Lab: Implementing and using WebApi

# Scenario

You have been asked to implement the comments functionality, as follows:

* When viewing the details of a photo, a user should also see the existing comments for the given photo.
* Without leaving the page, the user should also be able to add a new comment
* The new comment should appear in the list of existing comments without refreshing the whole page.

To implement this feature you decide to create a RESTful service to manage the comments, then write a JavaScript client that will handle the user interactions, invoke the RESTful API and dynamically build the user interface client side.

# Objectives

After completing this lab, you will be able to:

* Create and implement a RESTful service through an Api Controller
* Create and implement a JavaScript client that makes use of the RESTful service

Estimated Time: 60 minutes

# Exercise 1: Create a RESTful service

## Scenario

You have been asked to implement the comments feature.

* Get the existing comments for the given photo.
* Add a new comment
* Get an existing comment by its id

In this exercise, you will:

* Create an infrastructure layer to manage comments by implementing a repository
* Create a CommentsController as an ApiController
* Implement the actions to create and retrieve comments and configure their routes.

**Results**: At the end of this exercise, you will create a CommentsController RESTful service.

# Exercise 2: Create a JavaScript client

## Scenario

To implement this feature you decide to use the following pattern.

Repository

ViewModel

View

* The View contains the UI, such as input boxes and buttons for the creation of new comments, and placeholders and a template to show the existing comments. The user interacts with the View.
* The Repository is pure JavaScript. It is responsible of the communication with the RESTful API, and serialization and deserialization of comments.
* The ViewModel is a JavaScript class that:
  + During startup, invokes the Repository to get the comments and dynamically builds the View with the result
  + Handles the click event of the view, invokes the repository to add a comment and refreshes the UI with the result

In this exercise, you will:

* Modify the Photo Details view to include
  + An empty container for the comments
  + An HTML template for a comment
  + HTML input fields and a button to add a comment
  + Javascript code to create an instance of a javascript ViewModel class
* Create a JavaScript commentsRepository object, with methods to get comments given a photo id and to add a comment
* Create a JavaScript ViewModel class that, in its constructor
  + Invokes the repository to get the comments for the given photo id
  + Updates the user interface by dynamically creating copies of the template on the view and adding them to the comments container on the view.
  + Adds an event listener for the click of the add button on the view, so that when the user clicks it
    - the values on the view input boxes are used to set the properties of a new comment object that is sent as a parameter to the repository
    - the UI gets updated with the result

**Result: After completing this exercise, you will be able to build a JavaScript client that fetches data from an API service and uses partial page updates to dynamically build the HTML view client side.**