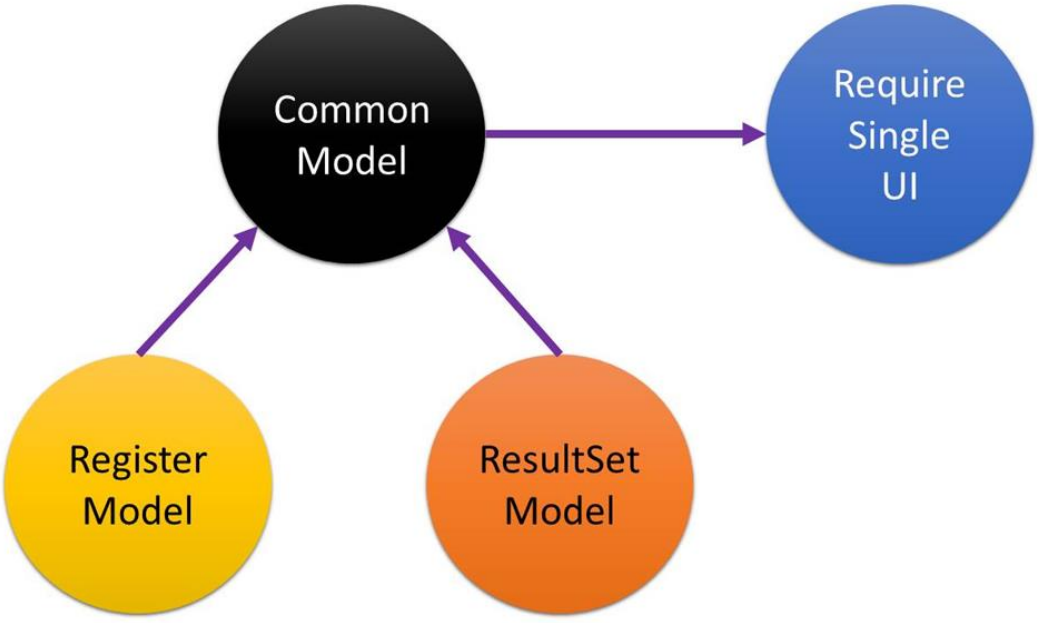
ASP.NET MVC 5 - Multiple View Models In Single View

This article is about implementing multiple view models in a single view via ASP.NET MVC5 platform.


Asma Khalid Apr 12 2017

MultiModelSingleView.zip
Download Free .NET & JAVA Files API



There is a common scenario, which is being asked quite a lot in ASP.NET MVC platforms. Thus, I have decided to demonstrate my take on this particular scenario. The scenario is as follows; i.e., create a single page, which will display the data from two different view models. Now, the question that arises here is if the ASP.NET MVC platform only attaches a single model to a single view, how can this be achieved? Let me demonstrate how this is done.

Following are some prerequisites before you proceed further in this tutorial.

2. Knowledge of HTML.
3. Knowledge of JavaScript.
4. Knowledge of Bootstrap.
5. Knowledge of jQuery.
6. Knowledge of C# programming.

You can download the complete source code for this tutorial or you can follow the step by step discussion below. The sample code is being developed in Microsoft Visual Studio 2015 Enterprise.

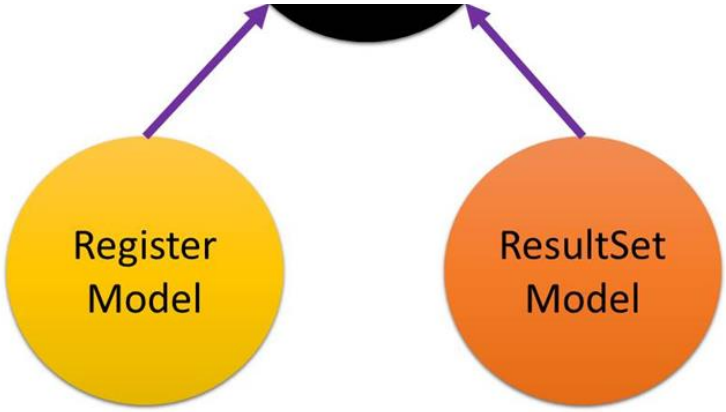
Before we jump into the technical working of this scenario, let's build a little conceptual understanding of how will we achieve this scenario in an ASP.NET MVC5 platform. To do achieve this, we will proceed in our code, as shown below i.e.

