MVC Notes: https://www.youtube.com/watch?v=zWFoZb6EiwU

* MVC = “Model, View, Controller”
* Ruby on Rails is based off MVC
* Useful for organizing code, markup and control flow.

**Useful Shortcut keys.**

* Ctrl + F5: Run without debug.
* F5: Run W/ debugger.
* Alt + Up: Moves selected text up lines and moves lines in its path below the selection.
* Ctrl + Shift + B: Builds project.

**Quick Ref**

* @\* \*@ are comment tokens for Razor.
* @: Will allow plain text in Razor
  + You can also just put it within the <text> tag.
* -m suffix at the end of an int (54.64m) will treat the number as an int and not a float.
* ActionResult methods can be used to do many things. One of the main is creating views.
  + URL Patterns
    - home.index, home/about
      * home/{action}
      * {controller}/{action}
      * {Controller}/{action}/{id}
* Routes.IgnoreRoute(“{resource}.axd/{\*pathInfo}”);
  + Any request for .axd will be ignored because this files will be used for other resources then we will be using here.

routes.MapRoute(

name: “Default”,

url: “{controller}/{action}/{id}”,

defaults: new {controller = “Home”, action = “Index”, id = UrlPerameter.Optional }

);

Called when the application starts.

Within the global applications file (Global.asax.cd) we can see the RouteConfig.RegisterRoutes method call.

This method lives within the RouteConfig.cs file and the code is found above.

Routes will return the first method that works. So if there is a default then we must move the custom Route above the default one. I believe that the default should always be the last.

Action Result Types

* PartialView
  + Without the header files or CSS.
  + Barebones.
* ContentResult
  + Returns user-defined content (text, xml)
* JsonResult
  + Returns a JSON object.
* RedirectToRouteResult
  + Issues a redirect to another action
  + Such as changing the webpage after a form is filled out.

Action Selectors

* [HttpPost] //The brackets indicate that this should only be invoked in response to an HTTP Request.

[HttpPost]

Public ActionResult Create(Customer customer)

{

db.Customers.Add(customer);

db.SaveChanges();

return RedirectToAction(“Index”);

}

[HttpGet]

Public ActionResult Create()

{

Return view();

}

Filters

Authorization filter

* Allows you to specify what users or groups have access to particular controllers or methods.

[Authorize(Roles=”administrator”, Users=”jsmith”)]

[HttpPost]

Public ActionResult Create(Customer customer)

{

Some code….

}

//Anyone who has the role as admin. Will be able to create customer objects along with any user named jsmith regardless of role.

[Authorize]

[HttpPost]

//Authorize without any params will default to only allow access to users who a logged in and deny anonymous visitors.

ActionFilter

Exception Filters.

* When errors occur in the web app and the app detects that you are running on the same server, It assumes that it is the developer who is working with it. Thus it will display the offending code and a detailed error page.
* This is called the Yellow Screen of Death.
  + But if we want to see what the error would look like as a remote user then do this:
    - SolutionExplorer -> Web.config -> <system.web> and insert “<customErrors mode= “On” />

Razor Syntax (imbedded C#)

* Smart. Wont get confused about where the HTML ends and the C# begins. It is also able to identify and differentiate emails which have the @ sign.
* Twitter handles will confuse it though.
  + Avoid with double @@ThisShitTwitter.
* Whats really incredible is that we can take the cshtml and imbed C# code into it, but then within the C# code we can seamlessly switch back to HTML.

You are using @Request.Browser.Browser

<!--Tester for the razor method below. The -m suffix will treat these numbers as int and not floats.-->

@{var amounts = new List<Decimal> { 100, 25.50m, -40, 275.99m };

}

<ul>

@foreach (decimal amount in amounts)

{

<li>@formatAmount(amount)</li>

}

</ul>

<!--Example method to show how dollar amounts are formated.-->

@helper formatAmount(decimal amount)

{

var color = "green";

if (amount < 0)

{

color = "red";

}

//Format currency based on default culture.

<span style="color:@color">@String.Format("{0:c}", amount)</span>

}

Layouts

* We can find the default layout selector in the \_ViewStart.cshtml file. This file tells the program where to find the default .cshtml file for the page layouts.
* The default is set to \_Layout.cshtml and can be found in the Shared folder.
* Here we see the @Styles.Render, @RenderBody and @Scripts.Render. These are the insertion points for the ViewFile.
  + We can separate out things like style sections, content sections(html) and script sections since these things are typically separated.