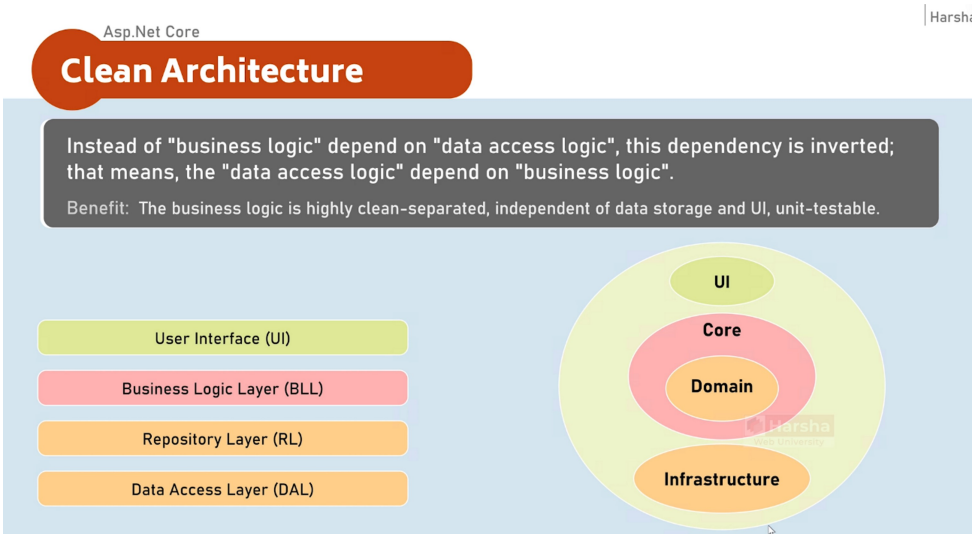
Section 24: Clean Architecture [MVC and Web API]

05 May 2025 10:49



For medium to large-scale applications, **clean architecture** is the most popular and recommended approach by most architects.

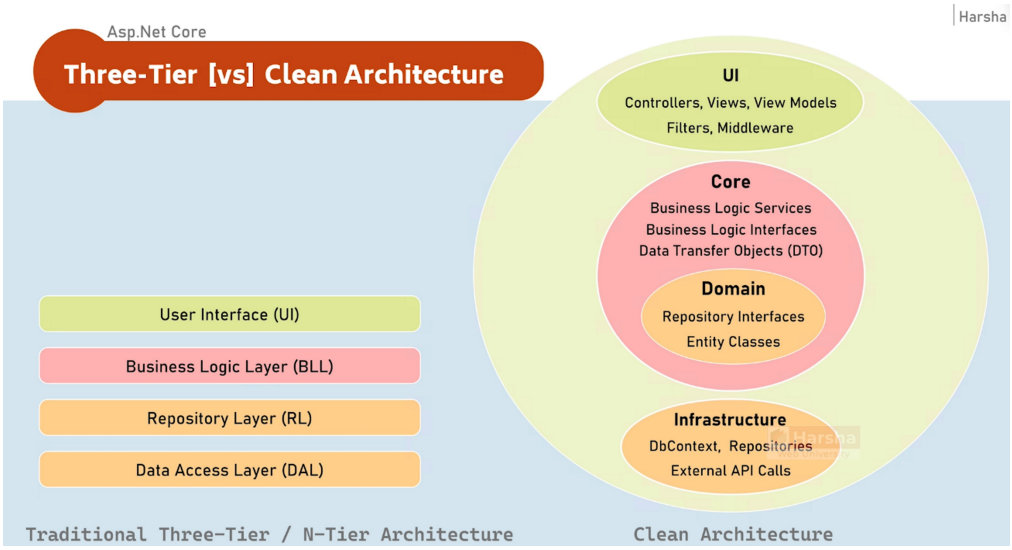
So, what's the difference between regular **3-tier (or n-tier) architecture** and **clean architecture**?
In the case of **n-tier architecture**, the **business logic layer depends on the data access layer**.
But in **clean architecture**, the dependency direction is **inward**. Instead of the business logic depending on the data access layer, the **data access layer depends on the business logic layer**.

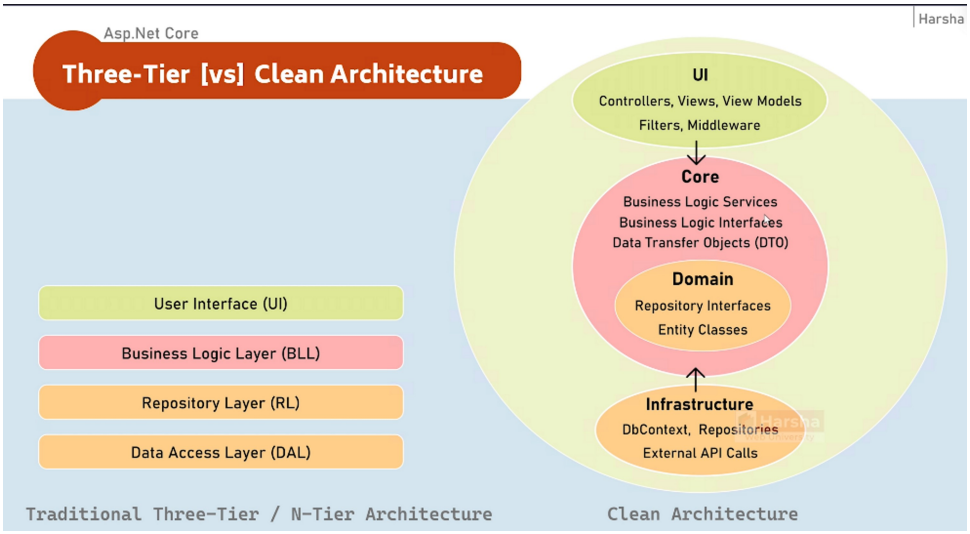
Moreover, clean architecture is **not stacked**, unlike n-tier architecture. It's also known as **onion architecture**, where one layer surrounds the previous one.

For example:
- At the center, there's the **domain layer**.
- Surrounding it is the **core layer**.
- Then come the **infrastructure** and **UI layers**, which are built on top of the core layer.

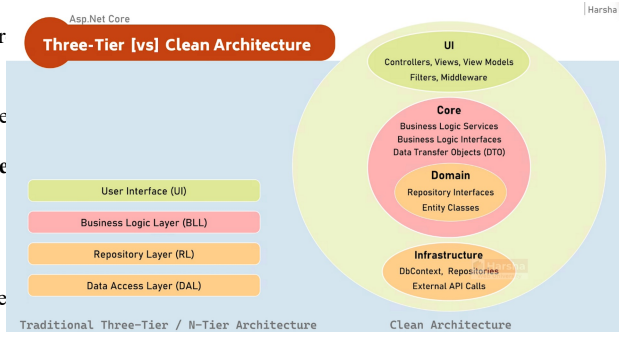
This means it's **not a simple top-down stack** like in regular n-tier architecture.
The main goal of clean architecture is to **separate the business logic** and make it **independent** and **cleanly isolated**.
This separation is crucial to scale out applications for larger projects.

So, the **core idea** of clean architecture is:
The **business logic layer sits at the center** of the application, and **all other layers** (like UI or data access) are built **around and based on that core**.





VS Clean Architecture vs 3-Tier Architecture (a detailed comparison)
When comparing **3-tier (or n-tier) architecture** with **clean architecture**, there ar



Dependency Direction
In **3-tier architecture**, the **Business Logic Layer (BLL)** typically **depends** on the changes, it directly affects the BLL.
But in **clean architecture**, the dependency is **inverted** – the **DAL depends on the logic** from changes in external systems like databases, APIs, etc.

Layer Breakdown

- **UI Layer** (ASP.NET Core Project)
  - Contains: Controllers, ViewModels, Views, Middlewares, Validators, Filte
  - Still the same in both architectures
- **Core Layer** (Business Logic)
  - Includes:
    - Service interfaces (aka **service contracts**)
    - Service implementations
    - DTOs (Data Transfer Objects)
  - Handles: Business rules, app-level logic, validations, and decision-making
  - **Independent** of database or UI. It doesn't know or care whether you're using SQL Server, MongoDB, or Razor Views.

- **Domain Layer** (inside Core)

    - Contains:
        - Entity classes (like Country, Person)
        - Validation attributes (e.g., [Required], [Range])
        - Repository interfaces (e.g., ICountryRepository, IPersonRepository)
    - Purpose: Defines what your app **is**, not how it **works**.
- **Infrastructure Layer** (aka DAL or External Layer)
    - Purpose: Anything external – databases, file systems, 3rd party services, microservice calls.
    - Contains:
        - Implementation of repository interfaces
        - DBContext that accesses Entity classes from Domain
    - Depends on: **Core Layer** (never the other way around)

💬 **About the Repository Pattern**
Yes, **repository layer is technically optional**, but most projects do use it. Some architects skip it, but that's personal preference.
we're **not** talking about **Generic Repository + Unit of Work** here. That pattern is already **built into Entity Framework Core** – no need to manually implement it.
Instead, we're talking about creating **table-specific repositories**, like:

csharp
CopyEdit
ICountryRepository, IPersonRepository
…like you've probably done already in your projects.

🎯 **Summary: Key Differences**

Aspect	3-Tier Architecture	Clean Architecture
Dependency Direction	BLL → DAL	DAL → BLL
Stack Structure	Layered/Stacked	Onion/Concentric
Core Focus	Tightly Coupled	Decoupled & Scalable
Business Logic	Knows about DAL	Totally isolated
Repositories	Used optionally	Interfaces in Domain, Impl in Infra
UI Layer	Same	Same

> So yeah, **Clean Architecture** is all about **protecting your core logic**, keeping things **modular, scalable, and testable**. It's like giving your app boundaries so it doesn't turn into a tangled mess 😅

📦 **Three-Tier Architecture (a.k.a the Classic)**
Layers:
1. **UI Layer (Presentation)** – Your front-end or API endpoints.
2. **Business Logic Layer (BLL)** – Contains service logic, sometimes even talking directly to the DB.
3. **Data Access Layer (DAL)** – Repositories, EF DBContexts, SQL queries, etc.

Flow:
UI ➡️ BLL ➡️ DAL ➡️ DB
Cons:
- BLL is tightly coupled with DAL.
- Any change in DAL can impact BLL directly (😬 not cool for long-term maintenance).
- Not easily testable or scalable in modern cloud-native systems.

🧠 **Clean Architecture (Modern, Maintainable, Scalable AF)**
Key Layers (outside-in):

1. **UI / Presentation Layer** – Your ASP.NET Core controllers, views, Razor pages, etc.
2. **Application / Core Layer** – Contains all your **business logic**, **DTOs**, and **service interfaces**.
3. **Domain Layer** (part of Core) – Your entities and **repository interfaces**.
4. **Infrastructure Layer** – Actual implementation of repositories, DBContexts, API calls, etc.

Flow (Dependency Rule):
Only inward dependency is allowed ⬇️
Infrastructure ➡️ Core
UI ➡️ Core
Core ❌ depends on nothing external

Noteworthy Things You Mentioned:

- Repository Layer = optional ✅
  But still widely used because it makes life easier and separation cleaner.
- Generic Repository + Unit of Work = ❌ needed if using EF Core
  Because EF already implements that internally.
- Your **Core layer should not care** about whether you're using SQL Server or PostgreSQL or even MongoDB.
  That's the Infrastructure layer's job.
- **Domain Layer = entities + repository interfaces**, fully decoupled from infrastructure and UI.
- **Infrastructure Layer =** where DBContext, 3rd party APIs, file system access live.

🎯 **Design Goals of Clean Architecture:**

- ✅ **DB Independent** – swap SQL Server for PostgreSQL? No sweat.
- ✅ **UI Independent** – want to make a mobile app or switch from Razor to Blazor? Go ahead.
- ✅ **Easily Testable** – you can test business logic without hitting real DBs.
- ✅ **Scalable** – scale up (bigger server) or scale out (more servers), it's ready.
- ✅ **Future-Proof** – if traffic spikes, you can handle it.
- ✅ **Maintainable AF** – new devs can jump in without spaghetti nightmares.

🧠 **Interview Pro Tip:**
If you've got **1+ year** experience and you say you've worked with Clean Architecture, **be ready to explain**:

- What the core layer contains
- Why infra depends on core, not vice versa
- How to test business logic without the DB
- How repository interfaces help with decoupling

⚙️ **Clean Architecture = Same Vibes as:**

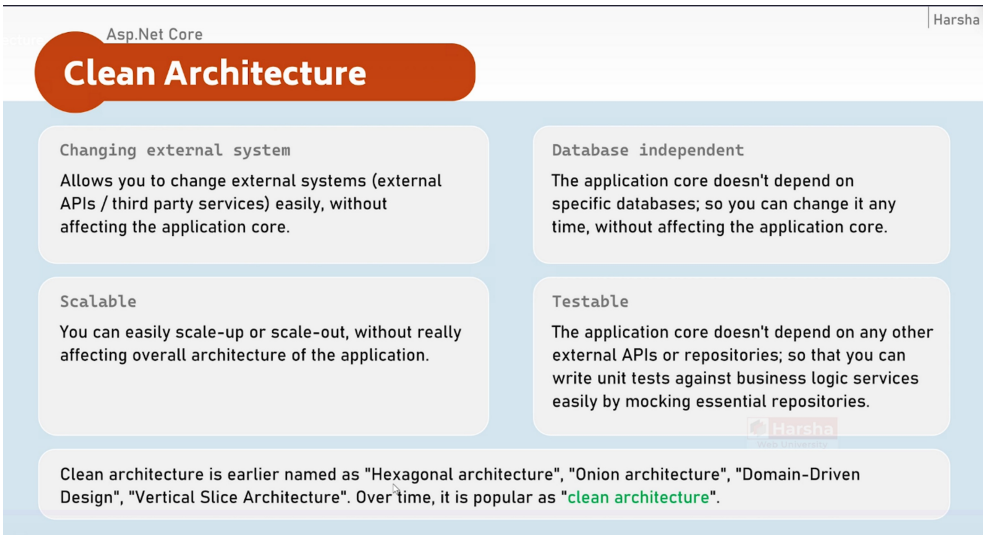
- **Onion Architecture**
- **Hexagonal Architecture**
- **Domain-Driven Design (DDD)**

Different names, same vibes – just a few tweaks in structure or terminology.

🛠️ **You already built all layers, now what?**
You just gotta *reorganize your project structure*:

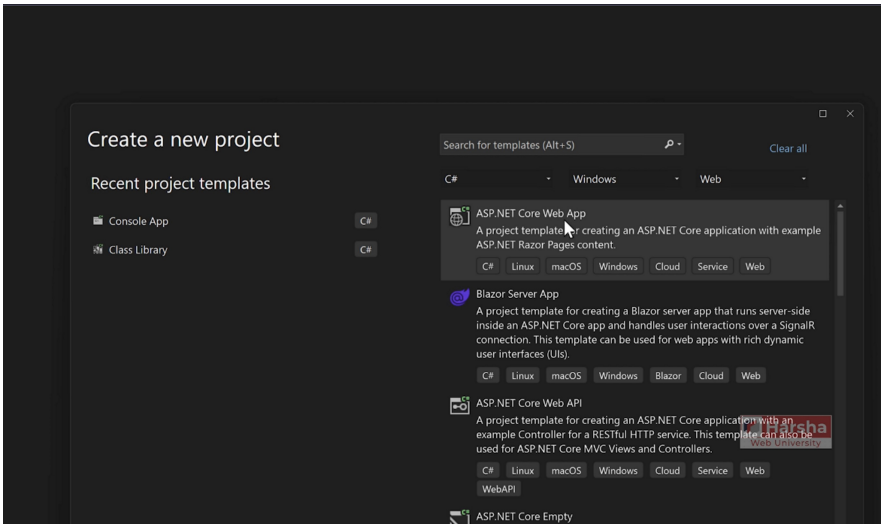
- Move core stuff to Core project
- Put Entities and Repo interfaces in Domain
- Place all EF/infra stuff (like DbContext, repo implementations) in Infrastructure
- Keep Controllers, middlewares, filters in UI

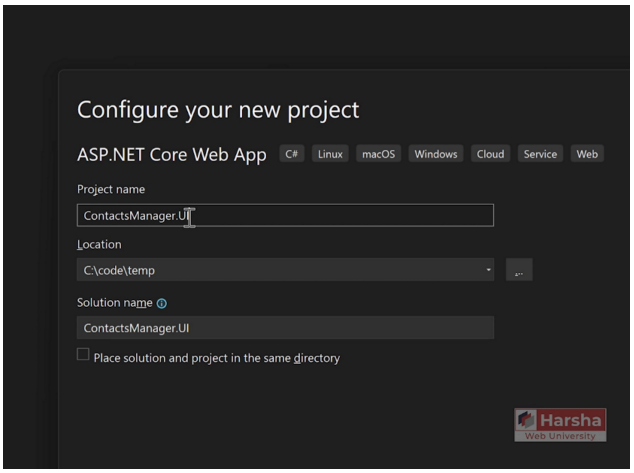
So yeah, Clean Architecture ain't just hype – it's solid for long-term projects, especially if you wanna build maintainable, testable, scalable systems.

