Using Entity Framework Core in our Controllers

KEVIN DOCKX

ARCHITECT

@KevinDockx https://www.kevindockx.com

Coming Up

Introducing the Repository Pattern

Reading, Creating, Updating and Deleting Resources via Entity Framework Core

Using AutoMapper

Introducing the Repository Pattern

No Repository Pattern

Code duplication

More error-prone code

Harder to test the consuming class

Repository Pattern

The Repository Pattern

An abstraction that reduces complexity and aims to make the code, safe for the repository implementation, persistence ignorant

Introducing the Repository Pattern

No Repository Pattern	Repository Pattern
Code duplication	No duplication
More error-prone code	Less error-prone code
Harder to test the consuming class	Better testability of the consuming class
	Persistence ignorant: switching out the persistence technology is not the main purpose. Choosing the best one for each repository method is.

Demo

Introducing the repository pattern

Demo

Returning data from the repository when requesting resources

Demo

Using AutoMapper to map between entities and DTOs

Demo

Creating a resource

Demo

Updating a resource

Demo

Partially updating a resource

Demo

Deleting a resource

Building a RESTful API in ASP.NET Core 3

- Basic concerns like CRUD, RESTful contract design, content negotiation, data validation ...

Implementing Advanced RESTful Concerns in ASP.NET Core 3

- Advanced concerns like HATEOAS, media types, caching, concurrency, ...

Building an Async API with ASP.NET Core

- Focus on correctly implementing asynchronicity for APIs

Securing ASP.NET Core 2 with OAuth2 and OpenID Connect

- Learn how to secure an ASP.NET Core web application and API

Summary

The repository pattern is an abstraction that reduces complexity and aims to make the code, safe for the repository implementation, persistence ignorant

Using AutoMapper greatly reduces error-prone mapping code

you are ready
to be Awesome !!!
@kevinDockx