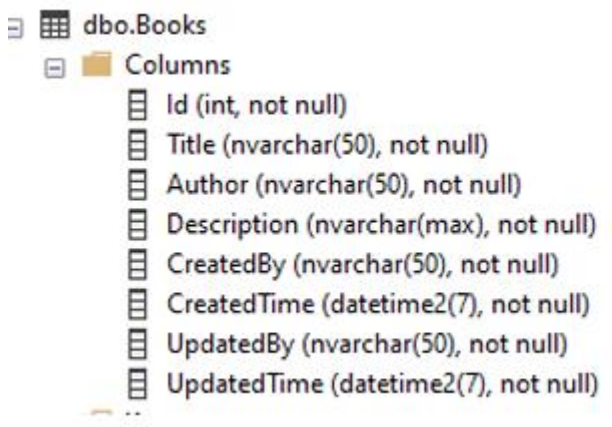
Repository-> Service-> Controller Short Guide

Prerequisite:

1. Must have SQLserver set-up.

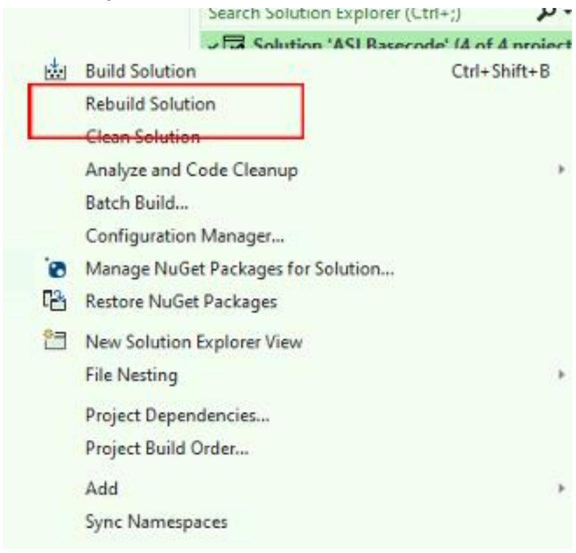
2. Create a table for your data



dbo.Books
Columns
Id (int, not null)
Title (nvarchar(50), not null)
Author (nvarchar(50), not null)
Description (nvarchar(max), not null)
CreatedBy (nvarchar(50), not null)
CreatedTime (datetime2(7), not null)
UpdatedBy (nvarchar(50), not null)
UpdatedTime (datetime2(7), not null)

※In this example, we will be creating repository for Books.

3. Rebuild your solution



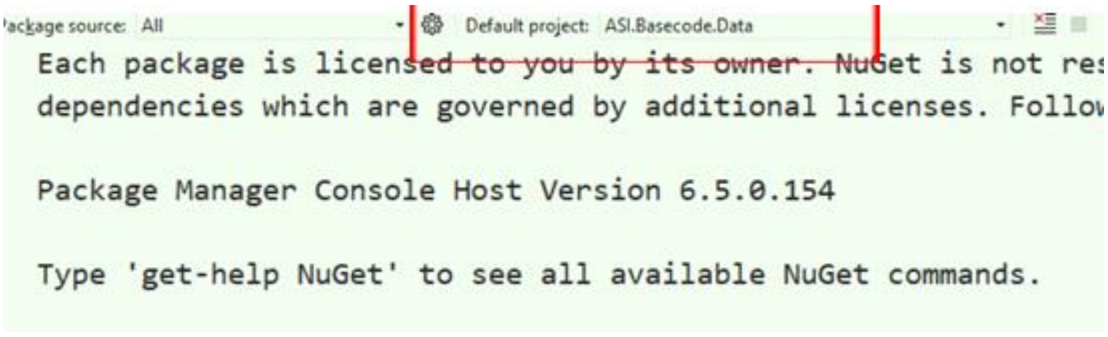
※This allow us to check if our current solution has issues.

※Next step will not work if there are current issues.

