Dependency Injection 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: Create a Settings Class

Right mouse-click on the **EntityLayer** folder and add a new class named **AdvWorksAPIDefaults**. Replace the entire contents of this new file with the following code.

```
namespace AdvWorksAPI.EntityLayer;

public class AdvWorksAPIDefaults
{
  public AdvWorksAPIDefaults()
  {
    Created = DateTime.Now;
    DefaultTitle = "Mr.";
    DefaultEmail = "LastName.FirstName@adventure-
works.com";
  }

  public DateTime Created { get; set; }
  public string DefaultTitle { get; set; }
  public string DefaultEmail { get; set; }
}
```

Add Scoped AdvWorksAPIDefaults

Open the **Program.cs** file and add the following line of code after the **Add and Configure Services** comment

```
// ********************************************
// Add and Configure Services
// ********************************************
builder.Services.AddScoped<AdvWorksAPIDefaults,
AdvWorksAPIDefaults>();
```

Add Settings API

Below where you added the customer endpoints, add a new endpoint to display the default settings object.

```
app.MapGet("/api/GetSettings", (AdvWorksAPIDefaults
settings) => Results.Ok(settings)).WithTags("Settings");
```

Try it Out

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

You should see the hard-coded settings data.

Lab 2: Use Settings Class in Two API Calls

Add another settings API just below the other one.

```
app.MapGet("/api/GetSettingsAgain", (AdvWorksAPIDefaults
settings) => Results.Ok(settings)).WithTags("Settings");
```

Try it Out

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

View the data and record the date/time field

Click on the **GET /api/Settings/GetSettingsAgain** button.

View the data and notice that the date/time field is **DIFFERENT**

Lab 3: Singleton

Open the **Program.cs** file and **modify** the injection of the AdvWorksAPIDefaults class that you just added. Change the AddScoped() to AddSingleton().

```
builder.Services.AddSingleton<AdvWorksAPIDefaults,
AdvWorksAPIDefaults>();
```

Try it Out

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

View the data and record the date/time field

Click on the **GET /api/Settings/GetSettingsAgain** button.

View the data and notice that the date/time field is **the SAME**

Lab 4: Interfaces & Injection

You should try to always use interfaces with dependency injection.

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

Right mouse-click on the **Interfaces** folder and add a new class named **IRepository**

Replace the entire contents of this new file with the following code.

```
namespace AdvWorksAPI.Interfaces;

public interface IRepository<T>
{
  List<T> Get();
  T? Get(int id);
}
```

Open the **CustomerRepository.cs** file and add a using statement

```
using AdvWorksAPI.Interfaces;
```

Modify the class declaration to inherit from the IRepository interface

```
public class CustomerRepository : IRepository<Customer>
```

Compile the code and everything should still work.

Open the **Program.cs** file and add a new scoped service below the AdvWorksAPIDefaults service you added in the last lab.

```
builder.Services.AddScoped<IRepository<Customer>,
CustomerRepository>();
```

Inject into Customer APIs

Inject the IRepository<Customer> interface into the /api/Customer API

```
app.MapGet("/api/Customer", (IRepository<Customer> repo)
=>
{
  List<Customer> list;

  // Get all customers
  list = repo.Get();

  // REST OF THE CODE HERE
}
```

Do the same for the /api/Customer/{id} API

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

  // Attempt to get a single customer
  entity = repo.Get(id);

  // REST OF THE CODE HERE
}
```

Try it Out

Run the application and run all the Customer APIs to ensure they still work.