Using EF Core in ASP.NET Core Apps

Julie Lerman

Most Trusted Authority on Entity Framework Core

@JulieLerman www.thedatafarm.com

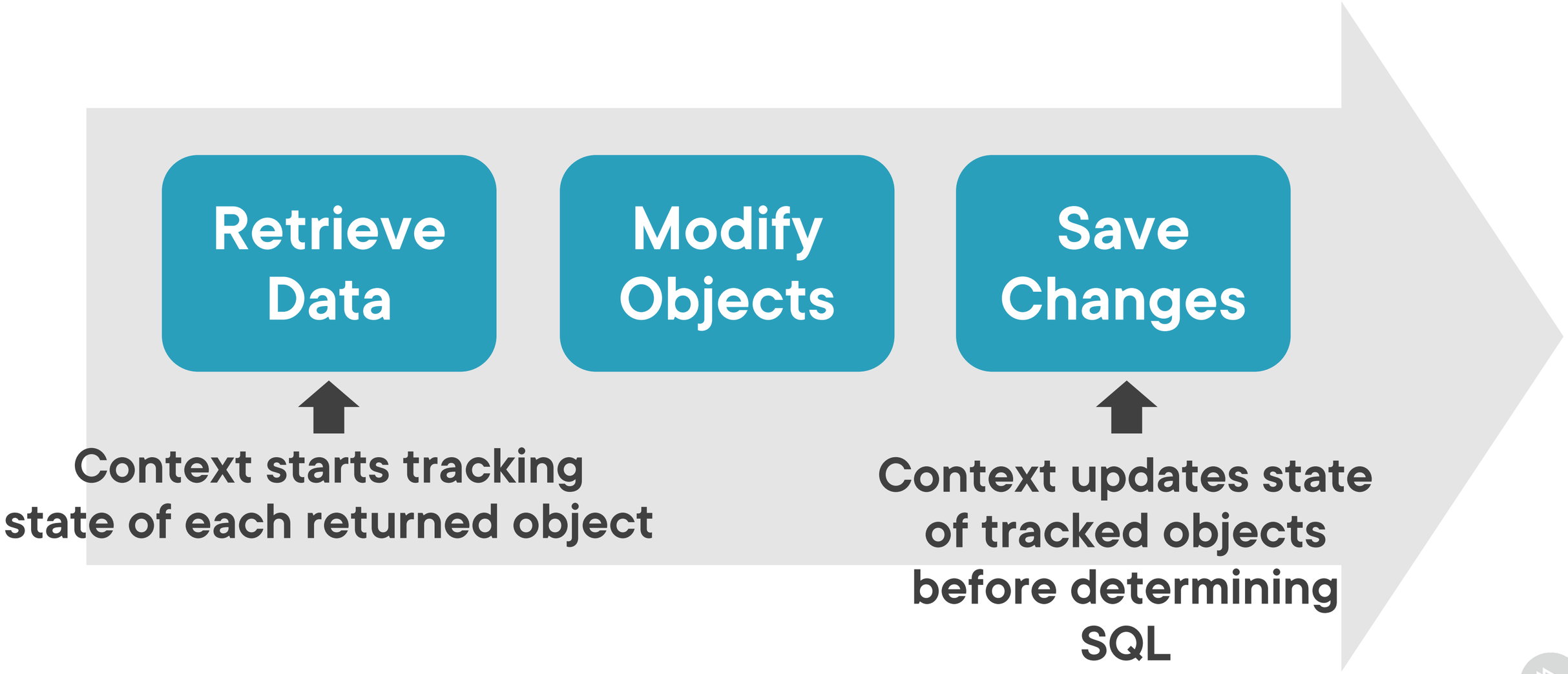
Overview

- Review lifecycle of DbContext in web apps
- Create an ASP.NET Core API project
- Create a template generated controller
- Wire up ASP.NET Core to PubContext
- Learn how to combine entities and DTOs
- Learn tips for debugging and logging
- Interact with controller methods to read, write and delete data

We won’t do a lot of work with related data in this demo.

