CENTRIQ TRAINING

SIMPLE TEMPLATE CONVERSION

This assumes that no exterior data is being used and Authentication/Authorization is not required.

1.	Choose a template.
2.	Add an **_Archive** folder to your **MVC.UI layer**
3.	Unzip the template into the _Archive Folder
4.	Copy **Images, Styles, Js, along with the index.html** *(or appropriate template page – SingleColumn.html, TwoColumn.html…)* into the **_Archive** Folder
5.	Rename **_Layout.cshtml** in **Views/Shared/ to _OriginalLayout.cshtml**
6.	**Add a new View** in **Views/Shared/** called **_Layout**(make sure the checkbox for *Use Layout is deselected)*
7.	Copy all HTML code from _archive/index.html (or appropriate template page – SingleColumn.html, TwoColumn.html…) and paste over content in **_Layout.cshtml** – then *close the .html file*
8.	Copy and then comment out the "main content" from **_Layout.cshtml** and **paste over all HTML in Index**.cshtml a. Make it your own – **Do these in SMALL steps so you can ctrl+Z if necessary** b. In **_Layout** you should try to leave as much of the structured HTML as possible (Items like ***header, nav, footer, any main content wrapper***, etc) c. In **Index.cshtml** add a ViewBag.Title in a razor block at the top of the page a. @{ ViewBag.Title = "Home"; }
9.	In **_Layout.cshtml** add the ViewBag.Title in the **<title>** a. **<title>@ViewBag.Title</title>**
10	In **_Layout.cshtml** add **@RenderBody()** below the main content that was commented out
11.	In **_Layout.cshtml** update ***all file paths*** (CSS, JS, **Images and main navigation**) a. For **CSS, JS, and Images** you should only need to add **"~/Content/"** i. Example stylesheet link – change ii. <link rel="stylesheet" href="css/custom.css" /> to <link rel="stylesheet" href="**~/Content/**css/custom.css" /> iii. This is a good time to check the favicon as well. If you want to generate one you can use www.favicon.io b. For hyperlinks you can use a Url.Action(), Html.ActionLink() i. **<li class="active"><a *class="active-link"***

	href="index.html">Home</a></li> to <li class="active">@Html.ActionLink("Home", "Index", "Home", *null*, *new {@class="active-link"}*)</li> OR **ii.** <li class="active"><a *class="active-link"* href="@Url.Action("Index", "Home")">Resume</a></li>
12.	*(Optional)* It is best practice, but not required to include a RenderSection() for scripts in_Layout.cshtml below other <script> references a. Example <script src="~/Scripts/jquery.1.10.4.js"></script> <script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script> @RenderSection("scripts", required: false)
13.	*(Optional)* For any additional HTML pages in the template you can create a new Action in the HomeController (*public ActionResult [pagename](){ return View();}*) a. Right click inside the Action and Add View b. Check 'Use a layout page' c. Copy all unique HTML from the HTML page and paste over content in the View

(OPTIONAL) Bundling Styles or Scripts

Bundling is not required but can improve performance by reducing how many times we make a request to the server. When bundling, we are going to make changes to 2 files:

BundleConfig.cs

a. In AppStart/BundleConfig.cs there is a collection (BundleCollection bundles) that groups.js and .css filestogether. You can add a new ScriptBundle, StyleBundle, or add files to an existing bundle.

Option 1: Add files to an existing bundle in AppStart/BundleConfig.cs	
1.	In the Include() method, add a string file path to the comma-separated list a. CSS Example (adding a custom.css file to the bundle) *bundles.Add(new StyleBundle("~/Content/css").Include("~/Content/bootstrap.css", "~/Content/site.css","~/Content/custom.css"));* b. JS Example i. *bundles.Add(new ScriptBundle("~/bundles/jquery").Include("~/Scripts/jquery-* *ii.* *{version}.js", "~/Content/js/custom.js"));*

	Option 2: Create a new bundle in AppStart/BundleConfig.cs
1.	Use **Add()** to create a new **ScriptBundle** or **StyleBundle** 1. **CSS Example** (StyleBundle) a. *bundles.Add(new StyleBundle("~/Content/custom").Include("~/Content/css/customstyles.css"));* i. **"~/Content/custom"** is the virtual path for this bundle – this is how you will reference the bundle in your **_Layout.cshtml** ii. ***"~/Content/css/customstyles.css"*** is the path pointing to the CSS file 2. JS Example (ScriptBundle) a. *bundles.Add(new* ScriptBundle("~/Content/js").Include("~/Content/js/custom.js")); b. ***"~/Content/js"*** is the virtual path for this bundle – referenced in **_Layout.cshtml** c. ***"~/Content/js/custom.js"*** is the path pointing to the JS file
	Referencing the bundles in _Layout.cshtml
2.	**_Layout.cshtml** a.To reference CSS bundles, in the **<head>** add the following @Styles.Render("~/Content/css") @Styles.Render("~/Content/custom") *Use the virtual path from BundleConfig.cs* b. To reference JS Bundles, add the following **above the closing </body>** tag OR **above RenderSection()** @Scripts.Render("~/bundles/jquery") @Scripts.Render("~/Content/js") *"~/bundles/jquery" and "~/Content/js" are virtual paths*