**CS 555**

**HOMEWORK 4**

**Name:** Ivan Sangines Escrig

**ID#:** 968606

**Instructor:** Dr. Mahmood

**Date:** September 12, 2018

**TABLE OF CONTENT**

[**INTRODUCTION**](#_Toc446970371) 3

[**SCREENSHOTS:**](#_Toc446970373) 4

[**SOURCE CODE:**](#_Toc446970374) 22

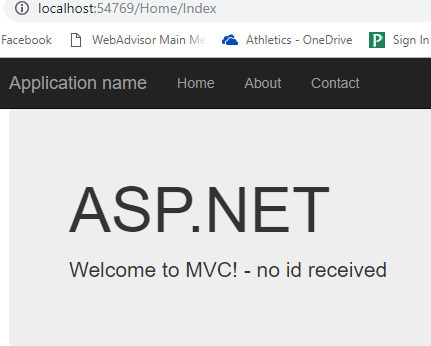
**[CONCLUSION:](#_Toc446970375)**53

**INTRODUCTION**

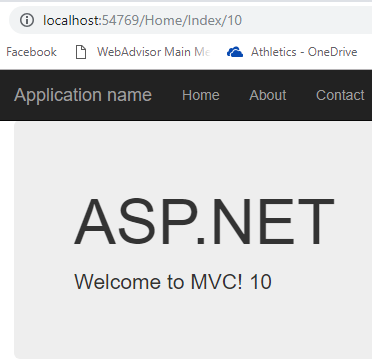
The objective of this assignment is to get familiar with MVC and all its features. By doing the first part of the handout for MVC, we will get to know and get familiar with route tables, understand better URLS. The second part of the handout will teach us how to use model binging in order to validate our intputs.

The last part of this assignment will help us understand better how session objects work and how to protect our different “stages” of an application, based on those session objects.

**SCREEN SHOTS:**

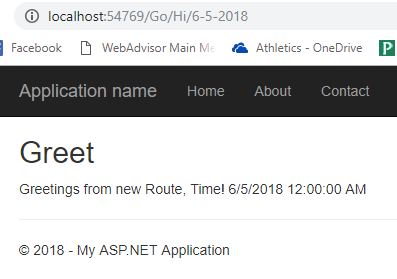
****

Same page but this time using a given ID. Using id=10:

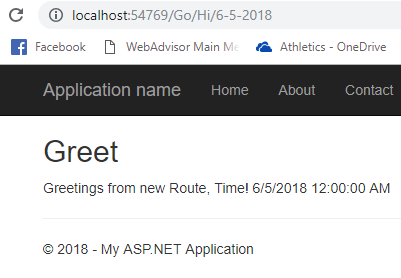
****

Using route called Greetings. We access to it by using the url:

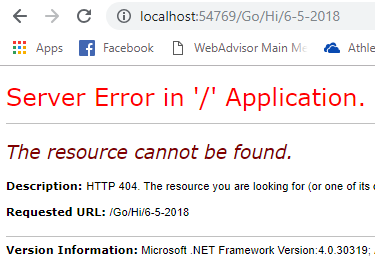
http://localhost:54769/**Go/Hi/6-5-2018**



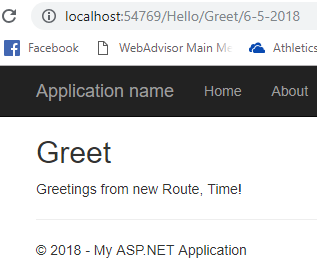
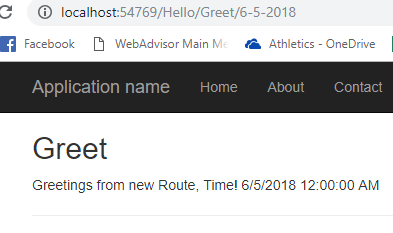
Same result using HelloController:



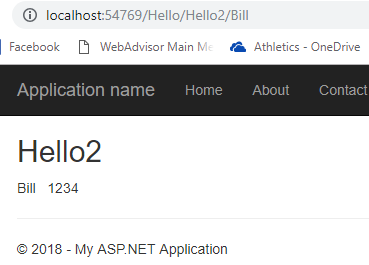
Error while reversing the order between Defaul and Greeting routings:



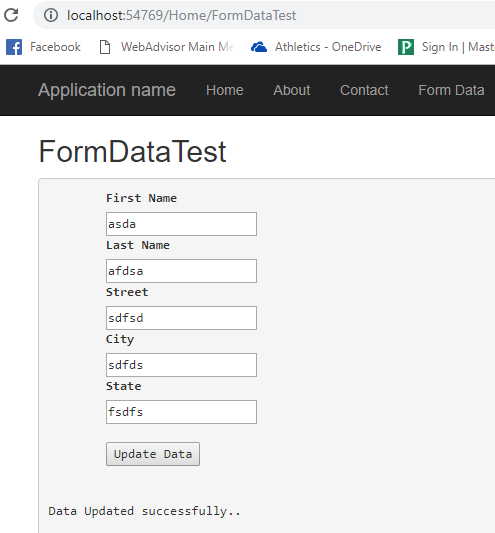
Using the proper url, we will get the same result as before but without the date passed. That is because the passing parameters are called differently, we need to name it the same.

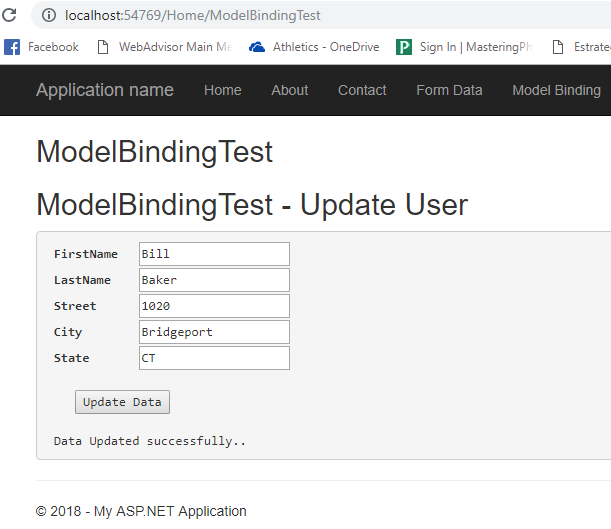
Output using a new action Hello2. Also, I have changed the parameter id in the default routing table for pname. The output is the same but the passing value for Hello2 action needed to be changed as pname as well:



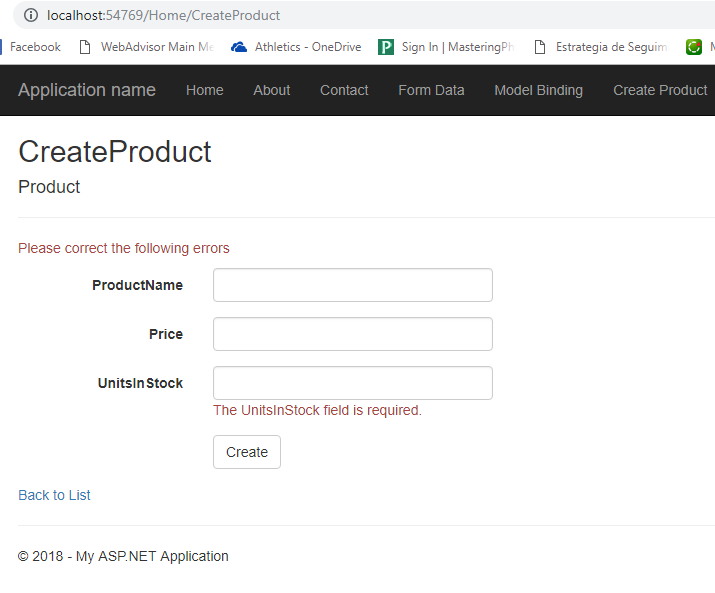
In this output we can see the DataForm from the output. It is created as a new action in the HomeController. That is why the url appears as …./Home/FormDataTest. If the data is correctly “written” to the DB a msg will appear at the bottom of the page.

