Overview

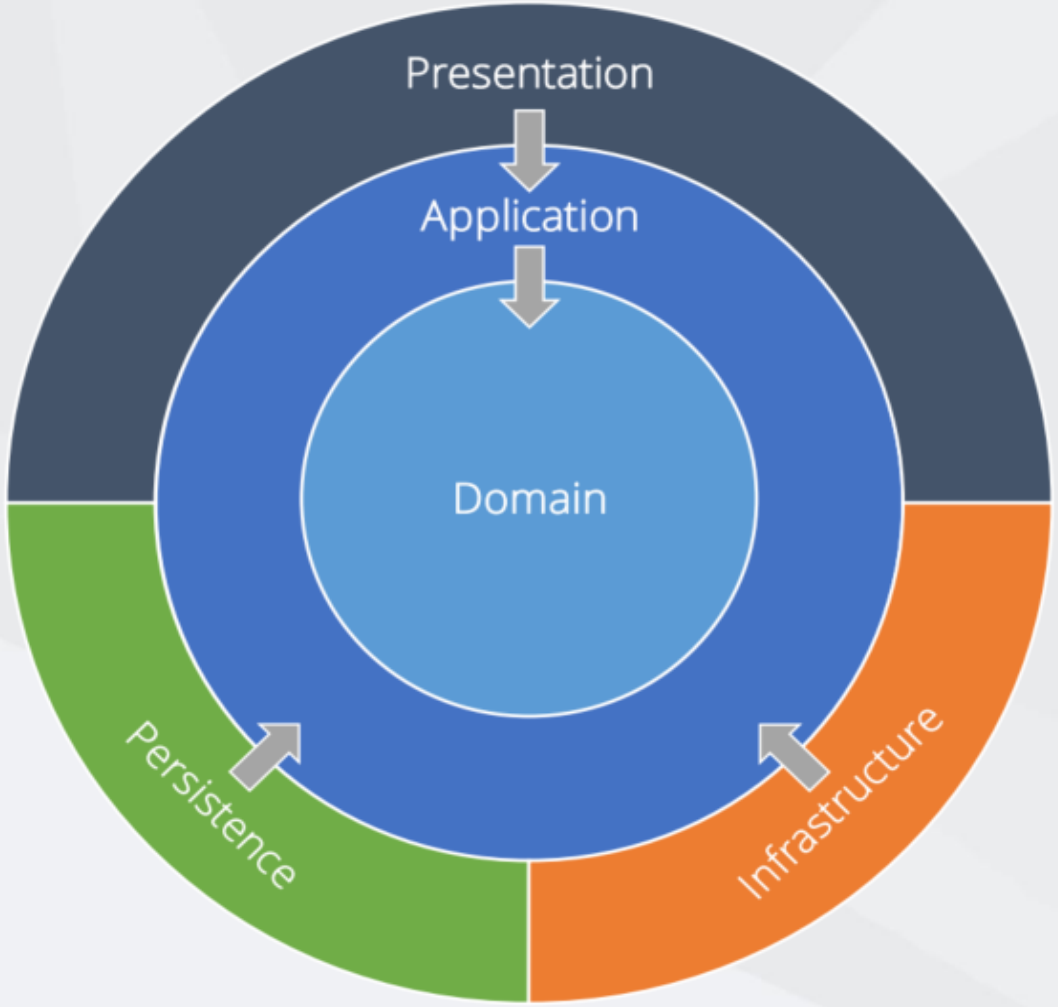
Independent of frameworks

Testable

Independent of UI

Independent of database

Independent anything external



Key Points

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

Overview

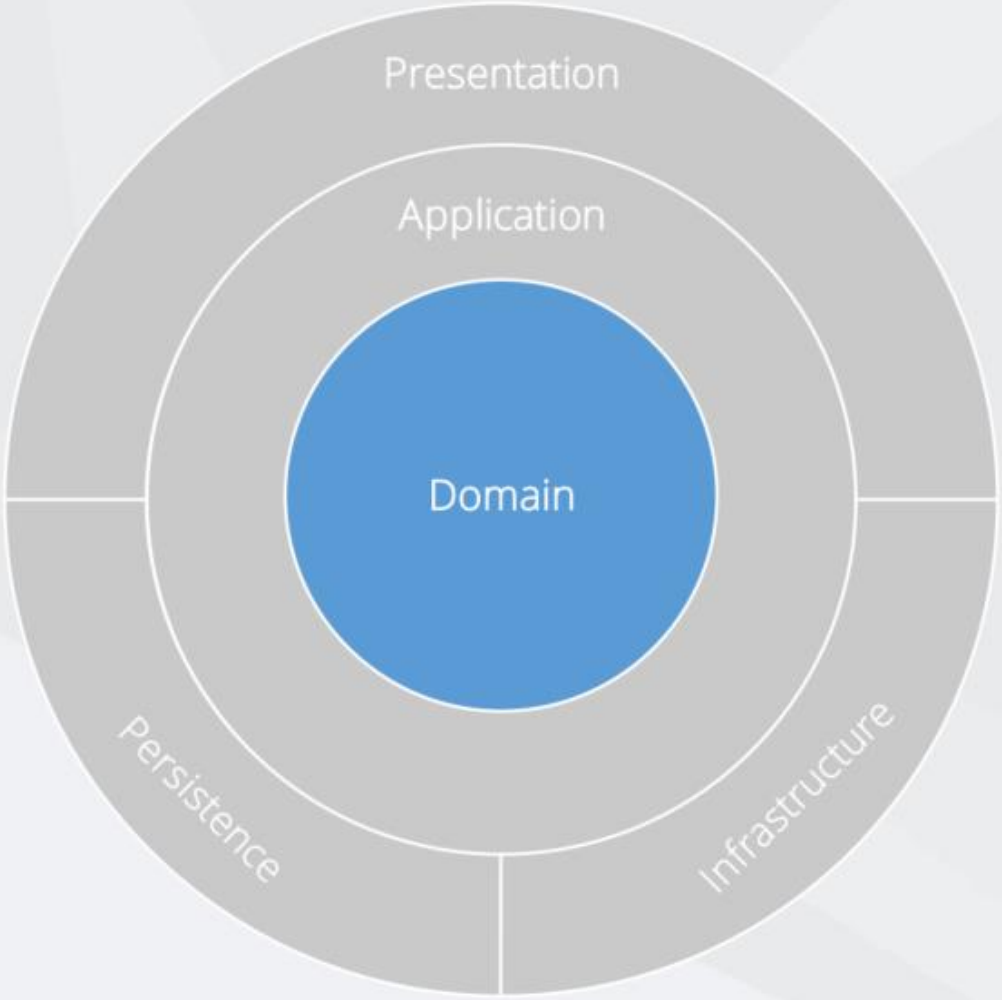
