

Rest Exercise 1: First REST controller

In this exercise you must program a simple REST controller using the model and repository classes you already have (from Programming).

Hint: If you don't already have Postman installed, it's a good idea to install it now

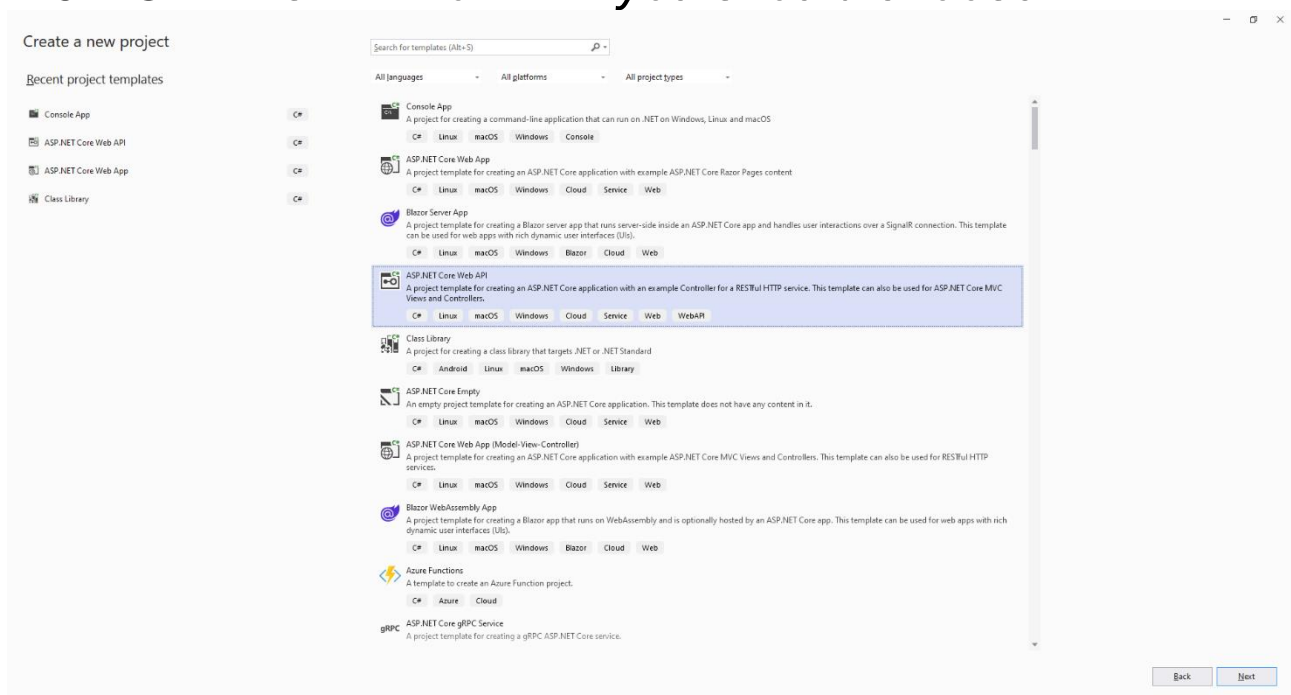
I recommend the desktop version of Postman, it makes it easier to test your projects running on your computer (see [Tools](#)).

