.NET App Dev Hands-On Lab

Razor Pages/MVC Lab 9a – Data Services

This lab builds the data services for the ASP.NET Core applications. Before starting this lab, you must have completed Razor Pages/MVC Lab 8.

Part 1: Add the Data Services Interface and DAL Classes

The data services will encapsulate the calls for CRUD operations.

- Add the following to the `GlobalUsings.cs` in the `AutoLot.Services` project:

```
global using AutoLot.Dal.Repos.Base;
global using AutoLot.Dal.Repos.Interfaces.Base;
global using AutoLot.Models.Entities;
global using AutoLot.Models.Entities.Base;
```

- Add a directory named `DataServices` in the root of the `AutoLot.Services` project

Step 1: Add the Interfaces

- Add a new directory named `Interfaces` under the `DataServices` directory. In that folder, add another folder named `Base`, and in that folder, add a new interface named `IDataServiceBase` and update the code to the following:

```
namespace AutoLot.Services.DataServices.Interfaces.Base;
public interface IDataServiceBase<TEntity> where TEntity : BaseEntity, new()
{
    Task<IEnumerable<TEntity>> GetAllAsync();
    Task<TEntity> FindAsync(int id);
    Task<TEntity> UpdateAsync(TEntity entity, bool persist = true);
    Task DeleteAsync(TEntity entity, bool persist = true);
    Task<TEntity> AddAsync(TEntity entity, bool persist = true);
    //implemented ghost method since it won't be used by the API data service
    void ResetChangeTracker() { }
}
```

- Add the following to the `GlobalUsings.cs` in the `AutoLot.Services` project:

```
global using AutoLot.Services.DataServices;
global using AutoLot.Services.DataServices.Interfaces;
global using AutoLot.Services.DataServices.Interfaces.Base;
```

- Add a new interface named `ICarDataService` to the Interfaces folder and update the code to the following:

```
namespace AutoLot.Services.DataServices.Interfaces;
public interface ICarDataService : IDataServiceBase<Car>
{
  Task<IEnumerable<Car>> GetAllByMakeIdAsync(int? makeId);
}
```

- Add a new interface named `IMakeDataService` and update the code to the following:

```
namespace AutoLot.Services.DataServices.Interfaces;
public interface IMakeDataService : IDataServiceBase<Make> { }
```

Step 2: Add the DalDataServiceBase Class

- Add a new directory named `Dal` under the `DataServices` directory. In that folder, add a directory named `Base`. In that folder, add a new class named `DalDataServiceBase` and update the code to the following:

```
namespace AutoLot.Services.DataServices.Dal.Base;

public abstract class DalDataServiceBase<TEntity, TDataService>(
  IAppLogging<TDataService> appLogging,
  IBaseRepo<TEntity> mainRepo) : IDataServiceBase<TEntity>
    where TEntity : BaseEntity, new()
    where TDataService : class
{
  protected readonly IBaseRepo<TEntity> MainRepo = mainRepo;
  protected readonly IAppLogging<TDataService> AppLoggingInstance = appLogging;
  public Task<IEnumerable<TEntity>> GetAllAsync()
    => Task.FromResult(MainRepo.GetAllIgnoreQueryFilters());
  public Task<TEntity> FindAsync(int id) => Task.FromResult(MainRepo.Find(id));
  public Task<TEntity> UpdateAsync(TEntity entity, bool persist = true)
  {
    MainRepo.Update(entity, persist);
    return Task.FromResult(entity);
  }
  public Task DeleteAsync(TEntity entity, bool persist = true)
    => Task.FromResult(MainRepo.Delete(entity, persist));
  public Task<TEntity> AddAsync(TEntity entity, bool persist = true)
  {
    MainRepo.Add(entity, persist);
    return Task.FromResult(entity);
  }
  public void ResetChangeTracker()
  {
    MainRepo.Context.ChangeTracker.Clear();
  }
}
```

- Add the following to the `GlobalUsings.cs` class:

```
global using AutoLot.Services.DataServices.Dal;
global using AutoLot.Services.DataServices.Dal.Base;
```

Step 3: Add the CarDalDataService Class

- Add a new class named `CarDalDataService` in the `Dal` directory and update the code to the following:

```
namespace AutoLot.Services.DataServices.Dal;

public class CarDalDataService(IAppLogging<CarDalDataService> appLogging, ICarRepo repo)
  : DalDataServiceBase<Car, CarDalDataService>(appLogging, repo), ICarDataService
{
  public Task<IEnumerable<Car>> GetAllByMakeIdAsync(int? makeId)
    =>  Task.FromResult(makeId.HasValue
                        ? repo.GetAllBy(makeId.Value)
                        : MainRepo.GetAllIgnoreQueryFilters());
}
```

Step 4: Add the MakeDalDataService Class

- Add a new class named `MakeDalDataService` in the `Dal` directory and update the code to the following:

```
namespace AutoLot.Services.DataServices.Dal;

public class MakeDalDataService(IAppLogging<MakeDalDataService> appLogging, IMakeRepo repo)
  : DalDataServiceBase<Make, MakeDalDataService>(appLogging, repo), IMakeDataService;
```

Part 2: Add the RemoveAsync Extension Method

- Add the following method to the `StringExtensions.cs` class in the `AutoLot.Services` project:

```
public static string RemoveAsyncSuffix(this string original)
  => original.Replace("Async", "", StringComparison.OrdinalIgnoreCase);
```

Summary

This lab added the common code for the DAL Data Services to be used by the ASP.NET Core projects.

Next steps

In the next part of this tutorial series, you will use the data services in the ASP.NET Core project.