Reviewing EF Core's Lifecycle in Disconnected Apps

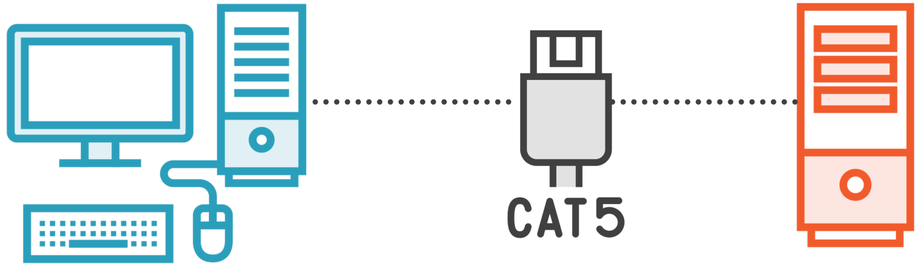
Working in a Single DbContext Instance



Connected Data Access

Client Storing Data Locally

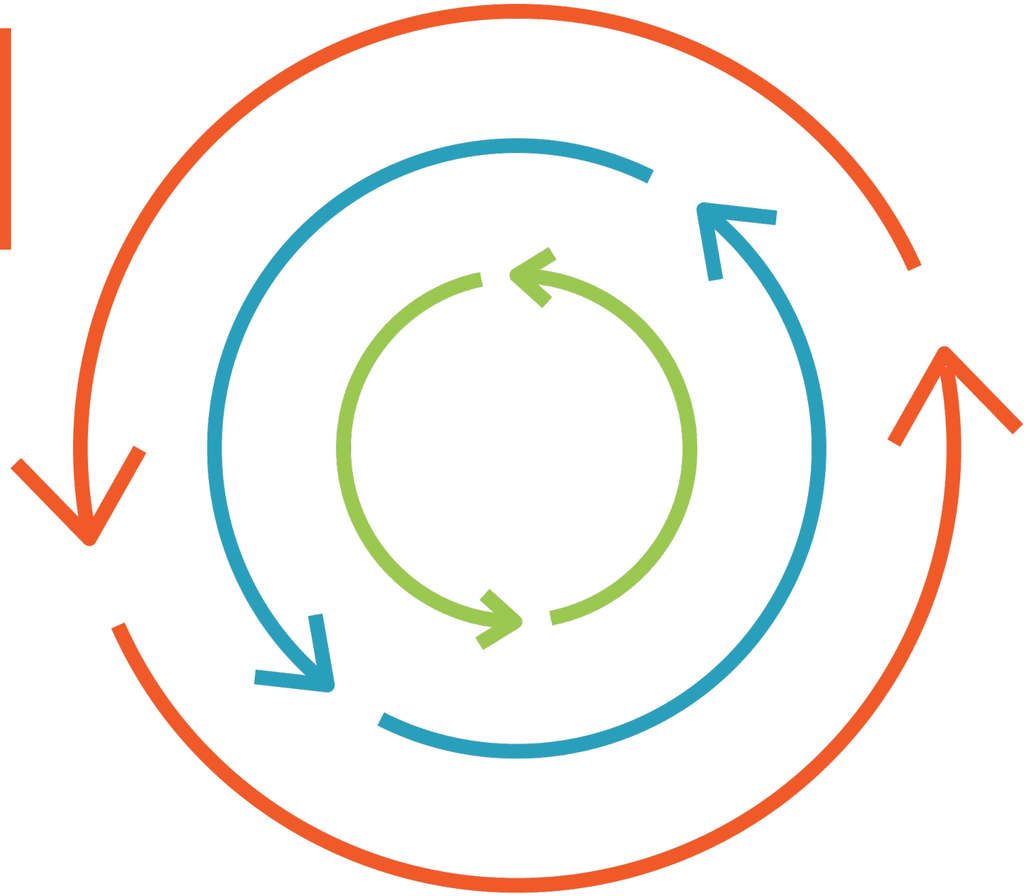
Network Connected Clients



Disconnected Clients

Short-Lived DbContexts

Web application lifetime (app start to recycle)