4. Perform Scaffold command.

(Tools > Nuget Package Manager > Package Manager Console)

4.1 Change Default Project to ASI.Basecode.Data



```
Each package is licensed to you by its owner. NuGet is not res
dependencies which are governed by additional licenses. Follov

Package Manager Console Host Version 6.5.0.154

Type 'get-help NuGet' to see all available NuGet commands.
```

4.2 Execute below command:

```
Scaffold-DbContext "Addr=localhost; database=AsiBasecodeDB;
Trusted_Connection=True;MultipleActiveResultSets=true;TrustServerCertificate=True"
Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models -ContextDir Data/.. -Force
```

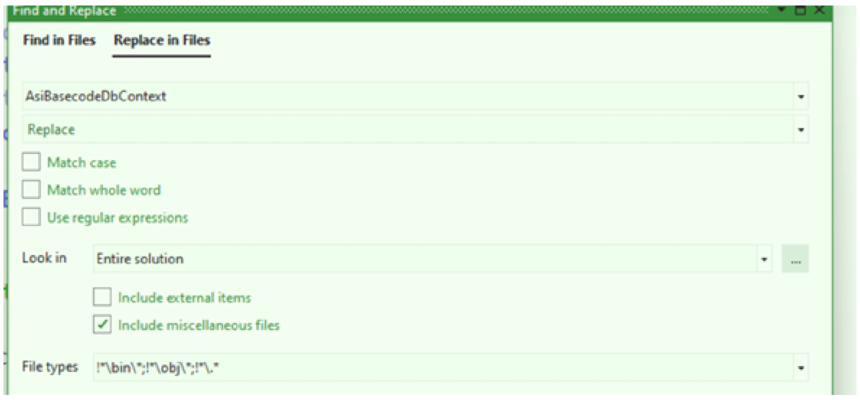
※ IMPORTANT: Update **Addr** and **database** with your server name and db name.

5. Once the above is executed, a Context file will be generated.

5.1 If your Context is not AsiBasecodeDbContext, remove this file.

5.2 Once removed, update all AsiBasecodeDbContext references with your context.

5.3 Press CTRL+SHIFT+F then click on Replace in Files.



AsiBasecodeDbContext

Replace

☐ Match case

☐ Match whole word

☐ Use regular expressions

Look in: Entire solution

☐ Include external items

☑ Include miscellaneous files

File types: !*\bin*;!*\obj*;!*\.*

※Make sure that the **Look in** value is set to Entire solution.

6. Once above is done, Rebuild solution once again to make sure that there are no issues.

7. If there are no more issues, we are now ready for setting up our connection to db using the repository.

Files Setup:

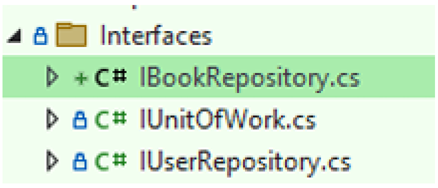
※The following steps can be interchangeable depending on you coding style.

※If you are still not familiar, you can just follow the flow below.

//REPOSITORY

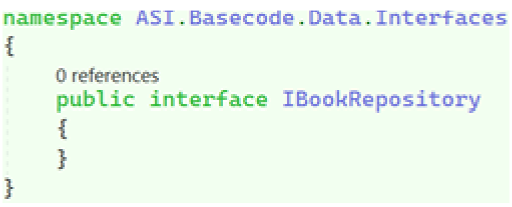
1. Repository file creation on project **ASI.Basecode.Data**

1.1 Create a new interface repository under the **Interfaces** folder.



```
◢ Interfaces
    ▷ + C# IBookRepository.cs
    ▷ C# IUnitOfWork.cs
    ▷ C# IUserRepository.cs
```

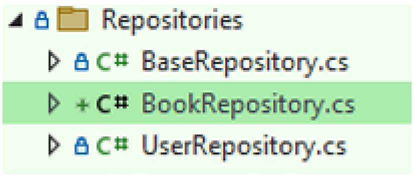
※Always start with an **I** then add **Repository** to your filename.



```
namespace ASI.Basecode.Data.Interfaces
{
    0 references
    public interface IBookRepository
    {
    }
}
```

