EF Core 6 and Domain-Driven Design

Understanding Where EF Core 6 Fits Alongside DDD

Julie Lerman

EF Core Expert and Serial DDD Advocate

@julielerman | thedatafarm.com

Version Check

This version was created using:

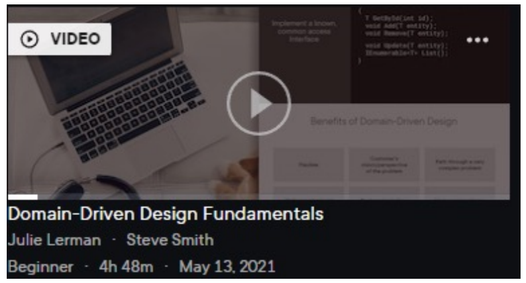
- Entity Framework Core 6
- .NET 6
- Visual Studio 2022
- C# 10
- *Calls out relevant EF Core 7 features*

Not Applicable

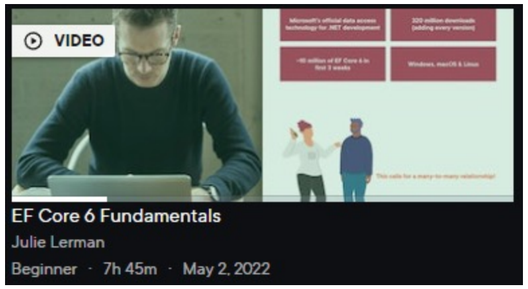
This course is NOT applicable to:

- EF Core 1.0 – EF Core 5
- Entity Framework 6

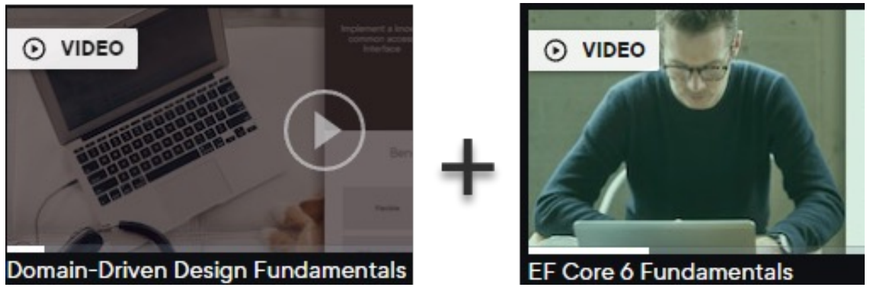
Who Is This Course For?



Familiarity with Domain-Driven Design



Experience with EF Core



Want to know how to best leverage EF Core to map domain models

Domain-Driven Design Fundamentals

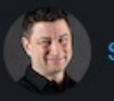
by Steve Smith and Julie Lerman

This course will teach you the fundamentals of Domain- Driven Design (DDD) through a demonstration of customer interactions and a complex demo application, along with advice from renowned DDD experts.

This course is part of: Domain-Driven Design Path

Expand All

Course authors


Steve Smith


Julie Lerman

Course info

Level	Beginner
Rating	★★★★½ (312)
Duration	4h 48m
Released	13 May 2021

Share course

Domain-Driven Design Fundamentals

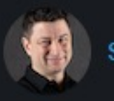
by Steve Smith and Julie Lerman

This course will teach you the fundamentals of Domain- Driven Design (DDD) through a demonstration of customer interactions and a complex demo application, along with advice from renowned DDD experts.