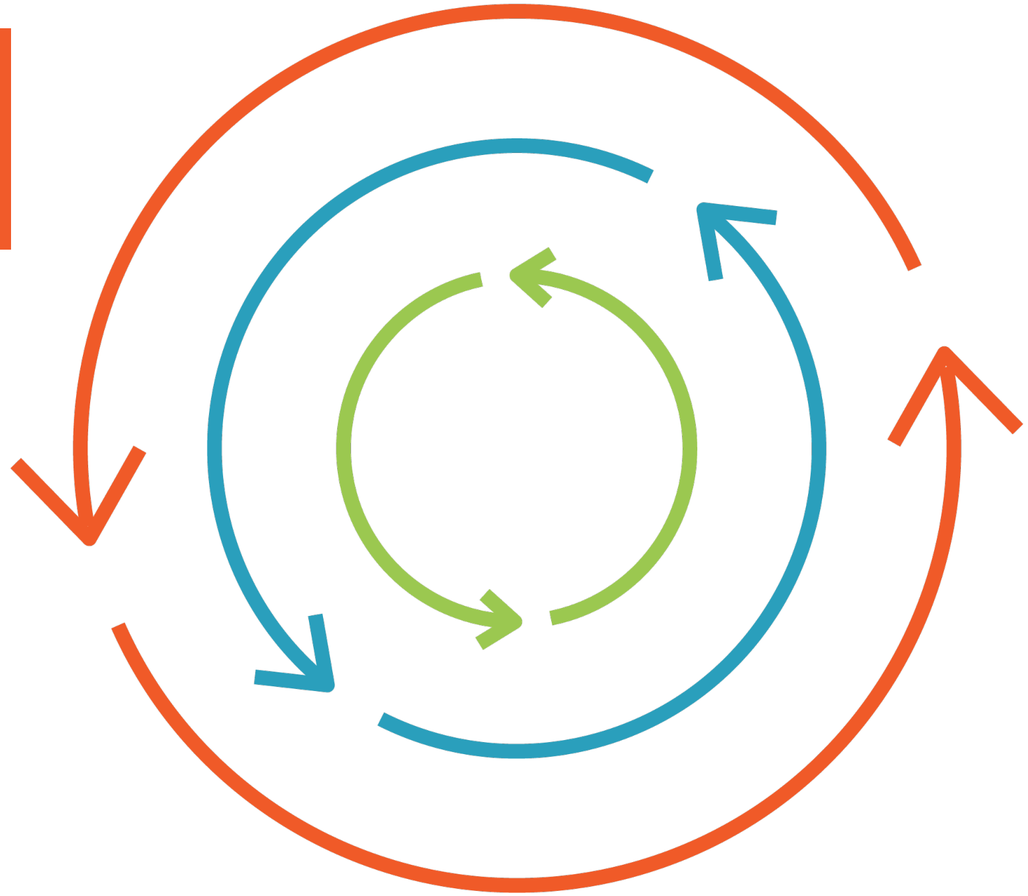


Controller instance per Request

DbContext per controller instance

Short-Lived DbContexts

Web application lifetime (app start to recycle)