※Don't forget to set it to public

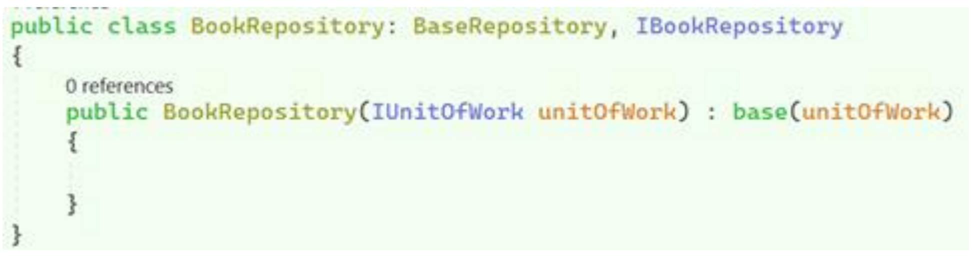
1. 2 Create a new repository under the **Repositorie**s folder.



```
◢ Repositories
    ▷ C# BaseRepository.cs
    ▷ + C# BookRepository.cs
    ▷ C# UserRepository.cs
```

※Always add **Repository** to your filename for naming convention.

1.3 On your **BookRepository**, inherit the **BaseRepository** and **IBookRepository** then set up the UnitOfWork.

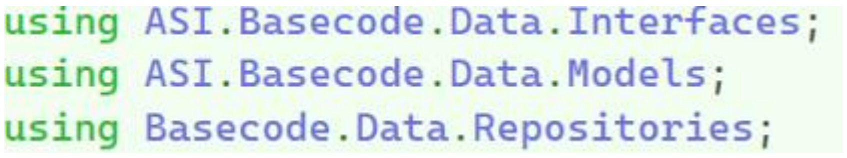


```
public class BookRepository: BaseRepository, IBookRepository
{
    0 references
    public BookRepository(IUnitOfWork unitOfWork) : base(unitOfWork)
    {

    }
}
```

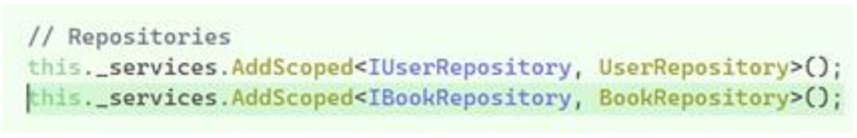
※Don't forget to set it to public.

※Make sure that the following is imported to avoid errors:



```
using ASI.Basecode.Data.Interfaces;
using ASI.Basecode.Data.Models;
using Basecode.Data.Repositories;
```

1.4 Configure the **IBookRepository** and **BookRepository** on the **StartUp.DI**.

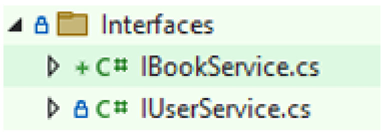


```
// Repositories
this._services.AddScoped<IUserRepository, UserRepository>();
this._services.AddScoped<IBookRepository, BookRepository>();
```

//SERVICE

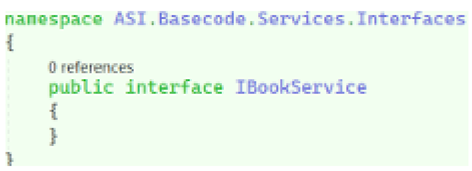
2. Service file creation on project **ASI.Basecode.Services.**

2.1 Create a new interface service under the **Interfaces** folder.



```
Interfaces
    IBookService.cs
    IUserService.cs
```

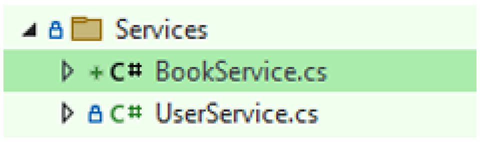
※Always start with an **I** then add **Service** to your filename.



```
namespace ASI.Basecode.Services.Interfaces
{
    0 references
    public interface IBookService
    {
    }
}
```