Make the project

Open Visual Studio and create a new project

Choose **ASP.NET Core Web Api**

NOTICE: It is API not APP you should choose

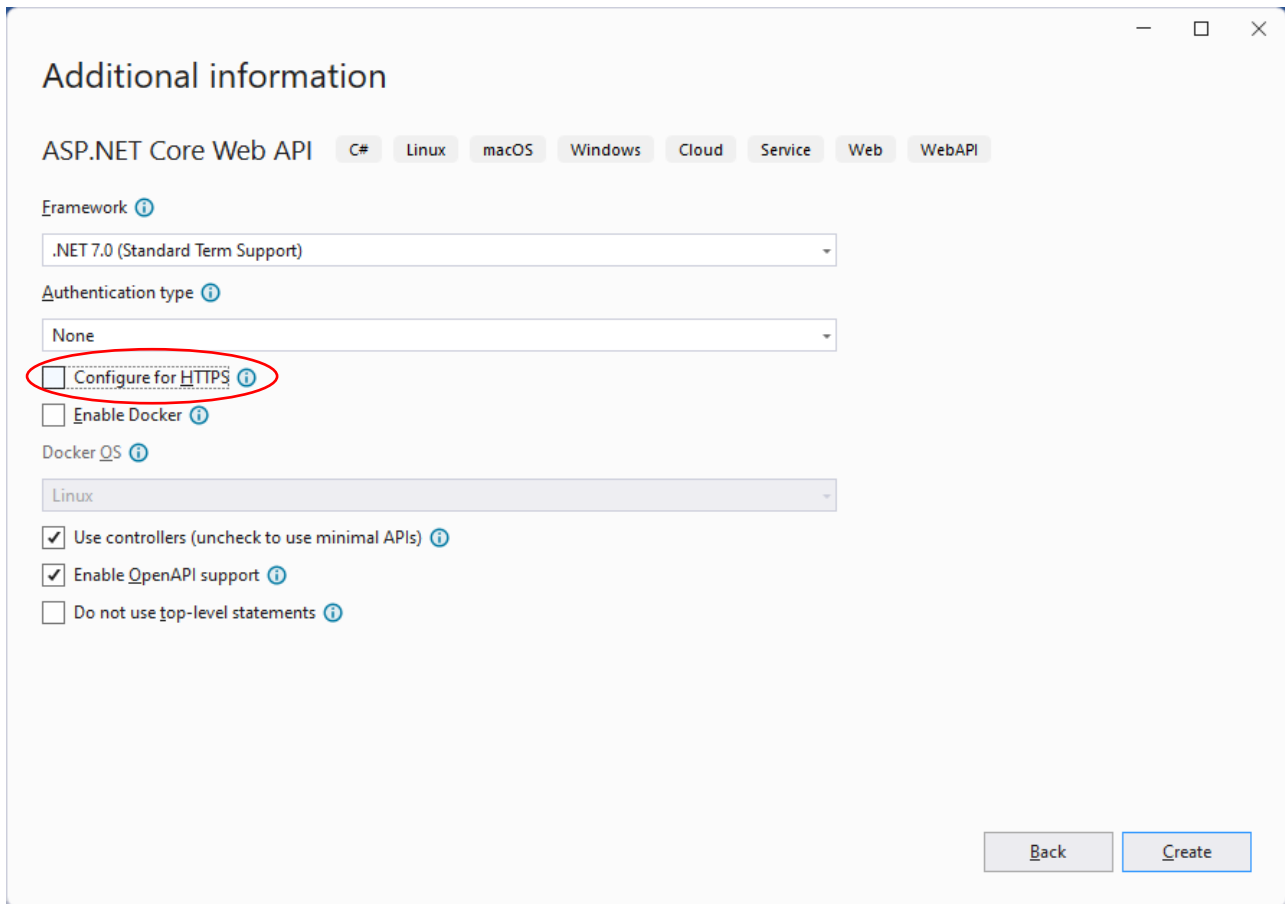


Give your project a name (could be RestExercise1?)

Then select the following options

Select .NET 7.0

Uncheck "Configure for HTTPS"



Your project settings should look like the picture above

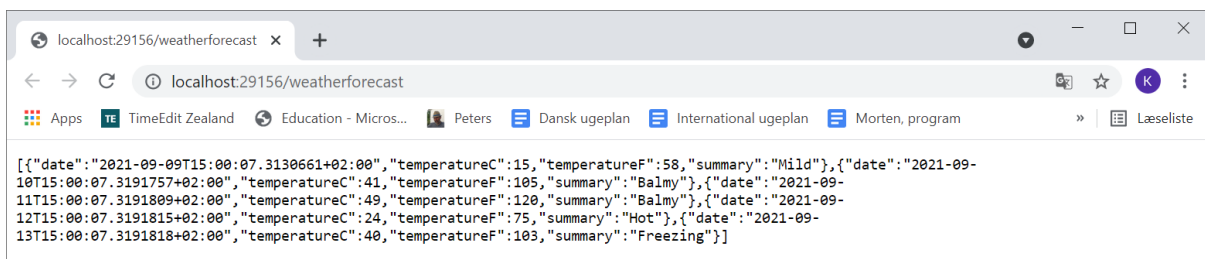
Run the project

The project you created is not empty: It has a REST controller called WeatherForecastController.

