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**.

```
namespace AdvWorksAPI.EntityLayer;

public class AdvWorksAPIDefaults
{
  public AdvWorksAPIDefaults()
  {
    Created = DateTime.Now;
    DefaultTitle = "Ms.";
    DefaultEmail = "LastName.FirstName@advworks.com";
  }

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

Add Scoped AdvWorksAPIDefaults

Open the **Program.cs** file.

At the top of the file add a using statement.

```
using AdvWorksAPI.EntityLayer;
```

Add the following line of code after the **Add and Configure Services** comment.

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

Lab 2: Create SettingsController

Right mouse-click on the **Controllers** folder and create a new class named **SettingsController**.

This controller returns the values in the **AdvWorksAPIDefaults** object.

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

namespace AdvWorksAPI.Controllers;

[Route("api/[controller]")]
[ApiController]
public class SettingsController : ControllerBase
{
  private readonly AdvWorksAPIDefaults _Settings;

  public SettingsController(AdvWorksAPIDefaults
settings)
  {
    _Settings = settings;
  }

  [HttpGet]
  [Route("GetSettings")]
  [ProducesResponseType(StatusCodes.Status200OK)]
  public ActionResult<IEnumerable<AdvWorksAPIDefaults>>
GetSettings()
  {
    return Ok(_Settings);
  }

  [HttpGet]
  [Route("GetSettingsAgain")]
  [ProducesResponseType(StatusCodes.Status200OK)]
  public ActionResult<IEnumerable<AdvWorksAPIDefaults>>
GetSettingsAgain()
  {
    return Ok(_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 code in the new file with the following:

```
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 server you added in the last lab.

```
// Add and Configure Repository Classes
builder.Services.AddScoped<IRepository<Customer>,
CustomerRepository>();
```

Inject into Customer Controller

Open the **CustomerController.cs** file and add a new field and a constructor into which you inject the CustomerRepository object you put into the Services collection.

```
private readonly IRepository<Customer> _Repo;

public CustomerController(IRepository<Customer> repo)
{
  _Repo = repo;
}
```

Modify Controller Action Methods to use Repo Field

Open the **CustomerController.cs** file.

Change all four methods that have the code: "**new CustomerRepository()**", replace that code with **"_Repo**"

```
list = _Repo.Get();
```

Try it Out

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

You have just eliminated a dependency between the Controller and the Repository.

You can now remove the **using AdvWorksAPI.RepositoryLayer** statement at the top of the **CustomerController** class.