* The default MVC mapping is /[Controller]/[Action Name]/[Parameters]

public string Welcome(string name, int numTimes = 1) {

    return HttpUtility.HtmlEncode("Hello " + name + ", NumTimes is: " + numTimes);

* }---This code is used for the default parameters,
* Note\_\_\_\_: numTimes = 1 states that it is an optional parameter and if nothing is mentioned then by default it picks up 1

**The above is an example (Run it on explorer).**

**\_\_\_\_\_:Below is an example of routeconfig file where we are defining the mapping of routes**

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 }

    );

}

\_\_\_\_\_\_:Next is an example of modified route config file

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 }

      );

      routes.MapRoute(

           name: "Hello",

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

       );

   }

}

**\_\_\_\_\_Note\_\_\_\_\_The below is the modified set of code where we are defining the modified route**

routes.MapRoute(

           name: "Hello",

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

       );

**\_\_\_\_Note\_\_\_\_: The below set of code is added in order to return the view:**

public ActionResult Index()

{

return View();

}

\_\_\_\_Note\_\_\_\_\_:Whenever changes are made we need to see wether it is build or not

In the **Build** menu, select **Build Solution** (or Ctrl+Shift+B) to make sure the project is compiled

LINQ, short for **Language-Integrated Query**, is a technology built into the .NET framework which will allow you to easily query and manipulate data across various sources.

Var names= new List<string>();

Shortnames=for name in names where name.length<8 order by name.length select name;

Foreach(var name in shortnames)

Console.writeline(name)

Console.readline()

**Tutorial: Create the Web Application and Data Models for EF Database First with ASP.NET MVC**

Use the below link\_\_\_\_\_\_:::

<https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/database-first-development/creating-the-web-application>

# ASP.NET Core Startup Class and Program.cs

Global.asax is no more in ASP.NET Core application. Startup.cs file is a replacement of Global.asax file in ASP.NET Core.

Startup.cs file is entry point of application level it handle request pipeline. Startup class which triggers at first when application launches.

**Description.**

Now question is startup.cs file is mandatory or not? Yes, startup.cs is mandatory, it can be decorated with any access modifier like public, private, internal. multiple Startup classes are allowed in a single application. ASP.NET Core will select the appropriate class based on its Enviroment.

If a class **Startup{EnvironmentName}** exists, that class will be called for that **EnvironmentName**.

Should we need to define class name with startup.cs? No it is not necessary that class name should be Startup.

We can define two method in startup file like ConfigureServices and Configure.

# SMALL PROJECT\_\_\_\_\_\_\_:

# Two web apps one for students where they can upload the file after logging/Registering corresponding to professors name, in another web app for teachers can edit student details and check files will be shown record only of those students who registered corresponding to their name.

# Client and server side validation using data annotation

# Using jquery for client side validation.

# As per MSDN, the Data Annotations attributes cause MVC to provide both client and server validation checks with no additional coding required by you. The Data Annotations attributes can be used with the Entity Data Model (EDM), LINQ to SQL, and other data models.

# Note: the following is the syntax of file uploading in asp.net:

The basic syntax of File Upload is:

<asp:FileUpload ID= "Uploader" runat = "server" />

Note : scripts.render has the following functionalities:

@Scripts.Render("~/bundles/jquery")

# @Scripts.Render("~/bundles/jqueryval")

# Bundling is all about compressing several JavaScript or stylesheets files without any formatting (also referred as minified) into a single file for saving bandwith and number of requests to load a page.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace OnlinePortal.Controllers

{

public class RetreiveDataController : Controller

{

// GET: Home

public ActionResult Index()

{

List<SelectListItem> ProfList = GetProfessors();

return View(ProfList);

}

[HttpPost]

public ActionResult Index(string ddlCustomers)

{

List<SelectListItem> ProfList = GetProfessors();

if (!string.IsNullOrEmpty(ddlCustomers))

{

RedirectToAction("Students");

}

return View(ProfList);

}

private static List<SelectListItem> GetProfessors()

{

OnlinePortal.Models.ProfessorNameEntities entities = new OnlinePortal.Models.ProfessorNameEntities();

List<SelectListItem> ProfList = (from p in entities.Professor\_Credential.AsEnumerable()

select new SelectListItem

{

Text = p.Prof\_Name,

Value = p.Prof\_ID.ToString()

}).ToList();

//Add Default Item at First Position.

ProfList.Insert(0, new SelectListItem { Text = "--Professor Name--", Value = "" });

return ProfList;

}

}

}