MUNI
FI

lasaris LAB OF SOFTWARE ARCHITECTURES AND INFORMATION SYSTEMS

PV179 – Dotazovanie a modifikácia persistentných dát

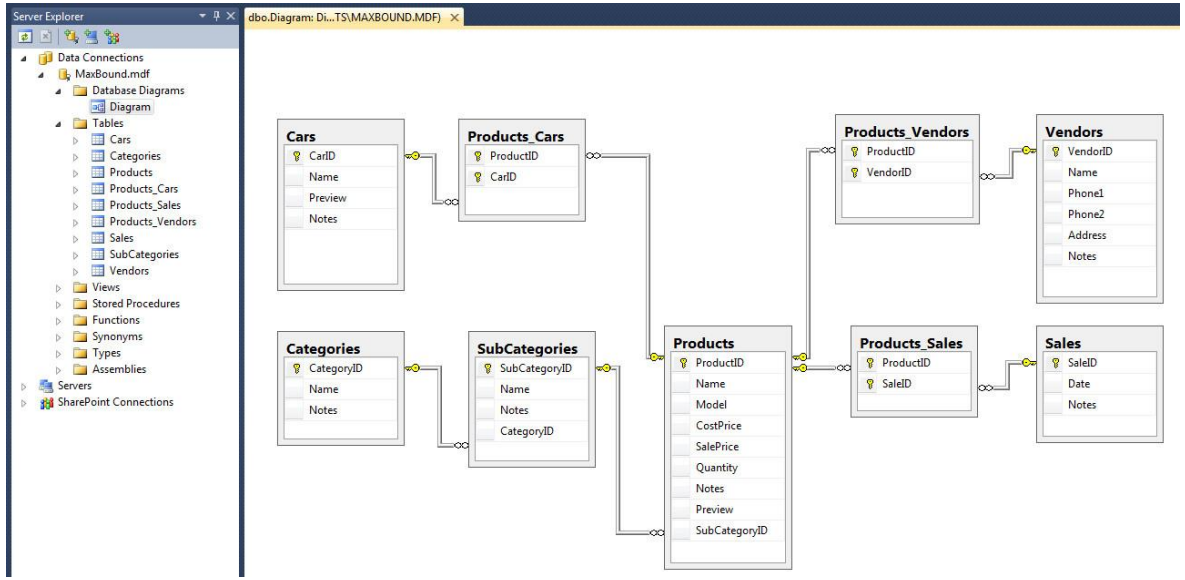
Martin Macák

Fakulta informatiky, Masarykova univerzita, Brno

FACULTAS ARTIS INFORMATICÆ • UNIVERSITAS MASARYKIANA •

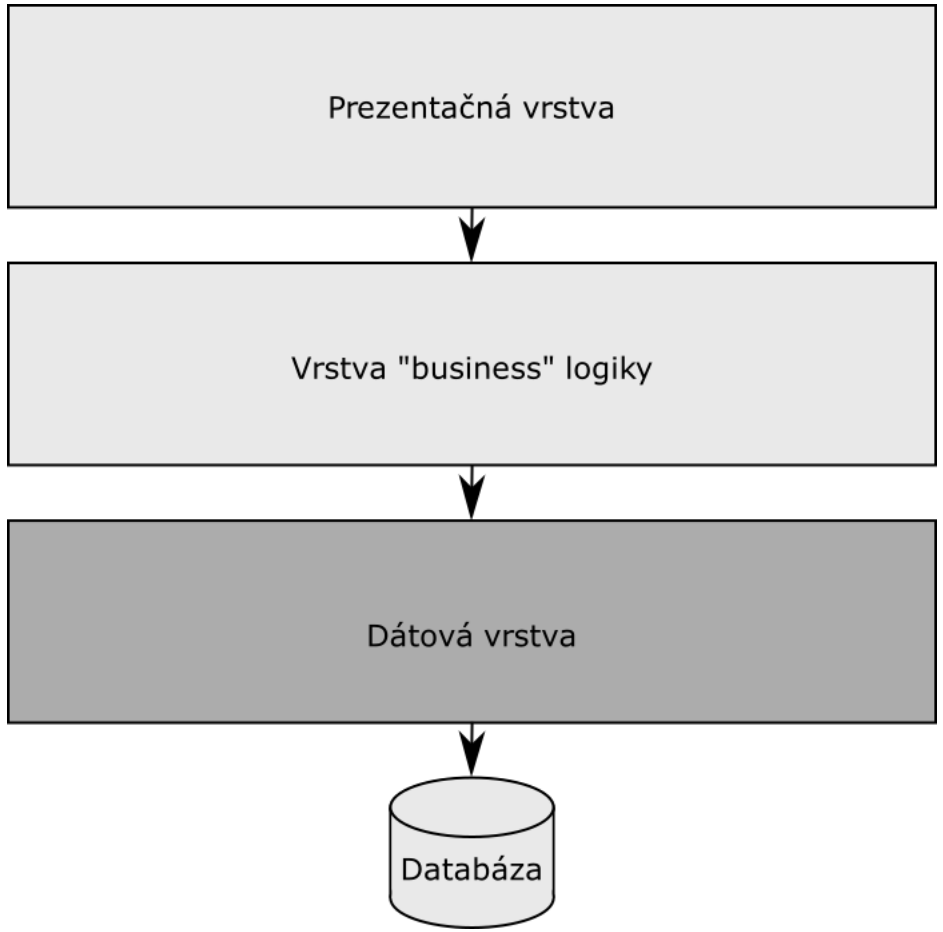