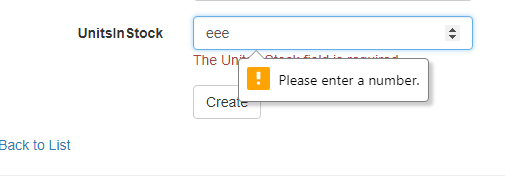


In this output we can se almost the same page as in the previous example. However, this time we used model binding which means that we are communitating with the view by passing an object of a class and not X number of parameters.

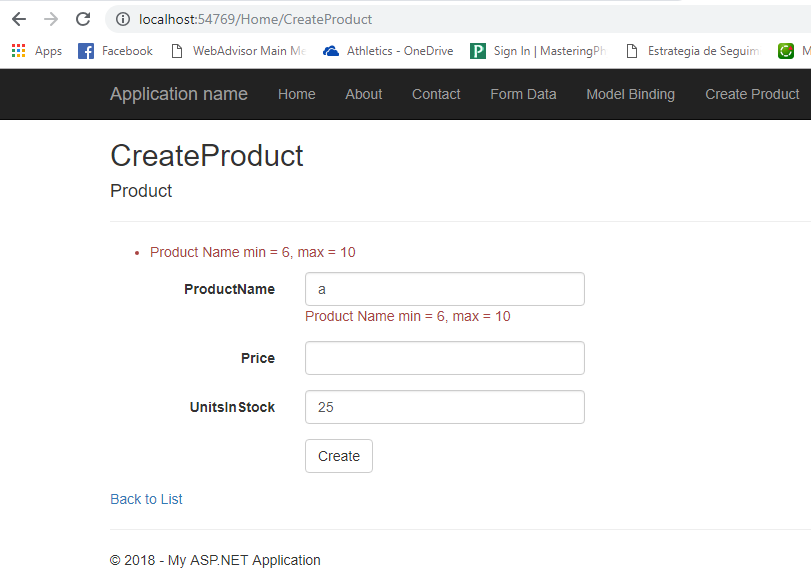


In the following example we are able to see how validation works. Whenever we try to leave the fields empty (or try to send a wrong input for the fields), we will get a message asking to modify the information provided.

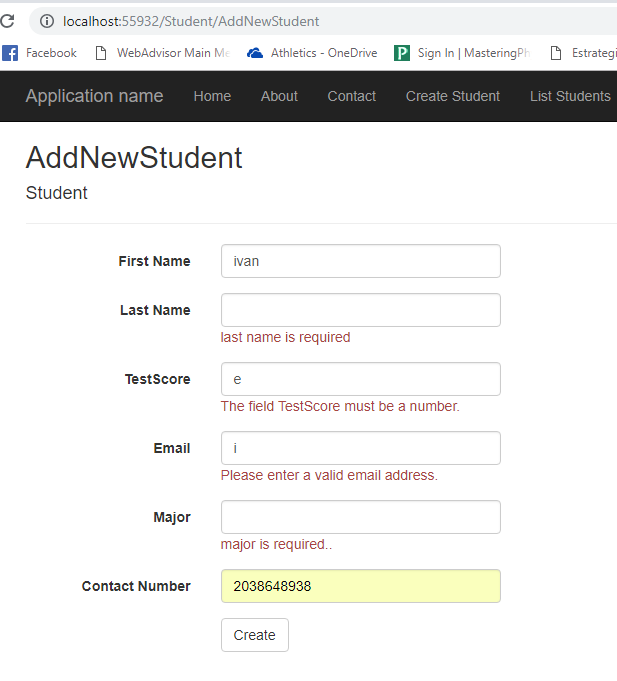




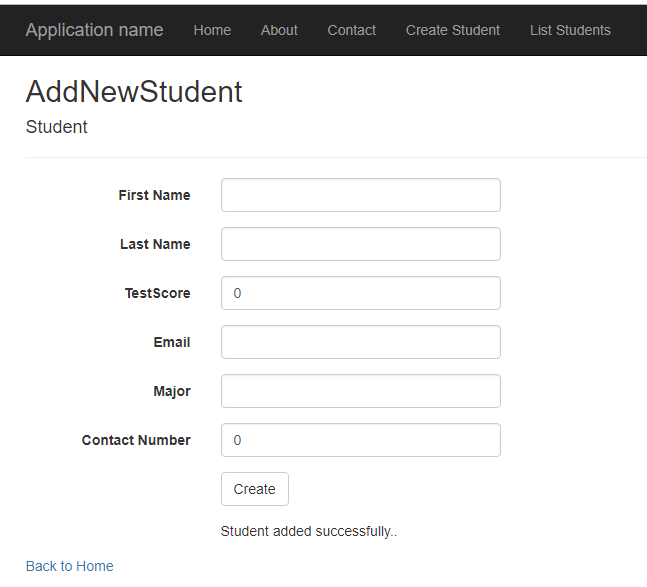
Customizing validation of our data:



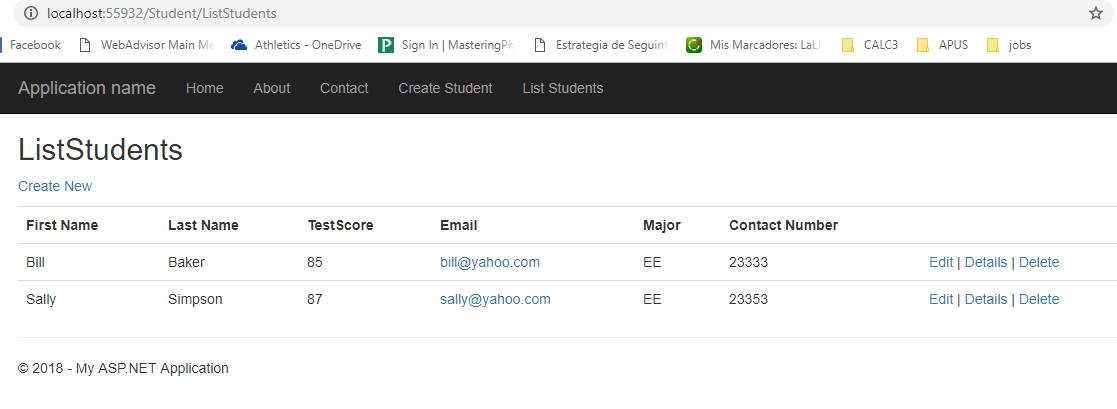
In the following output, we can an example of model validation and data notation. Each field in the form has a particular requirement previously designed in the class student (in the Models folder).