Entities

Value Objects

Enumerations

Logic

Exceptions



Key Points

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

Overview

Interfaces

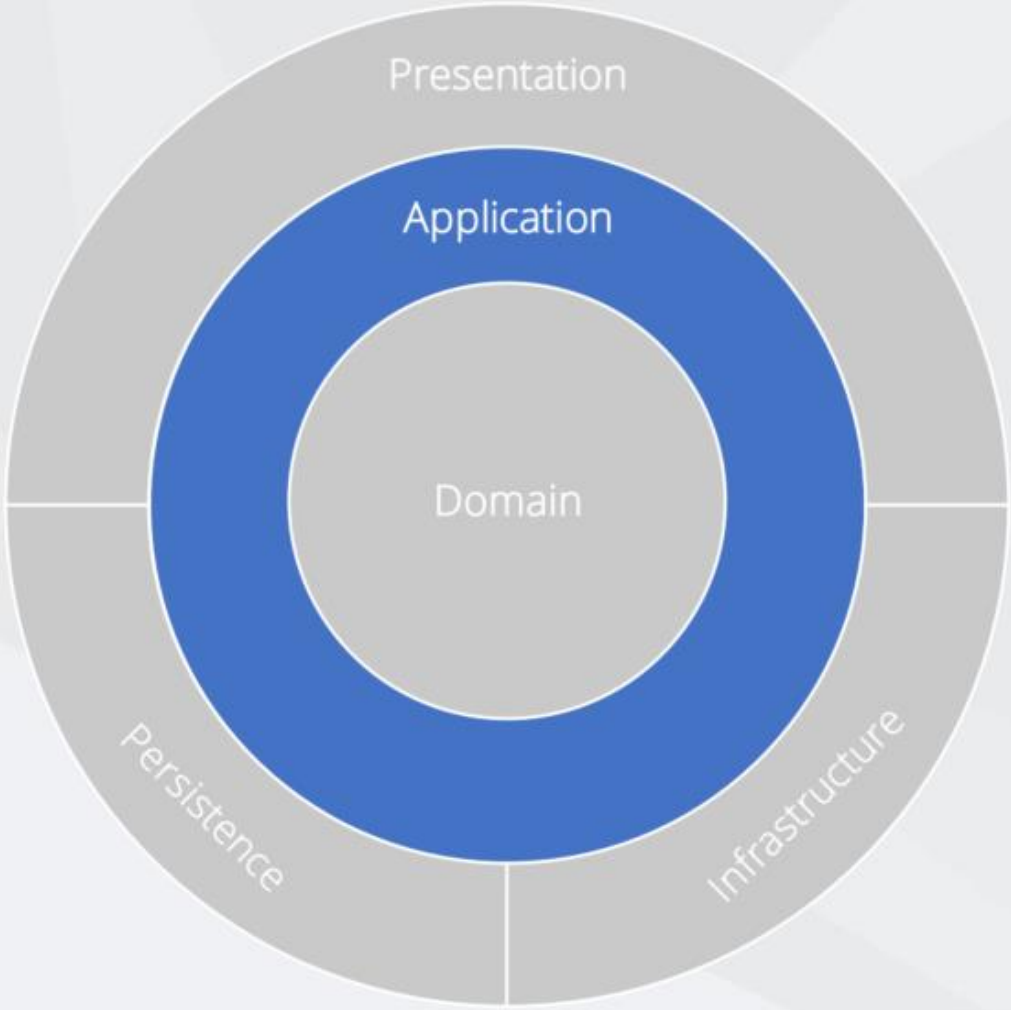
Models

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

