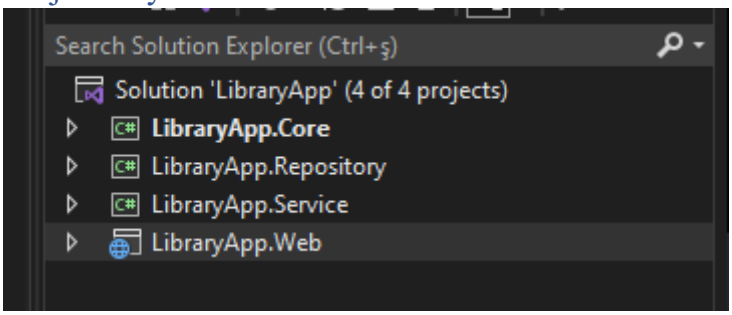
Library App Source Code Documentation

Project Layers



- Core Layer:
- Repository Layer:
- Service Layer:
- Web App Layer:

Core Layer

- Entities:

Book.cs

```
0 references
public class Book
{
    0 references
    public int Id { get; set; }
    0 references
    public string Name { get; set; }
    0 references
    public string Author { get; set; }
    0 references
    public string FilePath { get; set; }
    0 references
    public bool IsInLibrary { get; set; }
    0 references
    public string? Borrower { get; set; } // nullable
    0 references
    public DateTime? FinishDate { get; set; } // nullable
}
```

- Generic Repository (Interface)

```
0 references
public interface IGenericRepository<T> where T : class
{
    0 references
    Task<T> GetById(int id);
    0 references
    IQueryable<T> GetAll();
    0 references
    IQueryable<T> Where(Expression<Func<T,bool>> expression); // OrderBy etc. (Alfabetik sıralama için kullanacağız)
    0 references
    Task AddAsync(T entity);
    0 references
    void Update(T entity); // (İlgili kitabı bir kullanıcıya atarken Update kullanacağız)
}
```

- Generic Service (Interface)

```
0 references
public interface IService<T> where T : class
{
    0 references
    Task<T> GetByIdAsync(int id);
    0 references
    Task<IEnumerable<T>> GetAllAsync();
    0 references
    IQueryable<T> Where(Expression<Func<T, bool>> expression); // OrderBy etc. (Alfabetik sıralama için kullanacağız)
    0 references
    Task AddAsync(T entity);
    0 references
    Task UpdateAsync(T entity); // (İlgili kitabı bir kullanıcıya atarken Update kullanacağız)
}
```

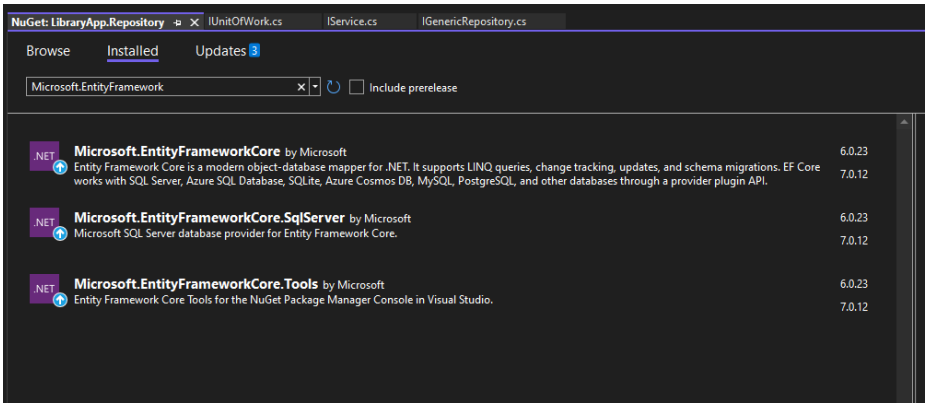
- UnitOfWork (Interface)

Note: Unit Of Work Pattern: Allows us to manage database operations through a single operation.

```
{
    0 references
    public interface IUnitOfWork
    {
        0 references
        Task CommitAsync(); // EF --> SaveChangeAsync
        0 references
        void Commit(); // EF --> SaveChange
    }
}
```

Repository (Data Access) Layer

- Entity Framework libraries downloaded.