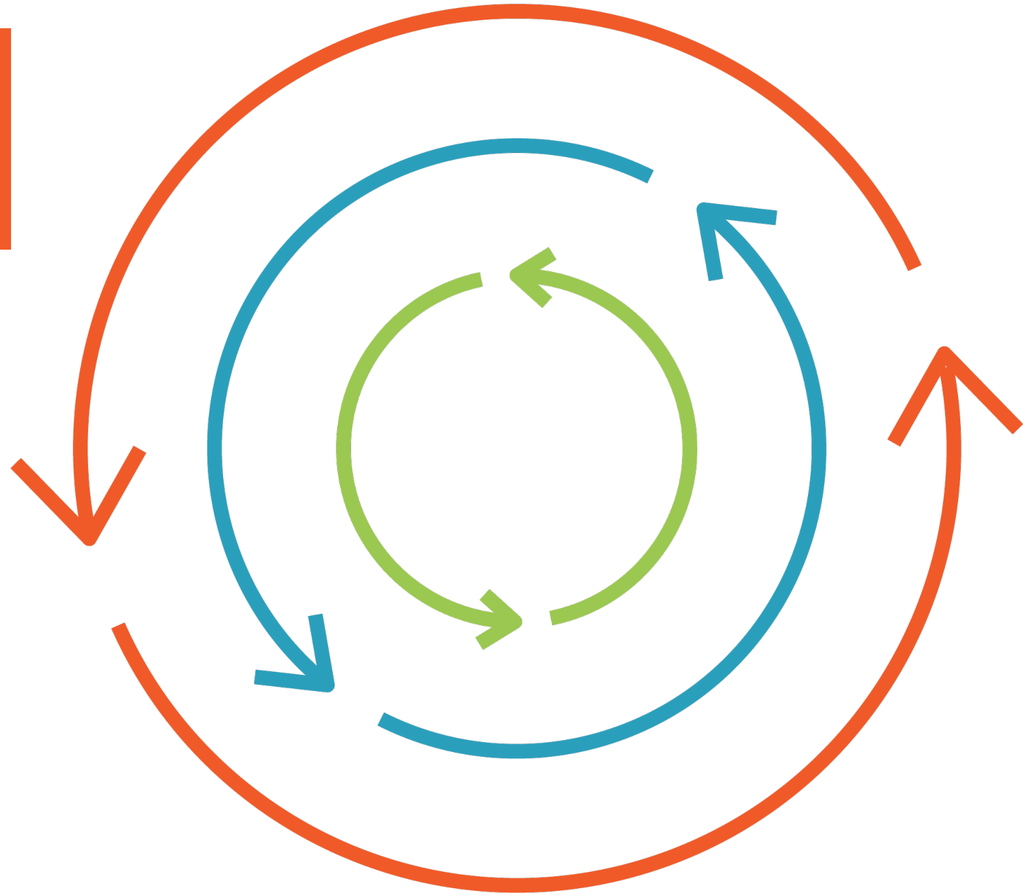
Request some data

DbContext query and return results

When the DbContext is disposed, any change tracking objects disappear. Creating them would be a wasted effort!

Short-Lived DbContexts

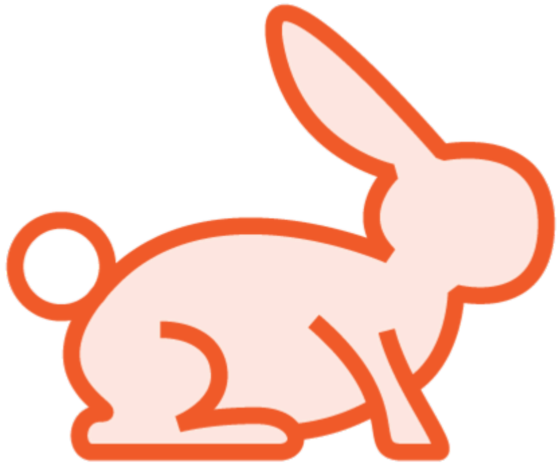
Web application lifetime (app start to recycle)



Save changes to that data

DbContext knows nothing about this data! Needs to know if it should add, update or delete!

