Hello World - Our User Interface

# Introduction

Last week we created a simple Web API (Application Programming Interface) that responded with a random greeting and deployed that API to the cloud via Microsoft Azure.

This week we will create a prototype of a Web application that will consume the output of our API and display it for the user, along with one or two simple controls. We won't be connecting to the API yet, but will instead be designing our user interface layout using a combination of ASP.NET Layouts and Bootstrap.

In the interests of simplicity, we're going to have our code generator do most of the work for us.

* Use the aspnet generator to create a Web Application Basic called GreetingUI
* Open the project in VS Code.
* Delete the following files:

Views/Home/About.cshtml

Views/Home/Contact.cshtml

Views/\_ViewImports.cshtml

Views/\_ViewStart.cshtml

* In Controllers/HomeController.cs, delete the following lines:

|  |
| --- |
| public IActionResult About()  {  ViewData["Message"] = "Your application description page.";  return View();  }  public IActionResult Contact()  {  ViewData["Message"] = "Your contact page.";  return View();  } |

* Open Views/Shared/\_Layout.cshtml and delete the following:

|  |
| --- |
| <nav 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="sr-only">Toggle navigation</span>  <span class="icon-bar"></span>  <span class="icon-bar"></span>  <span class="icon-bar"></span>  </button>  <a asp-area="" asp-controller="Home" asp-action="Index" class="navbar-brand">GreetingUI</a>  </div>  <div class="navbar-collapse collapse">  <ul class="nav navbar-nav">  <li><a asp-area="" asp-controller="Home" asp-action="Index">Home</a></li>  <li><a asp-area="" asp-controller="Home" asp-action="About">About</a></li>  <li><a asp-area="" asp-controller="Home" asp-action="Contact">Contact</a></li>  </ul>  </div>  </div>  </nav> |

* Open Views/Home/Index.cshtml and replace the contents with this:

|  |
| --- |
| @{  ViewData["Title"] = "Home Page";  Layout = "\_Layout";  } |

* Build and run your application with dotnet run and open your browser to <http://localhost:5000>.
* Your code should compile without errors and your browser will show a blank page with a copyright notice.
* Now that we have a blank canvas we can begin.

# Bootstrap Basics

The Bootstrap framework is a free library of styles, scripts and templates that you can easily incorporate into your Web pages and apps. You can get a nice, responsive design that looks good on different devices without a lot of code.

Bootstrap uses a grid-based layout system. The grid consists of a series of rows that span the page. Each row can be up to twelve columns wide. So if you want a twelve column row, you would use:

|  |
| --- |
| <div class="row">  <div class="col-md-12"></div> </div> |

A row with three equal columns:

|  |
| --- |
| <div class="row">  <div class="col-md-4"></div>  <div class="col-md-4"></div>  <div class="col-md-4"></div> </div> |

You can even have unequal size columns:

|  |
| --- |
| <div class="row">  <div class="col-md-2"></div>  <div class="col-md-9"></div>  <div class="col-md-1"></div> </div> |

As long as the columns add up to twelve, you're fine.

Rows need to be placed in a *container*. A container is an invisible box that holds your site's contents. Containers can be nested, but that's not usually required.

So, if we wanted to put those three rows above into a container, we'd use:

|  |
| --- |
| <div class = "container"> <div class="row">  <div class="col-md-12"></div> </div>  <div class="row">  <div class="col-md-4"></div>  <div class="col-md-4"></div>  <div class="col-md-4"></div> </div>  <div class="row">  <div class="col-md-2"></div>  <div class="col-md-9"></div>  <div class="col-md-1"></div> </div>  </div> |

# Bootstrap Hands-On

It would help if we went ahead and got some hands-on experience rather than going into a lot of detail on Bootstrap. Our design is going to be as simple as we can make it since this is a very simple app..

In Index.cshtml, add the following at the end of the file:

|  |
| --- |
| <h1>Hello World!</h1> |

Take a look at your Web application. (Use dotnet run if you need to start the server, otherwise just refresh the page)

You should see Hello World! in nice, big, bold letters.

