**CS 504 – Software Engineering**

**HOP03 – Blazor Page Application – Working with Components**

10/05/2020 Developed by Kim Nguyen

10/19/2020 Reviewed by Spencer (Yu Che) Liu

Center for Information Assurance (CIAE) @City University of Seattle (CityU)

 Icon

Description automatically generated

**Before You Start**

* Version numbers may not match with the most current version at the time of writing. If given the option to choose between stable release (long-term support) or most recent, please choose the stable release rather than beta-testing version.
* This tutorial targets Windows users and MacOS users.
* There might be subtle discrepancies along the steps. Please use your best judgement while going through this cookbook style tutorial to complete each step.
* For your working directory, use your course number. This tutorial may use a different course number as an example.
* The directory path shown in screenshots may be different from yours.
* If you are not sure what to do or confused with any steps:
  1. Consult the resources listed below.
  2. If you cannot solve the problem after a few tries, ask a TA for help.

**Learning Outcomes**

Students will be able to:

* Further understanding of Blazor application, and Single Page Application.
* Further understading of Blazor app structure.
* Adding components to Blazor Application
* Modifying components of Blazor Application.

**Resources:**

* Microsoft | ASP .NET - <https://dotnet.microsoft.com/apps/aspnet>

**Blazor Pages Web App – Structure breakdown**

Last week, we have learned about Blazor App - a newer, modified version of Razor Pages (we created in Module 1). Let’s dive deeper into the project structure:

Text

Description automatically generated Text

Description automatically generated

In the two pictures above, the one on the left is from Week 1 – Razor Pages. The one on the right

is from Week 2 – Blazor App. As we can see, in the Blazor App structure, we have 2 new files:

\_Imports.razor and App.razor. In this HOP, let’s focus on the Imports.razor file.

.razor files define the UI components of the app, and the \_Imports.razor file act as a connection

between the .razor files and the rest of the project. This \_Imports.razor file contains the

dependencies used in the project. By default, when we generate a Blazor App, there are some

commonly used dependencies are auto-added for us.

Simply speaking, in Blazor App, we have components of Razor pages already combined and

imported, making it easier to deal with handling web UI.

This is one of the reasons why Blazor is considered to be the improved or simplified version of

Razor Pages Application.

More about Blazor App: <https://www.telerik.com/blogs/difference-between-blazor-vs-razor>

**Customize Blazor Web Application:**

Last week, we have created the Blazor Application, this week, we will continue to customize and learn more about this application. **Please move the Blazor App folder from “Module 2” folder to “Module 3”.**

Then, let’s open the Blazor App project in VSCode.

1. Run the application by typing the following command into VSCode terminal:

dotnet watch run

You should see the following messages on the terminal:

Graphical user interface, text, application

Description automatically generated

Ctrl + click on the url (http://localhost:5000) to view the website:

Graphical user interface, text, application, email

Description automatically generated

1. Add a component.

Modify your index.razor file to add the Counter component to the home page:

Graphical user interface, application

Description automatically generated

Save the changes by pressing Crtl + S, on the browser, reload the page to see the difference:

Graphical user interface, text, application, chat or text message

Description automatically generated

*If you are using Google Chrome, you might experience some unexpected errors, if so, try a different browser and it should work.*

This is how easy it can be to build we UI using Blazor! Feel free to test the new Counter component we just added.

1. Modify a component

The Counter component we just added was defined in the Counter.razor file, open Counter.razor to see:

Graphical user interface, text, application

Description automatically generated

Based on the code, the counter starts counting at 0, and each time a user clicks on the “Click me” button, the counter will add 1 to the count.

Now, let’s modify the counting rule, so that each time the user clicks, the counter will add 10 instead of 1. Modify your Counter.razor file to match the following:

Text

Description automatically generated

Then, modify the Index.razor file to match the following:

A picture containing graphical user interface, text, application

Description automatically generated

Save changes, reload the web page, test the “Click me” button again, you should see after one click, the Current count gets added 10:

Graphical user interface, application

Description automatically generated

The Index component now has its own counter that increments by ten each time the Click me button is selected. The Counter component (Counter.razor) at /counter continues to increment by one.

**Challenge 1:**

**Add another button to the home page, name it “Click me to see minus”, so that each time a user clicks on this button, the Current count is minus by 1.**

Expected result:

Graphical user interface, text, application

Description automatically generated

**Challenge 2:**

**Allow users to input any number, then pass that user input to the method of the “Click me to see minus” button you created in the previous challenge. For example, if a user entered 10, the Current Count will minus by 10.**

Expected result:

Graphical user interface, text, application, chat or text message

Description automatically generated

***Hint: Use data binding***

**Push your work to GitHub**

Open the terminal from the VSCode by hitting the control + ~ key, make sure you are in the right path, for example: /Desktop/CS504/HOP03-YourGitHubUsername/Module 3

Type the following command:

git add . (to copy all changes you have made)

git commit -m “Submission for Module 3 – Your Name” (To add a message to your submission)

git push origin master (to upload your work to Github)