Query Performance Bonus in Web Apps



The context can't track data on the client device

Don't waste time and resources tracking

Use no-tracking queries

In fact … just make the entire DbContext a no-tracking context

Various Ways to Inform Context of State

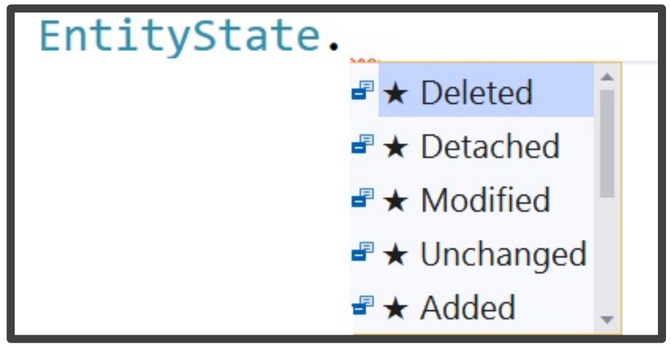
DbSet Methods

```
Authors.Add(newAuth);

Authors.Update(existingAuth);

Authors.Remove(existingAuth);
```

Set DbEntry.State



Retrieve and modify from database

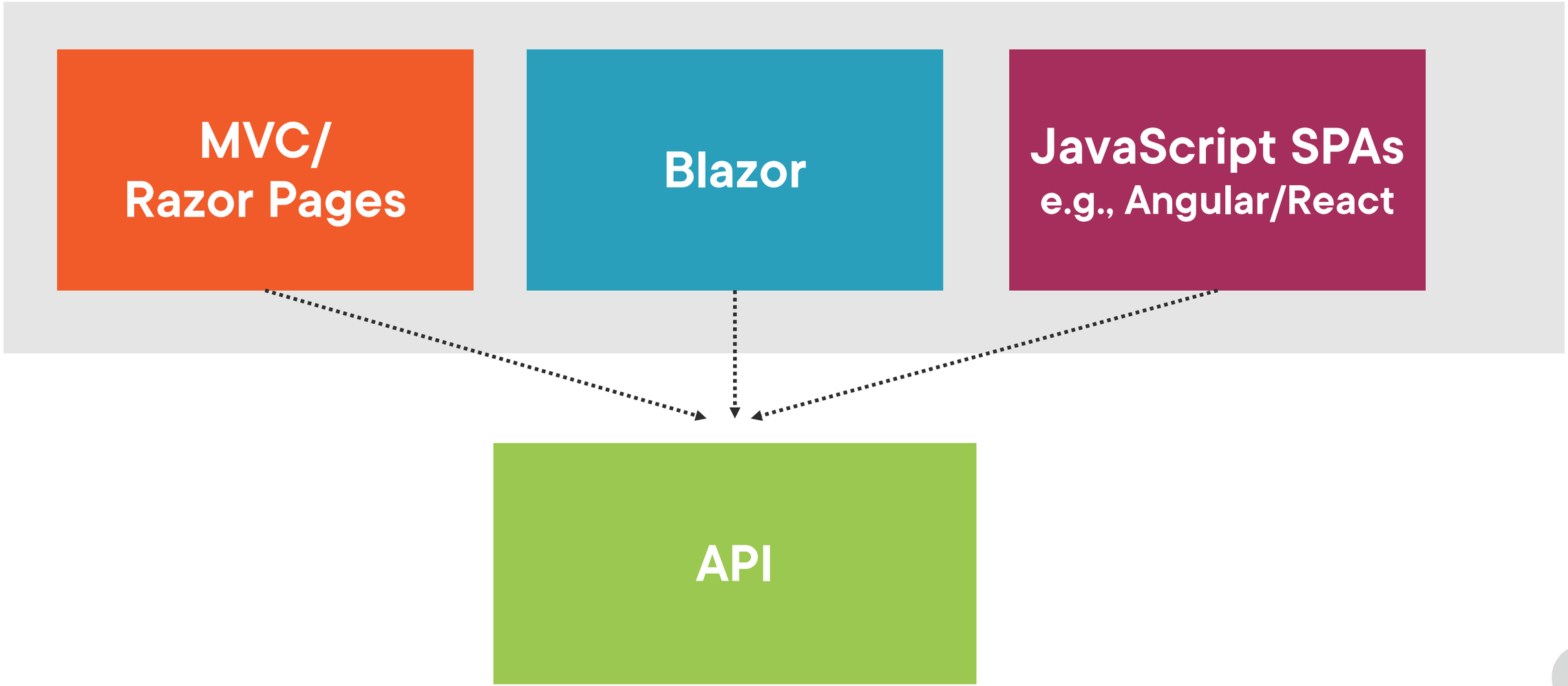
```
void UpdateDBAuthorValues(Author aFromRequest)
{
    var a = _context.Authors
        .Find(aFromRequest.AuthorId);
    //set values with aFromRequestValues
}
```

