Exercise

ASP.NET MVC - Controllers

April 2019

The MusicStore web application (Part 1)

Complete the steps below.

Step 1 – Create the MusicStore appliction

Create a new ASP.NET MVC project called “MusicStoreCore” in a solution named “MusicStoreCore” **using the “Web Application (Model-View-Controller)” template.**

Add a Nunit test project (.Net Core) “MusicStoreCore.Tests”. Add the “Moq” nuget package and the “Microsoft.AspNetCore.Mvc.ViewFeatures” package to the project.

In the next steps we will be working with the HomeController.

Step 2 – Make some actions return a simple content string

- Make sure the Index and About actions return a string (ContentResult). The actions should return a string containing the name of the controller and the name of the action method (e.g. ‘’Home:Index”). This information should be retrieved from de “ControllerContext”. Do this in a Test Driven way.
    - Add a test class HomeControllerTest.
    - Add a unit test “Index_ReturnsContentContainingControllerNameAndActionName”
    - Add a unit test “About_ReturnsContentContainingControllerNameAndActionName”
    - Tips
        - Normally MVC passes the RouteData to the controller by setting the “ControllerContext” property of the controller. In a unit test you must do this yourself, if needed. Setup the controller with a ControllerContext that contains some RouteData (“controller” and “action”)
        - Refactor. Certainly after writing the second test.
        - Stay DRY (don’t repeat yourself)
- Add a Details action that accepts an id as parameter and returns a string (ContentResult). The action should return a string containing the name of the controller, the name of the action method and the value of the id parameter (e.g. ‘’Home:Details:23”). Do this in a Test Driven way.
    - Add a unit test “Details_ReturnsContentContainingControllerNameActionNameAndParamName”

Step 3 – Define a new Route

Add a route named “Search”. Using this route it should be possible to search for albums of a genre. The “Search” route should map http requests to a “Search” action in the HomeController. The “Search” action will accept a “genre“ (string) parameter. In the next step we will create the “Search” action.

In this step the “Search” route should be added to the RouteCollection via the RouteConfig.

Urls that match the “Search” route look like this:

- /SearchMusic/Rock (Shows a list of rock albums)
- / SearchMusic /Jazz (Shows a list of jazz albums)
- …

Also make sure that “Rock” is the default value for “genre”

Step 4 – Create the Search action method

The “Search” action method should return the following ActionResults depending on the “genre”

- Genre = “Rock“ -> Permanent redirect
    - To https://www.youtube.com/watch?v=v2AC41dglnM or one of your favorite rock video’s
    - Add a unit test “Search_Rock_PermanentRedirect”
- Genre = “Jazz” -> Temporary redirect to “Index” action of HomeController
    - Add a unit test “Search_Jazz_RedirectToIndexAction”
    - Tips
        - Use the “RedirectToAction” method of the “Controller” class. This method returns an instance of “RedirectToActionResult”
- Genre = “Metal” -> Temporary redirect to “Details” action of HomeController
    - Pass an “id” with a random value
    - Add a unit test “Search_Metal_RedirectToDetailsActionWithARandomId”
- Genre = “Classic” -> Return the content of “site.css” (in “wwwroot\css” folder of project)

- Add a unit test “Search_Classic_ContentOfSiteCssFile”
    - Use the “File” method of the “Controller” class to return a FileContentResult that contains the bytes of the file, the content type “text/css” and the download name “site.css”
    - Define an “IFileProvider” interface with a method “byte[] GetFileBytes(string relativePath);”. The controller should use the provider to retrieve the bytes of the file.
- Create a “HostFileProvider” class that implements “IFileProvider”.

```
public class HostFileProvider : IFileProvider
{
    private readonly IHostingEnvironment _environment;

    public HostFileProvider(IHostingEnvironment environment)
    {
        _environment = environment;
    }

    public byte[] GetFileBytes(string relativePath)
    {
        var fullPath = Path.Combine(_environment.ContentRootPath, relativePath);
        return File.ReadAllBytes(fullPath);
    }
}
```

- Register “HostFileProvider” in the dependency injection container (as a singleton)

- Genre = any other genre -> Show the name of the controller, action and value of the “genre” parameter (as Content)
    - Add a unit test “Search_UnknownGenre_ReturnsContentContainingControllerNameActionNameAndGenreParameter”