Step 1

We have two independent models Register Model & ResultSet Model

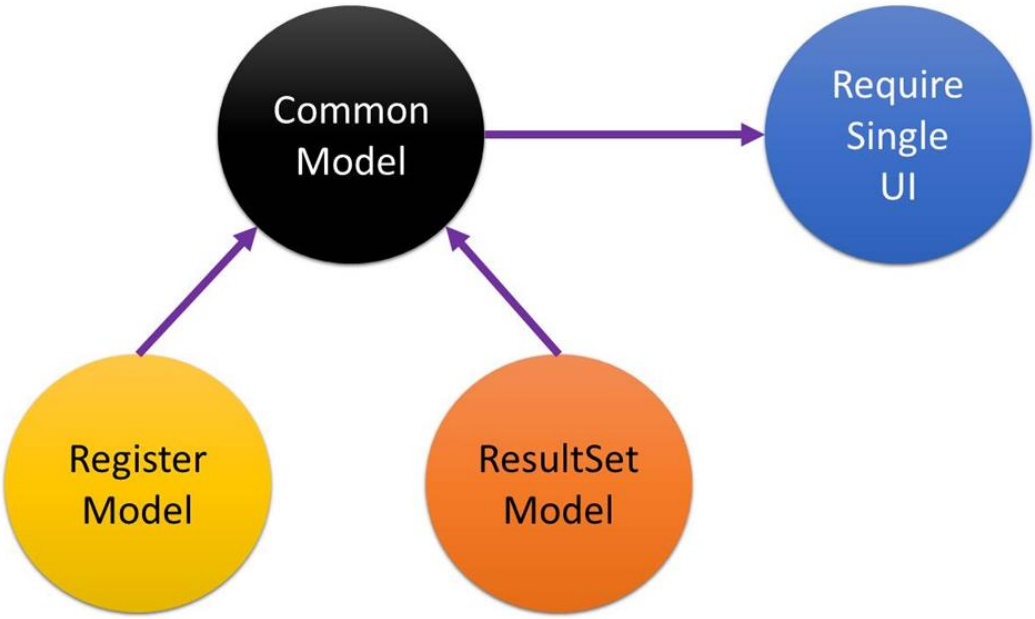
Step 2

Since we know the constraint on ASP.NET MVC platform i.e. we can only attach a single model to a single view. So, here we will create a parent place holder kind of model; i.e., Common Model, and make our actual view models the child to this model, as shown below



Step 3

After combining our two separate models into a single model, we then can easily attach our common place holder model to our single view, as shown below.



Now, let's see, how can we code this.

1. Create a new MVC Web project and name it MultiModelSingleView.
2. Create a file LoginObj.cs under Models folder and replace the code given below in it

```
01. using System;
02. using System.Collections.Generic;
03. using System.Linq;
04. using System.Web;
```

```
    {
        public class LoginObj
        {


            public string Username { get; set; }

            public string Password { get; set; }
        }
    }
```

Now, create our view models files -- AccountViewModel.cs, ResultSetViewModel.cs & CommonViewModel.cs -- under Models folder and replace the code given below in each file accordingly, as shown below

Replace the code given below in AccountViewModel.cs file

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace MultiModelSingleView.Models
{
    public class AccountViewModel
    {
        [Display(Name = "Id")]
        public int Id { get; set; }

        [Display(Name = "Enter Username")]
        public string Username { get; set; }

        [Display(Name = "Enter Password")]
        public string Password { get; set; }
    }
}
```

Replace the code given below in ResultSetViewModel.cs file i.e.

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace MultiModelSingleView.Models
{
    public class ResultSetViewModel
    {
        [Display(Name = "Result")]
        public List<LoginObj> ResultSet { get; set; }
    }
}
```

```
using System.Collections.Generic;
using System.ComponentModel.DataAnn
{
    public class CommonViewModel
    {
        public AccountViewModel AccountVM { get; set; }
        public ResultSetViewModel ResultSetVM { get; set; }
    }
}
```

In all the snippets given above, we have created our view models according to the conceptual understanding, which we have developed.

