

EF Core Essentials

Key Optimizing Strategies for Better Performance



SoftUni Team
Technical Trainers



SoftUni



Software University

<https://about.softuni.bg/>

sli.do

#csharp-db

Table of Contents

1. EF Core Features Overview
2. Generic Repository Design Pattern
3. Inversion of Control
4. Dependency Injection





Features in EF Core

Overview

Recap of EF Core Basics

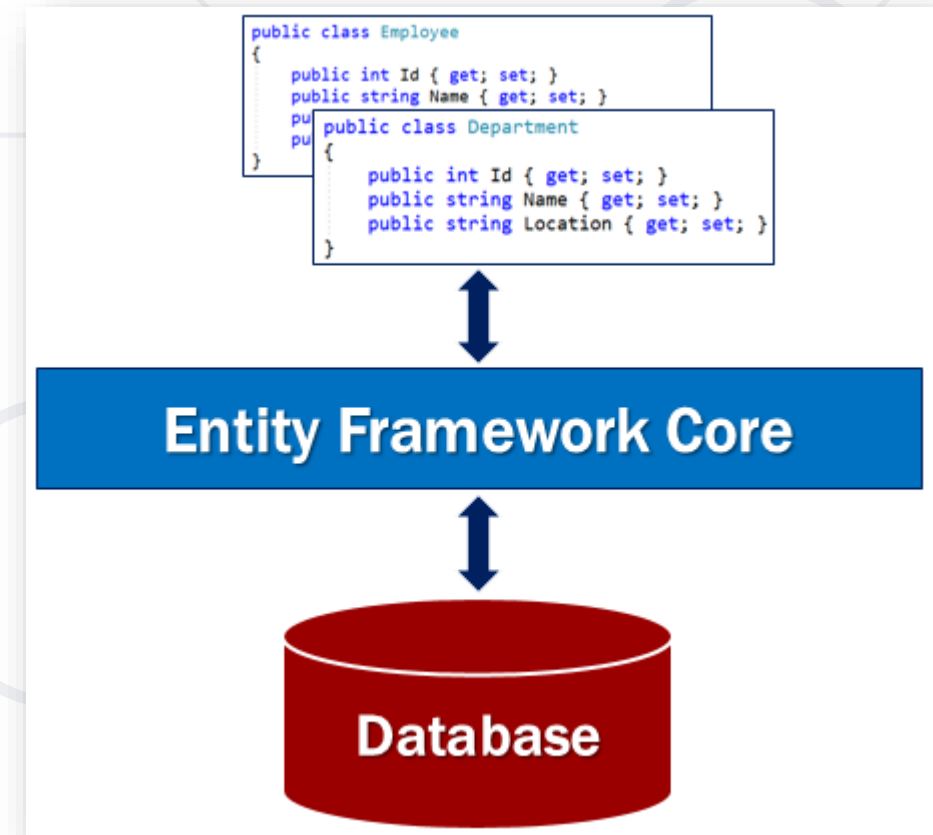
- Entity Framework Core is a lightweight, extensible, **cross-platform**
- ORM (**Object Relational Mapping**) framework for .NET applications
- Provides a **high-level abstraction** for managing relational databases with minimal code
- Supports **many database engines**



- With EF Core, **data access** is performed **using a model**
- A model is made up of **entity classes** and a **context object** that represents a **session with the database**
- The context object allows **querying** and **saving data**



- Generate a model from an **existing database**
- Hand **code a model** to **match the database**
- Once a model is created, use **EF Migrations** to create a database from the model
- Migrations allow **evolving the database** as the model changes



- **Instances** of entity classes are **retrieved from the database** using **Language Integrated Query (LINQ)**
- LINQ allows you to use C# to write **strongly typed queries**
- It **uses your derived context** and entity classes to reference database objects



Querying

- EF Core **passes a representation of the LINQ query** to the **database provider**
- Database providers in turn **translate it** to **database-specific query language**
- Queries are **always executed against the database** even if the entities returned in the result already exist in the context
- Querying allows you to **read data from the database**



- Data is **created**, **deleted**, and **modified** in the database using instances of your entity classes
- Saving data means **adding new entities to the database**, **removing entities**, or **modifying** the properties of existing entities in some way



- Entity Framework Core (EF Core) supports **two fundamental approaches** for saving data to the database:
 - **Change tracking** and **SaveChanges**
 - **ExecuteUpdate** and **ExecuteDelete** ("bulk update")