Run the project: Press Ctrl+F5 (Build menu -> Start without debugging).

This should compile the application and open a browser with a GET request to the WeatherForecastController.

You should see something similar to this



The port number (29156 in this case) might be different.

The layout of the JSON document might be different - depending on the browser.

Copy the URL to Postman.

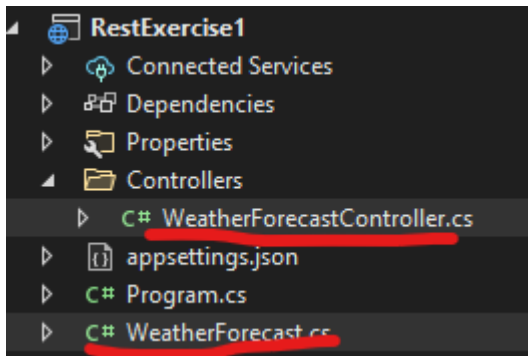
Send a GET request from POSTMAN.

You should see a similar JSON document in the response.

Clean up the project

As you can see, the project will by default contain a weatherforecast service. This is not needed for your project.

Delete the *WeatherForecast* and the *WeatherForecastController* files:



Instead you should create your own Model and Controller classes (and a Repository)

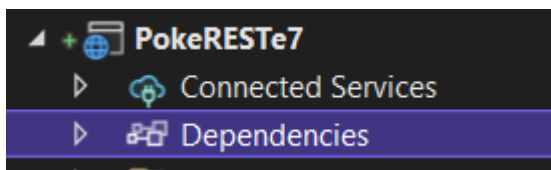
Import DLL

Now you should import the library project that you have created in the programming course, as a DLL file.

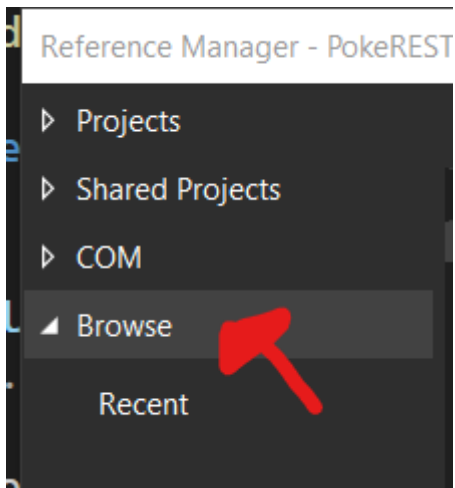
This library should contain both a Model class and a Repository class.

Note: If your model class doesn't have a default/empty constructor, you might run into some problems, so now is a good time to add that, if that is the case.

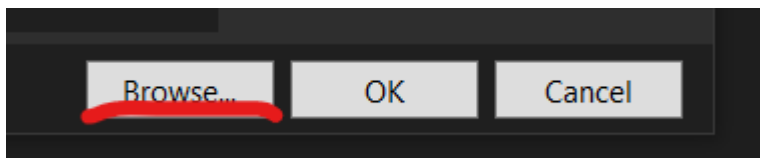
In order to import the file, you should (in the REST project) right click on the "Dependencies" item, in the Solution explorer (your name of the projects are different!)



The go to the Browse tab



And then the browse button at the bottom.