Create a new controller file AccountController.cs under Controllers folder and replace the code given below in it

```
//-----------------------------------------------------------------------
// <copyright file="HomeController.cs" company="None">
//     Copyright (c) Allow to distribute this code.
// </copyright>
// <author>Asma Khalid</author>
//-----------------------------------------------------------------------

namespace MultiModelSingleView.Controllers
{
    using System;
    using System.Globalization;
    using System.Linq;
    using System.Security.Claims;
    using System.Threading.Tasks;
    using System.Web;
    using System.Web.Mvc;
    using Microsoft.AspNet.Identity;
    using Microsoft.AspNet.Identity.Owin;
    using Microsoft.Owin.Security;
    using MultiModelSingleView.Models;
    using System.Collections.Generic;
    using System.IO;
    using System.Reflection;

    public class AccountController : Controller
    {

        #region Register method

        #region GET: /Account/Register
```

```
        // GET: /Account/Register
        [AllowAnonymous]
        public ActionResult Register()


            CommonViewModel model = new CommonViewModel();
            model.AccountVM = new AccountViewModel();
            model.ResultSetVM = new ResultSetViewModel();

            // Get Result
            model.ResultSetVM.ResultSet = this.LoadData();

            return View(model);
        }

        #endregion

        #region  POST: /Account/Register

        //
        // POST: /Account/Register
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public ActionResult Register(CommonViewModel model)
        {
            if (ModelState.IsValid)
            {
                // Inserting.
                this.StoreData(model.AccountVM.Username, model.AccountVM.P

                // Get Result
                model.ResultSetVM = new ResultSetViewModel();
                model.ResultSetVM.ResultSet = this.LoadData();
            }

            // If we got this far, something failed, redisplay form
            return View(model);
        }

        #endregion

        #endregion

        #region Helpers

        #region Load Data

        /// <summary>
        /// Load data method.
        /// </summary>
        /// <returns>Returns - Data</returns>
```

```
            // Initialization.
            List<LoginObj> lst = ne
```

```
                // Initialization.
                string line = string.Empty;
                string srcFilePath = "content/files/login_list.txt";
                var rootPath = Path.GetDirectoryName(Assembly.GetExecuting
                var fullPath = Path.Combine(rootPath, srcFilePath);
                string filePath = new Uri(fullPath).LocalPath;
                StreamReader sr = new StreamReader(new FileStream(filePath

                // Read file.
                while ((line = sr.ReadLine()) != null)
                {
                    // Initialization.
                    LoginObj infoObj = new LoginObj();
                    string[] info = line.Split(',');

                    // Setting.
                    infoObj.Id = Convert.ToInt32(info[0].ToString());
                    infoObj.Username = info[1].ToString();
                    infoObj.Password = info[2].ToString();

                    // Adding.
                    lst.Add(infoObj);
                }

                // Closing.
                sr.Dispose();
                sr.Close();
            }
            catch (Exception ex)
            {
                // info.
                Console.Write(ex);
            }

            // info.
            return lst;
        }

        #endregion

        #region Store Data

        /// <summary>
        /// Store data method.
        /// </summary>
        private void StoreData(string username, string password)
        {
```

```
142             try
143             {
146                                 ? (this.LoadData().OrderByDescending(p => p.Ic
147                                 : 1;
148
149                 // Initialization.
150                 string line = string.Empty;
151                 string srcFilePath = "content/files/login_list.txt";
152                 var rootPath = Path.GetDirectoryName(Assembly.GetExecuting
153                 var fullPath = Path.Combine(rootPath, srcFilePath);
154                 string filePath = new Uri(fullPath).LocalPath;
155                 StreamWriter sw = new StreamWriter(new FileStream(filePatl
156
157                 // Write file.
158                 string content = idVal.ToString() + "," + username + "," -
159                 sw.WriteLine(content);
160
161                 // Closing.
162                 sw.Dispose();
163                 sw.Close();
164             }
165             catch (Exception ex)
166             {
167                 // info.
168                 Console.Write(ex);
169             }
170         }
171
172         #endregion
173
174         #endregion
175     }
176 }
```

