Controllers Lab

Perform these labs on your own computer using Visual Studio 2022 to ensure you understand the lessons presented in the corresponding videos and lectures.

Lab 1: Add Entity & Repository Classes

Right mouse-click on the project and add a new folder named **EntityLayer**

Right mouse-click on the **EntityLayer** folder and add a new class named **Customer**.

Replace the contents in this file with the following code.

```
namespace AdvWorksAPI.EntityLayer;

public partial class Customer
{
  public Customer()
  {
    Title = string.Empty;
    FirstName = string.Empty;
    MiddleName = string.Empty;
    LastName = string.Empty;
    CompanyName = string.Empty;
    EmailAddress = string.Empty;
    Phone = string.Empty;
  }

  public int CustomerID { get; set; }
  public string? Title { get; set; }
  public string FirstName { get; set; }
  public string? MiddleName { get; set; }
  public string LastName { get; set; }
  public string? CompanyName { get; set; }
  public string? EmailAddress { get; set; }
  public string? Phone { get; set; }
  public DateTime ModifiedDate { get; set; }

  #region ToString Override
  public override string ToString()
  {
    return $"{LastName}, {FirstName} ({CustomerID})";
  }
  #endregion
}
```

Add a Repository Class

Right mouse-click on the project and add a new folder named **RepositoryLayer**

Download Customer Repository Class

Navigate to https://github.com/PaulDSheriff/Training-Samples/tree/main/CSharp-WebAPI-Fundamentals.

Download the **CustomerRepository.cs** file to your hard drive.

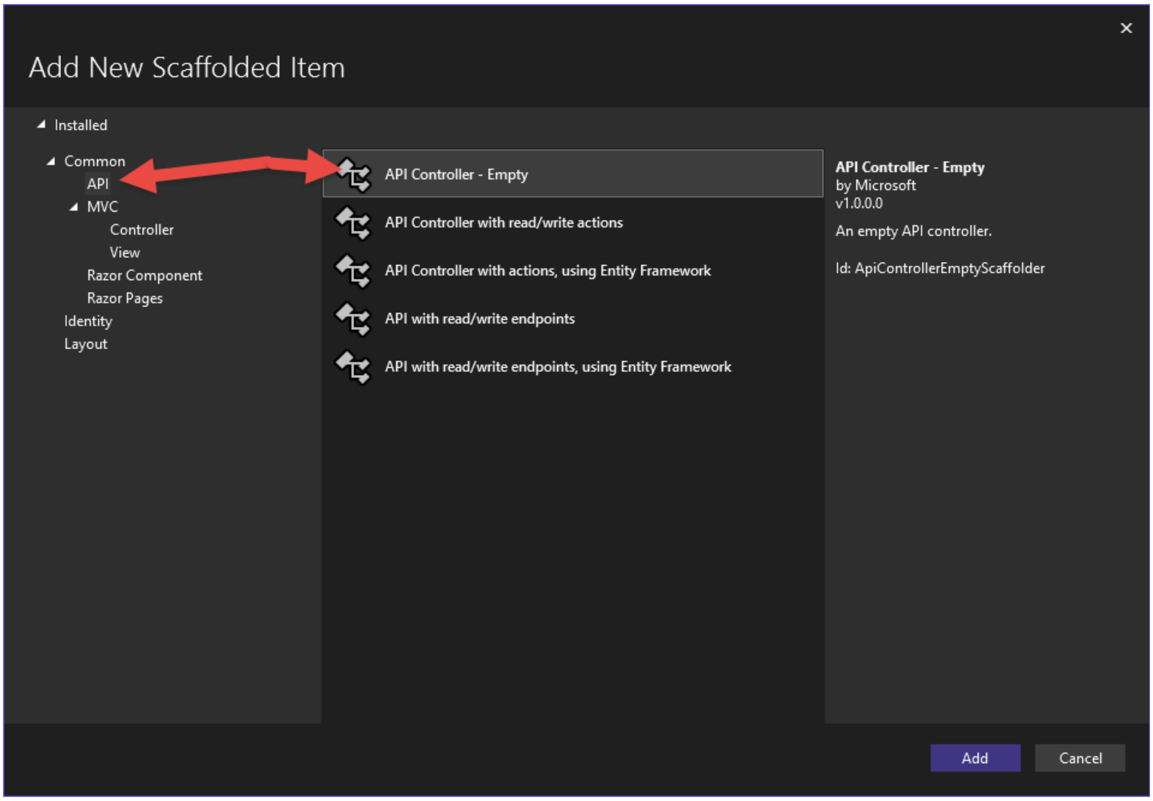
Add the **CustomerRespository.cs** file to the RepositoryLayer folder.

Build the solution to ensure everything compiles correctly.

Lab 2: Create Customer Controller

Right mouse-click on the **Controllers** folder and select **Add | Controller…** from the menu.

Select the API tab, the choose the **API Controller – Empty** template.



