes and types for establishing connection using TCP and also UDP (User Datagram Protocol) another method of network Communication on the internet.
- The Microsoft.NET Framework system .Net . Sockets class wraps function of the windows Sockets API, which was the standard way for writing Internet application software.

4.8 A simple TCP client application

- TCP client application begin with establing a connection to a server.
- The TcpClient class contains functionally to open and close connection and to obtain stream object on which to send and receive data.
- This Example implement a very Simple Web Client.
- HTTP Get command is sent over the socket connection and the response from the server is received and then displayed


Server
Client
Socket
Socket
Bind
Listen
Accept
Connect
Receive
Send
Send
Receive
Close
Close



TcpClient Class Methods

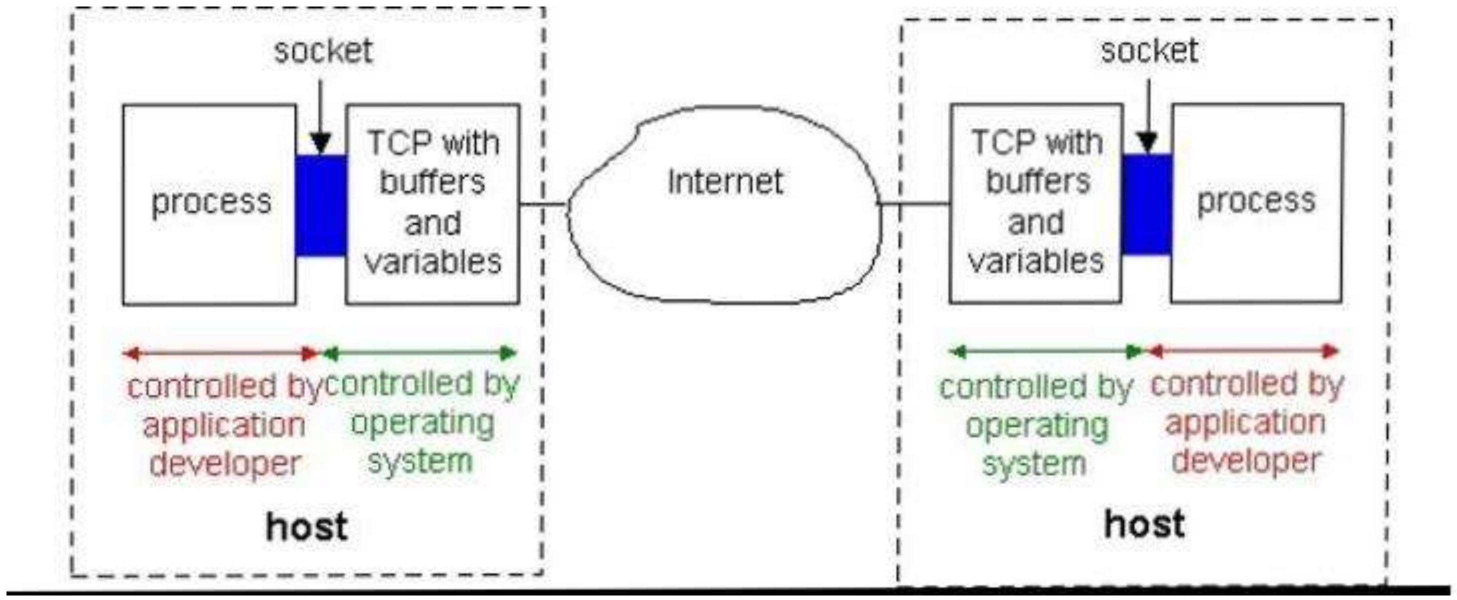
The TcpClient class contains a helpful collection of properties and methods to assist your efforts in writing TCP client applications

TcpClient Methods :

Method	**Description**
Close()	Closes the TCP connection
Connect()	Attempts to establish a TCP connection with a remote device.
Equals()	Determines if two TcpClient objects are equal
GetHashCode()	Gets a hash code suitable for use in hash functions
GetStream()	Gets a Stream object that can be used to send and receive data
GetType()	Gets the Type of the current instance
ToString()	Converts the current instance to a String object

A simple TCP server application

- The .NET Framework system .Net . Socket classes also allow you to write TCP server.
- A program for a rudimentary TCP server that is displays incoming data as it is received.
- The small server below listen for connection request and display incoming data from the TCP client.


socket
process
TCP with buffers and variables
controlled by application developer
controlled by operating system
host
Internet
socket
TCP with buffers and variables
process
controlled by operating system
controlled by application developer
host

Properties of the Tcplistener class

• Active	• Gets or sets a value that Indicates whether the listener's socket has been bound to a port started listening.
• Server	• Gets or sets the underlying network societ.

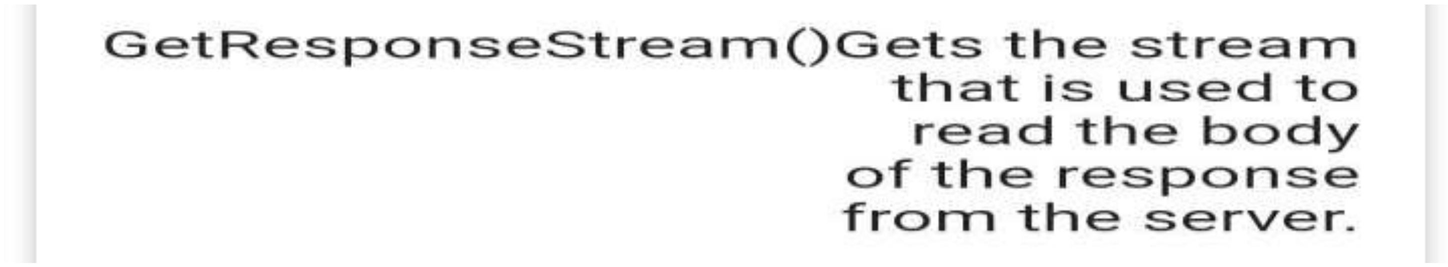
OFILE
SEARCH ENGINE
INFORMATION
ONLINE
NEWS FEED
ONLINE
INFORMATION
WORLD WIDE
HTTP://
INTERNET
PROFILE
SEARCH
GLOBAL

HTTP communication

- The TCP client example showed how you can use the TcpClient class to retrieve documents from the Web server using HTTP commands.
- Two classes in the .Net Framework, system .Net .WebRequest and System .Net .WebResponse, handle the most of the low-level details of HTTP based network communication.
- To implement yourself if you resorted to using the System .Net . Sockets classes.

