**Difference b/w stored procedure and function**

1.You cannot use stored procedure in a select statement like  
select <spname > but this is possible with UDF .  
  
2. you can not return a table type variable with stored procedure but this is   
possible with user defined functions in sql server.  
  
3. Stored procedure may or may not return a value but a function has to return   
a value.  
  
4. UDFs can accept a smaller number of parameters than stored procedures. UDFs can have up to 1024 parameters, whereas stored procedures support up to 2100 parameters. This is a relatively minor limitation because most routines require a much smaller number of parameters.   
  
  
5. UDFs cannot call stored procedures (except extended procedures), whereas stored procedures can call other procedures

1>Procedure can return zero or n values whereas function can return one value which is

Mandatory.  
  
2>Procedures can have input,output parameters for it whereas functions can have only input parameters.  
  
3>Procedure allow select as well as DML statement in it whereas function allow only select statement in it.  
  
4>Functions can be called from procedure whereas procedures cannot be called from function.  
  
5>Exception can be handled by try-catch block in a procedure whereas try-catch block cannot be used in a function.  
  
6>We can go for transaction management in procedure whereas we can't go in function.  
  
7>Procedures cannot be utilized in a select statement whereas function can be embedded in a select statement.

8>By using function we cannot change state of database but we can change state of database by using stored procedure.

9> Function can be used as table (can perform join between 2 functions) but same cannot be done with stored procedure.

it is clear that HTTPHandlers are used by ASP.NET to handle the specific requests based on extensions. HTTPModule, on the other hand, is used if we want to have our own functionality working along with the default ASP.NET functionality. There is one Handler for a specific request but there could be N number of modules for that.

Static constructors have the following properties:

* A static constructor does not take access modifiers or have parameters.
* A static constructor is called automatically to initialize the [class](http://msdn.microsoft.com/en-us/library/0b0thckt.aspx) before the first instance is created or any static members are referenced.
* A static constructor cannot be called directly.
* The user has no control on when the static constructor is executed in the program.
* A typical use of static constructors is when the class is using a log file and the constructor is used to write entries to this file.
* Static constructors are also useful when creating wrapper classes for unmanaged code, when the constructor can call the **LoadLibrary** method.
* If a static constructor throws an exception, the runtime will not invoke it a second time, and the type will remain uninitialized for the lifetime of the application domain in which your program is running.

<script type="text/javascript">

function LoadValues(me) {

{

var Email = me.email;

var Name = me.name;

var Id = me.id;

var Dob = me.birthday;

var Gender = me.gender;

alert(Email)

document.getElementById('auth-loggedout').style.display = 'none';

document.getElementById('auth-loggedin').style.display = 'block';

var data2Send = '{"fbemail": '+Email+', "fbname":'+Name+', "fbid":'+Id+',"fbname":'+Dob+',"fbname":'+Name+' }';

$.ajax({

type: "POST",

url: 'webform2.aspx/Testmethod',

data: data2Send,

contentType: "application/json; charset=utf-8",

dataType: "json",

success: function (arg) {console.log(arg) //call successfull

$("#lbltxt").text(arg);

},

error: function (xhr) {

alert(xhr); //error occurred

}

});

}

</script>

$.post("test.aspx/testMethod",

{

name: "Donald Duck",

city: "Duckburg"

}

);

In my server side (test.aspx.cs) method, I have

public void testMethod()

{

string name = Request.Form("name");

string city = Request.Form("city");

}

**Extension Method**

Help you to add new methods to existing type without modifying original code , Inheriting or aggregating.

If we need to add a method in a class then we need to write below code.

Extension method has to be ststic

Public static class SomeMoreMethod

{

Public Static int Subtract(this ClassName in which you need to add method, int num1, int num2)

{

Return num1-num2;

}

}

**Anonymous Method**

Anonymous methods provide a way by which we can directly write logic for delegate instead of passing a method.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Web.Services;

using System.Diagnostics;

public partial class Default6 : System.Web.UI.Page

{

public delegate int objDel(int num1, int num2);

protected void Page\_Load(object sender, EventArgs e)

{

Stopwatch sw = new Stopwatch();

objDel objDelAdd = new objDel(Add);

for (int j = 0; j < 10; j++)

{

sw.Reset();

sw.Start();

for (int i = 0; i < 1000; i++)

{

objDelAdd(5, 3);

}

sw.Stop();

Response.Write(sw.ElapsedTicks.ToString() + "<br/>");

}

Response.Write("<br/><br/>");

for (int j = 0; j < 10; j++)

{

sw.Reset();

sw.Start();

for (int i = 0; i < 1000; i++)

{

objDelAdd = delegate(int num1, int num2)

{

return num1 + num2;

};

objDelAdd.Invoke(2, 2);

}

sw.Stop();

Response.Write(sw.ElapsedTicks.ToString() + "<br/>");

}

}

public int Add(int num1, int num2)

{

return num1 + num2;

}

}

An application pool is a group of one or more URLs that are served by a worker process or a set of worker processes. Application pools set boundaries for the applications they contain, which means that any applications that are running outside a given application pool cannot affect the applications in the application pool.

Application pools offer the following benefits:

Improved server and application performance. You can assign resource-intensive applications to their own application pools so that the performance of other applications does not decrease.

Improved application availability. If an application in one application pool fails, applications in other application pools are not affected.

Improved security. By isolating applications, you reduce the chance that one application will access the resources of another application.

To answer this question in a single statement, you can use HttpContext.Current.Items for very short term storage. By Short term storage we mean that this data is valid for a **single HTTP Request**. There is a lot of confusion around regarding storing data in HttpContext.Current.Items and **storing data in Session variable**. In this blog post, I am going to describe what are the different scenarios where we can use HttpContext.Current.Items and what is the exact difference with session variable.

Items collections of HttpContext is and IDictionary key-value collections and that are shared across a singleHTTPRequest. Yes, HttpContext.Current.Items is valid for a single HTTPRequest. Once after processing, server information is sent back to the browser, the variables that were set in the Items[] collection will be lost. Whereas for Session variable, the information is valid for multiple requests as this is user specific. The sessionvariable only expires either on Session Time Out or explicitly clears the values.

Let’s have a quick look at how we can store information in HttpContext.Current.Items:

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/146455/When-can-we-use-HttpContext-Current-Items-to-store)

HttpContext.Current.Items["ModuleInfo"] = "Custom Module Info”

And retrieve it like:

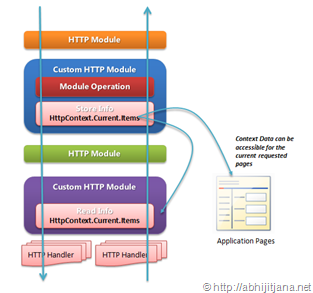
http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/146455/When-can-we-use-HttpContext-Current-Items-to-store)

string contextData = (string)(HttpContext.Current.Items["ModuleInfo"]);

As I said, HttpContext.Current.Items stores data for a very limited time period, then when can we use this? Yes, this is extremely useful when we want to share content between HttpModule and HTTPHandler.

[](http://abhijitjana.files.wordpress.com/2011/01/image9.png)

Because each and every client request passes through the **HTTP Pipeline** and**HTTP Pipeline consists** of **HTTP Module** and **HTTP Handler**. So If you are writing one custom HTTP Module by Implementing IHttpModule and you want pass some information from this module to the current requested page or any other module, you can use the HttpContext.Current.Items to store the data.

[](http://abhijitjana.files.wordpress.com/2011/01/image10.png)

Similarly, you use HTTPContext Items collection when you are sharing the same information across the different instance based on the user request and that request could be changed for a different request.

While using Context Items collections you need to keep one things that, Items collections holds the objects, so you need to do proper type casting while retrieving it.

To summarize, in ASP.NET HttpContext.Current.Items allows us to hold information **for a single request**. We can use it to store short term information. Storing such kind of information is extremely helpful to send information across different custom modules or to requested pages. You have to make sure that the data you are using inHttpContext.Current.Items is only valid for that current request and data should be flashed out automatically when request is sent to a browser for any new request you have to store the data again. Whereas session variable is valid for every request unless session timeout is not reached or we explicitly clear the session.

HttpContext is the object that contains all the information about the particular, individual HTTP request that is currently running. That is the request when this HttpContext is accessed. For example when you use Request property on Page, you are actually using System.Web.HttpContext.Current.Request that is the Request related stuff bundled with the current HTTP request. System is such that when IIS receives request and forwards it to be handled by ASP.NET, the HTTP runtime in ASP.NEt resolves the request to determine the proper HTTP handler or Http handler factory that is invoked. Then HttpModules perform preprocessing of the request and after that Http handler will process the request. After that some HTTP modules perform postprocessing of the request and finally response is sent to the client. One of the first object HTTP runtime creates after request is received, is the HttpContext. After that HTTP runtime makes it available for other components in the processing like the Page. So HttpContext is key link object in delivering request related data over the whole request processing (that involves HTTP runtime, HttpApplication (where HTTP modules reside and do their work), and alsoy the System.Web.UI.Page in case of ASP.NEt Pages to process the request))

dynamic test = 1;

var test2 = 2;

If I hover my mouse over the “var” in the code above, IntelliSense will show me that the compiler has correctly inferred that it is an Int32.  If I hover over “dynamic”, it will continue to be typed as “dynamic” since dynamic types aren’t resolved until runtime.

However, var is statically typed, and dynamic is not.

// Can a dynamic change type?

dynamic test = 1;

test = "i'm a string now"; // compiles and runs just fine

var test2 = 2;

test2 = "i'm a string now"; // will give compile error

This is one of the key differences between dynamic and var.  A var is an implicitly typed variable that is inferred by the compiler, but it is just as strongly typed as if you had explicitly typed it yourself using “int test2 = 2;”.  A dynamic variable bypasses all compile-time type checking and resolves everything at runtime.

I’ll comment out the last line in the code above to get the code to compile, and add some code to verify the types of the variables.

// Can a dynamic change type?

dynamic test = 1;

Console.WriteLine("Dynamic as " + test.GetType() + ": " + test);

test = "i'm a string now"; // compiles and run just fine

Console.WriteLine("Dynamic as " + test.GetType() + ": " + test);

var test2 = 2;

//test2 = "i'm a string now"; // will give compile error

Console.WriteLine("Var as " + test2.GetType() + ": " + test2);

This produces the following output:

Dynamic as System.Int32: 1   
Dynamic as System.String: i'm a string now   
Var as System.Int32: 2

#### Int32.parse(string)

Int32.Parse (string s) method converts the string representation of a number to its 32-bit signed integer equivalent. When s is a null reference, it will throw ArgumentNullException. If s is other than integer value, it will throw FormatException. When s represents a number less than MinValue or greater than MaxValue, it will throw OverflowException. For example:

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/32885/Difference-Between-Int32-Parse-Convert-ToInt32-and)

string s1 = "1234";

string s2 = "1234.65";

string s3 = null;

string s4 = "123456789123456789123456789123456789123456789";

int result;

bool success;

result = Int32.Parse(s1); *//-- 1234*

result = Int32.Parse(s2); *//-- FormatException*

result = Int32.Parse(s3); *//-- ArgumentNullException*

result = Int32.Parse(s4); *//-- OverflowException*

#### Convert.ToInt32(string)

Convert.ToInt32(string s) method converts the specified string representation of 32-bit signed integerequivalent. This calls in turn Int32.Parse () method. When s is a null reference, it will return 0 rather than throw ArgumentNullException. If s is other than integer value, it will throw FormatException. When srepresents a number less than MinValue or greater than MaxValue, it will throw OverflowException. For example:

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/32885/Difference-Between-Int32-Parse-Convert-ToInt32-and)

result = Convert.ToInt32(s1); *//-- 1234*

result = Convert.ToInt32(s2); *//-- FormatException*

result = Convert.ToInt32(s3); *//-- 0*

result = Convert.ToInt32(s4); *//-- OverflowException*

#### Int32.TryParse(string, out int)

Int32.Parse(string, out int) method converts the specified string representation of 32-bit signed integer equivalent to out variable, and returns true if it is parsed successfully, false otherwise. This method is available in C# 2.0. When s is a null reference, it will return 0 rather than throw ArgumentNullException. If s is other than an integer value, the out variable will have 0 rather than FormatException. When s represents a number less than MinValue or greater than MaxValue, the out variable will have 0 rather than OverflowException. For example:

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/32885/Difference-Between-Int32-Parse-Convert-ToInt32-and)

success = Int32.TryParse(s1, out result); *//-- success => true; result => 1234*

success = Int32.TryParse(s2, out result); *//-- success => false; result => 0*

success = Int32.TryParse(s3, out result); *//-- success => false; result => 0*

success = Int32.TryParse(s4, out result); *//-- success => false; result => 0*

Convert.ToInt32 is better than Int32.Parse since it returns 0 rather than an exception. But again, according to the requirement, this can be used. TryParse will be the best since it always handles exceptions by itself.

**Type Of Function in Sql Server**

There are three types of User-Defined functions in SQL Server 2000 and they are Scalar, Inline Table-Valued and Multi-statement Table-valued.

# [User-Defined Functions](http://sqlhints.com/2011/08/14/user-defined-functions/)

[August 14, 2011](http://sqlhints.com/2011/08/14/user-defined-functions/)[Sql Server](http://sqlhints.com/category/sql-server/), [User Defined Function](http://sqlhints.com/category/sql-server/user-defined-function/)[Inline Table-Valued User Defined Function](http://sqlhints.com/tag/inline-table-valued-user-defined-function/), [Multi-statement Table-Valued User Defined Function](http://sqlhints.com/tag/multi-statement-table-valued-user-defined-function/), [Scalar User Defined Function](http://sqlhints.com/tag/scalar-user-defined-function/), [Sql Server](http://sqlhints.com/tag/sql-server/), [UDF](http://sqlhints.com/tag/udf/), [User Defined Function](http://sqlhints.com/tag/user-defined-function/)[Basavaraj Biradar](http://sqlhints.com/author/bbiradar/)

In this Article we will learn about User-Defined Functions (UDFs) in Sql Server. All the examples in this article uses the pubs database.

There are three Types of UDFS in Sql Server:  
1. Scalar  
2. Inline Table-Valued  
3. Multi-statement Table-Valued

Let us go through each of these with examples:

**1.  Scalar User-Defined Function**

A Scalar UDF can accept 0 to many input parameter and will return a single value. A Scalar user-defined function returns one of the scalar (int, char, varchar etc) data types. Text, ntext, image and timestamp data types are not supported. These are the type of user-defined functions that most developers are used to in other programming languages.

Example 1: Here we are creating a Scalar UDF AddTwoNumbers which accepts two input parameters @a and @b and returns output as the sum of the two input parameters.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | CREATE FUNCTION AddTwoNumbers  (  @a int,  @b int  )  RETURNS int  AS  BEGIN  RETURN @a + @b  END |

Once the above function is created we can use this function as below:

|  |
| --- |
| PRINT dbo.AddTwoNumbers(10,20)    --OR    SELECT dbo.AddTwoNumbers(30,20) |

Note: For Scalar UDFS we need to use Two Part Naming Convention i.e. in the above two statements we are using **dbo.AddTwoNumbers.**

Whether Below statement is correct? No, because it is not using two-part naming convention. Try executing the below statement it will error out…

|  |
| --- |
| PRINT AddTwoNumbers(10,20) |

**2.  Inline Table-Valued User-Defined Function**

An inline table-valued function returns a variable of data type table whose value is derived from a single SELECT statement. Since the return value is derived from the SELECT statement, there is no BEGIN/END block needed in the CREATE FUNCTION statement. There is also no need to specify the table variable name (or column definitions for the table variable) because the structure of the returned value is generated from the columns that compose the SELECT statement. Because the results are a function of the columns referenced in the SELECT, no duplicate column names are allowed and all derived columns must have an associated alias.

Example: In this example we are creating a Inline table-valued function**GetAuthorsByState**which accepts state as the input parameter and returns firstname and lastname  of all the authors belonging to the input state.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | USE PUBS  GO    CREATE FUNCTION GetAuthorsByState  ( @state char(2) )  RETURNS table  AS  RETURN (  SELECT au\_fname, au\_lname  FROM Authors  WHERE state=@state  )  GO |

We can use the below statement to get all the authors in the state CA.

|  |
| --- |
| SELECT \* FROM GetAuthorsByState('CA') |

**3. Multi-statement Table-Valued User-Defined Function**  
A Multi-Statement Table-Valued user-defined function returns a table. It can have one or more than one T-Sql statement. Within the create function command you must define the table structure that is being returned. After creating this type of user-defined function, we can use it in the FROM clause of a T-SQL command unlike the behavior found when using a stored procedure which can also return record sets.

Example: In this example we are creating a Multi-Statement Table-Valued function**GetAuthorsByState**which accepts state as the input parameter and returns author id and firstname of all the authors belonging to the input state. If for the input state there are no authors then this UDF will return a record with no au\_id column value and firstname as ‘No Authors Found’.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | USE PUBS  GO    CREATE FUNCTION GetAuthorsByState  ( @state char(2) )  RETURNS  @AuthorsByState table (  au\_id Varchar(11),  au\_fname Varchar(20)  )  AS  BEGIN    INSERT INTO @AuthorsByState  SELECT  au\_id,  au\_fname  FROM Authors  WHERE state = @state    IF @@ROWCOUNT = 0  BEGIN  INSERT INTO @AuthorsByState  VALUES ('','No Authors Found')  END    RETURN  END  GO |

We can use the below statements to get all the authors in the given input state:

|  |
| --- |
| SELECT \* FROM GetAuthorsByState('CA')  SELECT \* FROM GetAuthorsByState('XY') |

when using SQL Server built-in functions that return a table, you must now add the prefix :: to the name of the function.

Example from Books Online: Select \* from ::fn\_helpcollations()

A Multi-Statement Table-Value user-defined function returns a table and is also an exceptional alternative to a view as the function can support multiple T-SQL statements to build the final result where the view is limited to a single SELECT statement.

**Introduction**

User Defined Functions are compact pieces of Transact SQL code, which can accept parameters, and return either a value, or a table. They are saved as individual work units, and are created using standard SQL commands. Data transformation and reference value retrieval are common uses for functions. LEFT, the built in function for getting the left part of a string, and GETDATE, used for obtaining the current date and time, are two examples of function use. User Defined Functions enable the developer or DBA to create functions of their own, and save them inside SQL Server.

**Advantages of User Defined Functions**

Before SQL 2000, User Defined Functions (UDFs), were not available. Stored Procedures were often used in their place. When advantages or disadvantages of User Defined Functions are discussed, the comparison is usually to Stored Procedures.

One of the advantages of User Defined Functions over Stored Procedures, is the fact that a UDF can be used in a Select, Where, or Case statement. They also can be used to create joins. In addition, User Defined Functions are simpler to invoke than Stored Procedures from inside another SQL statement.

**Disadvantages of User Defined Functions**

User Defined Functions cannot be used to modify base table information. The DML statements INSERT, UPDATE, and DELETE cannot be used on base tables. Another disadvantage is that SQL functions that return non-deterministic values are not allowed to be called from inside User Defined Functions. GETDATE is an example of a non-deterministic function. Every time the function is called, a different value is returned. Therefore, GETDATE cannot be called from inside a UDF you create.

**Types of User Defined Functions**

There are three different types of User Defined Functions. Each type refers to the data being returned by the function. Scalar functions return a single value. In Line Table functions return a single table variable that was created by a select statement. The final UDF is a Multi-statement Table Function. This function returns a table variable whose structure was created by hand, similar to a Create Table statement. It is useful when complex data manipulation inside the function is required.

**Scalar UDFs**

Our first User Defined Function will accept a date time, and return only the date portion. Scalar functions return a value. From inside Query Analyzer, enter:

CREATE FUNCTION dbo.DateOnly(@InDateTime datetime)

RETURNS varchar(10)

AS

BEGIN

DECLARE @MyOutput varchar(10)

SET @MyOutput = CONVERT(varchar(10),@InDateTime,101)

RETURN @MyOutput

END

To call our function, execute: SELECT dbo.DateOnly(GETDATE())

Notice the User Defined Function must be prefaced with the owner name, DBO in this case. In addition, GETDATE can be used as the input parameter, but could not be used inside the function itself. Other built in SQL functions that cannot be used inside a User Defined Function include: RAND, NEWID, @@CONNCECTIONS, @@TIMETICKS, and @@PACK\_SENT. Any built in function that is non-deterministic.

