

Clean Architecture

Jason Taylor





**Click 'Rate Session'
to rate session
and ask questions.**



Clean Architecture with ASP.NET Core 3.0





Jason Taylor

SSW Solution Architect

Visiting from Australia

.NET Developer Since 2002

Keep It Simple, Stupid!

 [jasongtau](#)

 github.com/jasongt

 codingflow.net

 youtube.com/jasongt

Join the Conversation #DotNetCoreSuperpowers @SSW_TV

Agenda

Clean Architecture

Domain Layer

Application Layer

Infrastructure Layer

Presentation Layer

Next Steps

Clean Architecture

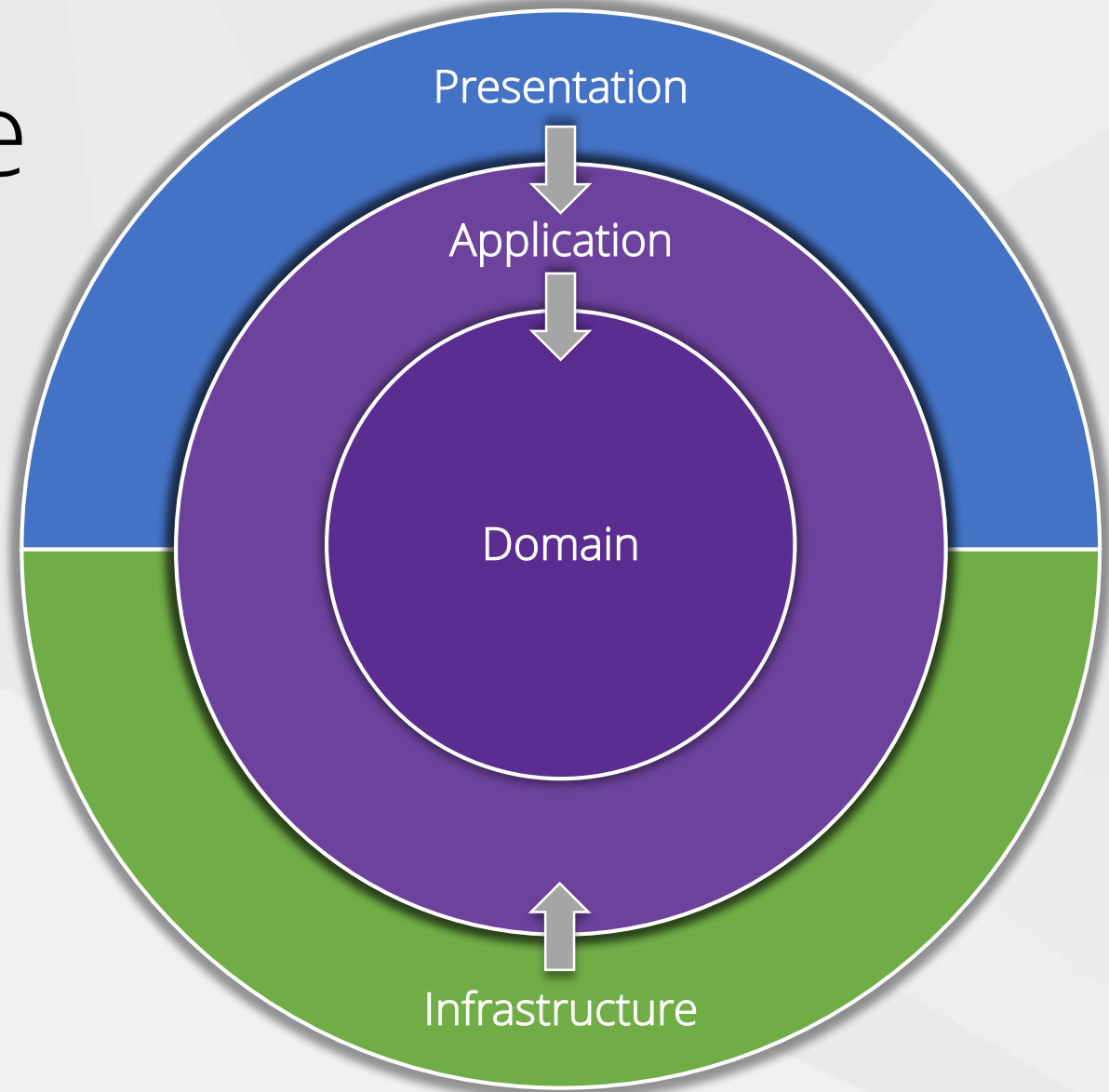
Independent of frameworks

Testable

Independent of UI

Independent of database

Independent anything external



Northwind Traders Sample

Clean Architecture

ASP.NET Core 3.0

Entity Framework Core 3.0

ASP.NET Core Identity 3.0

Repo bit.ly/northwind-traders



Clean Architecture Template



.NET Core Template Package

ASP.NET Core 3.0

Entity Framework Core 3.0

ASP.NET Core Identity 3.0

Repo bit.ly/ca-sln



Key Points

- ✓ Domain contains enterprise-wide logic and types
- ✓ Application contains business-logic and types
- ✓ Infrastructure contains all external concerns
- ✓ Presentation and Infrastructure depend only on Application
- ✓ Infrastructure and Presentation components can be replaced with minimal effort