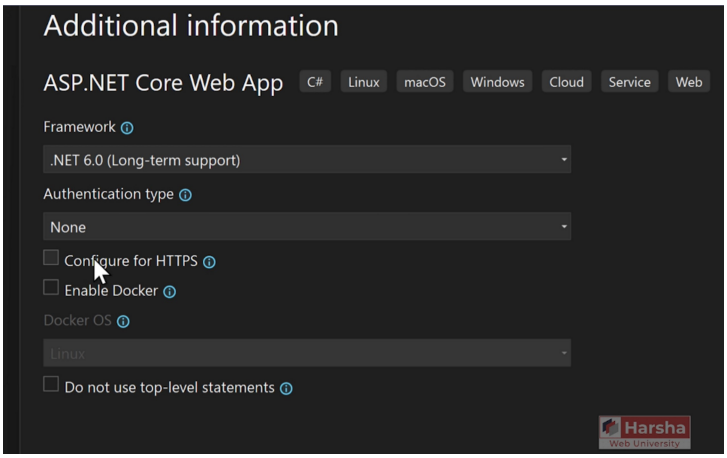


Asp.Net Core

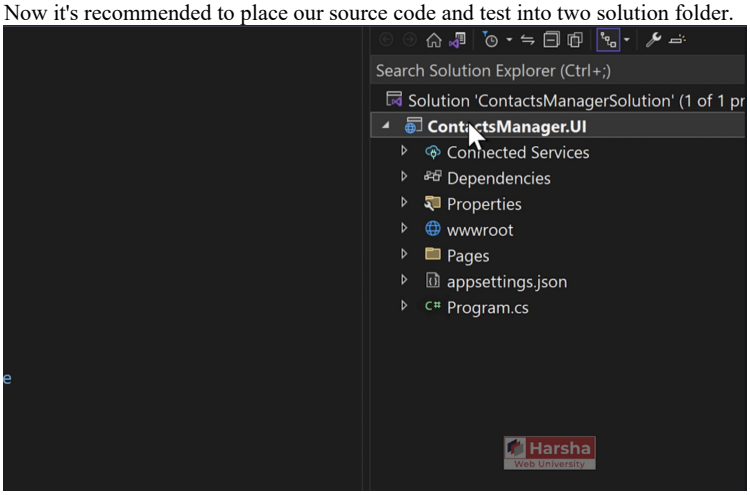
Clean Architecture – Core

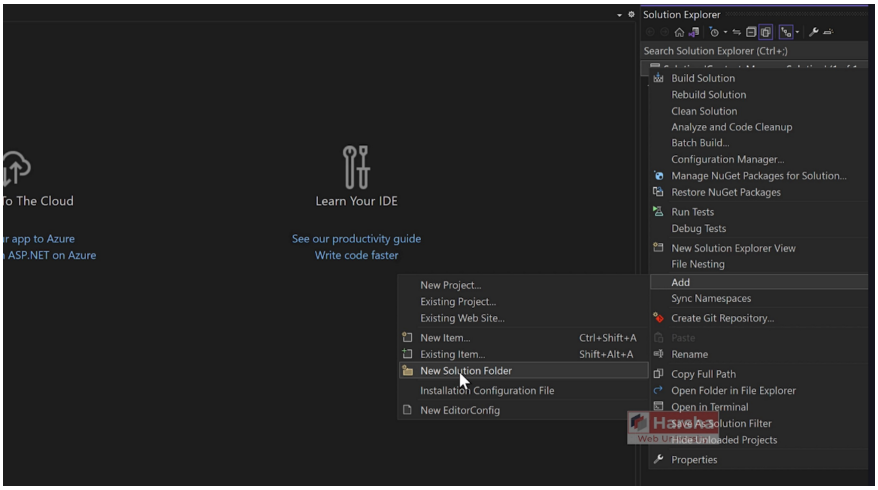


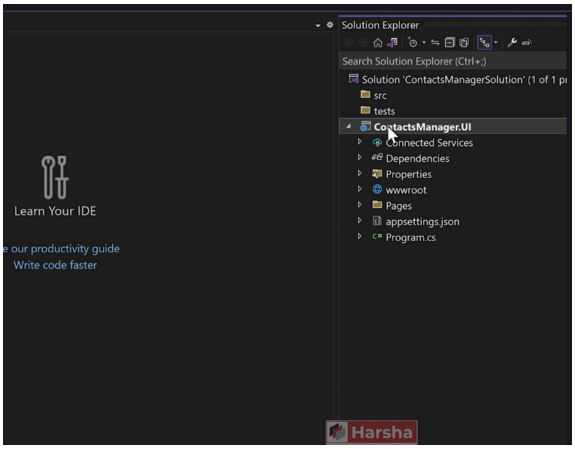


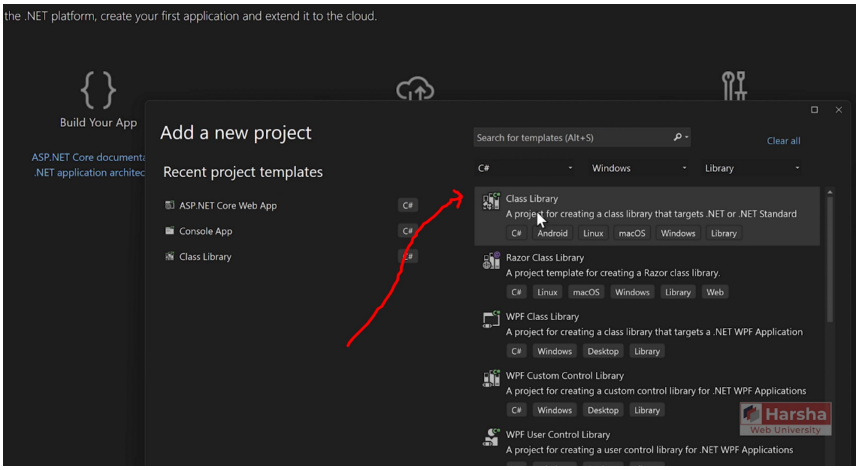


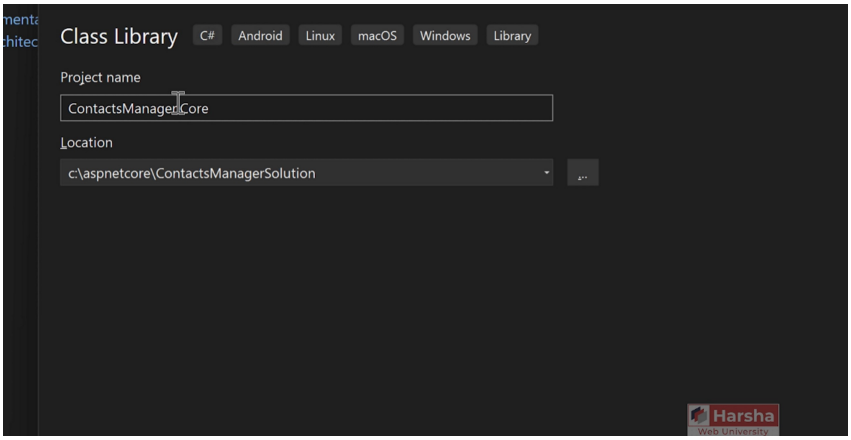
Now it's recommended to place our source code and test into two solution folder.









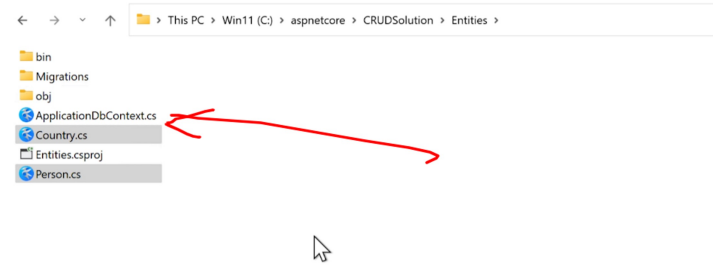


Now from our old 3 tier architecture project, we have to copy these things to our 'Core Project'.

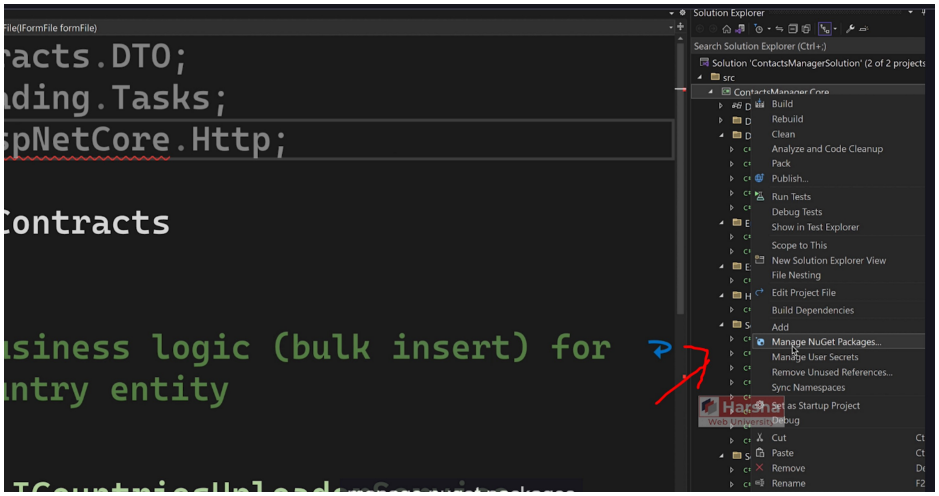
But we should not copy directly. Because these are projects.

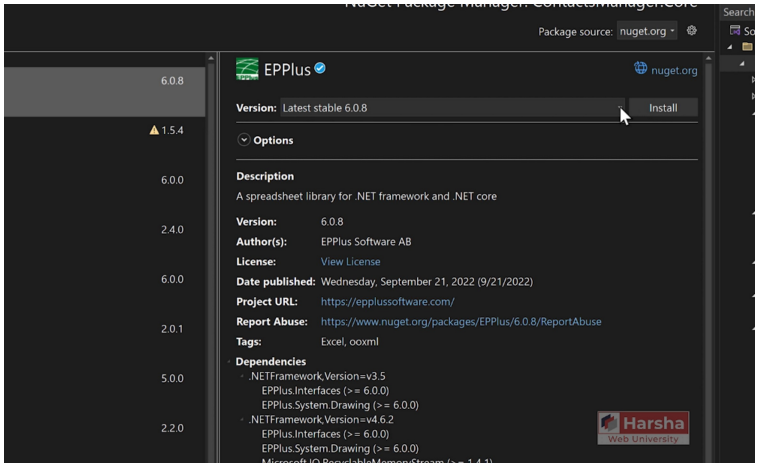
They contain 'bin' and 'object' folder. We have to copy each individual file one by one.

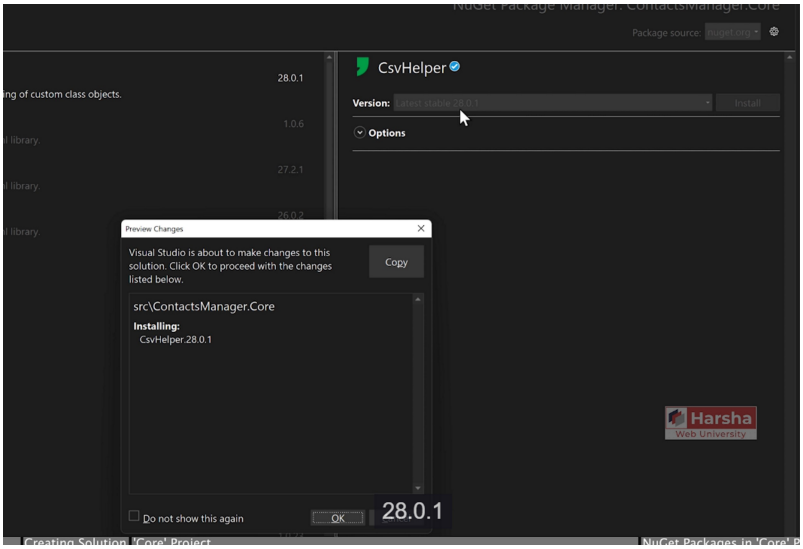
Don't copy ApplicationDbContext.cs file, since it will belong to 'Infrastructure' layer.

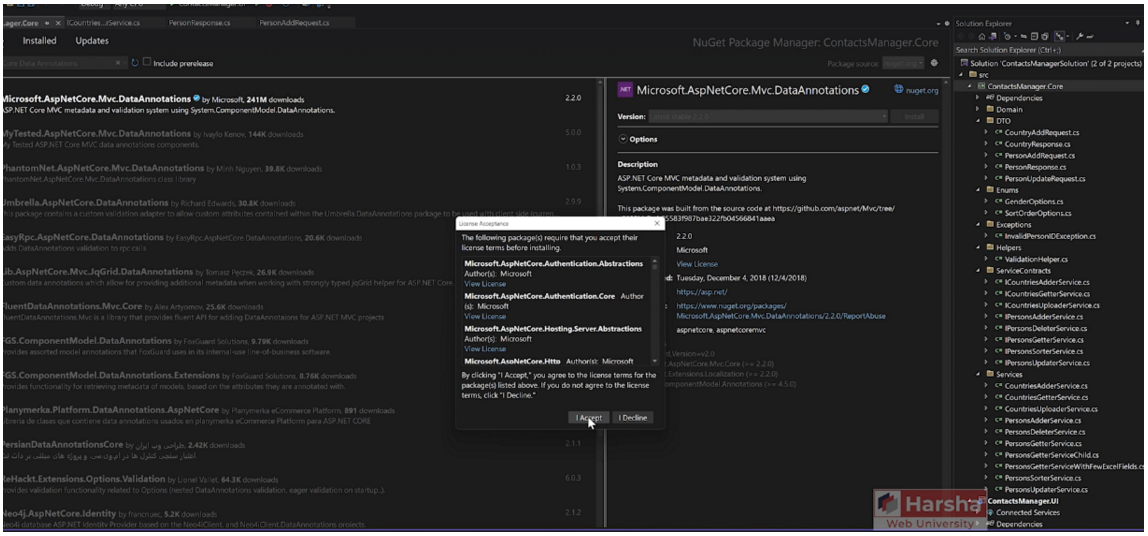


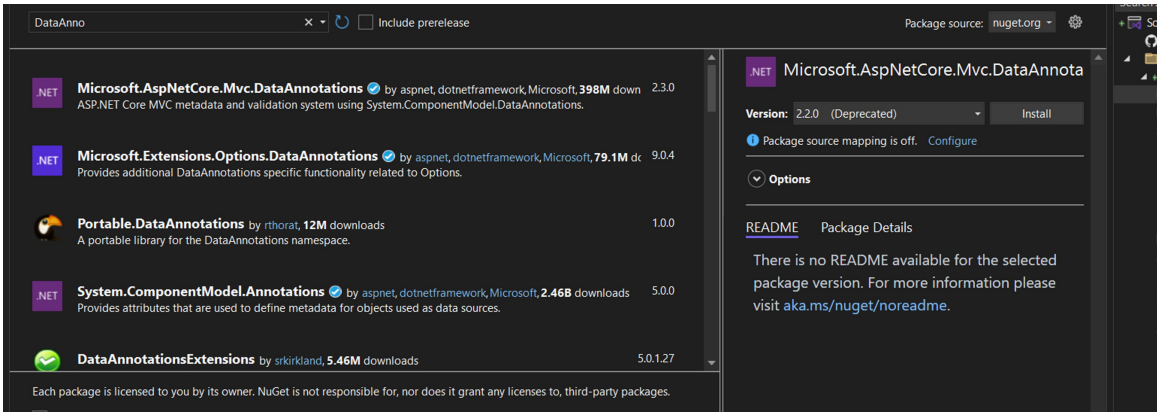
Let’s now install required packages.

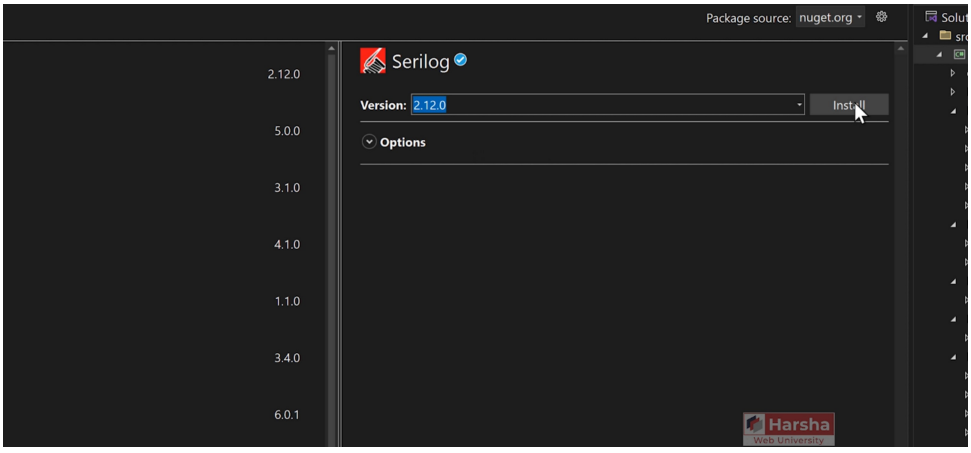

Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'ContactsManagerSolution' (2 of 2 projects)
src
Build
Rebuild
Clean
Analyze and Code Cleanup
Pack
Publish...
Run Tests
Debug Tests
Show in Test Explorer
Scope to This
New Solution Explorer View
File Nesting
Edit Project File
Build Dependencies
Add
Manage NuGet Packages...
Manage User Secrets
Remove Unused References...
Sync Namespaces
Set as Startup Project
Debug
Cut
Paste
Remove
Rename
acts.DTO;
ding.Tasks;
pNetCore.Http;
ontracts
usiness logic (bulk insert) for
ntry entity

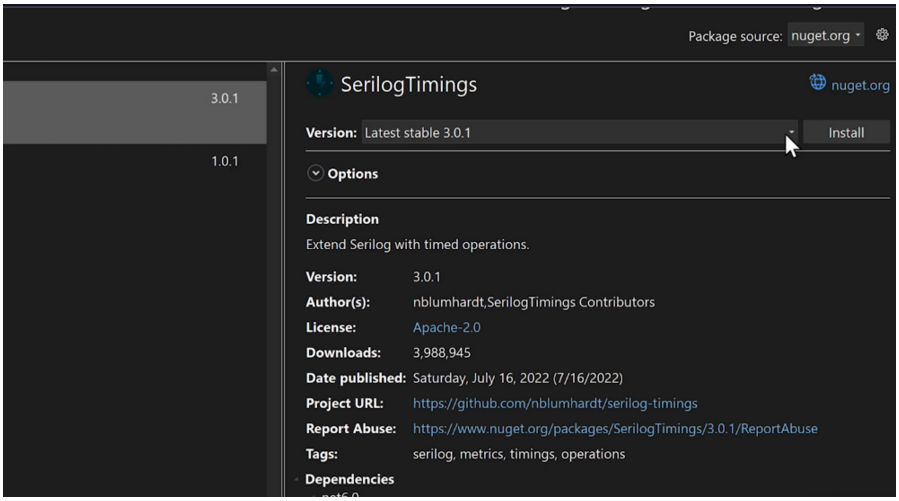

Package source: nuget.org
EPPlus
nuget.org
Version: Latest stable 6.0.8
Install
Options
Description
A spreadsheet library for .NET framework and .NET core
Version: 6.0.8
Author(s): EPPlus Software AB
License: View License
Date published: Wednesday, September 21, 2022 (9/21/2022)
Project URL: https://epplussoftware.com/
Report Abuse: https://www.nuget.org/packages/EPPlus/6.0.8/ReportAbuse
Tags: Excel, ooxml
Dependencies
.NETFramework,Version=v3.5
EPPlus.Interfaces (>= 6.0.0)
EPPlus.System.Drawing (>= 6.0.0)
.NETFramework,Version=v4.6.2
EPPlus.Interfaces (>= 6.0.0)
EPPlus.System.Drawing (>= 6.0.0)
6.0.8
1.5.4
6.0.0
2.4.0
6.0.0
2.0.1
5.0.0
2.2.0
Harsha
Web University