The statement begins by supplying a function name and input parameter list. In this case, a date time value will be passed in. The next line defines the type of data the UDF will return. Between the BEGIN and END block is the statement code. Declaring the output variable was for clarity only. This function should be shortened to:

CREATE FUNCTION testDateOnly(@InDateTime datetime)

RETURNS varchar(10)

AS

BEGIN

RETURN CONVERT(varchar(10),@InDateTime,101)

END

**Inline Table UDFs**

These User Defined Functions return a table variable that was created by a single select statement. Almost like a simply constructed non-updatable view, but having the benefit of accepting input parameters.

This next function looks all the employees in the pubs database that start with a letter that is passed in as a parameter. In Query Analyzer, enter and run:

USE pubs

GO

CREATE FUNCTION dbo.LookByFName(@FirstLetter char(1))

RETURNS TABLE

AS

RETURN SELECT \*

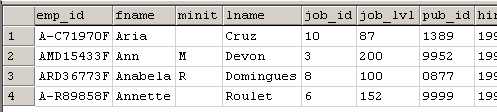
FROM employee

WHERE LEFT(fname, 1) = @FirstLetter

To use the new function, enter:

SELECT \* FROM dbo.LookByFName('A')

All the rows having a first name starting with A were returned.



The return is a Table Variable, not to be confused with a temporary table. Table variables are new in SQL 2000. They are a special data type whose scope is limited to the process that declared it. Table variables are stated to have performance benefits over temporary tables. None of my personal testing has found this result though.

1. [**difference between convert**.**tostring** and .**tostring**](http://interviews.c-sharpcorner.com/Answer/Answers.aspx?QuestionId=1216&MajorCategoryId=1&MinorCategoryId=16)

Convert.toString(.........) handles null while i.tostring() does not handles null

Int i=0;  
MessageBox.Show(i.ToString());  
MessageBox.Show(Convert.ToString(i));  
We can convert the integer “i” using “i.ToString()” or “Convert.ToString” so what’s the difference.  
The basic difference between them is “Convert” function handles NULLS while “i.ToString()”  
does not it will throw a NULL reference exception error. So as good coding practice using  
“convert” is always safe.

**Exception handling**

"*Exception* is a runtime error which arises because of abnormal conditions in a code sequence". In C# Exception is a class in the system namespace. An object of an exception is that describe the exceptional conditions occur in a code That means, we are catching an exception, creating an object of it, and then throwing it. C# supports exceptions in a very much the same way as Java and C++.

There are three ways to handle exceptions/errors in ASP.NET:

1. try-catch block. This is also called Structured Exception Handling (SEH).
2. Error Events.
3. Custom Error Page.

You will look at each one of them in detail in the next sections.

**try-catch Block**

Enclose code that accesses files, databases, and so forth inside a try-catch block because access to those resources might be denied due to various reasons causing an exception. The third part of this block is finally. It is executed irrespective of the fact that an exception has been raised. Hence, use the finally block to complete the housekeeping jobs.

As a good programming practice, always catch specific exceptions. To view the exception types supported by the .NET Framework, use the **Debug** menu and select **Exceptions** in Visual Studio.NET.

In the following code, you try to access a table that does not exist in the Northwind database; therefore, an exception is raised. By using the try catch and finally block, you handle the exception and display a message.

try

{

con = new SqlConnection("integrated security=SSPI;

data source= (local);persist security info=False;

initial catalog=Northwind");

da = new SqlDataAdapter("Select \* from TblNotExisits", con);

ds = new DataSet();

da.Fill(ds);

}

catch(SqlException ex)

{

return "Connection Unsuccessful " + ex.Message;

}

finally

{

con.Dispose();

}

return "Connection Successful";

**Using Error Events**

There are three different error events in ASP.NET that can be used in conjunction with SEH so that all exceptions are handled and the user is presented with a user-friendly error message.

1. Page\_Error: Occurs when an error occurs within the Web page. This event is in the Web form.
2. Global\_Error: Occurs when an error occurs within the application. This event is in the Gloabl.asax file.
3. Application\_Error: Occurs when an error occurs within the application. This event is in the Gloabl.asax file.

Methods in the Server object are used to handle the exception in the error events.

1. GetLastError: Gets the last exception that occurred on the server.
2. ClearError: Use this method to handle the exception and stop the error to trigger the subsequent error event or display the error to the user.

In the following code, you handle the exception in all the above three mentioned events but call the ClearError method only in the Application\_Error event so that the error is propogated to the above level.

private void Page\_Error(object sender, System.EventArgs e)

{

Exception ex = Server.GetLastError();

Response.Write("Handled error from Page<br>");

//Server.ClearError();

}

protected void Application\_Error(Object sender, EventArgs e)

{

Exception ex = Server.GetLastError();

Response.Write("Handled error from Application <br>");

Server.ClearError();

}

protected void Global\_Error(Object sender, EventArgs e)

{

Exception ex = Server.GetLastError();

Response.Write("Handled error from Global <br>");

}

**Using Custom Error Pages**

Use custom error page to handle HTTP exceptions such as page not found, unauthorized access, and so forth. You can specify custom error pages in two places:

1. customErrors section of the web.config file. This setting specifies the application-wide error page to display for unhandled HTTP errors. HTTP errors are identified by the HTTP status code. Include the <error> tag in the customErrors to display a status code-specific error page. Does not work with .htm or .html files. Set the mode attribute to "On" to view the error page locally.
2. errorPage attribute of the @Page directive of the Web form to display the error page for the error generated on the particular Web form.

The customsError section in the web.config file specifies the application to redirect to Error404.aspx file if a non-existent file is requested.

<customErrors mode="On" defaultRedirect="Error.aspx">

<error statusCode="404" redirect="Error404.aspx" />

</customErrors>

The @Page directive specifies the error page to be redirected to if an error occurs in the Web page.

<%@ Page language="c#" Codebehind="PageErr.aspx.cs"

AutoEventWireup="false"

Inherits="ExceptionHandling.PageErr"

errorPage="Error.aspx" %>

**Difference Between overloading and overriding**

OverLoading :- Method Name remains the same with different signatures.   
For ex - Take a method name vehicle.  
if u are passing two parameters to it , it will generate informations for 2 wheelers .  
if u are passing four parameters to it, it will generate informations for 4 wheelers.  
  
shortly we can tell same with different characteristics.  
  
Overriding : - Method name and signatures must be same.  
In overriding user can change the behaviour of the method for the derived class.  
  
class A  
{  
int x;  
public virtual void PrintFields() {  
Console.WriteLine("x = {0}", x);  
}  
}  
class B: A  
{  
int y;  
public override void PrintFields() {  
base.PrintFields();  
Console.WriteLine("y = {0}", y);  
}  
}

Overloading - means "add" more behavior  
Over riding - means "change" existing behavior.

Overloading :  
is several methods with different signature but same name.  
  
Overriding:  
  
different methods with same name and signature but different functionality.

Overloading..  
1.Same name but there are different definitions and parameters..  
2.Here, the definitions are extented.  
3.Seperate methods share the same name.   
4.It is mainly for operators.  
5.It must have different method signatures.  
  
Overriding.  
1.Here replacement of methods.  
2.It is used in inheritance.  
3.subclass methods replaces the superclass.  
4.It is mainly for functions.  
5.It must have same signature

Difference between #Temptable and declare @temptable

**The first difference** is that transaction logs are not recorded for the table variables. Hence, they are out of scope of the transaction mechanism, as is clearly visible from this example:

create table #T (s varchar(128))   
declare @T table (s varchar(128))   
insert into #T select 'old value #'   
insert into @T select 'old value @'   
begin transaction   
     update #T set s='new value #'   
     update @T set s='new value @'   
rollback transaction   
select \* from #T   
select \* from @T   
  
s   
---------------   
old value #   
  
s   
---------------   
new value @

After declaring our temporary table #T and our table-variable @T, we assign each one with the same "old value" string. Then, we begin a transaction that updates their contents. At this point, both will now contain the same "new value" string. But when we rollback the transaction, as you can see, the table-variable @T retained its value instead of reverting back to the "old value" string. This happened because, even though the table-variable was updated within the transaction, it is not a part of the transaction itself.

**The second** major difference is that any procedure with a temporary table cannot be pre-compiled, while an execution plan of procedures with table variables can be statically compiled in advance. Pre-compiling a script gives a major advantage to its speed of execution. This advantage can be dramatic for long procedures, where recompilation can be too pricy.

**Common Language Runtime** (CLR)—A runtime environment that manages the execution of .NET program code, and provides services such as memory and exception management, debugging and profiling, and security. The CLR is a major component of the [.NET Framework](http://www.developer.com/net/net/article.php/#.NET Framework), and provides much of its functionality by following the rules defined in the [Common Type System](http://www.developer.com/net/net/article.php/#Common Type System). Also known as the *Virtual Execution System (VES)*.

1. **Common Language Specification** (CLS)—A set of common conventions used to promote interoperability between programming languages and the .NET Framework. The CLS specifies a subset of the [Common Type System](http://www.developer.com/net/net/article.php/#Common Type System) and set of conventions that are adhered to by both programming language designers and framework class library authors.
2. **Common Type System** (CTS)—The .NET Framework specification which defines the rules of how the [Common Language Runtime](http://www.developer.com/net/net/article.php/#Common Language Runtime) defines, declares, and manages [types](http://www.developer.com/net/net/article.php/#Types), regardless of the programming language. All .NET components must fulfill to the CTS specification.

**Difference between viewstate and session**

ViewState persist the values of controls of particular page in the client (browser) when post back operation done. When user requests another page previous page data no longer available.  
  
SessionState persist the data of particular user in the server. This data available till user close the browser or session time completes.

Strong name

A strong name consists of the assembly's identity — its simple text name, version number, and culture information (if provided) — plus a public key and a digital signature. It is generated from an assembly file (the file that contains the assembly manifest, which in turn contains the names and hashes of all the files that make

**Various modes of storing Asp.Net Session state**

1 In Proc : In this mode of session state is stored in the memory space of aspnet\_wp.exe.

Process: this is default setting if the IIS rboots or web application restarts then session state is lost.

2.Out Proc/State server:In this mode session state is serialized and stored in aspnet\_state.exe therefore state can be stored on a separate computer(a state server).

1. Sql Server: in this mode state is serialized and stored in the sql server database.

**View**

View is an alternating way of looking one or more table in a database. it is a virtual table which is subset of columns and rows from one or more tables. However view does not exist as a set of stored data values in a database. rows and columns come from tables according to query. Data is displayed directly from the table at the time of execution.

View is precompiled so if we are making any change in table structure which is used in view then we need to refresh or alter view to get correct data.

**Syntax**

CREATE VIEW view\_name  
AS select\_statement [WITH CHECK OPTION]

Where:

view\_name specifies the name of the view and must follow the rules for identifiers.

column\_name specifies the name of the column to be used in view. If the column\_name option is not specified, then the view is created with the same columns as specified in the select\_statement.

A web service is any service that is available over the Internet, uses a standardized XML

messaging system, and is not tied to any one operating system or programming language.

### What are the basic differences between user controls and custom controls?

Now that you have a basic idea of what user controls and custom controls are and how to create them, let's take a quick look at the differences between the two.

|  |  |  |
| --- | --- | --- |
| **Factors** | **User control** | **Custom control** |
| Deployment | Designed for single-application scenarios  Deployed in the source form (.ascx) along with the source code of the application  If the same control needs to be used in more than one application, it introduces redundancy and maintenance problems | Designed so that it can be used by more than one application  Deployed either in the application's Bin directory or in the global assembly cache  Distributed easily and without problems associated with redundancy and maintenance |
| Creation | Creation is similar to the way Web Forms pages are created; well-suited for rapid application development (RAD) | Writing involves lots of code because there is no designer support |
| Content | A much better choice when you need static content within a fixed layout, for example, when you make headers and footers | More suited for when an application requires dynamic content to be displayed; can be reused across an application, for example, for a data bound table control with dynamic rows |
| Design | Writing doesn't require much application designing because they are authored at design time and mostly contain static data | Writing from scratch requires a good understanding of the control's life cycle and the order in which events execute, which is normally taken care of in user controls |

User control  
  
1.Compiled at runtime  
2.HTML design (Visual design possible)  
3.ASP.Net page model with code behind  
4.Needs ASP.NET .aspx page to exist (can be used)  
5.No design time [interface](http://www.hotscripts.com/forums/asp-net/31174-difference-between-user-control-custom-server-controls.html)  
(Only a box representing the user control is available on an .aspx page)  
6.Cannot be added to the ToolBox  
  
  
Custom Server Control  
  
1.Precompiled  
2.No visual design. HTML needs to be declared programmatically  
3.Component model  
4.can be used in .aspx pages, user controls or other custom server controls.  
5.Has design-time and run-time interface  
6. Can be added to the ToolBox (using drag and drop)  
5.Since the dll assembly of a custom control is being used,a custom control developed in C# can be used in a project developed in VB.NET or any other managed code and vice versa.

This is not possible with user controls.They are language spefic

What is Web.Config File?

It is an optional XML File which stores configuration details for a specific asp.net web application.

Note: When you modify the settings in the Web.Config file, you do not need to restart the Web service for the modifications to take effect. By default, the Web.Config file applies to all the pages in the current directory and its subdirectories.

Extra: You can use the <location> tag to lock configuration settings in the Web.Config file so that they cannot be overridden by a Web.Config file located below it. You can use the allowOverride attribute to lock configuration settings. This attribute is especially valuable if you are hosting untrusted applications on your server.

Difference between Machine.Config and Web.Config

Machine.Config:

i) This is automatically installed when you install Visual Studio. Net.

ii) This is also called machine level configuration file.

iii)Only one machine.config file exists on a server.

iv) This file is at the highest level in the configuration hierarchy.

Web.Config:

i) This is automatically created when you create an ASP.Net web application project.

ii) This is also called application level configuration file.

iii)This file inherits setting from the machine.config

Custom controls are controls that are developed by the developer or a third party vendor. Custom controls are not provided along with .NET.

Difference between Custom Controls and User Controls.

1.User Control is a page file with extension .ascx which can only be used within

a single application. But custom controls are assemblies(dll files) that can be

used in multiple applications.

2.User Controls cannot be added to the ToolBox of VS.NET . To use a user Control with in an

aspx page u have to drag the user Control from the solution Explorer to designer page.

But Custom Controls can be added to ToolBox of VS.NET.

3.User Controls can be viewed as a sort of generic controls during the design time.

The proper GUI of user controls can be viewed only during the run time.

But Custom Controls can be viewed during the design time.

4. User controls are created from existing Webserver and html server controls .

But a developer who creates custom controls have to render every thing from the scratch.

5.Since the dll assembly of a custom control is being used,a custom control developed in C# can be used in a project developed in VB.NET or any other managed code and vice versa.

This is not possible with user controls.They are language spefic

**Partial Class**

Partial class is a class which can be define over two or more source file by using keyword partial . Functionally, partial classes are not different from normal classes. All part of the class are combined at compile time. All part of the class must be defined within same assembly.

Difference b/w Primary key and Unique Key.

Primary key can not have null value but unique key can have one null value.

Primary Key: Primary key is the key by which we can identify each and every row of a table.

**Difference between Primary Key & Unique Key**

Primary Key can't accept null values.

Unique key can accept only one null value.

By default, Primary key is clustered index and data in the database table is physically organized in the sequence of clustered index.

By default, Unique key is a unique non-clustered index.

We can have only one Primary key in a table.

We can have more than one unique key in a table.

Primary key can be made foreign key into another table.

In SQL Server, Unique key can be made foreign key into another table.

## Define Primary key and Unique key

1. **CREATE TABLE Employee**
2. **(**
3. **EmpID int PRIMARY KEY, *--define primary key***
4. **Name varchar (50) NOT NULL,**
5. **MobileNo int UNIQUE, *--define unique key***
6. **Salary int NULL**
7. **)**

Different between Delete Table and Truncate Table

Truncate and Delete both are used to delete data from the table. These both command will only delete data of the specified table, they cannot remove the whole table data structure.Both statements delete the data from the table not the structure of the table.

* TRUNCATE is a DDL (data definition language) command whereas DELETE is a DML (data manipulation language) command.
* You can use WHERE clause(conditions) with DELETE but you can't use WHERE clause with TRUNCATE .
* You cann't rollback data in TRUNCATE but in DELETE you can rollback data.TRUNCATE removes(delete) the record permanently.
* A trigger doesn’t get fired in case of TRUNCATE whereas Triggers get fired in DELETE command.
* If tables which are referenced by one or more FOREIGN KEY constraints then TRUNCATE will not work.
* TRUNCATE resets the Identity counter if there is any identity column present in the table where delete not resets the identity counter.
* Delete and Truncate both are logged operation.But DELETE is a logged operation on a per row basis and TRUNCATE logs the deallocation of the data pages in which the data exists.
* TRUNCATE is faster than DELETE.

INDEX: Indexes are created on columns in tables or views. The index provides a fast way to look up data based on the values within those columns. For example, if you create an index on the primary key and then search for a row of data based on one of the primary key values, SQL Server first finds that value in the index, and then uses the index to quickly locate the entire row of data. Without the index, a table scan would have to be performed in order to locate the row, which can have a significant effect on performance.

You can create indexes on most columns in a table or a view. The exceptions are primarily those columns configured with large object (LOB) data types, such as **image**, **text,** and **varchar(max)**. You can also create indexes on XML columns, but those indexes are slightly different from the basic index and are beyond the scope of this article. Instead, I'll focus on those indexes that are implemented most commonly in a SQL Server database.

**''Deleting duplicate rows from table.**

SET ROWCOUNT 1

DELETE j FROM j a

WHERE

(

SELECT COUNT(\*) FROM j b WHERE b.k = a.k

) > 1

WHILE @@rowcount > 0

DELETE j FROM j a WHERE

(

SELECT COUNT(\*) FROM j b WHERE b.k = a.k

) > 1

SET ROWCOUNT 0

**INTERFACE VS ABSTRACT CLASS**

* interfaces can have no state or implementation
* a class that implements an interface must provide an implementation of all the methods of that interface
* abstract classes may contain state (data members) and/or implementation (methods)
* abstract classes can be inherited without implementing the abstract methods (though such a derived class is abstract itself)
* interfaces may be multiple-inherited, abstract classes may not (this is probably the key concrete reason for interfaces to exist separately from abtract classes - they permit an implementation of multiple inheritance that removes many of the problems of general MI).

**Generics**

Generics introduced in C# 2.0. Generics allow you to define a class with placeholders or Anonymous type of its fields, methods, parameters, etc. Generics replace these placeholders with some specific type at compile time.

class MyGenericClass<T>

{

private T genericMemberVariable;

public MyGenericClass(T value)

{

genericMemberVariable = value;

}

public T genericMethod(T genericParameter)

{

Console.WriteLine("Parameter type: {0}, value: {1}", typeof(T).ToString(),genericParameter);

Console.WriteLine("Return type: {0}, value: {1}", typeof(T).ToString(), genericMemberVariable);

return genericMemberVariable;

}

public T genericProperty { get; set; }

}

**ARRAY VS ARRAYLIST**

**Arrays** belong to System.**Array** namespace whereas **Arraylist** belongs to System.Collection namespaces . **Array** is strongly typed . This means that an **array** can store only specific type of items\elements. As a result, it is type safe, **and** is also the most efficient, both in terms of memory **and** performance. ArrayList grows automatically and you don't need to specify size.

**Array can’t accept null value but ArraryList can accept null values.**

**ArrayList**

ArrayList is a non-generic type of collection in C#. It can contain elements of any data types. It is similar to an [array](http://www.tutorialsteacher.com/csharp/array-csharp), except that it grows automatically as you add items in it. Unlike an array, you don't need to specify the size of ArrayList.

Example: Initialize ArrayList

ArrayList myArryList = new ArrayList();

ArrayList arryList1 = new ArrayList();

arryList1.Add(1);

arryList1.Add("Two");

arryList1.Add(3);

arryList1.Add(4.5);

ArrayList arryList2 = new ArrayList();

arryList2.Add(100);

arryList2.Add(200);

//adding entire arryList2 into arryList1

arryList1.AddRange(arryList2);

**ARRAY**

An array can be declared and initialized at the same time using the new keyword. The following example shows the way of initializing an array.