- AppDbContext: Class that manages database connections.

```
namespace LibraryApp.Repository
{
    2 references
    public class AppDbContext : DbContext
    {
        0 references
        public AppDbContext(DbContextOptions<AppDbContext> options) : base(options)
        {

        }

        // my tables:
        0 references
        public DbSet<Book> Books { get; set; }
    }
}
```

- Configurations: Setting table permissions with Entity Framework

```
0 references
public class BookConfiguration : IEntityTypeConfiguration<Book>
{
    0 references
    public void Configure(EntityTypeBuilder<Book> builder)
    {
        //throw new NotImplementedException();
        builder.HasKey(x => x.Id);
        builder.Property(x => x.Id).UseIdentityColumn();

        builder.Property(x=>x.Name).IsRequired().HasMaxLength(50);

        builder.Property(x => x.Author).IsRequired().HasMaxLength(50);

        builder.Property(x => x.FilePath).IsRequired().HasMaxLength(256);

        builder.Property(x => x.IsInLibrary).IsRequired();
    }
}
```

- Generic Repository (Receives implemantation from the IGenericRepository)

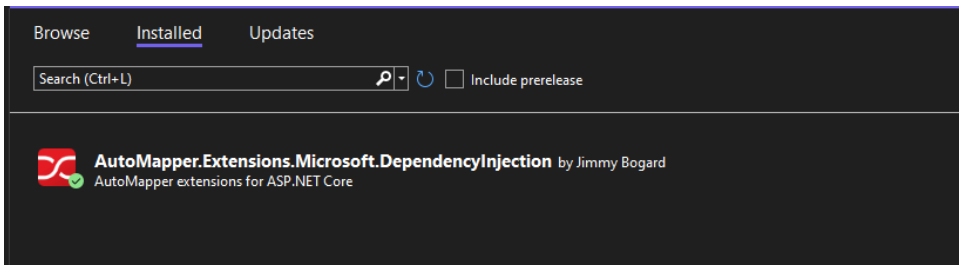
```
namespace LibraryApp.Repository.Repositories
{
    1 reference
    public class GenericRepository<T> : IGenericRepository<T> where T : class
    {
        protected readonly AppDbContext _context;
        private readonly DbSet<T> _dbSet;

        // constructor
        0 references
        public GenericRepository(AppDbContext context)
        {
            _context = context;
            _dbSet = _context.Set<T>();
        }

        1 reference
        public async Task AddAsync(T entity)
        {
            await _dbSet.AddAsync(entity);
        }

        1 reference
        public IQueryable<T> GetAll()
        {
            return _dbSet.AsNoTracking().AsQueryable();
        }

        1 reference
        public async Task<T> GetByIdAsync(int id)
        {
            return await _dbSet.FindAsync(id);
        }

        1 reference
        public void Update(T entity)
        {
            //_context.Entry(entity).State = EntityState.Modified; --> same thing
            _dbSet.Update(entity);
        }

        1 reference
        public IQueryable<T> Where(Expression<Func<T, bool>> expression)
        {
            return _dbSet.Where(expression);
        }
    }
}
```

- UnitOfWork (Receives implemantation from the IUnitOfWork)

