Zajímavé návrhové vzory

Tomáš Herceg

Microsoft Regional Director

Microsoft MVP (ASP.NET)

Proč návrhové vzory?

- Obecná řešení pro často se objevující situace
- Pojmenované → výhoda při komunikaci

- Různé typy návrhových vzorů
  - GOF
  - Enterprise návrhové vzory
  - Architektonické vzory

Domain Model

- Business data a business operace reprezentována pomocí objektů a jejich metod
- Nemusí být 1:1 se schématem databáze
  - Ani by neměl být
  - Návrh aplikace by měl začít právě návrhem domain modelu
    - Praxe je často jiná
- Alternativy
  - Data v SQL databázi, práce s nimi např. přes stored procedury, v aplikaci není nutná třída pro reprezentaci záznamu z databáze, používá se např. DataSet, DataTable
    - Dnes již překonané

Data Mapper

- Mapování dat ze SQL databáze (případně jiné) na objekty doménového modelu
- Typicky řeší i vazby mezi tabulkami
  - Lazy loading, eager loading
  - Dědičnost
    - Table per class
    - Table per type
    - Table per hierarchy
- ORM je implementací Data Mapperu
  - a dalších vzorů

Identity Map

- Řeší konkurenční přístup k datům v rámci jedné transakce
  - Např. načteme řádek z tabulky Order s Id=1
  - Upravíme na něm nějakou vlastnost
  - Řádek načteme znovu (třeba jako součást jiného dotazu)
  - Která verze platí?
- Identity Map v rámci každé transakce eviduje všechny entity (podle primárního klíče)
  - Pokud již entita existuje, není materializována znovu, ale použije se existující instance

Unit of Work

- Ohraničuje business transakci
- Typicky propojená s Identity Map
- Eviduje změněné objekty a umožňuje promítnout jejich změny do datového úložiště
  - Řeší i pořadí updatů
- Často používá IDisposable a vzor Registry
  - Vytvoření instance zaregistruje Unit of Work do aktuálního vlákna
  - Podpora vnořování UOW – použijeme Stack
  - Dispose Unit of Work odregistruje