Then you must find your library project, and then into the folders:

Bin->Debug->net7.0

Example:

C:\Users\zealand\source\repos\ActorAPI\ActorRepositoryLib\bin\Debug\net7.0

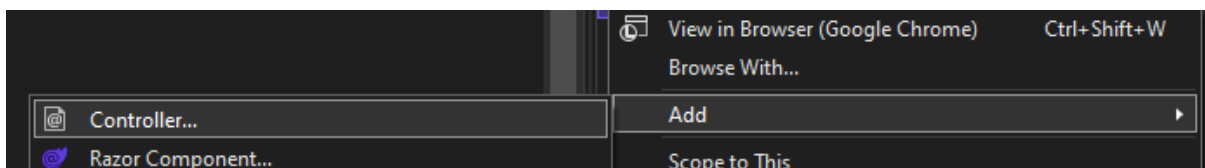
If there isn't any DLL file in the folder, then you must build your library project.

If everything went as expected, you should now have a reference to your library classes.

Controller class

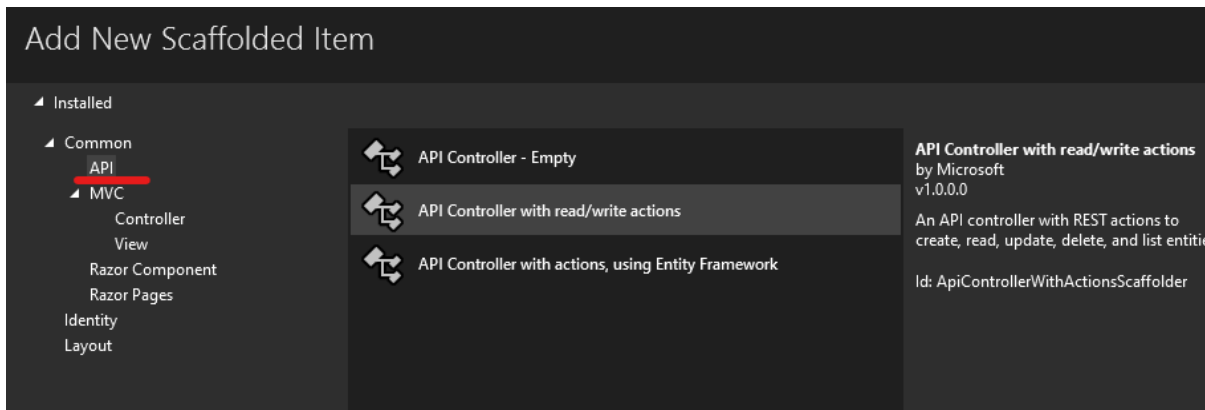
Now you can make the controller class.