Osnova prednášky

- Úvod do DAL architektúry
- Dotazovanie a modifikácia dát
- Micro ORMs



MUNI FI

V minulej časti ste videli...



MUNI FI

V minulej časti ste videli...

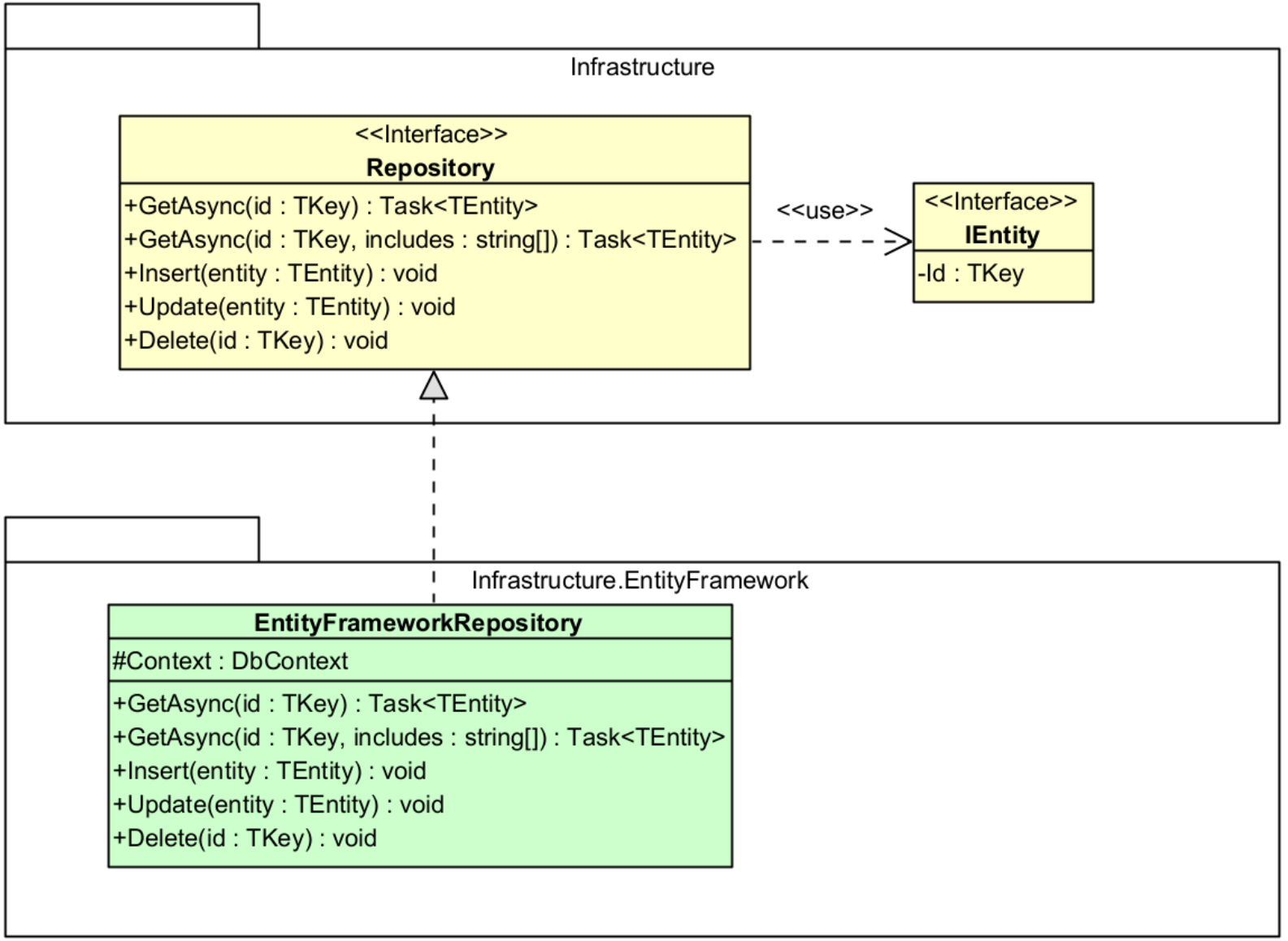
- Entity Framework
  - Entity
  - DbContext – **Identity Map, Unit of Work, Lazy Loading**
  - DbSety – **Repository**