This course is part of: Domain-Driven Design Path

Expand All

Course authors


Steve Smith


Julie Lerman

Course info

Level	Beginner
Rating	★★★★½ (312)
Duration	4h 48m
Released	13 May 2021

Share course

Article • 01/15/2019 • 15 minutes to read

September 2017

Volume 32 Number 9

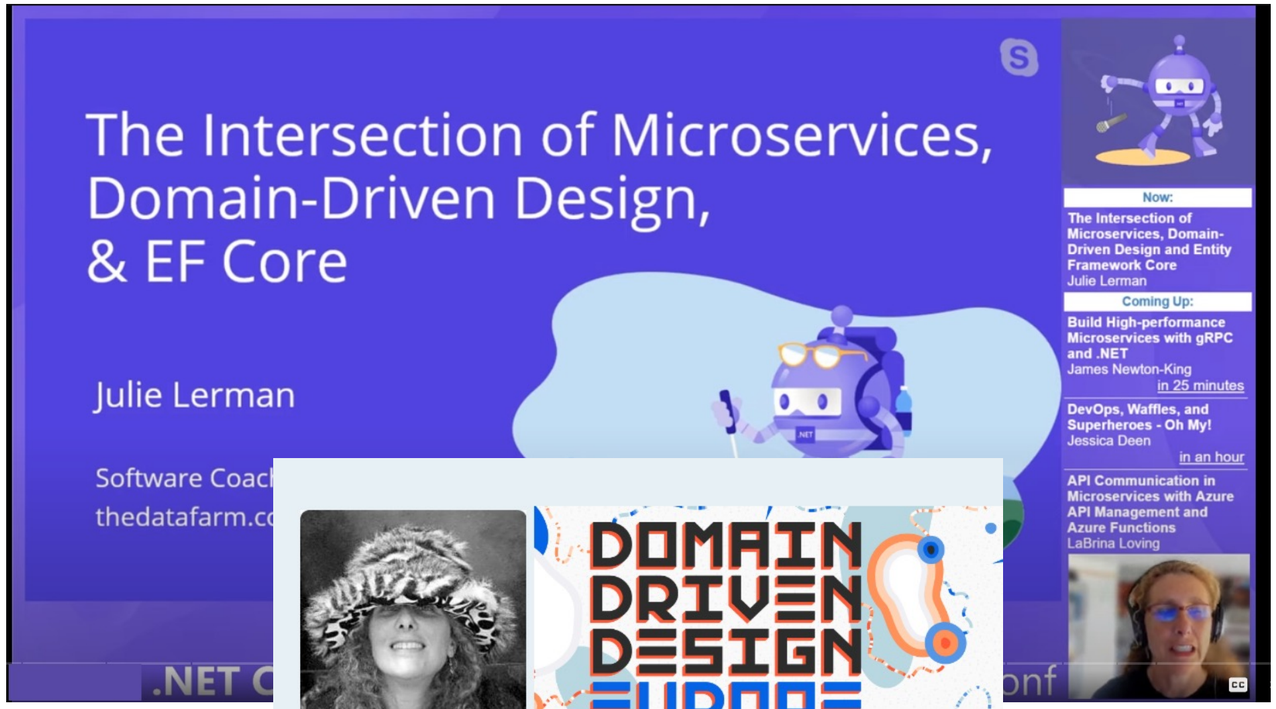
[Data Points]