Method of WebResponse class

Close()	**Closes the response stream.**



GetResponseStream()	Gets the stream that is used to read the body of the response from the server.

4.9 Manipulating XML Data

- So much has been said lately about XML that it may soon become a household name
- Rightfully so because XML is the technology that drives efficient data exchanges
- XML based enables elegant modelling of data
- .NET framework has extensive support for manipulating XML data
- The .NET framework includes two different XML parsers
  1. A tree based parser
  2. A stream based parser

Creating tree based XML documents

- The tree XML based parser reads an entire XML document
- Memory to construct a tree data structure ture representation of the XML data

Example

```
Xml Vali dating reader (objXml textreader)
Objxmldoc.load(objxmlvalidatingreader)
Displaynode (objxmldoc.childnodes)
Catch E As Exception
Console.writline(“ERROR:”&E.message)
End Try
End sub
```

Searching XML Nodes

- XML Node list object that contains the items that matched the query
- The query is expressed in a language called X path
- World wide web standard for searching the XML DOM

EXAMPLE

```
]>
<Songs>
<song format =“MP3”>
<title>Eh cumpar</title>
<artist>Julius LaRosa</artist>
<length>149<?length>
<category>Italian-American</category>
</song>
<song format=“co”>
<title>plasticity</title>
<artist>front line assembly </artist>
<length>486</length>
<category>industrial<category>
</song>
</songs>
```

Reading XML Data with validation

- Lets look at a modified example of reading in an XML document
- The sample below shows the previous XML file of songs with a new
- DTD section in boldface type that describes the valid data

EXAMPLE

```
<?xa1 version="1.0" encoding="UTF-8"?>
<!DOCTYPE Songs(song)>
<!ELEMENT artist(#PCDATA)
<!ELEMENT category(#PCDATA)>
<!ELEMENT length(#PCDATA)>
<!ELEMENT song(title, artist, length, category)>
<!ATTLIST song
Forest(mp3|LP|CS|CD)#REQUIRED>
<!ELEMENT title (#PCDATA)>
```

READING STREAM-BASED XML DATA:

- The **disadvantage of working with XML data** using the tree-based DOM method: the **entire XML file is read into the DOM memory structure** at once.
- It takes **a lot of memory**, plus **the processing expense of parsing all that data.**
- While dealing with **large XML files** that contain **many records**, **a more efficient method is required a stream-based mechanism**.

- The **XmlTextReader class** is the mechanism used for **reading XML data in a stream-based fashion**.
- There are **many properties and methods to this class**.

<u>Here's an example:</u>

```
Imports System
Imports System.Xml
Module Module1

        Sub Main()
                Dim objReader As XmlTextReader = Nothing
                Try
                        objReader = New XmlTextReader (“Songs.xml”)

                        while objReader.Read()
                                select Case objReader.NodeType
                                        Case XmlNodeType.Element
                                                Console.WriteLine(“<{0}>”, objReader.Name)
                                        Case XmlNodeType.Text
                                                Console.WriteLine(objReader.Value)
                                        Case XmlNodeType.XmlDeclaration
                                                Console.WriteLine(“<?xml version=‘1.0’?>”)
                                        Case XmlNodeType.EndElement
                                                Console.WriteLine(“</{0}>”, objReader.Name)
                                End select
End While
            Finally
                        if not (objReader Is Nothing) Then
                                    objReader.Clode()
                        End if
            End Try
   End sub
End Module
```

- It begins with creating a new **XmlTextReader** object in line.
- This program reads the entire file, so we construct a **while loop** beginning in line.
- XML files contains different types of nodes such as
    - ✓ Document declarations
    - ✓ Text nodes
    - ✓ Begging and ending element nodes
    - ✓ Entity references and so on.
- Each of the can be identified using the Node Type property of the **XmlTextReader** class.
- In this example, select…case block is to format the value of the node for display to the console.
- At the end of the loop, the program checks if all nodes have been read by checking whether XmlTextReader class contains a null reference (nothing).
- If so, the program closes the stream.

WRITING STREAM-BASED XML DATA:

- Just as we read XML file using XmlTextReader, we can also write XML file using XmlTextWriter class.
- The following code shows the use of XmlTextWriter class.

Here's a program to demonstrate:

```
Sub XmlWriteDemo()
        Dim writer As new **XmlTextWriter** ("Sample.xml", Nothing)

        writer.Formating = Formating.Intended

        writer.**WriteComment** ("Stream-XML Write demo")

        writer.**WriteStartElement** ("Songs")
        writer.WriteStartElement ("Song")

        writer.**WriteStartAttribute** ("", "Format", "")
        writer.**WriteString** ("MP3")
        writer.**writeEndAttribute()**

        writer.WriteStartElement ("title")
        writer.WriteString ("Plasticity")
        writer.writeEndElement()

        writer.WriteStartElement ("artist")
        writer.WriteString ("Front line assembly")
        writer.writeEndElement()
```

```
            writer.WriteStartElement (“category”)
            writer.WriteString (“Industrial”)
            writer.writeEndElement()

            writer.WriteStartElement (“length”)
            writer.WriteString (“406”)
            writer.writeEndElement()

            writer.WriteEndElement ()
            writer.WriteEndElement ()

            writer.Flush()
            writer.Close()

            Dim doc As New XmlDocument()
            doc.PreserveWhitespace=True
            doc.Load(“sample.xml”)

            Console.Write(doc.InnerXml)
End sub
```

- This functions begins with creating a new **XmlTextWriter** object in line.
- The **default behaviour** of the XmlTextWriter class is **not to include any of the white space**.
- The behaviour can be changed by setting the Formatting property as shown in line.
- A comment block to the file with **the writeComment() method** in a line.
- XML file to begin a code is **WriteStartElement()** and to end a code with **WriteEndElement(**).
- Like the beginning and ending, WriteStartElement() and WriteEndElement(), we use **WriteEndAttribute()** method in line and **WriteEndAttribute()** method in line.
- The text of the attribute is added with **WriteString()** method in line.

FORMATTING XML DATA FOR DISPLAY:

- XML is great for representing structured data, but it cant be displayed to the user in any meaningful way.
- XML is human-readable.
- It would be useful to transform XML data into a more convenient format for display.
- One can accomplish this by using XSLT (Extensible Stylesheet Transformation) and XslTransform class.

