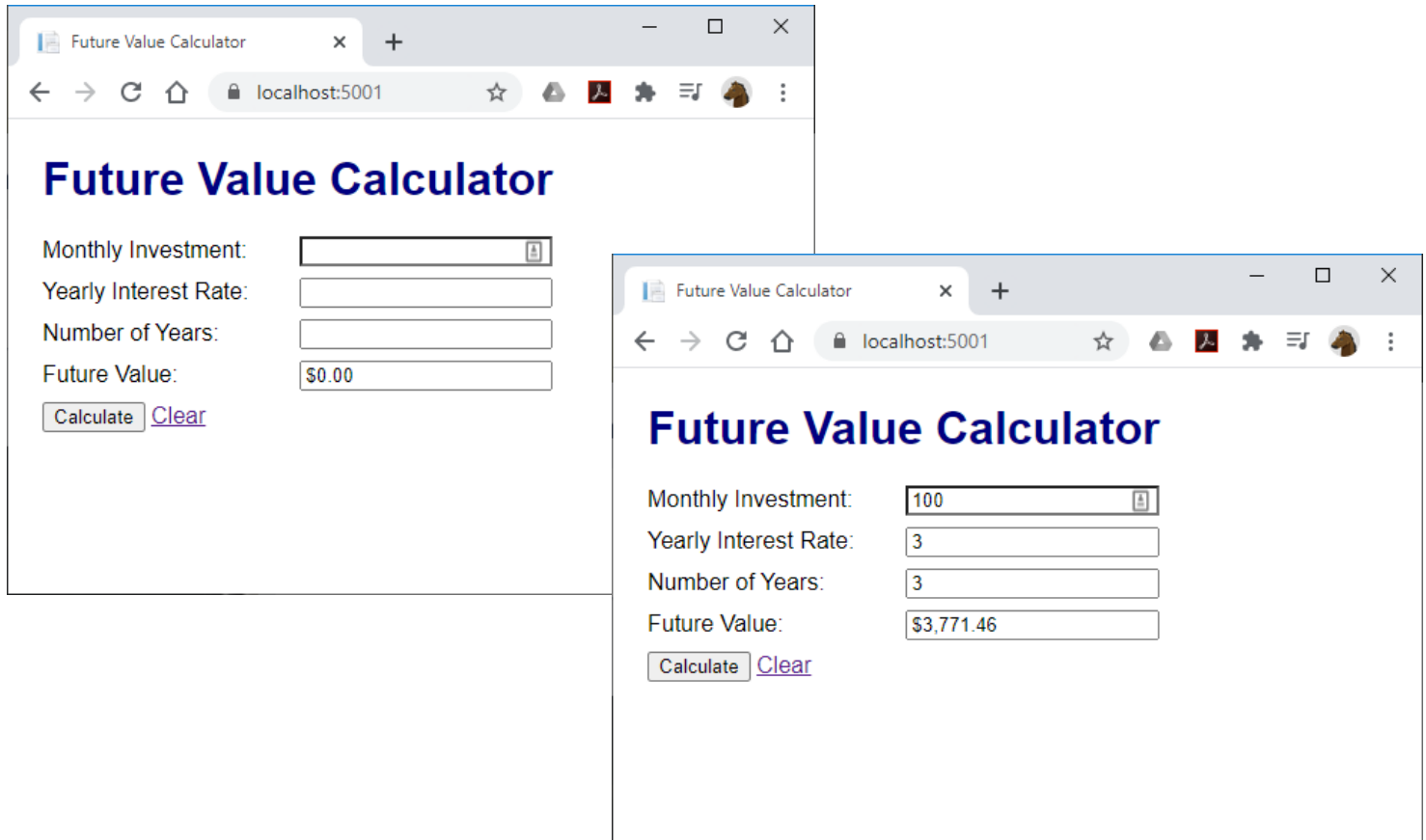


Practice 2-1 Build the Future Value app using the MVC template

This exercise guides you through the development of the Future Value app presented in this chapter. This gives you some hands-on experience using Visual Studio to build a web app.



Create and set up a web app using the MVC template

1. Start a web app that's based on the MVC template. Use a project name of Ch02FutureValue and a solution name of Ch02FutureValue and store it in the directory where you are keeping your files for class.
2. Delete all the files inside the Controllers, Models, and Views folders (including the files inside the Views/Home and Views/Shared folders), but don't delete the Home and Shared folders themselves.
3. Add a controller named HomeController to the Controllers folder and modify it so it contains the code from figure 2-4.
4. Add a new empty Razor view named Index to the Views/Home folder and modify it so it contains the code from figure 2-5.
5. Edit the Startup.cs file so it contains the code from figure 2-6.
6. Select the Kestrel server by selecting the name of the project (FutureValue) from the Start button drop-down list.