In disconnected scenarios,
it’s up to you to inform the
context about object state.

Not an EF Core problem, but a standard conundrum across all tech stacks

Adding the ASP.NET Core Project

Front Ends Talk to Server-Side APIs



Adding an Author Controller to the Project

Wiring up ASP.NET Core with EF Core

Add an Authors Controller

1. Add references to projects with entities and DbContext
2. Add controller (API with actions using EF Core)
   (This will add EF Core and other packages to csproj)

Controller with EF Core Actions Template

Stake in the ground starting point

Returning domain objects to or expecting them from the caller is an anti-pattern

Further on, we'll refactor to align with preferred practices

Wiring up the ASP.NET Core App with the DbContext

Wiring up ASP.NET Core with EF Core

Add an Authors Controller

1. Add references to projects with entities and DbContext
2. Add controller (API with actions using EF Core)
   (This will add EF Core and other packages to csproj)

Program.cs

4. Add services for DbContext with UseSqlServer to program.cs

appsettings.json

5. Add connection string config
6. Add EF Core logging config

PubContext.cs

7. Add constructor that takes in DbContextOptions
8. Remove optionsBuilder from OnConfiguring
9. Clean up using statements

Dependency Injection

C# 10 Dependency Injection

Henry Been

Loose coupling

SOLID Principals for C# Developers

Steve Smith

Running the Controller to See the Output and Logs

Getting Related Data

An important lesson about recursive data in our web application!

Refactoring the Controller to Align with Common Practices

Refactoring What Was Created from Controller with EF Core Actions Template

Stake in the ground starting point

Returning domain objects to or expecting them from the caller is an anti-pattern

Now

~~Further on~~, we'll refactor to align with preferred practices

Data Transfer Object (DTO)