Here’s a small program to demonstrate

```
Imports System
Imports System.Xml
Imports System.Xml.Xsl

Module Module1
        Sub Main()
                Dim objXSLT As XslTransform()
                Try
                        objXSLT.Load(“songs.xsl”)
                        objXSLT.Transform(“c:\songs.xml”,”songs.html”)
                Catch E As Exception
                        Console.WriteLine(“ERROR:“ & E.Message)
                End Try
        End Sub
End Module
```

Sending Internet E-mail

The Syetem.Web.Mail namespace offers classes that support Internet E-mail messages using the Simple Mail Transfer Protocol(SMTP).

E-mail message can be sent as plain text or HTML format and can include file attachments.

The SMTP functionally included in the System.Web.Mail namespace requires an available server.

The server must grant appropriate permission to the sender of the e-mail for the message to be properly routed to its destination.

The server can include any host from LAN or on the Internet.

SMTP server with IIS 5.0 can also be included.

The System.Web.Mail namespace has three classes that provide all the SMTP functionality to send messages:

(i) SMTPMail
(ii)MailMessage (no methods)
(iii) MailAttachment (no methods)

Property of SmtpServer

SmtpServer – the host name of smtp server. If it is omitted, IIS 5.0 SMTP service naming on the local machine is used.

Method of SmtpMail Class

Send - sends the mail message

Properties of the Mail Message class

Attachments – a collection of mailAttachment objects representing attachment to send with the e-mail message

BCC – a semicolon delimited list of e-mail address that receive a blind carbon copy of the e-mail message.

BodyEncoding –The encoding type of the e-mail body

BodyFormat – The content type of the e-mail body either MailFormat.Text, MailFormat.HTML
CC

Cc – A semicolon-delimited list of e-mail addresses that receive a carbon copy (CC) of the e-mail message

From – The e-mail address if the sender.

Header – Custom headers to send along with the message.

Priority - The priority of the e-mail message: MailPriority.High, MailPriority.Low or MailPriority.Normal

Subject – The subject line of the e-mail message

To – The e-mail message of the recipient

UrlcontentBase – The base URL to use for all relative links included in the body of the e-mail message.

UrlContentLocation – Gets or sets the Content – Location HTTP header for the e-mail message.

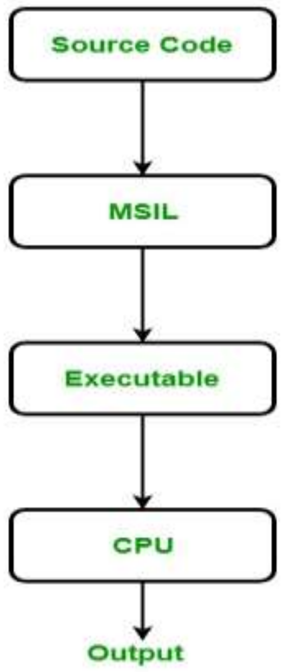
Properties of MailAttachment

Encoding – The encoding used to encode the e-mail attachment : either MailEncoding.Bse64 or MailEncoding.UUEncode.

Filename – The Pathname to the file attachment.

Unit - IV

5.1 MANAGED CODE



A code which is written to get the service of the managed runtime environment execution lie CLR(Common Language Runtime)in.NET Framework is know as Managed code . It always implemented by the managed runtime environment instead of directly executed by the operating system.

The application is written in the language like java, C#, VB.Net , etc.

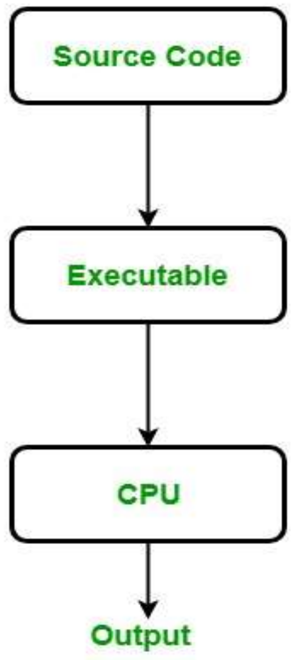
ADVANTAGE OF MANAGED CODE

- ❑ It improved the security of the application like when we use runtime environment, it automatically checks the memory buffer to guard against buffer overflow.
- ❑ It implement the garbage collection automatically
- ❑ It also provides runtime type checking/dynamic type checking.
- ❑ It also provides reference checking which means it checks whether the reference point to the valid object or not and also check they are not duplicate.

DISADVANTAGE OF MANAGED CODE

- The main disadvantage of managed language is that we are not allowed to allocated memory directly, or we cannot get the low-level access of the CPU architecture.

UNMANAGED CODE



- A code which is directly executed by the operating system is known as Unmanaged code. It always aimed for the processor architecture and depends upon computer architecture.
- In other words, whenever we want to execute the same code for the different architecture we have to recompile that code again according to that architecture.
- The application written in VB 6.0, C, C++ etc always in unmanaged code.

ADVANTAGE OF UNMANAGED CODE

- It also providers direct access to the hardware.
- It provides the low-level access to the programmer.
- It allows the programmer to bypass some parameters and restriction that are used by the managed code framework.

DISADVANTAGE OF UNMANAGED CODE

- ❑ It does not provide security to the application.
- ❑ Due to the access to memory allocation the issues related to memory occur like memory buffer overflow, etc.
- ❑ Error and exception are also handled by the programmer.
- ❑ It does not focus on garbage collection.

DIFFERENT BETWEEN MANAGED CODE AND UNMANAGED CODE

Managed Code

❖ It is executed by managed runtime environment or managed by the CLR .
❖ It provide security to the application written in.NET Framework.
❖ Memory buffer overflow does not occur.
❖ It does not provide low-level access to the programmer.
❖ The source code is compiled in the intermediate language known as IL or MISL or CIL.

Unmanaged Code

❖ It is executed directly by the operating system.
❖ It does not provide any security to the application.
❖ Memory buffer over-flow may occur.
❖ It provide low-level access to the programmer.
❖ The source code directly compile into native language.

5.2 .NET THE COMMON LANGUAGE RUNTIME 5.2 .NET THE COMMON LANGUAGE RUNTIME

THE COMMON LANGUAGE RUNTIME:

•The runtime environment for the .Net frame work is called the common language runtime

•Virtual machine components of Microsoft

THE COMMON TYPE SYSTEM:

•Data types is known as the common type system.

•Data types include simple value types, classes, enumerated value types, interfaces and delegates

•Simple value data types include primitive types.

•Primitive types include integers, Boolean valuesa and strings

.Net data type class name	VB.Net data type
Byte	Byte
Int16	Short
Int32	Integer
Int64	Long
String	String
Boolean	Boolean

JUST IN TIME CODE COMPILATION:

•Just in time Compilation compiles MSIL code right at the moment.

