# Routes and Return Types 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 entire 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-MinimalWebAPI-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: Return List of Customers

Open the **Program.cs** file and add two using statements.

```
using AdvWorksAPI.EntityLayer;
using AdvWorksAPI.RepositoryLayer;
```

Add a new endpoint to return a list of customers.

```
// ***********************************************
// Map Minimal API Routes
// ***********************************************
app.MapGet("/api/Customer", () =>
{
  List<Customer> list;

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

  // Simulate not getting any data
  list.Clear();

  if (list == null || list.Count == 0) {
    return Results.NotFound("No Customers Found.");
  }
  else {
    return Results.Ok(list);
  }
});
```

Notice that you either return a 200 or a 404 depending on if the list comes back with customers or not.

## Try it Out

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

You should get a **404** status because you cleared the list.

**Uncomment** the **list.Clear()** line and you should get a **200**.

# Lab 3: Return a Single Customer

Add another endpoint as shown in the code below.

```
app.MapGet("/api/Customer/{id:int}", (int id) =>
{
  Customer? entity;

  // Attempt to get a single customer
  entity = new CustomerRepository().Get(id);
  if (entity == null) {
    return Results.NotFound($"Customer with CustomerID =
'{id}' was not found.");
  }
  else {
    return Results.Ok(entity);
  }
});
```

Notice that you now have a check to determine if a 200 or a 404 should be returned.

## Try it Out

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

Enter a customer id of 5 to see a 200 status code.

Enter a customer id of 9999 to see a 404 status code.

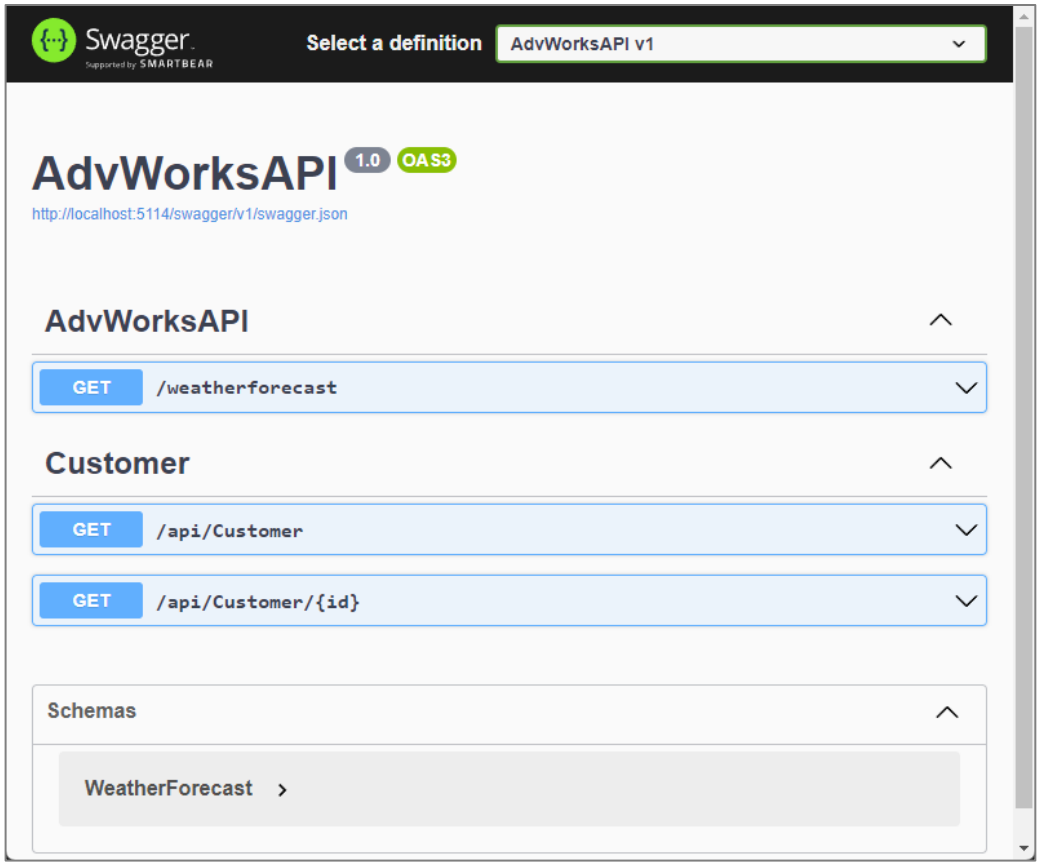
# Lab 4: Grouping APIs

Add the WithTags("TAG_NAME") method to the end of each endpoint to group them within Swagger.

```
app.MapGet("/api/Customer", () =>
{
   // REST OF THE CODE HERE

}).WithTags("Customer");

app.MapGet("/api/Customer/{id:int}", (int id) =>
{
   // REST OF THE CODE HERE

}).WithTags("Customer");
```

# Try it Out

Run the application and you should see something like the following.




# Lab 5: Generate Documentation

Locate the **/api/Customer** APIs and add the Produces() methods as shown in bold below.

```
app.MapGet("/api/Customer", () =>
{
   // REST OF THE CODE HERE

}).WithTags("Customer")
.Produces(200)
.Produces<List<Customer>>()
.Produces(404);

app.MapGet("/api/Customer/{id:int}", (int id) =>
{
   // REST OF THE CODE HERE

}).WithTags("Customer")
.Produces(200)
.Produces<Customer>()
.Produces(404);
```

# Try it Out

Run the application and click on the **GET /api/Customer** button and you should see something that looks like the following.