1. program for sorting a int array

string[] array = new string[] { "c", "b", "d", "a" };

Array.Sort(array);

Array.Reverse(array);

foreach (var str in array)

{

MessageBox.Show(str.ToString());

}

**Or**

string[] array = new string[] { "a", "b", "c", "d" };

array = array.OrderByDescending(c => c).ToArray();

foreach (var str in array)

{

MessageBox.Show(str.ToString());

}

**Or**

for(i=0; i<n; i++)

{

for(j=i+1; j<n; j++)

{

if(arr1[i] < arr1[j])

{

tmp = arr1[i];

arr1[i] = arr1[j];

arr1[j] = tmp;

}

}

}

1. Inherit Multiple Interfaces with the Same Method Name

using System;

interface A

{

    void Hello();

}

interface B

{

    void Hello();

}

class Test : A, B

{

    void A.Hello()

    {

        Console.WriteLine("Hello to all-A");

    }

    void B.Hello()

    {

        Console.WriteLine("Hello to all-B");

    }

}

public class interfacetest

{

    public static void Main()

    {

        A Obj1 = new Test();

        Obj1.Hello();

        B Obj2 = new Test();

        Obj2.Hello();

    }

}

Interface-3.jpg

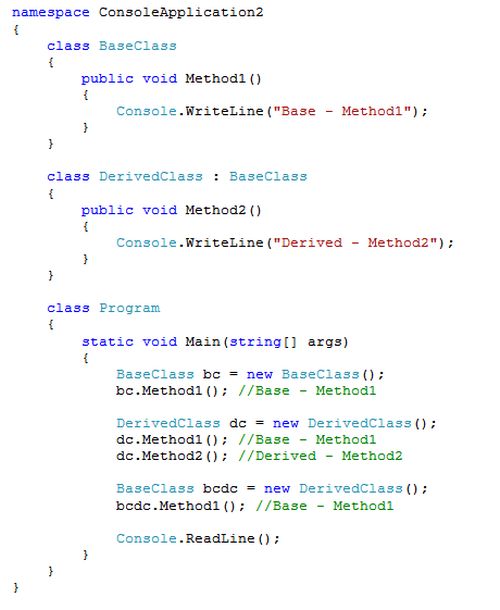
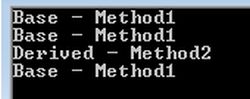
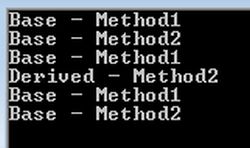
1. how to hide a method
2. The "override" modifier extends the base class method, and the "new" modifier hides it.
3. The "virtual" keyword modifies a method, property, indexer, or event declared in the base class and allows it to be overridden in the derived class.
4. The "override" keyword extends or modifies a virtual/abstract method, property, indexer, or event of the base class into the derived class.
5. The "new" keyword is used to hide a method, property, indexer, or event of the base class into the derived class.
6. If a method is not overriding the derived method then it is hiding it. A hiding method must be declared using the new keyword.
7. Shadowing is another commonly used term for hiding. The C# specification only uses "hiding" but either is acceptable. Shadowing is a VB concept.

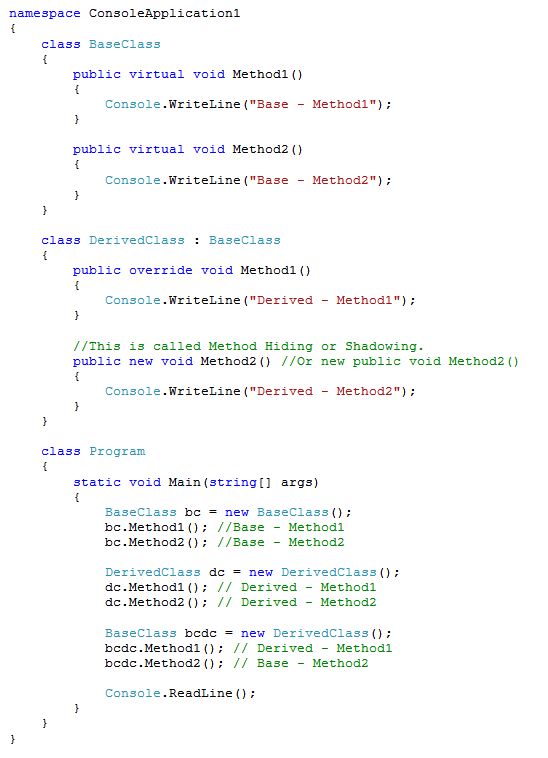
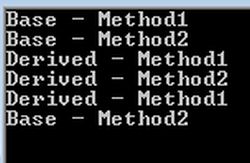
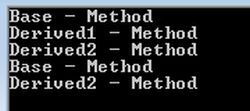
**What are the differences between method hiding and overriding in C#?**

1. For hiding the base class method from derived class simply declare the derived class method with the new keyword.  
   Whereas in C#, for overriding the base class method in a derived class, you need to declare the base class method as virtual and the derived class method as overriden.
2. If a method is simply hidden then the implementation to call is based on the compile-time type of the argument "this".  
   Whereas if a method is overridden then the implementation to be called is based on the run-time type of the argument "this".
3. New is reference-type specific, overriding is object-type specific.

**What are the differences between method hiding and method shadowing?**

1. Shadowing is a VB concept. In C#, this concept is called hiding.
2. The two terms mean the same in C#.  
   Method hiding == shadowing
3. In short, name "hiding" in C# (new modifier) is called shadowing in VB.NET (keyword Shadows).
4. In C# parlance, when you say "hiding" you're usually talking about inheritance, where a more derived method "hides" a base-class method from the normal inherited method call chain.
5. When you say "shadow" you're usually talking about scope; an identifier in an inner scope is "shadowing" an identifier at a higher scope.
6. In other languages, what is called "hiding" in C# is sometimes called "shadowing" as well.

**Examples 1: Simple one**  
  
  
  
**Output:  
  
  
  
Example 2: Method hiding with warning  
  
  
  
Output:  
  
**

**Example 3: Overriding, Hiding and Shadowing  
  
  
  
  
Output:  
  
  
  
Example 4 : Method Overriding and Method Hiding:   
  
  
  
Output:**

**Note: method is called based on the left hand side type. Exception is if left hand side type has virtual method, then the right hand side type method would be called (in case left and right hand side types are different). If derived class method has New, then also left hand side method is called (in case left and right hand side types are different).**

1. we have changed the controller names but still business has to use same url of a MVC with old controller and action names

The ASP.NET Routing module is responsible for mapping incoming browser requests to particular MVC controller actions.

**Global.asax.cs**

[RoutePrefix("example-name")]

public class example\_nameController : Controller

{

// Route: example-name/Index

[ActionName="newactionname"]

public ActionResult Index()

{

return View();

}

**Or**

public class RouteConfig

{

public static void RegisterRoutes(RouteCollection routes)

{

routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

routes.MapRoute(

name: "MyCustomRoute", // Route name

url: "MyEmployees/{action}/{id}", // URL with parameters

defaults: new

{

controller = "PersonalDetail",

action = "Create",

id = UrlParameter.Optional

} // Parameter defaults

);

routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{id}",

defaults: new

{

controller = "Home",

action = "Index",

id = UrlParameter.Optional

}

);

protected void Application\_Start()

{ RegisterRoutes(RouteTable.Routes); }



1. how to call similar methods of two differnt interfaces in a derived class

using System;

interface A

{

void Hello();

}

interface B

{

void Hello();

}

class Test : A, B

{

void **A.Hello**()

{

Console.WriteLine("Hello to all-A");

}

void **B.Hello**()

{

Console.WriteLine("Hello to all-B");

}

}

public class interfacetest

{

public static void Main()

{

**A Obj1 = new Test**();

Obj1.Hello();

**B Obj2 = new Test()**;

Obj2.Hello();

}

}

1. on a button click change the color of other buttons with class c1.

<p><input type="submit" value=" Once or Twice " name="btnsubmit" id="answer" style="width: 200px;" /></p>

$('input[type="submit"]').click(function(){

if($('input[type="submit"]').hasClass('red'))

$('input[type="submit"]').removeClass('red').**addClass**('green');

else if($('input[type="submit"]').hasClass('green'))

$('input[type="submit"]').removeClass('green').addClass('red');

else

$('input[type="submit"]').addClass('red');

});

.red

{

background-color:red;

}

.green

{

background-color:green;

}

JQuery class selector:

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script>

$(document).ready(function () {

$("**p.intro**").css("background-color", "yellow");

});

</script>

</head>

<body>

<h1>Welcome to My Homepage</h1>

<**p** **class="intro"**>My name is Donald.</p>

<p>I live in Duckburg.</p>

<p class="intro">My name is Dolly.</p>

<p>I live in Duckburg.</p>

</body>

</html>

**Few JQUERY Selectors**

Id selector: $("#test")

Class selector: $(".test")

$("ul li:first") Selects the first <li> element of the first <ul>

$("[href]") Selects all elements with an href attribute

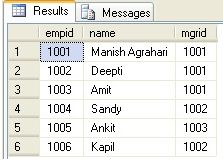
$(":button") Selects all <button> elements and <input> elements of type="button"

$("p") element selector

1. Difference between function and procedure in sql server
2. Function must return a value but in Stored Procedure it is optional (Procedure can return zero or n values).
3. Functions can have only input parameters for it whereas Procedures can have input/output parameters.
4. Functions can be called from Procedure whereas Procedures cannot be called from Function.
5. Procedure allows SELECT as well as DML (INSERT/UPDATE/DELETE) statement in it whereas Function allows only SELECT statement in it.
6. Procedures cannot be utilized in a SELECT statement whereas Function can be embedded in a SELECT statement.
7. Stored Procedures cannot be used in the SQL statements anywhere in the WHERE/HAVING/SELECT section whereas Function can be.
8. Functions that return tables can be treated as another row set. This can be used in JOINs with other tables.
9. Inline Function can be thought of as views that take parameters and can be used in JOINs and other Rowset operations.
10. Exception can be handled by try-catch block in a Procedure whereas try-catch block cannot be used in a Function.
11. We can go for Transaction Management in Procedure whereas we can't go in Function.
12. Joins:

Self Join- used in tree structure data, like phweb

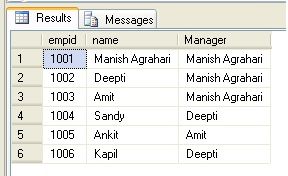
It joins each row of a table to all other rows of same table, based on the where condition. So its like a Cartesian join with its own replica.



SELECT e.empid, e.name, m.name  "Manager"

FROM Emp e, Emp m

WHERE e.mgrid=m.empid;



Here are the different types of the JOINs in SQL:

* **(INNER) JOIN**: Returns records that have matching values in both tables
* **LEFT (OUTER) JOIN**: Return all records from the left table, and the matched records from the right table
* **RIGHT (OUTER) JOIN**: Return all records from the right table, and the matched records from the left table
* **FULL (OUTER) JOIN**: Return all records when there is a match in either left or right table

Nth Highest salary:

Correlated subquery

SELECT name, salary FROM JoinsTest e1

WHERE N-1 = (SELECT COUNT(DISTINCT salary) FROM JoinsTest e2 WHERE e2.salary > e1.salary)

Using TOP keyword

SELECT TOP 1 salary FROM ( SELECT DISTINCT TOP N salary FROM #Employee ORDER BY salary DESC ) AS temp ORDER BY salary

Using Common Table Expression

With CTE as

(

SELECT DISTINCT TOP 2 salary FROM JoinsTest ORDER BY salary DESC

)

select top 1 salary from CTE ORDER BY salary ASC

OR

(USING DENSE\_RANK() METHOD)

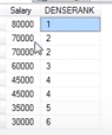
With CTE as

(

SELECT salary, **DENSE\_RANK() OVER (ORDER BY SALARY DESC)** AS DENSERANK FROM JoinsTest

)

select top 1 salary from CTE WHERE CTE.DENSERANK=N



ONLY 2ND Highest salary:

SELECT MAX(SALARY) FROM EMPLOYEES

WHERE SALARY < (SELECT MAX(SALARY) FROM EMPLOYEES)

1. Delegates

Delegate objects can be composed using the "+" operator. A composed delegate calls the two delegates it was composed from. Only delegates of the same type can be composed. The "-" operator can be used to remove a component delegate from a composed delegate.

Using this property of delegates you can create an invocation list of methods that will be called when a delegate is invoked. This is called **multicasting** of a delegate

using System;

**delegate int NumberChanger(int n);**

namespace DelegateAppl {

class TestDelegate {

static int num = 10;

public static int AddNum(int p) {

num += p;

return num;

}

public static int MultNum(int q) {

num \*= q;

return num;

}

public static int getNum() {

return num;

}

static void Main(string[] args) {

**//create delegate instances**

NumberChanger nc;

NumberChanger nc1 = new NumberChanger(AddNum);

NumberChanger nc2 = new NumberChanger(MultNum);

**nc = nc1;**

**nc += nc2;**

//calling multicast

**nc(5); //nc1=10+5=15; nc2=15\*5=75**

Console.WriteLine("Value of Num: {0}", getNum());

Console.ReadKey();

}

}

}

Click event handler is also an example:

**innerButton.Click += new EventHandler(innerButton\_Click);**

1. differnt types of generations in garbage collection

**0 Generation (Zero)**: This generation holds short-lived objects, e.g., Temporary objects. GC initiatesgarbage collection process frequently in this generation.