•The JIT compiler also peeforms security checks and verifies type safety.

CODE ASSEMBLIES:

•The component in the .Net framework is the assembly.

•An assembly is a collection of files, typically .d11 files and resource files.

APPLICATION DOMAINS:

•The portected spaces are known as application domains.

•Application domains can enforce security policies.

•Consume fewer system resources is the advantage of inherit type safety of the .Net framework code.

5.3 Com + component services

- The .NET Framework leverages many existing Windows services to make it a more robust application environment.
- A particular technology that deserves attention is COM+ Component Services.
- These technologies were the predecessors to the .NET Framework. To see how COM+ Component Services fits into the .NET Framework arena, let's explore a little about these technologies.

Overview of com

- COM objects are roughly equivalent to normal classes, but COM defines how these objects interact with other programs at the binary level.
- By binary to the compiled code, with all the methods and member variables of the class already built into an object.
- This "binary encapsulation" allows you to treat each COM object as a "black box’.
- This "binary encapsulation" is used treat each COM object as a "black box."

- In the Windows environment, these binary objects (COM objects) are packaged as either DLLs or executable programs.
- COM is also backed by a series of utility functions that provide routines for instantiating COM objects, process communication, and so on.
- COM was the first methodology to address object-oriented software reuse. COM has enjoyed great commercial success; many third-party software vendors provide COM objects to perform a wide range of tasks, from e-mail to image processing

Overview of transaction

- a transaction is a unit of work.
- The success or failure of the transaction depends on whether or not all of the smaller steps are completed successfully.
- If a failure occurs at any point during a transaction.
- A transaction is committed when all steps have succeeded. A failed transaction causes a rollback to occur (the "undo" operation).

ACID principles Atomicity, Consistency, Isolation, and Durability.

- Atomicity means that either the operation that the component performs is completely successful or the data that the component operates on does not change at all. This is important because if the transaction has to update multiple data items.
- Consistency deals with preserving the system state in the case of a transaction failure.
- Isolation means that a transaction acts as though it has complete control of the system. this means that transactions are executed one at a time. state consistent two components executed at the same time that operate on the same data can compromise the integrity of the system.
- Durability is the ability of a system to return to any state that was present before the execution of a transaction.

Example:

- Get the amount to be transferred, and check the source account for sufficient funds.
- Deduct the transfer amount from the source account.
- Get the balance of the destination account, and add the amount to be transferred to the balance.
- Update the destination account with the new balance.

Automatic transactions

- Many database systems include internal support for transactions.
- Such database systems contain native commands to begin, abort, and commit transactions.
- Using a database's internal transaction-processing system is referred to as manual transaction processing.
- automatic transactions are controlled by a system external to the database management system (DBMS).
- Earlier versions of Windows (95/98/NT) provide automatic transaction services using Microsoft Transaction Server (MTS).

- MTS works by coordinating database updates made by COM components grouped into a logical unit called a package.
- Each component in the package participates in the transaction.
- MTS then makes a determination to continue based on the success of the last component's signal of success or failure.
- If the transaction step was unsuccessful, the transaction is aborted immediately.
- If all steps execute successfully, MTS commits the transaction and tells the DBMS to commit changes to the data.

COM+ Application

- The release of window 2000 came the next version of COM And COM+.
- Performance enhancing features including load balancing.
- Which uses microsoft message Queue Server to handling request for COM+ components.
- Individual COM+ components in an Applications can be assigned different level of involvement in an automatic transformation.

COM+ Automatic Transaction Support Levels

Transaction Support	Description
Disabled	No transaction services are ever loaded by COM+ Computer Services.
Not Supported	This is the default setting for new MTS components Execution of the component is always outside a transaction regardless of whether or not a transaction has been initiated for the component.
Supported	You may run the component inside or outside a transaction without any ill effects.
Required	The component must run inside a transaction.
Required New	The component needs its own transaction in which to run. If the component is not called from within a transaction, a new transaction is automatically created.

COM+ Security

- Security of paramount importance, especially for application in ended to run on the internet.
- COM+ component service provides a security infrastructure for applications that uses to window 2000/XP users and groups.
- A role is a defined set of duties performed by particular individual.
- The role based security is not only easy to implement.
- This type of security is implemented programmatically using special .NET.

. Net classes and COM+ components Services

- Thus far. discussions about COM+ component service, transactions and Security deal specifically with COM+ components.
- COM+ still remain a dominate a technology and is a signification part of windows.
- The architecture for. Net managed components was designed to take advantage of all the futures COM+.

Building Web Services - Lesson 6

Standards – Based Functionality (XML and HTTP)

- The framework for web services in founded in open Internet standards; no part of web services uses proprietary technology.
- HTTP is the substrate for all communications between components and clients of those components.
- HTTP network traffic flows across firewalls since it uses the same network port as other web traffic.
- The payload of these communications is contained inside a special XML data format called the Simple Object Access Protocol(SOAP)
- Web services form a communication method that allows components to communicate with clients and each other.
- Components using different architecture can talk with each other using web services.

Separation of Data from presentation

- Websites are used by many people for day to day task.
- Though the information are provided to the user by mouse clicks, the limitation is that the data is tied to an HTML page, which in turn to the computer and a web browser.
- As the demand grows devices such as PDA, cell phones , notebook computers , home refrigerator have wireless capabilities and can access internet sites.
- As the size of the screen is small, only few lines of text can be displayed and they cannot display graphics. This makes the traditional browsing impossible.
- Traditional applications that attempt to use services provided by websites also has limitations.
- If the websites are exposed that look functionality as the web service, it would return only the relevant data.

Overview of Web Services

Def:
web services are compiled classes that are contained within .asmx file.

The .asmx File

- The .asmx file contains code, such as Visual Basic, that is compiled into a class.
- This file contains a special page directive and identifies the file to the compiler as a web service.
- Another page directive specifies the name of the web service class.

Eg:

```
<%@ webservice language =“VB” class =“sampleweb” %>
Imports System.Web.Services
```

Web Service classes and web methods:

Web service begins with class definition

```
Public Class samweb
                        Inherits System.Web.Services.WebService

<WebMethod> Public Function jhe() As String
                                    jhe= “Hello”
                         End Function

End Class
```

Explanation

The class starts by inheriting from System.Web.Services.WebService - is the class which supports web services.

It exposes ASP.NET intrinsic object such as those used for session and state management

<WebMethod> - Specifies that the member function will be publicly exposed through the web service.
This attribute class allows to define an optional description for the web method and specify additional parameters that can alter the behaviour of the web method.