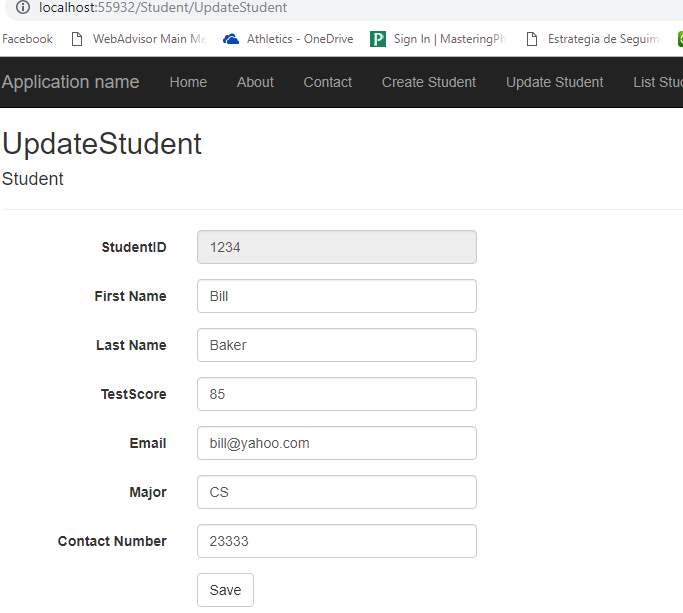
Whenever all the fields are correctly filled and we click button create, the form will clear up and a successful msg will appear.

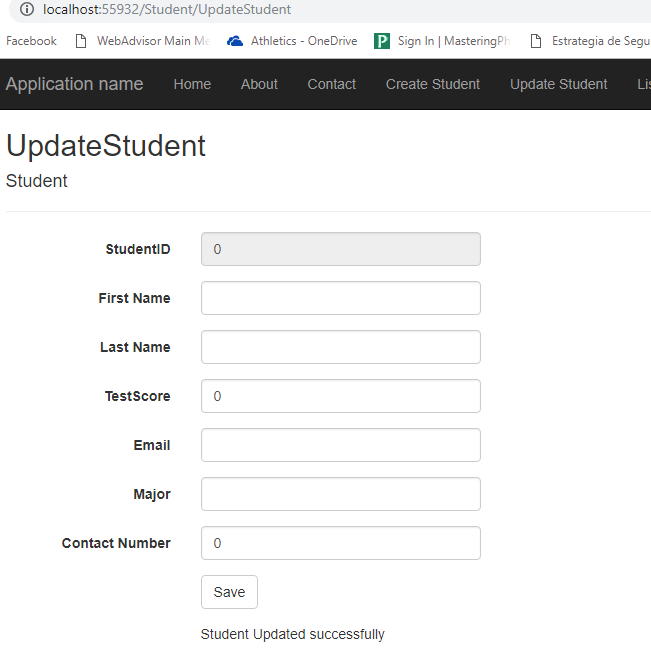


Listing the students:

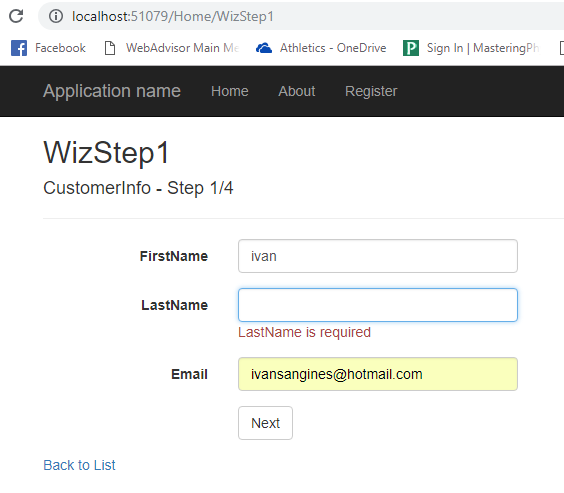


As last part of this assignment, I modified the Update Student page. The objective is to simulate an update of a student. Since we do not have DB and the students are already created automatically, I had to pass the id number as a default value on the UpdateStudent Action (using 1234, Bill id). Once the id is passed by default in the action, the information of the student with that id will show up. Once we modify the information we want, we click save and a successful message will show up, at the same time the page will be cleared up.

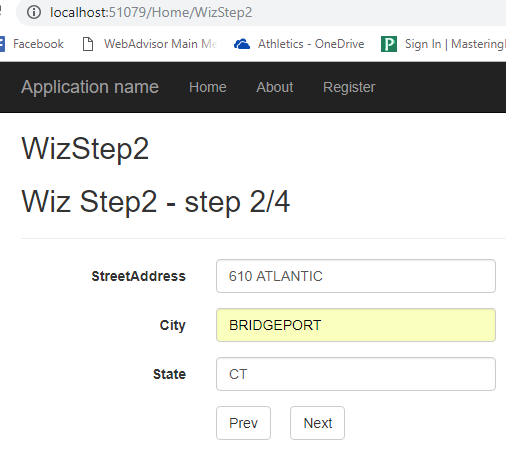




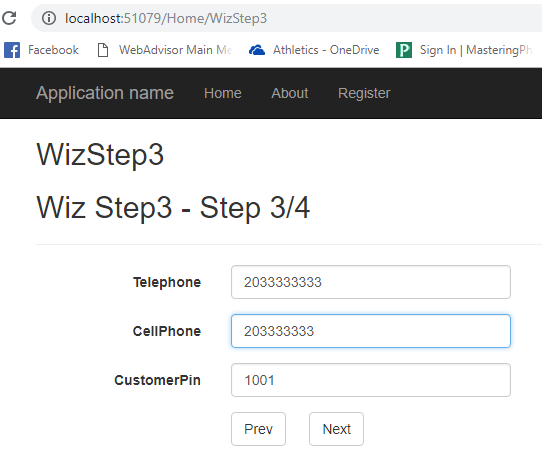
In the following output, we can see the first step of our wizzar web app. As we can see is using model validation and just asking for the first three fields of the costumer:



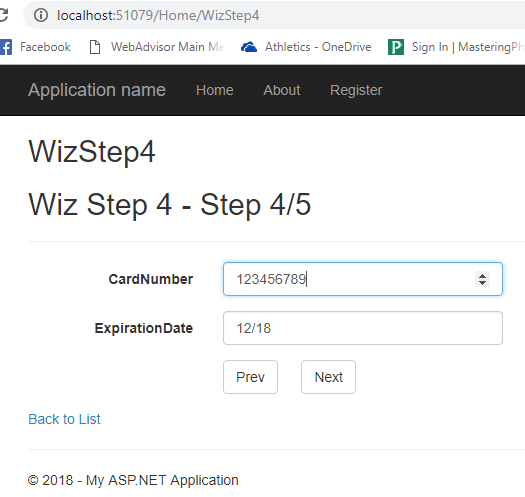
Once we fill up the first step, with proper data, and click next, the step 2 page will show up. This new page has two buttons, if we click on the PREV button, we will go back to the first step and the data entered will show up. If we click on the NEXT button, the step 3 page will show up:



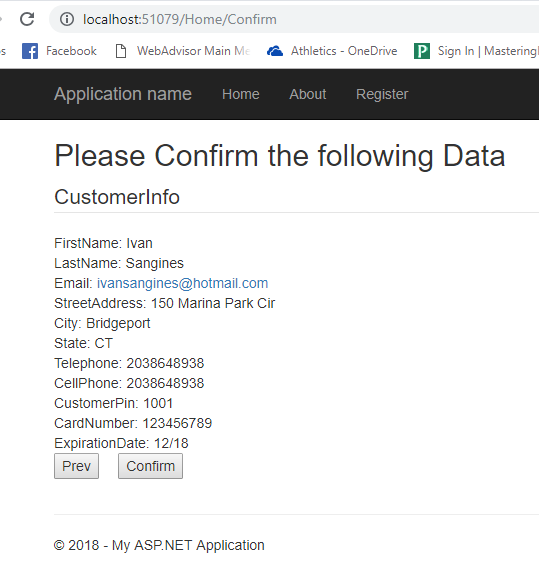
Third stage, same as before, if we click on the PREV button we will navigate to the previous page showing the data previously entered. If we click to the NEXT button and our data is correctly filled up, the last stage page with all the data entered will show up:



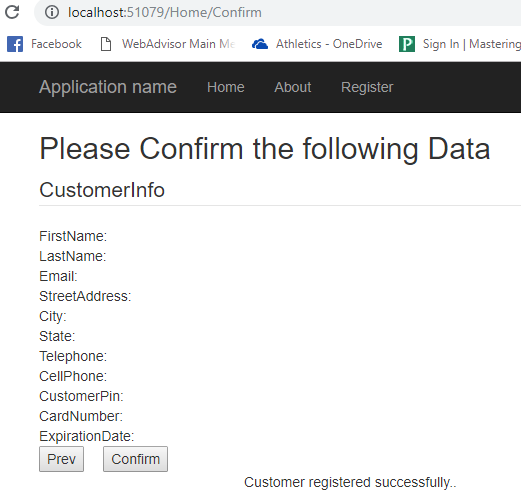
This is the fourth step where we provide the card number and expiration date. I made the card number to type creditcard and the date to be a date time with format MM/yy:



Last stage where we can check our data and go back with the PREV button if there is some mistake or confirm if everything is right:



After getting the message for successfully registration, I set all the objects in each state to null and clear the confirmation page. Once we click confirm (and we successfully regirster the costumer) if we click to the PREV button, we will be redirected to step 1.



**SOURCE CODE:**

**Part 1:**

**Routes.config**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using System.Web.Routing;

namespace MVCRoutes

{

public class RouteConfig

{

public static void RegisterRoutes(RouteCollection routes)

{

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

routes.MapRoute(

name: "Default",

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

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

);

routes.MapRoute(

name: "Greeting", // Route name

url: "Go/Hi/{greetDate}", // greetDate=paramter-name

defaults: new { controller = "Hello", action = "Greet", greetDate = UrlParameter.Optional }

);

}

}

}

**Produt.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace MVCRoutes.Models

{

public class Product

{

public int ProductId { get; set; }

[ProdName(MinimLength = 6, MaximLength = 10, ErrorMessage = "Product Name min = 6, max = 10")]

public string ProductName { get; set; }

public decimal Price { get; set; }

public int UnitsInStock { get; set; }

}

}

**ProdNameAtribute.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Web;

namespace MVCRoutes.Models

{

public class ProdNameAttribute : ValidationAttribute

{

public int MinimLength { get; set; }

public int MaximLength { get; set; }

public override bool IsValid(object value)

{

if (value == null)

{

return false;

}

var nm = (string)value;

if ((nm.Length < MinimLength) || (nm.Length > MaximLength))

{

return false;

}

return true;

}

}

}

**HelloPerson.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace MVCRoutes.Models

{

public class HelloPerson

{

public string Name { get; set; }

public int Id { get; set; }

}

}

**Address.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace MVCRoutes.Models

{

public class Address

{

public string Street { get; set; }

public string City { get; set; }

public string State { get; set; }

}

}

**HomeController.cs:**

using MVCRoutes.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace MVCRoutes.Controllers

{

public class HomeController : Controller

{

public ActionResult Index(int? id)

{

if (id != null)

ViewBag.Message = "Welcome to MVC! " + id.ToString();

else

ViewBag.Message = "Welcome to MVC! - no id received";

return View();

}

public ActionResult About()

{

ViewBag.Message = "Your application description page.";

return View();

}

public ActionResult Contact()

{

ViewBag.Message = "Your contact page.";

return View();

}

// parameter names have to match the names declared in the View

public ActionResult FormDataTest(string fname, string lname, string street, string city, string state)

{

if ((fname != null) && (lname != null) && (street != null) && (city != null) && (state != null))

{

Address a1 = new Address

{

Street = street,

State = state,

City = city

};

UserInfo ui = new UserInfo

{

FirstName = fname,

LastName = lname,

UserAddress = a1

};

if (DBLayer.WriteUserToDB(ui))

ViewBag.Message = "Data Updated successfully..";

}

return View("FormDataTest");

}

// triggered when the page is first asked for

public ActionResult ModelBindingTest()

{

Address a1 = new Address

{

City = "Bridgeport",

State = "CT",

Street = "1020"

};

UserInfo ui = new UserInfo

{

FirstName = "Bill",

LastName = "Baker",

UserAddress = a1

};

return View("ModelBindingTest", ui);

}

[HttpPost] // triggered when page is posted back

public ActionResult ModelBindingTest(UserInfo ui)

{

if (DBLayer.WriteUserToDB(ui))

ViewBag.Message = "Data Updated successfully..";

return View("ModelBindingTest", ui);

}

public ActionResult CreateProduct()

{

return View("CreateProduct");

}

[HttpPost]

public ActionResult CreateProduct([Bind(Exclude = "Price")] Product pr)

{ // validation will be excluded for Price field

if (ModelState.IsValid)

{

// DBLayer.CreateProduct(pr); to do

ViewData["message"] = "Product ceated successfully..";

}

return View("CreateProduct");

}

}

}

**HelloController.cs:**

using MVCRoutes.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace MVCRoutes.Controllers

{

public class HelloController : Controller

{

// GET: Hello

public ActionResult Index()

{

return View();

}

public ActionResult Greet(DateTime? id)

{

ViewBag.GreetInfo = "Greetings from new Route, Time! " + id.ToString();

return View();

}

public ActionResult Hello2(string pname) //string theName)