**1 Generation (One)**: This generation is the buffer between short-lived and long-lived objects.

**2 Generation (Two)**: This generation holds long-lived objects like a static and global variable, that needs to be persisted for a certain amount of time. Objects which are not collected in generation Zero, are then moved to generation 1, such objects are known as *survivors*, similarly objects which are not collected in generation One, are then moved to generation 2 and from there onwards objects remain in the same generation.

Garbage Collection Phases:

**Marking Phase** - In this phase garbage collector finds and creates a list of all live objects.

**Relocating Phase** - In this phase garbage collector updates the references to the objects that will be compacted.

|  |  |
| --- | --- |
| **Dispose** | **Finalize** |
| It is used to free unmanaged resources like files, database connections etc. at any time. |  |
| Explicitly, it is called by user code and the class which is implementing dispose method, must has to implement IDisposable interface. | Internally, it is called by Garbage Collector and cannot be called by user code. |
| It belongs to IDisposable interface. | It belongs to Object class |
| It's implemented by implementing IDisposable interface Dispose() method. | It's implemented with the help of destructor in C++ & C#. |

**Compacting Phase** - In this phase garbage collector reclaims the memory occupied by the dead objects and compacts the surviving objects. The compacting phase moves the surviving objects toward the older end of the memory segment.

GC.Collect() asks the system to perform a collection "now". You shouldn't mess with this; the system usually has a much better idea than you do of when collection is necessary.

1. *// Using Dispose and Finalize method together*
2. public class MyClass : IDisposable
3. {
4. private bool disposed = false;
5. *//Implement IDisposable.*
6. public void Dispose()
7. {
8. Dispose(true);
9. GC.SuppressFinalize(this);
10. }
12. protected virtual void Dispose(bool disposing)
13. {
14. if (!disposed)
15. {
16. if (disposing)
17. {
18. *// TO DO: clean up managed objects*
19. }
20. *// TO DO: clean up unmanaged objects*
21. disposed = true;
22. }
23. }
24. *//At runtime C# destructor is automatically Converted to Finalize method*
25. ~MyClass()
26. {
27. Dispose(false);
28. }
29. }
30. Changes in different versions of C#

**Version 2.0**

Nullable Types:

int? num = null; // or Nullable<int> i = null;

// Is the HasValue property true?

if (num.HasValue)

{

Console.WriteLine("num = " + num.Value);

}

else

{

Console.WriteLine("num = Null");

}

Generics:

public class GenericList<T>

{

public void Add(T input) { }

}

class TestGenericList

{

private class ExampleClass { }

static void Main()

{

// Declare a list of type int.

GenericList<int> list1 = new GenericList<int>();

list1.Add(1);

Anynomous Methods:

Declared using **delegate** keyword.

Print print = delegate(int val) {

Console.WriteLine("Inside Anonymous method. Value: {0}", val);

};

print(100);

**Version 3.0**

Anonymous Types

var v = new { Amount = 108, Message = "Hello" };

Console.WriteLine(v.Amount + v.Message);

Extension methods

An extension method enables us to add methods to existing types without creating a new derived type, recompiling, or modify the original types. We can say that it extends the functionality of an existing type in .NET. An extension method is a static method to the existing staticclass.

It is a static method.

It must be located in a static class.

It uses the "this" keyword as the first parameter with a type in .NET and this method will be called by a given type instance on the client side.

using System;

namespace ExtensionMethodsExample

{

public static class Extension

{

public static int WordCount(this string str)

{

string[] userString = str.Split(new char[] { ' ', '.', '?' },

StringSplitOptions.RemoveEmptyEntries);

int wordCount = userString.Length;

return wordCount;

}

}

}

Call it as:

using System;

namespace ExtensionMethodsExample

{

class Program

{

static void Main(string[] args)

{

string userSentance = string.Empty;

int totalWords = 0;

int totalCharWithoutSpace = 0;

Console.WriteLine("Enter the your sentance");

userSentance = Console.ReadLine();

*//calling Extension Method WordCount*

totalWords = userSentance.WordCount();

Console.WriteLine("Total number of words is :"+ totalWords);

**Version 4.0**

Named and Optional Arguments

class NamedExample

{

static void Main(string[] args)

{

// The method can be called in the normal way, by using positional arguments.

PrintOrderDetails("Gift Shop", 31, "Red Mug");

// Named arguments can be supplied for the parameters in any order.

PrintOrderDetails(orderNum: 31, productName: "Red Mug", sellerName: "Gift Shop");

// Named arguments mixed with positional arguments are valid

// as long as they are used in their correct position.

PrintOrderDetails("Gift Shop", 31, productName: "Red Mug");

// However, mixed arguments are invalid if used out-of-order.

}

static void PrintOrderDetails(string sellerName, int orderNum, string productName)

{

// Do something

}

}

Dynamic Types

The dynamic type enables the operations in which it occurs to bypass compile-time type checking. Instead, these operations are resolved at run time.

class ExampleClass

{

// A dynamic field.

static dynamic field;

// A dynamic property.

dynamic prop { get; set; }

// A dynamic return type and a dynamic parameter type.

public dynamic exampleMethod(dynamic d)

{

// A dynamic local variable.

dynamic local = "Local variable";

int two = 2;

if (d is int)

{

return local;

}

else

{

return two;

}

}

}

1. differnce between joins and unions

In simple terms, **joins combine data into new columns**.  If two tables are joined together, then the data from the first table is shown in one set of column alongside the second table’s column in the same row.

**Unions combine data into new rows**.  If two tables are “unioned” together, then the data from the first table is in one set of rows, and the data from the second table in another set.  The rows are in the same result.

SELECT C.Name

FROM Production.ProductCategory AS C

UNION ALL

SELECT S.Name

FROM Production.ProductSubcategory AS S

In order to union two table there are a couple of requirements:

1. The number of columns must be the same for both select statements.
2. The columns, in order, must be of the same data type.
3. When rows are combined duplicate rows are eliminated.  If you want to keep all rows from both select statement’s results use the **ALL** keyword
4. MVC FILters

MVC Filters are used to execute custom logic before or after executing action method.

MVC Filter types:

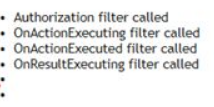
| Filter Type | **Description** | **Built-in Filter** | **Interface** |
| --- | --- | --- | --- |
| **Authorization filters** | Performs authentication and authorizes before executing action method. | [**Authorize**], [**RequireHttps**] | IAuthorizationFilter |
| **Action filters** | Performs some operation before and after an action method executes. |  | IActionFilter |
| **Result filters** | Performs some operation before or after the execution of view result. | [**OutputCache**] | IResultFilter |
| **Exception filters** | Performs some operation if there is an unhandled exception thrown during the execution of the ASP.NET MVC pipeline. | [**HandleError**] | IExceptionFilter |

Filters can be applied globally in FilterConfig class, at controller level or action method level.

Custom filter class can be created by implementing FilterAttribute class and corresponding interface.

**Order of filter execution:**





Last two dots are “OnResultExecuted called” and “OnException”. This two are not shown as when resultexecuted, the view is already rendered.

**Action Filter**

Action filter executes before and after an action method executes. Action filter attributes can be applied to an individual action method or to a controller or at global level.

[OutputCache(Duration=100)]

public ActionResult Index()

{

return View();

}

**Custom Action Filter can also be created:**

New custom class must inherit from ActionFilterAttribute

Every custom action filter must override OnActionExecuted, OnActionExecuting, OnResultExecuted, OnResultExecuting methods.

OnActionExecuting - Runs before execution of Action method.

OnActionExecuted - Runs after execution of Action method.

OnResultExecuting - Runs before content is rendered to View.

OnResultExecuted - Runs after content is rendered to view.

public class LogAttribute : ActionFilterAttribute

{

public override void OnActionExecuted(ActionExecutedContext filterContext)

{

Log("OnActionExecuted", filterContext.RouteData);

}

public override void OnActionExecuting(ActionExecutingContext filterContext)

{

Log("OnActionExecuting", filterContext.RouteData);

}

….

Now Apply that filter

**[Log]**

public class StudentController : Controller

{

public ActionResult Index()

{

return View();

}

**Authorization FIlter**

// Restrict by role:

[Authorize(Roles="Administrators")]

public class ValuesController : ApiController

{

}

1. Role based authentication using Authorization filter
2. Add this to the controller action method

[Authorize(Roles = "Admin,Editor")]

public ActionResult About()

{

1. This would redirect to

<http://localhost:3428/Account/Login?ReturnUrl=%2fHome%2fAbout>

user uses userid and password in login.cshtml page.

1. Then to public ActionResult Login(LoginModel model, string returnUrl) of AccountController. Here validate the user and create an authentication ticket(FormsAuthenticationTicket). Then with the ticket it creates an HttpCookie for the supplied user name & roles and adds it to the cookies collection of the response object of HttpContext or to the URL if you are using cookieless authentication.
2. Once this authentication is complete, it goes to Application\_PostAuthenticateRequest() of Global.asax
3. Since now cookie is there, it would decrypt the authentication token and extract the user data and put it into HttpContext.Current.User

HttpContext.Current.User = new System.Security.Principal.GenericPrincipal(new FormsIdentity(authTicket), roles);

1. Page lifecycle

**Init is bottoms up approach…first controls then page**

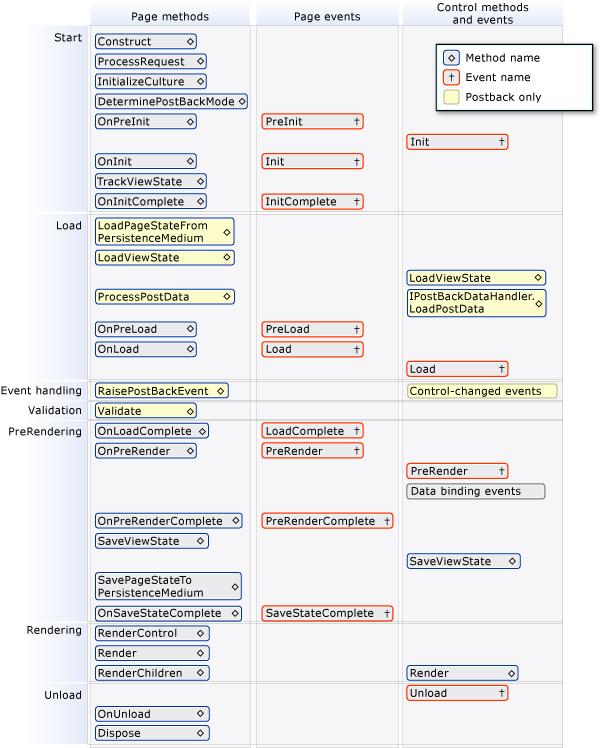
**Viewstate is loaded between Init and Load.**

**Load is top down approach.**

**Postback events and validation (IsPostBack() etc) happen between Load of control and Loadcomplete**

**PreRender is top down. DataBinding happens between PreRender of control and PreRenderComplete.**

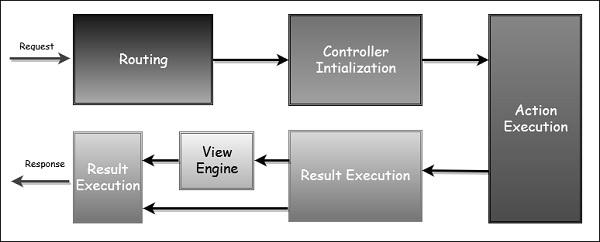
**ViewState is saved after PreRenderComplete and SaveStateComplete events. Finally Unload event.**



**Also**

<https://www.c-sharpcorner.com/UploadFile/8911c4/page-life-cycle-with-examples-in-Asp-Net/>?

**MVC Lifecycle**



1. VAR Vs DYNAMIC

**1. When were they introduced**

* var was introduced in C# 3.0
* dynamic was introduced in C# 4.0

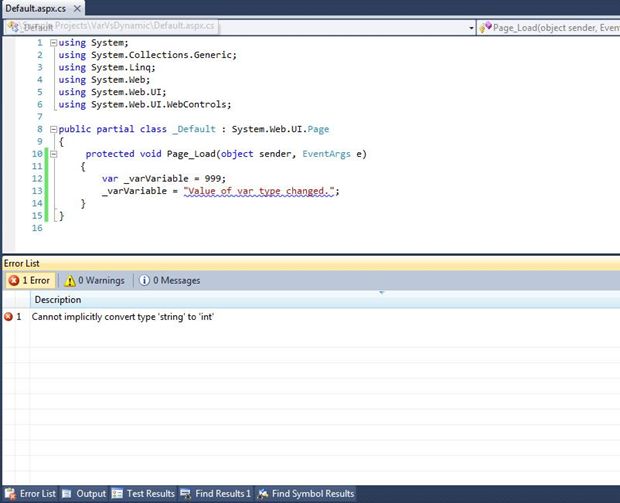
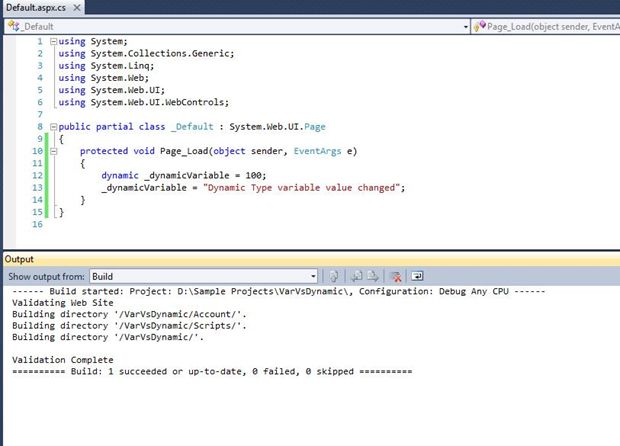
**2. Type inference of variables**

* var is a statically typed variable. It results in a strongly typed variable, in other words the data type of these variables are inferred at compile time. This is done based on the type of value that these variables are initialized with.
* dynamic are dynamically typed variables. This means, their type is inferred at run-time and not the compile time in contrast to var type.

