

ASP.NET MVC – Part 4

Agenda

- More on Controllers
- ViewBag, ViewData, TempData, ViewModel

Tools

- Visual Studio 2013

Pre-requisite

- ASP.NET MVC – Part 3
- Source code of Part 3 will be used

Version	Last updated	Comments	Modified By
V1.0	10-05-2016		Bilal Shahzad

Brief Introduction

Actions

By default type of action in controller is 'HttpGet' but we can also use other HTTP types e.g. HttpPost, HttpPut. When request is landed in controller, type of action is also checked while invoking the function. Let say we have 'Login' action in User controller and Login action has 'HttpPost' attribute on it. When we'll try to access '/User/Login' with GET, there will be error.

Data Posting to Actions

When data is posted to an action, data can be captured by different ways. (Let say posted data has data with two keys a) PhoneNumber b) Age.

- 1- By using 'Request' object: By providing the key to Request object in action, data can be retrieved. Suppose, to get posted 'PhoneNumber', we can get it by Request["PhoneNumber"].
- 2- By using Request.Form object: We can directly access this collection to get data. To get posted PhoneNumber, we can get it by Request.Form["PhoneNumber"].
- 3- By declaring parameters with names same as names of keys.
 - a. ActionResult Register(String PhoneNumber, int Age)
- 4- By declaring object of complex type and that complex type has properties with names same as names of keys.
 - a. Class Test {
 public String PhoneNumber {get;set;}
 public int Age {get;set;}}
 - b. ActionResult Register(Test obj)

Data Passing from Controller to View (on Server Side)

View is independent of Controller but view might be dependent on some data. There are different ways of passing data to view from controller.

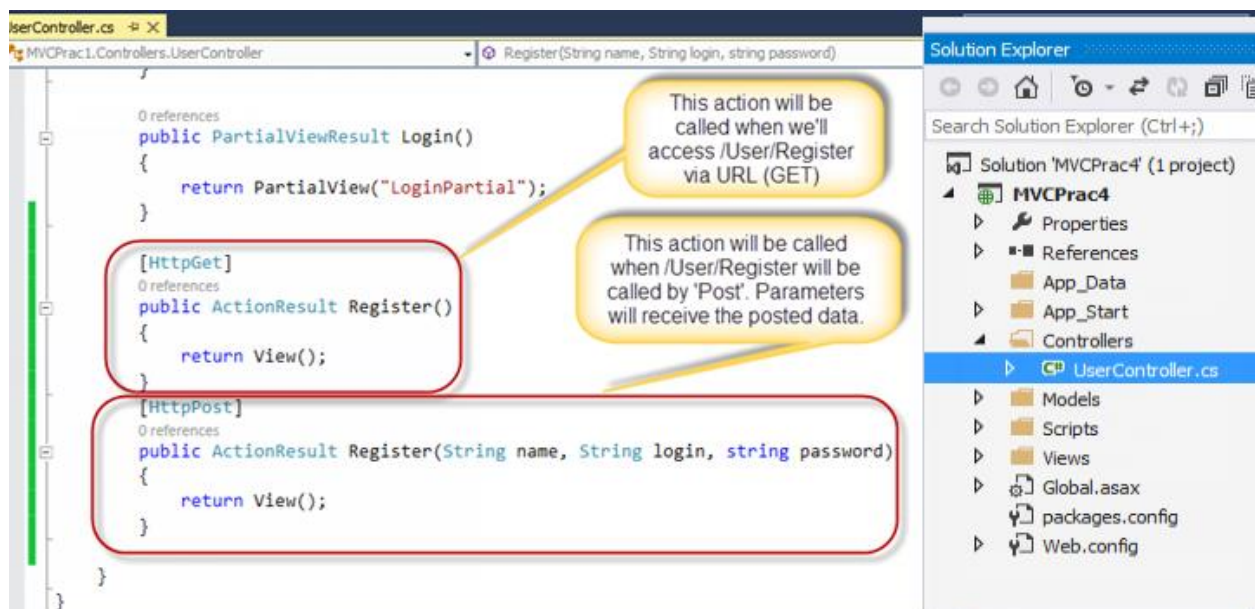
- 1- ViewBag is a dynamic property. Typecasting is required while accessing it.
- 2- ViewData is a dictionary of objects that is accessible using keys. Both ViewBag & ViewData points to same data. Typecasting is required while accessing it.
- 3- TempData is also dictionary of objects that is accessible using keys. The main difference is that TempData lives as long as Http request is alive. If we set value in TempData in one action, we can access that value in other action if same request is in progress. Typecasting is required while accessing it.
- 4- ViewModel: In this approach, we can pass a single object (simple or complex or list of complex etc.) to view from controller. We'll have to tell the type of expected object in view by using '@model ...' syntax and then can access this object by '@Model' in view.