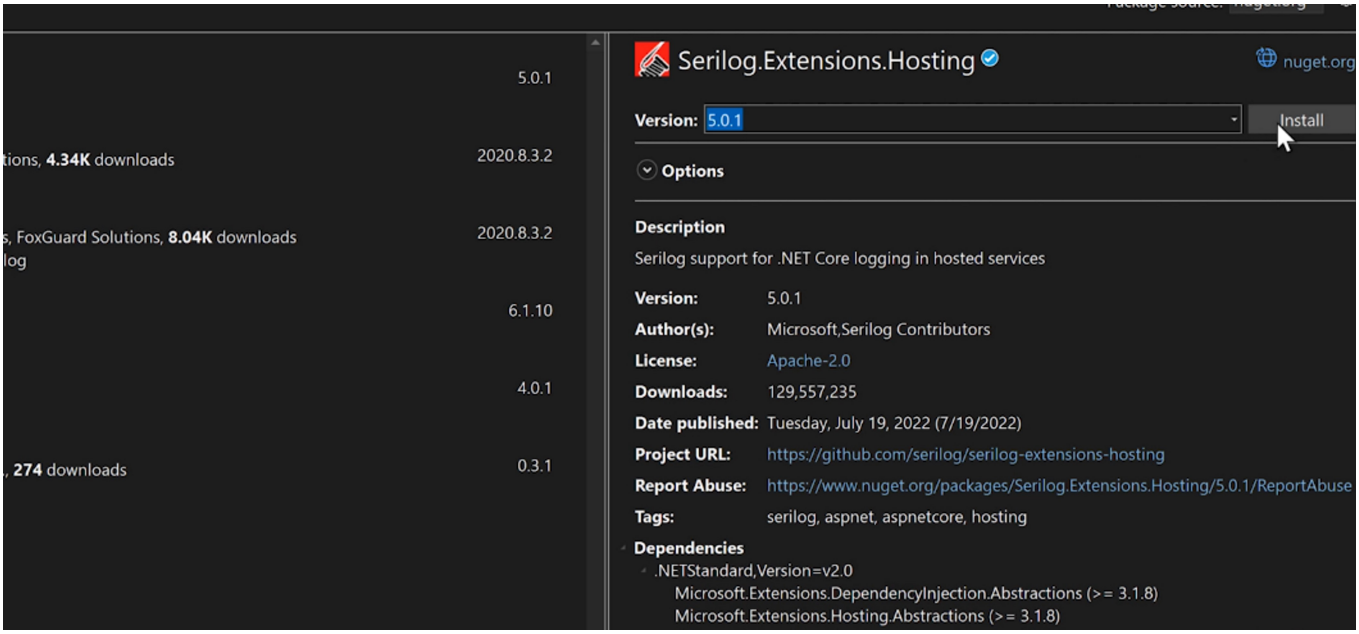

Package source: nuget.org
CsvHelper
Version:
Install
Options
28.0.1
1.0.6
27.2.1
Preview Changes
Visual Studio is about to make changes to this solution. Click OK to proceed with the changes listed below.
Copy
src\ContactsManager.Core
Installing:
CsvHelper.28.0.1
Do not show this again
OK
28.0.1
Harsha
Web University


Installed
Updates
NuGet Package Manager: ContactsManager.Core
Include prerelease
Microsoft.AspNetCore.Mvc.DataAnnotations by Microsoft, 241M downloads
ASP.NET Core MVC metadata and validation system using System.ComponentModel.DataAnnotations.
2.2.0
Microsoft.AspNetCore.Mvc.DataAnnotations
nuget.org
Version:
Install
Options
Description
ASP.NET Core MVC metadata and validation system using System.ComponentModel.DataAnnotations.
This package was built from the source code at https://github.com/aspnet/Mvc/tree/
License Acceptance
The following package(s) require that you accept their license terms before installing.
Microsoft.AspNetCore.Authentication.Abstractions
Author(s): Microsoft
View License
Microsoft.AspNetCore.Authentication.Core
Author(s): Microsoft
View License
Microsoft.AspNetCore.Hosting.Server.Abstractions
Author(s): Microsoft
View License
Microsoft.AspNetCore.Http
Author(s): Microsoft
By clicking "I Accept," you agree to the license terms for the package(s) listed above. If you do not agree to the license terms, click "I Decline."
I Accept
I Decline
Solution Explorer
Solution 'ContactsManagerSolution' (2 of 2 projects)
ContactsManager.Core
Dependencies
Domain
DTO
Enums
Exceptions
Helpers
ServiceContracts
Services
ContactsManager.UI
Connected Services
Dependencies
Harsha
Web University


DataAnno
Include prerelease
Package source: nuget.org
Microsoft.AspNetCore.Mvc.DataAnnotations by aspnet, dotnetframework, Microsoft, 398M down
ASP.NET Core MVC metadata and validation system using System.ComponentModel.DataAnnotations.
2.3.0
Microsoft.Extensions.Options.DataAnnotations by aspnet, dotnetframework, Microsoft, 79.1M do
Provides additional DataAnnotations specific functionality related to Options.
9.0.4
Portable.DataAnnotations by rthorat, 12M downloads
A portable library for the DataAnnotations namespace.
1.0.0
System.ComponentModel.Annotations by aspnet, dotnetframework, Microsoft, 2.46B downloads
Provides attributes that are used to define metadata for objects used as data sources.
5.0.0
DataAnnotationsExtensions by srkirkland, 5.46M downloads
5.0.1.27
Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.
Microsoft.AspNetCore.Mvc.DataAnnota
Version: 2.2.0 (Deprecated)
Install
Package source mapping is off. Configure
Options
README
Package Details
There is no README available for the selected package version. For more information please visit aka.ms/nuget/noreadme.

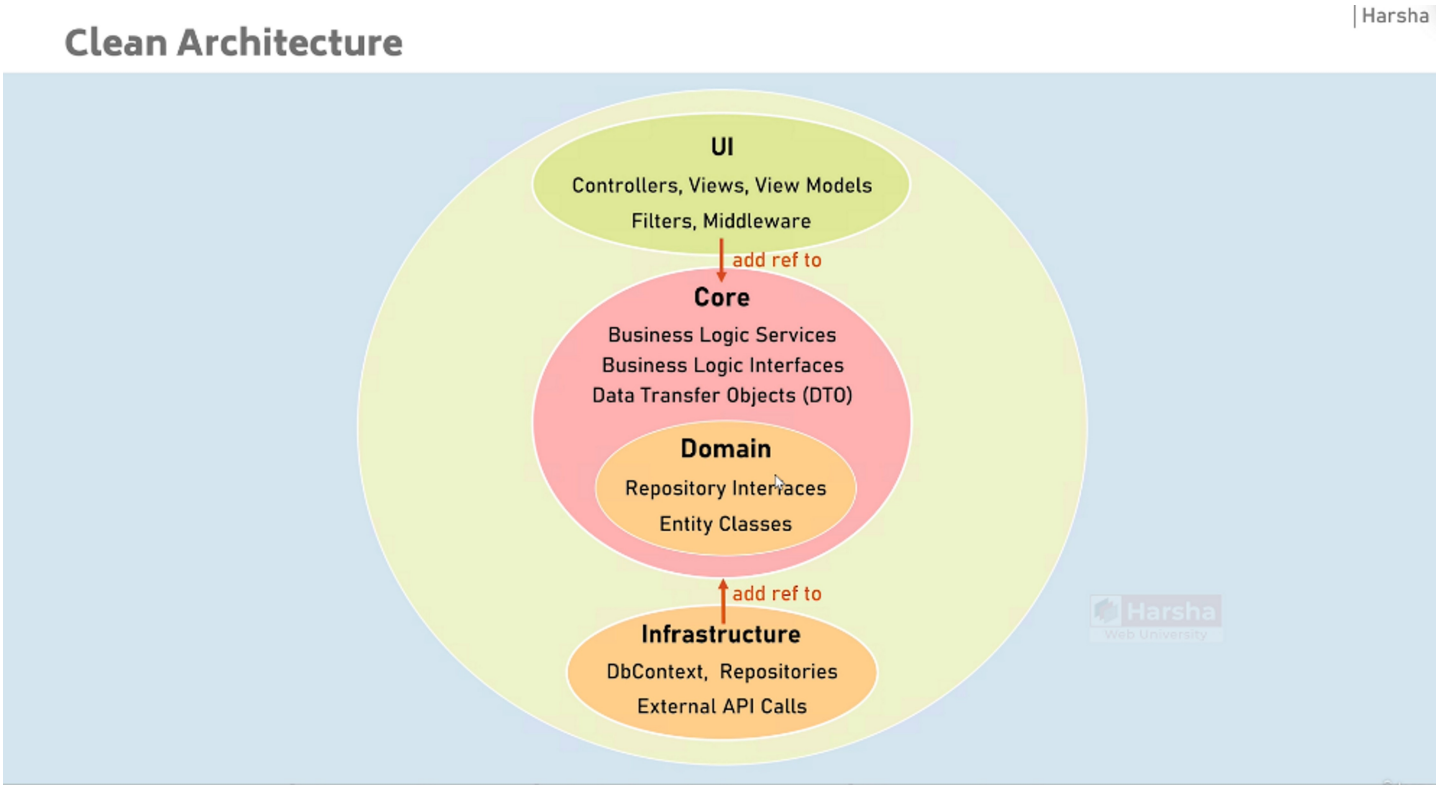

Package source: nuget.org
2.12.0
5.0.0
3.1.0
4.1.0
1.1.0
3.4.0
6.0.1
Serilog
Version: 2.12.0
Install
Options
Harsha
Web University


Package source: nuget.org
3.0.1
1.0.1
SerilogTimings
nuget.org
Version: Latest stable 3.0.1
Install
Options
Description
Extend Serilog with timed operations.
Version: 3.0.1
Author(s): nblumhardt,SerilogTimings Contributors
License: Apache-2.0
Downloads: 3,988,945
Date published: Saturday, July 16, 2022 (7/16/2022)
Project URL: https://github.com/nblumhardt/serilog-timings
Report Abuse: https://www.nuget.org/packages/SerilogTimings/3.0.1/ReportAbuse
Tags: serilog, metrics, timings, operations
Dependencies


5.0.1
tions, 4.34K downloads
2020.8.3.2
s, FoxGuard Solutions, 8.04K downloads
log
2020.8.3.2
6.1.10
4.0.1
, 274 downloads
0.3.1
Serilog.Extensions.Hosting
nuget.org
Version: 5.0.1
Install
Options
Description
Serilog support for .NET Core logging in hosted services
Version: 5.0.1
Author(s): Microsoft,Serilog Contributors
License: Apache-2.0
Downloads: 129,557,235
Date published: Tuesday, July 19, 2022 (7/19/2022)
Project URL: https://github.com/serilog/serilog-extensions-hosting
Report Abuse: https://www.nuget.org/packages/Serilog.Extensions.Hosting/5.0.1/ReportAbuse
Tags: serilog, aspnet, aspnetcore, hosting
Dependencies
.NETStandard,Version=v2.0
Microsoft.Extensions.DependencyInjection.Abstractions (>= 3.1.8)
Microsoft.Extensions.Hosting.Abstractions (>= 3.1.8)

Asp.Net Core

Clean Architecture – Infrastructure



🎯 **Core Recap:**
You've already built the **Core Layer**, which is like the *heart* of your application.
It includes:

- 💡 **Domain Entities** (like User, Order, Product, etc.)
- 🧩 **Interfaces** for Repositories and Services (contract only, no implementation)
- 🧠 **Business Logic**, Use Cases, Rules

Note: This layer is **pure C#**, no NuGet packages for DB, email, or whatever. Just logic + contracts.

💪 **Now We Build the Infrastructure Layer.**

What's in the Infrastructure Layer?

- Entity Framework DbContext
- Repository **implementations** (concrete classes based on interfaces from Core)
- Email sender services (like SendGrid)
- Cloud services (e.g., Microsoft Azure Blob Storage, AWS S3)
- API Clients (e.g., third-party REST APIs)

Dependency Rule:

Infrastructure Layer → depends on → **Core Layer**

✗ NEVER the other way around.

So, in your .csproj of the Infrastructure project, you add:

```
<ProjectReference Include="..\YourProject.Core\YourProject.Core.csproj" />
```

That way, you can:

- Implement interfaces from the **Core**
- Use domain entities from the **Core**

Example:

```
public class UserRepository : IUserRepository
{
   private readonly YourDbContext _context;
public UserRepository(YourDbContext context)
   {
      _context = context;
   }
public Task<User> GetUserByIdAsync(Guid id)
   {
      return _context.Users.FindAsync(id).AsTask();
   }
}
```

Here IUserRepository is from Core → UserRepository is in Infrastructure.

Reminder: Clean vs Three-Tier

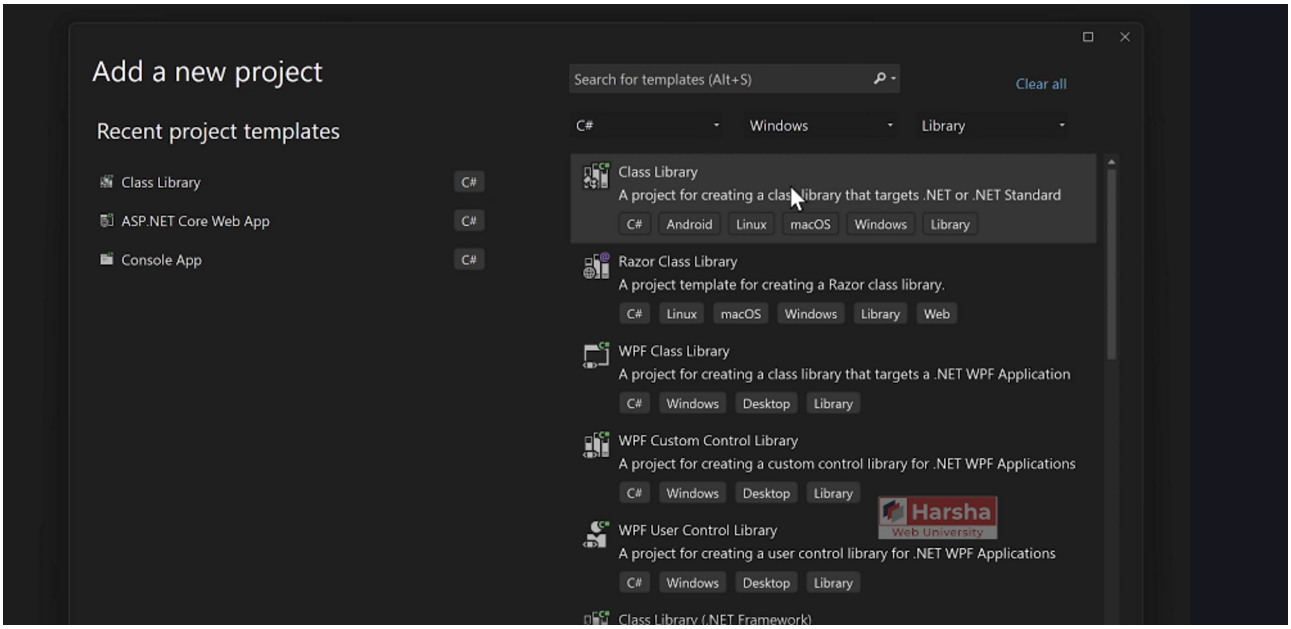
	Three-Tier	Clean Architecture
BLL → DAL	Tightly coupled	Loosely coupled
Data access logic	Spread in BLL	Encapsulated in Infra
Swappable DB	✗ Hard	✓ Easy
Testable Core	✗ Painful	✓ Cakewalk

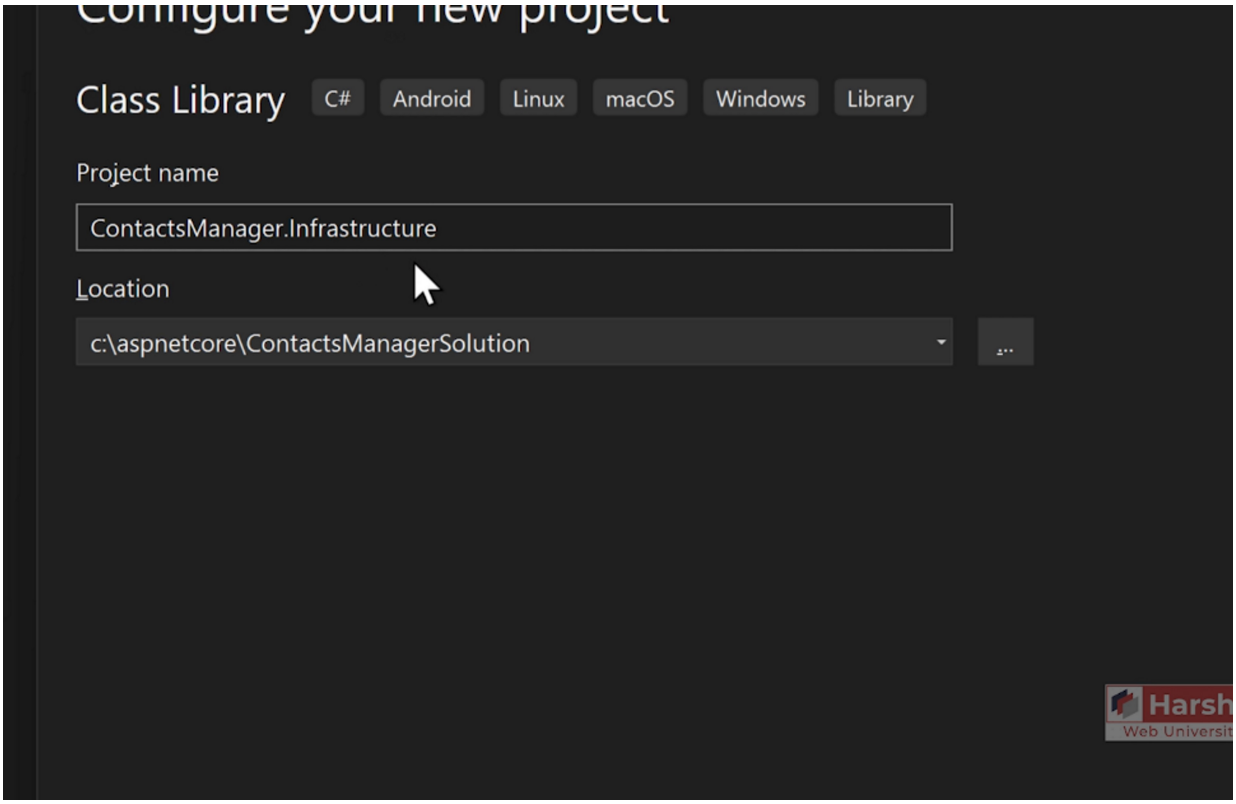
Takeaway:

You're making your app **future-proof**:

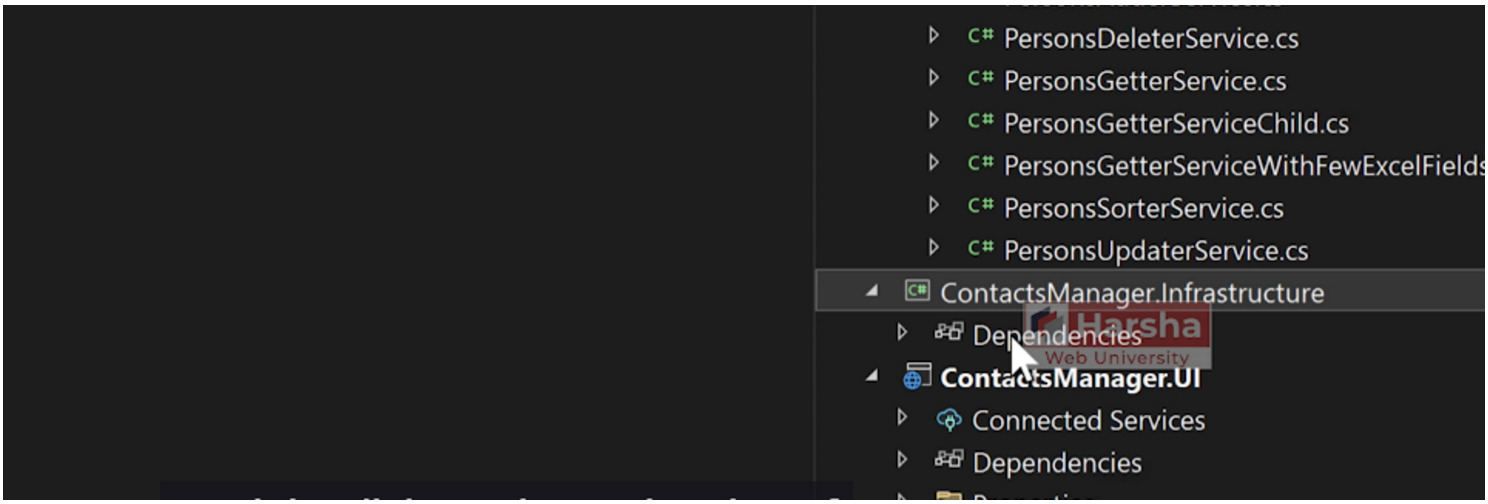
- Want to switch from SQL Server to PostgreSQL? ✓ Easy
- Want to swap SendGrid for MailGun? ✓ Easy
- Want to write unit tests for business logic without DB? ✓ EASY!