// defining array with size 5 and adding values at the same time

int[] intArray2 = new int[5]{1, 2, 3, 4, 5};

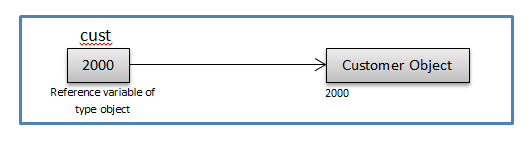
**ARRAYLIST VS LIST**

First, one should know what is upcasting? **Upcasting is converting derived type into base type.** In .NET, all data-types are derived from Object. So we can typecast any type to Object type. For example, if Customer is class, then we can create object of Customer like this:

Hide   Copy Code

Object cust = new Customer()

Here new Customer() will create object on heap and its address we are putting in reference variable of type Object.



## ArrayList

Hide   Copy Code

ArrayList marks = new ArrayList();

marks.Add(50);

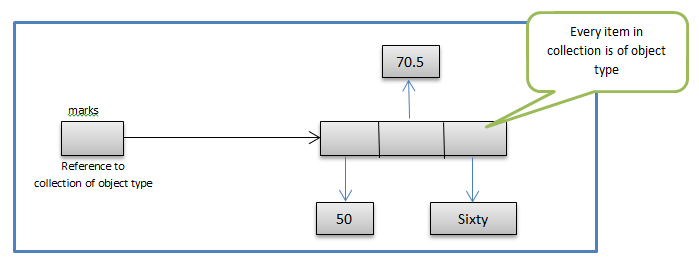
marks.Add(70.5);

marks.Add("Sixty");

In the above code snippet, we are creating object of ArrayList and adding different type of data in it. But actually ArrayList is a collection of **Object**type, and when we add any **item**to ArrayList, it first **converts**it to object type (upcasting) and then adds it to collection object.

**Interesting Fact**: As ArrayList can only create collection for Object type, it is said to be non-generic class. It might be confusing as it seems that we can add any datatype value like int, float, string to ArrayListcollection so in that sense it should be called as generic class. But in fact, it internally converts all these datatypes in object type and then adds to collection.

We can visualise it as:



## List

Hide   Copy Code

List<int> marks = new List<int>();

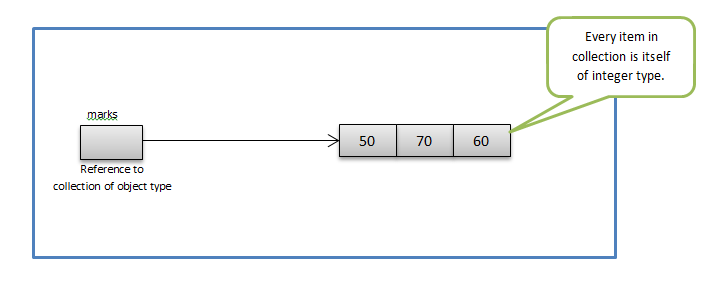
marks.Add(50);

marks.Add(70);

marks.Add(60);

In the above snippet, we can observe that while creating object of list class, we have mentioned datatype of collection we want to create. We need to pass datatype while creating object as List class doesn’t hard code it internally. So the above declaration will create marks as collection of **integers**; and **not** collection of **objects** as in case of ArrayList.

We can visualize it as:



**Interesting fact**: In the above collection “marks” you can only add integers and no other type. In that sense, it should be referred to as non-generic! But wait, using the same List class, you can also create collection of string type:

Hide   Copy Code

List<string> names = new List<string>();

Or even you can create collection of **custom type**s. For example, collection of Student type can be created as:

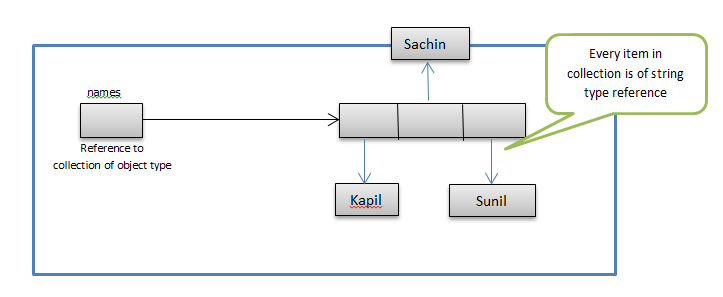
Hide   Copy Code

List<Student> students = new List<Student>();

And as using same **List**class, now you are able to create **collection** **of any data-type** as integers, strings or students; this class is known as **Generic**class.

One of the benefits of using generic collection is no need of boxing and unboxing while tackling with collections of **value types.**

We can visualise List of string type (or any ref type) as:

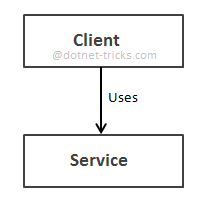


**DEPENDENCY INJECTION**

Dependency Injection (DI) is a software design pattern that allows us to develop loosely coupled code. DI is a great way to reduce tight coupling between software components. DI also enables us to better manage future changes and other complexity in our software. The purpose of DI is to make code maintainable.

The Dependency Injection pattern uses a builder object to initialize objects and provide the required dependencies to the object means it allows you to "inject" a dependency from outside the class.

**For example**, Suppose your Client class needs to use a Service class component, then the best you can do is to make your Client class aware of an IService interface rather than a Service class. In this way, you can change the implementation of the Service class at any time (and for how many times you want) without breaking the host code.



We can modify this code by the DI different ways. We have following different ways to implement DI :

## Constructor Injection

1. This is the most common DI.
2. Dependency Injection is done by supplying the DEPENDENCY through the class’s constructor when instantiating that class.
3. Injected component can be used anywhere within the class.
4. Should be used when the injected dependency is required for the class to function.
5. It addresses the most common scenario where a class requires one or more dependencies.
6. public interface IService
7. {
8. void Serve();
9. }
11. public class Service : IService
12. {
13. public void Serve()
14. {
15. Console.WriteLine("Service Called");
16. *//To Do: Some Stuff*
17. }
18. }
20. public class Client
21. {
22. private IService \_service;
24. public Client(IService service)
25. {
26. this.\_service = service;
27. }
29. public void Start()
30. {
31. Console.WriteLine("Service Started");
32. this.\_service.Serve();
33. *//To Do: Some Stuff*
34. }
35. }
36. class Program
37. {
38. static void Main(string[] args)
39. {
40. Client client = new Client(new Service());
41. client.Start();
43. Console.ReadKey();
44. }
45. }

The Injection happens in the constructor, by passing the Service that implements the IService-Interface. The dependencies are assembled by a "Builder" and Builder responsibilities are as follows:

1. knowing the types of each IService
2. according to the request, feed the abstract IService to the Client

## Property injection

1. Also called Setter injection.
2. Used when a class has optional dependencies, or where the implementations may need to be swapped. Different logger implementations could be used this way.
3. May require checking for a provided implementation throughout the class(need to check for null before using it).
4. Does not require adding or modifying constructors.
5. public interface IService
6. {
7. void Serve();
8. }
10. public class Service : IService
11. {
12. public void Serve()
13. {
14. Console.WriteLine("Service Called");
15. *//To Do: Some Stuff*
16. }
17. }
19. public class Client
20. {
21. private IService \_service;
23. public IService Service
24. {
25. set
26. {
27. this.\_service = value;
28. }
29. }
31. public void Start()
32. {
33. Console.WriteLine("Service Started");
34. this.\_service.Serve();
35. *//To Do: Some Stuff*
36. }
37. }
38. class Program
39. {
40. static void Main(string[] args)
41. {
42. Client client = new Client();
43. client.Service = new Service();
44. client.Start();
46. Console.ReadKey();
47. }
48. }

## Method injection

1. Inject the dependency into a single method, for use by that method.
2. Could be useful where the whole class does not need the dependency, just the one method.
3. Generally uncommon, usually used for edge cases.
4. public interface IService
5. {
6. void Serve();
7. }
9. public class Service : IService
10. {
11. public void Serve()
12. {
13. Console.WriteLine("Service Called");
14. *//To Do: Some Stuff*
15. }
16. }
18. public class Client
19. {
20. private IService \_service;
22. public void Start(IService service)
23. {
24. this.\_service = service;
25. Console.WriteLine("Service Started");
26. this.\_service.Serve();
27. *//To Do: Some Stuff*
28. }
29. }
30. class Program
31. {
32. static void Main(string[] args)
33. {
34. Client client = new Client();
35. client.Start(new Service());
37. Console.ReadKey();
38. }
39. }

## Key points about DI

1. Reduces class coupling
2. Increases code reusing
3. Improves code maintainability
4. Improves application testing

## What is Factory Method Pattern?

In Factory pattern, we create object without exposing the creation logic. In this pattern, an interface is used for creating an object, but let subclass decide which class to instantiate. The creation of object is done when it is required. The Factory method allows a class later instantiation to subclasses

**OVERRIDING**

Method overriding in C# is a feature like the virtual function in C++. Method overriding is a feature that allows you to invoke functions (that have the same signatures) that belong to different classes in the same hierarchy of inheritance using the base class reference. C# makes use of two keywords: virtual and overrides to accomplish Method overriding. Let's understand this through small examples.

### P1.cs

Hide   Copy Code

class BC

{

public void Display()

{

System.Console.WriteLine("BC::Display");

}

}

class DC : BC

{

new public void Display()

{

System.Console.WriteLine("DC::Display");

}

}

class Demo

{

public static void Main()

{

BC b;

b = new BC();

b.Display();

}

}

#### Output

Hide   Copy Code

BC::Display

The above program compiles and runs successfully to give the desired output. It consists of a base class BC and a derived class DC. Class BC consists of function Display(). Class DC hides the function Display() it inherited from the base class BC by providing its on implementatin of Display(). Class Demo consists of entrypoint function Main(). Inside Main() we first create a reference b of type BC. Then we create an object of type BC and assign its reference to reference variable b. Using the reference variable b we invoke the function Display(). As expected, Display() of class BC is executed because the reference variable b refers to the object of class BC.

Now we add a twist of lime to the above program.

### P2.cs

Hide   Copy Code

class BC

{

public void Display()

{

System.Console.WriteLine("BC::Display");

}

}

class DC : BC

{

new public void Display()

{

System.Console.WriteLine("DC::Display");

}

}

class Demo

{

public static void Main()

{

BC b;

b = new BC();

b.Display();

b = new DC();

b.Display();

}

}

#### Output

Hide   Copy Code

BC::Display

BC::Display

Here we are creating an object of Derived class DC and storing its reference in the reference variable b of type BC. This is valid in C#. Next, using the reference variable b we invoke the function Display(). Since b contains a reference to object of type DC one would expect the function Display() of class DC to get executed. But that does not happen. Instead what is executed is the Display() of BC class. That's because the function is invoked based on type of the reference and not to what the reference variable b refers to. Since b is a reference of type BC, the function Display() of class BC will be invoked, no matter whom b refers to. Take one more example.

### P3.cs

Hide   Shrink https://www.codeproject.com/images/arrow-up-16.png   Copy Code

class BC

{

public void Display()

{

System.Console.WriteLine("BC::Display");

}

}

class DC : BC

{

new public void Display()

{

System.Console.WriteLine("DC::Display");

}

}

class TC : BC

{

new public void Display()

{

System.Console.WriteLine("DC::Display");

}

}

class Demo

{

public static void Main()

{

BC b;

b = new BC();

b.Display();

b = new DC();

b.Display();

b = new TC();

b.Display();

}

}

#### Output

Hide   Copy Code

BC::Display

BC::Display

BC::Display

The output of the above program is a receipt of the fact that no matter to whom base class reference b refers, it invokes the functions of the class that matches its type. But doesn't this sound absurd? If b contains the reference to a perticular derived class object, then its supposed to invoke the function of that class. Well, C# helps us do this by the usage of keywords virtual and override as shown in the following program.

### P4.cs

Hide   Copy Code

class BC

{

public virtual void Display()

{

System.Console.WriteLine("BC::Display");

}

}

class DC : BC

{

public override void Display()

{

System.Console.WriteLine("DC::Display");

}

}

class Demo

{

public static void Main()

{

BC b;

b = new BC();

b.Display();

b = new DC();

b.Display();

}

}

#### Output

Hide   Copy Code

BC::Display

DC::Display

The above program compiles and runs successfully to give the expected desired output. The function Display()of Base class BC is declared as virtual, while the Derived class's implementation of Display() is decorated with the modifier override. Doing so enables C# to invoke functions like Display() based on objects the reference variable refers to and not the type of reference that is invoking the function. Hence in the above program when brefers to the object of class BC it invokes Display() of BC and then when b refers to the object of class DC it invokes Display() of class DC. Let's see if this holds true for the third generation of derived classes. Take the following program.

### P4.cs

Hide   Shrink https://www.codeproject.com/images/arrow-up-16.png   Copy Code

class BC

{

public virtual void Display()

{

System.Console.WriteLine("BC::Display");

}

}

class DC : BC

{

public override void Display()

{

System.Console.WriteLine("DC::Display");

}

}

class TC : DC

{

public override void Display()

{

System.Console.WriteLine("TC::Display");

}

}

class Demo

{

public static void Main()

{

BC b;

b = new BC();

b.Display();

b = new DC();

b.Display();

b = new TC();

b.Display();

}

}

#### Output

Hide   Copy Code

BC::Display

DC::Display

TC::Display

The above program compiles and runs successfully to give the expected desired output. The function Display()of Base class BC is declared as virtual, while the implementation of Display() in successive Derived classes is decorated with the modifier override. Next, we succesively create objects of each class and store their reference in base class reference variable b and invoke Display(). The rite versions of Display get invoked based on the object the reference variable refers to. Time for a tiny teaser! Guess what the output would be in the following program?

### P5.cs

Hide   Shrink https://www.codeproject.com/images/arrow-up-16.png   Copy Code

class BC

{

public virtual void Display()

{

System.Console.WriteLine("BC::Display");

}

}

class DC : BC

{

public override void Display()

{

System.Console.WriteLine("DC::Display");

}

}

class TC : DC

{

}

class Demo

{

public static void Main()

{

BC b;

b = new TC();

b.Display();

}

}

#### Output

Hide   Copy Code

DC::Display

Since TC has no implementation of Display(), it inherits Display() from DC as TC is derived from DC. Hence Display() from Derived class DC gets executed. It's as if the derived class TC looked like this:

Hide   Copy Code

class TC

{

public override void Display()

{

System.Console.WriteLine("DC::Display");

}

}

to the compiler. Take one more example. Guess what its output will be.

### P6.cs

Hide   Shrink https://www.codeproject.com/images/arrow-up-16.png   Copy Code

class BC

{

public virtual void Display()

{

System.Console.WriteLine("BC::Display");

}

}

class DC : BC

{

public override void Display()

{

System.Console.WriteLine("DC::Display");

}

}

class TC : DC

{

public new void Display()

{

System.Console.WriteLine("TC::Display");

}

}

class Demo

{

public static void Main()

{

BC b;

b = new TC();

b.Display();

}

}

#### Output

Hide   Copy Code

DC::Display

Agreed that TC defines its own new version of Display(). But its version of display is not invoked as Display() of TC does not override the Display() of the base class. With this understood we are done with Method overriding in C#.

# Knowing When to Use Override and New Keywords (C# Programming Guide)

* 07/20/2015
* 10 minutes to read
* Contributors
  + [Bill Wagner](https://github.com/BillWagner)

* + [Maira Wenzel](https://github.com/mairaw)

* + [Matt Hoffman](https://github.com/mjhoffman65)

* + [](https://github.com/mikeblome)

* + [Luke Latham](https://github.com/guardrex)

* + [all](https://github.com/dotnet/docs/blob/master/docs/csharp/programming-guide/classes-and-structs/knowing-when-to-use-override-and-new-keywords.md)

In C#, a method in a derived class can have the same name as a method in the base class. You can specify how the methods interact by using the [new](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/new) and [override](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/override) keywords. The override modifier extends the base class method, and the new modifier hides it. The difference is illustrated in the examples in this topic.

In a console application, declare the following two classes, BaseClass and DerivedClass. DerivedClass inherits from BaseClass.

C#Copy

class BaseClass

{

public void Method1()

{

Console.WriteLine("Base - Method1");

}

}

class DerivedClass : BaseClass

{

public void Method2()

{

Console.WriteLine("Derived - Method2");

}

}

In the Main method, declare variables bc, dc, and bcdc.

* bc is of type BaseClass, and its value is of type BaseClass.
* dc is of type DerivedClass, and its value is of type DerivedClass.
* bcdc is of type BaseClass, and its value is of type DerivedClass. This is the variable to pay attention to.

Because bc and bcdc have type BaseClass, they can only directly access Method1, unless you use casting. Variable dc can access both Method1 and Method2. These relationships are shown in the following code.

C#Copy

class Program

{

static void Main(string[] args)

{

BaseClass bc = new BaseClass();

DerivedClass dc = new DerivedClass();

BaseClass bcdc = new DerivedClass();

bc.Method1();

dc.Method1();

dc.Method2();

bcdc.Method1();

}

// Output:

// Base - Method1

// Base - Method1

// Derived - Method2

// Base - Method1

}

Next, add the following Method2 method to BaseClass. The signature of this method matches the signature of the Method2 method in DerivedClass.

C#Copy

public void Method2()

{

Console.WriteLine("Base - Method2");

}

Because BaseClass now has a Method2 method, a second calling statement can be added for BaseClass variables bc and bcdc, as shown in the following code.

C#Copy

bc.Method1();

bc.Method2();

dc.Method1();

dc.Method2();

bcdc.Method1();

bcdc.Method2();

When you build the project, you see that the addition of the Method2 method in BaseClass causes a warning. The warning says that the Method2 method in DerivedClass hides the Method2 method in BaseClass. You are advised to use the new keyword in the Method2 definition if you intend to cause that result. Alternatively, you could rename one of the Method2 methods to resolve the warning, but that is not always practical.

Before adding new, run the program to see the output produced by the additional calling statements. The following results are displayed.

C#Copy

// Output:

// Base - Method1

// Base - Method2

// Base - Method1

// Derived - Method2

// Base - Method1

// Base - Method2

The new keyword preserves the relationships that produce that output, but it suppresses the warning. The variables that have type BaseClass continue to access the members of BaseClass, and the variable that has type DerivedClass continues to access members in DerivedClass first, and then to consider members inherited from BaseClass.

To suppress the warning, add the new modifier to the definition of Method2 in DerivedClass, as shown in the following code. The modifier can be added before or after public.

C#Copy

public new void Method2()

{

Console.WriteLine("Derived - Method2");

}

Run the program again to verify that the output has not changed. Also verify that the warning no longer appears. By using new, you are asserting that you are aware that the member that it modifies hides a member that is inherited from the base class. For more information about name hiding through inheritance, see [new Modifier](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/new-modifier).

To contrast this behavior to the effects of using override, add the following method to DerivedClass. The override modifier can be added before or after public.

C#Copy

public override void Method1()

{

Console.WriteLine("Derived - Method1");

}

Add the virtual modifier to the definition of Method1 in BaseClass. The virtual modifier can be added before or after public.

C#Copy

public virtual void Method1()

{

Console.WriteLine("Base - Method1");

}

Run the project again. Notice especially the last two lines of the following output.

C#Copy

// Output:

// Base - Method1

// Base - Method2

// Derived - Method1

// Derived - Method2

// Derived - Method1

// Base - Method2

The use of the override modifier enables bcdc to access the Method1 method that is defined in DerivedClass. Typically, that is the desired behavior in inheritance hierarchies. You want objects that have values that are created from the derived class to use the methods that are defined in the derived class. You achieve that behavior by using override to extend the base class method.

The following code contains the full example.