Now let's put a container around it.

|  |
| --- |
| <div class="container"> <h1>Hello World!</h1> </div> |

Now when you refresh the page, you shouldn't see a lot of change, but you might notice that your greeting shifted slightly to the right. This is because the container sets up a small margin around your Web content.

Let's make your greeting really stand out by using a Bootstrap style element called a *jumbotron*. This is a large, grey box with rounded corners that you use to make certain elements on your page stand out. It's great for things like logos and headlines.

A jumbotron needs a container so we'll put it inside the one we already have, along with our greeting:

|  |
| --- |
| <div class="container">  <div class="jumbotron">  <h1>Hello World!</h1>  </div>  </div> |

Refresh the page and you'll see a dramatic change. Your greeting is larger (a jumbotron automatically makes the fonts inside it bigger) and is inside a gray box the extends the width of your container.

Now let's center our text by modifying the <h1> tag:

|  |
| --- |
| <h1 class="text-center">Hello World!</h1> |

Refresh and make sure your greeting is centered in the jumbotron.

So far so good. Now let's add a row under the jumbotron where we can put the local information for our greeting as well as have a place for a nice image. So we'll need a row with a four column section and an eight column section:

|  |
| --- |
| <div class="row">  <div class="col-md-4">  </div>  <div class="col-md-8">    </div>  </div> |

In the four column section, we'll put our locale ID:

|  |
| --- |
| <div class="col-md-4">  <h2>Locale: en-us</h2>  </div> |

(Remember, this is just a placeholder. We will be pulling the greeting string and the local code from our API.)

In the second section, we'll place an image of a city. (This could be stored locally but for convenience we'll use a Creative Commons licensed image from <http://lorempixel.com/>)

|  |
| --- |
| <div class="col-md-8">  <img src="http://lorempixel.com/400/200/city" alt="">  </div> |

This time when you refresh, you'll notice that it takes a few moments longer since you're fetching the image file from another site. For better performance, consider storing your images locally in your application root folder.

The final element we need to add is a button that the user can press to change the greeting on the page. Since we're just doing layout, our button won't be connected to anything for now.

First let's create a row with three sections of four columns each, just under our Locale row:

|  |
| --- |
| <div class="row">  <div class="col-md-4"></div>  <div class="col-md-4"></div>  <div class="col-md-4"></div>  </div> |

Finally, we add a button to the center row so it's nice and centered on our page. We'll also use one of the Bootstrap button styles btn-primary:

|  |
| --- |
| <div class="row">  <div class="col-md-4"></div>  <div class="col-md-4"><button type="button" class="btn btn-primary" onclick="">Howdy!</button></div>  <div class="col-md-4"></div>  </div> |

When we refresh it this time, the button looks too close to the bottom of our locale row. So let's add some padding to our image and locale elements to give us some breathing room:

|  |
| --- |
| <div class="row">  <div class="col-md-4">  <h2 style="padding:50px;">Locale: en-us</h2>  </div>  <div class="col-md-8">  <img style="padding:50px;" src="http://lorempixel.com/400/200/city/1" alt="">  </div>  </div> |

Now it looks a bit nicer. It's not very sophisticated at this point but it gave us a decent introduction to Bootstrap styles plus it's good enough for now. There's a lot more to Bootstrap so we can always do a bit of research and see what looks good.

* Save your work.
* Stop the server with <Ctrl>-C.
* If you haven't already, commit your code to your local git repository.

# Summary

When designing a user interface, it's good practice to do a sample layout so you can get an idea of what the final product will look like. This also lets you consult with users, clients or other team members and let them provide feedback until you have a UI that they'll be happy with.

You can create a prototype UI using almost any drawing tool, from a paper and pencil to a technical drawing program. There are even software packages that are specifically for creating UI prototypes, complete with menus, buttons and other interface elements.

Since we're designing a Web UI, it made sense for us to use a sample Web application template so we could experiment directly with the code we'll need to use in our final design. CSS style frameworks like Bootstrap make it much easier to lay out your Web page elements in a nice-looking design.