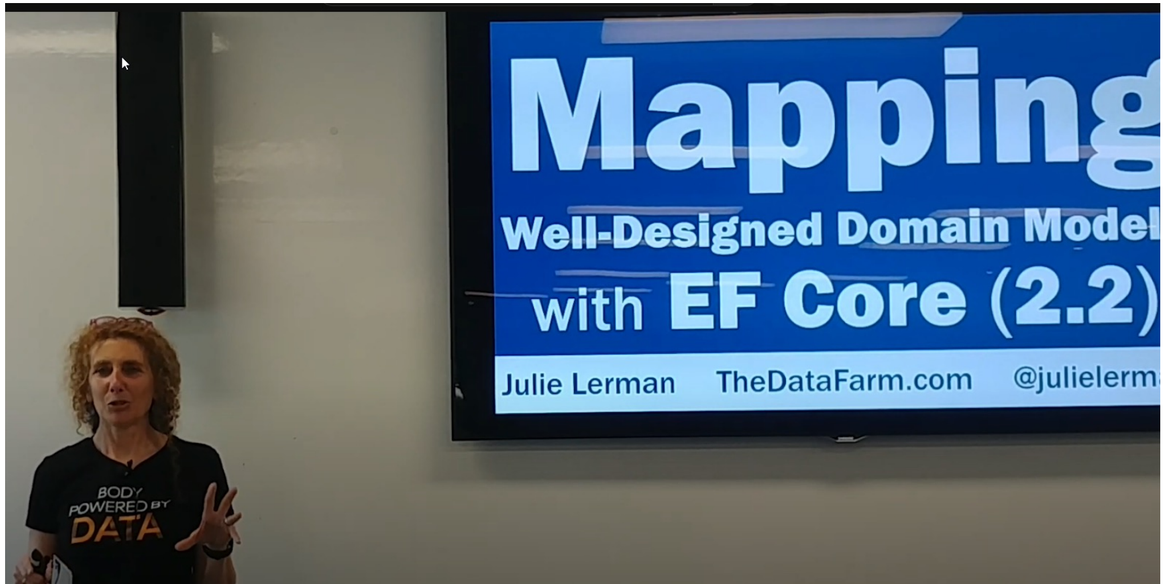
DDD-Friendlier EF Core 2.0

By Julie Lerman

If you've been following this column for a while, you may have noticed quite a few articles about implementing Entity Framework (EF) when building solutions that lean on Domain-Driven Design (DDD) patterns and guidance. Even though DDD is focused on the domain, not on how data is persisted, at some point you need data to flow in and out of your software.

In past iterations of EF, its patterns (conventional or customized) have allowed users to map uncomplicated domain classes directly to the database without too much friction. And my guidance has generally been that if the EF mapping layer takes care of getting your nicely designed domain models into and out of the database without having to create an additional data model, this is sufficient. But at the point when you find yourself tweaking your domain classes and logic to make them work better with Entity Framework, that's a red flag that it's time to create a model for data persistence and then map from the domain model classes to the data model classes.





Julie Lerman

Hands-on

Exploring EF Core Support for DDD Patterns

Entity Framework half-heartedly supported DDD patterns. But the new-from-scratch EF Core has brought new hope for DDD practitioners to be able to map their DDD designed domain classes to a database without making so many concessions that a separate data model was needed. EF Core 2 is very DDD friendly even supporting things like fully encapsulated collections, backing fields and the return of support for value objects. In this hands on workshop you'll work with well-designed aggregates and explore how far EF Core 2 goes to act as the data model between your domain classes and your data store.

Be prepared for this class with the following :

- Windows/macOS or Linux machine
- .NET Core SDK 2.0 SDK Installed
- Visual Studio Code installed

Attendees will be encouraged to work in pairs on a single computer.

Course Overview

How EF Core fits alongside Domain-Driven Design

Strategic and tactical design of the demo app

How EF Core maps domain models by default

How to improve EF Core domain model mapping

Some features of CosmosDB provider

Organizing your persistence logic to support DDD

Module Overview

Quick DDD overview

Why do we care about persistence when talking about DDD?

Where does EF Core fit into building apps via DDD?

EF Core DbContext vs. DDD Bounded Context

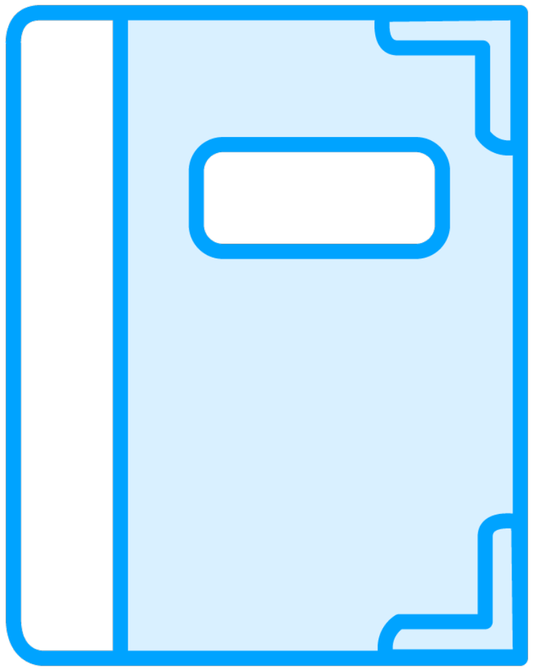
Where's the Code?

We'll dig into the code further on .

Demos can be downloaded from Pluralsight ...

..or github.com/julielerman/EFCore6andDDDPluralsight.

