Lab: Implementing APIs in ASP.NET MVC Core Web Applications

# Scenario

Your manager wants to ensure that the photos stored in the Photo Sharing application can be searched by title in a user friendly way, by using a search textbox that can show suggestions as the user types his query. He also wants to ensure that the results are displayed on the same page without refreshing the whole page. To build the frontend, your company decides to use Twitter Typeahead for the search textbox and Handlebars to display the results.

On the backend a RESTful Web API is going to be necessary in order to execute the queries and send the data back to the front end.

A front end developer has already created the view with the search textbox and the template for the results. You will need to create the RESTful service that sends the suggestions for the search textbox and the search results.

# Objectives

After completing this lab, you will be able to:

* Create a Web API by using the new features of ASP.NET MVC Core.
* Add routes and controllers to an application to handle REST requests.

Estimated Time: 30 minutes

# Exercise 1: Adding a Web API to the Photo Sharing Application

## Scenario

The view created by the front end developer expects two sets of data coming from a RESTful service. You have been asked to implement a Web API for the Photo Sharing application to ensure that the page can display the results.

On the view, as the user types on the search textbox, the TypeAhead library (a Twitter JavaScript library) sends the textbox value to the “/api/search/titles/%QUERY” url (where %QUERY is the textbox value). It expects in return an array of strings with all the titles that contain the query value.

When the user presses Enter, jQuery sends the textbox value to the “/api/search/photos/%QUERY” url (where %QUERY is the textbox value). It expects in return an object with the following structure (example):

{

"photos":[

{

"id":7,

"title":"Headland",

"description":"The grass was green, the sky and the water were blue",

"createdDate":"2016-12-14T00:00:00",

"ownerName":"sue@sue.su",

"image":"…” (image data omitted for brevity)

},

{

"id":8,

"title":"New Adventure Works Bike",

"description":"It's the bees knees!",

"createdDate":"2016-12-14T00:00:00",

"ownerName":"bert@bert.be",

"image":"…” (image data omitted for brevity)

}

]

}

In this exercise, you will

* Add a Search Web API controller
* Add the action for the titles suggestions
* Add the action for the search results
* Test the API by using a browser
* Test the Search page

The main tasks for this exercise are as follows:

1. Add a Search API controller.
2. Add the action for the titles suggestions
3. Add the action for the search results
4. Test the Web API with a browser
5. Test the Search page

## Task 1: Add a Search Web API controller.

1. Add a new Web API controller to the PhotoSharingApplication project by using the following information:

* Name: SearchController
* Template: Web API Controller Empty

1. Add a using statement for the following namespace to SearchController.cs:

* PhotoSharingApplication.Data
* Microsoft.EntityFrameworkCore

1. Configure the Route of the Search Controller to be “api/[controller]”
2. Add a new field to the SearchController class by using the following information:

* Scope: private
* Type: ApplicationDbContext
* Name: \_context
* Initial value: none

1. Add a constructor accepting an ApplicationDbContext parameter
2. In the constructor, save the parameter into the \_context field

## Task 2: Add the action for the titles suggestion.

1. Add a new action to the SearchController by using the following information:

* Scope: public
* Return type: IActionResult
* Name: Titles
* Parameters: string title

1. Configure the action Route to be “titles/{title}”
2. In the Titles action, initialize a List of string with the value of a linq query that selects the title property of each photo whose title contains the value of the “title” parameter.
3. Return an ObjectResult with the result of the query.
4. Save all the changes.

## Task 3: Add the action for the search results.

1. Add a new action to the SearchController by using the following information:

* Scope: public
* Return type: IActionResult
* Name: Photos
* Parameters: string title

1. Configure the action Route to be “photos/{title}”
2. In the Photos action, initialize a variable “photos” with the value of a linq query.
   1. The query should search all the photos whose title contains the that selects the title property of each photo whose title contains the value of the “title” parameter.
   2. The query should return a list of anonymous objects with the following structure:
      1. id = photo.Id
      2. title = photo.Title
      3. description = photo.Description
      4. createdDate = photo.CreatedDate
      5. ownerName = photo.Owner.UserName
      6. image = $"data:{photo.ImageMimeType};base64,{Convert.ToBase64String(photo.PhotoFile)}"
3. Return an ObjectResult with an anonymous object with a photos property containing the result of the query.
4. Save all the changes.

## Task 4: Test the Web API with a browser.

1. Start that web application in debugging mode.
2. Request the titles with title “es” by using the Web API. Check that the returned JSON result is the following:  
   ["Blackberries","Pebbles","Ripples on a river","A Track between the trees"]
3. Request the photos with title “bla” by using the Web API. Check that the returned JSON result is the following:   
   {"photos":[{"id":4,"title":"Blackberries","description":"Some were not ripe yet","createdDate":"2016-12-14T00:00:00","ownerName":"fred@fred.fr","image":"…”} (image data omitted for brevity)
4. Close Visual Studio and stop debugging.

## Task 5: Test the Web API with the Search page.

1. Start that web application in debugging mode.
2. On the navigation bar on top of the home page, click on Search Photos
3. On the Search Textbox, type “es”
4. Check that the suggestion show on a dropdown list
5. Press enter
6. Check that the pictures show under the search result.

**Results**: At the end of this exercise, you will be able to create a simple Web API for an ASP.NET MVC Core web application.

**Question**: How do the API actions you added to the SearchController controller in Exercise 1 differ from other actions in MVC controllers?