Let's break down the code given above, method by method

```
01. #region Load Data
02.
03. /// <summary>
04. /// Load data method.
05. /// </summary>
06. /// <returns>Returns - Data</returns>
07. private List<LoginObj> LoadData()
08. {
09.     // Initialization.
10.     List<LoginObj> lst = new List<LoginObj>();
11.
12.     try
```

```
        string line = string.Empty;
        string srcFilePath = "conte
        var rootPath = Path.GetDirectoryName(Assembly.GetExecutingAssembly

        StreamReader sr = new StreamReader(new FileStream(filePath, FileM

        // Read file.
        while ((line = sr.ReadLine()) != null)
        {
            // Initialization.
            LoginObj infoObj = new LoginObj();
            string[] info = line.Split(',');

            // Setting.
            infoObj.Id = Convert.ToInt32(info[0].ToString());
            infoObj.Username = info[1].ToString();
            infoObj.Password = info[2].ToString();

            // Adding.
            lst.Add(infoObj);
        }

        // Closing.
        sr.Dispose();
        sr.Close();
    }
    catch (Exception ex)
    {
        // info.
        Console.Write(ex);
    }

    // info.
    return lst;
}

#endregion
```

The code given above creates a LoadData() method, which will load the data from a file, if the file contains any data, initially, the file is empty, since we have not registered any account.

```
#region Store Data

/// <summary>
/// Store data method.
/// </summary>
private void StoreData(string username, string password)
{
    // Initialization.
```

```
        try
        {
            // Setting.

                        : 1;

            // Initialization.
            string line = string.Empty;
            string srcFilePath = "content/files/login_list.txt";
            var rootPath = Path.GetDirectoryName(Assembly.GetExecutingAssembly
            var fullPath = Path.Combine(rootPath, srcFilePath);
            string filePath = new Uri(fullPath).LocalPath;
            StreamWriter sw = new StreamWriter(new FileStream(filePath, FileM

            // Write file.
            string content = idVal.ToString() + "," + username + "," + passwo
            sw.WriteLine(content);

            // Closing.
            sw.Dispose();
            sw.Close();
        }
        catch (Exception ex)
        {
            // info.
            Console.Write(ex);
        }
    }

    #endregion
```

Now, the code given above creates a StoreData(...) method, which will store the data into the file, as we register any new account.

```
#region GET: /Account/Register

        //
        // GET: /Account/Register
        [AllowAnonymous]
        public ActionResult Register()
        {
            // Initialization.
            CommonViewModel model = new CommonViewModel();
            model.AccountVM = new AccountViewModel();
            model.ResultSetVM = new ResultSetViewModel();

            // Get Result
            model.ResultSetVM.ResultSet = this.LoadData();

                    ...(model)
```

```
19.         #endregion
```

individual model to the view, we are returning our common view model, since we have attached that with our View.

```
01. #region  POST: /Account/Register
02.
03.         //
04.         // POST: /Account/Register
05.         [HttpPost]
06.         [AllowAnonymous]
07.         [ValidateAntiForgeryToken]
08.         public ActionResult Register(CommonViewModel model)
09.         {
10.             if (ModelState.IsValid)
11.             {
12.                 // Inserting.
13.                 this.StoreData(model.AccountVM.Username, model.AccountVM.P
14.
15.                 // Get Result
16.                 model.ResultSetVM = new ResultSetViewModel();
17.                 model.ResultSetVM.ResultSet = this.LoadData();
18.             }
19.
20.             // If we got this far, something failed, redisplay form
21.             return View(model);
22.         }
23.
24.         #endregion
```

The code given above creates our view method Register() HTTP POST, which will simply load our data to ResultSetViewModel after storing the register account details received via account view model. Notice here again that now in our view instead of returning an individual model to the view, we are returning our common view model, since we have attached that with our view.