A Quick Review of DDD



Eric Evans' groundbreaking book:

Domain-Driven Design
Tackling Complexity in the Heart of Software

Published 2003

Fondly known as "The Blue Book"

Focus on solving the problems of the domain

Client collaboration drives the effort

Two Principal Areas of Work

Strategic Design

Tactical Design

Strategic Design

Analyze & model the business problem

Strategize solving problems

Identify core domain, supporting sub-domains, generic sub-domains

Discover boundaries of different problem domains

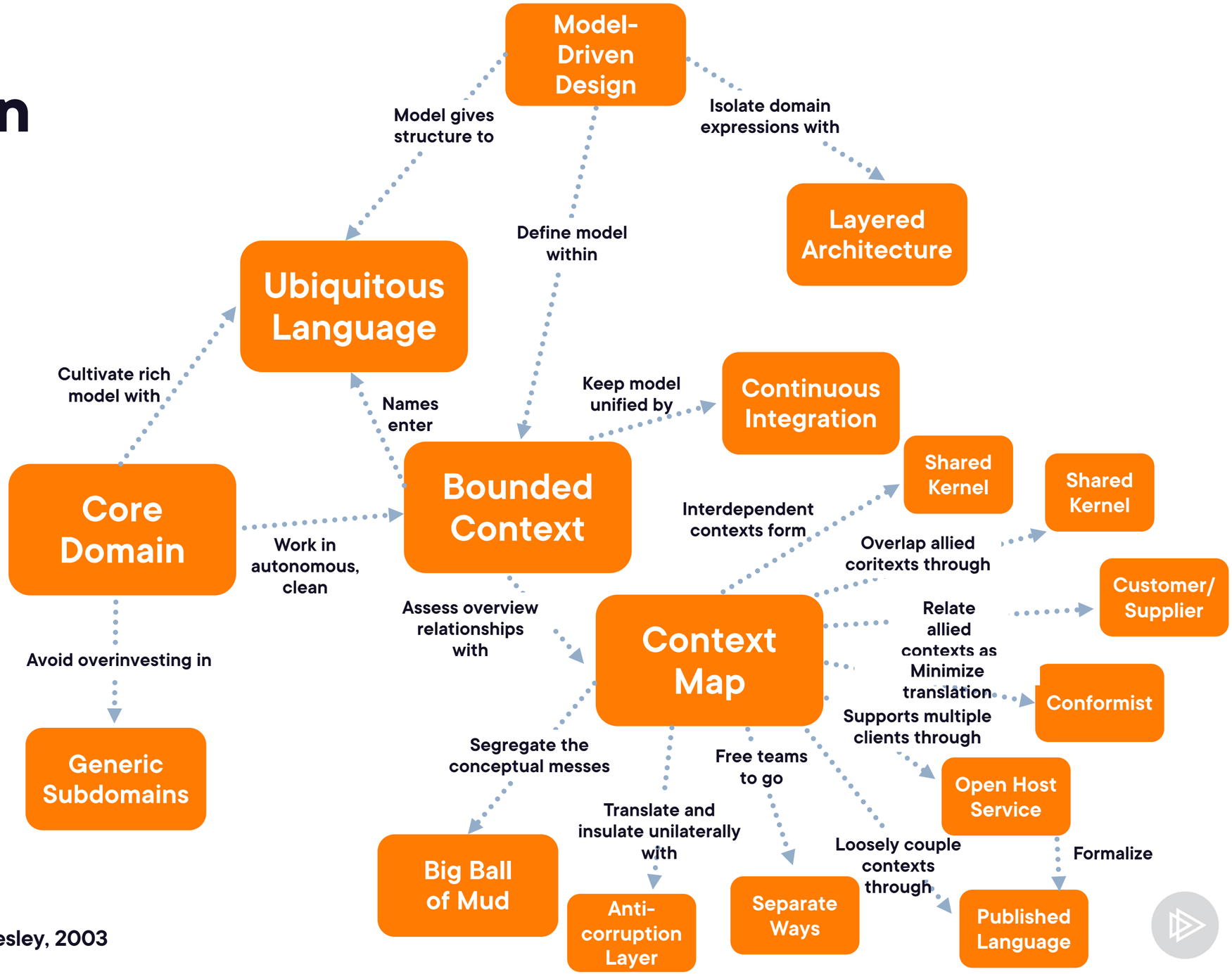


Image based on DDD Mind Map in
Domain-Driven Design, Eric Evans, Addison-Wesley, 2003