Right click on the Controllers folder, and click Add -> Controller



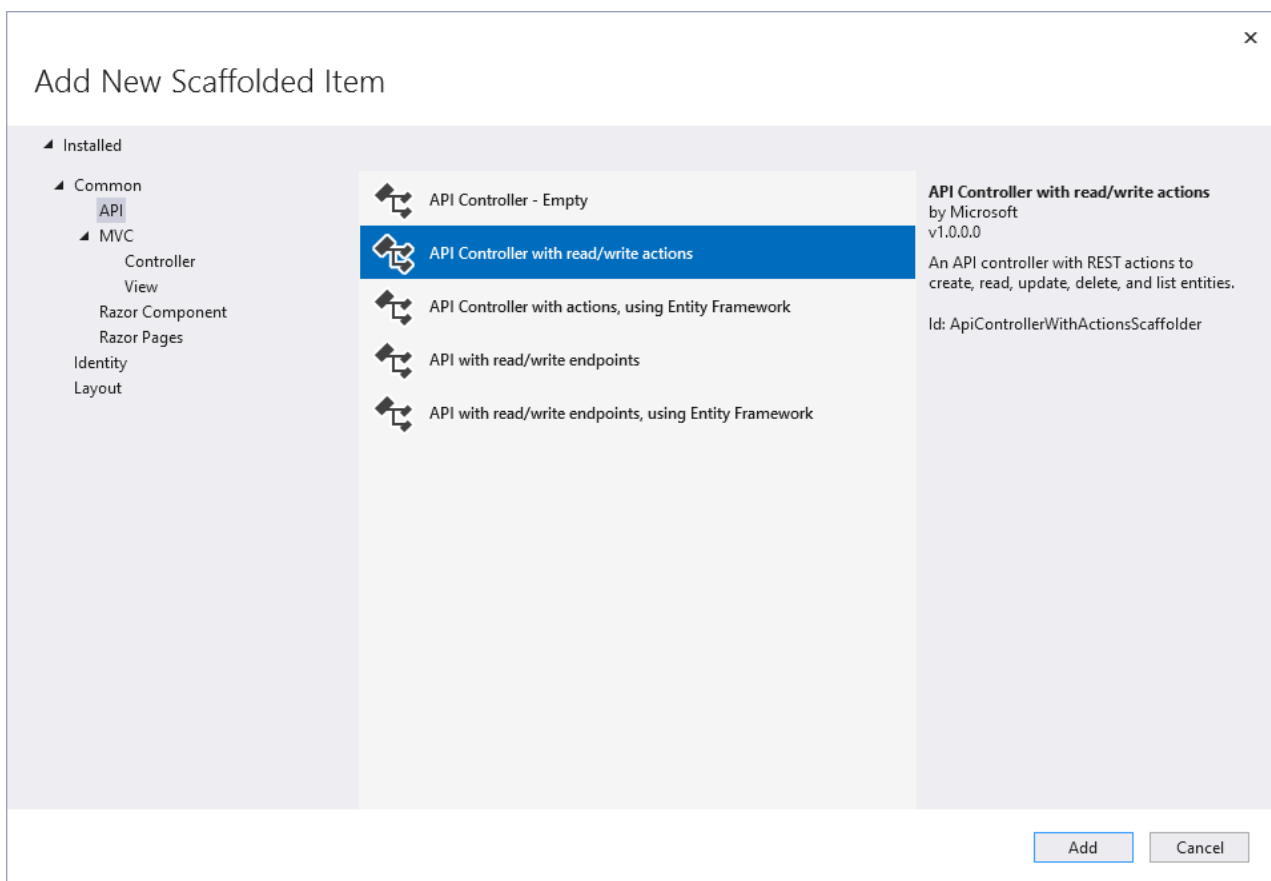
It is **important** that you choose the API to the left, and then select the "API Controller with read/write actions".

You can also choose "Api Controller - Empty" but the read/write option gives you some methods you can easily change



Now name the Controller *xClassController* (replace the Italic *xClass* with the model name, but put it in plural, for example BooksController).

General naming scheme: Noun in plural + Controller.



Change the contents of your controller to something similar to the class ActorController.

A few things to notice

The Http related annotations like [ApiController], [Route], [HttpGet] etc.

The controller methods generally just call the manager methods.

You might also notice that the Controller constructor expects an initialized Repository object.

This is because we want the same object to be used every time. So we need to apply the Singleton pattern.

Luckily we can easily do this in the program.cs file.