// the parameter has to be id

{ // to use default route in RoutesConfig.cs

HelloPerson person = new HelloPerson();

person.Name = pname; //changed id for pname in the default routing table

person.Id = 1234;

ViewBag.PersonData = person;

return View();

}

}

}

**Global.asax**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using System.Web.Optimization;

using System.Web.Routing;

namespace MVCRoutes

{

public class MvcApplication : System.Web.HttpApplication

{

protected void Application\_Start()

{

AreaRegistration.RegisterAllAreas();

FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);

RouteConfig.RegisterRoutes(RouteTable.Routes);

BundleConfig.RegisterBundles(BundleTable.Bundles);

}

}

}

**\_Layout.cshtml:**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>@ViewBag.Title - My ASP.NET Application</title>

@Styles.Render("~/Content/css")

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

</head>

<body>

<div class="navbar navbar-inverse navbar-fixed-top">

<div class="container">

<div class="navbar-header">

<button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

@Html.ActionLink("Application name", "Index", "Home", new { area = "" }, new { @class = "navbar-brand" })

</div>

<div class="navbar-collapse collapse">

<ul class="nav navbar-nav">

<li>@Html.ActionLink("Home", "Index", "Home")</li>

<li>@Html.ActionLink("About", "About", "Home")</li>

<li>@Html.ActionLink("Contact", "Contact", "Home")</li>

<li>@Html.ActionLink("Form Data", "FormDataTest", "Home")</li>

<li>@Html.ActionLink("Model Binding", "ModelBindingTest", "Home")</li>

<li>@Html.ActionLink("Create Product", "CreateProduct", "Home")</li>

</ul>

</div>

</div>

</div>

<div class="container body-content">

@RenderBody()

<hr />

<footer>

<p>&copy; @DateTime.Now.Year - My ASP.NET Application</p>

</footer>

</div>

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

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

@RenderSection("scripts", required: false)

</body>

</html>

**ModelBindingTest.cshtml:**

@model MVCRoutes.Models.UserInfo

@{

ViewBag.Title = "ModelBindingTest";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h2>ModelBindingTest</h2>

<h2>ModelBindingTest - Update User </h2>

@using (Html.BeginForm())

{

<pre>

@Html.LabelFor(model => model.FirstName) @Html.TextBoxFor(model => model.FirstName)

@Html.LabelFor(model => model.LastName) @Html.TextBoxFor(model => model.LastName)

@Html.LabelFor(model => model.UserAddress.Street) @Html.TextBoxFor(model => model.UserAddress.Street)

@Html.LabelFor(model => model.UserAddress.City) @Html.TextBoxFor(model => model.UserAddress.City)

@Html.LabelFor(model => model.UserAddress.State) @Html.TextBoxFor(model => model.UserAddress.State)

<input type="submit" value="Update Data" name="btnUpdate" />

@ViewBag.Message

</pre>

}

**FormDataTest.cshtml:**

@{

ViewBag.Title = "FormDataTest";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h2>FormDataTest</h2>

@using (Html.BeginForm())

{

<pre>

@Html.Label("fname", "First Name", new { @class = "myLabel" })

@Html.TextBox("fname", "", new { @class = "myText" })

@Html.Label("lname", "Last Name", new { @class = "myLabel" })

@Html.TextBox("lname", "", new { @class = "myText" })

@Html.Label("street", "Street", new { @class = "myLabel" })

@Html.TextBox("street", "", new { @class = "myText" })

@Html.Label("city", "City", new { @class = "myLabel" })

@Html.TextBox("city", "", new { @class = "myText" })

@Html.Label("state", "State", new { @class = "myLabel" })

@Html.TextBox("state", "", new { @class = "myText" })

<input type="submit" value="Update Data" name="btnUpdate" />

<p>@ViewBag.Message</p>

</pre>

}

**Greet.cshtml:**

@{

ViewBag.Title = "Greet";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h2>Greet</h2>

<p>@ViewBag.GreetInfo</p>

**Hello2.cshtml:**

@{

ViewBag.Title = "Hello2";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h2>Hello2</h2>

@ViewBag.PersonData.Name &nbsp;&nbsp;@ViewBag.PersonData.Id

**Part 2:**

**StudentController.cs:**

using ModelValidationDataAnnotations.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace ModelValidationDataAnnotations.Controllers

{

public class StudentController : Controller

{

// GET: Student

public ActionResult Index()

{

return View();

}

public ActionResult AddNewStudent()

{

ViewBag.Message = "";

Student st = new Student();

return View(st);

}

[HttpPost]

public ActionResult AddNewStudent(Student st)

{

ViewBag.Message = "";

if (ModelState.IsValid)

{

// write st to DB – to do

ViewBag.Message = "Student added successfully..";

ModelState.Clear(); // to clear the form text fields

return View(new Student());

}

else

{

ViewBag.Message = "Invalid entry.."; //Never showing it?

return View(st);

}

}

public ActionResult ListStudents()

{

ViewBag.Message = "";

// Get Students from DB

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

Student s1 = new Student

{

FirstName = "Bill",

LastName = "Baker",

StudentID = 1234,

Major = "EE",

Email = "bill@yahoo.com",

Telephone = 23333,

TestScore = 85

};

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

{

if ((int)(TempData["StudentID"]) != 1234)

STList.Add(s1);

}

else

STList.Add(s1);

Student s2 = new Student

{

FirstName = "Sally",

LastName = "Simpson",

StudentID = 1235,

Major = "EE",

Email = "sally@yahoo.com",

Telephone = 23353,

TestScore = 87

};

STList.Add(s2);

return View(STList);

}

public ActionResult UpdateStudent(int id=1234)

{

ViewBag.Message = "";

Student st = null;

if (id == 1234)

{

st = new Student

{

FirstName = "Bill",

LastName = "Baker",

StudentID = 1234,

Major = "CS",

Email = "bill@yahoo.com",

Telephone = 23333,

TestScore = 85

};

}

if (id == 1235)

{

st = new Student

{

FirstName = "Sally",

LastName = "Simpson",

StudentID = 1235,

Major = "EE",

Email = "sally@yahoo.com",

Telephone = 23353,

TestScore = 87

};

}

return View(st);

}

[HttpPost]

public ActionResult UpdateStudent(Student st)

{

ViewBag.Message = "";

if (ModelState.IsValid)

{

// update student in DB

ModelState.Clear();

ViewBag.Message = "Student Updated successfully";

}

return View(new Student());

}

public ActionResult DeleteStudent(int id)

{

ViewBag.Message = "";

// Delete student in DB

TempData["StudentID"] = id;

ViewBag.Message = "Student Updated successfully";

return RedirectToAction("ListStudents");

}

}

}

**Student.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Web;

namespace ModelValidationDataAnnotations.Models

{

public class Student

{

[ScaffoldColumn(false)] // will not generate UI element

public int StudentID { get; set; }

[DataType(DataType.Text)]

[Required(ErrorMessage = "first name is required"), MaxLength(40)]

[Display(Name = "First Name")]

public string FirstName { get; set; }

[DataType(DataType.Text)]

[Required(ErrorMessage = "last name is required"), MaxLength(50)]

[Display(Name = "Last Name")]

public string LastName { get; set; }

[Range(0, 100, ErrorMessage = "Invalid score..")]