Web Service Description Language

Web Services have the ability to describe themselves, ie, they can tell clients how to call their methods , what the parameters and types are and what data types to expect as return values from those methods.

This is done by describing the web service through an XML data format called web service description language(WSDL)

WSDL describes the methods of a web service , the data type definitions, the parameter definitions of each method the supported wire formats for calling the web service, the URL where the web service is located and other supporting information.

Web Service Discovery:

Discovery files, provide information about the location of web services and their corresponding WSDL.

The web service framework provides two method for discovering web services

Static discovery and dynamic discovery

Static discovery

It allows to explicitly define the location of individual web services.

It allows to group together the location of web services into a logical unit.

This is stored in a XML file with extension .disco extension

Managing state in web services

Web services provide the same state-management capabilities.

Web services can use ASP.NET state management using the Session and Application objects

Web service class contains many of the same objects that ASP.NET uses.

httpSessionState class is used to maintain a count of how many times the web service is used during a session.

Using Transactions in web services

Web services can participate in a COM+ component services transaction by specifying a TransactionOption in the WebMethod() attributee

The two preliminary requirements for using a transaction in a web service are

1. Include import statements such as

```
Imports System
Imports System.Web.Services
Imports System.EnterpriseServices
```

2. The transaction option must be set for web methods that will execute as part of a transaction.

The RequiresNew option will begin the new transaction.
When the code of the web method finishes executing, it will automatically commit or roll back the transaction based on whether or not an exception is thrown inside web method.

ACCESSING DATA WITH ADO.NET

OVERVIEW OF DATA ACCESS ON THE WEB

Data access technologies originates from file based data stores to ADO.NET .

Flat files
Before relational database management systems(DBMS) were widely used, data were stored in flat files.

Flat files refers to any kind of data existing in files and they are not a part of DBMS.

They are also called as text files.

Fixed - length files store their data in records separated by a terminating character.

Each element of data in the record occupies to fixed number of characters, padded with white-space characters as needed to fill the fixed length.

Flat files are not very flexible as a data-storage solution for several resons

- Flat files limit the kind of data that can be stored in them since, they can contain only textual data
- Flat files don't work well in multiuser situation because the operating systems typically limit the amount of streams that can be opened on a file.
- Problems with dat integrity arise out of concurrent updates with flat files.

Legacy or mainframe data

Some companies wants to keep pace with the web revolution. So they need a way to expose the data to the web in a way that is minimally intrusive.

This process is known as screen scraping, it retrieves data from a virtual terminal session with the mainframe.

This virtual terminal session is triggered by server side code invoked from a web page.

The program controlling the terminal session, in turn passes the "scraped" data back to the browser.

Proprietary Database API's

Database systems are shipped with an API that was specific to that particular database system.

Standardization of data access was made using SQL.

SQL is the primary language used to describe queries to a database but the underlying programming models used to access data from application code are fragmented.

Some language compilers support embedded SQL, which allows programmers to place SQL Statements inline with the host language.

STANDARD API'S

Standard API for data access was done with the help of Open Database Connectivity(ODBC).

For a database to be ODBC complaint, the database vendor should provide ODBC Driver software that translated the ODBC API calls into the native database query language.

Drawbacks of ODBC

1. ODBC is designed to be used only with C/C++.

2. It requires many calls to the ODBC API to perform the simplest operations.

With the introduction of COM and its language independent architecture, a new technology called OLE DB came into existence.

OLE-DB communicates with a data provider , which is the software that provides access to the physical data.

The data provider enables programmers to write OLE – DB Code which communicates with ODBC data sources using the COM methodology.

It provides fastest access to data.

ADO
It provides an easy programming interface to data sources.

It serves as a wrapper around OLE-DB to make programming easier.

It unifies the functionality of previous data access methods and reduces the number of objects in the programming interface.

It is also referred as "universal data access"

ADO .NET : The next generation of data access technology

Flexibility in a mixed platform environment :

ADO passed data to its component using TCP/IP protocol. But ports were blocked by firewall. This hampered interoperability, which is the key platform behind .NET platform.

In ADO.NET, XML,HTTP are used to work around the interoperability problem.
Data records are encoded and transported using XML and HTTP.

XML is the format of choice for representing structured or hierarchical data

HTTP can easily ferried across firewalls along with the normal web traffic.

XML is a open standard , so even non-Microsoft platform can be used to retrieve data.

ADO.NET Programming Objects and Architecture

The architecture of ADO.NET is divided into two areas :
(i) Data set class and (ii) .NET managed provider

Data set class
It includes tables, columns, views and keys.
It has the ability to track changes made to itself
It doesn't have active connection with the data base

It has various objects. The major ones are

(i) The DataTable and DataTableCollection objects
- DataTable object is tabular representation of data.
- It has rows and columns .
- Like database it has the primary key.
- There is no limitation to the number of DataTable objects that a DataSet object.
- All the DataTable objects and DataSet object are housed in the DataTableCollection object.

(ii) The Datarow and DatarowCollection objects
- Each row in a DataTable object is represented by a DataRow object.
- The DataRow object contains the data that belongs to a particular record.
- All of the DataRow object are kept in the DataRowCollection object

(iii) The Datacolumn and DatacolumnCollection objects

- Columns are represented using DataColumn object
- DataColumn objects can be used to refer to a specific field in a DataTable or DataRow.
- All the DataColumn objects reside in the DataColumnCollection Object

(iv) The DataView Object

- A DataView object serves several functions.
- It provides a view of the data in the DataTable object
- This view can contain a sorted and filtered version of the data.

(iv) The DataRelation and DataRelationCollection objects

- DataRelation objects are used to represent foreign key relationships among DataTable objects.
- DataRelationCollection object contains all the DataRelation objects in the DataSet.

THE .NET MANAGED DATA PROVIDER

DataSet object is used to manipulate data that originated from a variety of formats. The .NET Managed Data provider handles connecting to a database, execution of queries and retrieval of results. These result populate the DataSet object.

.NET Managed Data provider is made up of four base classes. They are:

(i) The Connection class : it is used to provide a connection to a database and controlling manual transaction.
(ii) The Command class : this object executes queries and stored procedure.
(iii) The DataReader class: this object provides fast, forward only, read only access to data. This is useful to process a large number of records from a database but don't need to update the data or have random access to it.
(iv) The DataAdapter class: This object's job is to populate DataSet objects with data retrieved from the database.