Now, create a new view file Register.cshtml under Views\Account folder and replace the code given below in it

```
01. @model MultiModelSingleView.Models.CommonViewModel
02. @{
03.     ViewBag.Title = "Register";
04. }
05.
06. <h2>@ViewBag.Title.</h2>
07.
```

```
{
    @Html.AntiForgeryToken()
    <h4>Create a new account.</h4>


    <div class="form-group">
        @Html.LabelFor(m => m.AccountVM.Username, new { @class = "col-
md-2 control-label" })
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.AccountVM.Username, new { @class = "fo
control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.AccountVM.Password, new { @class = "col-
md-2 control-label" })
        <div class="col-md-10">
            @Html.PasswordFor(m => m.AccountVM.Password, new { @class = "f
control" })
        </div>
    </div>

    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" class="btn btn-
default" value="Register" />
        </div>
    </div>
}

<h2>Result List</h2>

@if (Model.ResultSetVM.ResultSet != null)
{
    for (int i = 0; i < Model.ResultSetVM.ResultSet.Count; i++)
    {
        <div class="row">
            <div class="col-md-2">
                <p>@Model.ResultSetVM.ResultSet[i].Id</p>
            </div>
            <div class="col-md-2">
                <p>@Model.ResultSetVM.ResultSet[i].Username</p>
            </div>
            <div class="col-md-2">
                <p>@Model.ResultSetVM.ResultSet[i].Password</p>
            </div>
        </div>
    }
}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
```

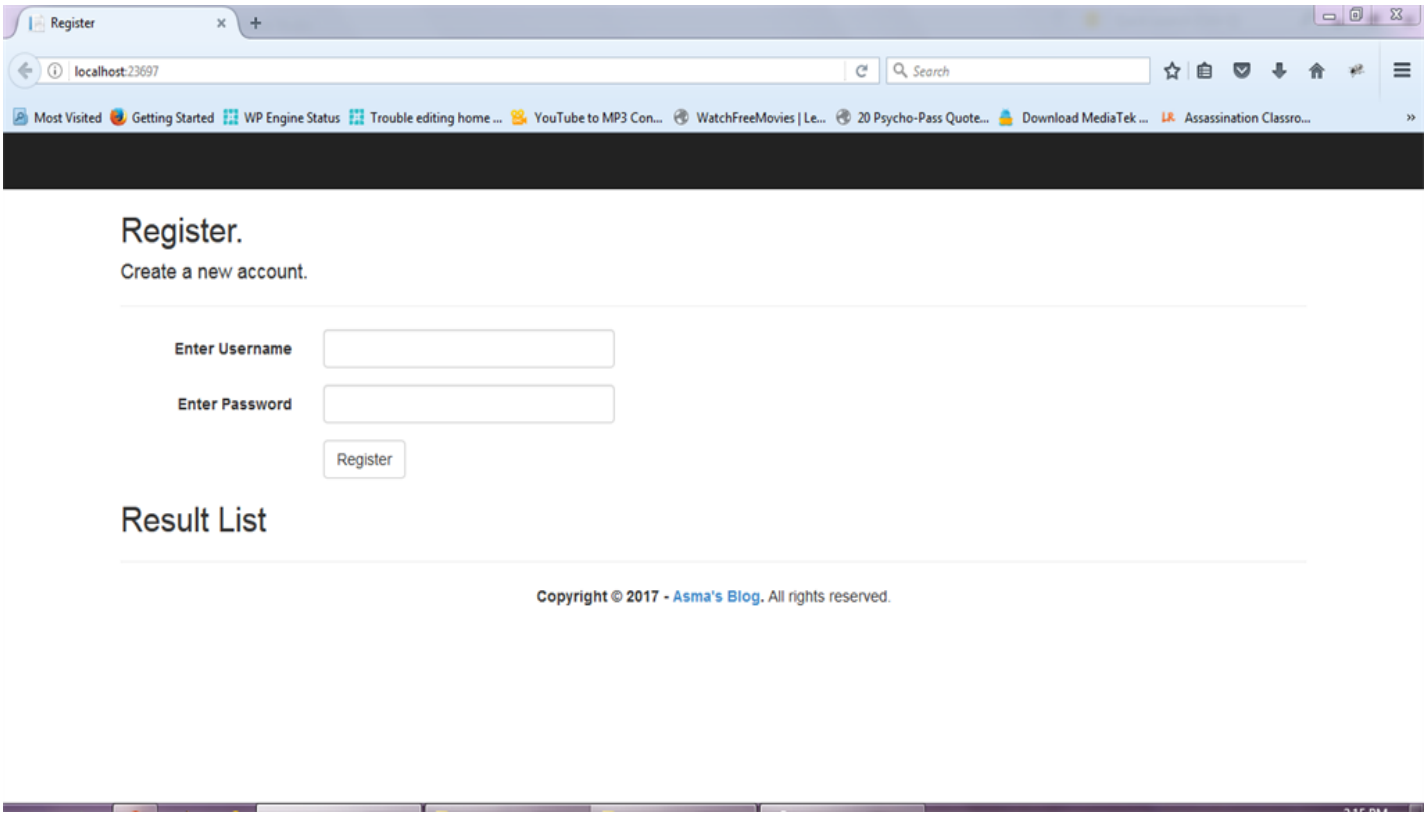
In the code given above, we have created a single