[Required(ErrorMessage = "test score is required")]

public int TestScore { get; set; }

[DataType(DataType.EmailAddress)]

[Required(ErrorMessage = "Email is required")]

[RegularExpression(@"^\w+([-+.']\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\*$", ErrorMessage = "invalid email")]

public string Email { get; set; }

[Required(ErrorMessage = "major is required..")]

[StringLength(10, ErrorMessage = "should be between 2 and 10 characters",MinimumLength =2)]

public string Major { get; set; }

[Required(ErrorMessage = "Telephone is required..")]

[Display(Name = "Contact Number")]

[DataType(DataType.PhoneNumber)]

public int Telephone { get; set; }

}

}

**AddNewStudent.cshtml:**

@model ModelValidationDataAnnotations.Models.Student

@{

ViewBag.Title = "AddNewStudent";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

@section MyJavascriptSection {

<script type="text/javascript">

$(function () {

$("#btnSubmit").click(function () {

$("#lblStatus").html("");

});

});

</script>

}

<h2>AddNewStudent</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<h4>Student</h4>

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

<div class="form-group">

@Html.LabelFor(model => model.FirstName, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.FirstName, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.FirstName, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.LastName, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.LastName, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.LastName, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.TestScore, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.TestScore, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.TestScore, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Email, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Email, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Email, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Major, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Major, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Major, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Telephone, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Telephone, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Telephone, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" id="btnSubmit" value="Create" class="btn btn-default" />

</div>

</div>

<div class="form-group">

<div id="lblStatus" class="col-md-offset-2 col-md-10">

@ViewBag.Message

</div>

</div>

</div>

}

<div>

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

</div>

@section Scripts {

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

}

**ListStudent.cshtml:**

@model IEnumerable<ModelValidationDataAnnotations.Models.Student>

@{

ViewBag.Title = "ListStudents";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h2>ListStudents</h2>

<p>

@Html.ActionLink("Create New", "Create")

</p>

<table class="table">

<tr>

<th>

@Html.DisplayNameFor(model => model.FirstName)

</th>

<th>

@Html.DisplayNameFor(model => model.LastName)

</th>

<th>

@Html.DisplayNameFor(model => model.TestScore)

</th>

<th>

@Html.DisplayNameFor(model => model.Email)

</th>

<th>

@Html.DisplayNameFor(model => model.Major)

</th>

<th>

@Html.DisplayNameFor(model => model.Telephone)

</th>

<th></th>

</tr>

@foreach (var item in Model) {

<tr>

<td>

@Html.DisplayFor(modelItem => item.FirstName)

</td>

<td>

@Html.DisplayFor(modelItem => item.LastName)

</td>

<td>

@Html.DisplayFor(modelItem => item.TestScore)

</td>

<td>

@Html.DisplayFor(modelItem => item.Email)

</td>

<td>

@Html.DisplayFor(modelItem => item.Major)

</td>

<td>

@Html.DisplayFor(modelItem => item.Telephone)

</td>

<td>

@Html.ActionLink("Edit", "UpdateStudent", new { id=item.StudentID }) |

@Html.ActionLink("Details", "StudentDetail", new { id=item.StudentID }) |

@Html.ActionLink("Delete", "DeleteStudent", new { id=item.StudentID })

</td>

</tr>

}

</table>

**UpdateStudent.cshtml:**

@model ModelValidationDataAnnotations.Models.Student

@{

ViewBag.Title = "UpdateStudent";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h2>UpdateStudent</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<h4>Student</h4>

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

@\*Html.HiddenFor(model => model.StudentID,"")\*@

<div class="form-group">

@Html.LabelFor(model => model.StudentID, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.StudentID, new { htmlAttributes = new { @class = "form-control", @readOnly = true } })

@Html.ValidationMessageFor(model => model.StudentID, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.FirstName, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.FirstName, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.FirstName, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.LastName, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.LastName, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.LastName, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.TestScore, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.TestScore, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.TestScore, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Email, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Email, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Email, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Major, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Major, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Major, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Telephone, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Telephone, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Telephone, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Save" class="btn btn-default" />

</div>

</div>

<div class="form-group">

<div id="lblStatus" class="col-md-offset-2 col-md-10">

@ViewBag.Message

</div>

</div>

</div>

}

<div>

@Html.ActionLink("Back to Main", "ListStudents")

</div>

@section Scripts {

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

}

**Part 3:**

**HomeController.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using WizardMVC.Models;

using WizardMVC.Utils;

namespace WizardMVC.Controllers

{

public class HomeController : Controller

{

public ActionResult Index()

{

return View();

}

public ActionResult About()

{

ViewBag.Message = "Your application description page.";

return View();

}

public ActionResult Contact()

{

ViewBag.Message = "Your contact page.";

return View();

}

public ActionResult WizStep1()

{

// user can come to this step from two different ways

// 1. starting the wizard, 2. user came back from step 2

CustomerInfo cinfo = null;

if (SessionFacade.WIZSTEP1 == null) // starting from beginning

cinfo = new CustomerInfo();

else

cinfo = SessionFacade.WIZSTEP1; // user is coming back from step 2

return View(cinfo);

}

[HttpPost]

public ActionResult WizStep1(CustomerInfo cinfo)

{

ModelState.Remove("StreetAddress"); //Removing from where??

ModelState.Remove("City");

ModelState.Remove("State");

ModelState.Remove("Telephone");

ModelState.Remove("CellPhone");

ModelState.Remove("CustomerPin");

ModelState.Remove("CardNumber");

ModelState.Remove("ExpirationDate");

if (ModelState.IsValid)

{

SessionFacade.WIZSTEP1 = cinfo; //When is cinfo created with the info provided? Obtaining it from view?

return RedirectToAction("WizStep2");

}

return View(cinfo);

}

public ActionResult WizStep2()

{

// some one can come to this step three different ways

// 1. directly to this step without going through step 1

// 2. From step1 to this step 2

// 3. Back from step 3

if (SessionFacade.WIZSTEP1 == null) // some one skipped step 1

return RedirectToAction("WizStep1");

else

{

CustomerInfo ci = null;

if (SessionFacade.WIZSTEP2 != null) // coming back from step 3

ci = SessionFacade.WIZSTEP2;

else

ci = SessionFacade.WIZSTEP1; // from step 1 to this step2

return View(ci);

}

}

[HttpPost]

public ActionResult WizStep2(string btnPrev, string btnNext, CustomerInfo cinfo)

{

if (btnPrev != null)

{

if (btnPrev.ToUpper() == "PREV")

{

SessionFacade.WIZSTEP2 = null; // remove data for step 2

return RedirectToAction("WizStep1");

}

}

if (SessionFacade.WIZSTEP1 == null)

return RedirectToAction("WizStep1");

else

{

CustomerInfo ci = SessionFacade.WIZSTEP1; //new object?