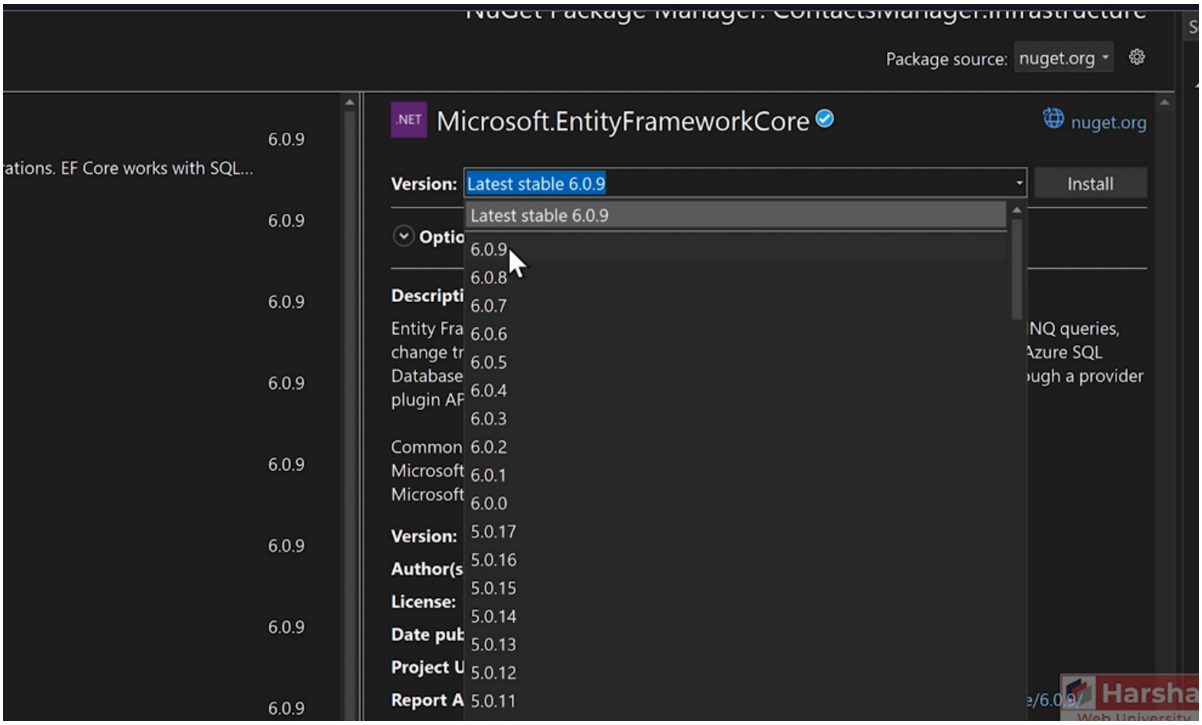
Click on the 'src' folder and then

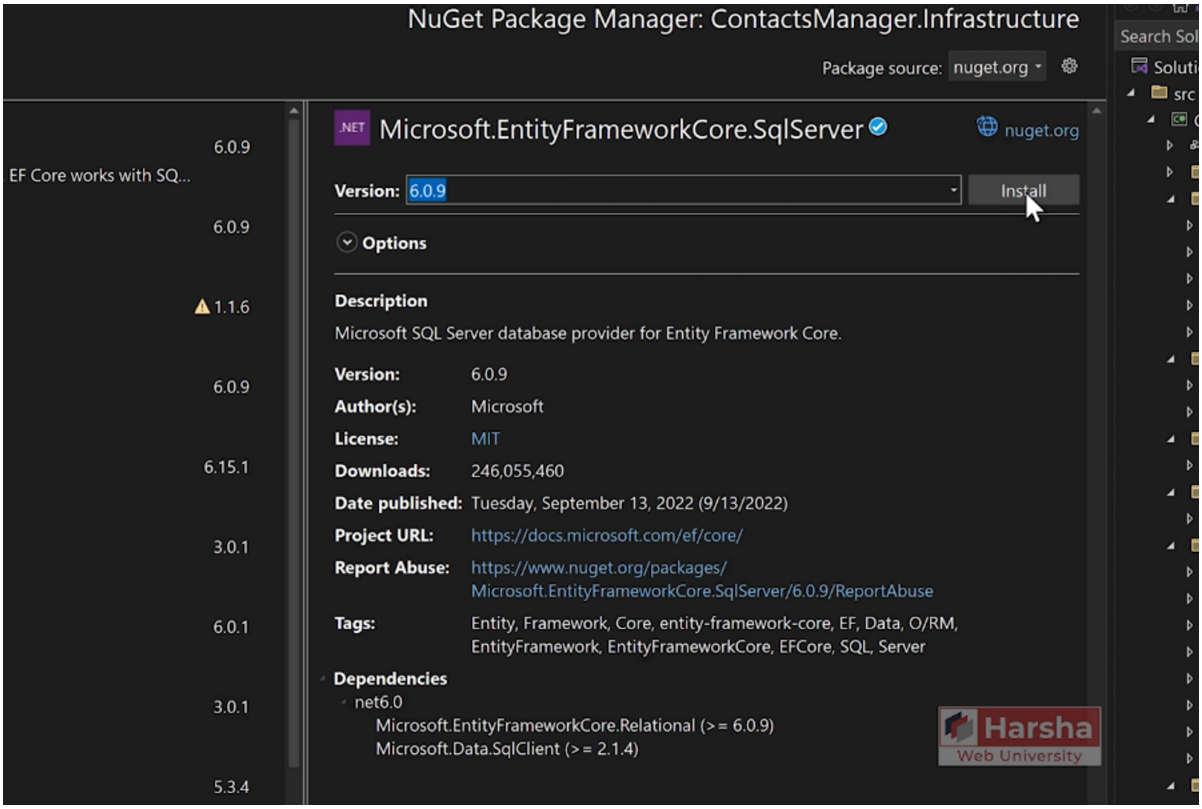




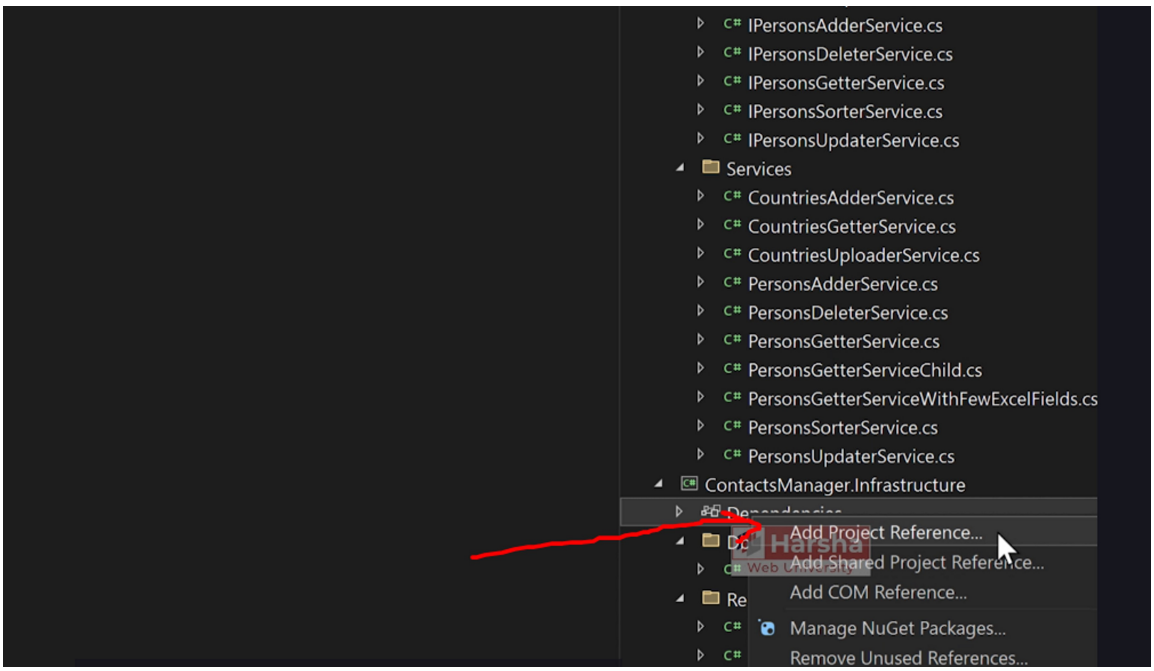
Now we have to install some nuget packages.

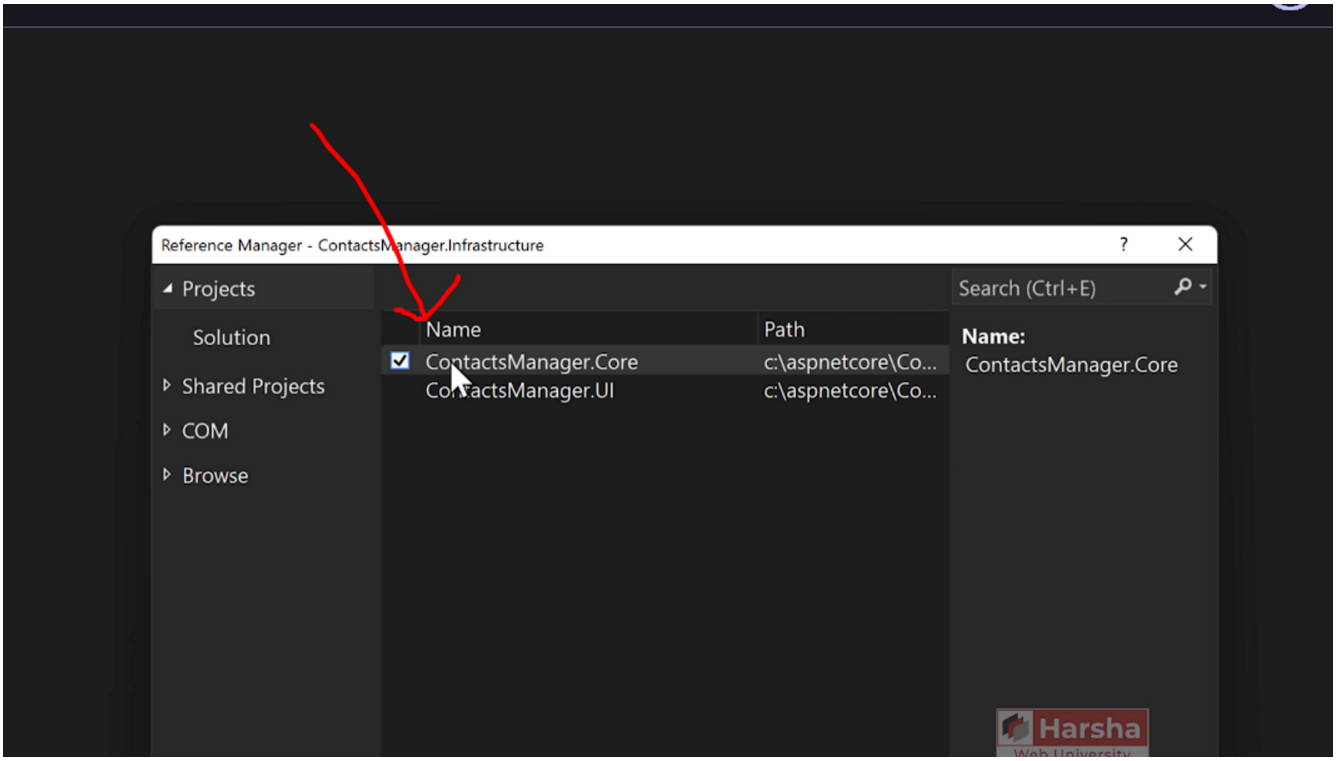

PersonsDeleterService.cs
PersonsGetterService.cs
PersonsGetterServiceChild.cs
PersonsGetterServiceWithFewExcelFields
PersonsSorterService.cs
PersonsUpdaterService.cs
ContactsManager.Infrastructure
Dependencies
ContactsManager.UI
Connected Services
Dependencies


Package source: nuget.org
Microsoft.EntityFrameworkCore
nuget.org
Version: Latest stable 6.0.9
Install
Latest stable 6.0.9
6.0.9
6.0.8
6.0.7
6.0.6
6.0.5
6.0.4
6.0.3
6.0.2
6.0.1
6.0.0
5.0.17
5.0.16
5.0.15
5.0.14
5.0.13
5.0.12
5.0.11



Now 'Infrastructure' project should add reference to the 'Core' Project. That's main part of the Clean Architecture.