**3. Initialization of variables**

* var type of variables are required to be initialized at the time of declaration or else they encounter the compile time error: Implicitly-typed local variables must be initialized.
* dynamic type variables need not be initialized when declared.

**4. Changing type of value assigned**

* var does not allow the type of value assigned to be changed after it is assigned to. This means that if we assign an integer value to a var then we cannot assign a string value to it. This is because, on assigning the integer value, it will be treated as an integer type thereafter. So thereafter no other type of value cannot be assigned. For example, the following code will give a compile time error:  
    
    
  + - dynamic allows the type of value to change after it is assigned to initially. In the code above, if we use dynamic instead of var, it will not only compile, but will also work at run-time. This is because, at run time, the value of the variable is first inferred as Int32 and when its value is changed, it is inferred to be a string type.  
        
      

**5. Restrictions on the usage**

* dynamic variables can be used to create properties and return values from a function.
* var variables cannot be used for property or return values from a function. They can only be used as local variable in a function.

1. what kind of errors come accross while fetching data from db

system execptions

Arthimetic exception - dividebyzero,overflowexception

Indexoutofrangeexecption

Invalidcastexecption

Nullrefernceexeception

1. Master Page lifecycle events along with user control

**First Load**:

Inti is bottoms up. Master page is merged during Init and represented as a control on content page. Load and PreRender are top down. Controls on same page events appear in the same order as they are positioned in design.

Content TextBox Init.

UserControl Page Init.

MasterPage Init.

ContentPage Init.

ContentPage Load.

MasterPage Load.

Content TextBox Load.

UserControl Page Load.

ContentPage PreRender.

MasterPage PreRender.

Content TextBox PreRender.

UserControl Page PreRender.

**Button on content page click or UserControl button click**:

Only the Button click event is added between last load and first prerender.

Content TextBox Init.

UserControl Page Init.

MasterPage Init.

ContentPage Init.

ContentPage Load.

MasterPage Load.

Content TextBox Load.

UserControl Page Load.

**Content Button Click./ UserControl Button Click.**

ContentPage PreRender.

MasterPage PreRender.

Content TextBox PreRender.

UserControl Page PreRender.

1. Insert a row in table with last row id incremented by 1

insert into Employee (EmployeeID,FirstName,LastName,EmpCode,Position,Office)

select MAX(EmployeeID)+1,'Sanjay','Chaudhary','SC','DotNet Developer','Bihar' from employee

1. WebAPI Verbs

|  |  |  |
| --- | --- | --- |
| **URL** | **VERB** | **Purpose** |
| */Dinners/* | GET | Display an HTML list of upcoming dinners. |
| */Dinners/Edit/[id]* | GET | Display an editable HTML form populated with Dinner data. |
| POST | Save the form changes for a particular Dinner to the database. |  |
| */Dinners/Create* | GET | Display an empty HTML form that allows users to define new Dinners. |
| POST | Create a new Dinner and save it in the database. |  |
| */Dinners/Delete/[id]* | GET | Display delete confirmation screen. |
| POST | Deletes the specified dinner from the database. |  |

[AcceptVerbs(HttpVerbs.Post)]

public ActionResult Edit(int id, FormCollection formValues) {

// Retrieve existing dinner

Dinner dinner = dinnerRepository.GetDinner(id);

// Update dinner with form posted values

dinner.Title = Request.Form["Title"];

dinner.Description = Request.Form["Description"];

dinner.EventDate = DateTime.Parse(Request.Form["EventDate"]);

dinner.Address = Request.Form["Address"];

dinner.Country = Request.Form["Country"];

dinner.ContactPhone = Request.Form["ContactPhone"];

// Persist changes back to database

dinnerRepository.Save();

// Perform HTTP redirect to details page for the saved Dinner

return RedirectToAction("Details", new { id = dinner.DinnerID });

}

**SOAP protocol is XML based. Works with HTTP, SMTP, FTP. Though SOAP can be used with JSON, but its not preferred. Its strict standard and bulky.**

**REST is an architectural style. REST permits different data format such as Plain text, HTML, XML, JSON, etc. But the most preferred format for transferring data is JSON**

1. Managed Vs UnManaged Code

Any language that is written in .NET Framework is **managed code**. **Managed code** uses CLR which in turns looks after your applications by managing memory, handling security, allowing cross - language debugging, and so on. The **code**, which is developed outside .NET, Framework is known as **unmanaged code**.

1. == vs Equals():

|  |  |  |
| --- | --- | --- |
|  | **==** | **Equals** |
| **Value types (int etc)** | **Content based Comparison** int i = 10; int y = 10; Console.WriteLine(i == y); // true Console.WriteLine(i.Equals(y)); // true | Content based Comparison |
| **Objects** | **Reference based Comparison** Customerobj = newCustomer(); obj.Name = "Shiv"; Customer obj1 = newCustomer(); obj1.Name = "Shiv"; Console.WriteLine(obj == obj1); // false Console.WriteLine(obj.Equals(obj1)); // false | **Reference based Comparison** Customerobj = newCustomer(); obj.Name = "Shiv"; Customer obj1 = obj; Console.WriteLine(obj == obj1); // true Console.WriteLine(obj.Equals(obj1)); // true |
| **String** | **Content based Comparison** objectstr = "test"; object str1 = "test"; Console.WriteLine(str==str1); Console.WriteLine(str.Equals(str1)); | Content based Comparison |
| **String with no interning** | **Reference based Comparison** objectstr = newstring(newchar[] { 't', 'e', 's', 't' }); object str1 = newstring(newchar[] { 't', 'e', 's', 't' }); Console.WriteLine(str==str1); // false Console.WriteLine(str.Equals(str1)); // true | Content based Comparison |
| **Type checking** | Compile time | Run time |
| **Nulls** | Works | Can crash |

1. Throwing vs handling there:

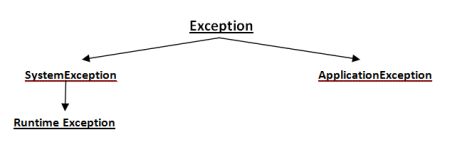
Throwing modifies the trace and the original source (line number etc) are lost.

throw; / throw ex; / throw new Exception(“additional text”,ex);

all 3 would loose the original error stack.

Example of exceptions:

* File I/O related problems
* Network communication problems
* Element access that is beyond the bound of an array
* Running out of memory during program execution
* Issue when doing a Divide-by-zero operation

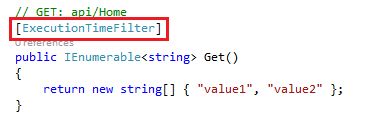


1. How to find the time taken by a action method to excute - by implementing action filters

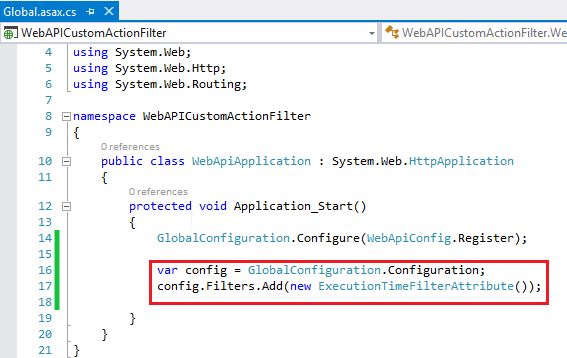
Add a customactionfileter class and onActionExecuting handler start a stopwatch and onActionExecuted handler stop it.



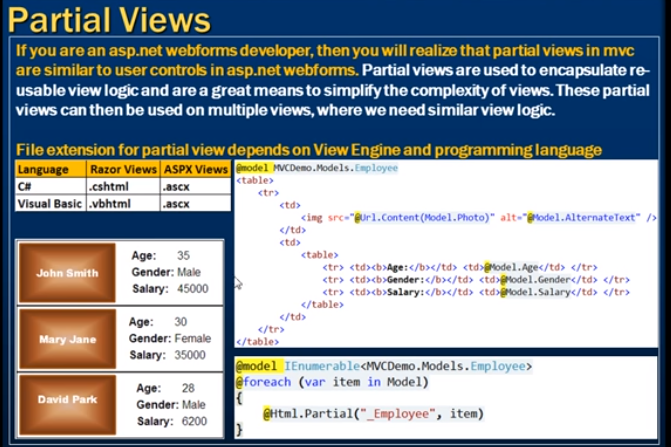
Then add this new custom action filter to the handler action method.



To add this attribute globally, we need to put it in Application\_start() method of global.asax



1. How to pass partial view to a div tag in a button click through ajax

Nothing but usercontrols.

The above is how to load partial views in another view. To load partial view using ajax, here is the extra code:

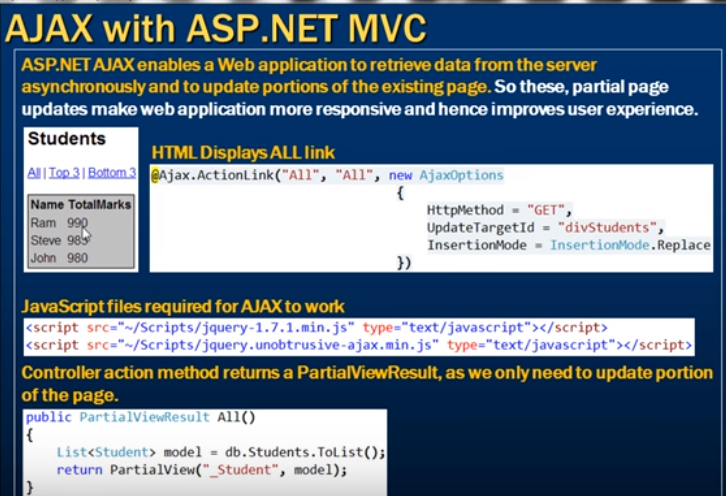
@Ajax.ActionLink("Load","DynamicLoading",new AjaxOptions{

UpdateTargetId="Div1",

InsertionMode=InsertionMode.Replace,

HttpMethod="GET"

})



1. how to post entire form to a controller

var myform = document.getElementById("myform");

var fd = new **FormData**(myform );

// FormData is default object lets you compile a set of key/value pairs to send

$.ajax({

url: "example.php",

data: fd,

type: 'POST',

success: function (dataofconfirm) {

// do something with the result

}

});

OR

var formdata = $("#EmailRequestForm").**serialize()**

1. Passing value to page in hidden field:

using (@Html.BeginForm("myMethod", "Home", FormMethod.Post))

{

//...

@Html.HiddenFor(m => m.JobId)

}

[HttpPost]

public FileStreamResult myMethod(Model model)

{

sting str = model.JobId;

}

1. ViewBag VS ViewData VS TempData

ViewData is a property of controller that exposes an instance of the ViewDataDictionary class.

ViewBag is a dynamic property (dynamic keyword which is introduced in .net framework 4.0). ViewBag is able to set and get value dynamically and able to add any number of additional fields without converting it to strongly typed.

TempData is a dictionary which is derived from TempDataDictionary class. TempData is stored data just like live session for short time. TempData keeps data for the time of HTTP Request, which means that it holds data between two consecutive requests. TempData helps us to transfer data between controllers or between actions. TempData internally use Session variables

**ViewData Example**

1. //Controller Code
2. **public** ActionResult Index()
3. {
4. List<string> Student = **new** List<string>();
5. Student.Add("Jignesh");
6. Student.Add("Tejas");
7. Student.Add("Rakesh");
9. ViewData["Student"] = Student;
10. **return** View();
11. }
12. //page code
13. <ul>
14. <% foreach (**var** student **in** ViewData["Student"] as List<string>)
15. { %>
16. <li><%: student%></li>
17. <% } %>
18. </ul>

**ViewBag Example**

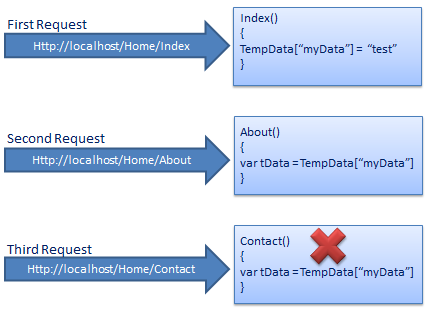
1. //Controller Code
2. **public** ActionResult Index()
3. {
4. List<string> Student = **new** List<string>();
5. Student.Add("Jignesh");
6. Student.Add("Tejas");
7. Student.Add("Rakesh");
9. ViewBag.Student = Student;
10. **return** View();
11. }
12. //page code
13. <ul>
14. <% foreach (**var** student **in** ViewBag.Student)
15. { %>
16. <li><%: student%></li>
17. <% } %>
18. </ul>

**TempData Example**

1. //Controller Code
2. **public** ActionResult Index()
3. {
4. List<string> Student = **new** List<string>();
5. Student.Add("Jignesh");
6. Student.Add("Tejas");
7. Student.Add("Rakesh");
9. TempData["Student"] = Student;
10. **return** View();
11. }
12. //page code
13. <ul>
14. <% foreach (**var** student **in** TempData["Student"] as List<string>)
15. { %>
16. <li><%: student%></li>
17. <% } %>
18. </ul>

**ViewData VS ViewBag VS TempData**

|  |  |  |
| --- | --- | --- |
| **ViewData** | **ViewBag** | **TempData** |
| It is Key-Value Dictionary collection | It is a type object | It is Key-Value Dictionary collection |
| ViewData is a dictionary object and it is property of ControllerBase class | ViewBag is Dynamic property of ControllerBase class. | TempData is a dictionary object and it is property of controllerBase class. |
| ViewData is Faster than ViewBag | ViewBag is slower than ViewData | NA |
| ViewData is introduced in MVC 1.0 and available in MVC 1.0 and above | ViewBag is introduced in MVC 3.0 and available in MVC 3.0 and above | TempData is also introduced in MVC1.0 and available in MVC 1.0 and above. |
| ViewData also works with .net framework 3.5 and above | ViewBag only works with .net framework 4.0 and above | TempData also works with .net framework 3.5 and above |
| Type Conversion code is required while enumerating | In depth, ViewBag is used dynamic, so there is no need to type conversion while enumerating. | Type Conversion code is required while enumerating |
| Its value becomes null if redirection has occurred. | Same as ViewData | TempData is used to pass data between two consecutive requests. |
| It lies only during the current request. | Same as ViewData | TempData only works during the current and subsequent request |

[](http://www.tutorialsteacher.com/Content/images/mvc/tempdata.png)

we add test data in TempData in the first request and in the second subsequent request we access test data from TempData which we stored in the first request. However, you can't get the same data in the third request because TempData will be cleared out after second request.