Agenda

Clean Architecture

Domain Layer

Application Layer

Infrastructure Layer

Presentation Layer

Next Steps

Overview

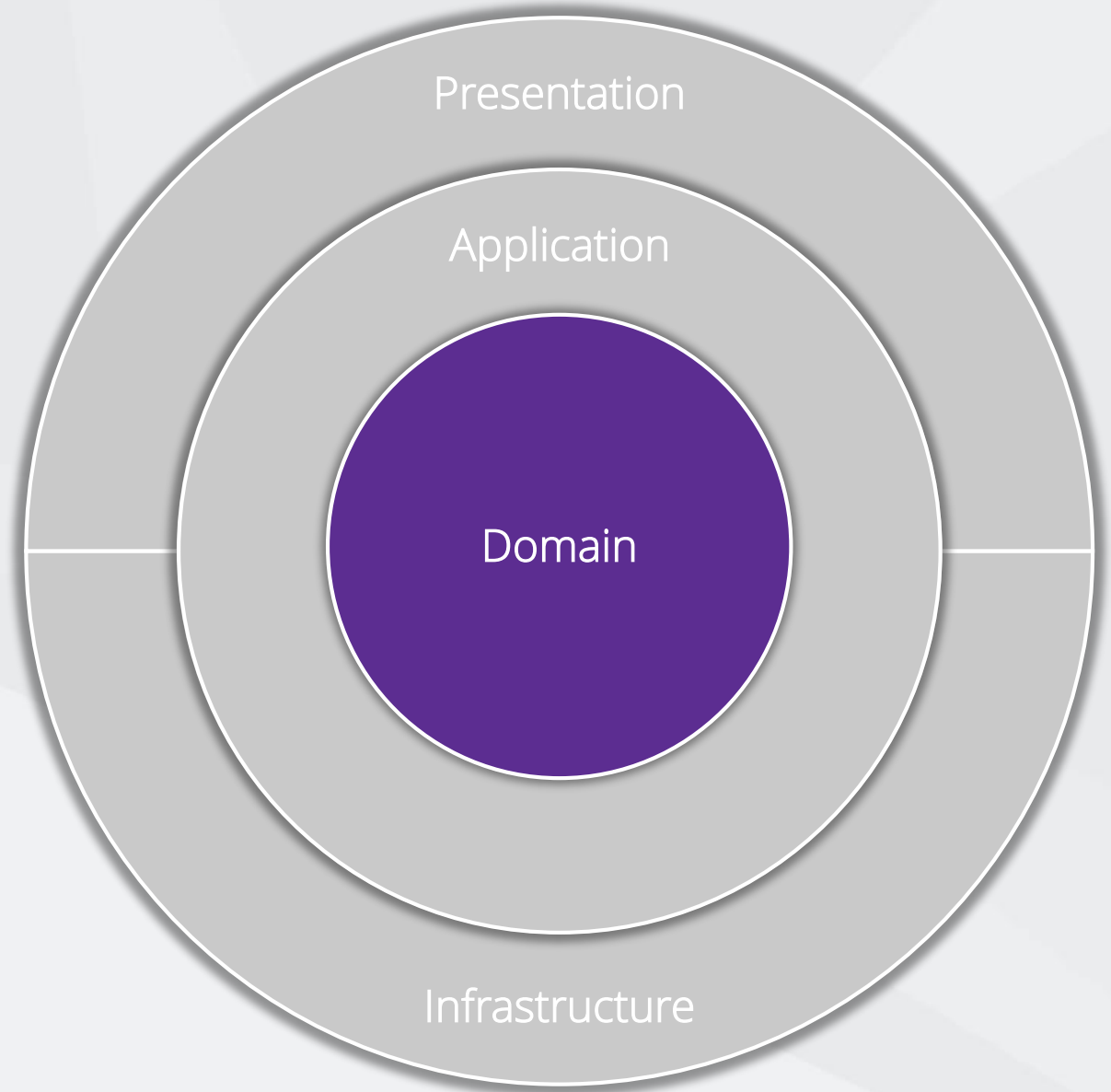
Entities

Value Objects

Enumerations

Logic

Exceptions



Demo

Reviewing the Domain layer



Key Points

- ✓ Avoid using data annotations
- ✓ Use value objects where appropriate
- ✓ Create custom domain exceptions
- ✓ Initialise all collections & use private setters
- ✓ Automatically track changes