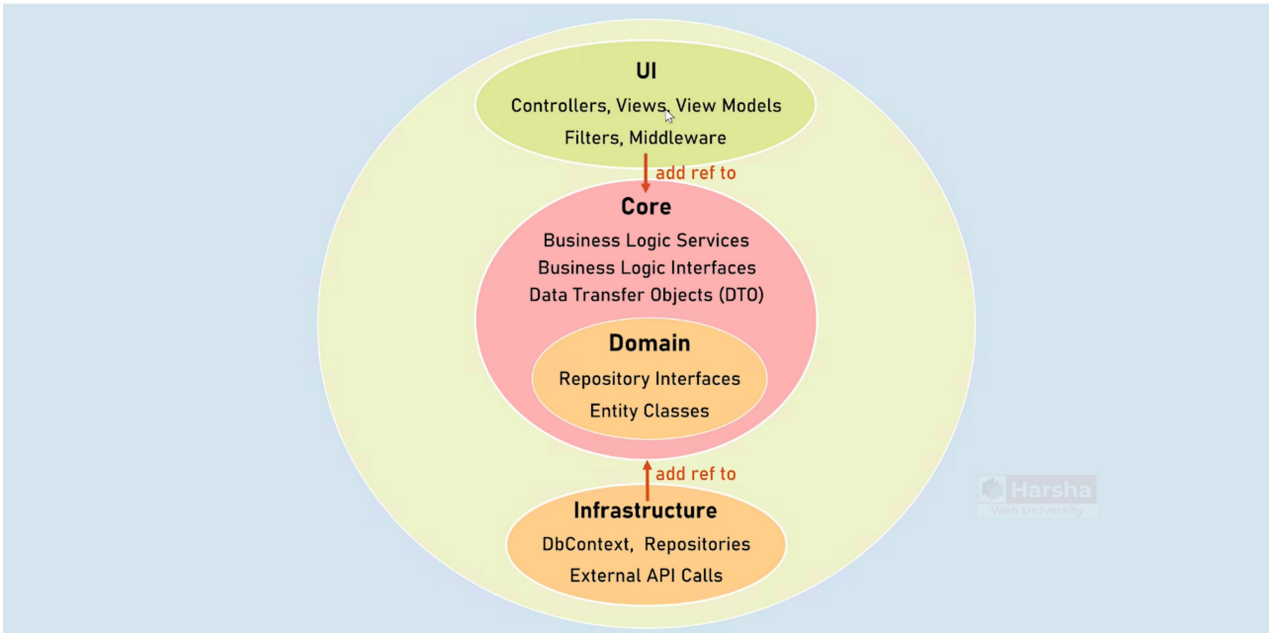


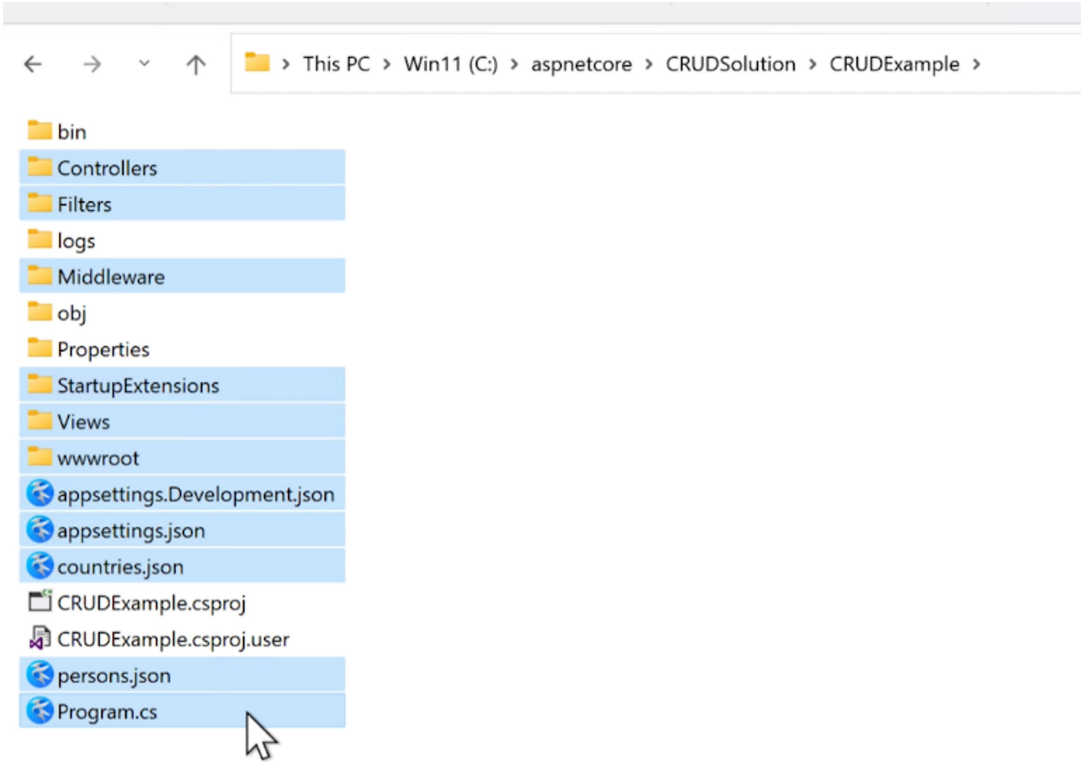


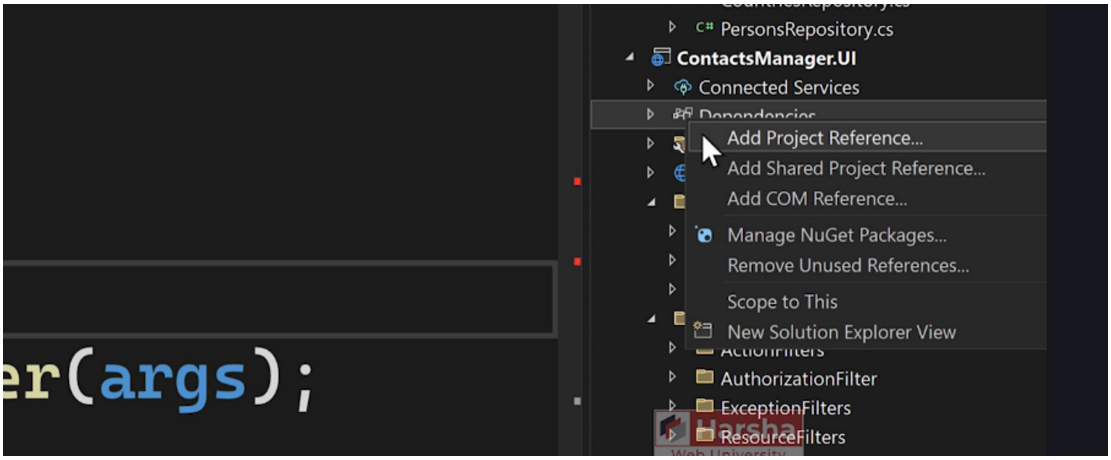
Asp.Net Core

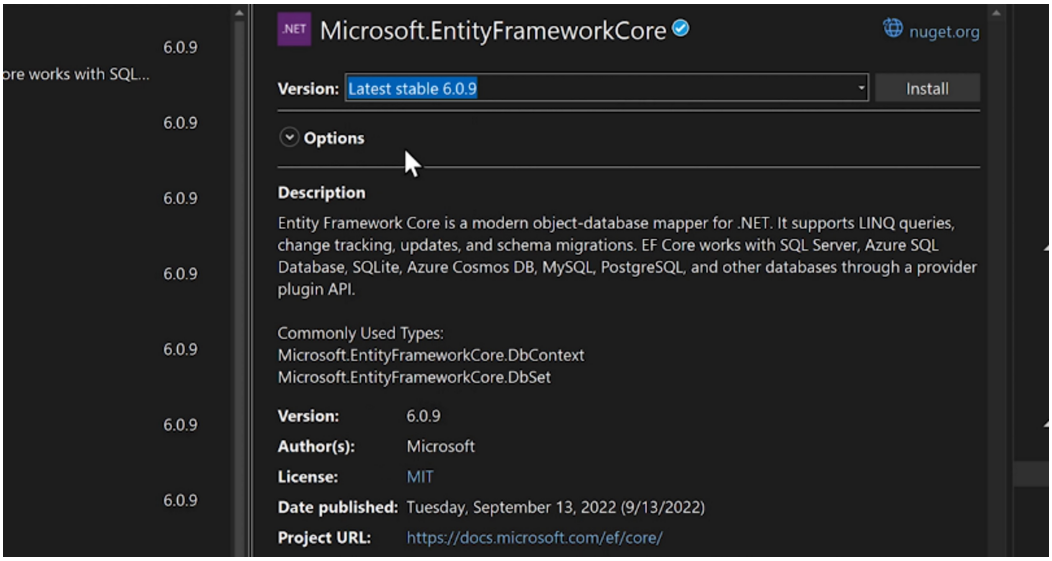
Clean Architecture – UI

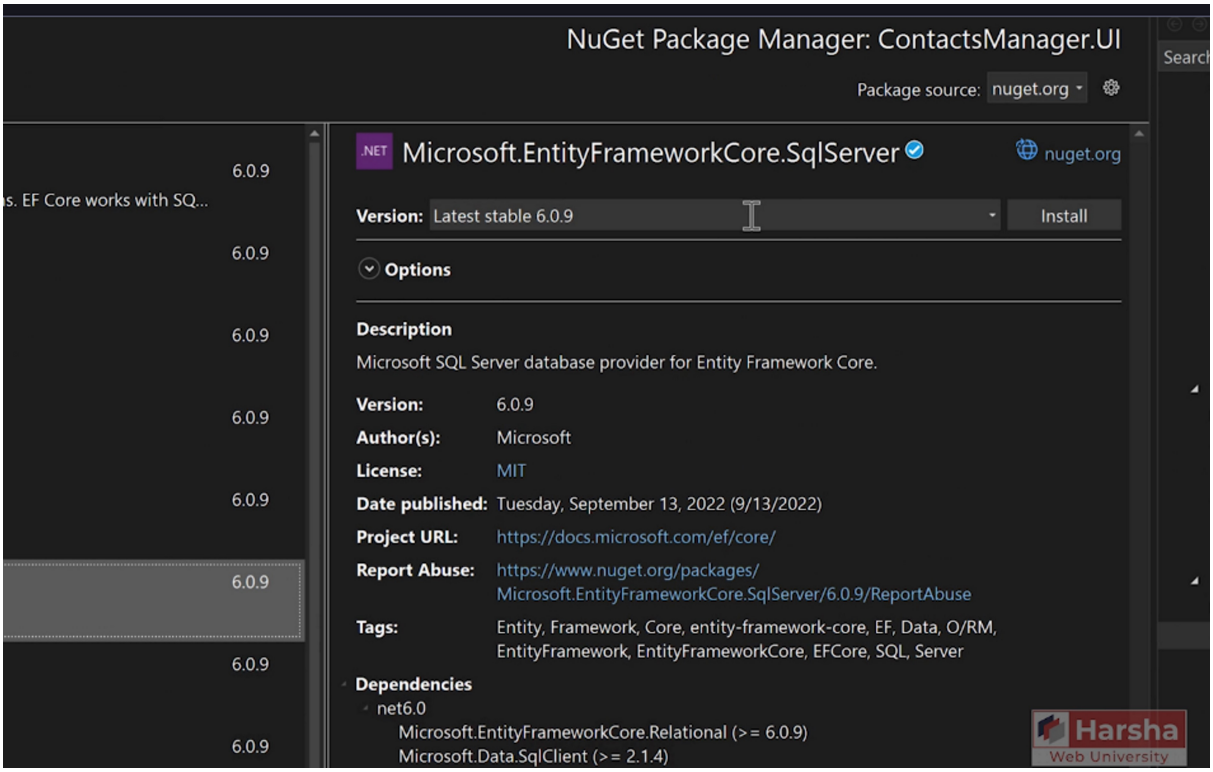
Clean Architecture

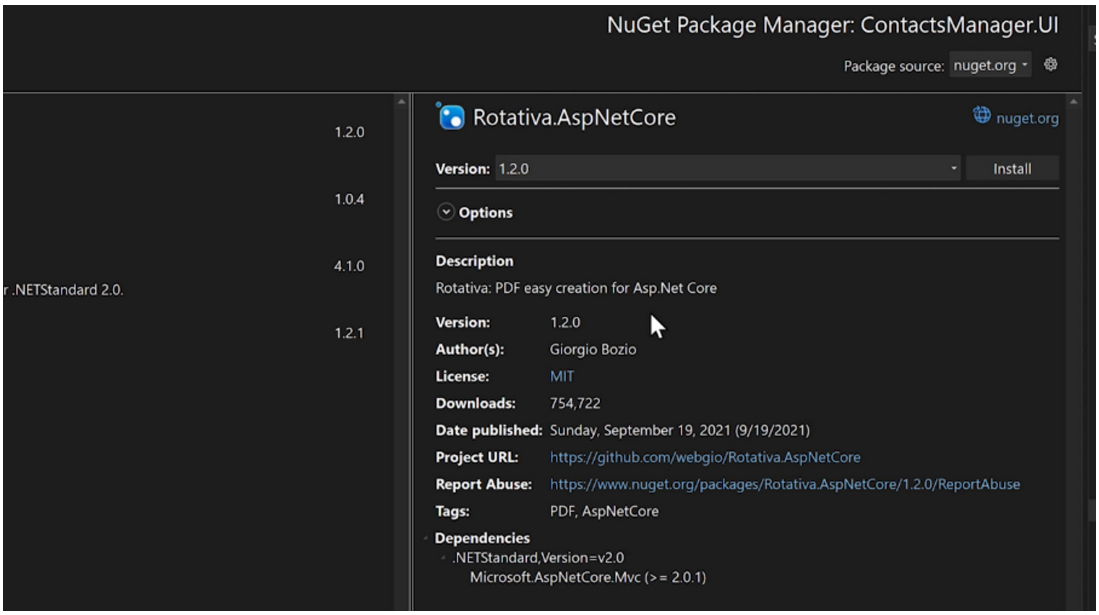


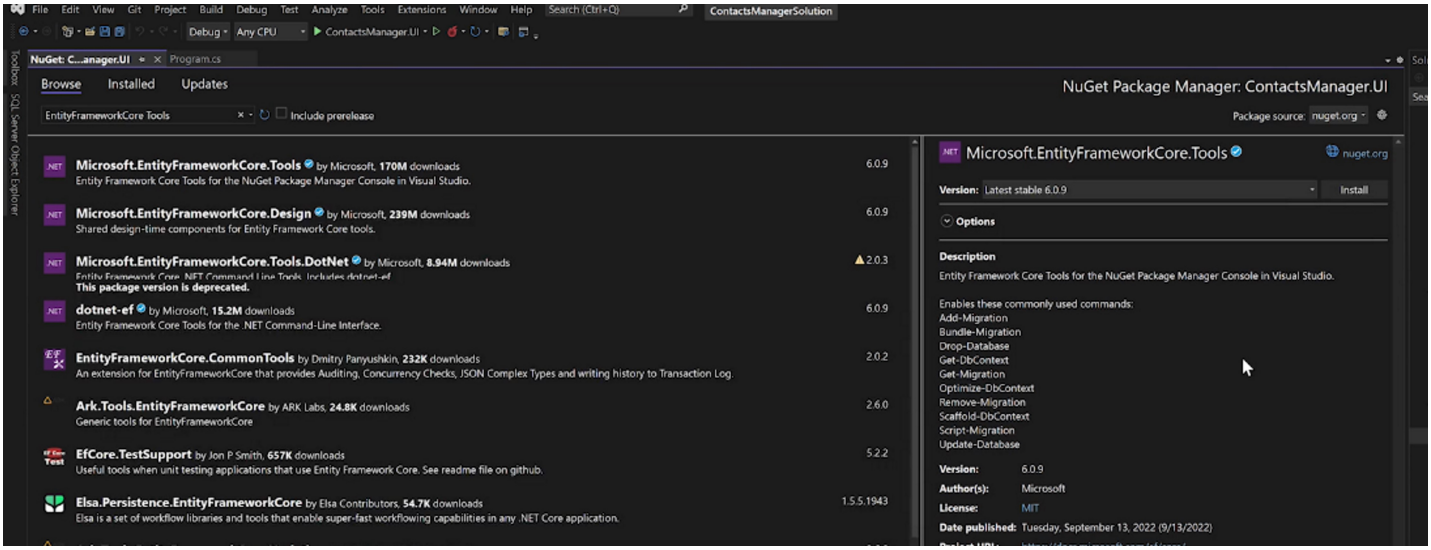


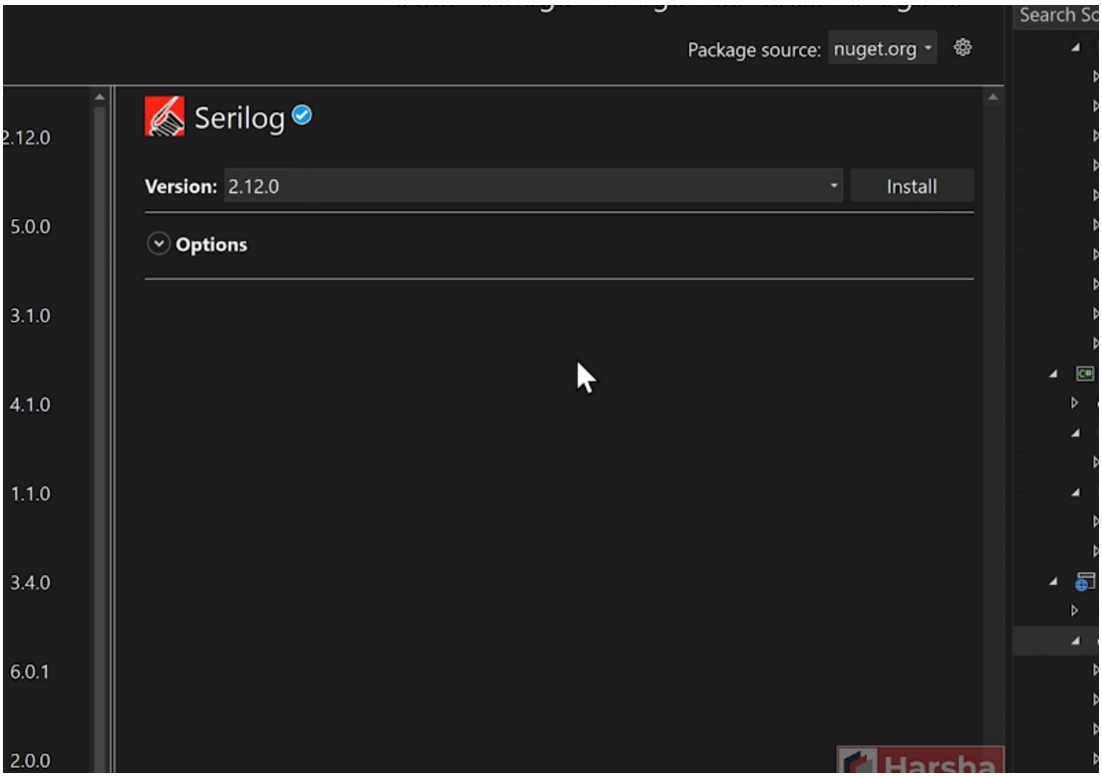

er(args);
PersonsRepository.cs
ContactsManager.UI
Connected Services
Add Project Reference...
Add Shared Project Reference...
Add COM Reference...
Manage NuGet Packages...
Remove Unused References...
Scope to This
New Solution Explorer View
AuthorizationFilter
ExceptionFilters
ResourceFilters

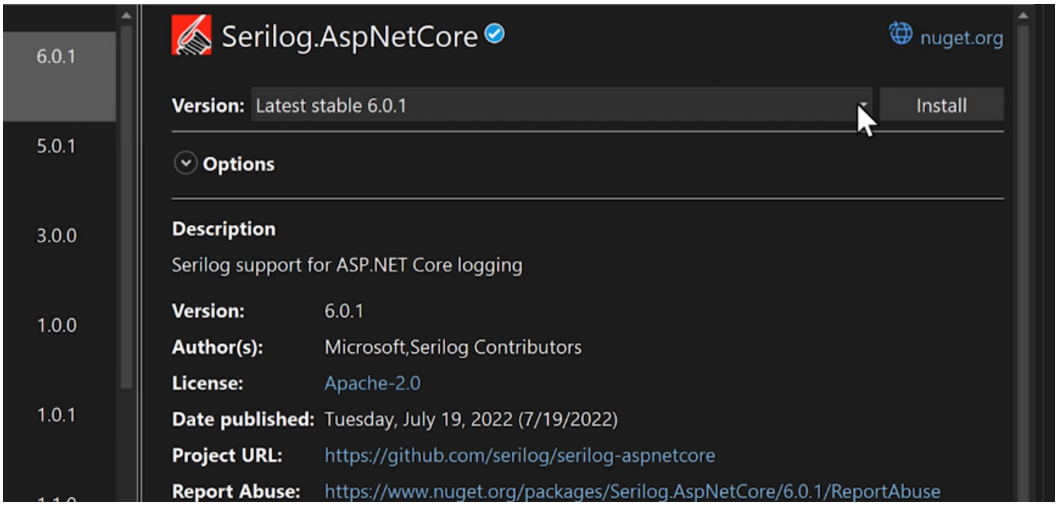

6.0.9
ore works with SQL...
6.0.9
6.0.9
6.0.9
6.0.9
6.0.9
Microsoft.EntityFrameworkCore
nuget.org
Version: Latest stable 6.0.9
Install
Options
Description
Entity Framework Core is a modern object-database mapper for .NET. It supports LINQ queries, change tracking, updates, and schema migrations. EF Core works with SQL Server, Azure SQL Database, SQLite, Azure Cosmos DB, MySQL, PostgreSQL, and other databases through a provider plugin API.
Commonly Used Types:
Microsoft.EntityFrameworkCore.DbContext
Microsoft.EntityFrameworkCore.DbSet
Version: 6.0.9
Author(s): Microsoft
License: MIT
Date published: Tuesday, September 13, 2022 (9/13/2022)
Project URL: https://docs.microsoft.com/ef/core/

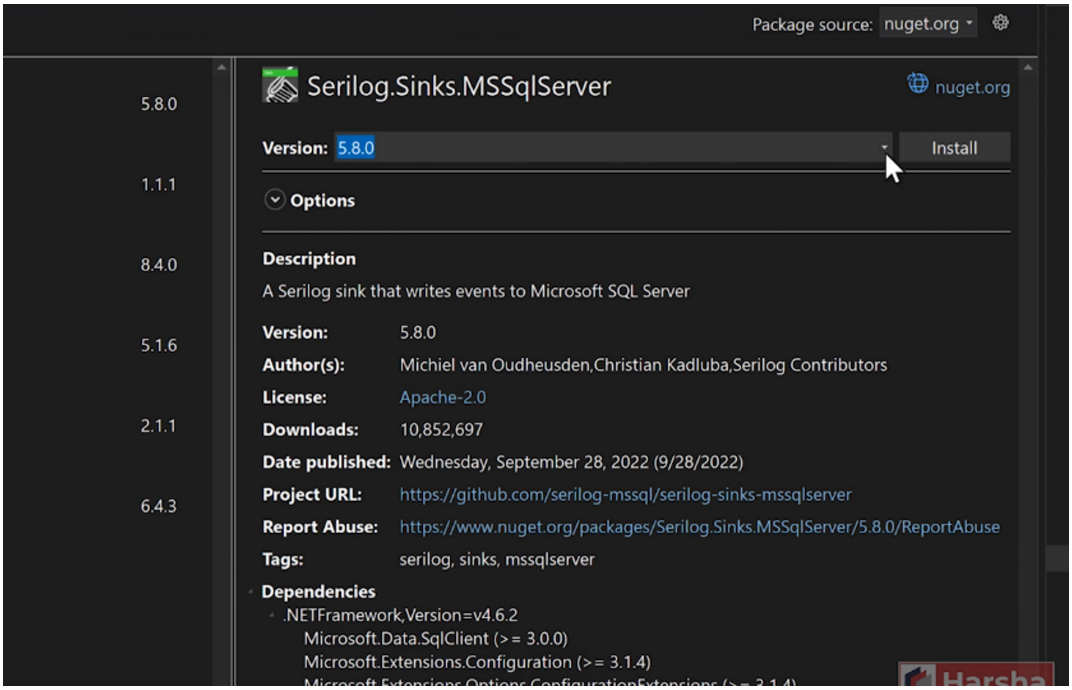

NuGet Package Manager: ContactsManager.UI
Package source: nuget.org
Search
6.0.9
s. EF Core works with SQ...
6.0.9
6.0.9
6.0.9
6.0.9
6.0.9
6.0.9
.NET
Microsoft.EntityFrameworkCore.SqlServer
nuget.org
Version: Latest stable 6.0.9
Install
Options
Description
Microsoft SQL Server database provider for Entity Framework Core.
Version: 6.0.9
Author(s): Microsoft
License: MIT
Date published: Tuesday, September 13, 2022 (9/13/2022)
Project URL: https://docs.microsoft.com/ef/core/
Report Abuse: https://www.nuget.org/packages/Microsoft.EntityFrameworkCore.SqlServer/6.0.9/ReportAbuse
Tags: Entity, Framework, Core, entity-framework-core, EF, Data, O/RM, EntityFramework, EntityFrameworkCore, EFCore, SQL, Server
Dependencies
net6.0
Microsoft.EntityFrameworkCore.Relational (>= 6.0.9)
Microsoft.Data.SqlClient (>= 2.1.4)
Harsha
Web University