Tactical Design

Implement the bounded contexts discovered in strategic

Model sub-domains

Provide structure to the ubiquitous language of a bounded context

Build your software

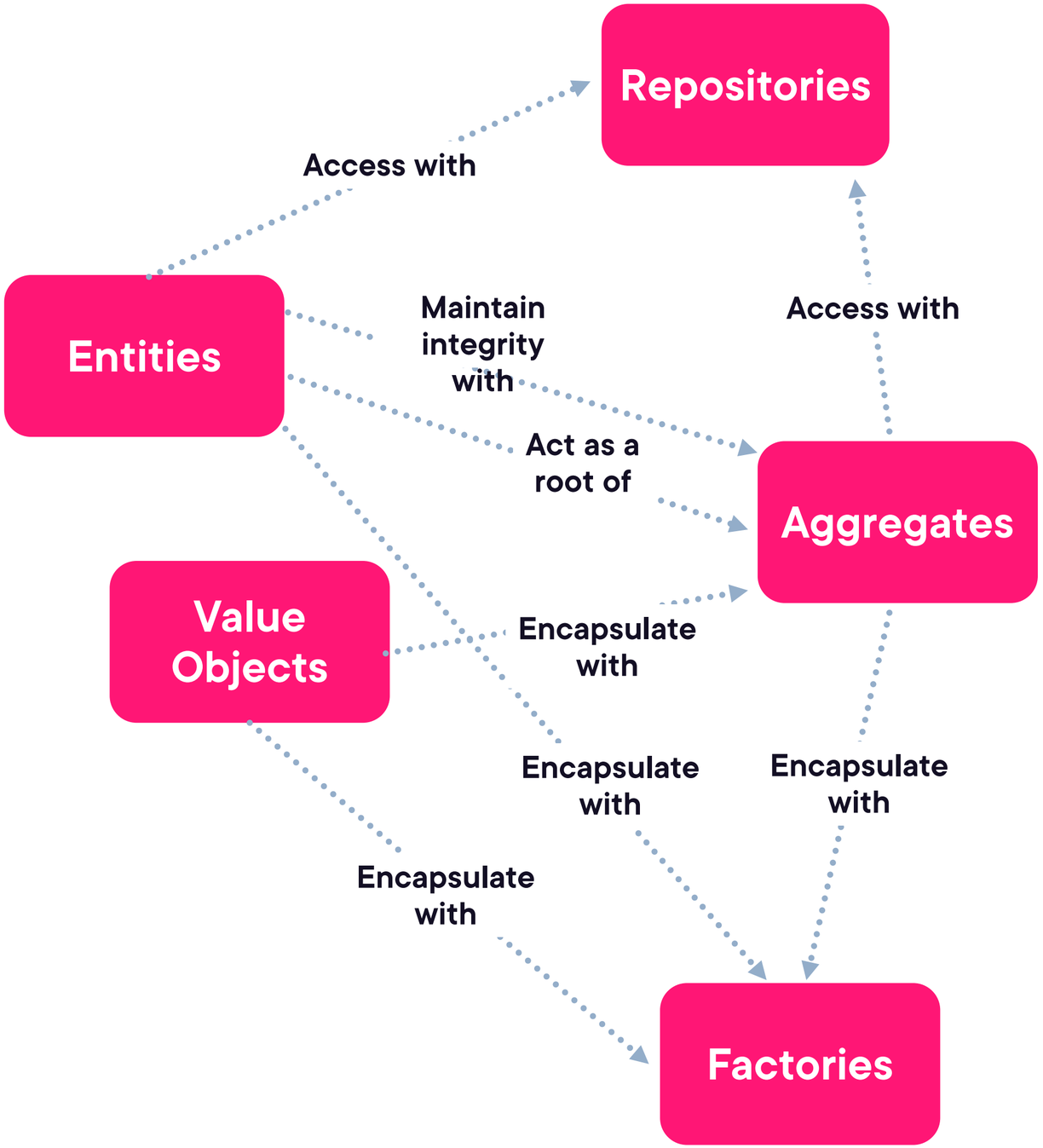
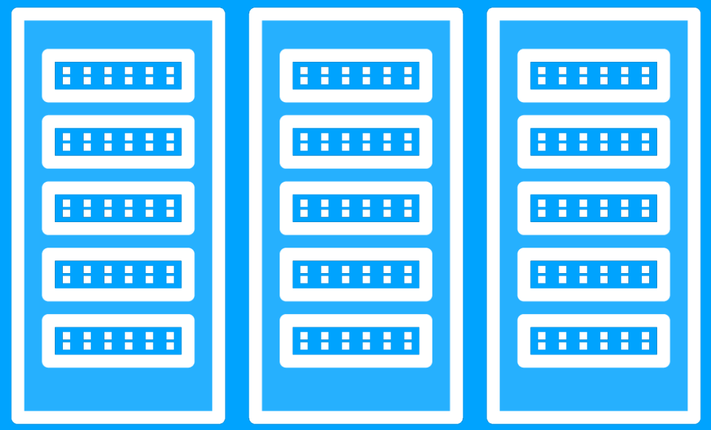