Call TempData.Keep() to retain TempData values in a third consecutive request.

Example: TempData.Keep()

public class HomeController : Controller

{

public HomeController()

{

}

public ActionResult Index()

{

TempData["myData"] = "Test data";

return View();

}

public ActionResult About()

{

string data;

if(TempData["myData"] != null)

data = TempData["myData"] as string;

TempData.Keep();

return View();

}

public ActionResult Contact()

{

string data;

if(TempData["myData"] != null)

data = TempData["myData"] as string;

return View();

}

}

1. How to pass control from one action method to another.

public ActionResult GetMDN(string msisdn)

{

int sngid=10;

TempData["ID"] = sngid;

return **RedirectToAction**("IIndex");

}

1. OOPS concept:

Oops known as object-oriented programming language system is the main feature of C# which further support the major features of oops including:

* Abstraction.
* Encapsulation.
* Inheritance and.
* Polymorphism.

**Abstraction Vs Encapsulation**

**Encapsulate** hides variables or some implementation that may be changed so often **in a class** to prevent outsiders access it directly. They must access it via getter and setter methods. Encapsulation is implemented by using **access specifiers**. An **access specifier** defines the scope and visibility of a class member. C# supports the following access specifiers −

* Public
* Private
* Protected
* Internal
* Protected internal

**Abstraction** is used to hiding something too but in a **higher degree(class, interface)**. Clients use an abstract class(or interface) do not care about who or which it was, they just need to know what it can do.

int number = 5;

string aStringNumber = number.ToString();

Here, ToString() is abstraction. And how this mechanism number variable converted to string and initialize into aStringNumber is encapsulation.

Polymorphism has two types:

* 1. COMPILE TIME POLYMORPHISM (EARLY BINDING OR OVERLOADING OR STATIC BINDING)
  2. RUNTIME POLYMORPHISM (LATE BINDING OR OVERRIDING OR DYNAMIC BINDING)

1. Lazy Loading?

**Lazy Loading In C#** 4.0 (Object On Demand) Object on Demand is also called **Lazy loading** pattern, **Lazy loading** delays the initialization of object.

public class Customer

{

private List<Order> \_Orders= null;

…

…

public Customer()

{

\_CustomerName = "Shiv";

\_Orders = LoadOrders(); *// Loads the order object even though //not needed*

}

private List<Order> LoadOrders()

{

List<Order> temp = new List<Order>();

Order o = new Order();

o.OrderNumber = "ord1001";

temp.Add(o);

o = new Order();

o.OrderNumber = "ord1002";

temp.Add(o);

return temp;

}

}

Solution without Lazy keyword

public class Customer

{

private List<Order> \_Orders= null;

…

…

public Customer()

{

\_CustomerName = "Shiv";

}

public List<Order> Orders

{

get

{

if (\_Orders == null)

{

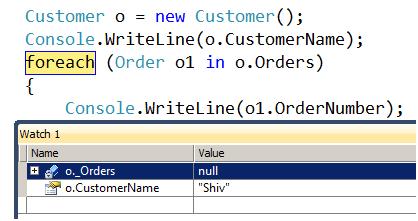
\_Orders = LoadOrders();

}

return \_Orders;

}

}



With Lazy keyword

In .NET we have the Lazy<T> class which provides automatic support for lazy loading.

Create the object of orders using the Lazy generic class.

Attach this Lazy<> object with the method which will help us load the order’s data.

Now as soon as any client makes a call to the \_Orders object, it will call the LoadOrders function to load the data.

You will get the List<orders> data in the Value property.

public class Customer

{

private Lazy<List<Order>> \_Orders= null;

public List<Order> Orders

{

get

{

return \_Orders.Value;

}

}

public Customer()

{

*// Makes a database trip*

\_CustomerName = "Shiv";

\_Orders = new Lazy<List<Order>>(() => LoadOrders());

}

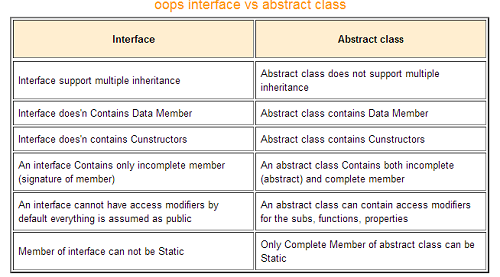
}

1. Abstract vs Sealed

Sealed must be used with override keyword, because if the base class method was not overridable, then current class or any sub class won’t be overriding it. Hence no point in making current class sealed.

|  |  |
| --- | --- |
| **Sealed Class/method** | **Abstract Class** |
| 1)Sealed class cannot be inherited by a normal class. | 1)Abstract class must be inherited by a class. |
| 2)Instance must be used for Sealed class for accessing its public methods. | 2)Instance cannot be created for Abstract class and it should be inherited for accessing its abstract methods. |
| 3)Sealed class methods cannot be override. | 3)Abstract class methods can be override. |
| 4) Sealed class can be a derived class but can't be a base class. |  |
| 5.)  A sealed class cannot also be an abstract class. |  |
| public class BaseClass {  public virtual void Display()  {  Console.WriteLine("Virtual method");  } }  public class DerivedClass : BaseClass {  // Now the display method have been sealed and can;t be overridden  public override sealed void Display()  {  Console.WriteLine("Sealed method");  } }  //public class ThirdClass : DerivedClass  //{  // public override void Display()  // {  // Console.WriteLine("Here we try again to override display method which is not possible and will give error");  // }  //} | |

Interface Vs Abstract Class:



|  |  |
| --- | --- |
| **Abstract Class** | **Interface** |
| An abstract class can have all access modifiers for member declaration of functions, subs, and properties. | An interface is not allowed to have access modifiers for member declaration of functions, subs, properties, etc. All the members are assumed as implicitly public. |
| A class can at the most use only one abstract class. | A class may inherit any number of interfaces. |
| Multiple inheritance is not supported in abstract class. | An interface may support multiple inheritance. |
| Technically, it’s a class which may or may not contain both definition and its implementation. | An interface can only have the signature of functionality so it’s basically an empty shell. |
| An abstract class can contain members. consts, defined methods, and method stubs. | An interface can only contain methods and consts. |
| It defines the core identity of a class and is used to describe objects of the same data type. | It is used to describe the peripheral abilities of a class. |
| Members of an abstract class cannot be static unless it’s a Complete Member. | Members of an interface cannot be static. |
| It’s ideal for implementations of the same kind and common behavior. | It’s better to use interfaces if several implementations share only method signatures. |
| It can have constructor declaration. | [It cannot have constructor declaration.](http://www.differencebetween.net/technology/difference-between-constructor-and-destructor/) |
| An abstract class has pre-defined fields and constrants. | Fields cannot be defined in interfaces. |
| It can have both abstract and non-abstract methods. | It can only have abstract methods. |

1. Singleton class

public class Singleton {

private static final Singleton instance;

private Singleton(){}

public static Singleton getInstance() {

**if (instance == null)**

**instance = new Singleton();**

return instance;

}

}

1. REST PRINCIPLES:

**Uniform interface** -Once a developer becomes familiar with one of your API, he should be able to follow the similar approach for other APIs.

**Client–server** - Servers and clients may also be replaced and developed independently, as long as the interface between them is not altered.

**Stateless** - No client context shall be stored on the server between requests. The client is responsible for managing the state of the application.

**Cacheable** - Well-managed caching partially or completely eliminates some client-server interactions, further improving scalability and performance

**Layered system** – 3 layered architecture

**Code on demand (optional)**

1. ASP.NET vs MVC:

**Asp.Net Web Forms:**

* Follows traditional event driven development model.
* Web Form has server controls.
* Web Form has state management (like as view state, session) techniques.
* Asp.Net Web Form has file-based URLs means file name exist in the URLs must have its physically existence.
* Asp.Net Web Form is not Open Source.

**Asp.Net MVC :**

* Asp.Net MVC is a lightweight and follow MVC (Model, View, Controller) pattern based development model.
* Asp.Net MVC has html helpers.
* Asp.Net MVC has no automatic state management techniques.
* Asp.Net MVC has route-based URLs means URLs are divided into controllers and actions and moreover it is based on controller not on physical file.
* Asp.Net MVC is lightweight, provide full control over markup and support many features that allow fast & agile development. Hence it is best for developing interactive web application with latest web standards.
* Asp.Net Web MVC is an Open Source.

1. WEB Service, WCF and WEB API:

## **Web Service**

* It is based on SOAP and return data in XML form.
* It support only HTTP protocol.
* It is not open source but can be consumed by any client that understands xml.
* It can be hosted only on IIS.

## **WCF**

* It is also based on SOAP and return data in XML form.
* It is the evolution of the web service(ASMX) and support various protocols like TCP, HTTP, HTTPS, Named Pipes, MSMQ.
* The main issue with WCF is, its tedious and extensive configuration.
* It is not open source but can be consumed by any client that understands xml.
* It can be hosted with in the applicaion or on IIS or using window service.

## **WCF Rest**

* To use WCF as WCF Rest service you have to enable webHttpBindings.
* It support HTTP GET and POST verbs by [WebGet] and [WebInvoke] attributes respectively.
* To enable other HTTP verbs you have to do some configuration in IIS to accept request of that particular verb on .svc files
* Passing data through parameters using a WebGet needs configuration. The UriTemplate must be specified.
* It support XML, JSON and ATOM data format.

## **Web API**

* This is the new framework for building HTTP services with easy and simple way.
* Web API is open source an ideal platform for building REST-ful services over the .NET Framework.
* Unlike WCF Rest service, it use the full feature of HTTP (like URIs, request/response headers, caching, versioning, various content formats)
* It also supports the MVC features such as routing, controllers, action results, filter, model binders, IOC container or dependency injection, unit testing that makes it more simple and robust.
* It can be hosted with in the application or on IIS.
* It is light weight architecture and good for devices which have limited bandwidth like smart phones.
* Responses are formatted by Web API’s MediaTypeFormatter into JSON, XML or whatever format you want to add as a MediaTypeFormatter.

To whom choose between WCF or WEB API

* Choose WCF when you want to create a service that should support special scenarios such as one way messaging, message queues, duplex communication etc.
* Choose WCF when you want to create a service that can use fast transport channels when available, such as TCP, Named Pipes, or maybe even UDP (in WCF 4.5), and you also want to support HTTP when all other transport channels are unavailable.
* Choose Web API when you want to create a resource-oriented services over HTTP that can use the full features of HTTP (like URIs, request/response headers, caching, versioning, various content formats).
* Choose Web API when you want to expose your service to a broad range of clients including browsers, mobiles, iphone and tablets.

1. BUNDLING and MINIFICATION:

### **Bundling**

To improve the performance of the application, ASP.NET MVC provides inbuilt feature to bundle multiple files into a single, file which in turn improves the page load performance because of fewer HTTP requests.

Bundling is a simple logical group of files that could be referenced by unique name and loaded with a single HTTP request.

By default, the MVC application's BundleConfig (located inside App\_Start folder) comes with the following code −

public static void RegisterBundles(BundleCollection bundles) {

// Following is the sample code to bundle all the css files in the project

// The code to bundle other javascript files will also be similar to this

bundles.Add(new StyleBundle("~/Content/themes/base/css").Include(

"~/Content/themes/base/jquery.ui.core.css",

"~/Content/themes/base/jquery.ui.tabs.css",

"~/Content/themes/base/jquery.ui.datepicker.css",

"~/Content/themes/base/jquery.ui.progressbar.css",

"~/Content/themes/base/jquery.ui.theme.css"));

}

The above code basically bundles all the CSS files present in Content/themes/base folder into a single file.