Step by Step Walkthrough

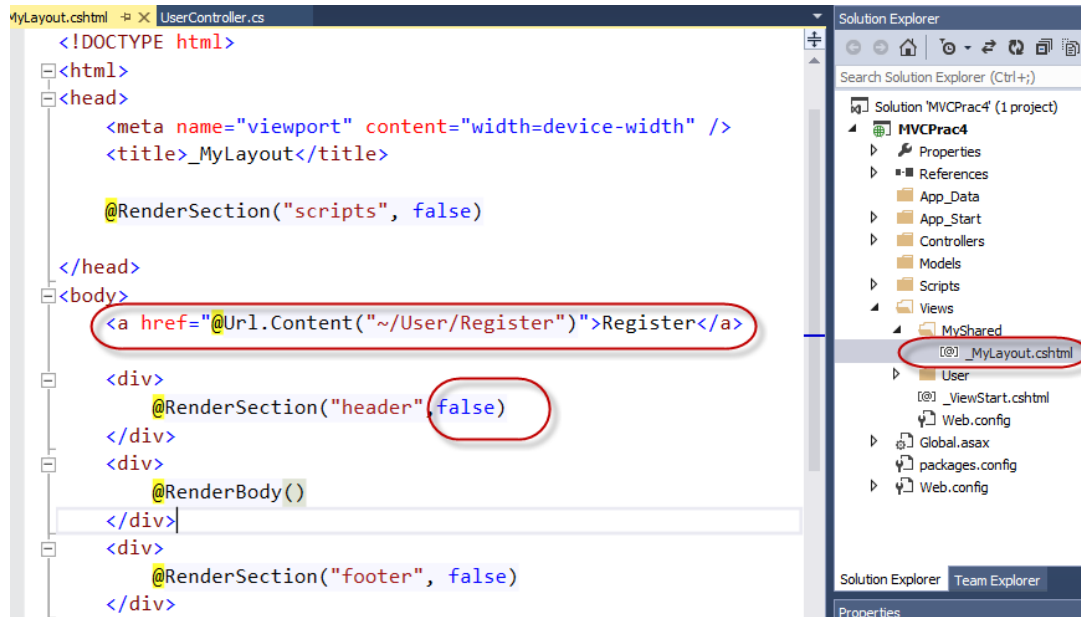
- 1- Let's create a new view "Register" in User folder (as shown in right panel). Write HTML in it as shown in following screenshot. When "Register" button will be clicked, data will be posted to "Register" action of "User" controller.



- 2- Now add following two actions in "UserController" class. First "Register" action will be called when `/User/Register` will be accessed via URL (means GET hit). When user will fill the form and will submit, POST request will be landed on second "Register" action and data will be loaded in parameters.



- 3- Now add a link in “Layout” file so that we’ll be able to click it to open “Register” page. Also set required attribute of “header” section to false so that we’ll not have to fill it anymore.



