Lab: Building a Resilient ASP.NET MVC Core Web Application

# Scenario

The senior developer has asked you to implement the following functionality in your Photo Sharing web application.

* Any visitor of the application, including anonymous users, should be able to mark a photograph as a favorite.
* If a user has marked a favorite, a link should be available to display the favorite photo.
* Favorite photos should be displayed in the slideshow view.

# Objectives

After completing this lab, you will be able to:

* Store a setting for an anonymous or authenticated user in session state.
* Check a user preference when rendering an action link.
* Render a webpage by checking state values in the application.

Estimated Time: 45 minutes

# Exercise 1: Creating Favorites Controller Actions

## Scenario

You have been asked to build functionality that stores the favorite photos of the visitors in the session state of the web application. After users add photos to their favorites, they will be able to view a slideshow of all the photos they selected as favorites.

In this exercise, you will:

* Setup Session
* Create the Favorites Slideshow action.
* Create the Add Favorite action.

The main tasks for this exercise are as follows:

1. Setup and extend the Session
2. Create the Favorites Slideshow action.
3. Create the Add Favorite action.

## Task 1: Setup and extend the Session.

1. In the NuGet Package Manager, install a new dependency by using the following information:

* Name: “Microsoft.AspnetCore.Session”
* Version: make sure that the version matches the rest of your dependencies

1. In the ConfigureServices method of the Startup class, invoke AddSession extension method of the services collection
2. In the Configure method of the Startup class, invoke the UseSession extension method of the IApplicationBuilder object.
3. In your project folder, add a new Session folder
4. In the Session folder, add a static class ISessionExtensions
5. In the ISessionExtensions class
   1. Declare a constant string with a name of “favoritesKey” and a value of “Favorites”
   2. Create an extension method for the ISession interface with the following signature  
      public static List<int> GetOrCreateFavoriteIds(this ISession session)
   3. In the GetOrCreateFavoriteIds method
   4. Create a List of integers called favoriteIds. Do not instantiate it yet.
   5. Invoke the Get method of the session object passing the favoritesKey constant and save the result in a favoriteIdsBytes byte array
   6. Check if the result is not null and the length is greater than zero and if this is true
      1. Invoke the GetString method of the System.Text.Encoding.UTF8 class passing the bytes array and save the result in a string variable called json
      2. Invoke the DeserializeObject method of the JsonConvert class passing the json variable as a parameter. Save the result into the favoriteIds list
   7. Else, initialize the list with a new empty List of integers
   8. Return the list
   9. Create another extension method for the ISession interface with the following signature  
      public static void AddFavoriteId(this ISession session, int id)
   10. In the AddFavoriteId method
   11. Invoke the GetOrCreateFavoriteIds extension method of the session object and save the result in a favoriteIds variable of type List of integer
   12. Add the id to the list if not already present
   13. Invoke the SerializeObject method of the JsonConvert class saving the result in a string variable named json
   14. Invoke the GetBytes method of the System.Text.Encoding.UTF8 class passing the json variable as a parameter and saving the result in a byte array named serializedResult
   15. Invoke the Set method of the session object passing the favoritesKey constant and the serializedResult variable as parameters

## Task 2: Create the Favorites Slideshow action.

1. In the PhotosController.cs code file, add a new action by using the following information:

* Scope: public
* Return type: async Task<IActionResult>
* Name: FavoritesSlideShow

1. In the FavoritesSlideShow action, create and instantiate a new List of Photo objects named favPhotos.
2. Create a new List of integers named favoriteIds. Set this list to be equal to the HttpContext.Session.GetOrCreateFavoriteIds extension method
3. Set the favPhotos variable equal to the following linq query:  
   await (from p in \_context.Photos  
   where favoriteIds.Contains(p.Id)  
   select p).ToListAsync()
4. At the end of the FavoritesSlideShow action, return the SlideShow view and pass the favPhotos list as a model class.
5. Save all the changes.