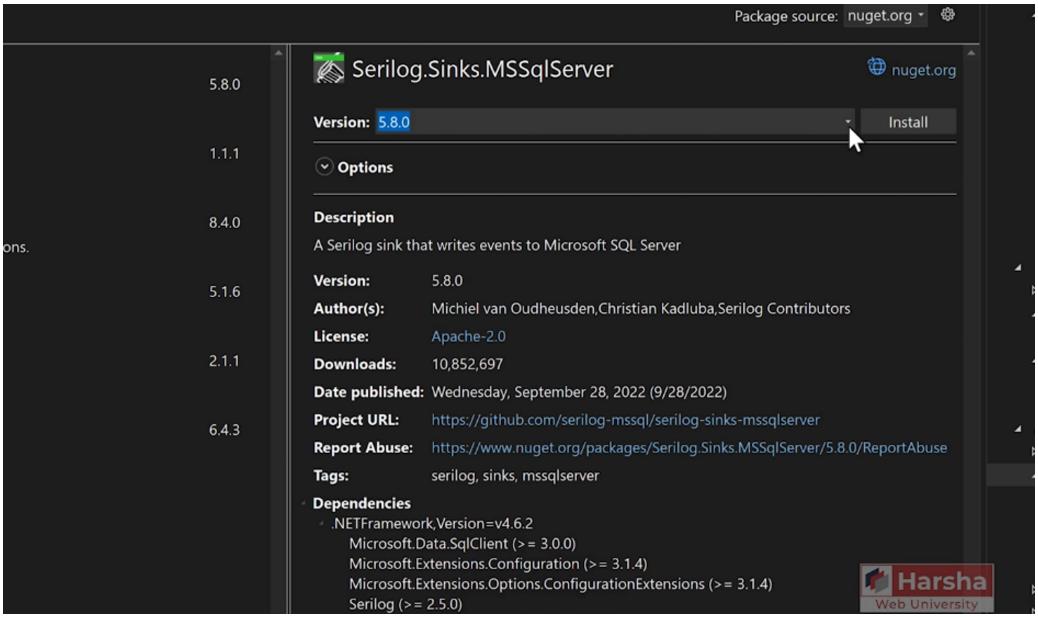

NuGet Package Manager: ContactsManager.UI
Package source: nuget.org
1.2.0
1.0.4
4.1.0
r .NETStandard 2.0.
1.2.1
Rotativa.AspNetCore
nuget.org
Version: 1.2.0
Install
Options
Description
Rotativa: PDF easy creation for Asp.Net Core
Version: 1.2.0
Author(s): Giorgio Bozio
License: MIT
Downloads: 754,722
Date published: Sunday, September 19, 2021 (9/19/2021)
Project URL: https://github.com/webgio/Rotativa.AspNetCore
Report Abuse: https://www.nuget.org/packages/Rotativa.AspNetCore/1.2.0/ReportAbuse
Tags: PDF, AspNetCore
Dependencies
.NETStandard,Version=v2.0
Microsoft.AspNetCore.Mvc (>= 2.0.1)

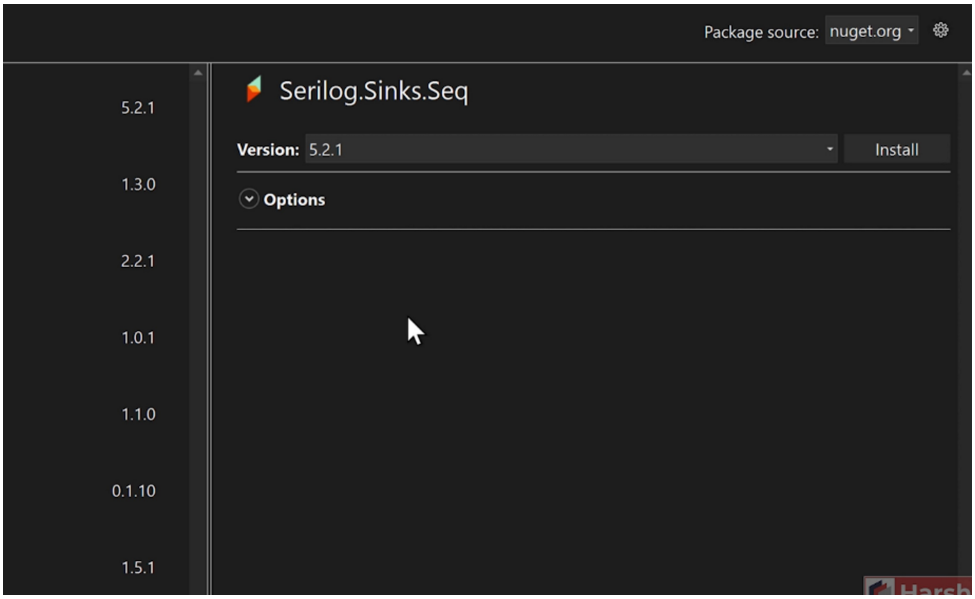

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
ContactsManagerSolution
NuGet: C...anager.UI
Program.cs
Browse
Installed
Updates
NuGet Package Manager: ContactsManager.UI
EntityFrameworkCore Tools
Include prerelease
Package source: nuget.org
Microsoft.EntityFrameworkCore.Tools by Microsoft, 170M downloads
Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.
6.0.9
Microsoft.EntityFrameworkCore.Design by Microsoft, 239M downloads
Shared design-time components for Entity Framework Core tools.
6.0.9
Microsoft.EntityFrameworkCore.Tools.DotNet by Microsoft, 8.94M downloads
This package version is deprecated.
2.0.3
dotnet-ef by Microsoft, 15.2M downloads
Entity Framework Core Tools for the .NET Command-Line Interface.
6.0.9
EntityFrameworkCore.CommonTools by Dmitry Panyushkin, 232K downloads
An extension for EntityFrameworkCore that provides Auditing, Concurrency Checks, JSON Complex Types and writing history to Transaction Log.
2.0.2
Ark.Tools.EntityFrameworkCore by ARK Labs, 24.8K downloads
Generic tools for EntityFrameworkCore
2.6.0
EfCore.TestSupport by Jon P Smith, 657K downloads
Useful tools when unit testing applications that use Entity Framework Core. See readme file on github.
5.2.2
Elsa.Persistence.EntityFrameworkCore by Elsa Contributors, 54.7K downloads
Elsa is a set of workflow libraries and tools that enable super-fast workflowing capabilities in any .NET Core application.
1.5.5.1943
Microsoft.EntityFrameworkCore.Tools
nuget.org
Version: Latest stable 6.0.9
Install
Options
Description
Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.
Enables these commonly used commands:
Add-Migration
Bundle-Migration
Drop-Database
Get-DbContext
Get-Migration
Optimize-DbContext
Remove-Migration
Scaffold-DbContext
Script-Migration
Update-Database
Version: 6.0.9
Author(s): Microsoft
License: MIT
Date published: Tuesday, September 13, 2022 (9/13/2022)

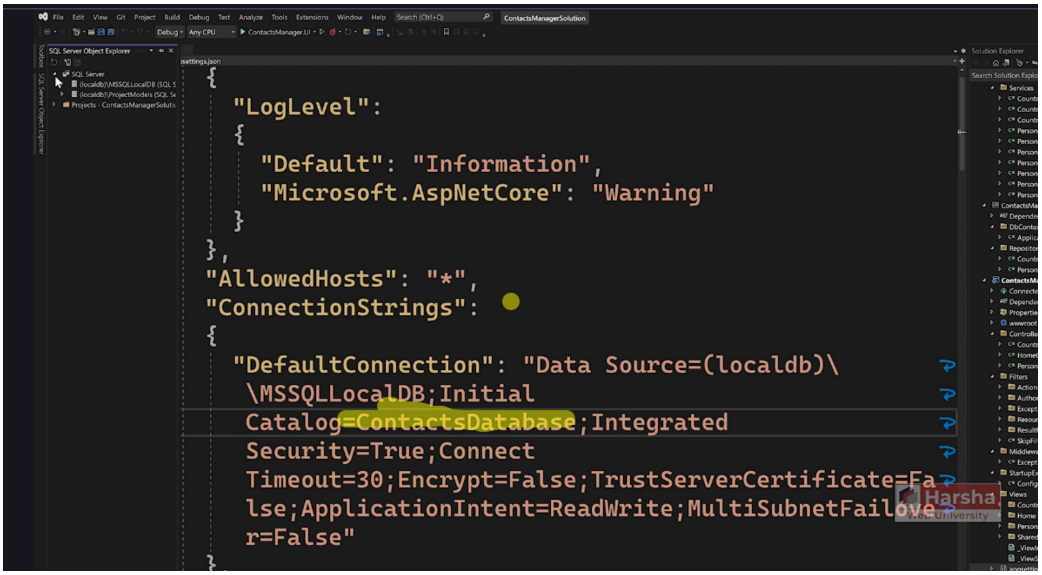

Package source: nuget.org
Serilog
Version: 2.12.0
Install
Options
2.12.0
5.0.0
3.1.0
4.1.0
1.1.0
3.4.0
6.0.1
2.0.0


6.0.1
5.0.1
3.0.0
1.0.0
1.0.1
Serilog.AspNetCore
nuget.org
Version: Latest stable 6.0.1
Install
Options
Description
Serilog support for ASP.NET Core logging
Version: 6.0.1
Author(s): Microsoft,Serilog Contributors
License: Apache-2.0
Date published: Tuesday, July 19, 2022 (7/19/2022)
Project URL: https://github.com/serilog/serilog-aspnetcore
Report Abuse: https://www.nuget.org/packages/Serilog.AspNetCore/6.0.1/ReportAbuse

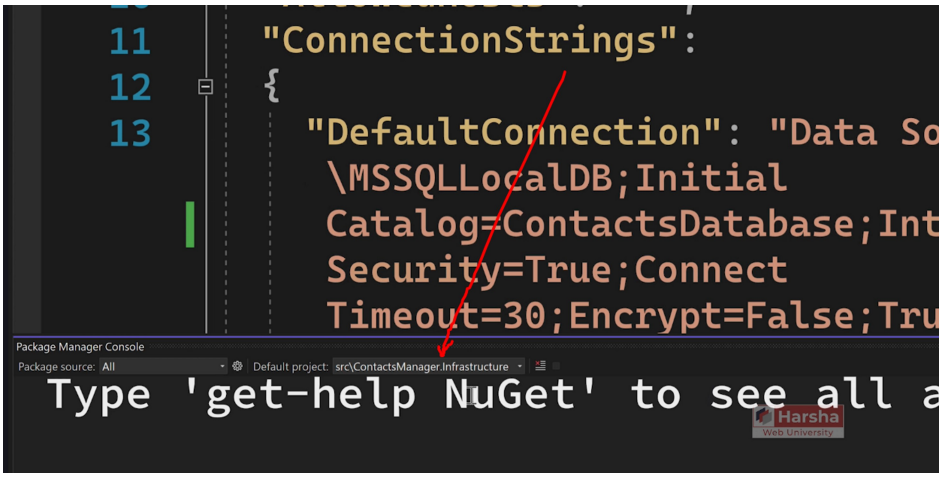

Package source: nuget.org
5.8.0
1.1.1
8.4.0
5.1.6
2.1.1
6.4.3
Serilog.Sinks.MSSqlServer
nuget.org
Version: 5.8.0
Install
Options
Description
A Serilog sink that writes events to Microsoft SQL Server
Version: 5.8.0
Author(s): Michiel van Oudheusden,Christian Kadluba,Serilog Contributors
License: Apache-2.0
Downloads: 10,852,697
Date published: Wednesday, September 28, 2022 (9/28/2022)
Project URL: https://github.com/serilog-mssql/serilog-sinks-mssqlserver
Report Abuse: https://www.nuget.org/packages/Serilog.Sinks.MSSqlServer/5.8.0/ReportAbuse
Tags: serilog, sinks, mssqlserver
Dependencies
.NETFramework,Version=v4.6.2
Microsoft.Data.SqlClient (>= 3.0.0)
Microsoft.Extensions.Configuration (>= 3.1.4)


Package source: nuget.org
5.8.0
1.1.1
8.4.0
5.1.6
2.1.1
6.4.3
Serilog.Sinks.MSSqlServer
nuget.org
Version: 5.8.0
Install
Options
Description
A Serilog sink that writes events to Microsoft SQL Server
Version: 5.8.0
Author(s): Michiel van Oudheusden,Christian Kadluba,Serilog Contributors
License: Apache-2.0
Downloads: 10,852,697
Date published: Wednesday, September 28, 2022 (9/28/2022)
Project URL: https://github.com/serilog-mssql/serilog-sinks-mssqlserver
Report Abuse: https://www.nuget.org/packages/Serilog.Sinks.MSSqlServer/5.8.0/ReportAbuse
Tags: serilog, sinks, mssqlserver
Dependencies
.NETFramework,Version=v4.6.2
Microsoft.Data.SqlClient (>= 3.0.0)
Microsoft.Extensions.Configuration (>= 3.1.4)
Microsoft.Extensions.Options.ConfigurationExtensions (>= 3.1.4)
Serilog (>= 2.5.0)
Harsha
Web University



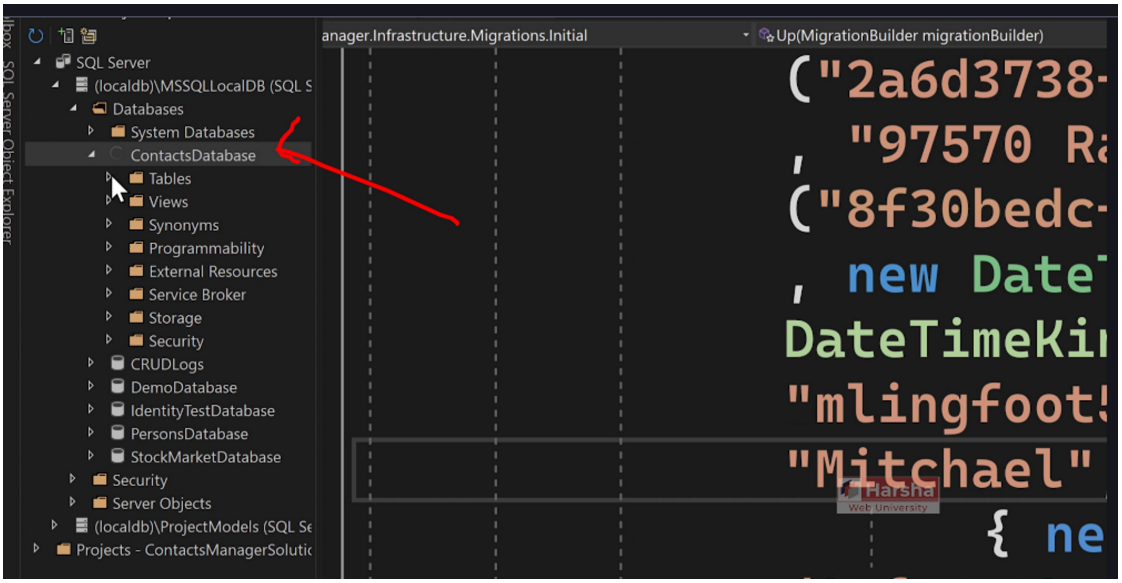


Select Infrastructure project because in that project, our 'ApplicationDbContext' class is present.



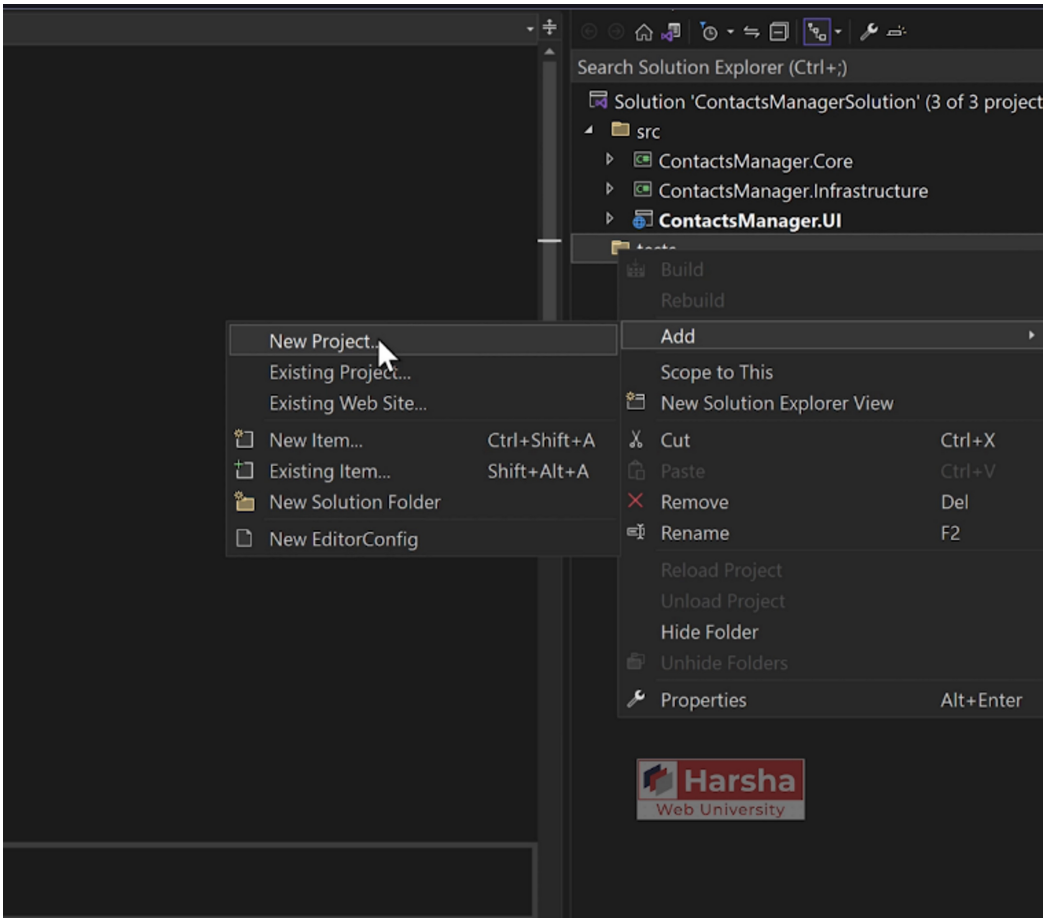

kage Manager Console
kage source: All
Default project: src\ContactsManager.Infrastructure
PM> Add-Migration Initial
Harsha
Web University