The screenshot shows the Visual Studio IDE with the `MyLayout.cshtml` file open. The code is as follows:

```
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-width" />
  <title>_MyLayout</title>

  @RenderSection("scripts", false)
</head>
<body>
  <a href="@Url.Content("~/User/Register")">Register</a>

  <div>
    @RenderSection("header", false)
  </div>
  <div>
    @RenderBody()
  </div>
  <div>
    @RenderSection("footer", false)
  </div>
</body>
</html>
```

The Solution Explorer on the right shows the project structure for `MVCPrac4`. The `MyLayout.cshtml` file is highlighted under the `Views` folder.

- 4- Run the project, click on Register link and verify if Register page is shown. Then enter some data and click on “Register” button. It will not do anything right now. You can apply debugger in Register action to check if data is reaching there or not. Also you will notice that once data is posted, page is shown again but without data.
- 5- So If we want that when page is refreshed (and we are on same page due to error or anything else), we should be seeing the entered data in controls then we’ll have to handle it by ourselves. So in function parameters we’ve data (sent from user) and we can share same data with view so that view can be generated with that data and sent back to client. Here we are placing our data in ViewBag property.

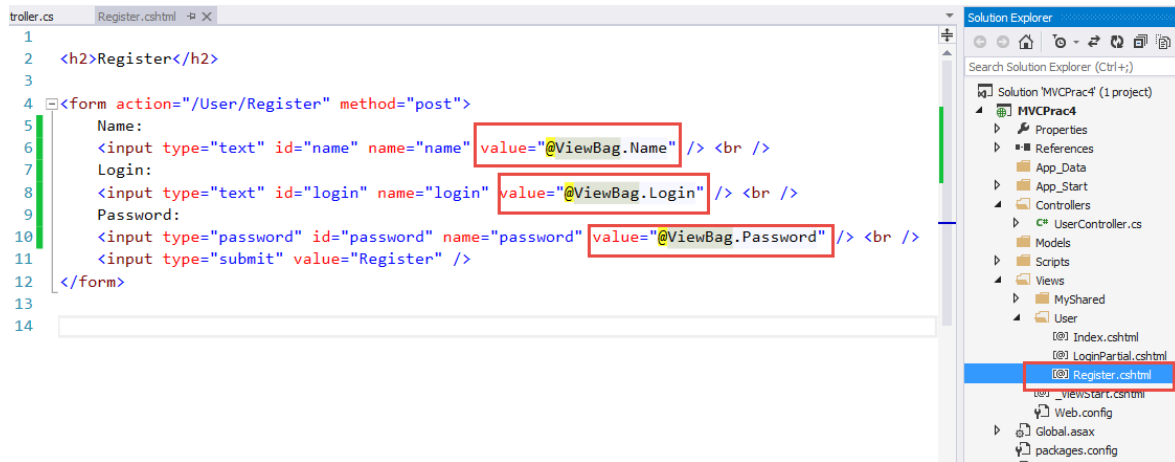


The screenshot shows the Visual Studio IDE with the `Register.cshtml` file open. The code is as follows:

```
17 {
18     return PartialView("LoginPartial");
19 }
20
21 [HttpGet]
22 public ActionResult Register()
23 {
24     return View();
25 }
26 [HttpPost]
27 public ActionResult Register(String name, String login, string password)
28 {
29     ViewBag.Name = name;
30     ViewBag.Login = login;
31     ViewBag.Password = password;
32     return View();
33 }
34
35 }
```

The Solution Explorer on the right shows the project structure for `MVCPrac4`. The `UserController.cs` file is highlighted under the `Controllers` folder.

6- Following screenshot is showing how to access data (stored using ViewBag) by using ViewBag.



The screenshot shows the Register.cshtml file in Visual Studio. The code is as follows:

```
1 <h2>Register</h2>
2
3
4 <form action="/User/Register" method="post">
5     Name:
6     <input type="text" id="name" name="name" value="@ViewBag.Name" /> <br />
7     Login:
8     <input type="text" id="login" name="login" value="@ViewBag.Login" /> <br />
9     Password:
10    <input type="password" id="password" name="password" value="@ViewBag.Password" /> <br />
11    <input type="submit" value="Register" />
12 </form>
13
14
```

The Solution Explorer on the right shows the project structure for MVCPrac4, with Register.cshtml selected under the Views/User folder.