Local Generic Repository

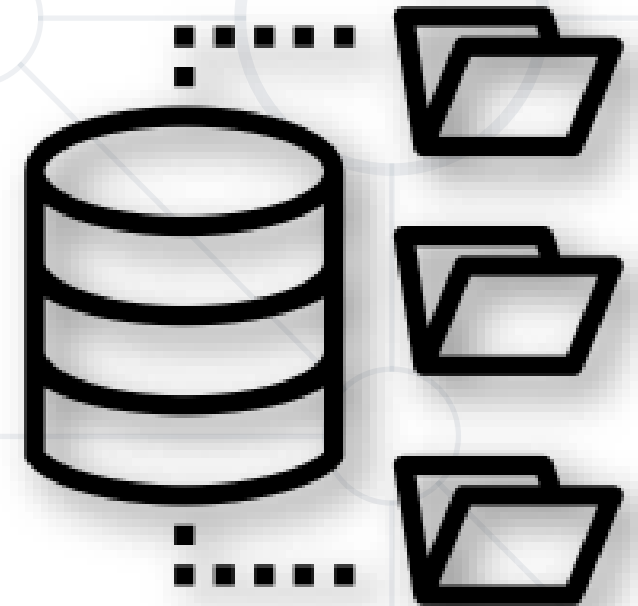
Repository Pattern

Repository Pattern

- Without Repository **direct access to DbContext**
- The **Repository Pattern** is a fundamental **design pattern** in software development
 - Provides an **abstraction layer** between the application's data access logic and the underlying data source
 - It promotes **separation of concerns** and enhances code **maintainability, testability, and scalability**



- The **Generic** Repository pattern in C# is a design pattern that abstracts the application's data layer, making it easier to manage data access logic across **different data sources**
- It aims to **reduce redundancy** by implementing **common data operations** in a single, **generic repository** rather than having separate repositories for each entity type



■ Generic Interface

- A generic repository typically starts with a generic interface defining common operations like **Add, Delete, Find, and Update**
- These operations are **defined in a generic way, applicable to any entity type**

■ Implementation

- The generic interface is then implemented in a concrete class
- This class handles the data source interactions, such as querying a database using an ORM (like Entity Framework)

■ Entity Framework Context

- The implementation will often utilize an Entity Framework context to interact with the database

Why Do We Need Generic Repository DP

- In a **Basic Repository** or **Non-Generic Repository**, we need to create **separate repositories for every entity** in our application
- For example, if we have three entities:
 - *Employee, Product, and Customer*
 - We need to create three repositories: *EmployeeRepository, ProductRepository, and CustomerRepository*



- Creating separate repositories is boring and **repetitive work**
 - Especially **if** all the repositories will **do** the **same kind of work**
 - Typically database **CRUD** operations
- This is against the **DRY (Don't Repeat Yourself)** principle
- To solve the above problem, the **Generic Repository Design Pattern** comes into the picture

Example

```
public ActionResult Index()  
{  
  
public ActionResult AddEmployee()  
{  
  
public ActionResult UpdateEmployee()  
{  
  
public ActionResult DeleteEmployee()  
{
```

EmployeeController



```
GetAll()  
GetById()  
Insert()  
Update()  
Delete()
```