Agenda

Clean Architecture

Domain Layer

Application Layer

Infrastructure Layer

Presentation Layer

Next Steps

Overview

Interfaces

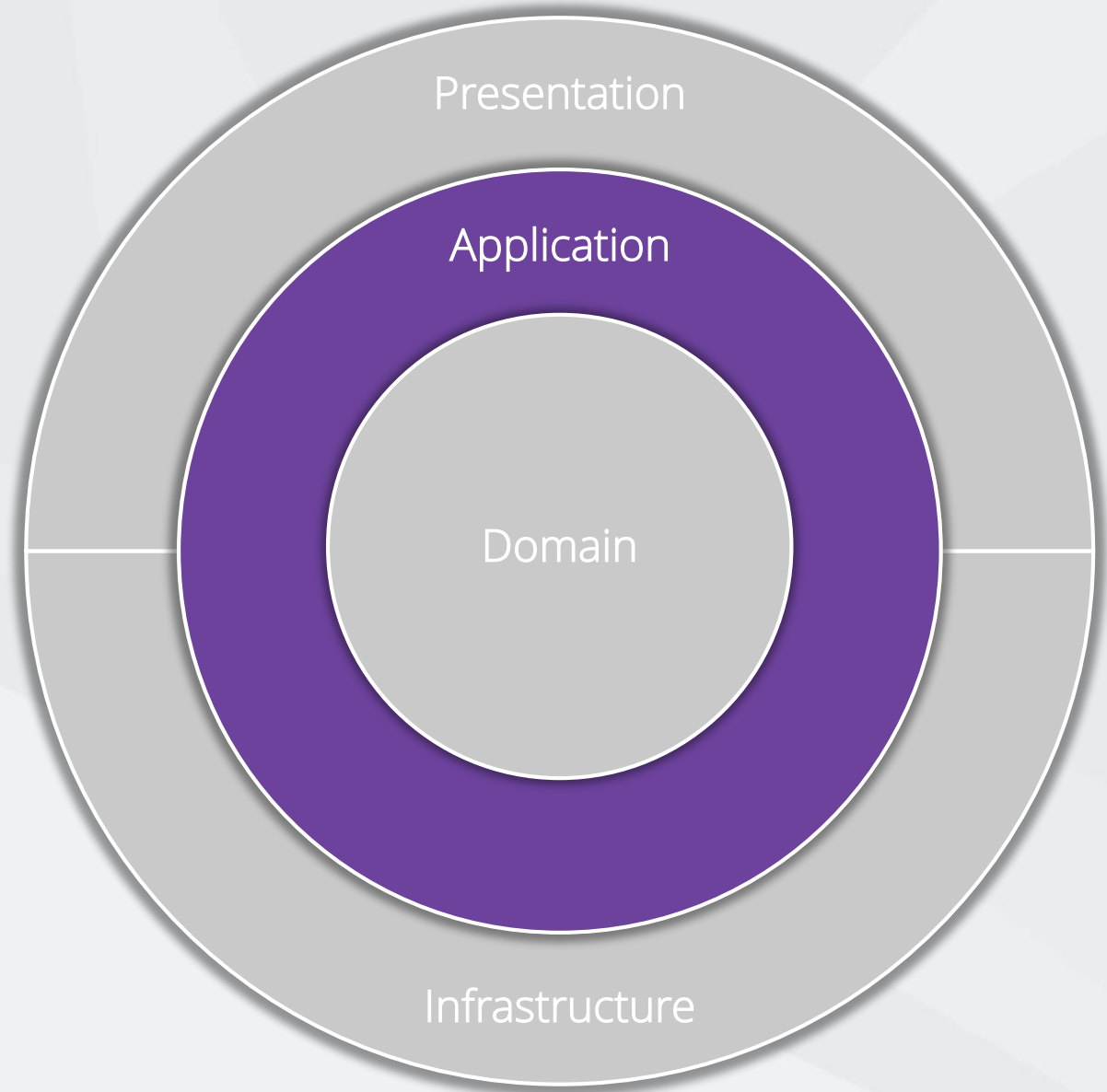
Models

Logic

Commands / Queries

Validators

Exceptions



CQRS

Command Query Responsibility Segregation

Separate reads (queries) from writes (commands)

Can maximise performance, scalability, and simplicity

Easy to add new features, just add a new query or command

Easy to maintain, changes only affect one command or query

CQRS + MediatR = ❤️

Define commands and queries as requests

Application layer is just a series of request / response objects

Ability to attach additional behaviour before and / or after each request, e.g. logging, validation, caching, authorisation and so on

Demo



Reviewing the Application layer

Key Points

- ✓ Using CQRS + MediatR simplifies your overall design
- ✓ MediatR simplifies cross cutting concerns
- ✓ Fluent Validation is useful for all validation scenarios
- ✓ AutoMapper simplifies mapping and projections
- ✓ Independent of infrastructure concerns