C#Copy

using System;

using System.Text;

namespace OverrideAndNew

{

class Program

{

static void Main(string[] args)

{

BaseClass bc = new BaseClass();

DerivedClass dc = new DerivedClass();

BaseClass bcdc = new DerivedClass();

// The following two calls do what you would expect. They call

// the methods that are defined in BaseClass.

bc.Method1();

bc.Method2();

// Output:

// Base - Method1

// Base - Method2

// The following two calls do what you would expect. They call

// the methods that are defined in DerivedClass.

dc.Method1();

dc.Method2();

// Output:

// Derived - Method1

// Derived - Method2

// The following two calls produce different results, depending

// on whether override (Method1) or new (Method2) is used.

bcdc.Method1();

bcdc.Method2();

// Output:

// Derived - Method1

// Base - Method2

}

}

class BaseClass

{

public virtual void Method1()

{

Console.WriteLine("Base - Method1");

}

public virtual void Method2()

{

Console.WriteLine("Base - Method2");

}

}

class DerivedClass : BaseClass

{

public override void Method1()

{

Console.WriteLine("Derived - Method1");

}

public new void Method2()

{

Console.WriteLine("Derived - Method2");

}

}

}

The following example illustrates similar behavior in a different context. The example defines three classes: a base class named Car and two classes that are derived from it, ConvertibleCar and Minivan. The base class contains a DescribeCar method. The method displays a basic description of a car, and then calls ShowDetails to provide additional information. Each of the three classes defines a ShowDetails method. The new modifier is used to define ShowDetails in the ConvertibleCar class. The override modifier is used to define ShowDetails in the Minivan class.

C#Copy

// Define the base class, Car. The class defines two methods,

// DescribeCar and ShowDetails. DescribeCar calls ShowDetails, and each derived

// class also defines a ShowDetails method. The example tests which version of

// ShowDetails is selected, the base class method or the derived class method.

class Car

{

public void DescribeCar()

{

System.Console.WriteLine("Four wheels and an engine.");

ShowDetails();

}

public virtual void ShowDetails()

{

System.Console.WriteLine("Standard transportation.");

}

}

// Define the derived classes.

// Class ConvertibleCar uses the new modifier to acknowledge that ShowDetails

// hides the base class method.

class ConvertibleCar : Car

{

public new void ShowDetails()

{

System.Console.WriteLine("A roof that opens up.");

}

}

// Class Minivan uses the override modifier to specify that ShowDetails

// extends the base class method.

class Minivan : Car

{

public override void ShowDetails()

{

System.Console.WriteLine("Carries seven people.");

}

}

The example tests which version of ShowDetails is called. The following method, TestCars1, declares an instance of each class, and then calls DescribeCar on each instance.

C#Copy

public static void TestCars1()

{

System.Console.WriteLine("\nTestCars1");

System.Console.WriteLine("----------");

Car car1 = new Car();

car1.DescribeCar();

System.Console.WriteLine("----------");

// Notice the output from this test case. The new modifier is

// used in the definition of ShowDetails in the ConvertibleCar

// class.

ConvertibleCar car2 = new ConvertibleCar();

car2.DescribeCar();

System.Console.WriteLine("----------");

Minivan car3 = new Minivan();

car3.DescribeCar();

System.Console.WriteLine("----------");

}

TestCars1 produces the following output. Notice especially the results for car2, which probably are not what you expected. The type of the object is ConvertibleCar, but DescribeCar does not access the version of ShowDetails that is defined in the ConvertibleCar class because that method is declared with the new modifier, not the override modifier. As a result, a ConvertibleCar object displays the same description as a Car object. Contrast the results for car3, which is a Minivanobject. In this case, the ShowDetails method that is declared in the Minivan class overrides the ShowDetails method that is declared in the Car class, and the description that is displayed describes a minivan.

C#Copy

// TestCars1

// ----------

// Four wheels and an engine.

// Standard transportation.

// ----------

// Four wheels and an engine.

// Standard transportation.

// ----------

// Four wheels and an engine.

// Carries seven people.

// ----------

TestCars2 creates a list of objects that have type Car. The values of the objects are instantiated from the Car, ConvertibleCar, and Minivan classes. DescribeCar is called on each element of the list. The following code shows the definition of TestCars2.

C#Copy

public static void TestCars2()

{

System.Console.WriteLine("\nTestCars2");

System.Console.WriteLine("----------");

var cars = new List<Car> { new Car(), new ConvertibleCar(),

new Minivan() };

foreach (var car in cars)

{

car.DescribeCar();

System.Console.WriteLine("----------");

}

}

The following output is displayed. Notice that it is the same as the output that is displayed by TestCars1. The ShowDetails method of the ConvertibleCar class is not called, regardless of whether the type of the object is ConvertibleCar, as in TestCars1, or Car, as in TestCars2. Conversely, car3 calls the ShowDetails method from the Minivan class in both cases, whether it has type Minivan or type Car.

C#Copy

// TestCars2

// ----------

// Four wheels and an engine.

// Standard transportation.

// ----------

// Four wheels and an engine.

// Standard transportation.

// ----------

// Four wheels and an engine.

// Carries seven people.

// ----------

Methods TestCars3 and TestCars4 complete the example. These methods call ShowDetails directly, first from objects declared to have type ConvertibleCar and Minivan (TestCars3), then from objects declared to have type Car (TestCars4). The following code defines these two methods.

C#Copy

public static void TestCars3()

{

System.Console.WriteLine("\nTestCars3");

System.Console.WriteLine("----------");

ConvertibleCar car2 = new ConvertibleCar();

Minivan car3 = new Minivan();

car2.ShowDetails();

car3.ShowDetails();

}

public static void TestCars4()

{

System.Console.WriteLine("\nTestCars4");

System.Console.WriteLine("----------");

Car car2 = new ConvertibleCar();

Car car3 = new Minivan();

car2.ShowDetails();

car3.ShowDetails();

}

The methods produce the following output, which corresponds to the results from the first example in this topic.

C#Copy

// TestCars3

// ----------

// A roof that opens up.

// Carries seven people.

// TestCars4

// ----------

// Standard transportation.

// Carries seven people.

The following code shows the complete project and its output.

C#Copy

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace OverrideAndNew2

{

class Program

{

static void Main(string[] args)

{

// Declare objects of the derived classes and test which version

// of ShowDetails is run, base or derived.

TestCars1();

// Declare objects of the base class, instantiated with the

// derived classes, and repeat the tests.

TestCars2();

// Declare objects of the derived classes and call ShowDetails

// directly.

TestCars3();

// Declare objects of the base class, instantiated with the

// derived classes, and repeat the tests.

TestCars4();

}

public static void TestCars1()

{

System.Console.WriteLine("\nTestCars1");

System.Console.WriteLine("----------");

Car car1 = new Car();

car1.DescribeCar();

System.Console.WriteLine("----------");

// Notice the output from this test case. The new modifier is

// used in the definition of ShowDetails in the ConvertibleCar

// class.

ConvertibleCar car2 = new ConvertibleCar();

car2.DescribeCar();

System.Console.WriteLine("----------");

Minivan car3 = new Minivan();

car3.DescribeCar();

System.Console.WriteLine("----------");

}

// Output:

// TestCars1

// ----------

// Four wheels and an engine.

// Standard transportation.

// ----------

// Four wheels and an engine.

// Standard transportation.

// ----------

// Four wheels and an engine.

// Carries seven people.

// ----------

public static void TestCars2()

{

System.Console.WriteLine("\nTestCars2");

System.Console.WriteLine("----------");

var cars = new List<Car> { new Car(), new ConvertibleCar(),

new Minivan() };

foreach (var car in cars)

{

car.DescribeCar();

System.Console.WriteLine("----------");

}

}

// Output:

// TestCars2

// ----------

// Four wheels and an engine.

// Standard transportation.

// ----------

// Four wheels and an engine.

// Standard transportation.

// ----------

// Four wheels and an engine.

// Carries seven people.

// ----------

public static void TestCars3()

{

System.Console.WriteLine("\nTestCars3");

System.Console.WriteLine("----------");

ConvertibleCar car2 = new ConvertibleCar();

Minivan car3 = new Minivan();

car2.ShowDetails();

car3.ShowDetails();

}

// Output:

// TestCars3

// ----------

// A roof that opens up.

// Carries seven people.

public static void TestCars4()

{

System.Console.WriteLine("\nTestCars4");

System.Console.WriteLine("----------");

Car car2 = new ConvertibleCar();

Car car3 = new Minivan();

car2.ShowDetails();

car3.ShowDetails();

}

// Output:

// TestCars4

// ----------

// Standard transportation.

// Carries seven people.

}

// Define the base class, Car. The class defines two virtual methods,

// DescribeCar and ShowDetails. DescribeCar calls ShowDetails, and each derived

// class also defines a ShowDetails method. The example tests which version of

// ShowDetails is used, the base class method or the derived class method.

class Car

{

public virtual void DescribeCar()

{

System.Console.WriteLine("Four wheels and an engine.");

ShowDetails();

}

public virtual void ShowDetails()

{

System.Console.WriteLine("Standard transportation.");

}

}

// Define the derived classes.

// Class ConvertibleCar uses the new modifier to acknowledge that ShowDetails

// hides the base class method.

class ConvertibleCar : Car

{

public new void ShowDetails()

{

System.Console.WriteLine("A roof that opens up.");

}

}

// Class Minivan uses the override modifier to specify that ShowDetails

// extends the base class method.

class Minivan : Car

{

public override void ShowDetails()

{

System.Console.WriteLine("Carries seven people.");

}

}

}

**Abstraction**  
  
Abstraction allows us to represent complex real world in simplest manner. It is process of identifying the relevant qualities and behaviors an object should possess, in other word represent the necessary feature without representing the back ground details. Abstraction is a process of hiding work style of an object and showing only those information which are required to understand the object. Abstraction means putting all the variables and methods in a class which are necessary.  
  
**Encapsulation**  
  
It is a process of hiding all the internal details of an object from the outside real world. The word Encapsulation, like Enclosing into the capsule. It restrict client from seeing its internal view where behavior of the abstraction is implemented. In Encapsulation, generally to hide data making it private and expose public property to access those data from outer world. Encapsulation is a method for protecting data from unwanted access or alteration. Encapsulation is the mechanism by which Abstraction is implemented.  
  
**Difference between Abstraction and Encapsulation**  
  
Abstraction is a process. It is the act of identifying the relevant qualities and behaviors an object should possess. Encapsulation is the mechanism by which the abstraction is implemented.

|  |  |
| --- | --- |
| Abstraction | Encapsulation |
| Abstraction solves the problem in the design level. | Encapsulation solves the problem in the implementation level. |
| Abstraction is used for hiding the unwanted data and giving only relevant data. | Encapsulation is hiding the code and data into a single unit to protect the data from outer world. |
| Abstraction is set focus on the object instead of how it does it. | Encapsulation means hiding the internal details or mechanics of how an object does something. |
| Abstraction is outer layout in terms of design.  For Example: - Outer Look of a iPhone, like it has a display screen. | Encapsulation is inner layout in terms of implementation. For Example: - Inner Implementation detail of a iPhone, how Display Screen are connect with each other using circuits |

#### What is a View?

View can be described as virtual table which derived its data from one or more than one table columns. It is stored in the database. View can be created using tables of same database or different database. It is used to implement the security mechanism in the SQL Server.

For example:

Hide   Copy Code

Create table Emp\_Details(EmpId int, EmpName nvarchar(200),

EmpLogin nvarchar(20), EmpPassword nvarchar(20) , EmploymentDate datetime )

And for example, table has the following data of employees:

Hide   Copy Code

EmpId EmpName EmpLogin Emppassword EmploymentDate

1 EmployeeA EmpA EmpAPwd 29/01/2006

2 EmployeeB EmpB EmpBPwd 06/02/2007

3 EmployeeC EmpC EmpCPwd 14/05/2007

4 EmployeeD EmpD Empd 30/03/2008

5 EmployeeE EmpE EmpEPwd 30/06/2007

6 EmployeeF EmpF EmpFPwd 12/09/2012

Now suppose that the Administrator do not want that the users to access the whole data of Emp\_Details table which contains some critical information (Emplogin, EmpPassword, etc.) of the Employees. So he can create a view which gives the empid, empname, employmentdate as the output and gives permission for the view to the user. In this way, the administrator does not need to give access permission for the table to the user.

#### Use of a View

Views are used for security purposes because they provide encapsulation of the name of the table. Data is in the virtual table, not stored permanently. Views display only selected data.

**Ref VS Out**

**Ref** keywords are used to pass an argument as a reference, meaning that when the value of that parameter changes after being passed through the method, the new value is reflected in the calling method. An argument passed using the ref keyword must be defined in the calling method before getting passed to the called method.

**Out** keywords similar to ref keywords in that, they are used to pass an argument, but they differ in that arguments passed using out keywords can be passed without any value to be assigned to it. An argument passed using the out keyword must be defined in the called method before being returning to the calling method.

public class Keywords

{

public static void Main() //calling method

{

int val1 = 0; //must be defined

int val2; //optional

Keywords1(ref val1);

Console.WriteLine(val1); // val1=7

Keywords2(out val2);

Console.WriteLine(val2); // val2=9

}

static void Keywords1(ref int value) //called method

{

value = 7;

}

static void Keywords2(out int value) //called method

{

value = 9; //must be defined

}

}

/\* Output

7

9

\*/

**Difference between clustered and non-clustered index**

|  |
| --- |
|  |
| There can be only 1 Clustered index in a table  where as nonclustered index can be upto 249  Clustered is physical sorted index..  non clustered index is faster than clustered index  The difference is that, Clustered index is unique for any  given table and we can have only one clustered index on a  table. The leaf level of a clustered index is the actual  data and the data is resorted in case of clustered index.  Whereas in case of non-clustered index the leaf level is  actually a pointer to the data in rows so we can have as  many non-clustered indexes as we can on the db.  **What are differences between Array and Hash table?**  **Ans:** 1) Hash table store data as name, value pair. While in array only value is store.  2) To access value from hash table, you need to pass name. While in array, to access value, you need to pass index number.  3) you can store different type of data in hash table, say int, string etc. while in array you can store only similar type of data.  **What are differences between system.stringbuilder and system.string?**  The main difference is system.string is immutable and system.stringbuilder is a mutable. Append keyword is used in string builder but not in system.string.  Immutable means once we created we cannot modified. Suppose if we want give new value to old value simply it will discarded the old value and it will create new instance in memory to hold the new value. Below are the few main difference between Abstract Class and Interface  a.    In abstract class method can have definition as well as declaration also. But Interface should have only definition. b.    All the Methods are Public as default and don’t have any access Modifier Controls in interface, whereas for abstract class we can have access modifier for methods. c.    Abstract class can have constructor or destructor, whereas interface not. d.    Abstract class can’t be part of multiple inheritance and we can implement multiple interface. **What is boxing and unboxing concepts in .net?**  **Ans:** Boxing is a process of converting value type into reference type  Unboxing is a process of converting reference type to value type.  **Why we go for page rendering in Asp.Net Page life cycle?**  **Ans:**Browser understands an only html control that’s why in page rendering we will convert the aspx controls into html controls.  **Can we change the index of primary key on table?**  **Ans:**No  **How to find out which index is defined on table?**  **Ans:**sp\_helpindex tablename  **Can you write the program to find the length of string without using library function?**  **Ans:**for (int i=0; str[i]!=”\n”; i++)  {  Count++;  }  **What are difference between GET and POST Methods?**  **Ans:**  **GET Method ():**   1) Data is appended to the URL.  2) Data is not secret.  3) It is a single call system  4) Maximum data that can be sent is 256.  5) Data transmission is faster  6) this is the default method for many browsers   **POST Method ():**   1) Data is not appended to the URL.  2) Data is Secret  3) it is a two call system.  4) There is no Limit on the amount of data. That is characters any amount of data can be sent.  5) Data transmission is comparatively slow.  6) No default and should be explicitly specified.  **If I write System.exit (0); at the end of the try block, will the finally block still execute?**  **Ans:**No in this case the finally block will not execute because when you say system. System.Environment.Exit(0),the control immediately goes out of the program, and thus finally never executes.  **What is dll hell problem in .NET and how it will solve?**  **Ans:**Dll hell, is kind of conflict that occurred previously, due to the lack of version supportability of dll for (within) an application  .NET Framework provides operating system with a global assembly cache. This cache is a repository for all the .net components that are shared globally on a particular machine. When a .net component installed onto the machine, the global assembly cache looks at its version, its public key and its language information and creates a strong name for the component. The component is then registered in the repository and indexed by its strong name, so there is no confusion between the different versions of same component, or DLL When should you use Abstract Class vs Interface while programming?   Ans: When we want that sub class must implement all the methods of base class. In such a situation we will implement the interface. In the other hand when we want only some method of base class in our sub class then use base class as abstract class. **What is the difference between application exception and system exception?**  **Ans:**The difference between application exception and system exception is that system exceptions are thrown by CLR and application exceptions are thrown by applications.  **What is the serialization?**  **Ans:**Serialization is a process of converting object into a stream of bites.  **What is the difference between view state and hidden field?**  **Ans:**viewstate is secured hidden field is insecure  Viewstate will store large amount of data but hidden filed will store small amount of data.  **What is the Difference between read only and constant variables?**  **Ans:**Read only can assign the values at runtime only.  Constant will assign the values at compile time only.  We cannot modify the both variable values.  **What is static keyword in .Net?**  **Ans:** Static is same as constant variable but we can change the value of static variable and we can access the variables without creating any instances What is Normalization? Normalization is the process of efficiently organizing data in a database. There are two goals of the normalization process: eliminating redundant data (for example, storing the same data in more than one [table](http://databases.about.com/library/glossary/bldef-table.htm)) and ensuring data dependencies make sense (only storing related data in a table). Both of these are worthy goals as they reduce the amount of space a database consumes and ensure that data is logically stored. First Normal Form (1NF) First normal form (1NF) sets the very basic rules for an organized database:   * Eliminate duplicative [columns](http://databases.about.com/library/glossary/bldef-column.htm) from the same table. * Create separate tables for each group of related data and identify each [row](http://databases.about.com/library/glossary/bldef-row.htm) with a unique column or set of columns (the [primary key](http://databases.about.com/library/glossary/bldef-primarykey.htm)).  Second Normal Form (2NF) Second normal form (2NF) further addresses the concept of removing duplicative data:   * Meet all the requirements of the first normal form. * Remove subsets of data that apply to multiple rows of a table and place them in separate tables. * Create relationships between these new tables and their predecessors through the use of [foreign keys](http://databases.about.com/library/glossary/bldef-foreignkey.htm).  Third Normal Form (3NF) Third normal form (3NF) goes one large step further:   * Meet all the requirements of the second normal form. * Remove columns that are not dependent upon the primary key.  Fourth Normal Form (4NF) Finally, fourth normal form (4NF) has one additional requirement:   * Meet all the requirements of the third normal form. * A relation is in 4NF if it has no multi-valued dependencies.   Remember, these normalization guidelines are cumulative. For a database to be in 2NF, it must first fulfill all the criteria of a 1NF database. Constraints Constraints allow you to define the way SQL Server automatically enforces the integrity of a database. Constraints define rules regarding the values allowed in columns and this is the standard mechanism for enforcing integrity. |

# [Local and Global temporary table in SQL SERVER](http://stackoverflow.com/questions/2920836/local-and-global-temporary-table-in-sql-server)

Local temporary tables are visible only to their creators during the same connection to an instance of SQL Server as when the tables were first created or referenced. Local temporary tables are deleted after the user disconnects from the instance of SQL Server. Global temporary tables are visible to any user and any connection after they are created, and are deleted when all users that are referencing the table disconnect from the instance of SQL Server.

* **Table variables** (DECLARE @t TABLE) are visible only to the   
  connection that creates it, are stored in RAM, and are deleted when  
  the batch or stored procedure ends.
* **Local temporary tables** (CREATE TABLE #t) are visible only to the connection that creates it, and are deleted when the connection is closed.
* **Global temporary tables** (CREATE TABLE ##t) are visible to everyone, and are deleted when the connection that created it is closed.

# Software Development Life Cycle Models and Methodologies

## Iterative and Incremental Method

##### Description

It is developed to overcome the weaknesses of the waterfall model. It starts with an initial planning and ends with deployment with the cyclic interactions in between. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental), allowing software developers to take advantage of what was learned during development of earlier parts or versions of the system.

##### The usage

It is used in shrink-wrap application and large system which built in small phases or segments. Also can be used in system has separated components, for example, ERP system. Which we can start with budget module as first iteration and then we can start with inventory module and so forth.