7. Press Ctrl+F5 to run the app. This should start the default web browser and display the Home/Index view, including the data that the HomeController stored in the ViewBag.

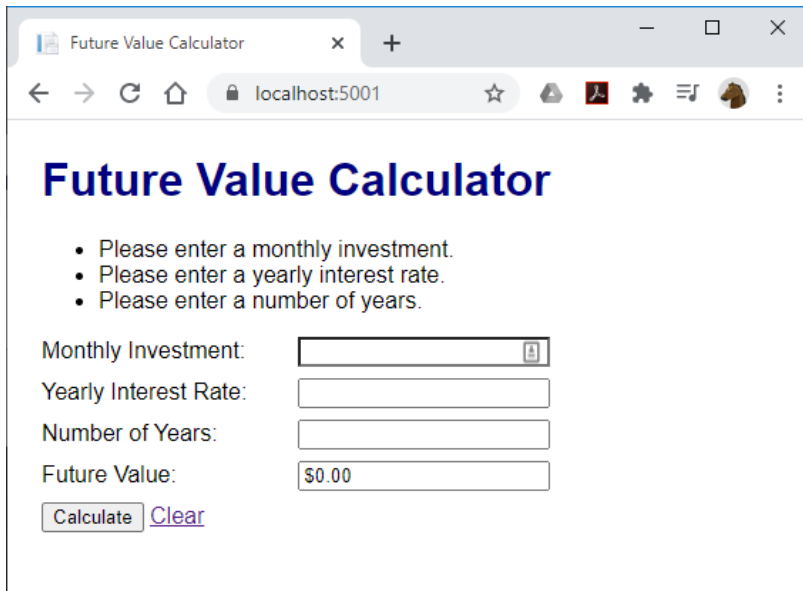
Add the model, Razor view imports page, and a strongly-typed view

8. Add a class named FutureValueModel to the Models folder and modify it so it contains the code from figure 2-9.
9. Add the Razor view imports page to the Views folder and modify it so it contains the code shown in figure 2-10.
10. Modify the code of the Home/Index view so it contains the code from figure 2-11. Make sure to include all the CSS style rules from figure 2-14 within the <style> element.
11. Modify the HomeController class to handle both GET and POST requests as shown in figure 2-12.
12. Run the app. If you enter valid data, it should calculate and display a future value. However, if you enter invalid data, you may get unexpected results.

Add the Razor layout and view start, and modify the Razor View

13. Add a custom.css file to the wwwroot/css folder. If necessary, create this folder first. Then, modify it so it contains the CSS style rules shown in figure 2-14.
14. Add a Razor layout named _Layout.cshtml to the Views/Shared folder and modify it so it contains the code shown in figure 2-16. Make sure to include a <link> element that points to the custom.css file.
15. Add a Razor view start named _ViewStart to the Views folder (not the Views/ Shared folder) and modify it so it contains the code shown in figure 2-16.
16. Modify the code in Home/Index view so it contains the code shown in figure 2-16. To do that, you can cut all elements that are already specified by the Razor layout.
17. Run the app. It should work the same as it did before

Add data validation to the Future Value app



Future Value Calculator

- Please enter a monthly investment.
- Please enter a yearly interest rate.
- Please enter a number of years.

Monthly Investment:

Yearly Interest Rate:

Number of Years:

Future Value:

[Clear](#)

18. Modify the FutureValueModel class so it specifies the Required and Range attributes as shown in figure 2-18.
18. To do that, you must use nullable types for the properties and the method.
19. Modify the HomeController class so it checks for invalid data as shown in figure 2-19.
20. Modify the Home/Index view so it displays a summary of validation messages as shown in figure 2-19.
21. Run the app. It should work correctly if you enter valid data, and it should display appropriate messages if you enter invalid data.

Turn in the Future Value app

1. Once you are satisfied with your application, close Visual Studio, compress the Future app at the root level, and rename the compressed folder using the following naming convention:

FirstInitial_LastName_Ch02FutureValue

For example:

M_Korzaan_Ch02FutureValue
2. Upload the renamed, compressed folder to the appropriate Dropbox in D2L.