Agenda

Clean Architecture

Domain Layer

Application Layer

Infrastructure Layer

Presentation Layer

Next Steps

Overview

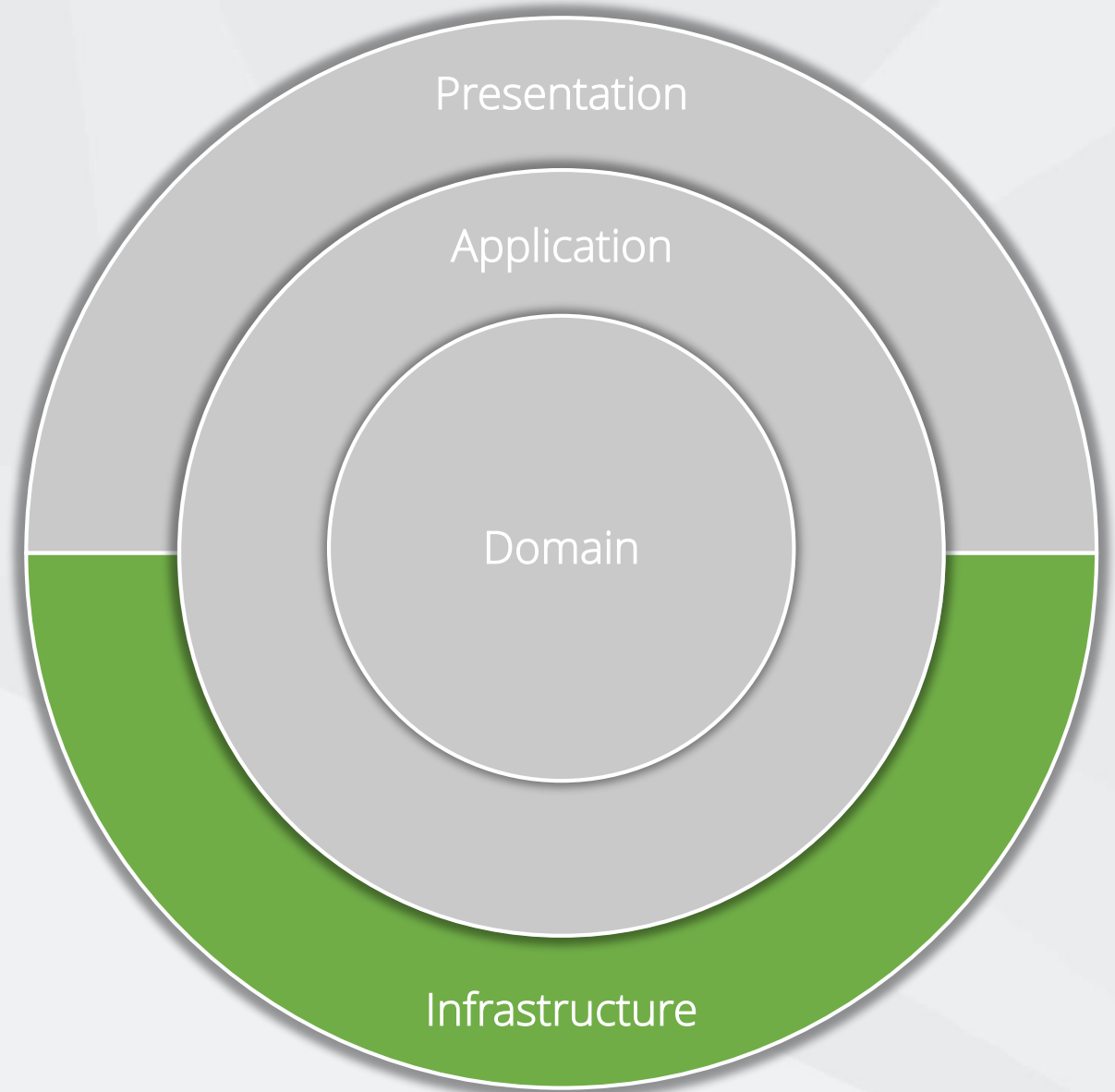
Persistence

Identity

File System

System Clock

API Clients



Unit of Work and Repository Patterns

Should we implement these patterns?



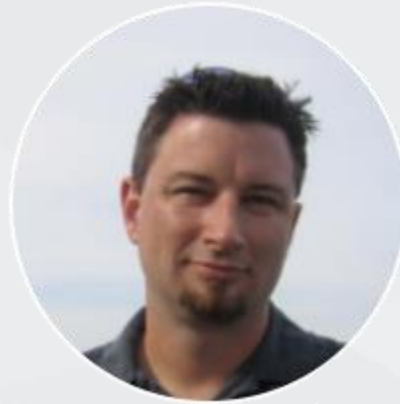
It isn't always the best choice, because:

- ✓ EF Core insulates your code from database changes
- ✓ DbContext acts as a unit of work
- ✓ DbSet acts as a repository
- ✓ EF Core has features for unit testing without repositories

What do the experts think?



I'm over Repositories, and definitely over abstracting your data layer.



No, you don't *need* a repository. But there are many benefits and you should consider it!



No, the repository/unit-of-work pattern isn't useful with EF Core.

Demo



Reviewing the Infrastructure layer

Join the Conversation #GOTOCph @JasonGtAu

Key Points

- ✓ Independent of the database
- ✓ Use Fluent API configuration over data annotations
- ✓ Prefer conventions over configuration
- ✓ Automatically apply all entity type configurations
- ✓ No layers depend on Infrastructure layer, e.g.