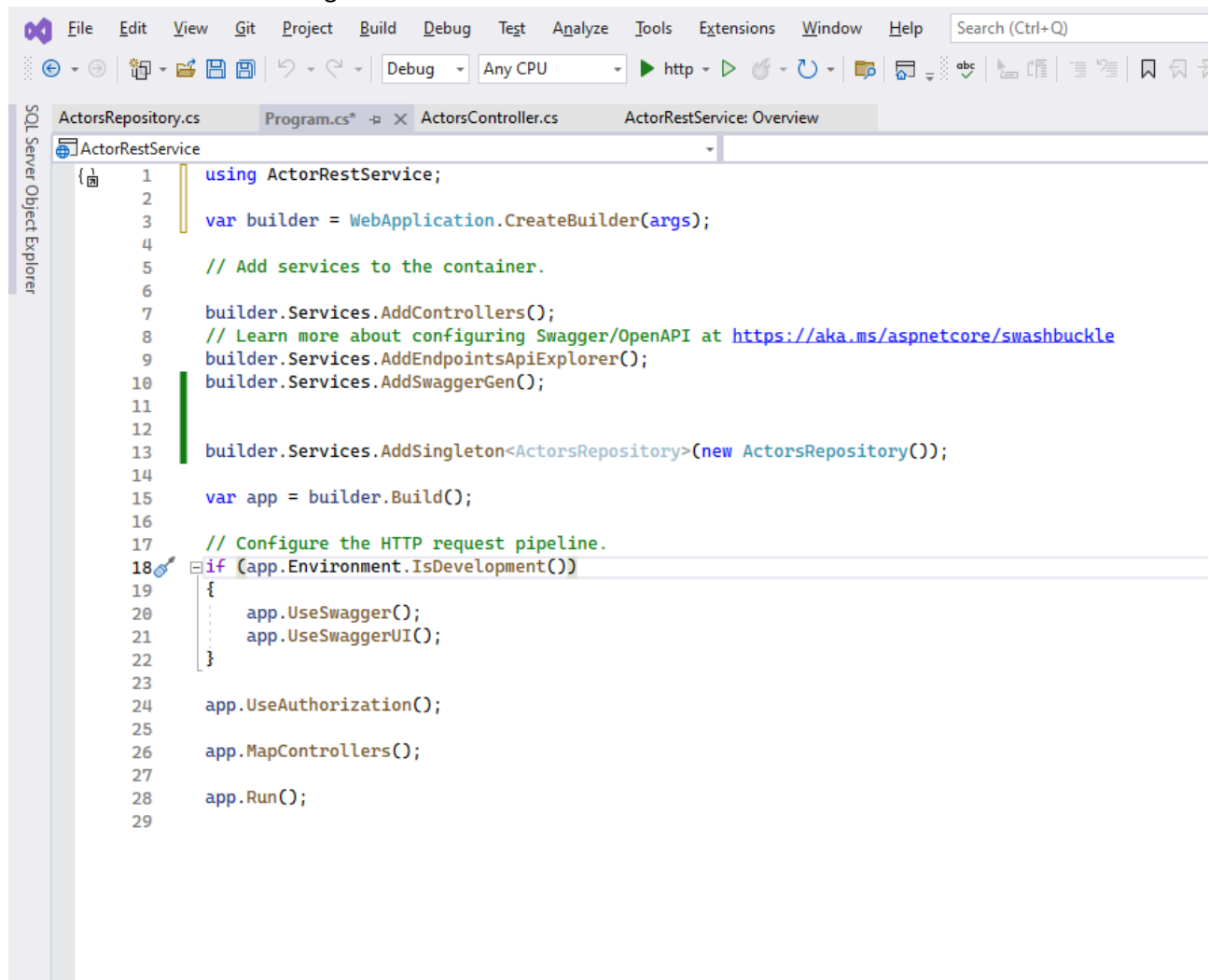
Simple add the following line (changing the ActorsRepository to your repository class name):

```
builder.Services.AddSingleton<ActorsRepository>(new ActorsRepository());
```

Remember to have the using statement. Especially because the Repository is most likely in another namespace.

After the `builder.Services.AddControllers();` line

It should now look something like this:



```
using ActorRestService;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

builder.Services.AddSingleton<ActorsRepository>(new ActorsRepository());

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseAuthorization();

app.MapControllers();

app.Run();
```