### **Minification**

Minification is another such performance improvement technique in which it optimizes the javascript, css code by shortening the variable names, removing unnecessary white spaces, line breaks, comments, etc. This in turn reduces the file size and helps the application to load faster.

1. AJAX call on button click:

<script src="http://code.jquery.com/jquery-1.11.0.min.js"></script>

<script>

$(document).ready(function(){

$("#Shareitem").click(function(e){

e.preventDefault();

$.ajax({type: "POST",

url: "/imball-reagens/public/shareitem",

data: { id: $("#Shareitem").val(), access\_token: $("#access\_token").val() },

success:function(result){

$("#sharelink").html(result);

}});

});

});

</script>

1. The noConflict() method releases the hold on the $ shortcut identifier, so that other scripts can use it.You can of course still use jQuery, simply by writing the full name instead of the shortcut:

$.noConflict();  
jQuery(document).ready(function(){  
    jQuery("button").click(function(){  
        jQuery("p").text("jQuery is still working!");  
    });  
});

The noConflict() method returns a reference to jQuery, that you can save in a variable, for later use.

**var jq = $.noConflict();**  
jq(document).ready(function(){  
    jq("button").click(function(){  
        jq("p").text("jQuery is still working!");  
    });  
});

1. SQL PROFILER:

SQL query tracing tool to analyze the performance of query.

1. Palindrome program:

using System;  
namespace palindrome  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            string s,revs="";  
            Console.WriteLine(" Enter string");  
            s = Console.ReadLine();  
            for (int i = s.Length-1; i >=0; i--) //String Reverse  
            {  
                revs += s[i].ToString();  
            }  
            if (revs == s) // Checking whether string is palindrome or not  
            {  
                Console.WriteLine("String is Palindrome \n Entered String Was {0} and reverse string is {1}", s, revs);  
            }  
            else  
            {  
                Console.WriteLine("String is not Palindrome \n Entered String Was {0} and reverse string is {1}", s, revs);  
            }  
            Console.ReadKey();  
        }  
    }  
}

1. Jira the agile tool:
2. SELECT **ISNULL**('Joydip', 'W3Schools.com') 🡪 ‘Joydip’

SELECT **ISNULL**(NULL, 'W3Schools.com') 🡪 'W3Schools.com’

1. Deleting Duplicate rows using Row\_Number() function:

With CTE as

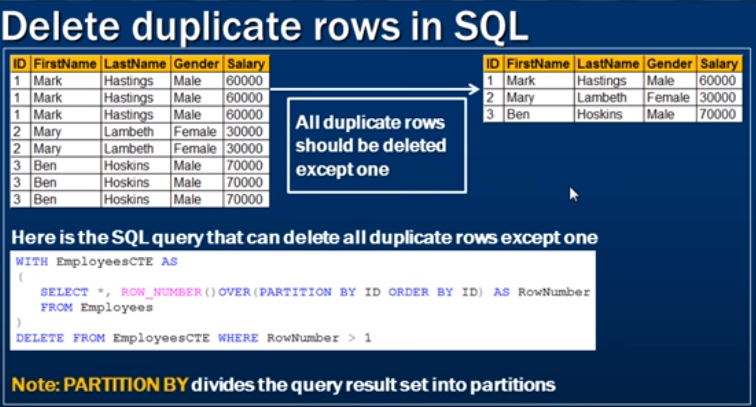
(

SELECT \*, ROW\_NUMBER() OVER (PARTITION BY ID ORDER BY ID DESC) AS ROWNUMBER

FROM EMPLOYEES

)

DELETE FROM EMPLOYEES WHERE ROWNUMBER >1



<https://www.youtube.com/watch?v=fvPddKyHxpQ&list=PL6n9fhu94yhXcztdLO7i6mdyaegC8CJwR>

**Removing Duplicates using self referencing method:**

DELETE S2

  FROM [dbo].[Student] S1,

  [dbo].[Student] S2

WHERE S1.StudentId < S2.StudentId

  AND S1.Name = S2.Name

  AND S1.Course = S2.Course

  AND S1.Marks = S2.Marks

  AND S1.ExamDate = S2.ExamDate

1. **Swapping of two numbers without third variable**

#include <stdio.h>

int main()

{

   int a, b;

   printf("Input two integers (a & b) toswap**\n**");

   scanf("%d%d", &a, &b);

   a = a + b;

   b = a - b;

   a = a - b;

  printf("a = %d**\n**b = %d**\n**",a,b);

   return 0;

}

1. **Passing Reference/out type and value type**

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

{

int a1 = 20;

int b1 = 30;

//getdata(ref a1, b1);

int c1= getdata( a1, b1);

Response.Write("Print a" + a1 + "Print b" + b1);

}

public int getdata(int a, int b)

{

//public int getdata(ref int a, int b)

//{

a = 21;

b = 31;

return a + b;

}

1. SP vs Function

|  |  |
| --- | --- |
| **Stored Procedure** | **UDF** |
| SP can return zero , single or multiple values. | Function must return a single value (which may be a scalar or a table). |
| We can use transaction in SP | No |
| SP can have input/output parameters | Only Input parameter |
| We can call function from SP | We can't call SP from function. |
| We can't use SP in SELECT/WHERE/ HAVING statement. | We can. |
| We can use exception handling using Try-Catch block in SP. | We cant |
| We can execute Dynamic SQL. | We cant |
| Temp Variables can be used | Temporary tables cannot be used |

1. Can a static method be in abstract class?

Yes, Static Method can be called from abstract method. Object of abstract class can't be created but to call static object is not required.

public abstract class TestAbstract

{

public static string test()

{

return "Yes we can call static method from abstract class";

}

}

//Calling part ===========

class Program

{

static void Main(string[] args)

{

string test = TestAbstract.test();

}

}

1. Should all methods of an abstract class be implemented?

A non-abstract class derived from an abstract class must include actual implementations of all inherited abstract methods and accessors. If the method is abstract, you must override.

1. How to call Code behind method from **JQUERY**

**Cs File (codebehind)**

[WebMethod]

public static string IsExists(string value)

{

return "True";

}

**Javascript/jQuery**

function IsExists(pagePath, dataString)

{

**$.ajax({**

**type:"POST",**

**url: pagePath,**

**data: dataString,**

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

**dataType:"json",**

**error:**

**function(XMLHttpRequest, textStatus, errorThrown) {**

**alert("Error");**

**},**

**success:**

**function(result) {**

**alert( result.d);**

**}**

**}**

**});}**

var pagePath = window.location.pathname + "/IsExists";

var dataString = "{ 'value':'ab" }";

IsExists(pagePath, dataString);

1. JQUERY Selectors

|  |  |  |
| --- | --- | --- |
| **Selector** | **Example** | **Selects** |
| [\*](https://www.w3schools.com/jquery/sel_all.asp) | $("\*") | All elements |
| [#id](https://www.w3schools.com/jquery/sel_id.asp) | $("#lastname") | The element with id="lastname" |
| [.class](https://www.w3schools.com/jquery/sel_class.asp) | $(".intro") | All elements with class="intro" |
| [element](https://www.w3schools.com/jquery/sel_element.asp) | $("p") | All <p> elements |
| [:first](https://www.w3schools.com/jquery/sel_first.asp) | $("p:first") | The first <p> element |
| [:last](https://www.w3schools.com/jquery/sel_last.asp) | $("p:last") | The last <p> element |
| [:even](https://www.w3schools.com/jquery/sel_even.asp) | $("tr:even") | All even <tr> elements |
| [:odd](https://www.w3schools.com/jquery/sel_odd.asp) | $("tr:odd") | All odd <tr> elements |
| [:first-child](https://www.w3schools.com/jquery/sel_firstchild.asp) | $("p:first-child") | All <p> elements that are the first child of their parent |
| [:nth-child(n)](https://www.w3schools.com/jquery/sel_nthchild.asp) | $("p:nth-child(2)") | All <p> elements that are the 2nd child of their parent |
| [parent > child](https://www.w3schools.com/jquery/sel_parent_child.asp) | $("div > p") | All <p> elements that are a direct child of a <div> element |
| [parent descendant](https://www.w3schools.com/jquery/sel_parent_descendant.asp) | $("div p") | All <p> elements that are descendants of a <div> element |
| [element + next](https://www.w3schools.com/jquery/sel_previous_next.asp) | $("div + p") | The <p> element that are next to each <div> elements |
| [element ~ siblings](https://www.w3schools.com/jquery/sel_previous_siblings.asp) | $("div ~ p") | All <p> elements that are siblings of a <div> element |
| [:focus](https://www.w3schools.com/jquery/sel_focus.asp) | $(":focus") | The element that currently has focus |
| [:contains(text)](https://www.w3schools.com/jquery/sel_contains.asp) | $(":contains('Hello')") | All elements which contains the text "Hello" |
| [:hidden](https://www.w3schools.com/jquery/sel_hidden.asp) | $("p:hidden") | All hidden <p> elements |
| [[attribute]](https://www.w3schools.com/jquery/sel_attribute.asp) | $("[href]") | All elements with a href attribute |
| [:input](https://www.w3schools.com/jquery/sel_input.asp) | $(":input") | All input elements |
| [:text](https://www.w3schools.com/jquery/sel_input_text.asp) | $(":text") | All input elements with type="text" |
| [:radio](https://www.w3schools.com/jquery/sel_input_radio.asp) | $(":radio") | All input elements with type="radio" |
| [:checkbox](https://www.w3schools.com/jquery/sel_input_checkbox.asp) | $(":checkbox") | All input elements with type="checkbox" |
| [:submit](https://www.w3schools.com/jquery/sel_input_submit.asp) | $(":submit") | All input elements with type="submit" |
| [:enabled](https://www.w3schools.com/jquery/sel_input_enabled.asp) | $(":enabled") | All enabled input elements |
| [:disabled](https://www.w3schools.com/jquery/sel_input_disabled.asp) | $(":disabled") | All disabled input elements |
| [:selected](https://www.w3schools.com/jquery/sel_input_selected.asp) | $(":selected") | All selected input elements |
| [:checked](https://www.w3schools.com/jquery/sel_input_checked.asp) | $(":checked") | All checked input elements |

1. Difference between **s**tring and **S**tring:

string greet = String.Format("Hello {0}!", place);

They are same. Only while referring to an object, string is used. But String is used to refer to Class

1. Using Keyword

**Using** can also be used in a block of code to dispose a IDisposable objects.

using(SqlConnection sqlconnection= new SqlConnection(connectionString) )

{

SqlCommand cmd = new SqlCommand(commandString, sqlconnection);

sqlconnection.Open();

1. How to call JQuery functions from code behind.

**JS:**

<script>

function Confirm()

{

$(".saved").fadeIn(500).fadeOut(500);

}

</script>

**c#:**

protected void btn\_click(object sender, Eventargs e)

{

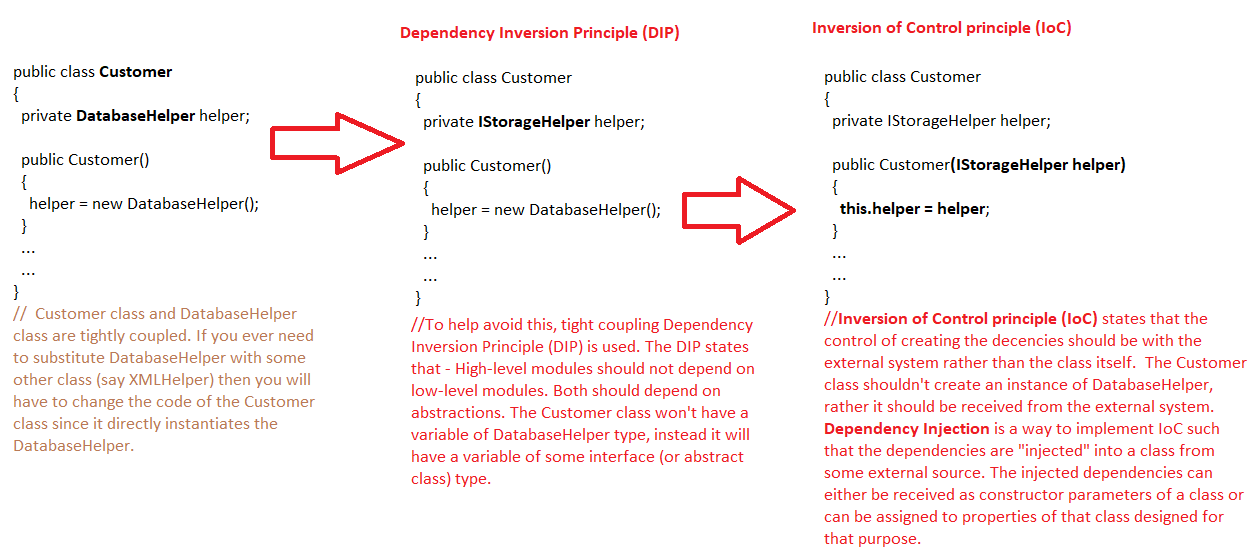
ScriptManager.RegisterStartupScript(this.Page,this.GetType(), "script", "Confirm();", true);