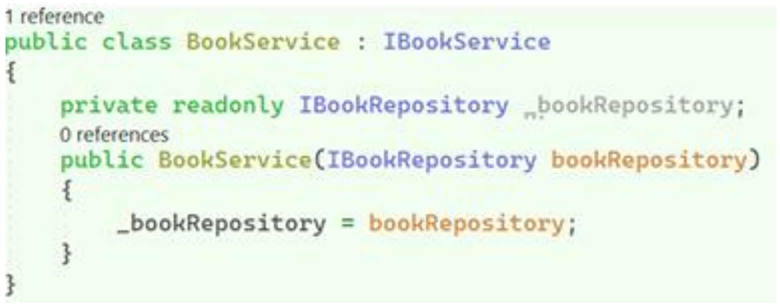
※Don't forget to set it to public

2.2 Create a new service under the **Services** folder.



```
◢ Services
    ▷ C# BookService.cs
    ▷ C# UserService.cs
```

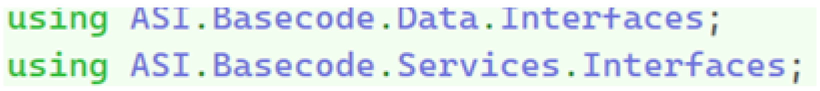
2.3 On your **BookService** inherit the **IBookService** and configure the **IBookRepository** so that we can use the functions under **BookRepository**.



```
1 reference
public class BookService : IBookService
{
    private readonly IBookRepository _bookRepository;
    0 references
    public BookService(IBookRepository bookRepository)
    {
        _bookRepository = bookRepository;
    }
}
```

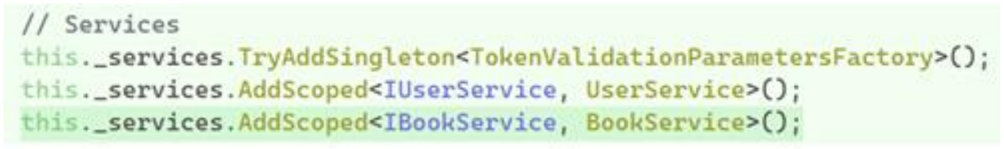
※Don't forget to set it to public

※Make sure that the following is imported to avoid errors:



```
using ASI.Basecode.Data.Interfaces;
using ASI.Basecode.Services.Interfaces;
```

2.4 Configure the **IBookService** and **BookService** on the **StartUp.DI**.

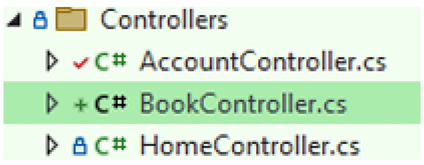


```
// Services
this._services.TryAddSingleton<TokenValidationParametersFactory>();
this._services.AddScoped<IUserService, UserService>();
this._services.AddScoped<IBookService, BookService>();
```

//Controller

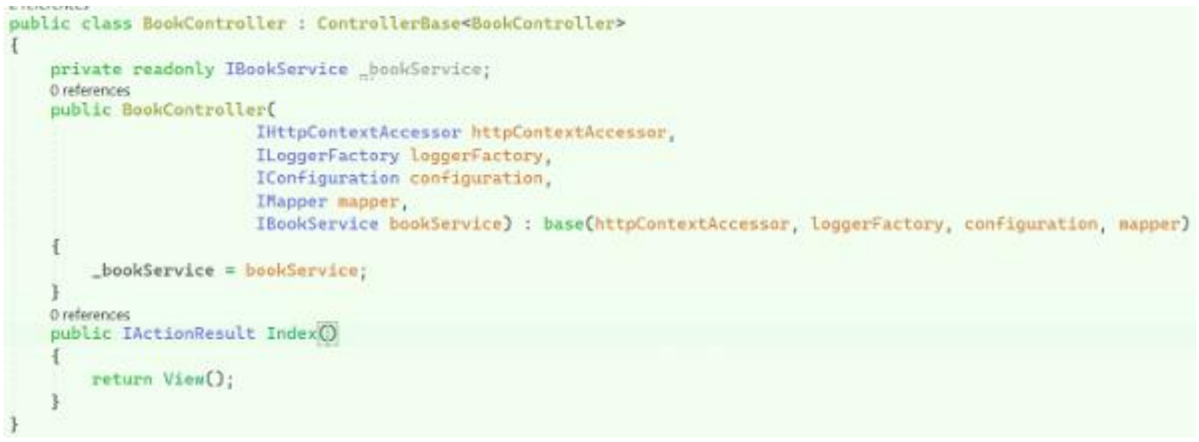
3. Controller file creation on project **ASI.Basecode.WebApp.**

