MVC Tutorial

# Basic MVC Application Setup

At CMU we make a few modifications to the initial application to facilitate a semantic approach to MVC. It is important to remember that there are several components of the MVC framework that function by convention (i.e. automagic or “it just works”). We don’t adjust any of these, typically, but we do rename some folders for semantics.

First, create a MVC application using Visual Studio 2013 using step 1 of this tutorial:

<http://www.asp.net/mvc/tutorials/mvc-5/introduction/getting-started>

Please try to change the authentication (using the Change Authentication button on the ASP.NET Project wizard) to Windows Authentication. This will be used later but is easiest to change at this stage.

Now, change the name of the Models folder to ViewModels, unless you need a separate folder for models, in which case you want to add a folder called ViewModel. Keep this name change in mind throughout the tutorial. If it has a namespace component of “Models” then it will be ViewModels in most instances. This implies an adherence to the Model-View-ViewModel pattern that is a variant of MVP, which, itself, is a variant of MVC. The components, in terms of ASP.NET MVC terms and semantics are:

* Model – these can be data-centric models often used for persisting application data to a data layer (like a database) or to represent business logic. These are Plain Old CS Objects (POCO) and are implemented as classes.
* View – views are used to assemble the HTML and CSS of the resulting HTTP response. All layout logic should reside within the view. No data access (other than, arguably, AJAX requests from the resulting HTML sent to the client) should be included in the view. Logic affecting what is displayed and the state of what is displayed is contained in the view.
* ViewModel – this is an aggregation of all the data necessary for the view. This includes data like FirstName properties, etc. but also data for logical operations like IsAdmin used for output decisions as well as collections used for drop down lists, etc. It is very literally the model for the view. If there is to be a *Model* folder, it should NOT contain models used for the views.

Several resources discuss some of the nuances of these patterns. <http://joel.inpointform.net/software-development/mvvm-vs-mvp-vs-mvc-the-differences-explained/> Remember that we aren’t trying to have a pure implementation, but we have adopted principles that seem to have borne out well in the separation of display, logic, and data access code, which helps in the long-term development and maintenance scenarios we encounter.

The changes you make to this application as part of this tutorial are designed to help emphasize the semantic reasoning for our design decisions and to familiarize you with MVVM and basic MVC concepts. You will be following along with the linked tutorial but some modifications will be provided.

Notes:

* Part 2 – when it updates the Welcome method to accept a parameter with the casing of “ID” our guidelines would state to have that remain camelCase (i.e., for this case "id" is correct).
* Uses of “ViewBag”… go ahead and use it as defined in the tutorial, but understand that it is very easy to abuse the ViewBag (you can just type ViewBag.anythingyouwant = “something” and it will store it, but discovering that later is not possible with the ViewBag, so from a maintenance perspective it offers up special challenges).
* Go ahead and try to make changes as you desire during this tutorial to test your understanding… if that means adding additional properties to display, calculations, etc. go ahead.
* Part 5 - You may run into issues because I don’t think SQL Server Express is installed. Let us know when you get to this point so we can get your test databases set up.