}

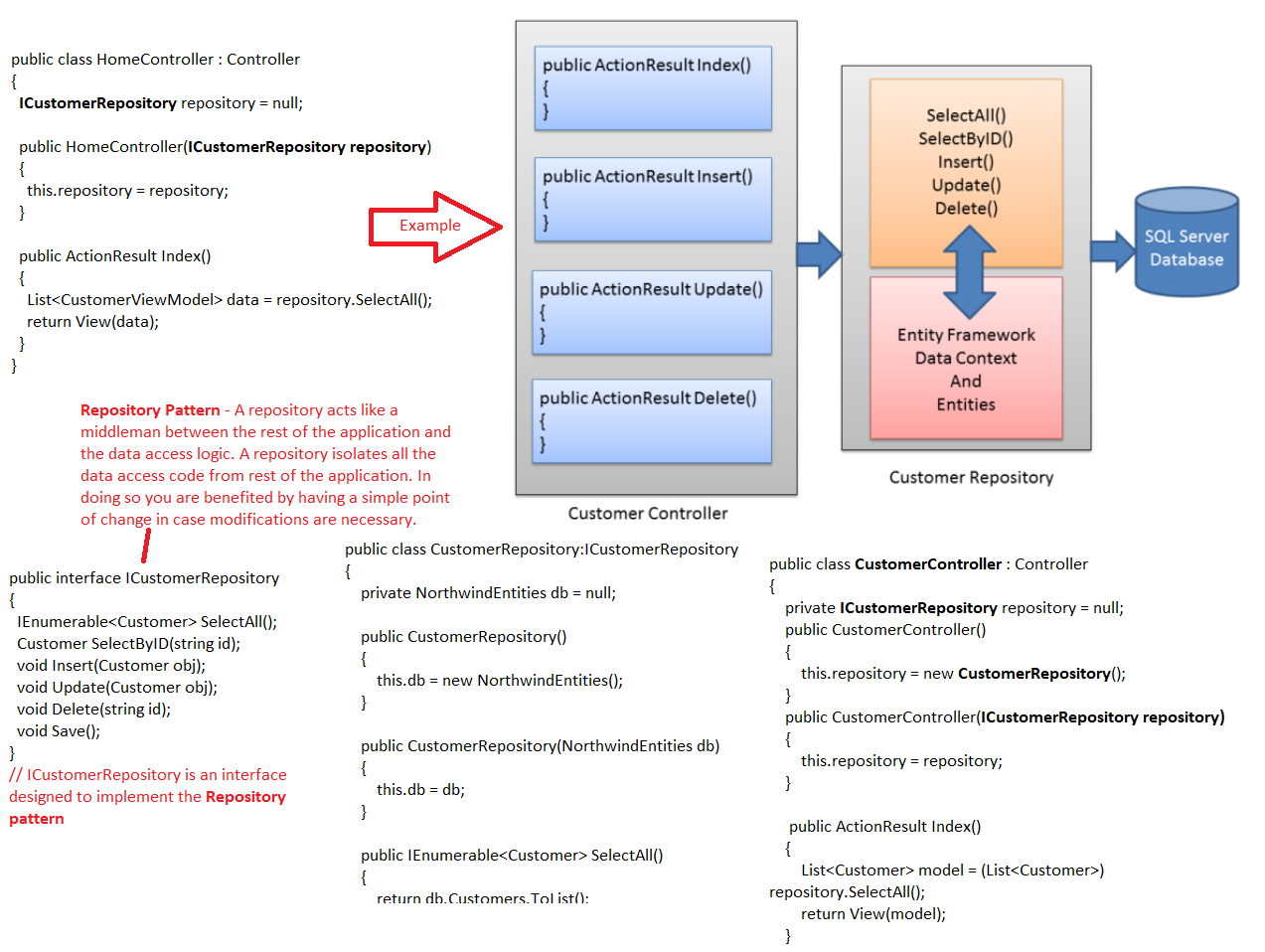
1. Dependency Injection

**DI** is a software design pattern that allow us to develop loosely coupled code. **Inversion of Control** **(IoC**) refers to a programming style where a framework controls the program flow with the help of Dependency Injection.

* 1. High-level modules should not depend on low-level modules. Both should depend on abstractions.
  2. Control of creating the decencies should be with the external system rather than the class itself



The abstraction (in this case the interface is designed in Repository pattern)



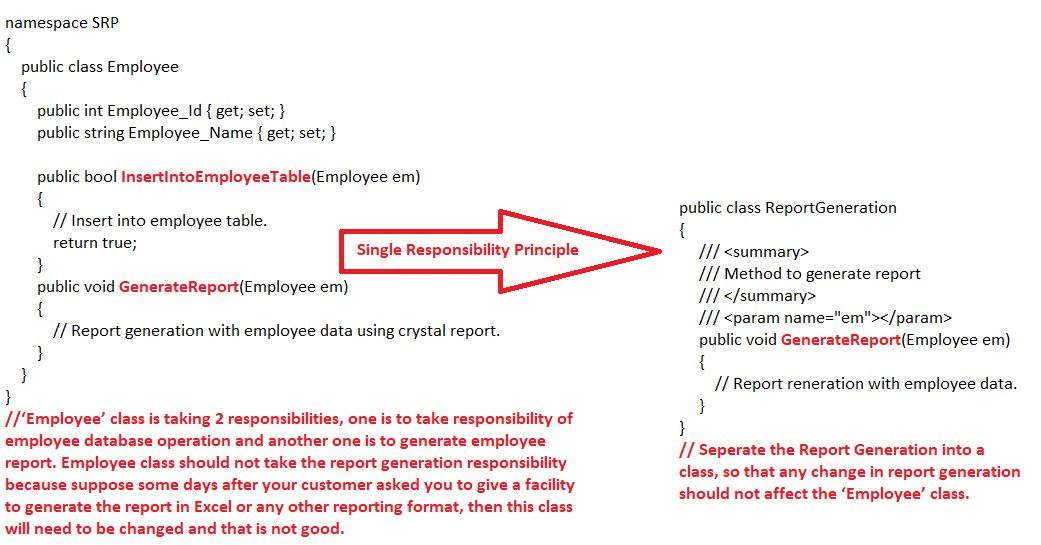
1. SOLID Principle:

## SOLID Principles

* [Single Responsibility Principle](https://blog.scottlogic.com/2018/06/26/solid-principles.html#single-responsibility-principle)
* Open for Extension Closed for Modification
* [Liskov Substitution Principle](https://blog.scottlogic.com/2018/06/26/solid-principles.html#liskov-substitution-principle)
* [Interface Segregation Principle](https://blog.scottlogic.com/2018/06/26/solid-principles.html#interface-segregation-principle)
* [Dependency Inversion](https://blog.scottlogic.com/2018/06/26/solid-principles.html#dependency-inversion)