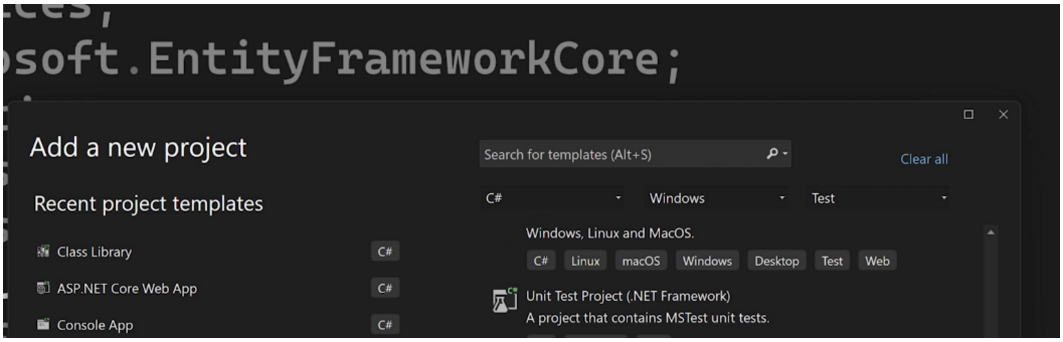

kage Manager Console
ckage source: All
Default project: src\ContactsManager.Infrastructure
PM> Update-Database
Build started...
Harsha
Web University

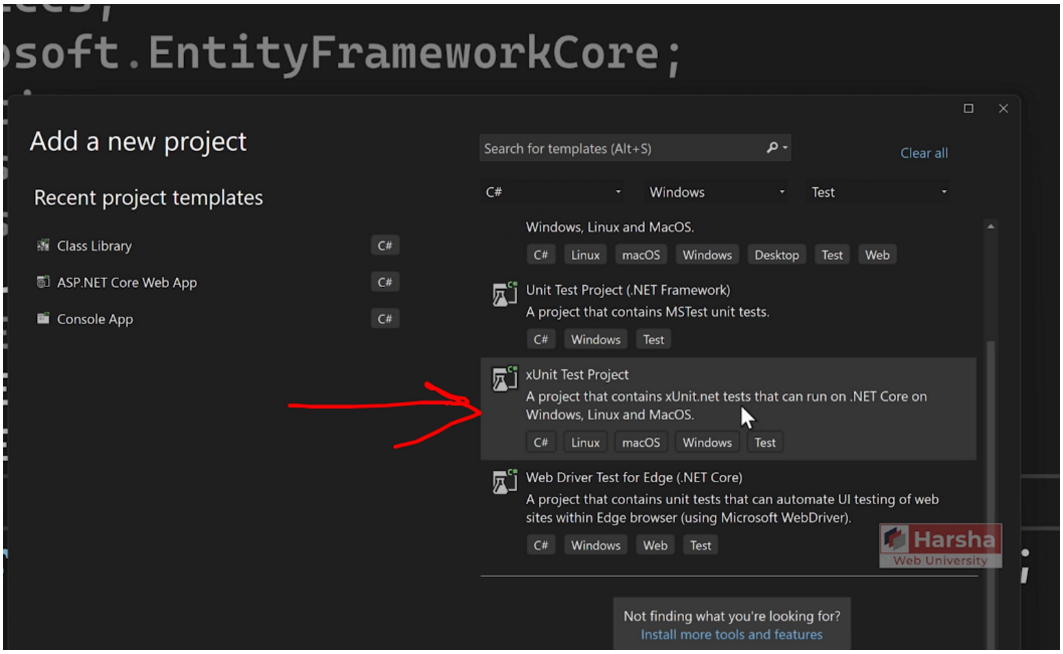

SQL Server
(localdb)\MSSQLLocalDB (SQL S
Databases
System Databases
ContactsDatabase
Tables
Views
Synonyms
Programmability
External Resources
Service Broker
Storage
Security
CRUDLogs
DemoDatabase
IdentityTestDatabase
PersonsDatabase
StockMarketDatabase
Security
Server Objects
(localdb)\ProjectModels (SQL Se
Projects - ContactsManagerSolutic
anager.Infrastructure.Migrations.Initial
Up(MigrationBuilder migrationBuilder)
("2a6d3738-
, "97570 Ra
("8f30bedc-
, new Date
DateTimeKi
"mlingfoot
"Mitchael"
{ ne
Harsha
Web University

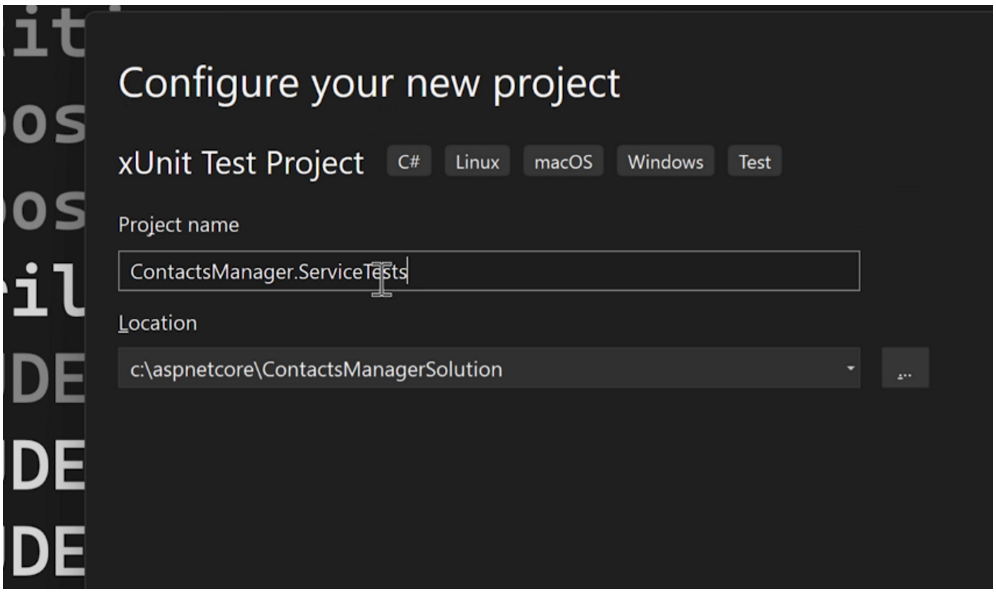
Asp.Net Core

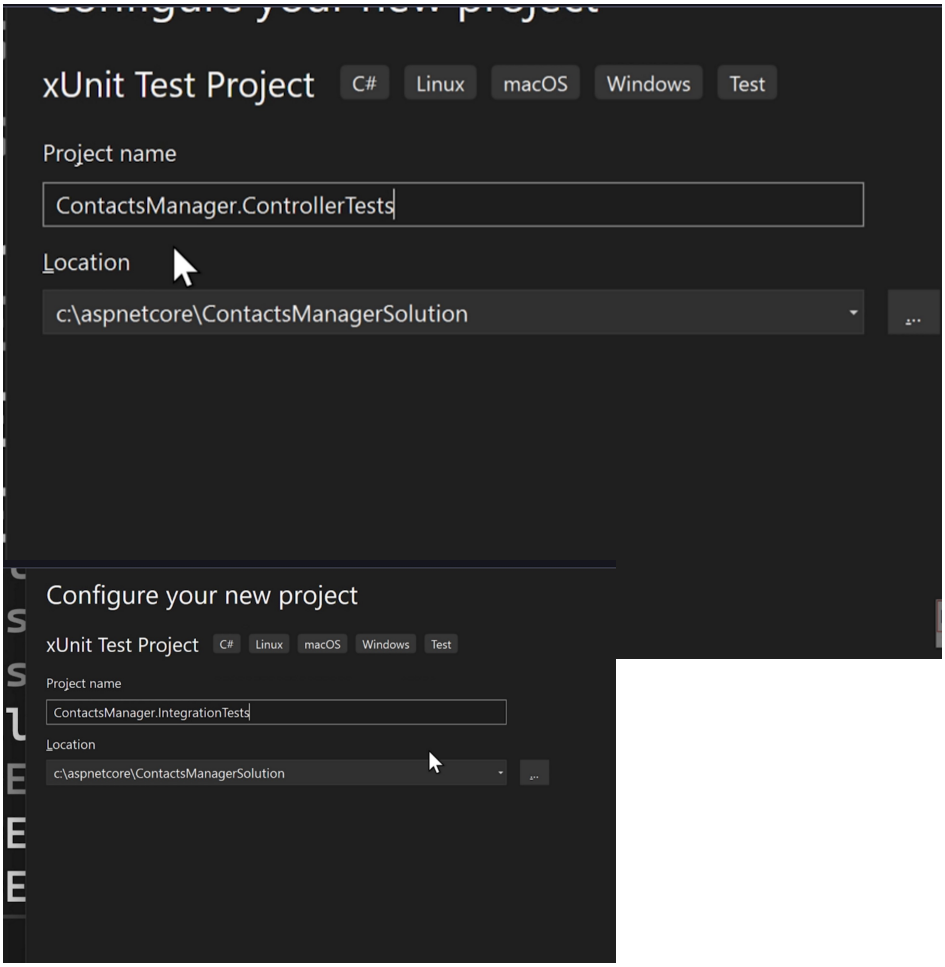
Clean Architecture – Tests



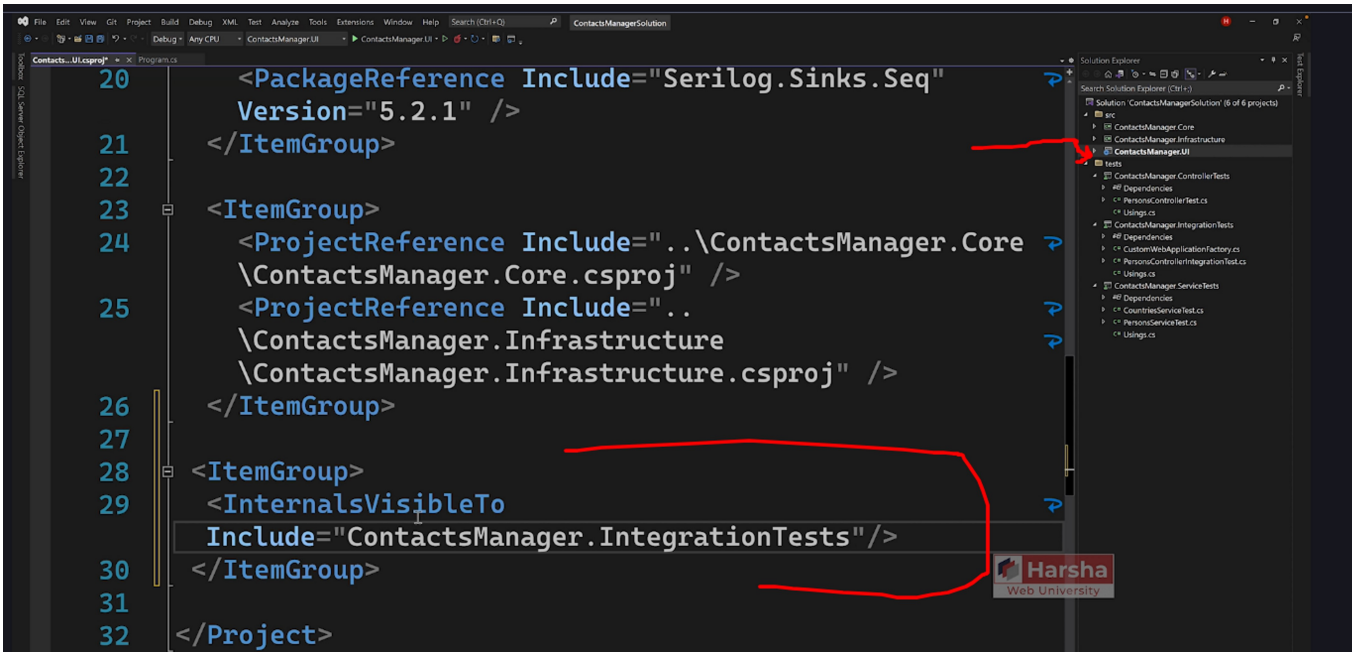



soft.EntityFrameworkCore;
Add a new project
Recent project templates
Class Library
C#
ASP.NET Core Web App
C#
Console App
C#
Search for templates (Alt+S)
Clear all
C#
Windows
Test
Windows, Linux and MacOS.
C# Linux macOS Windows Desktop Test Web
Unit Test Project (.NET Framework)
A project that contains MSTest unit tests.
C# Windows Test
xUnit Test Project
A project that contains xUnit.net tests that can run on .NET Core on Windows, Linux and MacOS.
C# Linux macOS Windows Test
Web Driver Test for Edge (.NET Core)
A project that contains unit tests that can automate UI testing of web sites within Edge browser (using Microsoft WebDriver).
C# Windows Web Test
Harsha
Web University
Not finding what you're looking for?
Install more tools and features

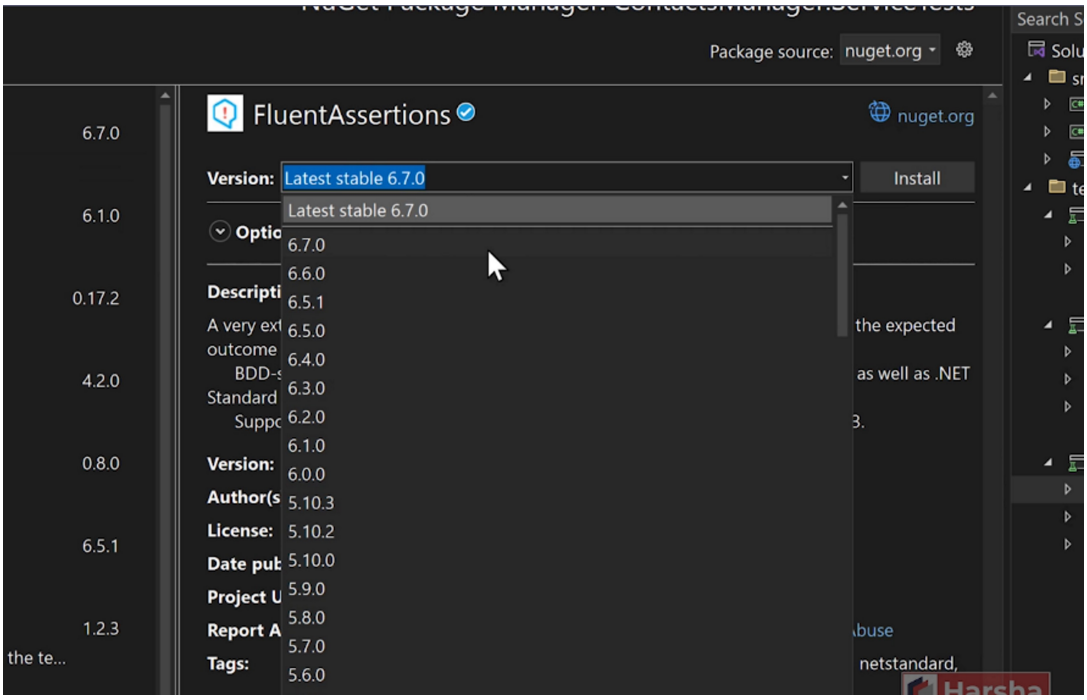

Configure your new project
xUnit Test Project
C# Linux macOS Windows Test
Project name
ContactsManager.ServiceTests
Location
c:\aspnetcore\ContactsManagerSolution
...

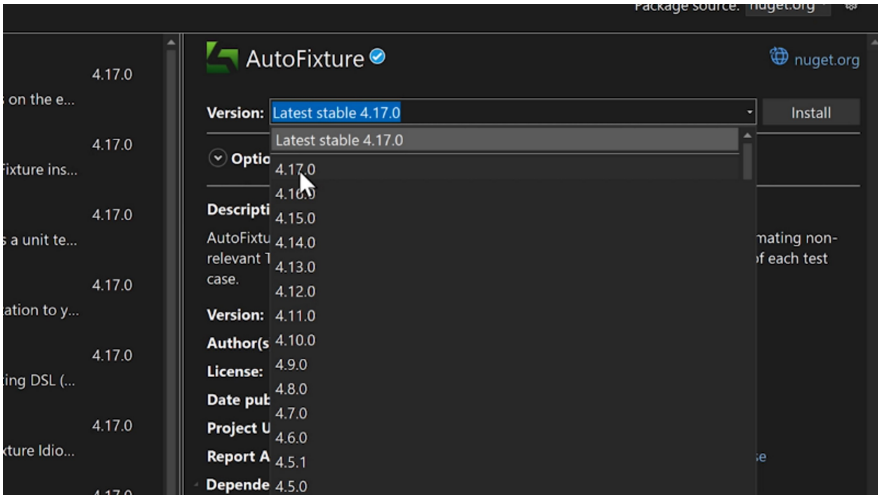


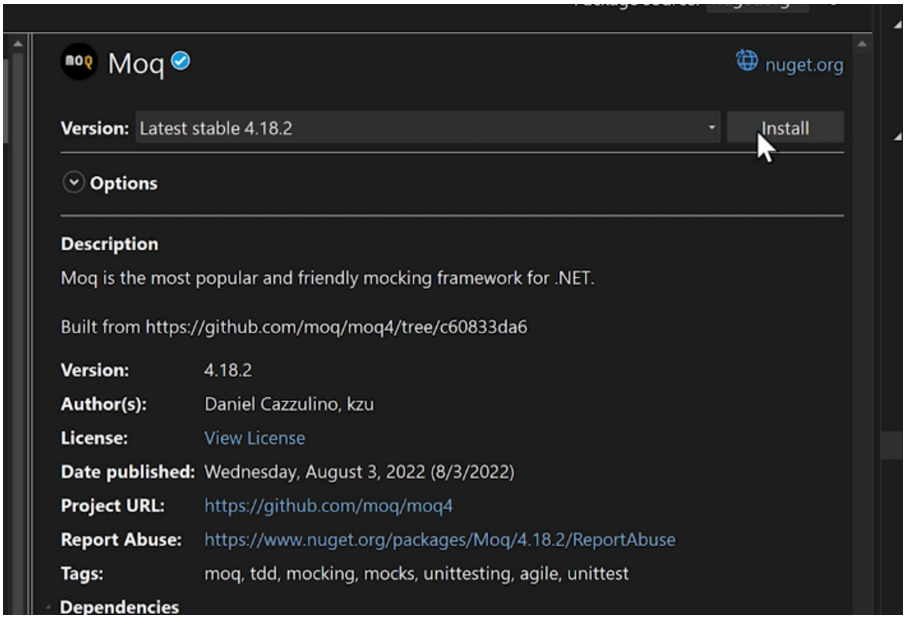
Now, our integration test project can access the program class of the UI project.