## Task 2: Create the Add Favorite action.

1. Add a new action to the PhotoController.cs file by using the following information:

* Scope: public
* Return type: IActionResult
* Name: AddFavorite
* Parameter: an integer named id

1. Invoke the HttpContext.Session.AddFavorietId extension method and pass the id as a parameter
2. Return HTML content by using the following information:

* Method: Content()
* Content: The picture has been added to your favorites
* Content type: text/plain
* Encoding: System.Text.Encoding.Unicode

1. Save all the changes.

**Results**: After completing this exercise, you will be able to create controller actions that store values in the session state of the web application, and retrieve values from the same session state.

# Exercise 2: Implementing Favorites in Views

## Scenario

You have created the necessary controller actions to implement favorite photos. Now, you should implement the user interface components to display a control for adding a favorite. If a user has favorites, you should display a link to the FavoritesSlideShow action.

In this exercise, you will:

* Add an AJAX action link in the Photo Display view.
* Add a link and update the site map.

The main tasks for this exercise are as follows:

1. Add an AJAX action link in the Photo Display view.
2. Add a link and update the site map.
3. Test favorites.

## Task 1: Add an AJAX action link in the Photo Details view.

1. Open the Details.cshtml view for the Photos controller.
2. After the end of the <dl> element, insert a new <div> element with the ID addtofavorites.
3. In the DIV element, include an <a> tag helper by using the following information:

* Text: Add to favorites.
* Include a glyph icon of value glyphicon-heart
* Action: AddFavorite
* Controller: Photos
* Route values: Pass the Model.Id value to the id route parameter.
* Class: btn btn-info
* Ajax Options:
  + data-ajax: true
  + data-ajax-mode: replace
  + data-ajax-update: addtofavorites
  + data-ajax-complete: replaceMenuItem

1. Save all the changes.

## Task 2: Update the link dynamically with JavaScript.

1. In the Details.cshtml view for the Photos controller.
2. Right before the end of the script section add a script tag.
3. In the script tag, declare a javascript function replaceMenuItem.
4. In the replaceMenuItem function, use jQuery to look for an item with an id of favorites-link and set the result into a new variable named favoritesLinkLi.
5. If the length of the variable is 0:
   1. Create a new list item wrapped into a jQuery object and set the result into the favoritesLinkLi variable
   2. Create a new html link wrapped into a jQuery object and set the result into a newly created variable named link.
   3. Set the href attribute of the link to the Url.Action helper using the following information:
      1. Helper: Url
      2. Method: Action
      3. actionName: FavoritesSlideShow
      4. controllerName: Photos
   4. Set the text of the link to Favorite Photos
   5. Append the link object to the favoritesLinkLi object
   6. Append the favoritesLinkLi object to the first unordered list with a class nav.
6. Save all the changes

## Task 3: Add a link to the menu.

1. Under the ViewComponents folder, add a FavoritesMenuItemViewComponent class deriving from ViewComponent
2. Add a public async Task<IViewComponentResult> InvokeAsync() method
3. Invoke the GetOrCreateFavoriteIds extension method of HttpContext.Session object and set the result in a favoriteIds variable.
4. Return the default View if favoriteIds.Count is greater than zero, or a “NoFavorites” view otherwise
5. Under the Views/Shared/Components folder, create a FavoritesMenuItem folder
6. Under the FavoritesMenuItem folder, add a Default.cshtml view
7. In the Default view, add a list item with a “favorites-link” id
8. In the list item, add an <a> tag helper to link to the FavoritesSlideShow action of the Photos controller
9. Under the Views/Shared/Components/FavoritesMenuItem folder, add a NoFavorites.cshtml view
10. Leave the NoFavorites.cshtml view completely empty
11. Open the \_Layout.cshtml file
12. Before the end of the <ul class="nav navbar-nav">, after the last link, add the favorites-menu-item view component
13. Save all the changes.

## Task 3: Test favorites.

1. Start the web application in debugging mode.
2. Add three photos of your choice to your favorite photos.
3. Browse to the home page and click the Favorite Photos link.
4. Stop debugging and close Visual Studio.

**Results**: After completing this exercise, you will be able to:

* Create the user interface components for the favorite photos functionality.
* Test the functionality of the user interface components.

**Question**: In this lab, you stored the list of favorite photos in the session state. While testing, your manager notices that authenticated users lose their favorite photos list whenever they close their browser. Where would you store a list of favorites for each authenticated user so that the list is preserved whenever a user logs on to the web application?