3.1 Create a new controller under the **Controllers** folder.



```
◢ Controllers
    ▷ C# AccountController.cs
    ▷ C# BookController.cs
    ▷ C# HomeController.cs
```

3.2 On your **BookController** , configure the **ControllerBase** for the **BookController**, and configure the **IBookService** so that we can use the functions under **BookService**.



```
public class BookController : ControllerBase<BookController>
{
    private readonly IBookService _bookService;
    0 references
    public BookController(
                        IHttpContextAccessor httpContextAccessor,
                        ILoggerFactory loggerFactory,
                        IConfiguration configuration,
                        IMapper mapper,
                        IBookService bookService) : base(httpContextAccessor, loggerFactory, configuration, mapper)
    {
        _bookService = bookService;
    }
    0 references
    public IActionResult Index()
    {
        return View();
    }
}
```

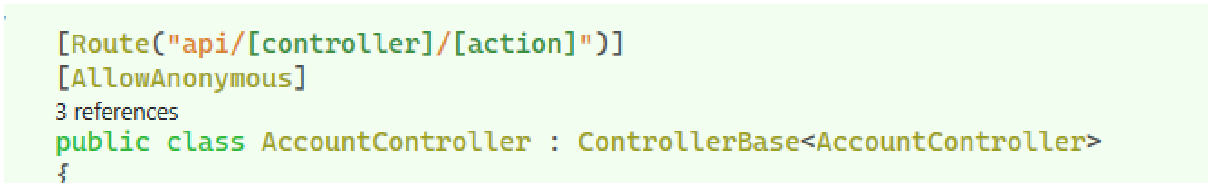
Summary:

Once above steps are configured correctly, the site will run without issues and you can now add CRUD logic for your Books screen

NOTES:

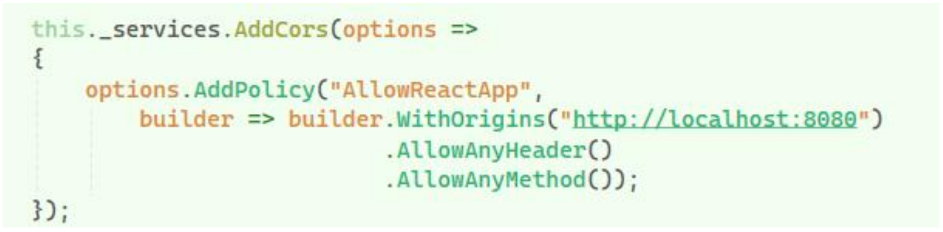
When Basecode is partnered with REACT:

1. Add [Route("api/[controller]/[action]")] to **Controller**
2. If Authentication is not yet set, add `[AllowAnonymous]`



```
[Route("api/[controller]/[action]")]
[AllowAnonymous]
3 references
public class AccountController : ControllerBase<AccountController>
{
```

3. Update React Url on **Startup.DI**



```
this._services.AddCors(options =>
{
    options.AddPolicy("AllowReactApp",
        builder => builder.WithOrigins("http://localhost:8080")
                          .AllowAnyHeader()
                          .AllowAnyMethod());
});
```

.

※Match it to your React url.