##### Advantages and Disadvantages

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| · Produces business value early in the development life cycle· Better use of scarce resources through proper increment definition· Can accommodate some change requests between increments· More focused on customer value than the linear approaches  · Problems can be detected earlier | · Requires heavy documentation· Follows a defined set of processes· Defines increments based on function and feature dependencies· Requires more customer involvement than the linear approaches  · Partitioning the functions and features might be problematic  · Integration between iteration can be an issue if this is not considered during the development. |

## Extreme programing (Agile development)

##### Description

It is based on iterative and incremental development, where requirements and solutions evolve through collaboration between cross-functional teams.

##### The usage

It can be used with any type of the project, but it needs more involvement from customer and to be interactive. Also, it can be used when the customer needs to have some functional requirement ready in less than three weeks.

##### Advantages and Disadvantages

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| · Decrease the time required to avail some system features.· Face to face communication and continuous inputs from customer representative leaves no space for guesswork.· The end result is the high quality software in least possible time duration and satisfied customer | · Scalability· Skill of the software developers· Ability of customer to express user needs· Documentation is done at later stages  · Reduce the usability of components.  · Needs special skills for the team. |

ASP.NET Page Life Cycle Overview

When a page request is sent to the Web server, the page is run through a series of events during its creation and disposal. In this article, I will discuss in detail the ASP.NET page life cycle Events

**(1) PreInit** The entry point of the page life cycle is the pre-initialization phase called “PreInit”. This is the only event where programmatic access to master pages and themes is allowed. You can dynamically set the values of master pages and themes in this event. You can also dynamically create controls in this event.

**EXAMPLE**: Override the event as given below in your code-behind cs file of your aspx page

using System;

using System.Collections.Generic;using System.Linq;using System.Web;using System.Web.UI;using System.Web.UI.WebControls; public partial class \_Default : System.Web.UI.Page{    protected void Page\_PreInit(object se`nder, EventArgs e)    {        //  Use this event for the following:          //  Check the IsPostBack property to determine whether this is the first time the page is being processed.        //  Create or re-create dynamic controls.        //  Set a master page dynamically.        //  Set the Theme property dynamically.           }

------------------------------------**------------------------------------**

**(2)Init**This event fires after each control has been initialized, each control's UniqueID is set and any skin settings have been applied. You can use this event to change initialization values for controls. The “Init” event is fired first for the most bottom control in the hierarchy, and then fired up the hierarchy until it is fired for the page itself. 

**EXAMPLE**: Override the event as given below in your code-behind cs file of your aspx page

protected void Page\_Init(object sender, EventArgs e)

{// Raised after all controls have been initialized and any skin settings have been applied. Use this event to read or initialize control properties.}

 ------------------------------------**-------------------------------**

**(3)InitComplete**Raised once all initializations of the page and its controls have been completed. Till now the viewstate values are not yet loaded, hence you can use this event to make changes to view state that you want to make sure are persisted after the next postback

**EXAMPLE**: Override the event as given below in your code-behind cs file of your aspx page

protected void Page\_InitComplete(object sender, EventArgs e)

{       // Raised by the  Page object. Use this event for processing tasks that require all initialization be complete. }

------------------------------------**------------------------------------**

**(4)PreLoad**Raised after the page loads view state for itself and all controls, and after it processes postback data that is included with the Request instance

**(1)Loads ViewState** : ViewState data are loaded to controls

Note : The page viewstate is managed by ASP.NET and is used to persist information over a page roundtrip to the server. Viewstate information is saved as a string of name/value pairs and contains information such as control text or value. The viewstate is held in the value property of a hidden <input> control that is passed from page request to page request.

**(2)Loads Postback data** : postback data are now handed to the page controls

Note : During this phase of the page creation, form data that was posted to the server (termed postback data in ASP.NET) is processed against each control that requires it. Hence, the page fires the LoadPostData event and parses through the page to find each control and updates the control state with the correct postback data. ASP.NET updates the correct control by matching the control's unique ID with the name/value pair in the NameValueCollection. This is one reason that ASP.NET requires unique IDs for each control on any given page.

**EXAMPLE**: Override the event as given below in your code-behind cs file of your aspx page

protected override void OnPreLoad(EventArgs e)

{        // Use this event if you need to perform processing on your page or control before the  Load event.        // Before the Page instance raises this event, it loads view state for itself and all controls, and then processes any postback data included with the Request instance.}

------------------------------------**------------------------------------**

**(5)Load**The important thing to note about this event is the fact that by now, the page has been restored to its previous state in case of postbacks. Code inside the page load event typically checks for PostBack and then sets control properties appropriately. This method is typically used for most code, since this is the first place in the page lifecycle that all values are restored. Most code checks the value of IsPostBack to avoid unnecessarily resetting state. You may also wish to call Validate and check the value of IsValid in this method. You can also create dynamic controls in this method.

**EXAMPLE**: Override the event as given below in your code-behind cs file of your aspx page

protected void Page\_Load(object sender, EventArgs e)

{        // The  Page calls the  OnLoad event method on the  Page, then recursively does the same for each child control, which does the same for each of its child controls until the page and all controls are loaded.        // Use the OnLoad event method to set properties in controls and establish database connections.}

------------------------------------**------------------------------------**

**(6)Control (PostBack) event(s)**ASP.NET now calls any events on the page or its controls that caused the PostBack to occur. This might be a button’s click event or a dropdown's selectedindexchange event, for example.These are the events, the code for which is written in your code-behind class(.cs file).

**EXAMPLE**: Override the event as given below in your code-behind cs file of your aspx page

protected void Button1\_Click(object sender, EventArgs e)

{        // This is just an example of control event.. Here it is button click event that caused the postback}

------------------------------------**---------------------------------**

**(7)LoadComplete**This event signals the end of Load.

**EXAMPLE**: Override the event as given below in your code-behind cs file of your aspx page

protected void Page\_LoadComplete(object sender, EventArgs e)

{        // Use this event for tasks that require that all other controls on the page be loaded.}

------------------------------------**----------------------------------**

**(8)PreRender**Allows final changes to the page or its control. This event takes place after all regular PostBack events have taken place. This event takes place before saving ViewState, so any changes made here are saved.For example : After this event, you cannot change any property of a button or change any viewstate value. Because, after this event, SaveStateComplete and Render events are called.

**EXAMPLE**: Override the event as given below in your code-behind cs file of your aspx page

protected override void OnPreRender(EventArgs e)

{        // Each data bound control whose DataSourceID property is set calls its DataBind method.        // The PreRender event occurs for each control on the page. Use the event to make final changes to the contents of the page or its controls.}

------------------------------------**-----------------------------------**

**(9)SaveStateComplete**Prior to this event the view state for the page and its controls is set. Any changes to the page’s controls at this point or beyond are ignored.

**EXAMPLE**: Override the event as given below in your code-behind cs file of your aspx page

protected override void OnSaveStateComplete(EventArgs e)

{        // Before this event occurs,  ViewState has been saved for the page and for all controls. Any changes to the page or controls at this point will be ignored.        // Use this event perform tasks that require view state to be saved, but that do not make any changes to controls.}

------------------------------------**------------------------------------**

**(10)Render**This is a method of the page object and its controls (and not an event). At this point, ASP.NET calls this method on each of the page’s controls to get its output. The Render method generates the client-side HTML, Dynamic Hypertext Markup Language (DHTML), and script that are necessary to properly display a control at the browser.

 Note:Right click on the web page displayed at client's browser and view the Page's Source. You will not find any aspx server control in the code. Because all aspx controls are converted to their respective HTML representation. Browser is capable of displaying HTML and client side scripts.

**EXAMPLE**: Override the event as given below in your code-behind cs file of your aspx page

// Render stage goes here. This is not an event

------------------------------------**------------------------------------**

**(11)UnLoad**This event is used for cleanup code. After the page's HTML is rendered, the objects are disposed of. During this event, you should destroy any objects or references you have created in building the page. At this point, all processing has occurred and it is safe to dispose of any remaining objects, including the Page object.Cleanup can be performed on-

     (a)Instances of classes i.e. objects

     (b)Closing opened files

     (c)Closing database connections.

**EXAMPLE**: Override the event as given below in your code-behind cs file of your aspx page

protected void Page\_UnLoad(object sender, EventArgs e)

{        // This event occurs for each control and then for the page. In controls, use this event to do final cleanup for specific controls, such as closing control-specific database connections.        // During the unload stage, the page and its controls have been rendered, so you cannot make further changes to the response stream.          //If you attempt to call a method such as the Response.Write method, the page will throw an exception.    }

------------------------------------------------------------------

**For More Reference** : Follow the below links (1)<http://www.15seconds.com/issue/020102.htm>(2) <http://msdn.microsoft.com/en-us/library/ms178472.aspx>(3)<http://www.devlifestyle.net/blogs/articles/archive/2009/05/24/asp-net-internals-viewstate-and-page-life-cycle.aspx> Thank You...

Page Event

Typical Use

PreInit

Use this event for the following:

Check the IsPostBack property to determine whether this is the first time the page is being processed.

Create or re-create dynamic controls.

Set a master page dynamically.

Set the Theme property dynamically.

Read or set profile property values.

Note:

If the request is a postback, the values of the controls have not yet been restored from view state. If you set a control property at this stage, its value might be overwritten in the next event.

Init

Raised after all controls have been initialized and any skin settings have been applied. Use this event to read or initialize control properties.

InitComplete

Raised by the Page object. Use this event for processing tasks that require all initialization be complete.

PreLoad

Use this event if you need to perform processing on your page or control before the Load event.

Before the Page instance raises this event, it loads view state for itself and all controls, and then processes any postback data included with the Request instance.

Load

The Page calls the OnLoad event method on the Page, then recursively does the same for each child control, which does the same for each of its child controls until the page and all controls are loaded.

Use the OnLoad event method to set properties in controls and establish database connections.

Control events

Use these events to handle specific control events, such as a Button control's Click event or a TextBox control's TextChanged event.

Note:

In a postback request, if the page contains validator controls, check the IsValid property of the Page and of individual validation controls before performing any processing.

LoadComplete

Use this event for tasks that require that all other controls on the page be loaded.

PreRender

Before this event occurs:

The Page object calls EnsureChildControls for each control and for the page.

Each data bound control whose DataSourceID property is set calls its DataBind method. For more information, see Data Binding Events for Data-Bound Controls later in this topic.

The PreRender event occurs for each control on the page. Use the event to make final changes to the contents of the page or its controls.

SaveStateComplete

Before this event occurs, ViewState has been saved for the page and for all controls. Any changes to the page or controls at this point will be ignored.

Use this event perform tasks that require view state to be saved, but that do not make any changes to controls.

Render

This is not an event; instead, at this stage of processing, the Page object calls this method on each control. All ASP.NET Web server controls have a Render method that writes out the control's markup that is sent to the browser.

If you create a custom control, you typically override this method to output the control's markup. However, if your custom control incorporates only standard ASP.NET Web server controls and no custom markup, you do not need to override the Render method. For more information, see Developing Custom ASP.NET Server Controls.

A user control (an .ascx file) automatically incorporates rendering, so you do not need to explicitly render the control in code.

Unload

This event occurs for each control and then for the page. In controls, use this event to do final cleanup for specific controls, such as closing control-specific database connections.

For the page itself, use this event to do final cleanup work, such as closing open files and database connections, or finishing up logging or other request-specific tasks.

Note:

During the unload stage, the page and its controls have been rendered, so you cannot make further changes to the response stream. If you attempt to call a method such as the Response.Write method, the page will throw an exception.

**Named parameters** are an alternate parameter syntax. They sometimes result in easier to read and clearer code. They are checked for correctness by the compiler. By specifying the formal parameter name, you can reorder the actual arguments you pass.

## Example



To start, this example program introduces two methods: the Main entry point, and the Test method. In the Main method, the Test method is called in four different ways.

**We use the** syntax "name:" and then a string literal to specify the value of the name parameter. The syntax "size:" and then an integer signifies the size parameter. You can reorder these named parameters in any way you want.

**Also:**You can specify names on only some parameters.

**Program that uses named parameters [C#]**

using System;

class Program

{

static void Main()

{

// Call the Test method several times in different ways.

Test(name: "Perl", size: 5);

Test(name: "Dot", size: -1);

Test(6, "Net");

Test(7, name: "Google");

}

static void Test(int size, string name)

{

Console.WriteLine("Size = {0}, Name = {1}", size, name);

}

}

**Output**

Size = 5, Name = Perl

Size = -1, Name = Dot

Size = 6, Name = Net

Size = 7, Name = Google

**ACID** (an acronymn for Atomicity Consistency Isolation Durability) is a concept that Database Professionals generally look for when evaluating databases and application architectures. For a reliable database all this four attributes should be achieved.

**Atomicity** is an all-or-none proposition.

**Consistency** guarantees that a transaction never leaves your database in a half-finished state.

**Isolation** keeps transactions separated from each other until they’re finished.

**Durability** guarantees that the database will keep track of pending changes in such a way that the server can recover from an abnormal termination.

**Delegate**

public **delegate** double **Delegate**\_Prod(int a,int b);

class Class1

{

static double fn\_Prodvalues(int val1,int val2)

{

return val1\*val2;

}

static void Main(string[] args)

{

*//Creating the* ***Delegate*** *Instance*

**Delegate**\_Prod delObj = new **Delegate**\_Prod(fn\_Prodvalues);

Console.Write("Please Enter Values");

int v1 = Int32.Parse(Console.ReadLine());

int v2 = Int32.Parse(Console.ReadLine());

*//use a* ***delegate*** *for processing*

double res = delObj(v1,v2);

Console.WriteLine ("Result :"+res);

Console.ReadLine();

}

}

**What are Events?**

As compared to delegates events works with source and listener methodology. So listeners who

are interested in receiving some events they subscribe to the source. Once this subscription is

done, the source raises events to its entire listener when needed. One source can have multiple

listeners.

**What is the difference between Shadowing and Overriding?**

Following are the differences between shadowing and overriding:-

• Overriding redefines only the implementation while shadowing redefines the whole

element.

• In overriding derived classes can refer the parent class element by using “ME” keyword,

but in shadowing you can access it by “MYBASE”.

# Encapsulation

Is the ability of an object to hide its data and methods from the rest of the world. It is one of the fundamental principles of OOPs.

Real time

e.g. / Ink is the important component in pen but it is hiding by some other material

Say we create a class, named Calculations. This class may contain a few members in the form of properties, events, fields or methods. Once the class is created, we may instantiate the class by creating an object out of it. The object acts as an instance of this class, the members of the class are not exposed to the outer world directly; rather, they are encapsulated by the class.

For example code

Public class Calculations

{

  private void fnMultiply(int x, int y)

  {

  return x \* y;

  }

}

...

...

Calculations obj;

int Result;

Result = obj.fnMultiply(5,10)

**ENCAPSULATION USING PROPERTIES:**

Properties are a new language feature introduced with C#. Only a few languages support this property. Properties in C# helps in protect a field in a class by reading and writing to it. The first method itself is good but Encapsulation can be accomplished much smoother with properties.

Now let's see an example.

using system;  
public class Department   
{  
private string departname;  
public string Departname  
{  
get  
{  
return departname;  
}  
set   
{  
departname=value;  
}  
}  
}  
public class Departmentmain  
{  
public static int Main(string[] args)  
{  
Department d= new Department();  
d.departname="Communication";  
Console.WriteLine("The Department is :{0}",d.Departname);  
return 0;  
}   
}

From the above example we see the usage of Encapsulation by using properties. The property has two accessor get and set. The get accessor returns the value of the some property field. The set accessor sets the value of the some property field with the contents of "value". Properties can be made read-only. This is accomplished by having only a get accessor in the property implementation.

The following are the benefits of encapsulation:

* Protection of data from accidental corruption
* Specification of the accessibility of each of the members of a class to the code outside the class
* Flexibility and extensibility of the code and reduction in complexity
* Lower coupling between objects and hence improvement in code maintainability

Encapsulation is used to restrict access to the members of a class so as to prevent the user of a given class from manipulating objects in ways that are not intended by the designer. While encapsulation hides the internal implementation of the functionalities of class without affecting the overall functioning of the system, it allows the class to service a request for functionality and add or modify its internal structure (data or methods) to suit changing requirements.  
  
Encapsulation is also known as information hiding

# Abstraction

"Abstraction" refers to the act of representing essential features without including the background details or explanations. Classes use the concept of abstraction and are defined as a list of abstract attributes.

**What are different types of caching using cache object of**

**ASP.NET?**

You can use two types of output caching to cache information that is to be transmitted to and

displayed in a Web browser:

• **Page Output Caching**

Page output caching adds the response of page to cache object. Later when

page is requested page is displayed from cache rather than creating the

page object and displaying it. Page output caching is good if the site is

fairly static.

• **Page Fragment Caching**

If parts of the page are changing, you can wrap the static sections as user

controls and cache the user controls using page fragment caching.

**WCF**

|  |  |  |
| --- | --- | --- |
| Features | Web Service | WCF |
| Hosting | It can be hosted in IIS | It can be hosted in IIS, windows activation service, Self-hosting, Windows service |
| Programming | [WebService] attribute has to be added to the class | [ServiceContraact] attribute has to be added to the class |
| Model | [WebMethod] attribute represents the method exposed to client | [OperationContract] attribute represents the method exposed to client |
| Operation | One-way, Request- Response are the different operations supported in web service | One-Way, Request-Response, Duplex are different type of operations supported in WCF |
| XML | System.Xml.serialization name space is used for serialization | System.Runtime.Serialization namespace is used for serialization |
| Encoding | XML 1.0, MTOM(Message Transmission Optimization Mechanism), DIME, Custom | XML 1.0, MTOM, Binary, Custom |
| Transports | Can be accessed through HTTP, TCP, Custom | Can be accessed through HTTP, TCP, Named pipes, MSMQ,P2P, Custom |
| Protocols | Security | Security, Reliable messaging, Transactions |

### 1. What is the difference between WCF and ASMX Web Services?

Simple and basic difference is that ASMX or ASP.NET web service is designed to send and receive messages using SOAP over HTTP only. While WCF can exchange messages using any format (SOAP is default) over any transport protocol (HTTP, TCP/IP, MSMQ, NamedPipes etc).

Another tutorial [WCF Vs ASMX](http://wcfanswers.blogspot.com/2012/06/wcf-vs-asmx-web-services.html) has detailed discussion on it.

### 2. What are WCF Service Endpoints? Explain.

For **Windows Communication Foundation**services to be consumed, it’s necessary that it must be exposed; Clients need information about service to communicate with it. This is where service endpoints play their role.

A **WCF service** endpoint has three basic elements

i.e. Address, Binding and Contract.

* **Address:** It defines "WHERE". Address is the URL that identifies the location of the service.
* **Binding:** It defines "HOW". Binding defines how the service can be accessed.
* **Contract:** It defines "WHAT". Contract identifies what is exposed by the service.

### 3. What are the possible ways of hosting a WCF service? Explain.

For a **Windows Communication Foundation** service to host, we need at least a managed process, a ServiceHost instance and an Endpoint configured. Possible approaches for hosting a service are:

1. Hosting in a Managed Application/ Self Hosting
   1. Console Application
   2. Windows Application
   3. Windows Service
2. Hosting on Web Server
   1. IIS 6.0 (ASP.NET Application supports only HTTP)
   2. Windows Process Activation Service (WAS) i.e. IIS 7.0 supports HTTP, TCP, NamedPipes, MSMQ.

### 4. How we can achieve Operation Overloading while exposing WCF Services?

By default, WSDL doesn’t support operation overloading. Overloading behavior can be achieved by using "Name" property of OperationContract attribute.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/426776/WCF-Top-10-Interview-Questions)

[ServiceContract]

interface IMyCalculator

{

[OperationContract(Name = "SumInt")]

int Sum(int arg1,int arg2);

[OperationContract(Name = "SumDouble")]

double Sum(double arg1,double arg2);

}

When the proxy will be generated for these operations, it will have 2 methods with different names i.e. SumInt and SumDouble.

**5. What Message Exchange Patterns (MEPs) supported by WCF? Explain each of them briefly.**

1. Request/Response 2. One Way 3. Duplex

#### Request/Response

It’s the default pattern. In this pattern, a response message will always be generated to consumer when the operation is called, even with the void return type. In this scenario, response will have empty SOAP body.