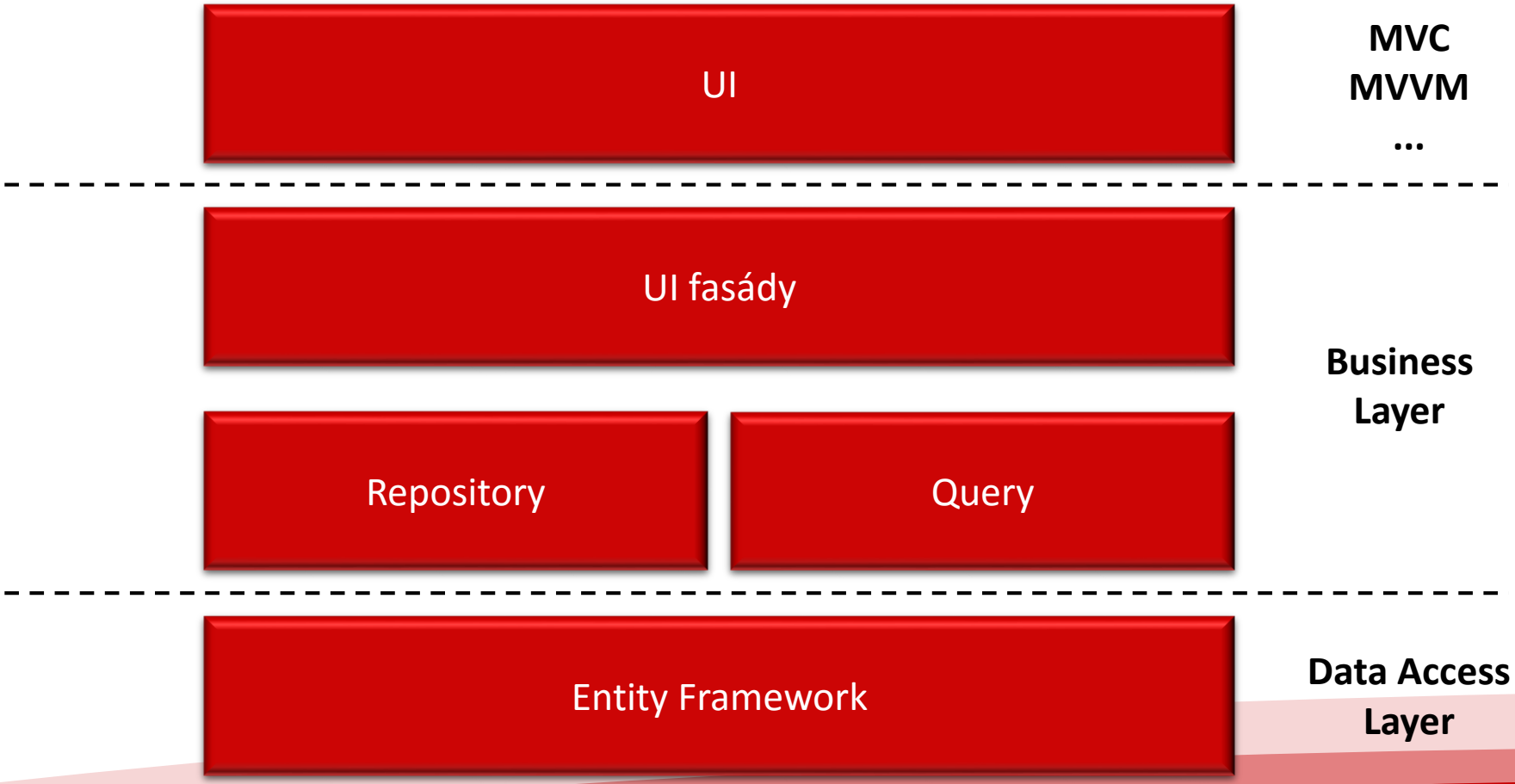
Repository

- Umožňuje CRUD operace nad kolekcí objektů
  - Např. databázovou tabulkou
- Add, Update, Delete, FindById
- Někdo umožňuje i vracení IQueryable
  - V praxi není úplně vhodné
    - Svádí to k psaní složitých dotazů přímo přes repozitář
    - Jednoduché dotazy se v aplikaci často opakují
  - Lepší je použít Query objekt

Query Object

- Dotaz nad daty reprezentován pomocí třídy
- Metoda Execute
- Další možnosti
  - Parametrizovatelnost dotazů
  - Obecná podpora stránkování, řazení, filtrování
  - Možnost přidat např. metodu GetTotalCount
  - Post-processing výsledků

Příklad architektury



Data Access Layer

- Entity Framework
  - EDMX nebo Code First model
  - Datové entity
    - Rozhraní IEntity<TKey>
    - Povinná vlastnost Id

Business Layer

- Repozitáře
  - GetById
  - Insert
  - Update
  - Delete

  - Žádné složité dotazy

- Query objekty
  - Stránkování
  - Řazení
  - Filtrování

  - Jediná možnost, jak se dotazovat databáze na více hodnot
  - Vrací DTO:
    Data Transfer Objekty

Business Layer

- Fasády
  - Třídy s metodami připravenými pro uživatelské rozhraní
    - Vrací DTO, přijímají DTO
    - Vyhazují výjimky (ty ošetří UI), kontrolují oprávnění
    - Připraveny být vystaveny jako web service
    - Nezávislé na prezentační vrstvě
- Unit Of Work
  - Ohraničuje business transakci
  - Poskytuje abstrakci nad Entity Frameworkem
    - Repozitáře a Queries jej používají pro komunikaci s DB

Business Layer

- Business objekty
  - Pro složitější entity (např. objednávka včetně položek, dodavatelů a všech přidružených dat)
    - Create – vytvoří a zinicializuje nový
    - Load – načte objekt z databáze
    - Save – zpropaguje změny do databáze
      (přidání položek, vyhození položek, změna dodavatele atd.)
    - Další metody pro příslušné procesy
- Identity Map
  - Zabraňuje vzniku více instancí jednoho BO v rámci jedné Unit of Work

DEMO

Implementace Unit of Work

SOLID

- 5 pravidel pro testovatelný kód
- Na netestovatelném kódu se IoC/DI používá špatně