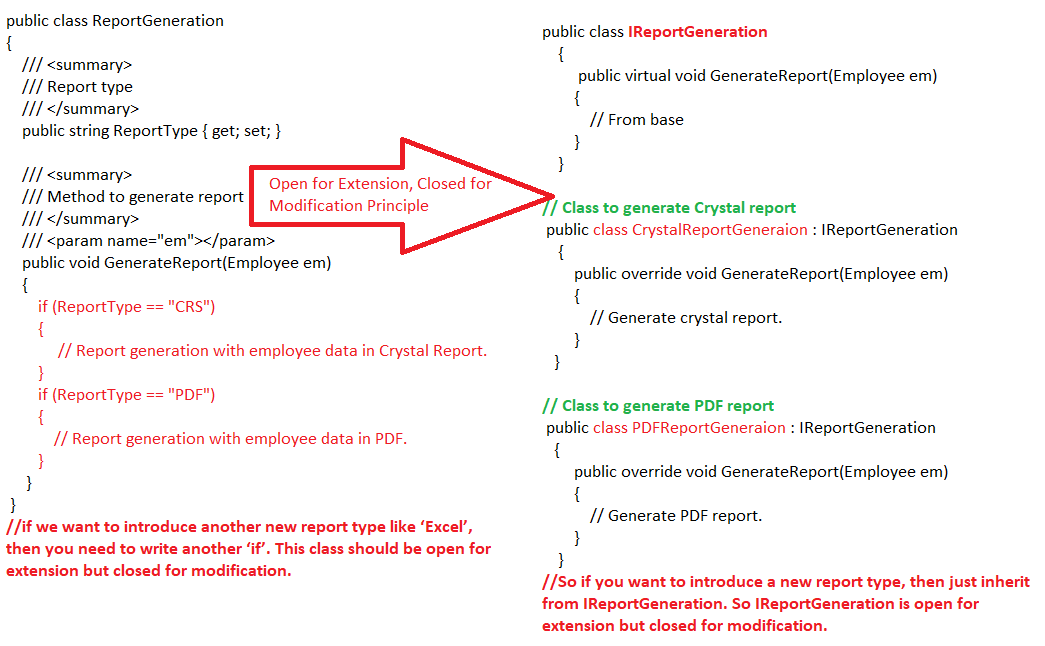
### Single Responsibility Principle

### A class should take care of a Single Responsibility.



### Open for Extension Closed for Modification

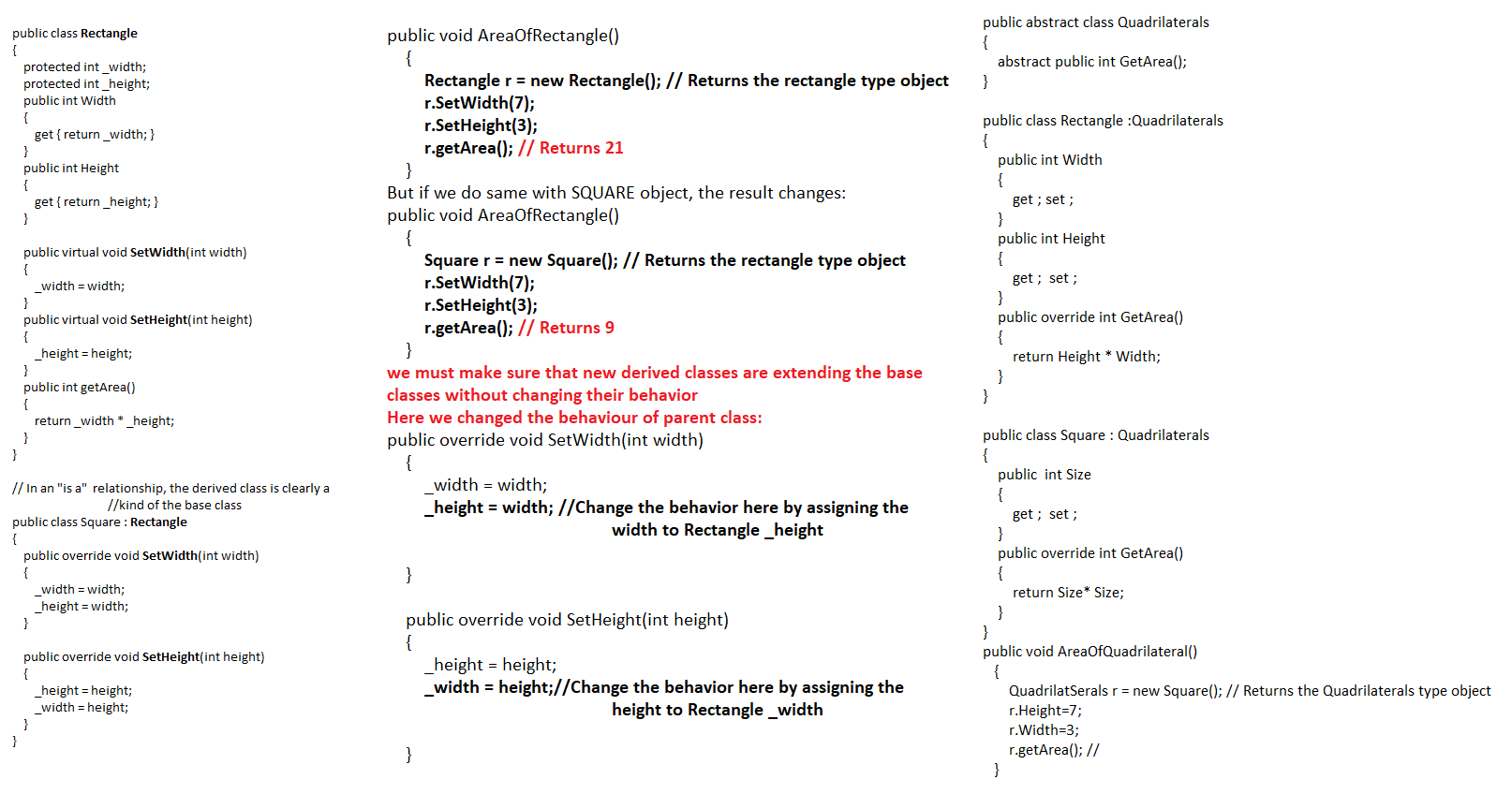
### Prefer extension over modification



### Liskov Substitution Principle

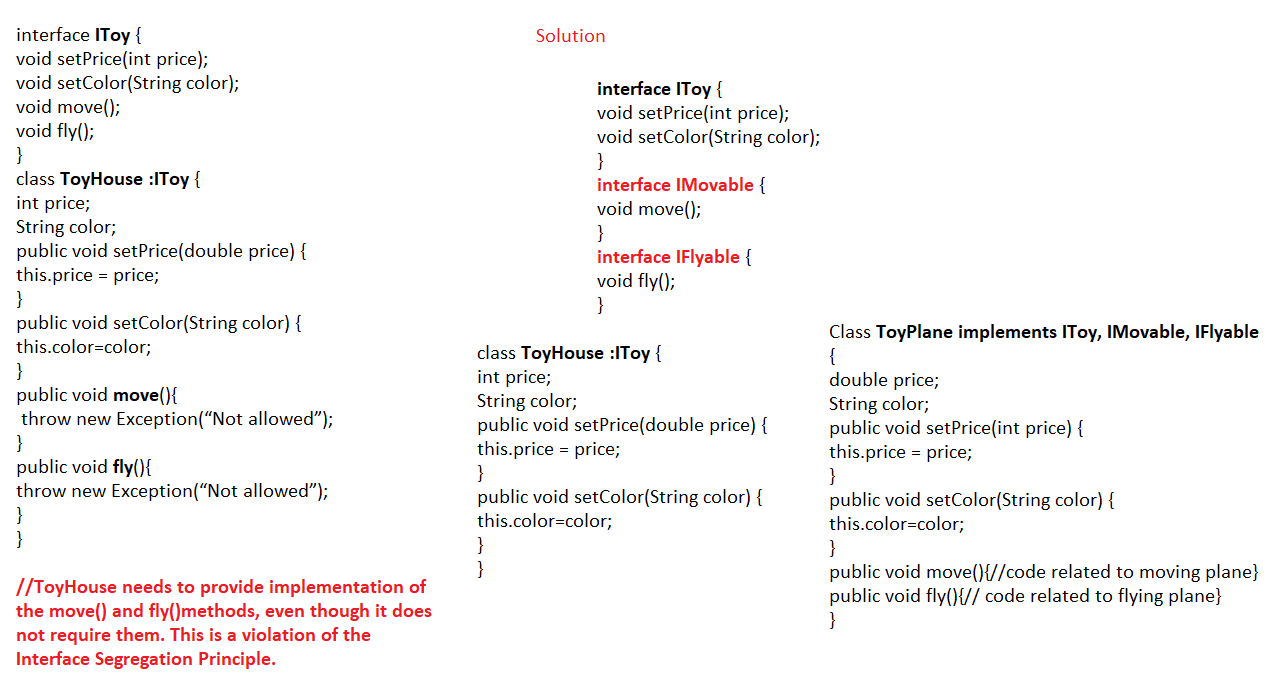
### The parent class should be able to refer child objects seamlessly during runtime polymorphism.





### Interface Segregation Principle

### A client should not be forced to use an interface, if it doesn’t need it.



### Dependency Inversion

### High level modules should not depend on low-level modules, but should depend on abstraction.

<https://medium.com/@mirzafarrukh13/solid-design-principles-c-de157c500425>

<https://www.codeproject.com/Tips/1033646/SOLID-Principle-with-Csharp-Example>

**Misc Vedios:**

<https://www.facebook.com/pg/DotNetInterviewQuestions/videos/?ref=page_internal>

1. How to pass table as parameter from C# to SP

CREATE TABLE CUSTOMER  
(  
      CustomerId INT NOT NULL,   
      CustomerName VARCHAR(MAX),   
      Isdeleted BIT,  
      PRIMARY KEY (CustomerId)  
)  
  
CREATE TYPE dbo.TableValuedTypeExample AS TABLE   
(  
      CustomerId INT NOT NULL,   
      CustomerName VARCHAR(MAX),   
      PRIMARY KEY (CustomerId)  
)

CREATE PROC InsertValue  
(@TempTable AS dbo.TableValuedTypeExample READONLY)  
AS  
BEGIN  
      INSERT INTO CUSTOMER (CustomerId,CustomerName ,Isdeleted )  
      SELECT CustomerId, CustomerName, 0 AS Isdeleted FROM @TempTable  
END

DataTable myTable = CreateTable();

static DataTable CreateTable()  
{  
    DataTable dt = new DataTable();  
    dt.Columns.Add("CustomerId", typeof(Int32));  
    dt.Columns.Add("CustomerName", typeof(string));  
    return dt;  
}

//Pass table Valued parameter to Store Procedure  
SqlParameter sqlParam = cmd.Parameters.AddWithValue("@TempTable", myTable);  
sqlParam.SqlDbType = SqlDbType.Structured;   
cmd.ExecuteNonQuery();

1. Can a table have multiple primary key

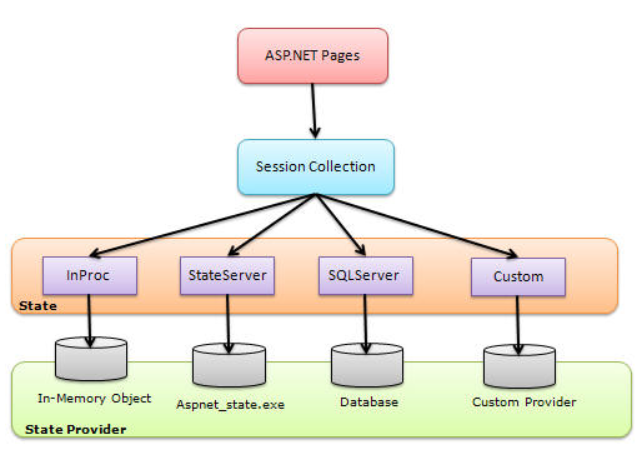
No. Only one. But there can be multiple candidate key.

There is something called Composite primary key. In this, multiple columns are together used to create a primary key.

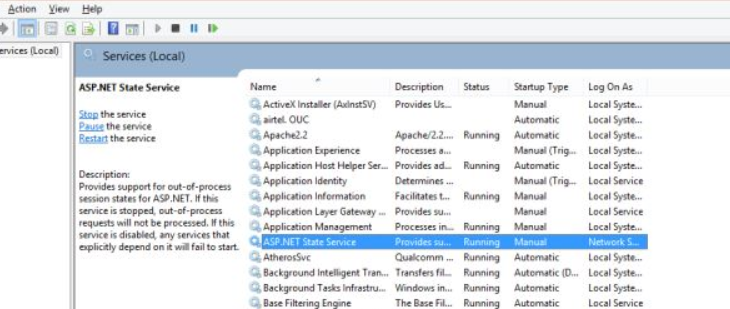
CREATE TABLE Persons (  
    ID int NOT NULL PRIMARY KEY,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int  
);

CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    CONSTRAINT PK\_Person PRIMARY KEY (ID,LastName)  
);

1. Constraints in SQL Server
   1. [**NOT NULL**](https://www.w3schools.com/sql/sql_notnull.asp) - Ensures that a column cannot have a NULL value
   2. [**UNIQUE**](https://www.w3schools.com/sql/sql_unique.asp) - Ensures that all values in a column are different
   3. [**PRIMARY KEY**](https://www.w3schools.com/sql/sql_primarykey.asp) - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
   4. [**FOREIGN KEY**](https://www.w3schools.com/sql/sql_foreignkey.asp) - Uniquely identifies a row/record in another table
   5. [**CHECK**](https://www.w3schools.com/sql/sql_check.asp) - Ensures that all values in a column satisfies a specific condition
   6. [**DEFAULT**](https://www.w3schools.com/sql/sql_default.asp) - Sets a default value for a column when no value is specified
   7. [**INDEX**](https://www.w3schools.com/sql/sql_create_index.asp) - Used to create and retrieve data from the database very quickly
2. ASP.Net State Management



sessionState mode="InProc". **Not Suitable for WebFarms and WebGardens where loadbalancer is there. Also Increase the load of server**



<sessionState mode="StateServer" stateConnectionString="tcpip=localhost:42424">

Generally the state services and web services are not in the same machine of a dedicated server so for the connection we need to write the stateConnectionString and here we need to provide the IP address or the name of the machine where the state service is running and here my services are run on localhost and 42424 is the port number.

Can be used with Web Farm and Web Garden: It supports a Web Farm and Web Garden both.

**Performance Decrease**

**Finally:**

<sessionState mode="SQLServer" sqlConnectionString="Server=DIVS\SQLEXPRESS;Integrated Security=true">

SQL Server is a more reliable and secure option.

1. Exception Filter

**[HandleError]**

public class HomeController : Controller

{

public ActionResult Index()

{

throw new Exception("This is unhandled exception");

return View();

}

now it will display Error page if any action method of HomeController would throw unhandled exception.

class MyErrorHandler : FilterAttribute, IExceptionFilter

{

public override void IExceptionFilter.OnException(ExceptionContext filterContext)

{

Log(filterContext.Exception);

base.OnException(filterContext);

}

private void Log(Exception exception)

{

//log exception here..

}

}

1. Http Handlers and Modules

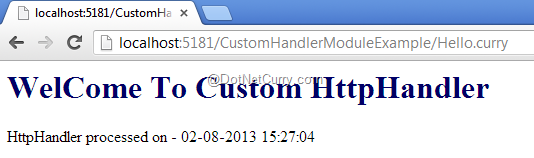
When a client makes a request for a resource located on the server in an ASP.NET application, each request is handled by the HTTP Handlers. Microsoft ASP.NET has number of built-in HTTP Handlers which serves different files like .ASPX, .ASMX etc. Based on the extension of the file, the appropriate HTTP Handlers gets loaded which is mapped to the extension and is responsible for processing the ASP.NET request.

|  |
| --- |
| public class CustomHandler:IHttpHandler  {      public bool IsReusable      {          get { return false; }      }        public void ProcessRequest(HttpContext context)      {          context.Response.Write("<h1 style='Color:#000066'>WelCome To Custom HttpHandler</h1>");          context.Response.Write("HttpHandler processed on - " + DateTime.Now.ToString());          using (StreamWriter SW=new StreamWriter(@"E:\HandlerMessages.txt",true))          {              SW.WriteLine("The message date time is - " + DateTime.Now.ToString());              SW.Close();          }      }  } |

Now let's configure our custom HttpHandler into Web.Config file as shown below -

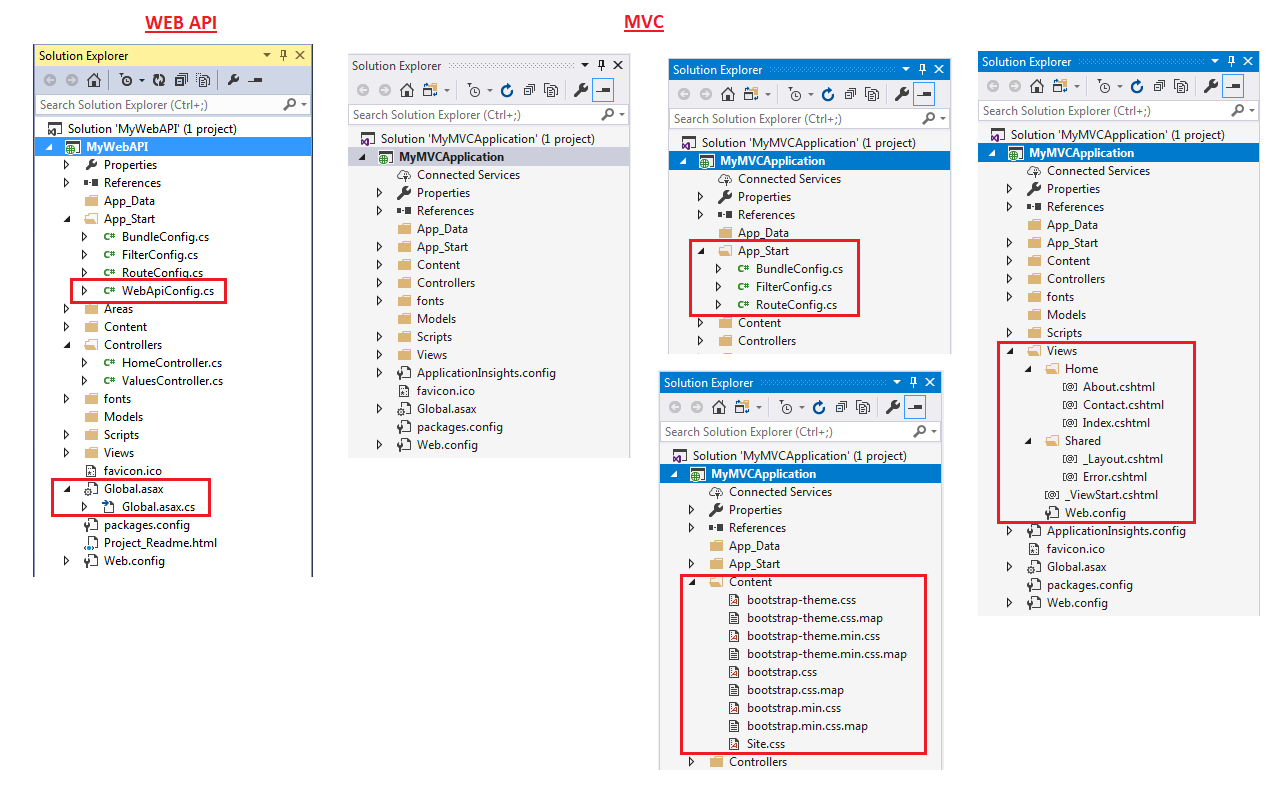
|  |
| --- |
| <httpHandlers>          <add verb="\*" path="\*.curry" type="CustomHandlerModuleExample.CustomHandler"/>  </httpHandlers> |

Now add a simple text file with extension ".curry" and browse the ".curry" extension file. It should look like the following-



In a single request processing, there can be more than one modules which gets executed. HttpModules take part in processing of the request by handling the Application events. There are number of events which you can handle during the HttpModule processing. For example - BeginRequest(), EndRequest(), AuthenticateRequest() etc.