#### One Way

In some cases, we are interested to send a message to service in order to execute certain business functionality but not interested in receiving anything back. OneWay MEP will work in such scenarios. If we want queued message delivery, OneWay is the only available option.

#### Duplex

The Duplex MEP is basically a two-way message channel. In some cases, we want to send a message to service to initiate some longer-running processing and require a notification back from service in order to confirm that the requested process has been completed.

### 6. What is DataContractSerializer and How its different from XmlSerializer?

Serialization is the process of converting an object instance to a portable and transferable format. So, whenever we are talking about **web services**, serialization is very important.

Windows Communication Foundation has DataContractSerializer that is new in .NET 3.0 and uses opt-in approach as compared to XmlSerializer that uses opt-out. Opt-in means specify whatever we want to serialize while Opt-out means you don’t have to specify each and every property to serialize, specify only those you don’t want to serialize. DataContractSerializer is about 10% faster than XmlSerializer but it has almost no control over how the object will be serialized. If we wanted to have more control over how object should be serialized that XmlSerializer is a better choice.

### 7. How we can use MessageContract partially with DataContract for a service operation in WCF?

MessageContract must be used all or none. If we are using MessageContract into an operation signature, then we must use MessageContract as the only parameter type and as the return type of the operation.

### 8. Which standard binding could be used for a service that was designed to replace an existing ASMX web service?

The basicHttpBinding standard binding is designed to expose a service as if it is an ASMX/ASP.NET web service. This will enable us to support existing clients as applications are upgrade to WCF.

### 9. Please explain briefly different Instance Modes in WCF?

WCF will bind an incoming message request to a particular service instance, so the available modes are:

* **Per Call**: instance created for each call, most efficient in term of memory but need to maintain session.
* **Per Session**: Instance created for a complete session of a user. Session is maintained.
* **Single**: Only one instance created for all clients/users and shared among all. Least efficient in terms of memory.

### 10. Please explain different modes of security in WCF? Or Explain the difference between Transport and Message Level Security.

In Windows Communication Foundation, we can configure to use security at different levels

**a. Transport Level security** means providing security at the transport layer itself. When dealing with security at Transport level, we are concerned about integrity, privacy and authentication of message as it travels along the physical wire. It depends on the binding being used that how WCF makes it secure because most of the bindings have built-in security.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/426776/WCF-Top-10-Interview-Questions)

<netTcpBinding>

<binding name="netTcpTransportBinding">

<security mode="Transport">

<Transport clientCredentialType="Windows" />

</security>

</binding>

</netTcpBinding>

**b. Message Level Security** For Tranport level security, we actually ensure the transport that is being used should be secured but in message level security, we actually secure the message. We encrypt the message before transporting it.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/426776/WCF-Top-10-Interview-Questions)

<wsHttpBinding>

<binding name="wsHttpMessageBinding">

<security mode="Message">

<Message clientCredentialType="UserName" />

</security>

</binding>

</wsHttpBinding>

It totally depends upon the requirements but we can use a mixed security mode also as follows:

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/426776/WCF-Top-10-Interview-Questions)

<basicHttpBinding>

<binding name="basicHttp">

<security mode="TransportWithMessageCredential">

<Transport />

<Message clientCredentialType="UserName" />

</security>

</binding>

</basicHttpBinding>

**(I) what are the advantages of hosting WCF Services in IIS as**

**compared to self-hosting?**

There are two main advantages of using IIS over self-hosting:-

**Automatic activation**

IIS provides automatic activation that means the service is not necessary to be running in

advance. When any message is received by the service it then launches and fulfills the request.

But in case of self hosting the service should always be running.

**Process recycling**

If IIS finds that a service is not healthy that means if it has memory leaks etc, IIS recycles the

process. Ok let us try to understand what is recycling in IIS process. For every browser instance, a

worker process is spawned and the request is serviced. When the browser disconnects the worker,

process stops and you loose all information. IIS also restarts the worker process. By default, the

worker process is recycled at around 120 minutes. So why does IIS recycle. By restarting the

worker process it ensures any bad code or memory leak do not cause issue to the whole system.

In case of self-hosting both the above features, you will need to code yourself. Lot of work

right!!. That is why IIS is the best option for hosting services until you are really doing something

custom.

Below figure shows where the recycle option is located in IIS. You need to click on the

DefaultAppool and then Properties.

**Introduction**

Here I will explain about differences between Asp.net 3.5 and Asp.net 4.0.

**Description**

In previous post I explained [differences between asp.net 1.0 and asp.net 2.0](http://www.aspdotnet-suresh.com/2011/10/what-is-differences-between-aspnet-10.html) and in another post I explained [differences between asp.net 2.0 and asp.net 3.5](http://www.aspdotnet-suresh.com/2011/10/what-are-difference-between-aspnet-20.html) . Now I will explain about differences between asp.net 3.5 and asp.net 4.0.

**Check below table for Asp.net 3.5 Features**

|  |  |  |
| --- | --- | --- |
| **SNo** | **Feature** | **Asp.net 3.5** |
| 1. | New Features | Asp.net 3.5 includes the following new features         1) Listview Control         2) DataPager Control         3) Nested Master Pages         4) Linq DataSource |
| 2. | Multi Targeting | Asp.net 3.5 supports multi –targeting **What is multi – targeting?** Check below description |
| 3. | Ajax Support | In Asp.net 3.5, Ajax is integrated in .NET framework, thereby making the process of building intuitive cool user interfaces easier |
| 4. | Silverlight Support | It support for silverlight |
| 5. | JavaScript Debugging | It support for JavaScript debugging |
| 6. | LINQ Support | It supports LINQ |

### Check below details for Asp.net 4.0 Features

### In Asp.net 4.0 many new features has included in the .NET Framework 4 in visual studio 2010.

## Web.config File Refactoring

The Web.config file that contains the configuration for a Web application has grown considerably over the past few releases of the .NET Framework as new features have been added, such as Ajax, routing, and integration with IIS 7. This has made it harder to configure or start new Web applications without a tool like Visual Studio. In .the NET Framework 4, the major configuration elements have been moved to themachine.config file, and applications now inherit these settings. This allows the Web.config file in ASP.NET 4 applications either to be empty or to contain just the following lines, which specify for Visual Studio what version of the framework the application is targeting:

|  |
| --- |
| <?xml version="1.0"?>  <configuration>  <system.web>  <compilation targetFramework="4.0" />  </system.web>  </configuration |

**Extensible Output Caching**

ASP.NET 4 adds an extensibility point to output caching that enables you to configure one or more custom output-cache providers. Output-cache providers can use any storage mechanism to persist HTML content. This makes it possible to create custom output-cache providers for diverse persistence mechanisms, which can include local or remote disks, cloud storage, and distributed cache engines.  
  
**Permanently Redirecting a Page**  
  
ASP.NET 4 adds a new RedirectPermanent helper method that makes it easy to issue HTTP 301 Moved Permanently responses, as in the following example:  
    
RedirectPermanent("/newpath/foroldcontent.aspx");   
Search engines and other user agents that recognize permanent redirects will store the new URL 

that is associated with the content, which eliminates the unnecessary round trip made by the

browser for temporary redirects.

## Shrinking Session State

ASP.NET 4 introduces a new compression option for both kinds of out-of-process session-state providers. When the compressionEnabled configuration option shown in the following example is set to true, ASP.NET will compress (and decompress) serialized session state by using the .NET FrameworkSystem.IO.Compression.GZipStream class.

|  |
| --- |
| <sessionState mode="SqlServer" sqlConnectionString="data source=dbserver;Initial Catalog=aspnetstate"allowCustomSqlDatabase="true" compressionEnabled="true"/> |

## Multi-Targeting

Asp.net 4 also supports Multi Targeting check this post for [multi- targeting](http://www.aspdotnet-suresh.com/2011/10/what-are-difference-between-aspnet-20.html) 

## jQuery Included with Web Forms and MVC

The Visual Studio templates for both Web Forms and MVC include the open-source jQuery library. When you create a new website or project, a Scripts folder containing the following 3 files is created:

* jQuery-1.4.1.js – The human-readable, unminified version of the jQuery library.
* jQuery-14.1.min.js – The minified version of the jQuery library.
* jQuery-1.4.1-vsdoc.js – The Intellisense documentation file for the jQuery library.

Include the unminified version of jQuery while developing an application. Include the minified version of jQuery for production applications.   
  
For example, the following Web Forms page illustrates how you can use jQuery to change the background color of ASP.NET TextBox controls to yellow when they have focus. 

|  |
| --- |
| <html xmlns="http://www.w3.org/1999/xhtml">  <head id="Head1" runat="server">  <title>Show jQuery</title>  </head>  <body>  <form id="form1" runat="server">  <div>  <asp:TextBox ID="txtFirstName" runat="server" />  <br />  <asp:TextBox ID="txtLastName" runat="server" />  </div>  </form>  <script src="Scripts/jquery-1.4.1.js" type="text/javascript"></script>  <script type="text/javascript">  $("input").focus( function() { $(this).css("background-color", "yellow"); });  </script>  </body>  </html> |

## Content Delivery Network Support

The Microsoft Ajax Content Delivery Network (CDN) enables you to easily add ASP.NET Ajax and jQuery scripts to your Web applications. For example, you can start using the jQuery library simply by adding a<script> tag to your page that points to Ajax.microsoft.com like this:

<script src="http://ajax.microsoft.com/ajax/jquery/jquery-1.4.2.js" type="text/javascript"></script>

## ScriptManager Explicit Scripts

In the past, if you used the ASP.NET ScriptManger then you were required to load the entire monolithic ASP.NET Ajax Library. By taking advantage of the new ScriptManager.AjaxFrameworkMode property, you can control exactly which components of the ASP.NET Ajax Library are loaded and load only the components of the ASP.NET Ajax Library that you need.  
The ScriptManager.AjaxFrameworkMode property can be set to the following values:

* Enabled -- Specifies that the ScriptManager control automatically includes the MicrosoftAjax.js script file, which is a combined script file of every core framework script (legacy behavior).
* Disabled -- Specifies that all Microsoft Ajax script features are disabled and that the ScriptManager control does not reference any scripts automatically.
* Explicit -- Specifies that you will explicitly include script references to individual framework core script file that your page requires, and that you will include references to the dependencies that each script file requires.

For example, if you set the AjaxFrameworkMode property to the value Explicit then you can specify the particular ASP.NET Ajax component scripts that you need:

|  |
| --- |
| <asp:ScriptManager ID="sm1" AjaxFrameworkMode="Explicit" runat="server">  <Scripts>  <asp:ScriptReference Name="MicrosoftAjaxCore.js" />  <asp:ScriptReference Name="MicrosoftAjaxComponentModel.js" />  <asp:ScriptReference Name="MicrosoftAjaxSerialization.js" />  <asp:ScriptReference Name="MicrosoftAjaxNetwork.js" />  </Scripts>  </asp:ScriptManager> |

# Web Forms

Web Forms has been a core feature in ASP.NET since the release of ASP.NET 1.0. Many enhancements have been in this area for ASP.NET 4, including the following:

* The ability to set meta tags.
* More control over view state.
* Easier ways to work with browser capabilities.
* Support for using ASP.NET routing with Web Forms.
* More control over generated IDs.
* The ability to persist selected rows in data controls.
* More control over rendered HTML in the FormView and ListView controls.
* Filtering support for data source controls.

## Setting Meta Tags with the Page.MetaKeywords and Page.MetaDescription Properties

ASP.NET 4 adds two properties to the Page class, MetaKeywords and MetaDescription. These two properties represent corresponding meta tags in your page, as shown in the following example:

|  |
| --- |
| <head id="Head1" runat="server">  <title>Untitled Page</title>  <meta name="keywords" content="These, are, my, keywords" />  <meta name="description" content="This is the description of my page" />  </head> |

## Enabling View State for Individual Controls

By default, view state is enabled for the page, with the result that each control on the page potentially stores view state even if it is not required for the application. View state data is included in the markup that a page generates and increases the amount of time it takes to send a page to the client and post it back. Storing more view state than is necessary can cause significant performance degradation. In earlier versions of ASP.NET, developers could disable view state for individual controls in order to reduce page size, but had to do so explicitly for individual controls. In ASP.NET 4, Web server controls include aViewStateMode property that lets you disable view state by default and then enable it only for the controls that require it in the page. 

## Changes to Browser Capabilities

In ASP.NET 4, browser definition files have been updated to contain information about recently introduced browsers and devices such as Google Chrome, Research in Motion BlackBerry smartphones, and Apple iPhone.

## Routing in ASP.NET 4

ASP.NET 4 adds built-in support for using routing with Web Forms. Routing lets you configure an application to accept request URLs that do not map to physical files. Instead, you can use routing to define URLs that are meaningful to users and that can help with search-engine optimization (SEO) for your application. For example, the URL for a page that displays product categories in an existing application might look like the following example:

http://website/products.aspx?categoryid=12

By using routing, you can configure the application to accept the following URL to render the same information:

http://website/products/software

## Persisting Row Selection in Data Controls

The GridView and ListView controls can let users select a row. In previous versions of ASP.NET, selection has been based on the row index on the page. For example, if you select the third item on page 1 and then move to page 2, the third item on that page is selected.   
Persisted selection was initially supported only in Dynamic Data projects in the .NET Framework 3.5 SP1. When this feature is enabled, the current selected item is based on the data key for the item. This means that if you select the third row on page 1 and move to page 2, nothing is selected on page 2. When you move back to page 1, the third row is still selected. Persisted selection is now supported for the GridViewand ListView controls in all projects by using the EnablePersistedSelection property, as shown in the following example:

|  |
| --- |
| <asp:GridView id="GridView2" runat="server" EnablePersistedSelection="true">  </asp:GridView> |

**ASP.NET Chart Control**

The ASP.NET Chart control expands the data-visualization offerings in the .NET Framework. Using theChart control, you can create ASP.NET pages that have intuitive and visually compelling charts for complex statistical or financial analysis. The ASP.NET Chart control was introduced as an add-on to the .NET Framework version 3.5 SP1 release and is part of the .NET Framework 4 release.   
The control includes the following features:

* 35 distinct chart types.
* An unlimited number of chart areas, titles, legends, and annotations.
* A wide variety of appearance settings for all chart elements.
* 3-D support for most chart types.
* Smart data labels that can automatically fit around data points.
* Strip lines, scale breaks, and logarithmic scaling.
* More than 50 financial and statistical formulas for data analysis and transformation.
* Simple binding and manipulation of chart data.
* Support for common data formats such as dates, times, and currency.
* Support for interactivity and event-driven customization, including client click events using Ajax.
* State management.
* Binary streaming.

## ListView Control Enhancements

The ListView control has been made easier to use in ASP.NET 4. The earlier version of the control required that you specify a layout template that contained a server control with a known ID. The following markup shows a typical example of how to use the ListView control in ASP.NET 3.5.

|  |
| --- |
| <asp:ListView ID="ListView1" runat="server">  <LayoutTemplate>  <asp:PlaceHolder ID="ItemPlaceHolder" runat="server"></asp:PlaceHolder>  </LayoutTemplate>  <ItemTemplate>  <% Eval("LastName")%>  </ItemTemplate>  </asp:ListView> |

In ASP.NET 4, the ListView control does not require a layout template. The markup shown in the previous example can be replaced with the following markup:

|  |
| --- |
| <asp:ListView ID="ListView1" runat="server">  <ItemTemplate>  <% Eval("LastName")%>  </ItemTemplate>  </asp:ListView> |

## Menu Control Improvements

Before ASP.NET 4, the Menu control rendered a series of HTML tables. This made it more difficult to apply CSS styles outside of setting inline properties and was also not compliant with accessibility standards.  
  
In ASP.NET 4, the control now renders HTML using semantic markup that consists of an unordered list and list elements. 

## Wizard and CreateUserWizard Controls

The ASP.NET Wizard and CreateUserWizard controls support templates that let you define the HTML that they render.   
**ASP.NET MVC**  
  
ASP.NET MVC was introduced as an add-on framework to ASP.NET 3.5 SP1 in March 2009. Visual Studio 2010 includes ASP.NET MVC 2, which includes new features and capabilities.  
  
**Dynamic Data**  
  
Dynamic Data was introduced in the .NET Framework 3.5 SP1 release in mid-2008. This feature provides many enhancements for creating data-driven applications, including the following:

* A RAD experience for quickly building a data-driven Web site.
* Automatic validation that is based on constraints defined in the data model.
* The ability to easily change the markup that is generated for fields in the GridView and DetailsViewcontrols by using field templates that are part of your Dynamic Data project.

For ASP.NET 4, Dynamic Data has been enhanced to give developers even more power for quickly building data-driven Web sites.

## Enabling Dynamic Data for Existing Projects

Dynamic Data features that shipped in the .NET Framework 3.5 SP1 brought new features such as the following:

* Field templates – These provide data-type-based templates for data-bound controls. Field templates provide a simpler way to customize the look of data controls than using template fields for each field.
* Validation – Dynamic Data lets you use attributes on data classes to specify validation for common scenarios like required fields, range checking, type checking, pattern matching using regular expressions, and custom validation. Validation is enforced by the data controls.

## Declarative DynamicDataManager Control Syntax

The DynamicDataManager control has been enhanced so that you can configure it declaratively, as with most controls in ASP.NET, instead of only in code. The markup for the DynamicDataManager control looks like the following example:

|  |
| --- |
| <asp:DynamicDataManager ID="DynamicDataManager1" runat="server" AutoLoadForeignKeys="true">  <DataControls>  <asp:DataControlReference ControlID="GridView1" />  </DataControls>  </asp:DynamicDataManager>  <asp:GridView id="GridView1" runat="server"  </asp:GridView> |

This markup enables Dynamic Data behavior for the GridView1 control that is referenced in theDataControls section of the DynamicDataManager control.

## New Field Templates for URLs and E-mail Addresses

ASP.NET 4 introduces two new built-in field templates, EmailAddress.ascx and Url.ascx. These templates are used for fields that are marked as EmailAddress or Url with the DataType attribute. For EmailAddressobjects, the field is displayed as a hyperlink that is created by using the mailto: protocol. When users click the link, it opens the user's e-mail client and creates a skeleton message. Objects typed as Url are displayed as ordinary hyperlinks.   
The following example shows how fields would be marked. 

|  |
| --- |
| [DataType(DataType.EmailAddress)]  public object HomeEmail { get; set; }  [DataType(DataType.Url)]  public object Website { get; set; } |

## Creating Links with the DynamicHyperLink Control

Dynamic Data uses the new routing feature that was added in the .NET Framework 3.5 SP1 to control the URLs that end users see when they access the Web site. The new DynamicHyperLink control makes it easy to build links to pages in a Dynamic Data site. The following example shows how to use theDynamicHyperLink control:

|  |
| --- |
| <asp:DynamicHyperLink ID="ListHyperLink" runat="server" Action="List" TableName="Products">  Show all products  </asp:DynamicHyperLink> |

This markup creates a link that points to the List page for the Products table based on routes that are defined in the Global.asax file. The control automatically uses the default table name that the Dynamic Data page is based on.

## Support for Inheritance in the Data Model

Both the Entity Framework and LINQ to SQL support inheritance in their data models. An example of this might be a database that has an InsurancePolicy table. It might also contain CarPolicy and HousePolicytables that have the same fields as InsurancePolicy and then add more fields. Dynamic Data has been modified to understand inherited objects in the data model and to support scaffolding for the inherited tables.

## Support for Many-to-Many Relationships (Entity Framework Only)

The Entity Framework has rich support for many-to-many relationships between tables, which is implemented by exposing the relationship as a collection on an Entity object. New ManyToMany.ascx andManyToMany\_Edit.ascx field templates have been added to provide support for displaying and editing data that is involved in many-to-many relationships.

# Visual Studio 2010 Web Development Improvements

Web development in Visual Studio 2010 has been enhanced for greater CSS compatibility, increased productivity through HTML and ASP.NET markup snippets and new dynamic IntelliSense JavaScript.

## Improved CSS Compatibility

The Visual Web Developer designer in Visual Studio 2010 has been updated to improve CSS 2.1 standards compliance. The designer better preserves integrity of the HTML source and is more robust than in previous versions of Visual Studio. 