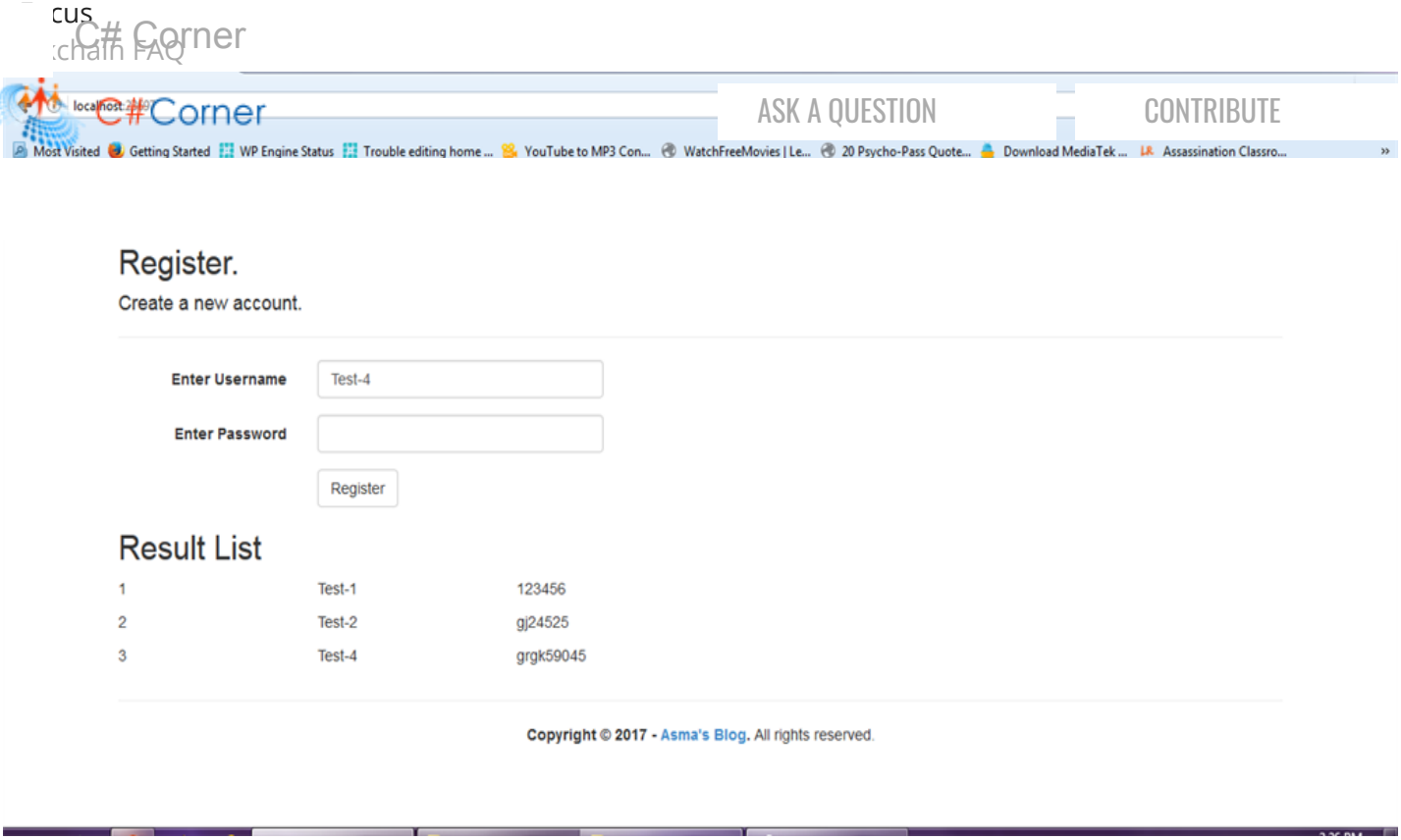
Create a our standard view layout file _Layout.cshtml under Views\Shared folder. Replace the code given below in it i.e.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")