1. MVC VS WEBAPI folder structure



1. [**https://dotnettutorials.net/course/csharp-dot-net-tutorials/**](https://dotnettutorials.net/course/csharp-dot-net-tutorials/)
2. <https://www.c-sharpcorner.com/UploadFile/puranindia/ASP-NET-MVC-Interview-Questions/>
3. **Tuple in C#**

The Tuple<T> class was introduced in .NET Framework 4.0. A tuple is a data structure that contains a sequence of elements of different data types. It can be used where you want to have a data structure to hold an object with properties, but you don't want to create a separate type for it.

Tuple<int, string, string> person =

new Tuple <int, string, string>(1, "Steve", "Jobs");

Also:

var person = Tuple.Create(1, "Steve", "Jobs");

person.Item1; // returns 1

person.Item2; // returns "Steve"

person.Item3; // returns "Jobs"

A tuple can only include maximum eight elements. It gives a compiler error when you try to include more than eight elements.

1. Indexer in C#

An Indexer is a special type of property that allows a class or structure to be accessed the same way as array for its internal collection. It is same as property except that it defined with **this** keyword with square bracket and parameters.

class StringDataStore

{

private string[] strArr = new string[10]; // internal data storage

public string this[int index]

{

get

{

if (index < 0 && index >= strArr.Length)

throw new IndexOutOfRangeException("Cannot store more than 10 objects");

return strArr[index];

}

set

{

if (index < 0 || index >= strArr.Length)

throw new IndexOutOfRangeException("Cannot store more than 10 objects");

strArr[index] = value;

}

}

}

class Program

{

static void Main(string[] args)

{

StringDataStore strStore = new StringDataStore();

**//class being directly accessed as array for the internal string arraystrArr**

strStore[0] = "One";

strStore[1] = "Two";

strStore[2] = "Three";

strStore[3] = "Four";

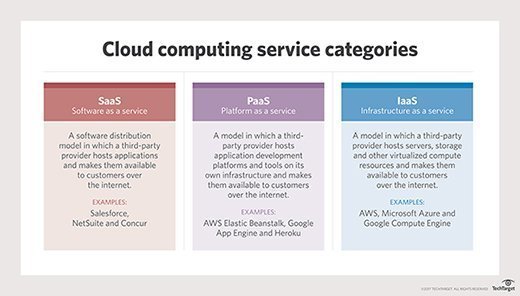
for(int i = 0; i < 10 ; i++)

Console.WriteLine(strStore[i]);

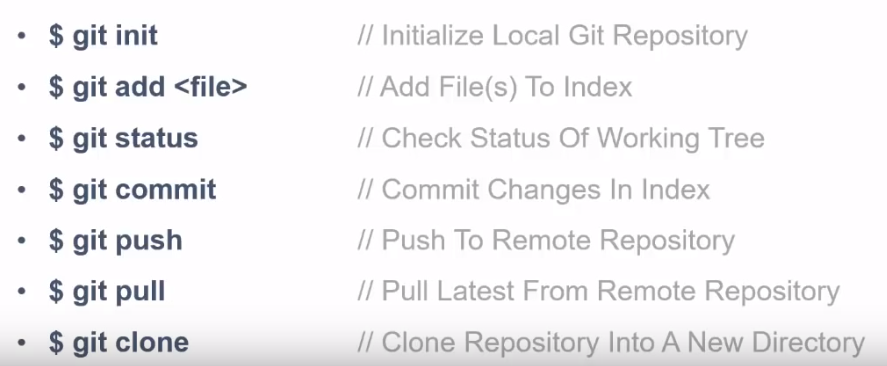
}

}

1. SaaS, PaaS and IaaS – Software Distribution Models



1. GIT Commands



Steps:

* Install git(including bash)
* Right click on the project folder and click on “Git Bash Here”
* Login to github and click create repository.([joydip.ch@gmail.com](mailto:joydip.ch@gmail.com) & Tiger@139139139)
* Issue **“git init”**
* Add username and email to git from git bash by

**git config –-global user.name ‘joydip chattoraj’**

**git config –-global user.email ‘joydip.ch@gmail.com’**

* Next issue the following to add all project files to local staging and then confirm if they got added to staging.

**git add .**

**git status**

* Next save the uploaded files from staging to final using

**git commit -m "first commit"**

* Next add the remote repository to local git using

**git remote add origin https://github.com/joydip-chattoraj/Joydip\_codes\_MEAN\_Stack.git**

* Finally push the files to the remote repository using

**git push -u origin master**

* To get files from remote repo to local use

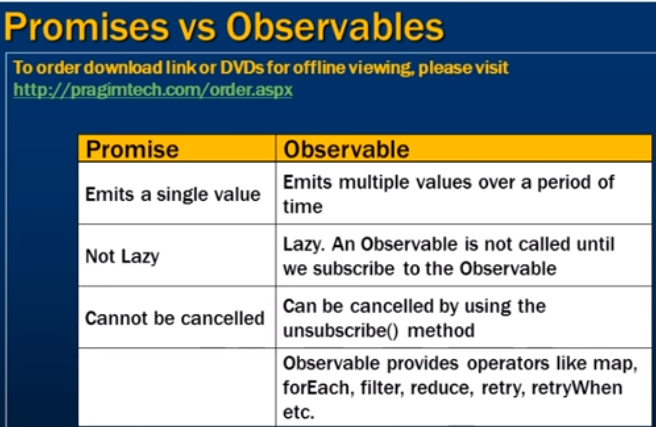
**git pull**

* To pull the whole project from remote repo, in bash go to the folder and issue

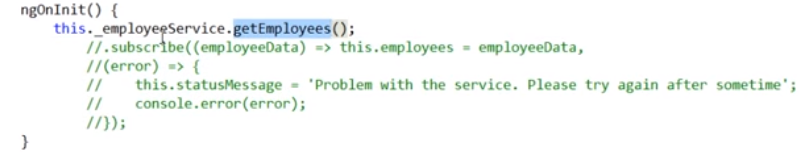
**git clone https://github.com/joydip-chattoraj/Joydip\_codes\_MEAN\_Stack.git**



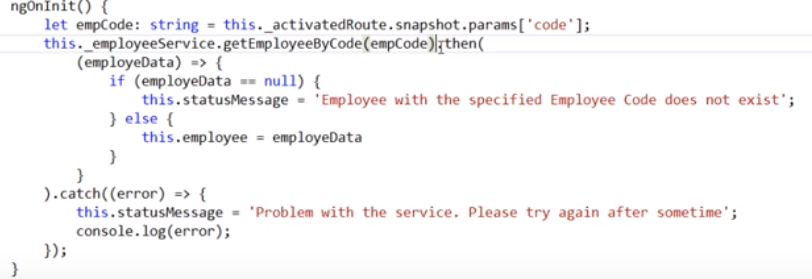
1. Promise vs Observables



Observable



Promise



1. **Explain the difference between "==" and "===" in Javascript?**

"==" checks only for equality in value whereas "===" is a stricter equality test and returns false if either the value or the type of the two variables are different.

1. Javascript **POP**() method:

var cloths = ["Shirt", "Pant", "TShirt"];

cloths.pop();

"TShirt" // this is popped and returned.

Cloths // this would return the items remaining in the array.

["Shirt", "Pant"]

1. Javascript **Closures**

Closure is a locally declared variable related to a function which stays in memory when the function has returned. To use a closure, define a function inside another function and expose it. To expose a function, return it or pass it to another function. The inner function will have access to the variables in the outer function scope, even after the outer function has returned.

function greeter(age) {

var message = " says howdy!! He is " + age + " years old";

return function greet(name) {

console.log(name+message);

};

}

// Generate the closure

var JamesGreeter = greeter(23);

// Use the closure

JamesGreeter("James");

1. **Call** Vs **apply** methods in JS

The function .call() and .apply() are very similar in their usage except a little difference. .call() is used when the number of the function's arguments are known to the programmer, as they have to be mentioned as arguments in the call statement. On the other hand, .apply() is used when the number is not known. The function .apply() expects the argument to be an array.

var someObject = {

myProperty : 'Foo',

myMethod : function(prefix, postfix) {

alert(prefix + this.myProperty + postfix);

}

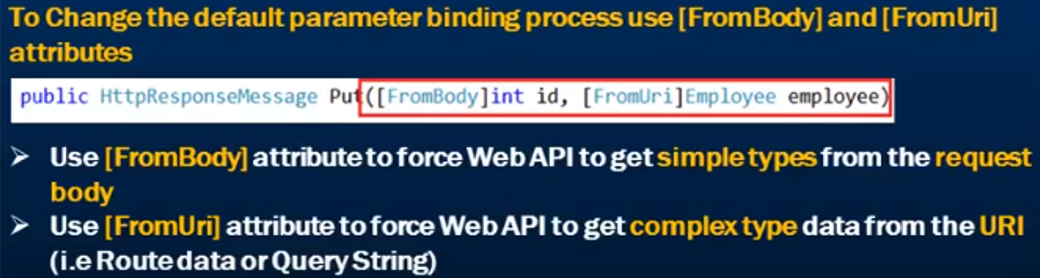
};

someObject.myMethod('<', '>'); // alerts '<Foo>' // **Direct calling**

someObject.myMethod.call(someObject, '<', '>'); // alerts '<Foo>' //**Calling with CALL**

someObject.myMethod.apply(someObject, ['<', '>']); //alerts '<Foo>'// **Calling with Apply**

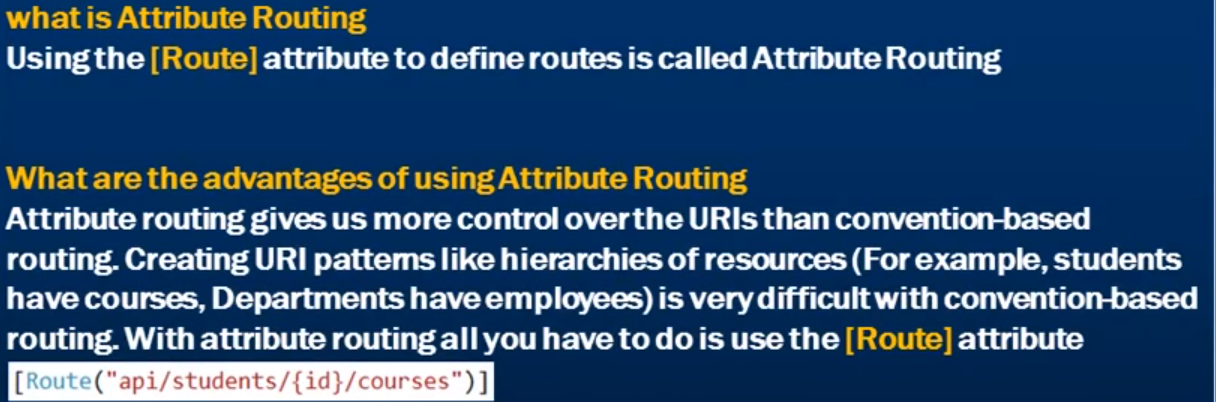
1. MVC FromBody and FromUri

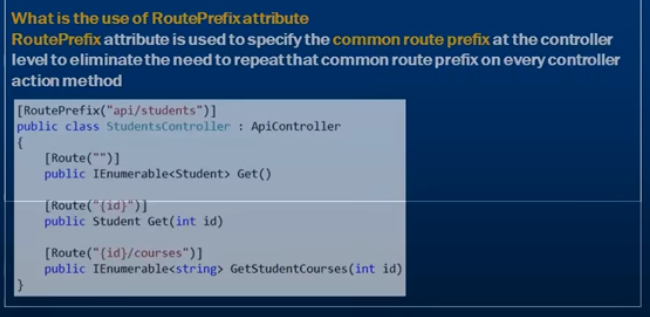


1. Ajax call to asp.net Web API



1. ATTRIBUTE ROUTING and route prefix IN WEB API





1. Output Caching in MVC

[**OutputCache**(Duration = 10, VaryByParam = "none")]

public ActionResult Profiles()

{

OutputCache label has a "Location" attribute and it is fully controllable. Its default value is "Any", however there are the [following locations](http://msdn.microsoft.com/en-us/library/system.web.ui.outputcachelocation(v=vs.100).aspx) available; as of now, we can use any one.

1. Any   2. Client   3. Downstream   4. Server   5. None   6. ServerAndClient

1. MVC lifecycle

**Creating Response object**: - The request object creation has four major steps.

**Step 1** **Fill route**: - MVC requests are mapped to route tables which in turn specify which controller and action to be invoked. So if the request is the first request the first thing is to fill the route table with routes collection. This filling of route table happens in the global.asax file.

**Step 2 Fetch route**:- Depending on the URL sent "UrlRoutingModule" searches the route table to create "RouteData" object which has the details of which controller and action to invoke.

**Step 3 Request context created**: - The "RouteData" object is used to create the "RequestContext" object.

**Step 4 Controller instance created**: - This request object is sent to "MvcHandler" instance to create the controller class instance. Once the controller class object is created it calls the "Execute" method of the controller class.

**Creating Response object**: - This phase has two steps executing the action and finally sending the response as a result to the view.

**Step 5 Execute Action**: - The "ControllerActionInvoker" determines which action to executed and executes the action.

**Step 6 Result sent**: - The action method executes and creates the type of result which can be a view result , file result , JSON result etc.

1. **MVC document page 11**





8. <https://www.youtube.com/watch?v=Nehk4tBxD4o&list=PLC3y8-rFHvwhBRAgFinJR8KHIrCdTkZcZ&index=23>