## HTML and JavaScript Snippets

Visual Studio 2010 includes over 200 snippets that help you auto-complete common ASP.NET and HTML tags, including required attributes (such as runat="server") and common attributes specific to a tag (such as ID, DataSourceID, ControlToValidate, and Text). 

## JavaScript IntelliSense Enhancements

In Visual 2010, JavaScript IntelliSense has been redesigned to provide an even richer editing experience. 

# Web Application Deployment with Visual Studio 2010

When ASP.NET developers deploy a Web application, they often find that they encounter issues such as the following:  
Web deployment features in Visual Studio 2010 include the following major areas:

* Web packaging
* Web.config transformation
* Database deployment
* One-click publish for Web applications

The following sections provide details about these features.

## Web Packaging

Visual Studio 2010 uses the MSDeploy tool to create a compressed (.zip) file for your application, which is referred to as a Web package. The package file contains metadata about your application plus the following content:

* IIS settings, which includes application pool settings, error page settings, and so on.
* The actual Web content, which includes Web pages, user controls, static content (images and HTML files), and so on.
* SQL Server database schemas and data.
* Security certificates, components to install in the GAC, registry settings, and so on.

## Web.config Transformation

For Web application deployment, Visual Studio 2010 introduces XML Document Transform (XDT), which is a feature that lets you transform a Web.config file from development settings to production settings. Transformation settings are specified in transform files named web.debug.config, web.release.config, and so on.   
The following example shows a portion of a web.release.config file that might be produced for deployment of your release configuration. The Replace keyword in the example specifies that during deployment theconnectionString node in the Web.config file will be replaced with the values that are listed in the example.

|  |
| --- |
| <connectionStrings xdt:Transform="Replace">  <add name="BlogDB" connectionString="connection string detail]" />  </connectionStrings> |

## Database Deployment

A Visual Studio 2010 deployment package can include dependencies on SQL Server databases. As part of the package definition, you provide the connection string for your source database. When you create the Web package, Visual Studio 2010 creates SQL scripts for the database schema and optionally for the data, and then adds these to the package. 

## One-Click Publish for Web Applications

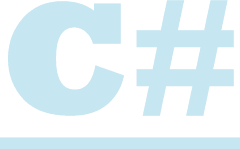
Visual Studio 2010 also lets you use the IIS remote management service to publish a Web application to a remote server.

**Jquery.Finish();**

When .finish() is called on an element, the currently-running animation and all queued animations (if any) immediately stop and their CSS properties set to their target values. All queued animations are removed.

The **string.Join** method combines many strings into one. It receives an array or IEnumerable type and a separator string. It places the separator string between every element of the collection in the returned string.

## Example



First, we combine strings in an array or List into a new single string with dividing characters in it. This example will produce the output with separating commas. The output data type is a string.

[**String Array**](http://www.dotnetperls.com/string-array)

**Note:**Join is a static method. It concatenates strings together with a separator string in between them.

[**Static Method**](http://www.dotnetperls.com/static-method)

**Program that joins strings: C#**

using System;

class Program

{

static void Main()

{

string[] arr = { "one", "two", "three" };

// "string" can be lowercase.

Console.WriteLine(**string.Join**(",", arr));

// ... "String" can be uppercase.

Console.WriteLine(**String.Join**(",", arr));

}

}

**Output**

one,two,three

one,two,three



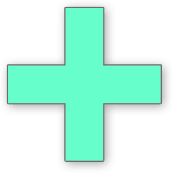
**The first line** in Main declares a new string array with three elements. The second two lines display the result of String.Join to the screen. You can use uppercase or lowercase String. You do not need to specify a string instance.

**Parameters:**You can specify four arguments on string.Join. The last two are the startIndex and the count.

**Note:**This overload is rarely useful in my experience, but could simplify some code.

[**Overload**](http://www.dotnetperls.com/overload)

## HTML



We can use string.Join to concatenate strings of HTML. Often with HTML you need a separating tag or element, such as a <br/> tag or horizontal rule. Join helps because it doesn't insert the separating tag at the end.

**Next:**The strings are concatenated with Join into four lines of markup in HTML, separated by the BR tag.

**Program that joins HTML strings: C#**

using System;

class Program

{

static void Main()

{

// Problem: combine these words into lines in HTML

string[] dinosaurs = new string[] { "Aeolosaurus",

"Deinonychus", "Jaxartosaurus", "Segnosaurus" };

// Solution: join with break tag.

string html = **string.Join**("<br/>\r\n", dinosaurs);

Console.WriteLine(html);

}

}

**Output**

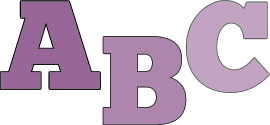
Aeolosaurus<br/>

Deinonychus<br/>

Jaxartosaurus<br/>

Segnosaurus

## StringBuilder



We can replace confusing code that appends strings in loops with much simpler string.Join code. The string.Join method is often much faster in addition to being simpler. The two methods below, CombineA and CombineB, have the same output.

**Note:**CombineA combines strings with Join.  
CombineB combines strings with StringBuilder and its Append method.

**Program that combines strings with Join: C#**

using System;

using System.Text;

class Program

{

static void Main()

{

string[] catSpecies = { "Aegean", "Birman", "Main Coon", "Nebulung" };

Console.WriteLine(**CombineA**(catSpecies));

Console.WriteLine(**CombineB**(catSpecies));

}

/// <summary>

/// Combine strings with commas.

/// </summary>

static string **CombineA**(string[] arr)

{

return string.Join(",", arr);

}

/// <summary>

/// Combine strings with commas.

/// </summary>

static string **CombineB**(string[] arr)

{

StringBuilder builder = new StringBuilder();

foreach (string s in arr)

{

builder.Append(s).Append(",");

}

return builder.ToString().TrimEnd(new char[] { ',' });

}

}

**Output**

Aegean,Birman,Main Coon,Nebulung

Aegean,Birman,Main Coon,Nebulung



**In this example,** the two methods CombineA and CombineB both concatenate each string into a single string with separators. The final method, CombineB, has to use ToString and TrimEnd to convert the StringBuilder into the result.

[**ToString: StringBuilder**](http://www.dotnetperls.com/stringbuilder-tostring)[**TrimEnd**](http://www.dotnetperls.com/trimend)

**Comparison.** String.Join is different from appending many strings together in a loop, such as with StringBuilder—it does not insert the delimiter at the end. Instead it only inserts the delimiter in between the strings.

## Exceptions



String.Join can throw three different exceptions: ArgumentNullException, ArgumentOutOfRangeException and OutOfMemoryException. The first two exceptions are often possible. The following example shows one possible exception.

**Note:**The exceptions encountered from String.Join are detailed in separate pages on this site.

[**ArgumentNullException**](http://www.dotnetperls.com/argumentnullexception)[**ArgumentOutOfRangeException**](http://www.dotnetperls.com/argumentoutofrangeexception)[**OutOfMemoryException**](http://www.dotnetperls.com/outofmemoryexception)

**Program that throws exception on Join: C#**

using System;

class Program

{

static void Main()

{

try

{

string bug = string.Join(null, null); // Null arguments are bad

}

catch (Exception ex)

{

Console.WriteLine(ex);

}

}

}

**Output**

System.ArgumentNullException: Value cannot be null.

Parameter name: value

**This code shows** what happens when you call string.Join with null parameters. It will throw an ArgumentNullException. Depending on your application, this must be dealt with using exception handling.

[**Exception Handling**](http://www.dotnetperls.com/exception)

## Join List



It is possible to join a List generic. Please note that this example includes the System.Collections.Generic namespace. In the Main entry point, a List is instantiated with three string literals in it.

[**List**](http://www.dotnetperls.com/list)[**String Literal**](http://www.dotnetperls.com/string-literal)

**Next,** the string.Join<string> method is invoked. The first argument indicates the separator, and the second argument is a reference to the List instance. The method returns a joined string containing the separator.

**Tip:**By using this version of the string.Join<string> method, you can reduce copies of your collection before joining.

**And:**For this reason, it is preferable to use this version on your List if you do not have an array of your strings handy.

**Program that joins List of strings: C#**

using System;

using System.Collections.Generic;

class Program

{

static void Main()

{

// Create a List of three strings.

var list = new List<string>() { "cat", "dog", "rat" };

// Join the strings from the List.

string joined = **string.Join<string>**("\*", list);

// Display.

Console.WriteLine(joined);

}

}

**Output**

cat\*dog\*rat

## Benchmark



Continuing on, we test the general performance of string.Join. I wanted to see the ballpark numbers for string.Join to ensure that it doesn't cause a severe slowdown. We see that string.Join performs well—often better than loops.

**Data used in benchmark**

string[] arr = { "one", "two", "three", "four", "five" };

**Methods that were benchmarked: C#**

1000000 iterations were tested.

static string CombineA(string[] arr)

{

return string.Join(",", arr);

}

static string CombineB(string[] arr)

{

var builder = new System.Text.StringBuilder();

foreach (string s in arr)

{

builder.Append(s).Append(",");

}

return builder.ToString(); // Has ending comma [difference]

}

**Results**

string.Join: 157 ms [faster]

StringBuilder Append method: 270 ms

**Required Join method results**

Input: one

two

three

Output: one\*two\*three

Polymorphism means having more than one form. **Overloading** and **overriding** are used to implement polymorphism. Polymorphism is classified into compile time polymorphism or early binding or static binding and Runtime polymorphism or late binding or dynamic binding.

When a message can be processed in different ways is called polymorphism. Polymorphism means many forms.

Polymorphism is one of the fundamental concepts of OOP.

**Polymorphism provides following features:**

* It allows you to invoke methods of derived class through base class reference during runtime.
* It has the ability for classes to provide different implementations of methods that are called through the same name.

**Polymorphism is of two types:**

1. Compile time polymorphism/Overloading
2. Runtime polymorphism/Overriding

**Compile Time Polymorphism**

Compile time polymorphism is method and operators overloading. It is also called early binding.

In method overloading method performs the different task at the different input parameters.

# Runtime Time Polymorphism

Runtime time polymorphism is done using inheritance and virtual functions. Method overriding is called runtime polymorphism. It is also called late binding.

When **overriding** a method, you change the behavior of the method for the derived class. **Overloading** a method simply involves having another method with the same prototype.

**Caution:**Don't confused method overloading with method overriding, they are different, unrelated concepts. But they sound similar.

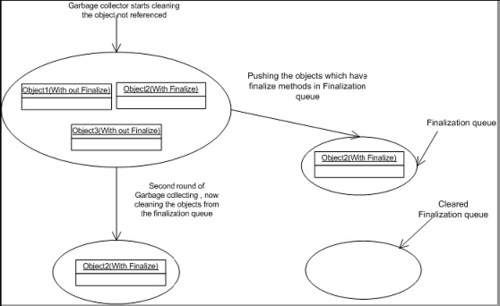
### What is the significance of Finalize method in .NET?

.NET Garbage collector does almost all clean up activity for your objects. But unmanaged resources (example: Windows API created objects, File, Database connection objects, COM objects, etc.) are outside the scope of .NET Framework. We have to explicitly clean our resources. For these types of objects, .NET Framework provides Object.Finalize method, which can be overridden and clean up code for unmanaged resources can be put in this section?

### Why is it preferred to not use finalize for clean up?

The problem with finalize is that garbage collection has to make two rounds in order to remove objects which have finalize methods.

The below figure will make things clear regarding the two rounds of garbage collection rounds performed for the objects having finalized methods.



**Note**: Few of the contents have been taken from various blogs/articles.

In this scenario, there are three objects, Object1, Object2, and Object3. Object2 has the finalize method overridden and remaining objects do not have the finalize method overridden.

Now when garbage collector runs for the first time, it searches for objects whose memory has to free. He can see three objects but only cleans the memory for Object1 and Object3.Object2 it pushes to the finalization queue.

Now garbage collector runs for the second time. He sees there are no objects to be released and then checks for the finalization queue and at this moment, it clears object2 from the memory. So if you notice, object2 was released from memory in the second round and not first. That is why the best practice is not to write clean up Non.NET resources in Finalizemethod rather use the DISPOSE.

### What is the use of DISPOSE method?

Dispose method belongs to ‘IDisposable’ interface. We had seen in the previous section how bad it can be to override the finalize method for writing the cleaning of unmanaged resources. So if any object wants to release its unmanaged code, the best is to implement IDisposable and override the Dispose method of IDisposable interface. Now once your class has exposed the Dispose method, it is the responsibility of the client to call the Dispose method to do the cleanup. How do I force the Dispose method to be called automatically, as clients can forget to call Dispose method?

Call the Dispose method in Finalize method and in Dispose method, suppress thefinalize method using GC.SuppressFinalize. Below is the sample code of the pattern. This is the best way we do clean our unallocated resources and yes not to forget we do not get the hit of running the Garbage collector twice.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/637480/Csharp-and-ASP-NET-Interview-Question-and-Answers)

public class CleanClass : IDisposable

{

public void Dispose()

{

GC.SuppressFinalize(this);

}

protected override void Finalize()

{

Dispose();

}

}

**Note**: Few of the content is taken from various blogs/articles.

What is connection pooling and how do you make your application use it?

Opening database connection is a time consuming operation.

Connection pooling increases the performance of the applications by reusing the active database connections instead of creating new connection for every request.

Connection pooling behaviour is controlled by the connection string parameters.

Following are the 4 parameters that control most of the connection pooling behaviour:

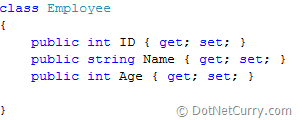
1. Connect Timeout
2. Max Pool Size
3. Min Pool Size
4. Pooling

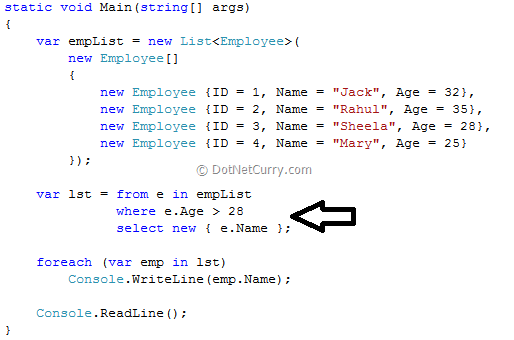
### What does the "EnableViewState" property do? Why would I want it on or off?

Enable ViewState turns on the automatic state management feature that enables server controls to re-populate their values on a round trip without requiring you to write any code. This feature is not free however, since the state of a control is passed to and from the server in a hidden form field. You should be aware of when ViewState is helping you and when it is not. For example, if you are binding a control to data on every round trip (as in the datagridexample in tip #4), then you do not need the control to maintain its view state, since you will wipe out any re-populated data in any case. ViewState is enabled for all server controls by default. To disable it, set the EnableViewState property of the control to false.

## ****Deferred Query Execution****

To understand Deferred Query Execution, let’s take the following example which declares some Employees and then queries all employees with Age > 28:

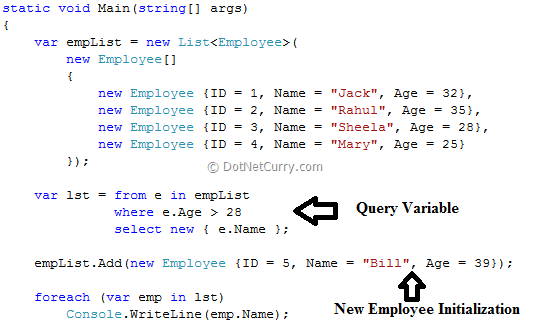




OUTPUT: Jack, Rahul

Looking at the query shown above, it appears that the query is executed at the point where the arrow is pointing towards. However that’s not true. The query is actually executed when the query variable isiterated over, not when the query variable is created. This is called deferred execution.

Now how do we prove that the query was not executed when the query variable was created? It’s simple. Just create another Employee instance ***after***the query variable is created



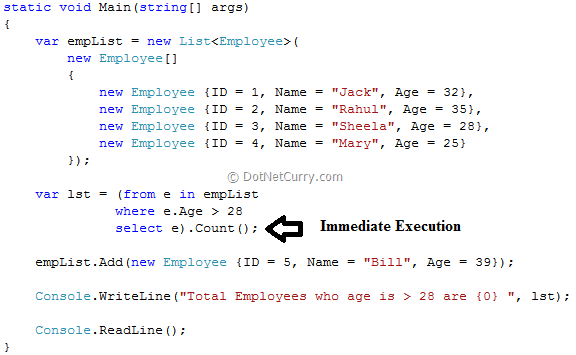
Notice we are creating a new Employee instance after the query variable is created. Now had the query been executed when the query variable is created, the results would be the same as the one we got earlier, i.e. only two employees would meet the criteria of Age > 28. However the output is not the same

OUTPUT: Jack, Rahul, Bill.

What just happened is that the execution of the query was deferred until the query variable was iterated over in a foreach loop. This allows you to execute a query as frequently as you want to, like fetching the latest information from a database that is being updated frequently by other applications. You will always get the latest information from the database in this case.

## Immediate Query Execution

You can also force a query to execute immediately, which is useful for caching query results. Let us say we want to display a count o.f the number of employees that match a criteria.



In the query shown above, it order to count the elements that match the condition, the query must be executed, and this is done automatically when Count( ) is called. So adding a new employee instanceafter the query variable declaration does not have any effect here, as the query is already executed. The output will be 2, instead of 3.

The basic difference between a Deferred execution vs Immediate execution is that Deferred execution of queries produce a sequence of values, whereas Immediate execution of queries return a singleton value and is executed immediately. Examples are using Count(), Average(), Max() etc.

**Note:** To force immediate execution of a query that does not produce a singleton value, you can call the ToList(), ToDictionary() or the ToArray() method on a query or query variable. These are called conversion operators which allow you to make a copy/snapshot of the result and access is as many times you want, without the need to re-execute the query.

## IEnumerable VS IQueryable

1. IQueryable exists in System.Linq Namespace.
2. IEnumerable exists in System.Collections Namespace.
3. IEnumerable is suitable for LINQ to Object and LINQ to XML queries.

IQueryable is suitable for LINQ to SQL queries.

1. IEnumerable is best to query data from in-memory collections like List, Array etc.
2. IQueryable is best to query data from out-memory (like remote database, service) collections.

##### JavaScript - What is the difference between undefined value and null value?

#### What is the difference between undefined value and null value?

**Undefined value:** A value that is not defined and has no keyword is known as undefined value. For example in the declaration, int number; the number has undefined value.

**Null value:** A value that is explicitly specified by the key word ‘null’ is known as null value. For example in the declaration, String str=null; the str has a null value. The keyword ‘null’ is used here.

FUNC<Input, Output> : Func can have input and output parameter.

Action<Input> : Action can have only input parameter it always return void.

Predicate<Input> : Predicate always takes input parameter and return bool value.

### What is a CDN?

Ans: A content delivery network or content distribution network (CDN) is a large distributed system of servers deployed in multiple data centers across the Internet. The goal of a CDN is to serve content to end-users with high availability and high performance.

## LEFT JOIN or LEFT OUTER JOIN in LINQ

LEFT JOIN returns all records or rows from left table and from right table returns only matched records. If there are no columns matching in the right table, it returns NULL values.

In LINQ to achieve LEFT JOIN behavior, it is mandatory to use "INTO" keyword and "DefaultIfEmpty()" method. We can apply LEFT JOIN in LINQ like as :

### C# Code

1. **var q=(from pd in dataContext.tblProducts join od in dataContext.tblOrders on pd.ProductID equals od.ProductID into t from rt in t.DefaultIfEmpty() orderby pd.ProductID select new { *//To handle null values do type casting as int?(NULL int)***
2. ***//since OrderID is defined NOT NULL in tblOrders***
3. **OrderID=(int?)rt.OrderID,**
4. **pd.ProductID,**
5. **pd.Name,**
6. **pd.UnitPrice,**
7. ***//no need to check for null since it is defined NULL in database***
8. **rt.Quantity,**
9. **rt.Price,**
10. **}).ToList();**

## INNER JOIN On Multiple Conditions in LINQ

Sometimes, we required to apply join on multiple coditions. In this case, we need to make two anonymous types (one for left table and one for right table) by using new keyword then we compare both the anonymous types.