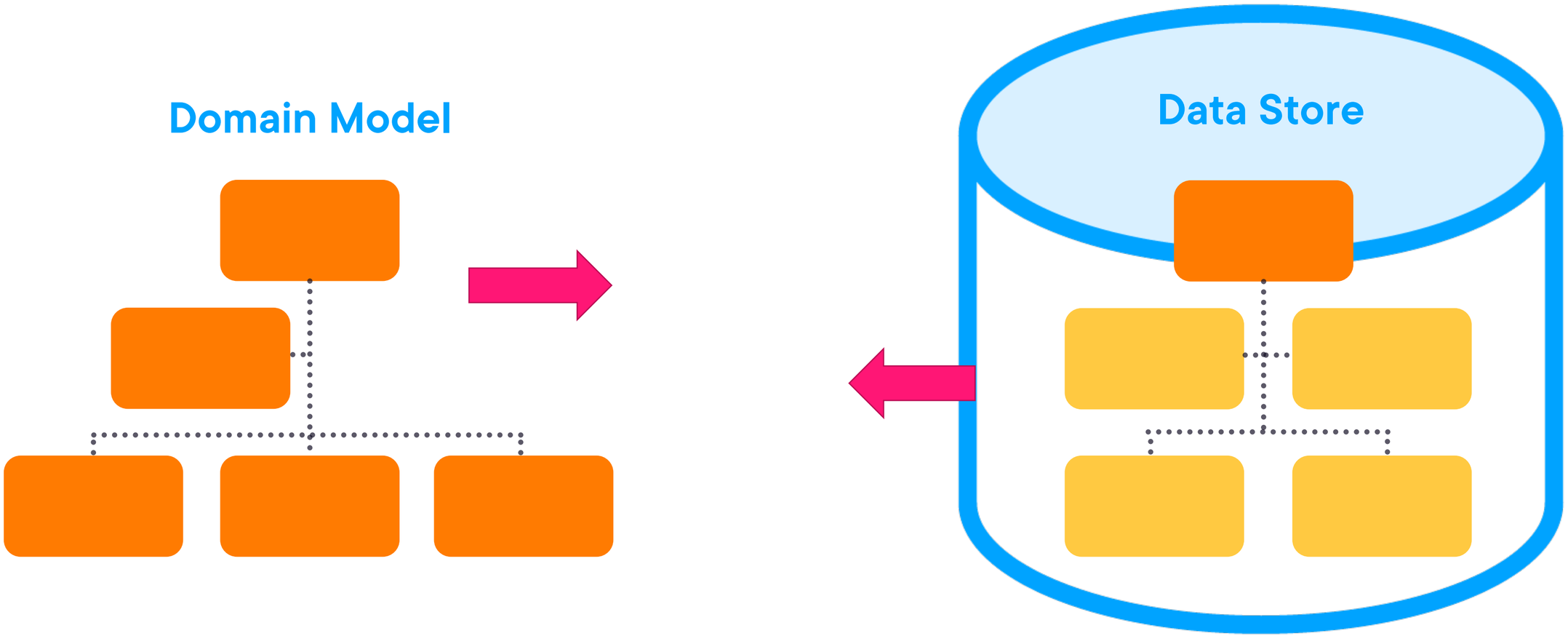
Image based on DDD Mind Map in
Domain-Driven Design, Eric Evans, Addison-Wesley, 2003

So, What About the Data?



Is Data Storage Really Your Business Domain?

Data Persistence Still Matters!

