Lab: Building Responsive Pages in ASP.NET MVC Core Web Applications

# Scenario

Your manager has highlighted that the performance of some pages in the application is too slow for a production site.

You want to return images in quick time, for which you decide to configure caching in your application.

The senior developer has also 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 photos.
* Favorite photos should be displayed in the slideshow view.

# Objectives

After completing this lab, you will be able to:

* Configure ASP.NET caches to quickly serve images.
* Implement a Favorites functionality using Sessions

Estimated Time: 60 minutes

# Exercise 1: Configuring the ASP.NET Cache

## Scenario

You have been asked to configure the ASP.NET cache in the Photo Sharing application to ensure optimal performance. Senior developers are particularly concerned that the All Photos gallery might render slowly because it will fetch and display many photos from the database at a time.

In this exercise, you will:

* Use a Distributed Cache to store and quickly retrieve images.
* Use the developer tools in the browser to examine the speed at which image files and pages render with and without caching.

The main tasks for this exercise are as follows:

1. Test the All Photos page with no caching.
2. Configure caching.
3. Use caching in the GetImage action.
4. Retest the All Photos page.

You’re going to serialize your images in the cache in a binary format using a BinaryFormatter. Seen the fact that you don’t need to save the whole Photo model but just the information necessary to create a File (the byte array and the mime type), you decide to create a CachedPhoto class that you will use to store the information in the cache.

You decide to create two extension methods for the IDistributedCache interface to hide the complexity of the serialization and deserialization. This way the code of your controller will be much simpler and the responsibility of dealing with the cache will be of the cache itself. These are the methods you want to create:

1. A SetPhotoAsync, accepting a Photo. This method creates a CachedPhoto from a given Photo, it uses a BinaryFormatter to serialize a given Photo in a MemoryStream, then caches the array of bytes from the memory stream
2. A GetPhotoAsync, accepting an id and returning a Photo. This method retrieves an array of bytes from the cache, then uses a MemorySTream and a BinaryFormatter to deserialize it into a CachedPhoto, then returns a Photo

In the PhotosController, the GetImage method will first try to get the value of the photo from the cache, calling the repository only if the photo could not be retrieved from the cache

Don’t forget to also update your unit tests to pass an instance of a MemoryDistributedCache to the PhotosController constructor. In order to create a new MemoryDistributedCache instance, you need to pass an IOptions of MemoryDistributedCacheOptions instance. You can create that by invoking the Create static method of the Options class.

**Results**: At the end of this exercise, you will create a Photo Sharing application with the Cache configured for caching photos.

# Exercise 2 (if time permits): Using Session for Favorite Photos

## 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 photo has already been marked as favorite, it should show it
* If a user has marked at least one favorite, a link in the navigation bar should be available to display the favorite photos.
* Favorite photos should be displayed in a slideshow view.

In this exercise, you will:

* Setup Session
* Create the Favorites Slideshow action and View.
* Create the Add Favorite action.
* Create a ViewComponent that checks if a given photo is already in Session and renders either an Add To Favorite Button or an indication that the photo is already a favorite
* Modify the Details View and the PhotoGallery to include the new ViewComponent
* Create a ViewComponent that checks if there is any favorite Photo in the Session and renders either nothing or a navigation item to the Favorite SlideShow
* Modify the Layout to include the new ViewComponent

The main tasks for this exercise are as follows:

1. Setup Session and extend Session
2. Create the Favorites Slideshow action and View.
3. Create the AddToFavorites action.
4. Create an AddToFavorites ViewComponent
5. Modify the Details View and the PhotoGallery to include the new ViewComponent
6. Create a FavoritesMenuItemViewComponent
7. Modify the Layout to include the new ViewComponent

You decide to create four extension methods that will be used throughout the application:

1. An AddFavoritePhoto, to store the photo id together with a string key. You will use this method in an action of the Photos controller, invoked whenever a user wants to mark a photo as favorite.
2. A GetFavoritePhotos, to retrieve the list of ids whose keys start with the prefix used during the add phase. You will use this method in a PhotosController method for the slideshow.
3. An IsThereAnyFavorite, that will return true if the session has any key starting with the prefix used during the add phase. You will use this to understand if you have to render the navigation link to the slideshow.
4. An IsFavoritePhoto, returning true if a given photo id is present in the session. You will use this to understand if you have to render an “Add To Favorite” button for each photo of the photo gallery and details views.

You also decide to create ViewComponents to render the (eventual) Add To Favorites button on each photo of the views and the (also eventual) Favorites SlideShow navigation link.

**Result: After completing this exercise, you will be able to use the Session object to store and retrieve the user favorite photos.**

**Question**: In Exercise 1, why was the Request timing for /Photos not reduced for the first request when you configured the output cache for the index action?