Presentation layer

Agenda

Clean Architecture

Domain Layer

Application Layer

Infrastructure Layer

Presentation Layer

Next Steps

Overview

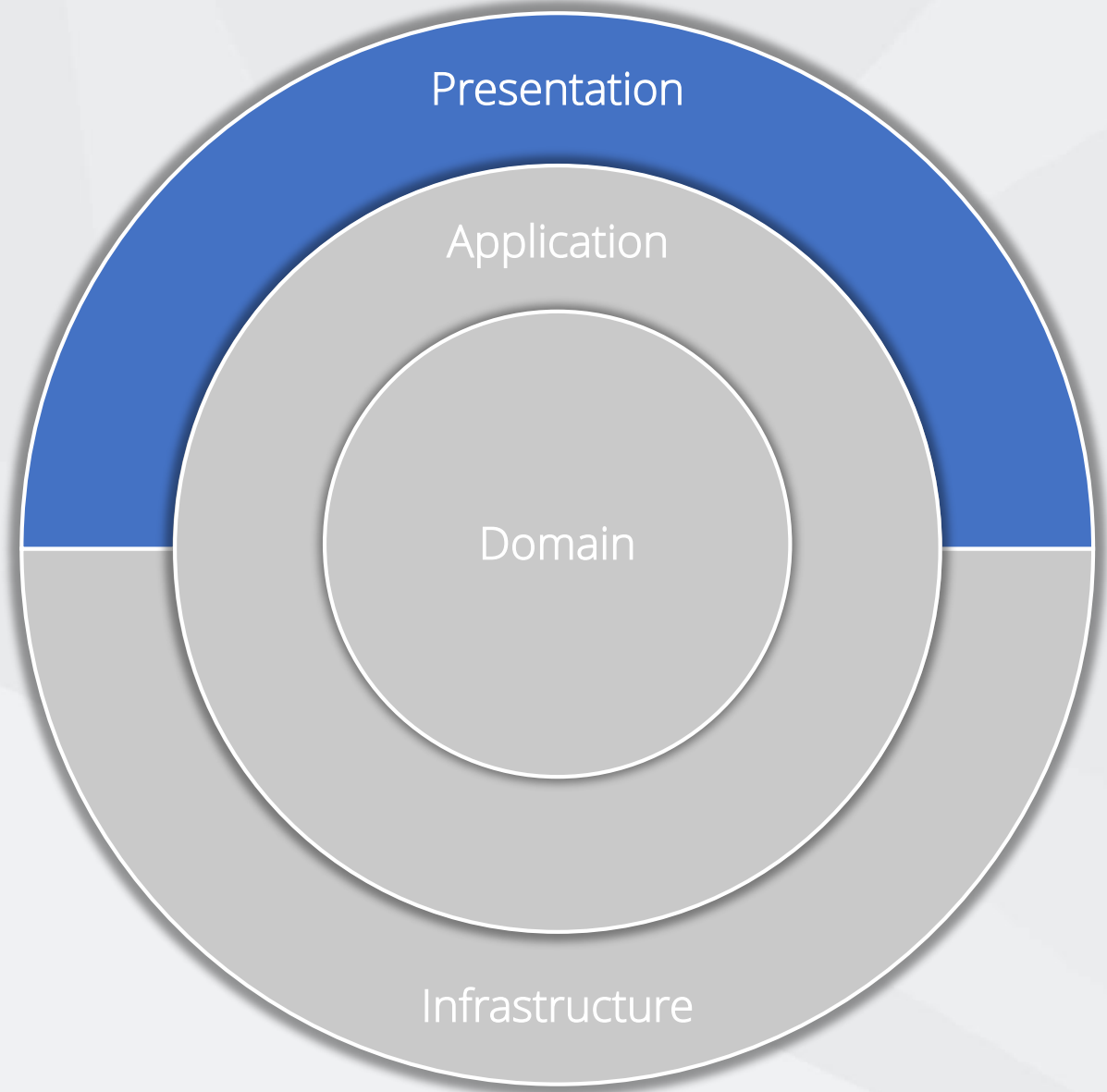
SPA – Angular, React, Vue

Web API

Razor Pages

MVC

Web Forms



Demo



Reviewing the Presentation layer

Key Points

- ✓ Controllers should not contain any application logic
- ✓ Create and consume well defined view models
- ✓ Open API bridges the gap between the front end and back end
- ✓ NSwag automates generation of Open API specification and clients