7- Following screenshot is showing how to access data (stored using ViewBag) by using ViewData.

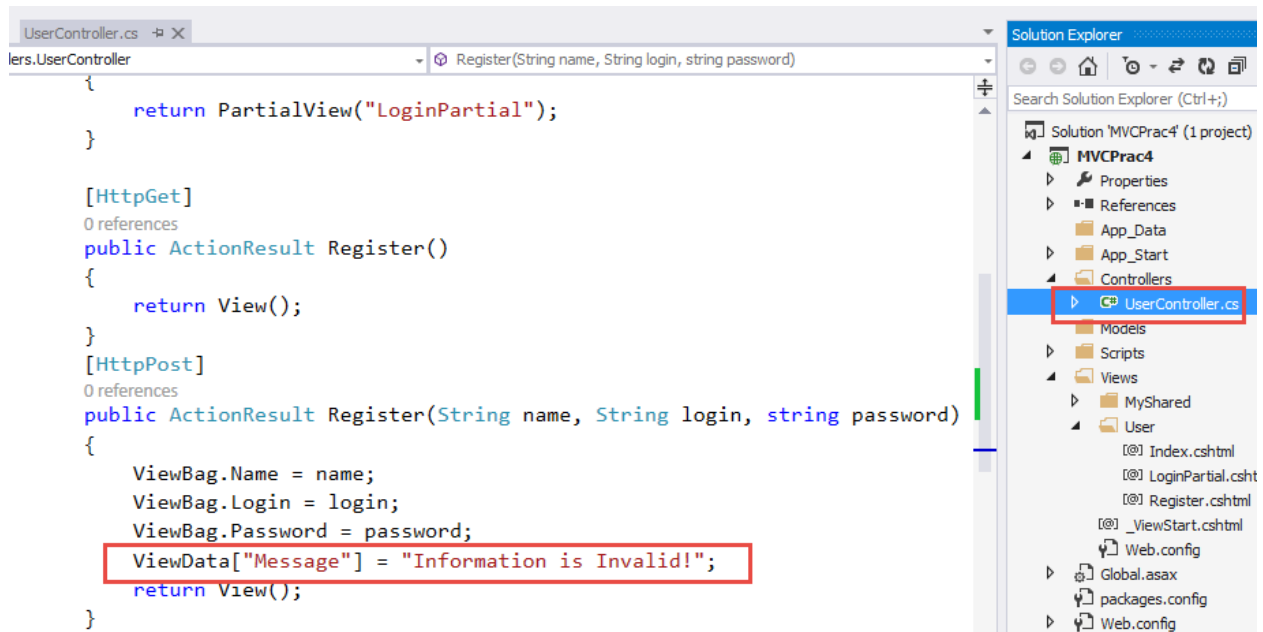


The screenshot shows the Register.cshtml file in Visual Studio. The code is as follows:

```
1 <h2>Register</h2>
2
3
4 <form action="/User/Register" method="post">
5     Name:
6     <input type="text" id="name" name="name" value="@ViewData["Name"]" /> <br />
7     Login:
8     <input type="text" id="login" name="login" value="@ViewData["Login"]" /> <br />
9     Password:
10    <input type="password" id="password" name="password" value="@ViewData["Password"]" /> <br />
11    <input type="submit" value="Register" />
12 </form>
13
14
```

The Solution Explorer on the right shows the project structure for MVCPrac4, with Register.cshtml selected under the Views/User folder.