Overview

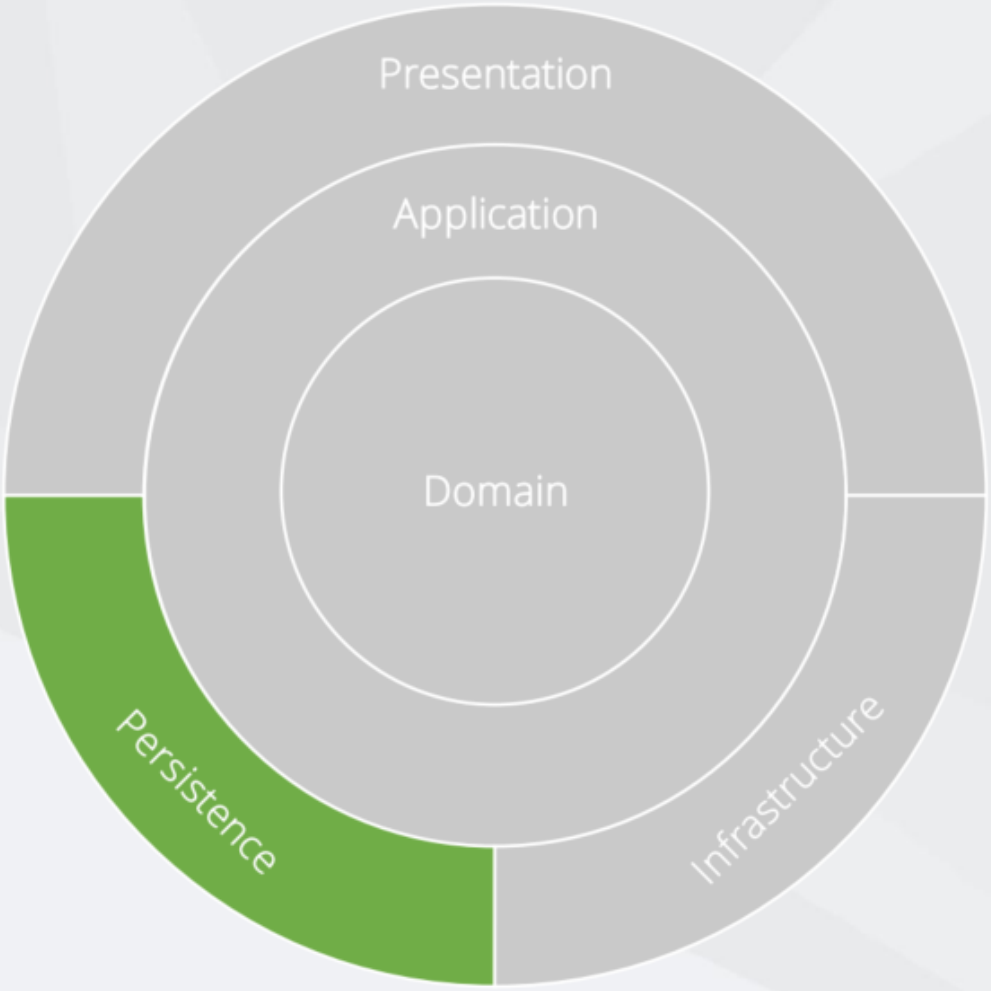
DbContext

Migrations

Configurations

Seeding

Abstractions



Overview

Implementations, e.g.

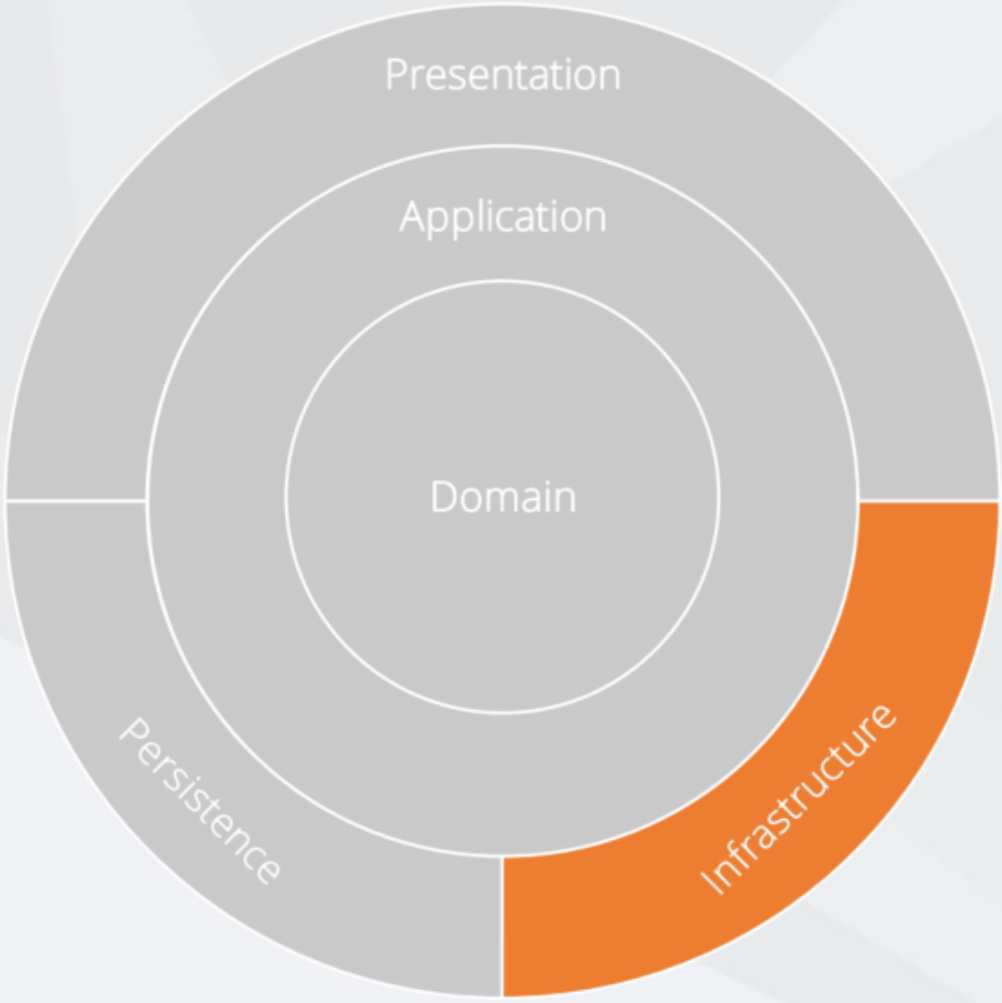
API Clients

File System

Email / SMS

System Clock

Anything external



Key Points

- Contains classes for accessing external resources
- Such as file systems, web services, SMTP and so on
- Implements abstractions / interfaces defined within the Application layer
- No layers depend on Infrastructure layer, e.g. Presentation layer