### C# Code

1. **var q=(from pd in dataContext.tblProducts join od in dataContext.tblOrders on pd.ProductID equals od.ProductID join ct in dataContext.tblCustomers on new {a=od.CustomerID,b=od.ContactNo} equals new {a=ct.CustID,b=ct.ContactNo} orderby od.OrderID select new { od.OrderID,**
2. **pd.ProductID,**
3. **pd.Name,**
4. **pd.UnitPrice,**
5. **od.Quantity,**
6. **od.Price,**
7. **Customer=ct.Name *//define anonymous type Customer***
8. **}).ToList();**

IEnumerable VS Var

Use Var type when you want to make a "custom" type on the fly.

Use IEnumerable when you already know the type of query result.

IEnumerable is a forward only collection and is useful when we already know the type of query result. In below query the result will be a list of employee that can be mapped (type cast) to employee table.

1. **IEnumerable<tblEmployee> lst =(from e in tblEmployee**
2. **where e.City=="Delhi"**
3. **select e);**

## Var Type

Since [Var](http://www.dotnet-tricks.com/Tutorial/csharp/RJK3170612-C-Sharp-Var-data-type-and-Anonymous-Type.html) is anonymous types, hence use it whenever you don't know the type of output or it is anonymous. Suppose you are joining two tables and retrieving data from both the tables then the result will be an Anonymous type.

1. **var q =(from e in tblEmployee**
2. **join d in tblDept on e.DeptID equals d.DeptID**
3. **select new**
4. **{**
5. **e.EmpID,**
6. **e.FirstName,**
7. **d.DeptName,**
8. **});**

In above query, result is coming from both the tables so use Var type.

1. **var q =(from e in tblEmployee where e.City=="Delhi" select new {**
2. **e.EmpID,**
3. **FullName=e.FirstName+" "+e.LastName,**
4. **e.Salary**
5. **});**

In above query, result is coming only from single table but we are combining the employee's FirstName and LastName to new type FullName that is annonymous type so use Var type. Hence use Var type when you want to make a "custom" type on the fly.

## IEnumerable

1. IEnumerable exists in System.Collections Namespace.
2. IEnumerable can move forward only over a collection, it can’t move backward and between the items.
3. IEnumerable is best to query data from in-memory collections like List, Array etc.
4. While query data from database, IEnumerable execute select query on server side, load data in-memory on client side and then filter data.
5. IEnumerable is suitable for LINQ to Object and LINQ to XML queries.
6. IEnumerable supports deferred execution.
7. IEnumerable doesn’t supports custom query.
8. IEnumerable doesn’t support lazy loading. Hence not suitable for paging like scenarios.
9. Extension methods supports by IEnumerable takes functional objects.

## IQueryable

1. IQueryable exists in System.Linq Namespace.
2. IQueryable can move forward only over a collection, it can’t move backward and between the items.
3. IQueryable is best to query data from out-memory (like remote database, service) collections.
4. While query data from database, IQueryable execute select query on server side with all filters.
5. IQueryable is suitable for LINQ to SQL queries.
6. IQueryable supports deferred execution.
7. IQueryable supports custom query using CreateQuery and Execute methods.
8. IQueryable support lazy loading. Hence it is suitable for paging like scenarios.
9. Extension methods supports by IEnumerable takes expression objects means expression tree.

## IList

1. IList exists in System.Collections Namespace.
2. IList is used to access an element in a specific position/index in a list.
3. Like IEnumerable, IList is also best to query data from in-memory collections like List, Array etc.
4. IList is useful when you want to Add or remove items from the list.
5. IList can find out the no of elements in the collection without iterating the collection.
6. IList supports deferred execution.
7. IList doesn't support further filtering.

## IEnumerable

1. IEnumerable exists in System.Collections Namespace.
2. IEnumerable can move forward only over a collection, it can’t move backward and between the items.
3. IEnumerable is best to query data from in-memory collections like List, Array etc.
4. IEnumerable doesn't support add or remove items from the list.
5. Using IEnumerable we can find out the no of elements in the collection after iterating the collection.
6. IEnumerable supports deferred execution.
7. IEnumerable supports further filtering.

## Introduction

This article aims at understanding the role of HTTPHandler and HTTPModule in ASP.NET applications. We will try to work on a basic example to see how these can be implemented.

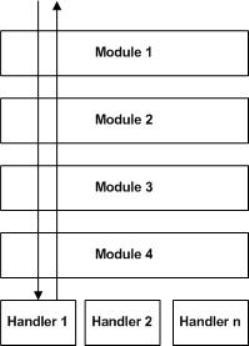
## Background

ASP.NET handles all the HTTP requests coming from the user and generates the appropriate response for it. ASP.NET framework knows how to process different kind of requests based on extension, for example, It can handle request for.aspx, .ascx and .txt files, etc. When it receives any request, it checks the extension to see if it can handle that request and performs some predefined steps to serve that request.

Now as a developer, we might want to have some of our own functionality plugged in. We might want to handle some new kind of requests or perhaps we want to handle an existing request ourselves to have more control on the generated response, for example, we may want to decide how the request for .jpg or .gif files will be handled. Here, we will need an HTTPHandler to have our functionality in place.

There are also some scenarios where we are ok with the way ASP.NET is handling the requests but we want to perform some additional tasks on each request, i.e., we want to have our tasks execute along with the predefined steps ASP.NET is taking on each request. If we want to do this, we can have HTTPModule in place to achieve that.

So from the above discussion, it is clear that HTTPHandlers are used by ASP.NET to handle the specific requests based on extensions. HTTPModule, on the other hand, is used if we want to have our own functionality working along with the default ASP.NET functionality. There is one Handler for a specific request but there could be N number of modules for that.



## Using the Code

Let us try to understand these two concepts by writing a small application for each. What we will do is we will try to have a mechanism where we can process the web pages with extension like .bspx and .cspx. Although this is a very unrealistic scenario, a similar concept is used to have search engine friendly URLs so perhaps it's not that realistic either.

**Note**: The HTTPHandler example here is just for demonstration purpose, I am not recommending the use ofHTTPHandlers for something that I am about to do now. HTTPHandlers should ideally be used to customize the handling of existing MIME types and not for serving search engine friendly URLs or non standard URLs.

## Implementing the HTTPHandler

So with our problem definition, let us try to see how we can handle the request for .cspx pages using HTTPHandlers. First we need to have the handler class with us, so let us create the handler class.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/335968/Implementing-HTTPHandler-and-HTTPModule-in-ASP-NET)

public class CspxHandler :IHttpHandler

{

public bool IsReusable

{

get { return false; }

}

public void ProcessRequest(HttpContext context)

{

}

}

The class should have a method ProcessRequest and a property called IsReusable. The property tells whether this handler can be reused or not and the method will be called whenever a request for that type comes. But wait, Where have we defined the type of request where this handler should be invoked? This can be defined either in IIS, if we have a handler common to all the sites running on that server or we can configure it in web.config file, if the handler is specific for a website. Let's do that in web.config file for now.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/335968/Implementing-HTTPHandler-and-HTTPModule-in-ASP-NET)

<httpHandlers>

<add verb="\*" path="\*.cspx" type="CspxHandler"/>

</httpHandlers>

Here we registered our handler to specify that if any request for .cspx file comes, it should be forwarded to our handler.

Now, since we don't have any "real" files with .cspx extension, what we will do is we will handle the request for .cspxand in turn push the user to the corresponding .aspx file.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/335968/Implementing-HTTPHandler-and-HTTPModule-in-ASP-NET)

public class CspxHandler :IHttpHandler

{

public bool IsReusable

{

get { return false; }

}

public void ProcessRequest(HttpContext context)

{

context.Response.ContentType = "text/plain";

if (context.Request.RawUrl.Contains(".cspx"))

{

string newUrl = context.Request.RawUrl.Replace(".cspx", ".aspx");

context.Server.Transfer(newUrl);

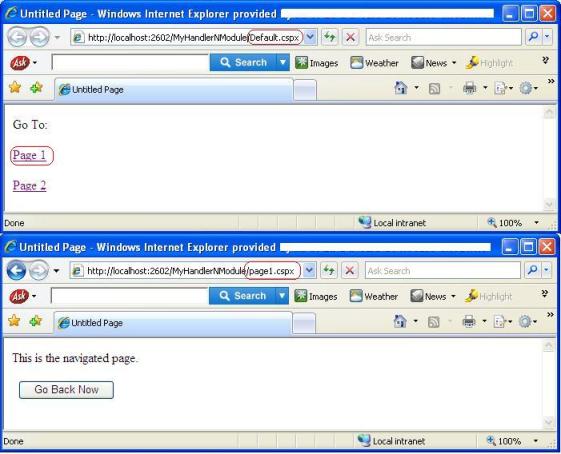
}

}

}

Whenever a request for .cspx file comes, we will handle it in our handler and show the corresponding .aspx file instead. Let's see how it works.

**Note**: I have also changed the startup page name to Default.cspx but there is no page like that. I want my handler to handle that and show me the actual default page.



**Important:** I reiterate, This example is just for illustration. This is not how HTTPHandlers should be used at all.HTTPHandlers should ideally be used to customize the handling of existing MIME types.

Well the pages seems to be working fine and the user will see .cspx URL for his request. But there is one problem. The way we wrote our handler is not good to handle the postback. If I add a button on any of these pages and do a postback, the original URLs will be visible. So it is not a good solution to the problem but it sure demonstrated the way Handlers can be used.

## Implementing the HTTPModule

How do we solve the problem we just saw. Well, our application needed URL rewriting and HTTPHandlers are a bad solution for that and should never be used for that. So perhaps the guys using this technique to have search friendly URLs should rethink their strategy. SO how can we solve this problem really.

Let us look at the requirement again, All we needed was to show the user URLs which are different than the real URLs and process the real URLs internally. So we don't need custom handlers, we are ok with the way ASP.NET engine is handling these requests but we need custom activities to be done during the processing phase. So it looks like we can solve it using HTTPModule.

So let's go ahead and write an HttpModule that will:

1. Check for file extension on request.
2. If it finds a .bspx extension it changes it to .aspx (or find real URLS if we are implementing search friendly URLs)
3. It will pass the request to the default handler, since the page is still aspx.
4. Once the response is generated, it will write back the original .bspx URL to users browser.

http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/335968/Implementing-HTTPHandler-and-HTTPModule-in-ASP-NET)

public class MyBModule : IHttpModule

{

public void Dispose()

{

}

public void Init(HttpApplication context)

{

context.BeginRequest += new EventHandler(context\_BeginRequest);

context.PreRequestHandlerExecute += new EventHandler(context\_PreRequestHandlerExecute);

context.EndRequest += new EventHandler(context\_EndRequest);

context.AuthorizeRequest += new EventHandler(context\_AuthorizeRequest);

}

void context\_AuthorizeRequest(object sender, EventArgs e)

{

*//We change uri for invoking correct handler*

HttpContext context = ((HttpApplication)sender).Context;

if (context.Request.RawUrl.Contains(".bspx"))

{

string url = context.Request.RawUrl.Replace(".bspx", ".aspx");

context.RewritePath(url);

}

}

void context\_PreRequestHandlerExecute(object sender, EventArgs e)

{

*//We set back the original url on browser*

HttpContext context = ((HttpApplication)sender).Context;

if (context.Items["originalUrl"] != null)

{

context.RewritePath((string)context.Items["originalUrl"]);

}

}

void context\_EndRequest(object sender, EventArgs e)

{

*//We processed the request*

}

void context\_BeginRequest(object sender, EventArgs e)

{

*//We received a request, so we save the original URL here*

HttpContext context = ((HttpApplication)sender).Context;

if (context.Request.RawUrl.Contains(".bspx"))

{

context.Items["originalUrl"] = context.Request.RawUrl;

}

}

}

Also we need to register our module so that it can be invoked, we will do that in our web.config file.

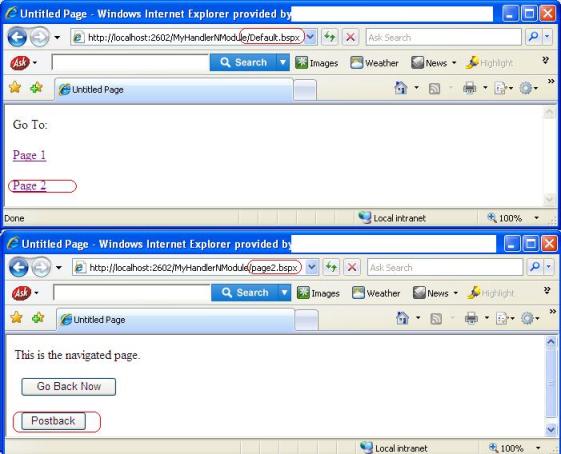
http://www.codeproject.com/images/minus.gif Collapse | [Copy Code](http://www.codeproject.com/Articles/335968/Implementing-HTTPHandler-and-HTTPModule-in-ASP-NET)

<httpModules>

<add name="MyBModule" type="MyBModule" />

</httpModules>

And now let's run the application:



So here we solved the problem of URL reverting back to the original on postback. This is also the ideal way of doing that.

## Points of Interest

In this article, we saw how we can implement a basic HTTPHandler and HTTPModule. We saw each of their roles in page processing frameworks. We worked on an example that tried to solve the URL rewriting first the wrong way by using HTTPHandler (but we understood how to write HTTPhandler) and then the right way of doing URL rewriting using HTTPModule (we got to understand that too.

The emphasis of this article was solely on understanding how we can have HTTPHandlers and HTTPModulesworking. The example is a little unrealistic and perhaps a little misleading too but since I made that point really clear, it shouldn't be a problem.

Before wrapping up, there is one last thing that we should know about handlers. It is also possible to handle the request asynchronously. ASP.NET provides a mechanism for creating asynchronous handler and then increases the performance of a web page (implements the IHttpAsyncHandler do that).

**(B) What is a delegate?**

Delegate is a class that can hold a reference to a method or a function. Delegate class has a

Signature and it can only reference those methods whose signature is compliant with the class.

Delegates are type-safe functions pointers or callbacks.

Below is a sample code, which shows a example of how to implement delegates.

Public Class FrmDelegates

Inherits System.Windows.Forms.Form

Public Delegate Sub DelegateAddString ()

Private Sub FrmDelegates\_Load (ByVal sender As System.Object, ByVal e

As System.EventArgs) Handles MyBase.Load

End Sub

Private Sub Add String ()

lstDelegates.Items.Add (“Running Add String () method”)

End Sub

Private Sub cmdDelegates\_Click (ByVal sender As System. Object, By Val

e As System.EventArgs) Handles cmdDelegates. Click

Dim objDelegateAddString As DelegateAddString

ObjDelegateAddString = AddressOf Add String

objDelegateAddString.Invoke ()

End Sub

End Class

In the above, there is a method called “Add String ()” which adds a string to a listbox.You can

also see a delegate declared as:-

Public Delegate Sub DelegateAddString()

This delegate signature is compatible with the “Add String” method. When we mean

compatibility that means that there, return types and passing parameter types are same. Later in

command click of the button object of the Delegate is created and the method pointer is received

from “AddressOf” keyword. Then by using the “Invoke” method, the method is invoked.

**What is GAC? What are the steps to create an assembly and add it to the GAC?**

The global assembly cache (GAC) is a machine-wide code cache that stores assemblies specifically designated to be shared by several applications on the computer. You should share assemblies by installing them into the global assembly cache only when you need to.

Steps

- Create a strong name using sn.exe tool eg: sn -k mykey.snk

- in AssemblyInfo.cs, add the strong name eg: [assembly: AssemblyKeyFile("mykey.snk")]

- recompile project, and then install it to GAC in two ways :

         drag & drop it to assembly folder (C:\WINDOWS\assembly OR C:\WINNT\assembly) (shfusion.dll tool)

         gacutil -i abc.dll

**What is a multicast delegate?**

Each delegate object holds reference to a single method. However, it is possible for a delegate object to hold references of and invoke multiple methods. Such delegate objects are called multicast delegates or combinable delegates.

1. Tell about your technical profile.  
2. What was your role in your project?  
3. In which module you have worked explain?  
4. Describe GridView events (Paper).  
5. How to change Label's color based on Label's value  (Paper)?  
6. Write the code to perform edit and delete operations using  GridView (Paper).  
7. What are the Validation controls in ASP.Net?  
8. How you will implement Custom Validator Control functionality (Paper)?  
9. What are Generics?  
10. What is the  difference between HashTable and Dictionary?  
11. What is Ajax and  Jquery?  
12. Which control have you used in AJAX?  
13. What is the use of ModalPopUpExtender (Paper)?  
14. WCF Basics (Types of binding)  
15. What are Database Constraints?  
16. Difference between Primary and Unique key?  
17. Difference between Function and Procedure?  
18. Can we store DataSet in View State?  
19. When we will store DataSet in Session then which memory will be filled client side or server side?  
20. Difference between reference and out parameter?  
21. How to execute stored procedure?  
  
-----------------------------------------------------------------------------------------------------------------------  
-----------------------------------------------------------------------------------------------------------------------  
  
1. What is View State?  
2. Where is View State is saved and how View State value is retained  
between Post Back. (Practical)?  
3. Form Authentication Process (Using Web.Config file and Database in paper).  
4. If View State value is "X" and I have changed it to "Y" in Page\_Load  
then what will be the final value of View State?  
5. Page Life cycle with use.  
6. Performance Analyzer tool.  
7. How to declare unique key?  
8. Diff. between Equi join and Right outer join (Paper)?  
9. Define Caching types.  
10. How to implement SQL Cache (Paper)?  
11. How to call Web service using AJAX (Paper)?  
12. How to change Color of Label using Jquery (Paper)?  
13. What is Table Object/Variable?  
14. How to call stored procedure using Entity Framework?  
15. What is the difference between Overloading and Overriding?  
16. What is the difference between ExecuteScalar() and ExecuteNonQuery()?  
  
17. What if I will pass Select \* in ExecuteScalar()?  
  
---------------------------------------------------------------------------------------------------------------------  
---------------------------------------------------------------------------------------------------------------------  
  
1. Tell me about projects.  
2. Your role in project.  
3. Which performance tool you have used?  
4. Define abstract class and interface?  
5. Why to use static members?  
6. What is partial class and advantages?  
7. GridView and DataList differences.  
8. State Management type.  
9. What is view state and use?  
10. Caching techniques.  
11. WCF and Web service differences.  
12. Define WCF contracts.  
13. Define design pattern.  
14. What is facade pattern?  
15. Triggers use and types.  
16. Define cursor.  
17. Difference between clustered and non-clustered index.  
18. How many clustered index can be declared for a table.  
19. What is view?  
20. What is AJAX and Update-panel?  
21. If you have 2 Update-panel then how to update one update panel from second?  
  
---------------------------------------------------------------------------------------------------------------------  
---------------------------------------------------------------------------------------------------------------------  
  
1. Define your Technical skills.  
2. Your Role in project.  
3. Define features of OOPS.  
4. How you will replace the functionality of parent class function in  
child class.(Paper)  
5. Difference between Interface and abstract class (Use).  
6. Functions of CLR.  
7. Where you will use Cookie, Session and View State.  
8. What are InProc and OutProc Session?  
9. If you will save session in OutProc then where it will be saved.  
and when it will be expired.  
10. Difference between Web service and WCF  
11. Design pattern.  
12. Write sequence of select query-  
  
 [ WITH ]  
SELECT select\_list [ INTO new\_table ]  
 [ FROM table\_source ]  
 [ WHERE search\_condition ]  
[ GROUP BY group\_by\_expression ]  
[ HAVING search\_condition ]  
[ ORDER BY order\_expression [ ASC | DESC ] ]  
  
The UNION, EXCEPT and INTERSECT operators can be used between queries  
to combine or compare their results into one result set.  
  
13. What is ROW\_NUMBER () in SQL Server?  
14.  Difference between Union and Join.  
15. Return value for ExecuteNonQuery().  
16. Can we execute DML command in ExecuteScalar()?  
17. Difference between DataSet and DataReader.  
18. If provider is given OracleClient then can we connect with SQL Server?  
19. Where you have used JQuery in your project?  
20. Namespace for web-part (SharePoint)    - System.Web.UI.WebControls.WebParts Namespace  
21. For creating a site what are the main concepts you need to consider in  
UL layer, middle-ware and Database Layer