Carson Perry

CST-247 Bill Hughes  3/15/2021

Activity 2

Graphical user interface, text, application

Description automatically generated

This demonstrates creating a default index page people will land on with just the directory name ‘products’ in the url. The controller automatically calls the index method when no argument is given past ‘/products/’ or ‘/products/index’. You can change this by changing the endpoints pattern in the startup.cs file

Graphical user interface, text, application

Description automatically generated

This is demonstrating returning a string from a controller, rather than returning an already made view. This can be really useful for things such as debugging, where you can print out error codes or log information onto the page itself.

Graphical user interface, text, application

Description automatically generated

This demonstrates a typical controller sending the user to a page after receiving a url.

Graphical user interface, text, application, website

Description automatically generated

This demonstrates displaying variable data, set within the controller, on the view itself using ViewBags.

Some of the key concepts here include calling methods within a controller, which then return a view that is shown on screen. That one concept itself is the View and Controller part of the MVC (Model, View, Controller) design pattern. Another concept would be that the controllers can return more than just views, such as strings or JSON data. When returning a view, it returns the file that has the name of the view specified.

Graphical user interface, text, application, website

Description automatically generated

This shows a working Login page. Working as in that it loads and there are no errors in rendering the html.

Graphical user interface, text, application

Description automatically generated

This shows a successful login using a hardcoded “if username= x etc.” to process the user model.

Graphical user interface, text, application

Description automatically generated

This shows the failed version of the hard coded controller logic from the last picture.

Graphical user interface, text, website

Description automatically generated

Login success screen where the login controller uses the newly created business service that contains a list of possible user credentials.

Graphical user interface, text, application

Description automatically generated

Login failed screen using the updated controller from the last screenshot.

Graphical user interface, text, application

Description automatically generated

A screenshot of a successful login with the business service using the newly created DAO to validate user credentials.

# Key Concepts:

Views send controllers info mainly in the form of models. There is more to the HTTP requests, but the data that we mainly manipulate is from Models. These models are then able to be manipulated to our heart’s content in our business services, which our controllers send the model into. Each of these are encapsulated in their own classes and folders, separated from each other by their purposes. Having a class with the sole purpose of getting specific information from the database using SQL commands, helps protect the database and ensure that we’re getting the correct information that we want.

# Part 3:

Graphical user interface, application

Description automatically generated

An appointment form with data validation that pops up before having to submit the form.

Graphical user interface, application

Description automatically generated

An updated appointment form with further data constraints for the programming challenge.

# Key Concepts:

You can create validation constraints for a model within the model class itself, rather than have to make the same constraints on any form that uses the same model. This makes it easier to use the same model elsewhere on the site, while still requiring the same type and size of data for each property.

Github Link: <https://github.com/arsonull/arsonull.github.io/tree/master/Sem4/C%23/act2>