.NET Framework ships with the following Data Providers:

(i) The SQL Server .NET Managed Data Provider: This provider is used for connecting to Microsoft SQL Server 7.0/2000 and Microsoft MSDE (Microsoft Data Engine)database servers.
(ii) The OLE-DB .NET Managed Data Provider: This provider is designed to connect to Microsoft SQL Server 6.5 or earlier Oracle and Microsoft Access.
(iii) The ODBC .NET Managed Data Provider : This driver is used when connecting to database systems that provide only in ODBC driver as an interface.

Displaying Database Data

- IDataReader Interface is used read the data from a single database.

IDataReader Interface
(System.Data.IDataReader)

The IDataReader is an interface(an abstract class). So it contains no implementation of its own.

It is a part of System.Data namespace. Some of the other interface included in this namespace is IDataAdapter,IDbConnection, and IDbCommand.

ADO.NET allows the .NET Managed data provider code to implement the various interfaces.
For Eg:
    The SQL Server .NET Managed Data Provider provides an implementation of the IDataReader interface with a class in the System.Data.SqlClient namespace called SQLDataReader database

Properties and methods of IDataReader interface

Properties

1. Depth - sets the depth of nesting for the current row
2. IsClosed – has a value True if the data reader is closed
3. RecordsAffected – specifies the number of rows affected by the execution of the SQL statement.

Methods

1. Close –closes the IDataReader
2. GetSchemaTable – Returns a DataTable object with schema data
3. NextResult – gets the next result when reading the results of batch SQL statements.
4. Read – advances to the next record.

The SQLConnection class handles connection to an SQL Server.
This class implements the IDBConnection interface for the SQL Server .NET Managed Data Provider.
SQL Server connection strings typically supply a host name (data source), a database name (Initial Catalog) , a user name (uid) and password (pwd).

Open the connection to the database using open() method.
After the connection is successfully established the query is run by SqlCommand object, a class that implements the IDbCommand interface, with the CreateCommand() method.

In SQL query is specified by setting the CommandText property.

The records are read by using SqlDataReader object

The read() method is used to read a record
The GetString() method is used to read the field value as a string.
The Close() method cleans up the SqlDataReader object.

Properties of the SqlConnection class

ConnectionString – specifies the connection string to be used for opening an SQL Server database
ConnectionTimeout - sets the maximam time(in seconds) to wait for successful connection
Database – specifies the name of the database to which to connect
DataSource – specifies the SQLinstance to which to connect.
PacketSize – sets the size of network packets to use when communicating with SQL Server
ServerVersion –specifies the Version of the server
State – gets the current state of the connection : BROKEN, CLOSED,CONNECTING,EXECUTING,FETCHING OR OPEN.
WorkstationId – specified the identifier of the database client.

Methods of SqlConnection class

BeginTransaction – starts a database transaction
ChangeDatabase – changes the database on an open connection
Close – closes the connection to the database
CreateCommand – creates a new SqlCommand object for the current connection
Open – opens the database connection using the connection string.

Properties of the SqlCommand class

CommandText - identifies the SQL/Transact – SQL statement to execute
CommandTimeout – sets the maximum time to wait for the completion of a command
CommandType – determines how to interpret the CommandText string : as StoredProcedure, TableDirect, orText

Connection – specifies the SqlConnection to use for excecution of this command
DesignTimeVisible – has a value of True if the object is visible on a windows forms designer control
Parameters – returns the SqlParameterCollection
Transaction – specifies the SqlTransaction object to use for the command
UpdatedRowSource – determines how command results are applied to the DataRow object when used by the Update() method of the DbDataAdapter.

Working with Command Parameters

Command can be constructed in two ways they are :
(i) supplying a formatted string to CommandText. This is done by taking combination of values form variables, converting them to strings, and concatenating them to build a Transact-SQL statement. This is applied for simple queries. When queries become complex they are prone to errors.

(ii) Constructing commands using SqlParameter objects to supply parameter to Transt-SQL statements.

SqlParameter objects:

SqlParameter Class

Namespace:
System.Data.SqlClient
Assembly:
System.Data.SqlClient.dll

Represents a parameter to a SqlCommand and optionally its mapping to DataSet columns. This class cannot be inherited.

Whenever a name is specified for the ParameterName property of the SqlParameter object it is in the following form :

@parameter_name

After creating the SqlParameter object we need to assign values to the class properties.
ParameterName property assigns the corresponding parameter name specified in the CommandText string.
DbType is set with the data type of the parameter

Displaying data in a DataGrid control

The DataGrid control works with template columns, which allow to specify exactly how the data items will appear in the grid.

The template column can contain child controls, eg, Label, textbox etc.,

If template column is used, then the AutoGenerateColumns property is set to false.

DataKeyField states which field is to be used as key for updating edited rows.

DataSource property is set with ExecuteReader() method of IDataReader interface.

Editing Data in the DataGrid Control

To make DataGrid editable three event handlers are written for three events they are (i) EditCommand (ii) CancelCommand   and (iii) UpdateCommand

EditCommand event

Syntax

```
   Private Sub MyDataGrid_EditCommand( ByVal Source As Object, ByVal e As
DataGridCommandEventArgs) Handles MyDataGrid.EditCommand
```

The Handles keyword is used to designate which event is written for event handler.

Event handling function consists of the Object parameter and the DataGridCommandEventArgs parameter.

Which row is going to be edited can be determined by the specifics in the DataGridCommandEventArgs parameter.

The DataGrid can be set to edit mode by assigning the EditItemIndex property of the DataGrid Control to the index of the row that was selected.

CancelCommand
Canceling an edit is done by setting the EditItemIndex to -1

UpdateCommand
the parameters which is to be updated is attached to the SQL UPDATE command and then the ExecuteNonQuery() method is used to execute the update.

Working with the DataSet and DataTable objects

Data access is an operation that involves connecting, retrieving and updating database.

New concepts are introduced in ADO.NET for data access. The use of system resource is decreased by this new concept.

1. A snapshot of the database data is taken.
2. The data is transferred to the client.
3. Then the connection is disconnected immediately.

4. The client then proceeds to query and update this downloaded data.
5. When the changes have been completed, the client uploads the changed data to the server.
6. The server then reconciles any changes to the database automatically.

ADO.NET can represent table, columns,keys and relations in memory.

The DataSet Class Summary

Properties of DataSet class

1. CaseSensitive – has the value of True if string comparisons within the DataSet object are case-sensitive.
2. DataSetName – specifies the name of the DataSet object
3. DefaultViewManager – returns a DatViewManager object used for filtering, searching and navigating records
4. EnforceConstraints – has a value true if constraint rules are to be considered when attempting any update operation.
5. ExtendedProperties – returns a collection of custom information supplied by the user
6. HasErrors - has a value of true if there are errors in any of the rows of the DataSet object
7. Locale – specifies the geographic region and rules to govern DataSet operation in CultureInfo object
8. Namespace – specifies the namespace of the DataSet object.