8- Let's add a property in our action using "ViewData" and access it in our view using "ViewBag"



The screenshot shows the UserController.cs file in Visual Studio. The code is as follows:

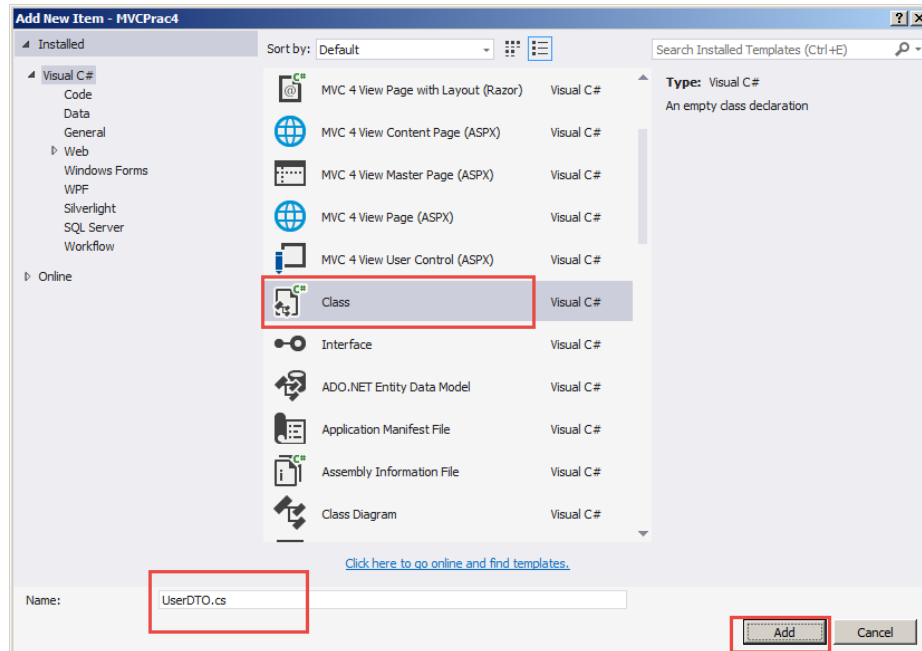
```
1
2
3
4 return PartialView("LoginPartial");
5
6
7 [HttpGet]
8 0 references
9 public ActionResult Register()
10 {
11     return View();
12 }
13 [HttpPost]
14 0 references
15 public ActionResult Register(String name, String login, string password)
16 {
17     ViewBag.Name = name;
18     ViewBag.Login = login;
19     ViewBag.Password = password;
20     ViewData["Message"] = "Information is Invalid!";
21     return View();
22 }
```

The Solution Explorer on the right shows the project structure for MVCPrac4, with UserController.cs selected under the Controllers folder.

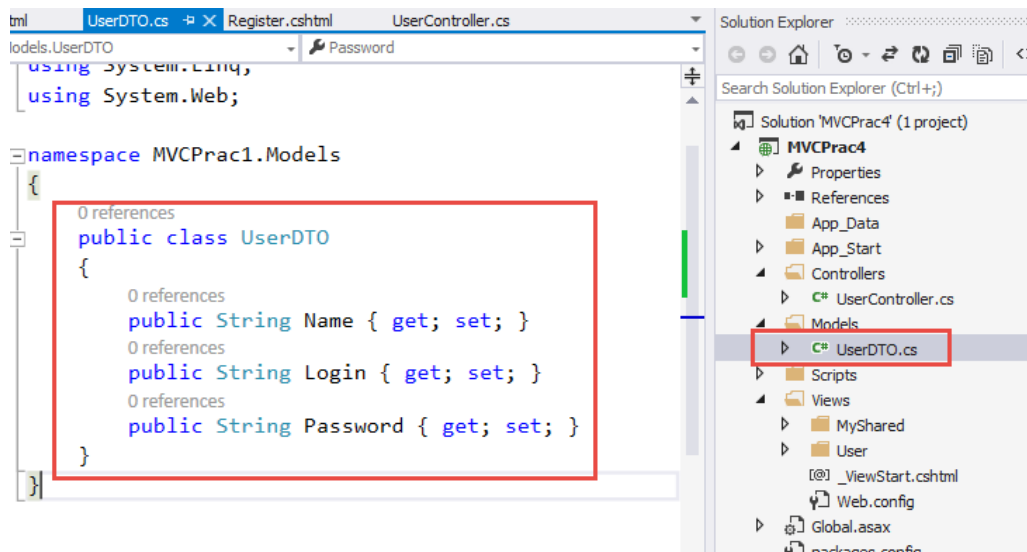
9- Following screenshot is showing how to access data (stored using ViewData) by using ViewBag.