Simple class to transfer data between processes

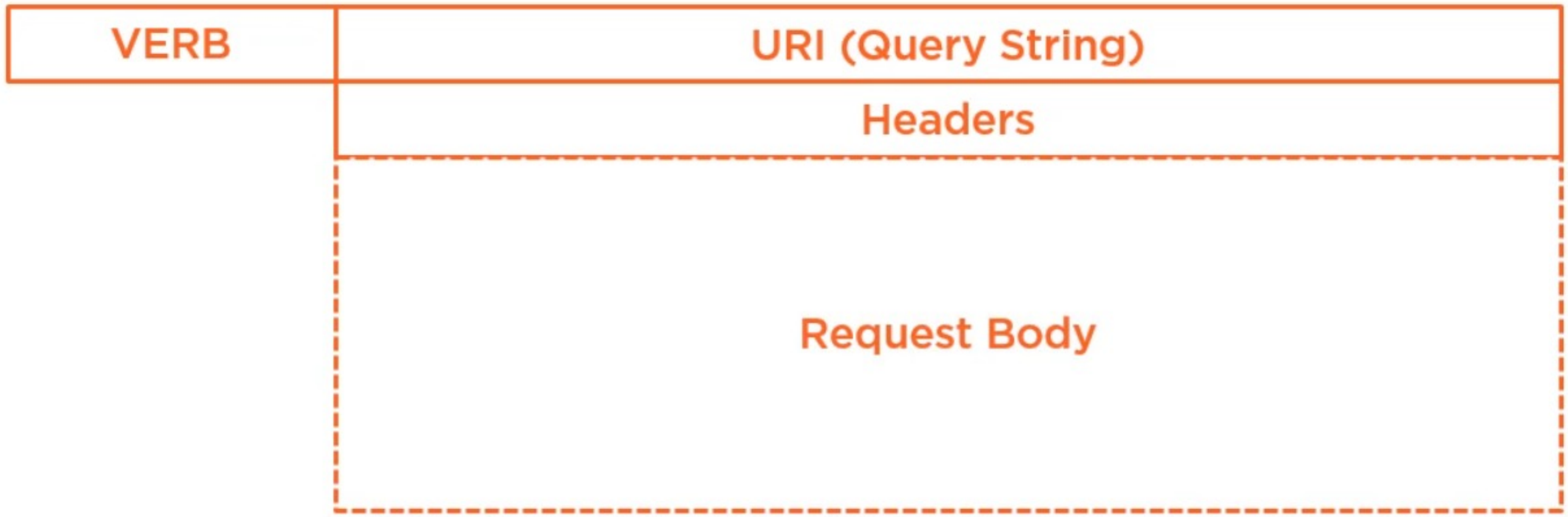
Controller with DTOs

Convert DTOs to entities
Convert entities to DTOs

Exploring and Debugging Insert, Update & Delete Controller Methods

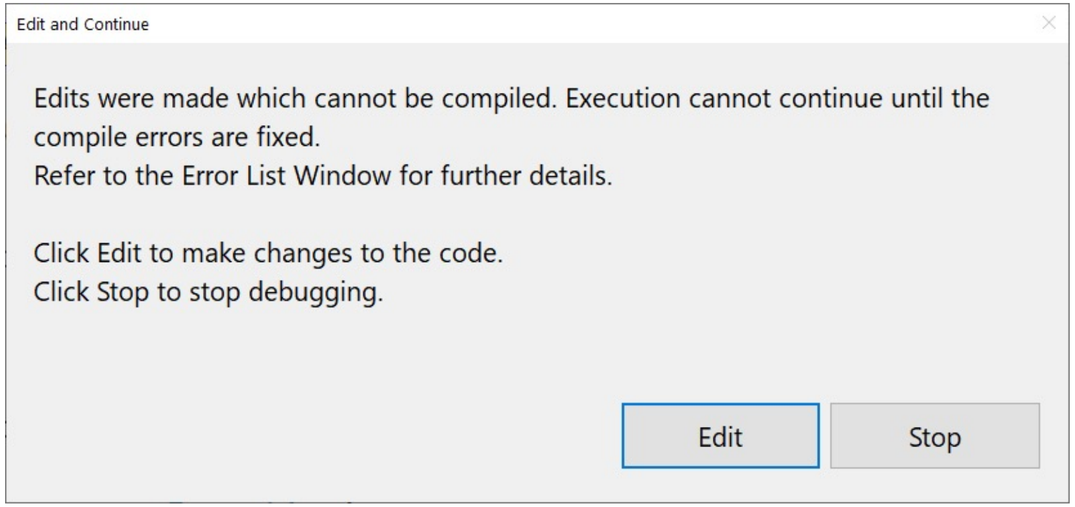
For these persistence methods, we will only work with author objects, not relationships.

