1. Installing ASP.NET MVC

What MVC version is already installed on my machine?

Control Panel=>Program and features

ASP.NET MVC3

<http://www.asp.net/mvc/mvc3>

ASP.NET MVC4

<http://www.asp.net/mvc/mvc4>

When we create new project, make sure you have .NET framework 4 selected in the drop down list.

If you have any other earlier.Net versions selected, then ASP.NETMVC3 web application and ASP.NETMVC4 web application templates will not be available.

2. What APS.NET MVC version

There are 2 ways to do this

1. At design time:- In solution explorer, expand “References” folder. Right click on System.Web.Mvc assembly and select “properties” and you can find the version that’s used as show below.
2. At runtime using code:-

return typeof(Controller).Assembly.GetName().Version.ToString();

3. Creating an MVC application.

http://localhost/MVCDemo/Home/Index

public class HomeController : Controller

{

//

// GET: /Home/

public ActionResult Index()

{

Return “hello from mvc application”;

}

}

In MVC URL’s are mapped to controller action methods. In web forms URL’s are mapped to physical files. Function in a controller is generally called as controller action Methods.

2. Creating MVC application

There are two ways to do this

1. At design time:- In the solution explorer, expand Preference” folder. Right click on system.Web.MVC assembly and select “properties” and you can find the version that’s
2. At runtime using code- typeof(Controller).Assembly.GetName().Version.ToString();

3.Controller in MVC

http://localhost/MVCDemo/Home/Index

public class HomeController : Controller

{

//

// GET: /Home/

public ActionResult Index()

{

Return “hello from mvc application”;

}

}

Q). How the URL mapped to the controller action method?

🡺ASP.Net routing, Notice that in Global.aspx we have Register Routes(). So it will mapped to controller .

protected void Application\_Start()

{

AreaRegistration.RegisterAllAreas();

RouteConfig.RegisterRoutes(RouteTable.Routes);

}

<http://localhost/MVCDemo/Home/Index/10?name=ganesh>

Asp.NET MVC will automatically pass any query string or form post parameter named “name” to index action method when it invoked

public string Index(string id,string Name)

{

return "Id=" + id + "Name" + Name;

//return "Hello from MVC application";

}

public string Index(string id)

{

return "the value of the id="+ id+ and name=+Request.QueryString[“name”];

}

Controller a

public class RouteConfig

{

public static void RegisterRoutes(RouteCollection routes)

{

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

routes.MapRoute(

name: "Default",

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

defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }

);

}

}

5.Views in MVC

*In controller:-*

public ViewResult Index()

{

ViewBag.Countries= new List<string>

{

"India",

"Nepal",

"UK",

"Canada"

};

return View();

}

*In Views:-*

@{

ViewBag.Title = "Countries List";

}

<h2>Countries List</h2>

<ul>

@foreach(string strCountry in ViewBag.Countries)

{

<li>@strCountry</li>

}

</ul>

ViewBag & ViewData is mechanism to pass data form controller to view.

Note:-To pass data from controller to a view. It’s always a good practice to user strongly type view model.

We use ‘@’ symbol to switch between html and C# code.

6.ViewData and ViewBag in MVC.

Both ViewData and ViewBag are used to pass data from a controller to view.ViewData is a dictionary of objects that are stored and retrieved using string as keys.

ViewData["YourKey"]="SomeData";

ViewBag.YourProperty="SomeData";

ViewBag:- uses the dynamic features that was introduced in to C#4.It allows an Object to have properties dynamically added to it.

Both ViewData & ViewBag does not provide compile time error checking. For Example, if you miss-speel keys or properties name, you wouldn’t get any compile error. You get to know about the error only at runtime. Internally ViewBag Properties are stored as name/value pairs in the ViwData Dictionary.

To pass the data form controller to view, It’s always a good practice to use strongly typed view models over ViewBag & ViewData. Strongly typed view models provide compile time error checking.

Examples:-

Controller:-

public ViewResult Index()

{

ViewData["Countries"] = new List<string>()

{

"India",

"US",

"UK",

"Canada"

};

return View();

}

View:-

@{

ViewBag.Title = "COuntries List";

}

<h2>Countries List</h2>

<ul>

@foreach(string strCountry in( List<string>) ViewData["Countries"])

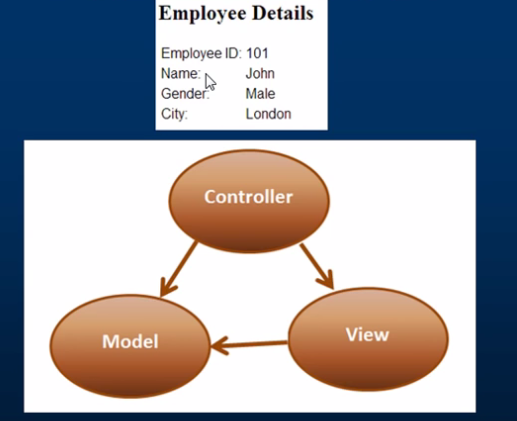
{

<li>@strCountry</li>

}

</ul>

7. Models in MVC application



Models in and MVC applications.-

We want to retrieve and Employee information from tblEmployee table and display as shown below

Controlller===>View,Model

View==>Model

The controller responds to URL request, gets data from a model and hands it over to the view.