ModelState.Remove("FirstName");

ModelState.Remove("LastName");

ModelState.Remove("Email");

ModelState.Remove("Telephone");

ModelState.Remove("CellPhone");

ModelState.Remove("CustomerPin");

ModelState.Remove("CardNumber");

ModelState.Remove("ExpirationDate");

if (ModelState.IsValid)

{

if (btnNext != null)

{

if (btnNext.ToUpper() == "NEXT")

{

ci.StreetAddress = cinfo.StreetAddress;

ci.City = cinfo.City;

ci.State = cinfo.State;

SessionFacade.WIZSTEP2 = ci;

return RedirectToAction("WizStep3");

}

}

}

}

return View(cinfo);

}

public ActionResult WizStep3()

{

// 3 ways some one can come to this step

if (SessionFacade.WIZSTEP2 == null)

return RedirectToAction("WizStep2");

else

{

CustomerInfo ci = null;

if (SessionFacade.WIZSTEP3 != null) // from step 4

ci = SessionFacade.WIZSTEP3;

else

ci = SessionFacade.WIZSTEP2; //from step 2

return View(ci);

}

}

[HttpPost]

public ActionResult WizStep3(string btnPrev, string btnNext, CustomerInfo cinfo)

{

if (btnPrev != null)

{

if (btnPrev.ToUpper() == "PREV")

{

SessionFacade.WIZSTEP3 = null; // remove data for step 2

return RedirectToAction("WizStep2");

}

}

if (SessionFacade.WIZSTEP2 == null)

return RedirectToAction("WizStep2");

else

{ // normal flow from step 2 to step 3

CustomerInfo ci = SessionFacade.WIZSTEP2;

ModelState.Remove("FirstName");

ModelState.Remove("LastName");

ModelState.Remove("Email");

ModelState.Remove("StreetAddress");

ModelState.Remove("City");

ModelState.Remove("State");

ModelState.Remove("CardNumber");

ModelState.Remove("ExpirationDate");

if (ModelState.IsValid)

{

if (btnNext != null)

{

if (btnNext.ToUpper() == "NEXT")

{

ci.Telephone = cinfo.Telephone;

ci.CellPhone = cinfo.CellPhone;

ci.CustomerPin = cinfo.CustomerPin;

SessionFacade.WIZSTEP3 = ci;

return RedirectToAction("WizStep4");

}

}

}

}

return View(cinfo);

}

public ActionResult WizStep4()

{

if (SessionFacade.WIZSTEP3 == null)

return RedirectToAction("WizStep3");

else

{

CustomerInfo ci = null;

if (SessionFacade.WIZSTEP4 != null) //coming back from confirm

ci = SessionFacade.WIZSTEP4;

else

ci = SessionFacade.WIZSTEP3; //coming from step 3

return View(ci);

}

}

[HttpPost]

public ActionResult WizStep4(string btnPrev, string btnNext, CustomerInfo cinfo)

{

if (btnPrev != null)

{

if (btnPrev.ToUpper() == "PREV")

{

SessionFacade.WIZSTEP4 = null; // remove data for step 3

return RedirectToAction("WizStep3");

}

}

if (SessionFacade.WIZSTEP3 == null)

return RedirectToAction("WizStep3");

else

{ // normal flow from step 3 to step 4

CustomerInfo ci = SessionFacade.WIZSTEP3;

ModelState.Remove("FirstName");

ModelState.Remove("LastName");

ModelState.Remove("Email");

ModelState.Remove("StreetAddress");

ModelState.Remove("City");

ModelState.Remove("State");

ModelState.Remove("Telephone");

ModelState.Remove("CellPhone");

ModelState.Remove("CustomerPin");

if (ModelState.IsValid)

{

if (btnNext != null)

{

if (btnNext.ToUpper() == "NEXT")

{

ci.CardNumber = cinfo.CardNumber;

ci.ExpirationDate = cinfo.ExpirationDate;

SessionFacade.WIZSTEP4 = ci;

return RedirectToAction("Confirm");

}

}

}

}

return View(cinfo);

}

public ActionResult Confirm()

{

if (SessionFacade.WIZSTEP4 == null)

return RedirectToAction("WizStep4");

else

{

CustomerInfo ci = SessionFacade.WIZSTEP4;

return View(ci);

}

}

[HttpPost]

public ActionResult Confirm(string btnPrev, string btnConfirm)

{

if (SessionFacade.WIZSTEP4 == null)

return RedirectToAction("WizStep4");

else

{

// normal flow - came here from step 4

if (btnConfirm != null)

{

if (btnConfirm.ToUpper() == "CONFIRM")

{

CustomerInfo ci = SessionFacade.WIZSTEP4;

// write ci to DB

ViewBag.Message = "Customer registered successfully..";

SessionFacade.WIZSTEP1 = null;

SessionFacade.WIZSTEP2 = null;

SessionFacade.WIZSTEP3 = null;

SessionFacade.WIZSTEP4 = null;

}

}

if (btnPrev != null)

{

if (btnPrev.ToUpper() == "PREV")

return RedirectToAction("WizStep4");

}

}

return View();

}

}

}

**CustomerInfo.cs (Model):**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Web;

namespace WizardMVC.Models

{

public class CustomerInfo

{

// fields that we need to collect in different wizard steps

[Required(ErrorMessage = "{0} is required")]

public string FirstName { get; set; }

[Required(ErrorMessage = "{0} is required")]

public string LastName { get; set; }

[Required(ErrorMessage = "{0} is required")]

[DataType(DataType.EmailAddress, ErrorMessage = "invalid email..")]

public string Email { get; set; }

[Required(ErrorMessage = "{0} is required")]

public string StreetAddress { get; set; }

[Required(ErrorMessage = "{0} is required")]

public string City { get; set; }

[Required(ErrorMessage = "{0} is required")]

[MaxLength(2, ErrorMessage = "needs to be two chars")]

[MinLength(2)]

public string State { get; set; }

[Required(ErrorMessage = "{0} is required")]

[DataType(DataType.PhoneNumber)]

public string Telephone { get; set; }

[Required(ErrorMessage = "{0} is required")]

[DataType(DataType.PhoneNumber)]

public string CellPhone { get; set; }

[Range(1000, 10000, ErrorMessage = "needs to be between 1000-10000")]

public int CustomerPin { get; set; }

[Required(ErrorMessage = "{0} is required")]

[DataType(DataType.CreditCard)]

public int CardNumber { get; set; }

[DisplayFormat(DataFormatString = "{0:MM/yy}", ApplyFormatInEditMode = true)]

public DateTime ExpirationDate { get; set; }

}

}

**SessionFacade.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using WizardMVC.Models;

namespace WizardMVC.Utils

{

public class SessionFacade

{

static readonly string wizstep1 = "WIZSTEP1";

public static CustomerInfo WIZSTEP1

{

get

{

if (HttpContext.Current.Session[wizstep1] != null)

return (CustomerInfo)HttpContext.Current.Session[wizstep1];

else

return null;

}

set

{

HttpContext.Current.Session[wizstep1] = value;

}

}

static readonly string wizstep2 = "WIZSTEP2";

public static CustomerInfo WIZSTEP2

{

get

{

if (HttpContext.Current.Session[wizstep2] != null)

return (CustomerInfo)HttpContext.Current.Session[wizstep2];

else

return null;