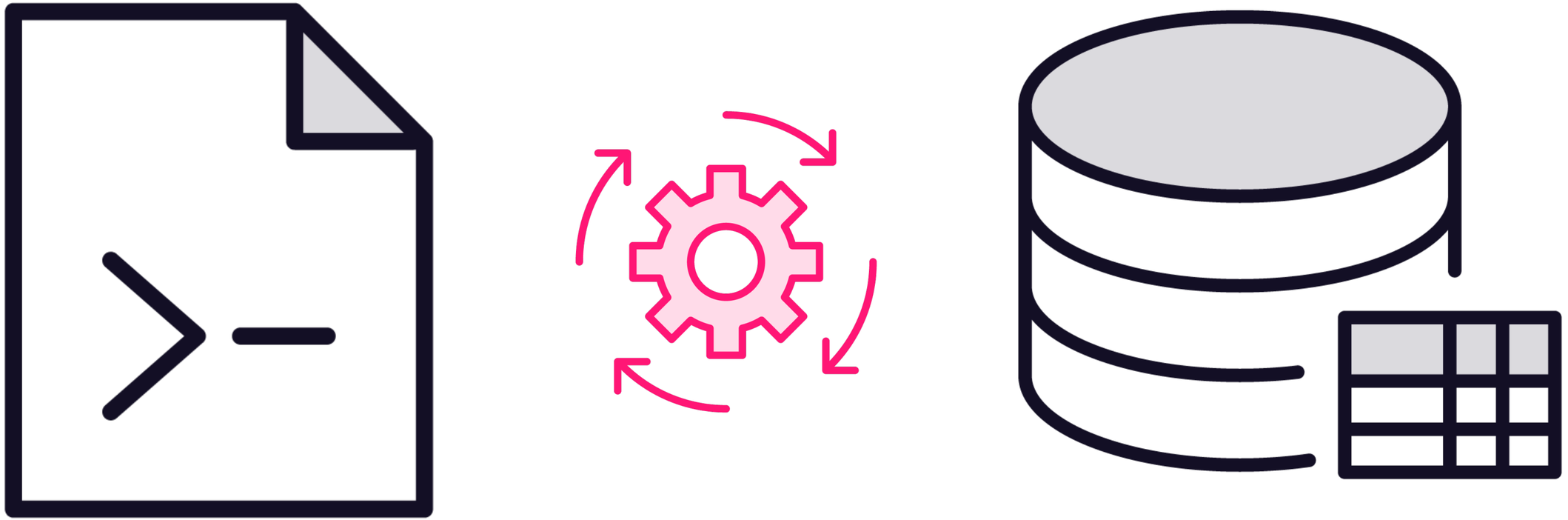


Relational databases do not naturally store domain models

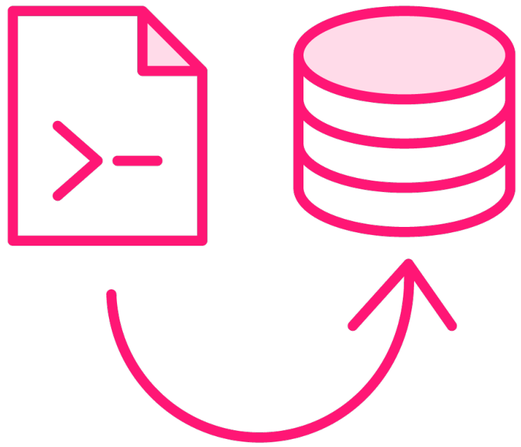
Developers may attempt to reshape the domain to better match how a database stores info

Forgetting that it is already designed to reflect the business process

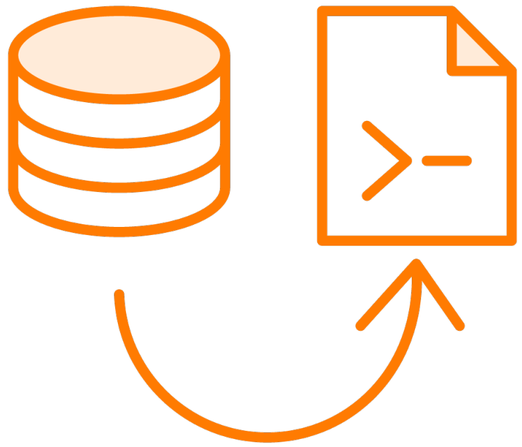
Translate Between Objects and Database Schema



EF Core Transforms Domain Models to RDBMS



Saves domain models into structured data.



Rehydrates objects from query results of that data

Some DDD Support Added Over EF Core Versions

EF Core
Encapsulated collections
Map to fields

EF Core 2
Owned entities support value objects
Value Converters
Keyless entity types
More ...