Overview

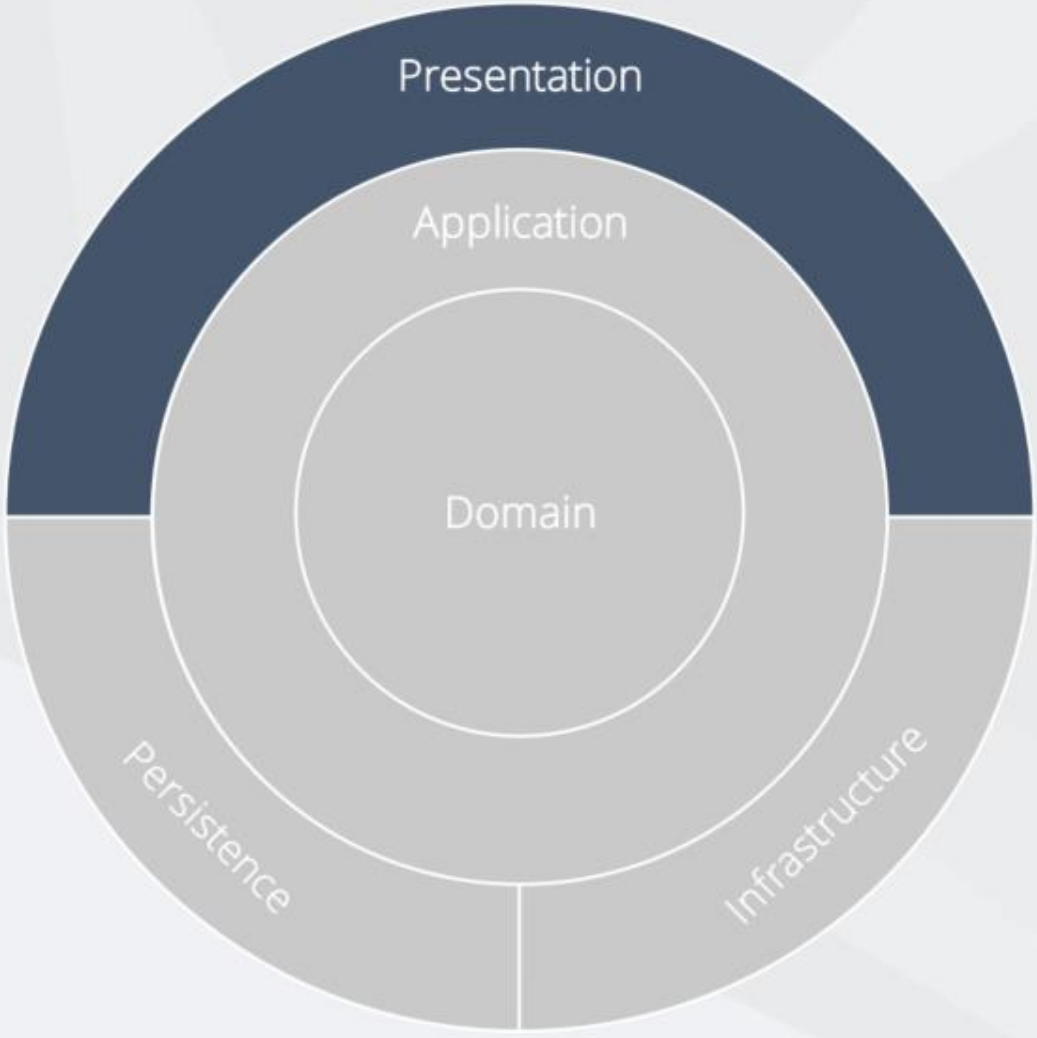
SPA – Angular or React

Web API

Razor Pages

MVC

Web Forms



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

Link: https://github.com/JasonGT/NorthwindTraders/raw/master/Slides.pdf