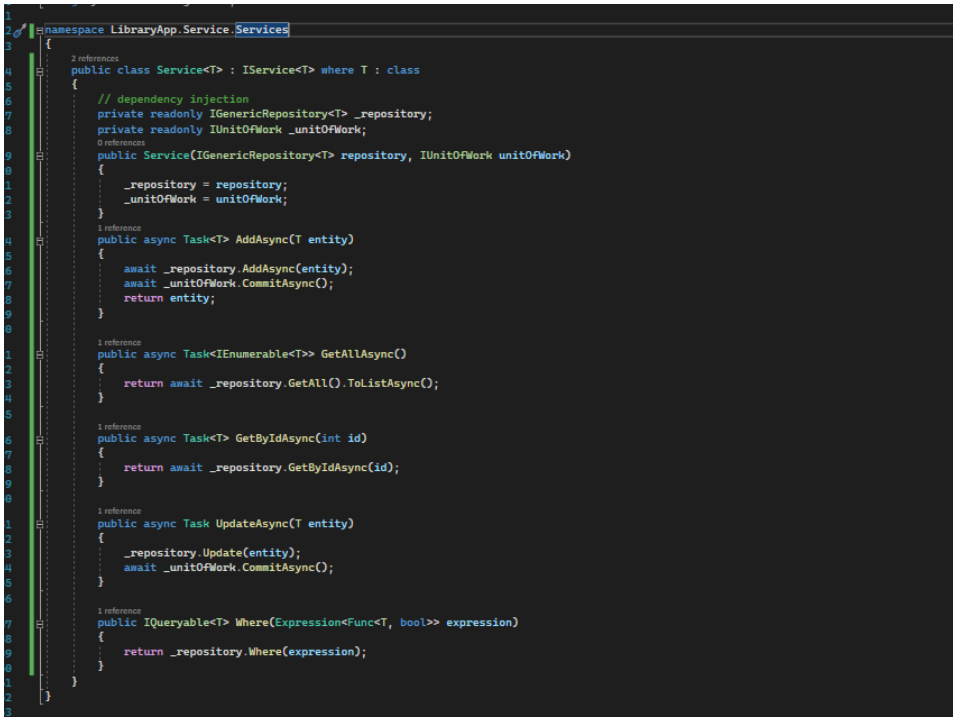
```
namespace LibraryApp.Repository.UnitOfWorks
{
    1 reference
    public class UnitOfWork : IUnitOfWork
    {
        private readonly AppDbContext _context;
        0 references
        public UnitOfWork(AppDbContext context)
        {
            _context = context;
        }
        1 reference
        public void Commit()
        {
            _context.SaveChanges();
        }

        1 reference
        public async Task CommitAsync()
        {
            _context.SaveChangesAsync();
        }
    }
}
```

Service Layer

- AutoMapper library downloaded.



- Generic Service (Receives implemantation from the IService)



```
namespace LibraryApp.Service.Services
{
    2 references
    public class Service<T> : IService<T> where T : class
    {
        // dependency injection
        private readonly IGenericRepository<T> _repository;
        private readonly IUnitOfWork _unitOfWork;
        0 references
        public Service(IGenericRepository<T> repository, IUnitOfWork unitOfWork)
        {
            _repository = repository;
            _unitOfWork = unitOfWork;
        }
        1 reference
        public async Task<T> AddAsync(T entity)
        {
            await _repository.AddAsync(entity);
            await _unitOfWork.CommitAsync();
            return entity;
        }

        1 reference
        public async Task<IEnumerable<T>> GetAllAsync()
        {
            return await _repository.GetAll().ToListAsync();
        }

        1 reference
        public async Task<T> GetByIdAsync(int id)
        {
            return await _repository.GetByIdAsync(id);
        }

        1 reference
        public async Task UpdateAsync(T entity)
        {
            _repository.Update(entity);
            await _unitOfWork.CommitAsync();
        }

        1 reference
        public IQueryable<T> Where(Expression<Func<T, bool>> expression)
        {
            return _repository.Where(expression);
        }
    }
}
```

Web UI Layer

MVC pattern is used in this layer.

- BooksController.cs (Controller)
- LibraryPageViewModel (Model)
- Index.cshtml (View)

Some Javascript libraries used: Bootstrap, JQuery, Flatpicker, Sweetalert etc.