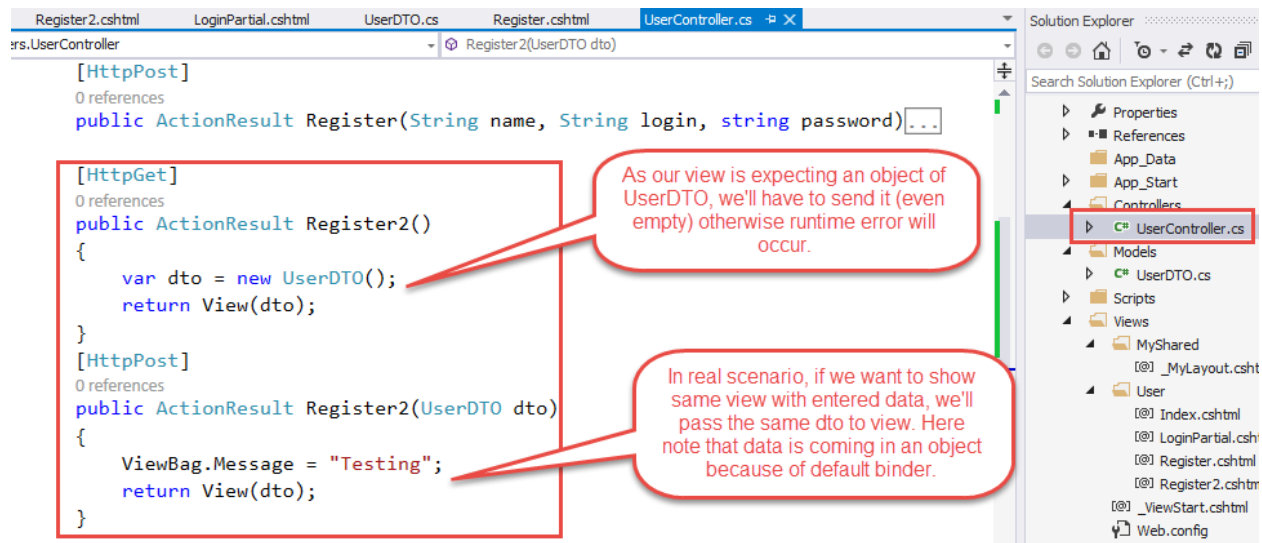
10- Now let's capture data in a DTO. So first create a class in "Models" folder. To do so, right click on "Models" => Add => New item. Name the class "UserDTO".