Agenda

Clean Architecture

Domain Layer

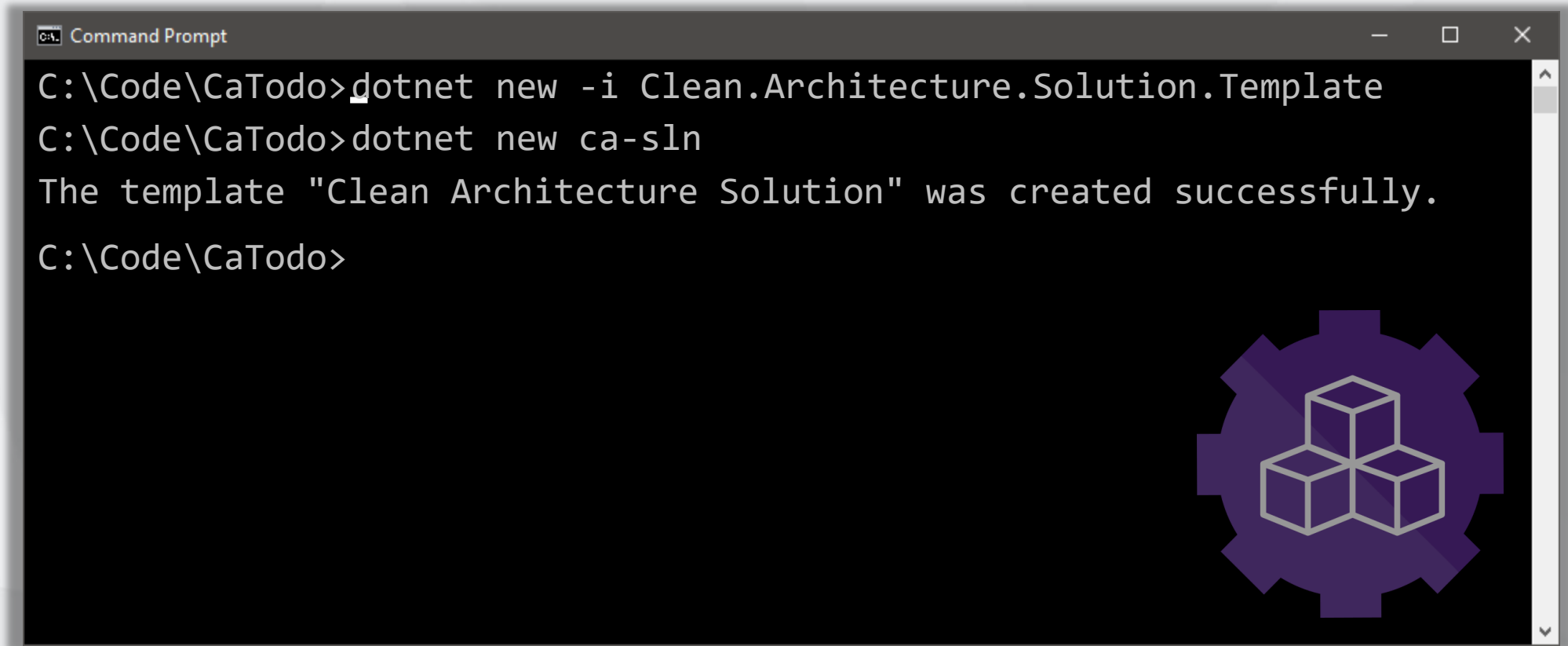
Application Layer

Infrastructure Layer


Presentation Layer

Next Steps

Using the Solution Template



```
C:\Code\CaTodo>dotnet new -i Clean.Architecture.Solution.Template
C:\Code\CaTodo>dotnet new ca-sln
The template "Clean Architecture Solution" was created successfully.
C:\Code\CaTodo>
```





Please

**Remember to
rate this session**

Thank you!



Did you **remember**
to rate the previous
session ?



Thank you!

 @jasongtau

bit.ly/ca-sln

bit.ly/northwind-traders

info@ssw.com.au

www.ssw.com.au

Sydney | Melbourne | Brisbane