MUNI FI

Čo nám ostáva? Príprava infraštruktúry

- Znovupoužiteľná časť kódu pre iné projekty
- Všeobecná časť
  - Nemá žiadnu závislosť na konkrétny ORM nástroj
  - Typicky abstraktné triedy, rozhrania alebo nejaká nemenná logika (napríklad predikáty)
  - Môže byť využívaná ako základný kameň pre hocijaké ORM
- Konkrétna časť
  - Plná závislosť na konkrétnom ORM
  - Triedy typicky využívajú elementy všeobecnej časti

Príklad častí infraštruktúry



MUNI FI

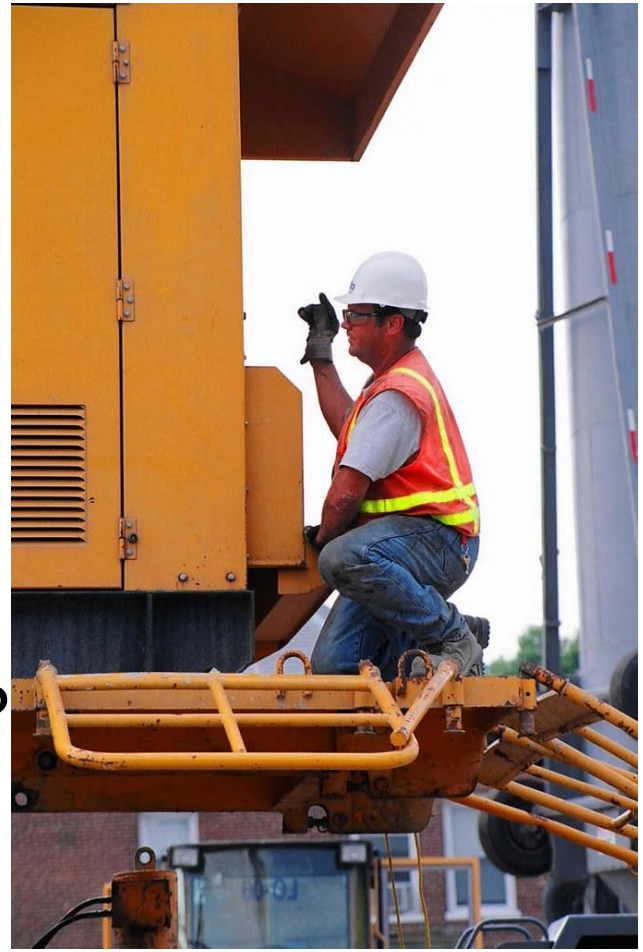
Návrhový vzor Repository

- Create, Read, Update, Delete

- Prečo robiť vlastný Repository, keď máme v EF DbSet?
  - Úroveň abstrakcie nad ORM frameworkom
  - Modifikácia dát na jednom mieste

Návrhový vzor Unit of Work

- Sleduje zmeny v entitách v priebehu danej transakcie
- Na záver transakcie sa zavolá commit
- Prečo robiť vlastný UoW, keď máme DbContext?
  - abstrakcia prístupu k dátam cez konkrétne ORM
  - prehľadnejšie transakcie



