Web Api & Rest Exercises

Scenario

You have been asked to create a web-based restaurant application for your organization's customers. To do this you need to create a page showing all the restaurant branches, enable users to order a reservation for a selected restaurant branch, add a wanted ad page, and allow submitting an application to a selected job. You will create a server-side Web API application and a client-side ASP.NET Core MVC application. In the client-side application, you will call the Web API actions by using **HttpClient**.

Exercise 1: Adding Actions and Calling them by using Postman

Scenario

In this exercise, you will first add a controller and an action to a Web API application. You will then run the application and view the outcome by using Postman. After that, you will add a controller and an action that gets a parameter. Finally, you will add a Post action to the Web API application.

Tasks (in the Server project):

- Add a new **RestaurantBranchesController.**
- Create a method that will return a list of **RestaurantBranch** called Get
- Add a new **ReservationController** and:
  - Create a method called **GetById** that will return an **OrderTable** object, handle the case when there is no object with the given id (status codes !!).
  - Add a new method called **Create** with an object **OrderTable** type parameter and add it in the collection (in the **RestaurantContext**). Make sure to return the appropriate status code / object. Try to validate your data before adding it in the list (a required Id, string with a min and max length, date format, phone number format etc.)
  - Add a delete method to delete **OrderTable** objects.
  - Bonus: add also a PUT method to edit **OrderTable** objects.
- Test your API with a browser/Postman. You should be able to check the following routes:
  - http://localhost:[port]/api/RestaurantBranches
  - http://localhost:[port]/api/Reservation/1 for GET and DELETE
  - http://localhost:[port]/api/Reservation and add a new **OrderTable** object then retrieve it using the **GetById** method.

Extra Bonus : Add at least one **PATCH** method for **OrderTable**

Create a **custom pipeline** and log all requests that created a new **OrderTable** (logs can be saved in a file).
https://docs.microsoft.com/en-us/aspnet/core/fundamentals/middleware/write?view=aspnetcore-3.1

Exercise 2: Calling a Web API by Using Server-Side Code

Scenario

In this exercise, you will call the Web API you developed in the previous exercise by using the **HttpClient** class. To do this, you will first register the **IHttpClientFactory** service in the **Startup.cs** file. You will then create an MVC controller and use the **HttpClient** class to call a Get action in the Web API. After that, you will create another MVC controller and use the **HttpClient** class to call a Post action in the Web API. Finally, you will add an action to the MVC controller in which you will use the **HttpClient** class to call a Get action in the Web API that gets a parameter.

Tasks (in the Client project):

- Register the **AddHttpClient** in **Startup.cs** (read about it here - https://docs.microsoft.com/en-us/aspnet/core/fundamentals/http-requests?view=aspnetcore-3.1)

- Create a new MVC Controller-Empty in the Controllers folder called **RestaurantBranchesController**

    - Create a method called **Index** (should already be there), from **IHttpClientFactory** you now should initialize a new **HttpClient** (check step one). Now use the client to call your first Server project API at route "api/RestaurantBranches", use a variable of type **HttpResponseMessage** to get the response from **HttpClient** and if it succeed deserialize the response's content in a variable (assuming is a list of **RestaurantBranch**). Based on the response (success or fail) return Controller's View (should create a blank View called **Index.cshtml** in Views folder) or an **Error** View.

- Create a **ReservationController** as the previous one and:

    - Create a new method to fetch restaurant branches from the Server as in the previous example and save it in a **ViewBag**

    - Add a new **Create** method and use its View along with an **OrderTable** model. The Create View will need a list of **RestaurantBranches** (use first step, check **Create.cshtml** line 14)

    - Now create a new method **CreatePostAsync** (it will be used by the View to submit the new **OrderTable**). Above the **CreatePostAsync** action with the ActionName attribute, pass "**Create**" as a parameter to the ActionName attribute.

    - Call the Server route that adds a new reservation (/api/Reservation). If everything was ok redirect the user to the Action **ThankYouAsync** and pass new { orderId = order.Id} as parameters to the RedirectToAction method. If reservation cannot be made return an Error View.

    - For the last part. Create the action **ThankYouAsync** with parameter orderId and with a **HttpClient** fetch the corresponding reservation from the Server "(/api/Reservation/ +orderId)".

    - Now you should be able to fetch **RestaurantBranches**, create and check the new reservation in the UI.

Additional information:

You should focus on **REST principle, status codes**, correct mapping of the HTTP verbs and way you validate your models.