Add New Scaffolded Item

- Installed
  - Common
    - API
    - MVC
      - Controller
      - View
    - Razor Component
    - Razor Pages
  - Identity
  - Layout

API Controller - Empty
API Controller with read/write actions
API Controller with actions, using Entity Framework
API with read/write endpoints
API with read/write endpoints, using Entity Framework

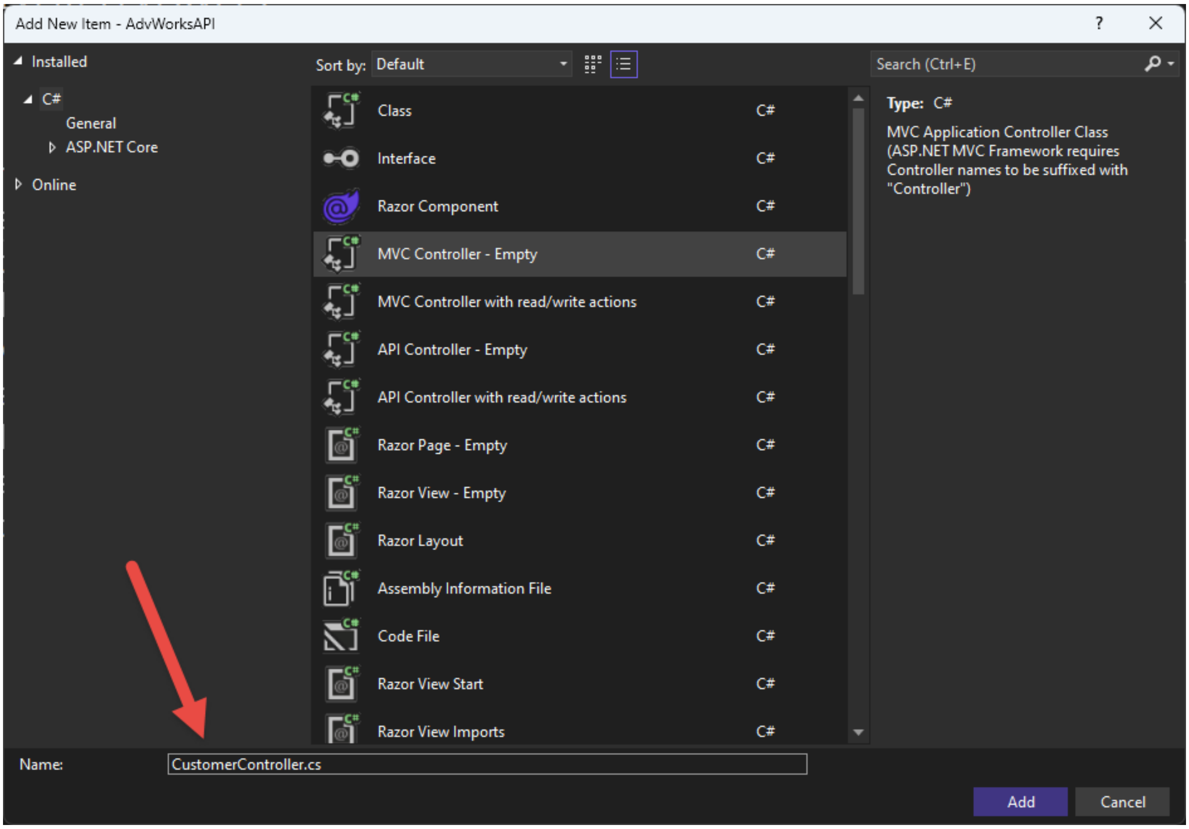
API Controller - Empty
by Microsoft
v1.0.0.0

An empty API controller.

Id: ApiControllerEmptyScaffolder

Click the **Add** button

Set the Name to **CustomerController.cs** as shown below.



Type: C#

MVC Application Controller Class (ASP.NET MVC Framework requires Controller names to be suffixed with "Controller")

Name: CustomerController.cs

Click the **Add** button.

Normally you would start writing the code appropriate for your Web API. However, you can just replace the entire contents of the file with the code shown below.

```
using AdvWorksAPI.EntityLayer;
using AdvWorksAPI.RepositoryLayer;
using Microsoft.AspNetCore.Mvc;

namespace AdvWorksAPI.Controllers;

[Route("api/[controller]")]
[ApiController]
public class CustomerController : ControllerBase
{
  [HttpGet]
  [ProducesResponseType(StatusCodes.Status200OK)]
  [ProducesResponseType(StatusCodes.Status404NotFound)]
  public ActionResult<IEnumerable<Customer>> Get()
  {
    ActionResult<IEnumerable<Customer>> ret;
    List<Customer> list;

    // Get all data
    list = new CustomerRepository().Get();

    if (list != null && list.Count > 0) {
      ret = StatusCode(StatusCodes.Status200OK, list);
    }
    else {
      ret = StatusCode(StatusCodes.Status404NotFound,
"No Customers are available.");
    }

    return ret;
  }
}
```

Try it Out

Run the application and click on the **GET /api/Customer** button.

Click on the **Try it Out** button.

Then click on the **Execute** button.

You should get an array of customer objects appear in the **Response body** field.

Lab 3: Get a Single Customer

Open the **CustomerController.cs** file and add a new action method, below the previous action method, to retrieve a single customer object.

```
[HttpGet("{id}")]
[ProducesResponseType(StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
public ActionResult<Customer> Get(int id)
{
  ActionResult<Customer> ret;
  Customer? entity;

  entity = new CustomerRepository().Get(id);
  if (entity != null) {
    // Found data, return '200 OK'
    ret = StatusCode(StatusCodes.Status200OK, entity);
  }
  else {
    // Did not find data, return '404 Not Found'
    ret = StatusCode(StatusCodes.Status404NotFound,
$"Can't find Customer with a Customer Id of '{id}'.");
  }

  return ret;
}
```

Try it Out

Run the application and click on the **GET /api/Customer/{id}** button.

Click on the **Try it Out** button.

Type the value **306** into the **id** field.

Click the **Execute** button.

You should see a single customer object appear in the **Response body** field.