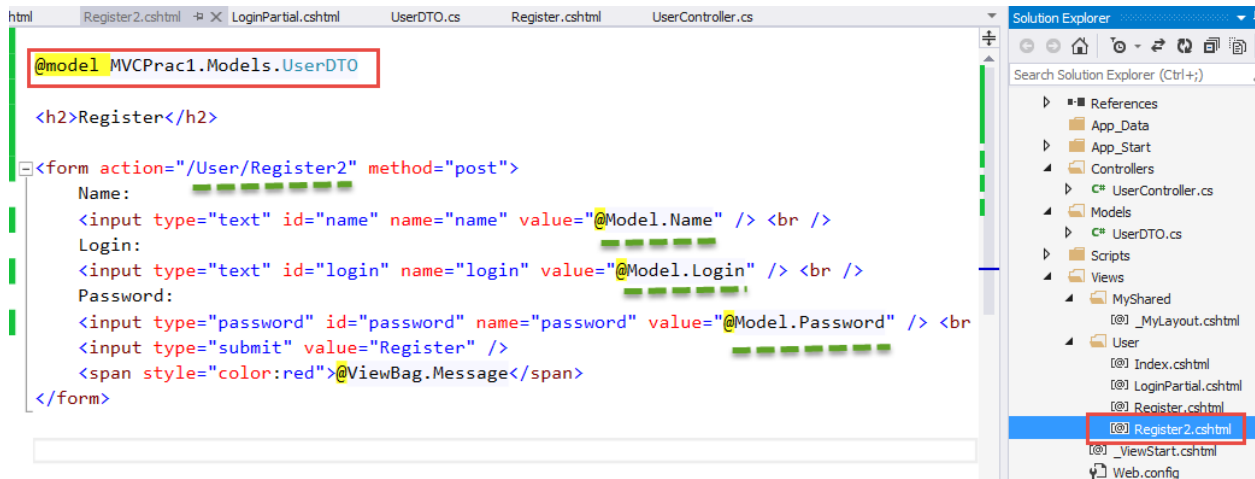
11- Add following properties in UserDTO class.



12- Let's add two more actions to test DTO approach. First "Register2" action will be called when we'll access `/User/Register2` via URL in browser. And when we'll post data after filling the form, request will land on second "Register2" action. Only difference between "Register" & "Register2" is that now we are capturing our data in complex type instead of separate variable. Also note that, now we are using other approach to pass data to view. We are directly passing our DTO while calling "View" function.



- 13- Create copy of “Register” view and name it “Register2.cshtml”. Then open “Register2.cshtml” file and make changes highlighted in following screenshot. Note that we’ll have to declare type of our “ViewModel” object first at top using “@model”. Then later, we can access the sent object by using “@Model” property. Also note that this form is posting data to “/User/Register2”.



```
html Register2.cshtml LoginPartial.cshtml UserDTO.cs Register.cshtml UserController.cs
@model MVCPrac1.Models.UserDTO

<h2>Register</h2>

<form action="/User/Register2" method="post">
  Name:
  <input type="text" id="name" name="name" value="@Model.Name" /> <br />
  Login:
  <input type="text" id="login" name="login" value="@Model.Login" /> <br />
  Password:
  <input type="password" id="password" name="password" value="@Model.Password" /> <br />
  <input type="submit" value="Register" />
  <span style="color:red">@ViewBag.Message</span>
</form>
```

- 14- Add a new link in Layout to access “Register2” page.



```
html Register2.cshtml LoginPartial.cshtml UserDTO.cs Register.cshtml
<html>
<head>
  <meta name="viewport" content="width=device-width" />
  <title>_MyLayout</title>

  @RenderSection("scripts", false)
</head>
<body>
  <a href="@Url.Content("~/User/Register")">Register</a>
  <a href="@Url.Content("~/User/Register2")">Register2</a>

  <div>
    @RenderSection("header", false)
  </div>
  <div>
    @RenderBody()
  </div>
```

- 15- Run the project, click on “Register2” link and verify if “Register2” page is shown. Then enter some data and click on “Register” button. It will not do anything right now. You can apply debugger in “Register2” action to check if data is reaching there or not. Also note that once page is reloaded, data is still appearing as we are sending the view with data.

Tasks for Practice

Once you are done with above tutorial. Try to work on these tasks.

- 1- Check “Lecture 11 of EAD” code at following link. It contains example of Model Binders
 - a. <https://www.dropbox.com/s/1tlutpv3mr24yrx/Lecture%2011-%20ASP.NET%20MVC%203.zip?dl=0>
- 2- Try accessing posted data using Request.
- 3- Try accessing posted data using Request.Form
- 4- Explore & Learn “TempData” by examples
- 5- Check what will happen if you pass your dto using ViewBag (e.g. ViewBag.Data = dto)