Maintaining Data Integrity with the Data Relation class

The tables in the database are linked to each other by foreign key and constraints.
In ADO.NET it is accomplished using DataRelation class and Constraint class.

An important functionality in database system is to maintain Data Integrity.
Data in the database are expected to be valid for the type of objects modeled.

Properties of DataRelation class

1. ChildColumns – Returns DataColumn object of this relation
2. ChildKeyConstraint – Gets the ForeignKeyConstraint for the relation
3. ChildTable – returns the child table of this relations
4. DataSet – returns the DataSet object to which the DataRelation object belongs
5. ExtendedProperties – returns the customized property collection
6. Nested – has a valur of True if DataRelation objects are needed
7. ParentColumns – returns an array of DataColumn objects that are the parent columns of this DataRelation object
8. ParentKeyConstraint – returns the UniqueConstraint of the parent column
9. ParentTable – returns the parent DataTable object of this DataRelation object
10. RelationName – specifies the name of the DataRelation object

Methods of DataRelation Class

CheckedStateForProperty - checks for a valid DataRelation object.

Using Manual Database Transactions

Automatic Transactions is provided in COM+ component service

ADO.NET provides manual transactions using SqlTransaction class and the SqlConnection class.

Properties of SqlTransaction Class

IsolationLevel – specifies the IsolationLevel for this transaction:
chaos, ReadCommitted,ReadUncommitted, RepeatableRead,Serializable, or Unspecified(IsolationLevel enumeration)

Methods of the SqlTransaction class

1. Commit – commits the database transaction
2. Rollback – Rolls back the current transaction

Syntax

1. Rollback()
2. Rollback(ByVal transactionName As String)

Save - creates a svepoint in the transaction to which a roll back can return

Save(ByVal savePointName As string)

Working with Typed DataSet objects

One of the powerful features of the.NET framework lies in the exgensibility of the class libraries.

Many of the classes in the .NET framework can be extended through inheritance.

The DataSet is one such class. By creating a new application- specific class that inherit from the DataSet class, can access table, column, and row information in a strongly typed fashion(the object can be referenced by name rather than using a collection based index.

Productivity enhancement can achieved using VS.NET IDE. The editor offers help through Intellisense with method property names for the typed DataSet object.

Securing .NET Applications

Protecting the site comes down to two tasks for a site administrator

1. Provide a way to restrict site access so that only authenticated users can visit the site.

2. Make sure that the data transmitted between the web user and the web server cannot be intercepted by a third party.

A secure system involves implementing the following processes.

Authentication:

It is the process by which the system determines a user's identity. The most common type of authentication is the combination of user name and password. Other means of authentication includes biometric devices, retinal scans, voice prints etc. It is the first step in any diligent security process.

Authorization:

It is the process of determining which resources an authenticated user can access and how those resources can be used. Users in a multiuser system have varying levels of privileged use assigned to them. The most common type of authorization deals with how files can be accessed.Eg: some person may be able to read a particular file but not write to it.

Impersonation :
It is the process of taking a user’s identity and having it assume the identity of another user or entity. This simplifies the work required in the authentication of public servers. It removes the need to assign user accounts to anonymous users since you have a shared account for public use.

Windows Security

The windows operating system is designed in such a way that any action you perform on the computer involves a security check.

It follows that whenever the user is working inside the windows environment and the user should have an identity.

This is the user account. These user names are stored in user accounts database.

The user accounts database can exist in two different areas,

1. Server – whenever a user attempt to gain access to the server. The security check is performed against the database residing on the server. The accounts in this database is called as local user accounts.

2. PDC (Primary Domain Controller) - it is used to authenticate users on a local area network. The local area network may have many servers. These servers are grouped to form a domain. The PDC provides a centralized area for authentication to occur for the machines in the domain. If the server wished to gain access to a particular server’s resource within

The domain, the user need to log into PDC.

File / Object System security:

Windows can secure many different types of resources.

Any object on windows system contains priveleges associated with it in an access control list(ACL)
The ACL contains a list of security identifiers(SID's) that correspond to a user account.
whenever a request is made , the currently logged in user is matched against the user SID in the ACL for the resource. The system then decides whether to grant access or not.

Windows NT stores files in file allocation table (FAT) method. Since windows NT has multiuser operating system , it requires security protections of a multiuser system. Different security attributes are assigned to files which includes the users who are allowed to access which files and in what ways(read, write and execute)

User Rights, Groups and Policies

IIS Authentication and Authorization Security

IIS implements its own authentication mechanisms and controls how to authenticate users.

Anonymous Access
IIS provides the ability for an anonymous user to access files on the web server without having to supply his or her credentials.

By default, IIS   enables this type of access. When it is not enabled, the user's web browser would pop up a dialog box asking for a user name and password.

With anonymous access enabled, IIS performs retrieval of files by using special windows user account , called as web guest. It has privileges to access files. This account is created upon installation of IIS. It is named IUSR machine name, where machine name is the name of the windows server.

The password is also configured automatically,making the entire authentication process transparent to the user.

The following are the reasons why the anonymous users do not gain access

1. The IUSR_machine name account is assigned guests permissions only. Which indicates that the anonymous access account is a member of default windows guest account.

2. It has limited permission only.

3. The users in the guest group has read permission only. The user can view the file in the web server.

4. Execute permission is given if .aspx files need to be executed.

IIS allows web users to view files in the virtual directory spaces only. A web user cannot specify a URL that points to a directory outside of this virtual directory space to gain access to a file.

Using cracking techniques of If the unauthorized user does know the path of a protected file, the permission set on the file will not include accounts in the guest group. Thus, the system will deny access.

BASIC Authentication:

This is the security implemented in the HTTP protocol.

BASIC Authentication uses the HTTP response headers to signal that a user name and password are required to access the web page.

How it works:

1. A request is made for a document, so an HTTP GET command is issued by the browser that looks like:
   GET /passed.aspx HTTP/1.0

2. IIS sees that the file, passwd.aspx is protected from guest access. So it return a 401 status code in the HTTP response. The 401 status code indicated that password authorization is required to access the file.

3. The 401 HTTP triggers a password dialog box to appear in the user’s browser. Now the user needs to send the user name and password information. When this information is entered the username and the password are concatenated into a single string separated by a colon (:).

4. This string is then encoded using basic64 encoding algorithm.

5. The encoded string is placed inside the HTTP header with the following request.

Base64encoded string contains the Base64-encoded version of the user name and password string.