    <!-- Font Awesome -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.4.0/css/font-awesome.min.css" />
</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>
            </div>
        </div>
    </div>
    <div class="container body-content">
        @RenderBody()
        <hr />
        <footer>
            <center>
                <p>
<strong>Copyright © @DateTime.Now.Year - <a href="http://www.asmak9.com/">
</strong> All rights reserved.</p>
            </center>
        </footer>
    </div>

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

```

```
cus
chain FAQ
40.  </body>
41  </html>
```


shown below.





Conclusion

In this article, you learned about mapping multiple view models to a single UI view in an ASP.NET MVC5 platform. You also learned about conceptual understanding behind this scenario along with how to achieve this scenario from the coding perspective.

ASP.NET MVC

Asma Khalid TOP 500

Computer Programmer by Profession, Computer Scientist by Heart, Fanatic Explorer, Technology Centric. I am a Versatile Computer Science Evangelist. Enjoy doodling with technology.

http://www.asmak9.com/

191 **2.2m** **3** **2**

7 **12**

Type your comment here and press Enter Key (Minimum 10 characters)

I am having errors in html page like (name 'model' etc doesn't exit in current context) what should i

Anmol Butt
1757 6 0
Dec 07, 2018
0

How to connect this with database.... please help me out in this also???
Anmol Butt
1757 6 0
Dec 07, 2018
0

Look into this https://bit.ly/2Sx78U9
Asma Khalid
191 9.6k 2.2m
Dec 07, 2018
0

What if i want to use paging in the index method how do i do that
yusuf ogundiji
1758 5 0
Jul 04, 2018
0 0 Reply

Nice article Asma. Alternate method is also used through Expando object.
Dilshad Ali
1216 559 10.8k
Apr 13, 2017
0 0 Reply

Thank you all for your support.
Asma Khalid
191 9.6k 2.2m
Apr 13, 2017
1 0 Reply

I have vs 2013 installed how can i convert or run in vs 2013 ??
Manav Pandya
85 19.7k 604.4k
Apr 13, 2017
0 1 Reply

Manav Pandya you can import .csproj file into VS 2013 instead of directly opening the .snl file. Hope this helps
Asma Khalid
191 9.6k 2.2m
Apr 13, 2017
0

Thanks for sharing ma'm .....
Manav Pandya
85 19.7k 604.4k
Apr 13, 2017
0 0 Reply

Nice article For start-up engineers
Hamid Khan
586 2.4k 175.2k
Apr 12, 2017
0 0 Reply

TRENDING UP

01 GraphQL In .NET Core Web API With Entity Framework Core - Part Three

02 C# Coding Standards 😎

03 Why Agile Is Important In Software Development

04 ASP.NET MVC Request Life Cycle

05 CRUD Operations On CosmosDB Mongo API Using ASP.NET Core

06 Create Registration And Login Page Using Angular 7 And Web API

07 GraphQL In .NET Core Web API With Entity Framework Core - Part One

08 What Does “var” Mean In C#?

09 Upload Files In Azure Blob Storage Using ASP.NET Core

10 Best Programming Language for Mobile App Development

View All