}

set

{

HttpContext.Current.Session[wizstep2] = value;

}

}

static readonly string wizstep3 = "WIZSTEP3";

public static CustomerInfo WIZSTEP3

{

get

{

if (HttpContext.Current.Session[wizstep3] != null)

return (CustomerInfo)HttpContext.Current.Session[wizstep3];

else

return null;

}

set

{

HttpContext.Current.Session[wizstep3] = value;

}

}

static readonly string wizstep4 = "WIZSTEP4";

public static CustomerInfo WIZSTEP4

{

get

{

if (HttpContext.Current.Session[wizstep4] != null)

return (CustomerInfo)HttpContext.Current.Session[wizstep4];

else

return null;

}

set

{

HttpContext.Current.Session[wizstep4] = value;

}

}

}

}

**View WizStep1:**

@model WizardMVC.Models.CustomerInfo

@{

ViewBag.Title = "WizStep1";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h2>WizStep1</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<h4>CustomerInfo - Step 1/4</h4>

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

<div class="form-group">

@Html.LabelFor(model => model.FirstName, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.FirstName, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.FirstName, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.LastName, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.LastName, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.LastName, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Email, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Email, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Email, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Next" class="btn btn-default" />

</div>

</div>

</div>

}

<div>

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

</div>

@section Scripts {

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

}

**View WizStep 2:**

@model WizardMVC.Models.CustomerInfo

@{

ViewBag.Title = "WizStep2";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h2>WizStep2</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<h2>Wiz Step2 - step 2/4</h2>

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

<div class="form-group">

@Html.LabelFor(model => model.StreetAddress, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.StreetAddress, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.StreetAddress, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.City, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.City, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.City, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.State, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.State, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.State, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Prev" name="btnPrev" class="btn btn-default cancel" /> &nbsp;&nbsp;&nbsp;&nbsp;<input type="submit"

value="Next" name="btnNext" class="btn btn-default" />

</div>

</div>

</div>

}

<div>

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

</div>

@section Scripts {

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

<script>

$("input[name=btnPrev]").click(function(){

$("#form1").validate().settings.ignore = "\*"; // disable validation

});

</script>

}

**View WizStep 3:**

@model WizardMVC.Models.CustomerInfo

@{

ViewBag.Title = "WizStep3";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h2>WizStep3</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<h2>Wiz Step3 - Step 3/4</h2>

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

<div class="form-group">

@Html.LabelFor(model => model.Telephone, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Telephone, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Telephone, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.CellPhone, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.CellPhone, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.CellPhone, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.CustomerPin, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.CustomerPin, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.CustomerPin, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Prev" name="btnPrev" class="btn btn-default cancel" /> &nbsp;&nbsp;&nbsp;&nbsp; <input type="submit"

value="Next" name="btnNext" class="btn btn-default" />

</div>

</div>

</div>

}

<div>

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

</div>

@section Scripts {

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

<script>

$("input[name=btnPrev]").click(function(){

$("#form1").validate().settings.ignore = "\*"; // disable validation

});

</script>

}

**View WizStep4:**

@model WizardMVC.Models.CustomerInfo

@{

ViewBag.Title = "WizStep4";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h2>WizStep4</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<h2>Wiz Step 4 - Step 4/5</h2>

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

<div class="form-group">

@Html.LabelFor(model => model.CardNumber, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.CardNumber, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.CardNumber, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.ExpirationDate, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.ExpirationDate, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.ExpirationDate, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Prev" name="btnPrev" class="btn btn-default" /> &nbsp;&nbsp;&nbsp;&nbsp; <input type="submit"

value="Next" name="btnNext" class="btn btn-default" />

</div>

</div>

</div>

}

<div>

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

</div>

@section Scripts {

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

<script>

$("input[name=btnPrev]").click(function(){

$("#form1").validate().settings.ignore = "\*"; // disable validation

});

</script>

}

**View Confirm:**

@model WizardMVC.Models.CustomerInfo

@{

ViewBag.Title = "Confirm";

}

<h2>Please Confirm the following Data</h2>

<fieldset>

<legend>CustomerInfo</legend>

<div class="display-label">

@Html.DisplayNameFor(model => model.FirstName):

@Html.DisplayFor(model => model.FirstName)

</div>

<div class="display-label">

@Html.DisplayNameFor(model => model.LastName):

@Html.DisplayFor(model => model.LastName)

</div>

<div class="display-label">

@Html.DisplayNameFor(model => model.Email):

@Html.DisplayFor(model => model.Email)

</div>

<div class="display-label">

@Html.DisplayNameFor(model => model.StreetAddress):

@Html.DisplayFor(model => model.StreetAddress)

</div>

<div class="display-label">

@Html.DisplayNameFor(model => model.City):

@Html.DisplayFor(model => model.City)

</div>

<div class="display-label">

@Html.DisplayNameFor(model => model.State):

@Html.DisplayFor(model => model.State)

</div>

<div class="display-label">

@Html.DisplayNameFor(model => model.Telephone):

@Html.DisplayFor(model => model.Telephone)

</div>

<div class="display-label">

@Html.DisplayNameFor(model => model.CellPhone):

@Html.DisplayFor(model => model.CellPhone)

</div>

<div class="display-label">

@Html.DisplayNameFor(model => model.CustomerPin):

@Html.DisplayFor(model => model.CustomerPin)

</div>

<div class="display-label">

@Html.DisplayNameFor(model => model.CardNumber):

@Html.DisplayFor(model => model.CardNumber)

</div>

<div class="display-label">

@Html.DisplayNameFor(model => model.ExpirationDate):

@Html.DisplayFor(model => model.ExpirationDate)

</div>

@using (Html.BeginForm())

{

<input type="submit" name="btnPrev" value="Prev" /> @:&nbsp;&nbsp;&nbsp; <input type="submit" name="btnConfirm" value="Confirm" />

<div class="form-group">

<div id="lblStatus" class="col-md-offset-2 col-md-10">

@ViewBag.Message

</div>

</div>

}

</fieldset>

**Conclusion:**

After doing this assignment I was able to practice different attributes about MVC. After doing the first part of the assignment, I got to understand better how the routes work. I now understand that the order of the routes and the names of the parameters are important in order to show the data correctly. Also, by doing the first part of the assignment, I got to understand how to read the url used. If we use the default route: ….{controller}/{action}/{id} means that the first parameter is calling a controller in the controller folder which has different actions, the second parameter is the action we are calling (from the specified controller) and the third parameter is the id we are passing to the action.

By doing the second part, I got to understand better how the relationship between controllers and view work. Also, I understood why Model Binding in MVC is needed. It consists in passing objects to the different actions instead of passing X number of parameters. It is really useful since the data we want to pass could be really big and we would have to send a large number of variables, which could lead to different typo errors and we will not have a clean code.

The third and last part of the assignment consisted in creating a simple Wizard. After creating the wizard, I understood better how the data is related and passed between different sessions using hidden field which is similar to the ViewState.