If the user name and password given match the user credentials, the user is grated access. A normal 200 status code is returned along with the contents of the request.

Though the password is encoded using Base64 encoding algorithm, it can be easily decoded using any cracking tool. The authentication used can be changed by the web server by using IIS configuration tools.

Integrated Windows Authentication:

This provides strong security.

Using this method the username and the password are not transmitted over the network. The user’s web browser does not prompt for username and password , instead the browser uses the user name and password used by the user when logging into the local workstation.

It is not the best choice for two reasons:

1. Internet Explorer is the only browser that supports Integrated Windows Authentication.
2. Many internet users have internet access only through firewall / prox. This may create problem with the network connection used for the authentication.

It is very useful in an intranet setting where persistent connection are more reliable.

Digital Authentication
It involves authentication using hashes. A hash is a fixed – length value that is derived from a chunk of data using a secret data string.

Implementing Data Encryption

The .NET Framework SDK includes classes for performing data encryption.
These classes are part of System.Security.Cryptography namespace.
RC2CryptoServiceProvider object provides a wrapper for the windows cryptographic server provider.

Properties of RC2CryptoServiceProvider class

BlockSize – gets or sets the block size (in bits) of the cryptographic operation

EffectiveKeySize – gets or sets the effective size O(in bits) of the secret key used by the RC2 algorithm,

FeedbackSize – gets or sets the feedback size (in bits) of the cryptographic operation

IV – gets or sets the initialization vector for the symmetric algorithm.

Key – gets or sets the secret key for the symmetric algorithm

KeySize – gets or sets the size of the secret keyused by the RC2 algorithm

LegalBlockSizes –gets the block sizes supported by the symmetric algorithm

LegalKeySizes – Gets the key sizes that are supported by the symmetric algorithm.

Mode – gets or sets the mode for operation of the symmetric algorithm

Padding – gets or sets the padding used in symmetric algorithm

KeySize property is inherited from the RC2 and the other properties are inherited from symmetric algorithm.

The System.Security.Crytography classes inherit functionality from other classes.

Eg:
RC2CryptoServiceProvider class inherits much of the functionality from the base class RC2 which inturn inherits from SymmetricAlgorithm class.

The base classes are provided to organize cryptographic functionality.

Methods of the RC2CryptoServiceProvider Class

CreateDecryptor – Creates a symmetric RC2 decryptor object.

CreateEncryptor – Creates a symmetric RC2 encryptor object.

GenerateIV – generates a random initialization vector to be used for algorithm.

GenerateKey – generates a random key to be used for the algorithm.

ValidKeySize – determines whether the specified key size is valid for the current algorithm.

ICryptoTransform interface defines the basic operations of cryptographic transformation.
The members in this interface performs the actual encryption and decryption on the blocks of data.

The encryption process begins by reading the contents of the input file into a byte array.

The write() method of the CryptoStream Object is used to dump the contents of the input Byte array to the output file (encrypted) file.

Properties of CryptoStream class

CanRead – gets a value indicating whether the current CryptoStream object is readable

CanSeek – gets a value indicating whether the current CryptoStream object is seekable

CanWrite – gets a value indicating whether the current CryptoStream object is writable

Length - gets the length in bytes of the stream

Position – gets or sets the position within the current stream.

Methods Of CryptoStream Class

FlushFinalBlock – updates the underlying data source or repository with the current state of the buffer, then clears the buffer.

Seek – sets the position within the current stream.

SetLengh –sets the length of the current stream

ASP.NET Authentication security

Authentication is the process of allowing certain users access to privileged resources. ASP.NET accomplishes this through authentication providers.

There are three different authentication providers Forms-Based, Windows, and Passport.

The Form-Based Authentication Provider

The Form-Based Authentication Provider uses developer-made HTML forms to collect user identification information information (user name and password).

Whenever the user requests the privileged resource for which the current use credentials do not suffice, a client-side redirection to an HTML form can be made.

The user credentials are collected for the resource at the login page.

If the login passes the user is allowed to access the resource.

The form based authentication provider works by using browser cookies.

For each request to a protected web site, ASP.NET checks to see if there is an authentication ticket attached to the request.

This authentication ticket is in the form of a browser cookie.

Upon successful login, a cookie is generated with the user’s identity.

An expiration time is set on the authentication ticket which is useful if a user logs in and then leaves his or her desk.

The time out period will expire and the user will need to repeat the login process.

The data contained in the authentication ticket is also usually encrypted , which prevents another indomitable user from accessing the contents.

Steps

1. Configure a special file called the web.config file.
2. The web.config file resides in the root of the ASP.NET application.
3. There is only one web.config file per application, and its use is optional.
4. It is a XML file used to store security settings for ASP.NET application.
5. There are two sessions they are <authentication> and <authorization>
6. The mode attribute designates which authentication provider to use.
7. In form-base authentication it is the forms.
8. Below the authentication node is the form node.
9. The name attribute is the name assigned to the authenticated ticket.
10. The path attribute specifies the virtual path for which the authentication ticket is valid.
11. A particular directory within the site or the entire site can be specified as members only by using “/”. This protects the entire site.

12. The loginurl attribute specifies where the login page is located.

13. It requires a minimum of two text fields one for user name and one for password.
14. The protection attribute specifies how much security is applied for authentication ticket. Eg: All :indicated that the authentication ticket should be encrypted and that data validation should be performed on the ticket based on specific optional settings.
15. Time out attribute specifies how long the authentication ticket is valid. It is expressed in minutes.

16. The <credentials> node contains the user accounts information, which are pair of username and password.

17. SHA and MD5 options are specified to encrypt passwords.

The Windows Authentication Provider

It allows the use of Windows users accounts and groups to authenticate visitors to a web site.
The web.config file is set up for the windows authentication provider.
The Web.config file allows only certain users into the site.

After setting the authentication mode to “Windows” the users and groups are designated that are allowed in to the site.
The <allow> node specifies who is allowed to access the site.
The users attribute contains a comma separated list of user account names who are allowed to access the site.
The <deny> node specifies the list of users who should not access the site.
“*” indicates that all are denied access except those who are allowed in the <allow> node.

The Microsoft Passport Authentication Provider

Services such as MSN Hotmail, the free e-mail service from Microsoft.
Along with is services such as MSN messanger, Hotmail provides another service called Microsoft Passport

Passport is a centralized authentication service that allows users to visit multiple web sites and be authenticated on each of them by logging in only once.

Using passport services requires additional software downloads and they require fee for it.