Try your REST controller

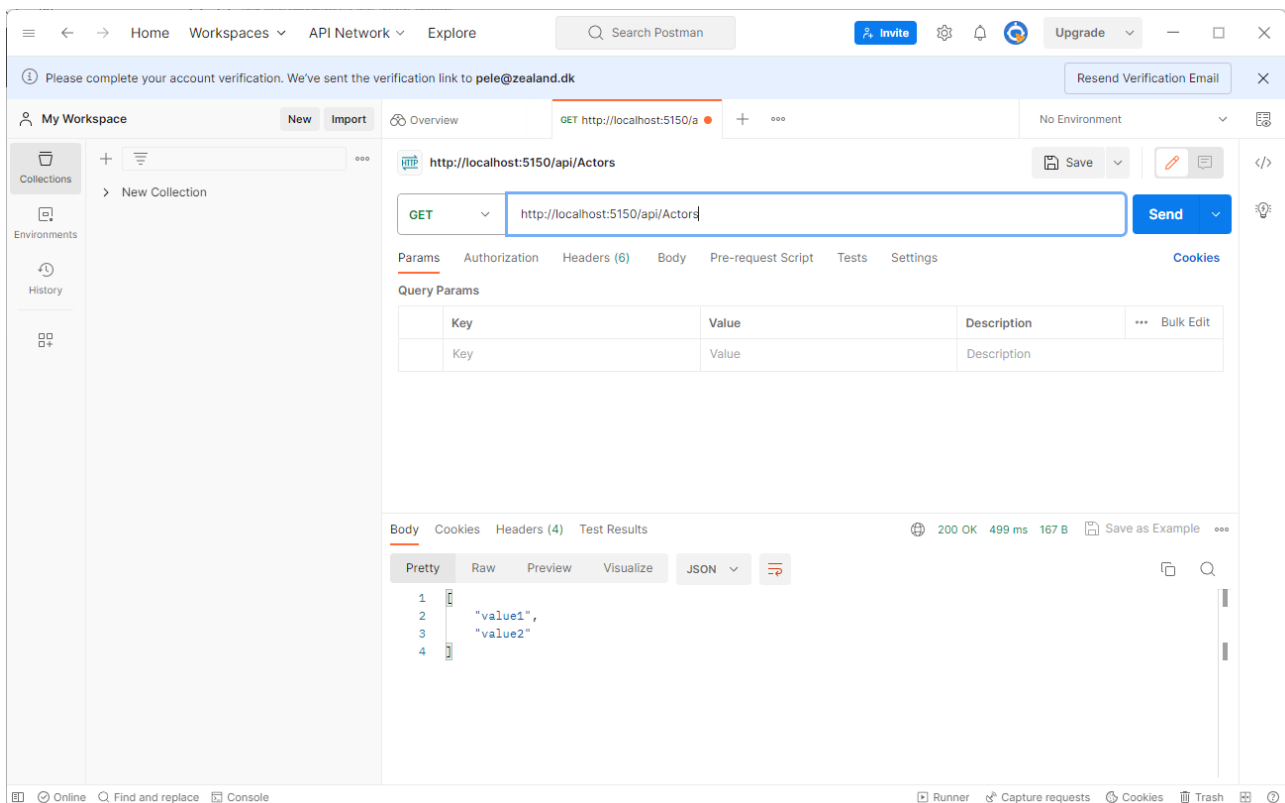
Run the application (Ctrl+F5).

Browser opens and tries to show the WeatherForecast, but that is now removed.

You want to interact with your controller: Browse toapi/Actors (still in plural, or your controller name minus the controller part of the name)

Copy the URL to Postman.

- Try GET
- Try GET by id
- Try POST, which requires a JSON document in the request body
- Try DELETE
- Try PUT, requires a JSON document in the request body
- POST, including JSON document in request body



What happens when you try to add the same ID more than once?

Be aware, when you are using Postman, it will by default tell the server that the data sent is formatted in plain text, not in JSON. To change this, you should add the highlighted header, and disable/remove any other header called Content-Type:

Params	Authorization	Headers (10)	Body ●	Pre-request Script	Tests	Settings
<input checked="" type="checkbox"/>		Accept-Encoding ⓘ				
<input checked="" type="checkbox"/>		Connection ⓘ				
<input checked="" type="checkbox"/>		Content-Type				

Naming the REST resource

If you are not happy with the name `.../api/xClass` you can change it.

In the top of Controller class change `[Route("api/[controller]")]` to `[Route("yourFavoriteName")]`

If you want the browser to start with your controller instead of WeatherForecast when you run the application then open the file `Properties/launchSettings.json` in your project.

Find `launchUrl` and change the value to the name of your REST controller.

How everything works ...

A simple sequence diagram showing how various classes react to an incoming HTTP request

