Lab: Developing ASP.NET MVC Core Controllers

# Scenario

You have been asked to add a controller in the photo sharing application to manage the Photo model class that you have created in an earlier module. The controller should include actions that respond when users upload photos, list all photos, display a single photo, and delete photos from the application. You should also add an action that returns the photo as a File to show on a webpage.

The members of your development team are new to ASP.NET MVC and they find the use of controller actions confusing. Therefore, you need to help them by adding a component that displays action parameters in the Visual Studio Output window whenever an action runs. You will add an action filter to achieve this.

# Objectives

* After completing this lab, you will be able to:
* Add an MVC controller with code that uses Entity Framework to manage a model.
* Write action filters that run code for multiple actions and log data about controllers and actions.

**Estimated Time**: 60 minutes

# Exercise 1: Add an MVC Controller

## Scenario

In this exercise, you will add an MVC controller that handles photo operations. You will implement the following actions:

* Index. This action gets a list of all the Photo objects and passes the list to the Index view for display.
* Display. This action takes an ID to find a single Photo object. It passes the Photo to the Display view for display.
* Create (GET). This action creates a new Photo object and passes it to the Create view, which displays a form that the visitor can use to upload a photo and describe it.
* Create (POST). This action receives a Photo object from the Create view and saves the details to the database.
* Delete (GET). This action displays a Photo object and returns a view that will request confirmation from the user to delete the Photo object.
* DeleteConfirmed (POST). This action deletes a Photo object after confirmation.
* GetImage: This action returns the photo image from the database as a JPEG file. This method will be called by multiple views to display the image.

The main tasks for this exercise are as follows:

1. Add a Photos controller.
2. Implement the Index action.
3. Implement the Details action.
4. Implement the Create actions for GET and POST HTTP verbs.
5. Implement the Delete actions for GET and POST HTTP verbs.
6. Implement the GetImage action.

TIP: Use Dependency Injection to get hold of the DbContext.

**Results**: After completing this exercise, you will be able to create an MVC controller that implements common actions for the Photo model class in the Photo Sharing application.

# Exercise 2: Optional—Writing the Action Filters in a Controller

## Scenario

Your development team is new to MVC and is having difficulty in passing the right parameters to controllers and actions. You need to implement a component that displays the controller names, action names, parameter names, and values in the Visual Studio Output window to help with this problem. In this exercise, you will create an action filter for this purpose.

This feature is a nice to have. Complete this exercise only if time permits.

The main tasks for this exercise are as follows:

1. Add an action filter class.
2. Initialize a logger using the Dependency Injection system to get hold of a LoggerFactory
3. Add a handler for the OnActionExecuting event to log the name of the controller and action and any other key / value pair in the route data.
4. Register the Action Filter as a service.
5. Register the Action Filter with the Photo Controller.

**Results**: After completing this exercise, you will be able to create an action filter class that logs the details of actions, controllers, and parameters to the Visual Studio Output window, whenever an action is called.

**Question**: Why did you use the ActionName annotation for the DeleteConfirmed action in the PhotoController class?

**Question**: In the lab, you added two actions with the name Create. Why is it possible to add these actions without using the ActionName annotation?