The view then render the data. Model can entities or business objects.

***Model:-***

public class Employee

{

public int EmployeeId { get; set; }

public string Name { get; set; }

public string Gender { get; set; }

public string CIty { get; set; }

}

***Controller:-***

public ActionResult Details()

{

Employee employee = new Employee()

{

EmployeeId = 10,

Name = "John",

Gender = "Male",

CIty = "London"

};

return View(employee);

}

***Views:-***

@model \_7\_ModelsinMVC.Models.Employee

@{

ViewBag.Title = "Details";

}

<h2>Employee Details</h2>

<table style="font-family:Arial">

<tr>

<td>

<b>Employee Id</b>

</td>

<td>

@Model.EmployeeId

</td>

</tr>

<tr>

<td>

<b>Name:</b>

</td>

<td>

@Model.Name

</td>

</tr>

<tr>

<td>

<b>Gender:</b>

</td>

<td>

@Model.Gender

</td>

</tr>

<tr>

<td>

<b>City:</b>

</td>

<td>

@Model.CIty

</td>

</tr>

</table>

Part 8:-Data access in MVC using entity framework

The controller responds to URl request, gets data from a model and hands it to the view. The view then renders the data. Model can be entities or business objects.

In session 7, we have built Employee entity

public class Employee

{

public int EmployeeId { get; set; }

public string Name { get; set; }

public string Gender { get; set; }

public string CIty { get; set; }

}

In this video, we will discuss, retrieving data from a database from a database table tblEmployee using entity framework. In a later video, we will discuss using business objects as out model.

Step 1:- Install entity framework using nugget package manager

Step 2: Add EmployeeContext.cs file to the Models folder

Step 3: Add a connection string, to the web. config file, in the root directory.

Step4:-Map "Employee “model class to the database table, tblEmployee using "Table" attribute.

Step5:-Make the required changes to "Details()" action method in "EmployeeController"

public ActionResult Details( int id)

{

EmployeeContext employeeContext = new EmployeeContext();

Employee employee=employeeContext.Employees.Single(emp => emp.Id == id);

return View(employee);

}

Step 6:-Paste the folowing code in Application\_Start() function, in Global.aspx file. Existing

database do not need,database initializer so itcan be turned off.

Database.SetInitializer<\_8\_Data\_access\_in\_an\_MVC\_application.Models.EmployeeContext>(null);

That's it, run the application and navigate the URL's.Notice that the relevant employee details

are displayed as expected.

Part:-9:- Generate hyperlinks using action link html helper

***Controller***

***For employee list***

public ActionResult Index()

{

EmployeeContext employeeContext = new EmployeeContext();

List<Employee> employees = employeeContext.Employees.ToList();

return View(employees);

}

***Views:-***

<ul>

@foreach(Employee employee in @Model)

{

<li>

@Html.ActionLink(employee.Name, "Details", new { @id = employee.Id });

</li>

}

</ul>



***For employee details information***

***Controller***

public ActionResult Details(int id)

{

EmployeeContext employeeContext = new EmployeeContext();

Employee employee = employeeContext.Employees.Single(emp => emp.EmployeeId == id);

return View(employee);

}

***Views***

<h2>Employee Details</h2>

<table style="font-family:Arial">

<tr>

<td>

<b>Employee Id</b>

</td>

<td>

@Model.EmployeeId

</td>

</tr>

<tr>

<td>

<b>Name:</b>

</td>

<td>

@Model.Name

</td>

</tr>

<tr>

<td>

<b>Gender:</b>

</td>

<td>

@Model.Gender

</td>

</tr>

<tr>

<td>

<b>City:</b>

</td>

<td>

@Model.City

</td>

</tr>

</table>

@Html.ActionLink("Back to List","Index")



***For Model***

Employee.cs class

[Table("tblEmployeees")]

public class Employee

{

public int EmployeeId { get; set; }

public string Name { get; set; }

public string Gender { get; set; }

public string City { get; set; }

public DateTime dateOfBirth { get; set; }

}

EmployeeContext.cs class

public class EmployeeContext : DbContext

{

public DbSet<Employee> Employees { get; set; }

}

Part10:-Working with multiple tables in MVC