EF Core 3
CosmosDB provider
Interceptors

EF Core 5
Direct many-to-many
Better owned entity

EF Core 6
Bulk configurations
More DocDB support

EF Core 7
Mapping JSON objects
IDs as value objects
Entity splitting

How Smart is EF Core in Persisting Your DDD Models?

Some mappings will just work

Some mappings require some effort & knowledge

Mapping Backing Fields

An important coding construct to protect properties from accidental misuse

A Construct That EF Recognizes by Default

```
//Private backing field matches its public property

public class Person
{
    private string _passportNumber;

    public void CollectPassport(string passNumber)
   {
      FixUpPassport(passNumber);
      _passportNumber=passNumber;
   }

    public string PassportNumber
    {
        get { return _passportNumber; }
    }
}
```

A Construct That Must Be Mapped using a Configuration

```
//Private backing field doesn’t match its public
property

public class Person
{
    private string _passNum;

    public void CollectPassport(string passNumber)
   {
      FixUpPassport(passNumber);
      _passNum=passNumber;
   }

    public string PassportNumber
    {
        get { return _ passNum; }
    }
}
```

Another Construct That Must Be Mapped using a Configuration

```
//Private field with no public property

public class Person
{
  private string _passportNumber;

  public void CollectPassport(string passNumber)
  {
    FixUpPassport(passNumber);
    _passportNumber =passNumber;
  }

  public bool PassportHasBeenCollected
  {
    get {return
        !String.IsNullOrEmpty( _passportNumber); }
  }
}
```

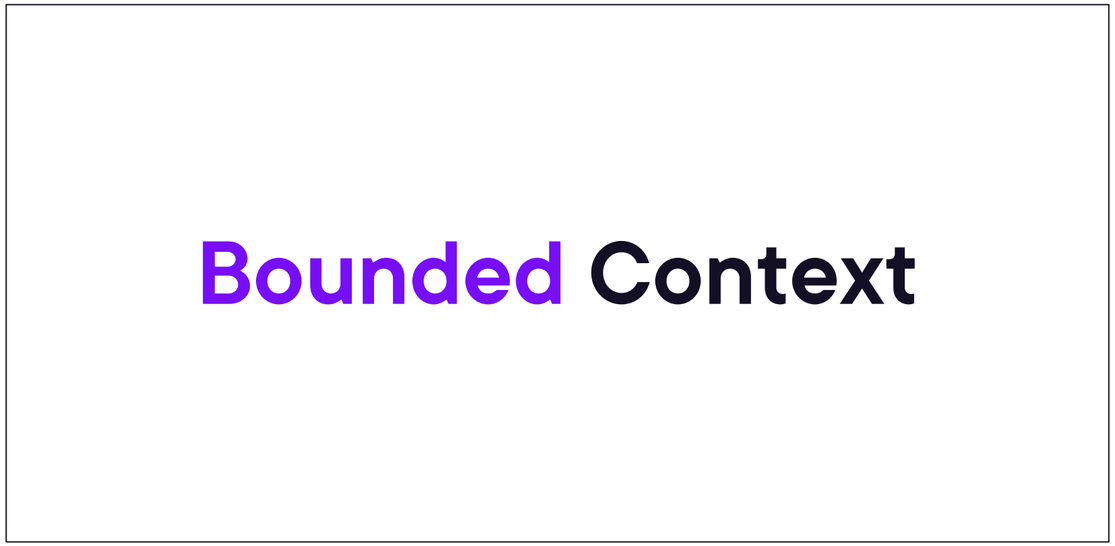
Developers may attempt to reshape the domain to better match how a database stores info

Forgetting that it is already designed to reflect the business process

Clarifying a Point of Confusion Around the Word “Context”

Bounded Context

DbContext



Context in the
ubiquitous language of
Domain-Driven Design

Context in the
ubiquitous language of
EF Core

They Are Very Different Contexts!

DDD Bounded Context	VS.	EF Core DbContext
Represents a bounded scope of a particular domain model and its language		Defines mapping between classes and database schema Coordinates a session with a database to query and store data Tracks state of objects used during a particular instance

Summary

Quick review of DDD & EF Core concepts

Strategic design to analyze the domain

Tactical design to implement the domain

Data persistence is outside of DDD but still needed

EF Core takes on the brunt of the effort of persisting data

Up Next:

Strategic and Tactical Design of Our Domain