REST APIs Have Several Parts

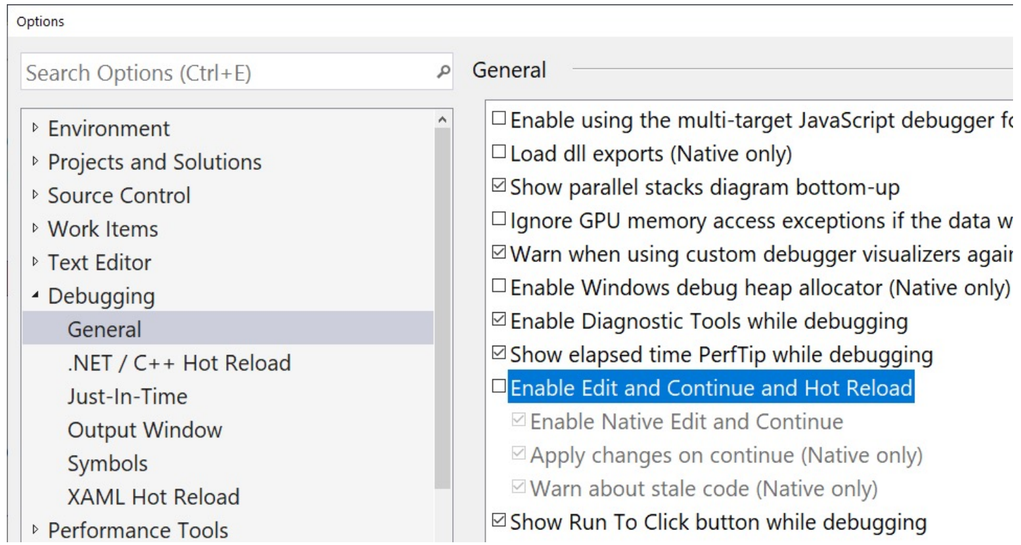


From Shawn Wildermuth's Designing RESTful Web APIs course

Debugging Asynchronous Methods



Debugging will fail in asynchronous methods if Edit and Continue is enabled



Disable Edit and Continue in Tools/Options/ Debugging/General

Using Raw SQL for Controller Methods

The template- generated controller with EF Core code is a pretty good stake in the ground.

Review

For disconnected apps:

- Short-lived DbContexts
- Asynchronous methods
- Non-tracking queries/DbContext

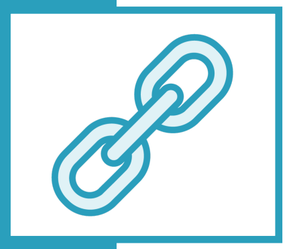
ASP.NET Core can do the hard work for D.I. and logging

Template controller is a stake in the ground

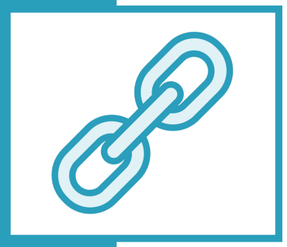
Use DTOs to communicate with calling client and entities with EF Core

Up Next: Testing with EF Core

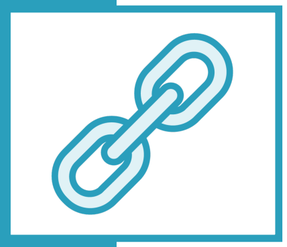
Resources



EF Core Documentation: docs.microsoft.com/ef



Swagger documentation: swagger.io



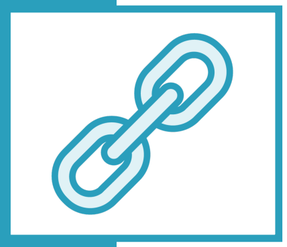
JSONVue extension for Google Chrome: github.com/gildas-lormeau/JSONVue

C# 10 Dependency Injection, Henry Been app.pluralsight.com/profile/author/henry-been

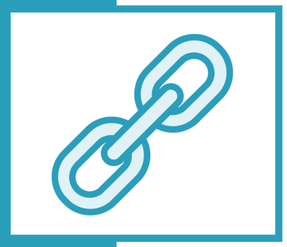
Resources Cont.

Designing RESTful Web APIs, Shawn Wildermuth
app.pluralsight.com/library/courses/designing-restful-web-apis

SOLID Principles for C# Developers
app.pluralsight.com/library/courses/csharp-solid-principles/table-of-contents



Logging in .NET Core and ASP.NET Core
https://docs.microsoft.com/en-us/aspnet/core/fundamentals/logging



Blazor Sample App with EF Core
docs.microsoft.com/en-us/aspnet/core/blazor/blazor-server-ef-core