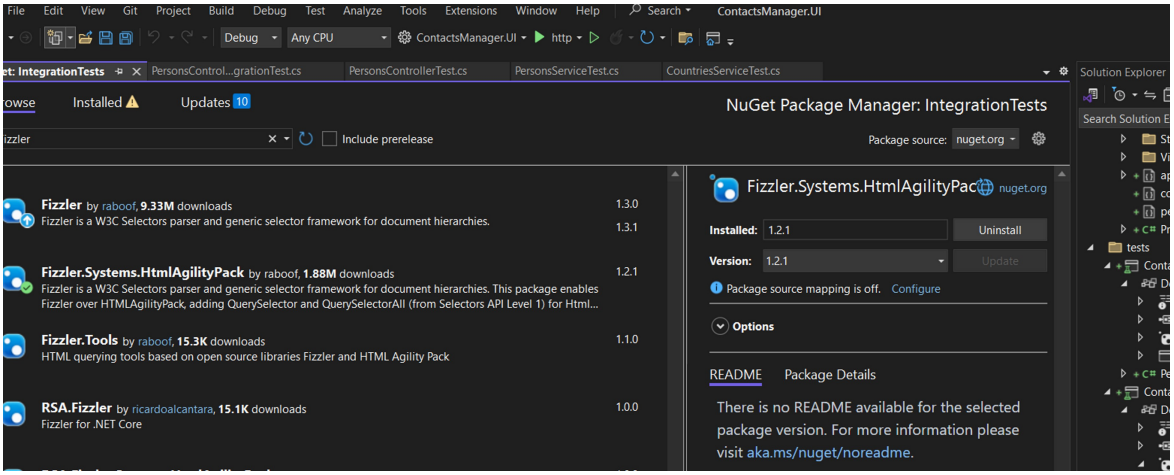

racts;
lters.ActionFilters;
ddleware;
Dependencies
PersonsControllerTest.cs
Usings.cs
ContactsManager.IntegrationTests
Dependencies
CustomWebApplicationFactory.cs
PersonsControllerIntegrationTest.cs
Usings.cs
ContactsManager.ServiceTests
Add Project Reference...
Add Shared Project Reference...
Add COM Reference...
Manage NuGet Packages...
Remove Unused References...
Scope to This
New Solution Explorer View
1
2
Harsha

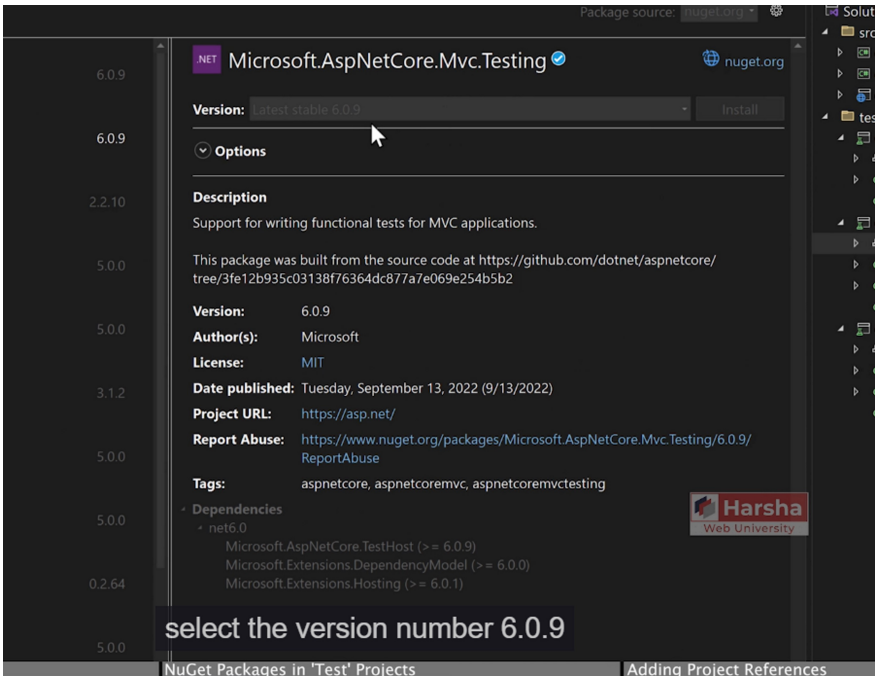

Package source: nuget.org
FluentAssertions
nuget.org
Version: Latest stable 6.7.0
Install
Latest stable 6.7.0
6.7.0
6.6.0
6.5.1
6.5.0
6.4.0
6.3.0
6.2.0
6.1.0
6.0.0
5.10.3
5.10.2
5.10.0
5.9.0
5.8.0
5.7.0
5.6.0
Description
Version:
Author(s
License:
Date pub
Project U
Report A
Tags:
netstandard,
Harsha

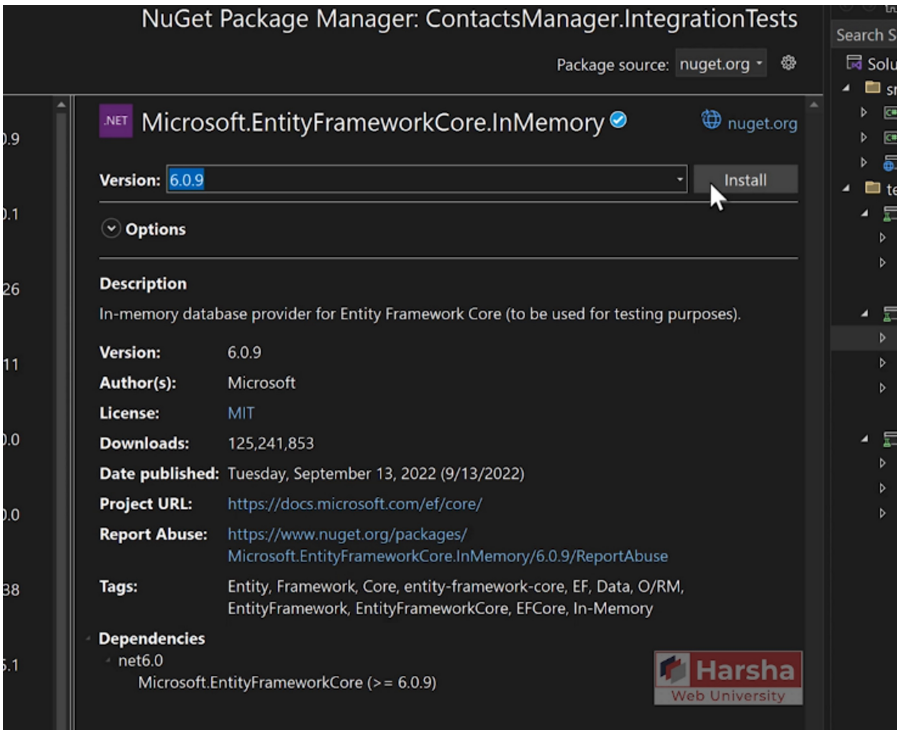

AutoFixture
nuget.org
Version: Latest stable 4.17.0
Install
Latest stable 4.17.0
4.17.0
4.16.0
4.15.0
4.14.0
4.13.0
4.12.0
4.11.0
4.10.0
4.9.0
4.8.0
4.7.0
4.6.0
4.5.1
4.5.0
Description
Version:
Author(s
License:
Date pub
Project U
Report A
Depende

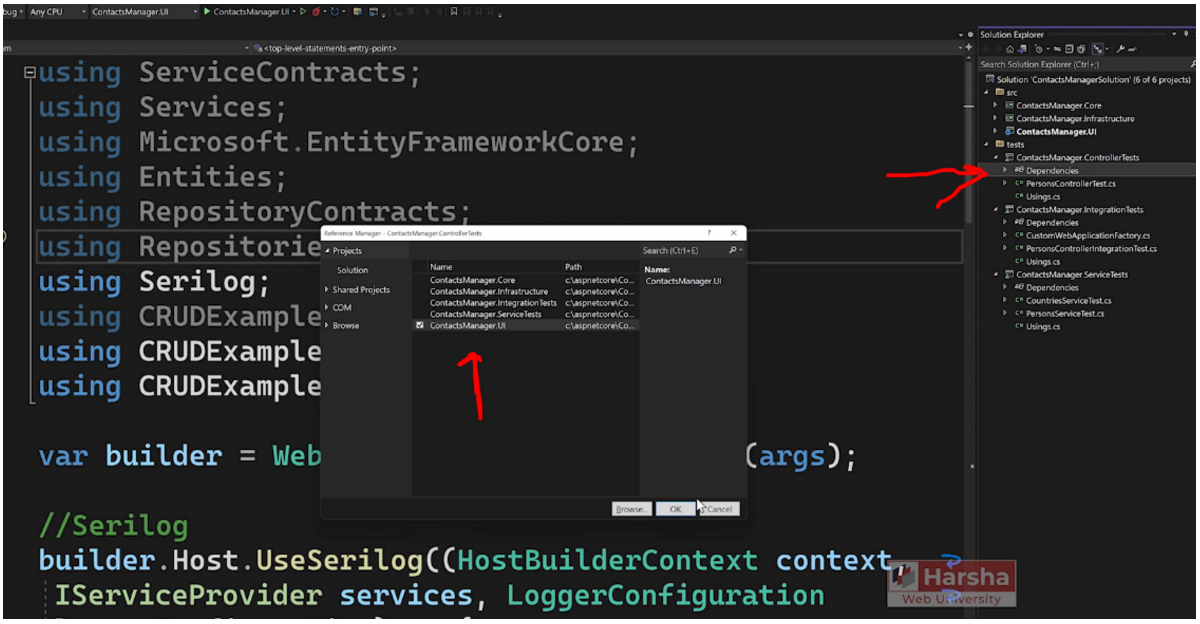


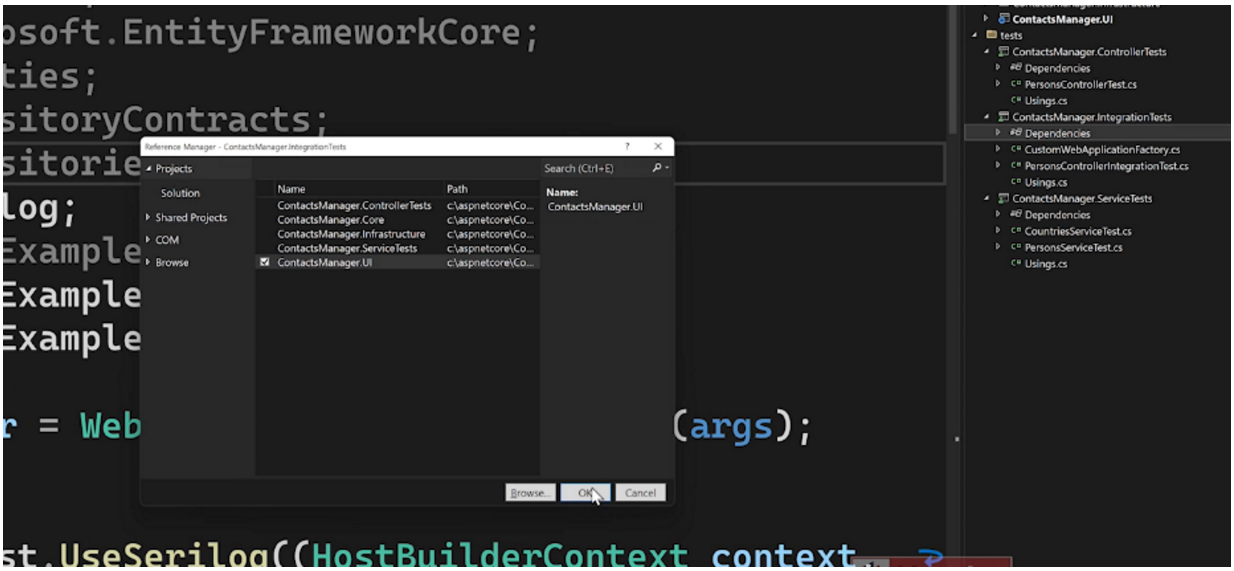
For Integration test, install the same packages. Additionally we need to install this following package.

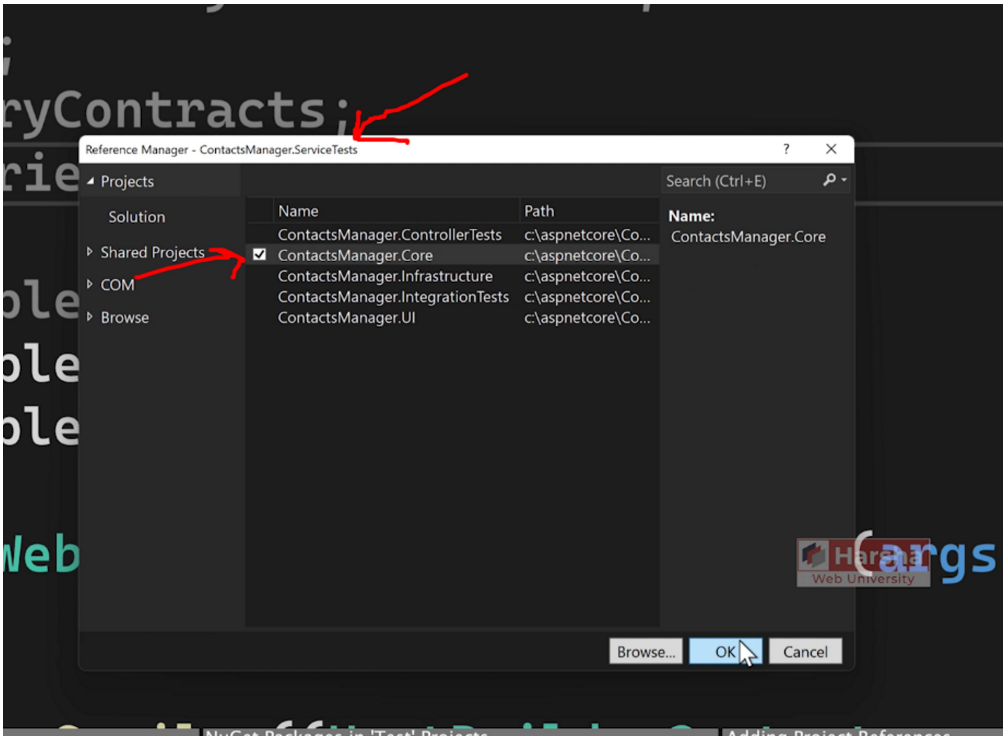


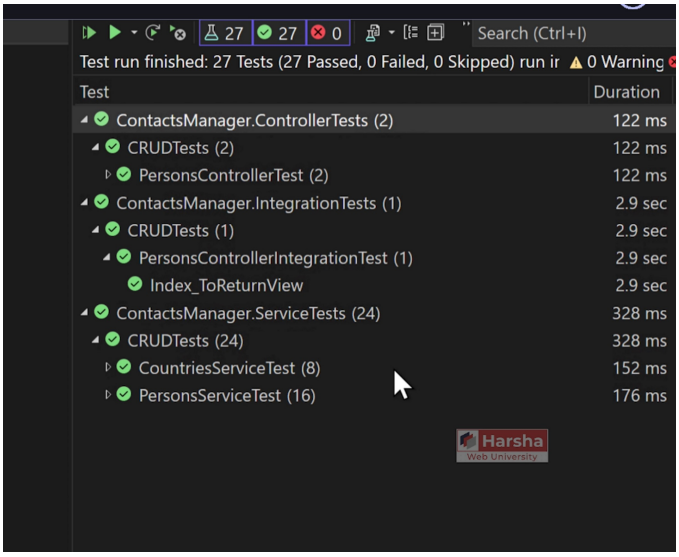

Package source: nuget.org
Microsoft.AspNetCore.Mvc.Testing
nuget.org
Version: Latest stable 6.0.9
Install
Options
Description
Support for writing functional tests for MVC applications.
This package was built from the source code at https://github.com/dotnet/aspnetcore/tree/3fe12b935c03138f76364dc877a7e069e254b5b2
Version: 6.0.9
Author(s): Microsoft
License: MIT
Date published: Tuesday, September 13, 2022 (9/13/2022)
Project URL: https://asp.net/
Report Abuse: https://www.nuget.org/packages/Microsoft.AspNetCore.Mvc.Testing/6.0.9/ReportAbuse
Tags: aspnetcore, aspnetcoremvc, aspnetcoremvctesting
Dependencies
net6.0
Microsoft.AspNetCore.TestHost (>= 6.0.9)
Microsoft.Extensions.DependencyModel (>= 6.0.0)
Microsoft.Extensions.Hosting (>= 6.0.1)
Harsha
Web University
select the version number 6.0.9
NuGet Packages in 'Test' Projects
Adding Project References


NuGet Package Manager: ContactsManager.IntegrationTests
Package source: nuget.org
Microsoft.EntityFrameworkCore.InMemory
nuget.org
Version: 6.0.9
Install
Options
Description
In-memory database provider for Entity Framework Core (to be used for testing purposes).
Version: 6.0.9
Author(s): Microsoft
License: MIT
Downloads: 125,241,853
Date published: Tuesday, September 13, 2022 (9/13/2022)
Project URL: https://docs.microsoft.com/ef/core/
Report Abuse: https://www.nuget.org/packages/Microsoft.EntityFrameworkCore.InMemory/6.0.9/ReportAbuse
Tags: Entity, Framework, Core, entity-framework-core, EF, Data, O/RM, EntityFramework, EntityFrameworkCore, EFCore, In-Memory
Dependencies
net6.0
Microsoft.EntityFrameworkCore (>= 6.0.9)
Harsha
Web University


using ServiceContracts;
using Services;
using Microsoft.EntityFrameworkCore;
using Entities;
using RepositoryContracts;
using Repositorie
using Serilog;
using CRUDExample
using CRUDExample
using CRUDExample
var builder = Web
(args);
//Serilog
builder.Host.UseSerilog((HostBuilderContext context
IServiceProvider services, LoggerConfiguration
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'ContactsManagerSolution' (6 of 6 projects)
src
ContactsManager.Core
ContactsManager.Infrastructure
ContactsManager.UI
tests
ContactsManager.ControllerTests
Dependencies
PersonsControllerTest.cs
Usings.cs
ContactsManager.IntegrationTests
Dependencies
CustomWebApplicationFactory.cs
PersonsControllerIntegrationTest.cs
Usings.cs
ContactsManager.ServiceTests
Dependencies
CountriesServiceTest.cs
PersonsServiceTest.cs
Usings.cs
Reference Manager - ContactsManager.ControllerTests
Projects
Solution
Shared Projects
COM
Browse
Search (Ctrl+E)
Name
Path
ContactsManager.Core
ContactsManager.Infrastructure
ContactsManager.IntegrationTests
ContactsManager.ServiceTests
ContactsManager.UI
Name:
ContactsManager.UI
Browse...
OK
Cancel
Harsha
Web University


osoft.EntityFrameworkCore;
ties;
sitoryContracts;
sitorie
log;
Example
Example
Example
r = Web
(args);
st.UseSerilog((HostBuilderContext context
Reference Manager - ContactsManager.IntegrationTests
Projects
Solution
Shared Projects
COM
Browse
Search (Ctrl+E)
Name
Path
ContactsManager.ControllerTests
ContactsManager.Core
ContactsManager.Infrastructure
ContactsManager.IntegrationTests
ContactsManager.ServiceTests
ContactsManager.UI
Name:
ContactsManager.UI
Browse...
OK
Cancel
ContactsManager.UI
tests
ContactsManager.ControllerTests
Dependencies
PersonsControllerTest.cs
Usings.cs
ContactsManager.IntegrationTests
Dependencies
CustomWebApplicationFactory.cs
PersonsControllerIntegrationTest.cs
Usings.cs
ContactsManager.ServiceTests
Dependencies
CountriesServiceTest.cs
PersonsServiceTest.cs
Usings.cs





For further reading about clean architecture, we can suggest this documentation from Microsoft Learn: https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures

Common web application architectures

Article • 09/21/2022 • 20 minutes to read • 15 contributors

Tip

This content is an excerpt from the eBook, Architect Modern Web Applications with ASP.NET Core and Azure, available on **.NET Docs** or as a free downloadable PDF that can be read offline.

Download PDF

> "If you think good architecture is expensive, try bad architecture." - *Brian Foote and Joseph Yoder*

Most traditional .NET applications are deployed as single units corresponding to an executable or a single web application running within a single IIS appdomain. This approach is the simplest deployment model and serves many internal and smaller public applications very well. However,