Entity Framework
Data Context
And Entities

Generic Repository



Inversion of Control Design Principle



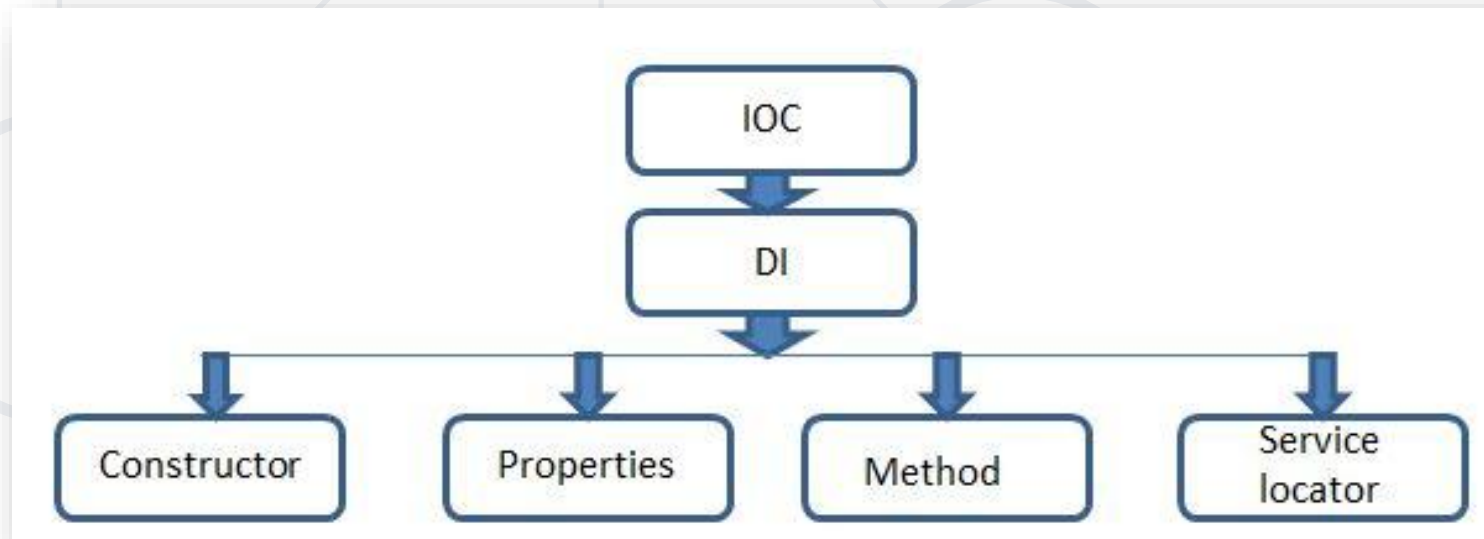
- The **IoC Design Principle** suggests the inversion of various types of controls in **object-oriented design** to achieve **loose coupling** between the application classes
- The **Main class** should **NOT** have a **concrete implementation** of an aggregated class
- It should **depend on the abstraction** of that class
- Here, control means any **extra responsibilities** a class has **other than its main** or fundamental responsibility

What is Dependency Injection?

- IoC can be done **using Dependency Injection (DI)**
- How to **inject concrete implementation** into a class using abstraction (**an interface inside**)
- The main idea of **dependency injection** is to **reduce the coupling between classes** and **move the binding of abstraction and concrete implementation out of the dependent class**
- DI is **how one object knows** about another abstracted dependent object

Ways to Achieve DI

- Injection via **Constructor**
- Injection via **Property**
- Injection via **Method**
- Injection via **Service Locator**





Live Demo

Cinema Web Application

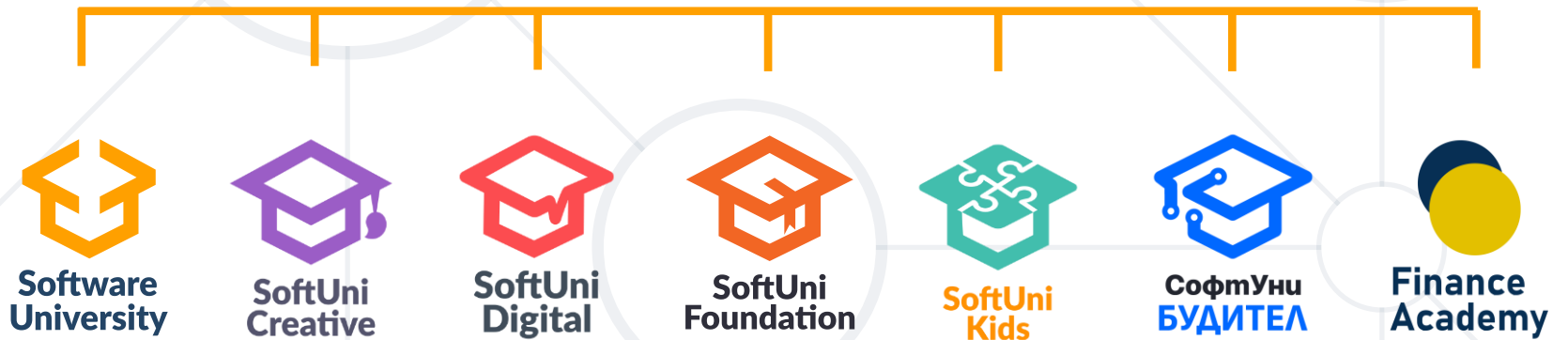
- **EF Core** main features Overview
- **Generic Repository Design Pattern**
- **IoC** and **DI**



Questions?



SoftUni



SoftUni Diamond Partners



THE CROWN IS YOURS



- Software University – High-Quality Education, Profession and Job for Software Developers

- softuni.bg, about.softuni.bg

- Software University Foundation

- softuni.foundation

- Software University @ Facebook

- facebook.com/SoftwareUniversity



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg>
- © Software University – <https://softuni.bg>