Dotazovanie a modifikácia dát

1. Čisté SQL
   - Databázovo závislé
   - Žiadna kontrola kompilátorom
   - Niekedy to inak nejde
2. LINQ to Entities
   - Podobný LINQ to Objects
   - IntelliSense
   - Kontrola kompilátorom
   - Nezvládne to úplne všetko

MUNI FI

Stavy entít

- Added – nie je v DB, sledovaná, má sa pridať
- Unchanged – je v DB, sledovaná, nezmenená
- Modified – je v DB, sledovaná, má sa zmeniť
- Deleted – je v DB, sledovaná, má sa zmazať
- Detached – nie je sledovaná

Nastavenie logovania SQL commandov

- Pre testovacie účely, ak vás zaujíma, čo sa na pozadí generuje
- Použijeme napríklad balík *Microsoft.Extensions.Logging.Console*

MUNI FI

Vyhodnotenie dotazu

- Na strane servera – dáta nie sú v pamäti, filtrovanie prebieha v databáze, typicky práca s *IQueryable*
- Na strane klienta – dáta sú v pamäti, filtrovanie prebieha na klientovi, typicky práca s *IEnumerable*

MUNI FI

Operácie v LINQ to Entities

- Select: metódy *ToList, First, Last, Find*, . . .
- Insert: metóda *Add, AddRange* alebo nastaviť stav *Added*
- Update: zmeniť danú property v tom istom DbContexte, metóda *Update, UpdateRange* alebo nastaviť stav *Modified*
- Delete: metóda *Remove, RemoveRange* alebo nastaviť stav *Deleted*

Pozor na nesúvislé spojenie!

- Dáta sú lokálne => žiadny problem
- Webová aplikácia => nemáme jeden, ten istý DbContext
  - Pošlem dáta zo servera, DbContext sa ukončí
  - Keď niečo zmením na klientovi, tak si vytvorím nový DbContext
  - Ten netuší, čo sa robilo pred tým, ako sa narodil
  - Je **vaša zodpovednosť** oznámiť mu relevantné stavy entít

MUNI FI

Práca s navigation properties

Predpokladajme, že máme entitu A, ktorá má ako navigačnú property List<B> (je s B vo vzťahu 1 : N)

- Insert
  - Ak pridávame do databázy B s nastavenou property typu A, pridá sa stav *Added* na B, ale automaticky aj na A
  - Toto nie je vždy to, čo chceme!
- Select
  - Pri vrátení entity A bude B štandardne **null**
  - Pre získanie B sa využívajú tieto techniky:

    Eager Loading – vráti všetky A hneď aj so všetkými B (metóda *Include*, prípadne projekcia)
    Explicit Loading – pre daný objekt A načíta všetky B (metóda Load)
    Lazy Loading – ak sa niekto dotáže na List<B> v danom objekte A, automaticky sa načíta (virtual + package EFCore.Proxies + UseLazyLoadingProxies v DbContexte)

Nemusí sa vám vyplatiť byť lazy

```
using (var db = new MyDbContext(connectionString))
{
    foreach (var team in db.Teams.ToList())
    {
        foreach (var stud in team.Students)
        {
            Console.WriteLine($"Name {stud.Name}, Team: {team.Name}");
        }
    }
}

using (var db = new MyDbContext(connectionString))
{
    foreach (var team in db.Teams.Include(t => t.Students).ToList())
    {
        foreach (var stud in team.Students)
        {
            Console.WriteLine($"Name {stud.Name}, Team: {team.Name}");
        }
    }
}
```

- V takomto prípade je najvhodnejšie použiť projekciu na Teams

Práca s navigation properties

Predpokladajme, že máme entitu A, ktorá má ako navigačnú property List<B> (je s B vo vzťahu 1 : N)

- Update
  - Ak nechcem entite B zmeniť property A, ale nejakú inú, nemusím načítať aj A
- Delete
  - Ak chcem zmazať entitu A, musím načítať aj entitu B, aby jej mohlo zmeniť cudzí kľúč (prípadne ju zmazať)

MicroORMs

- Rýchlejšie ORMs
- Jednoduché na použitie
- Obmedzená funkcionalita
- Napríklad:
  - Dapper
  - PetaPoco
  - Simple.Data

Raw SQL (db.Pets.First(p => p.Type == "dog")

```
// Raw SQL
using (SqlConnection con = new SqlConnection(connectionString))
{
    con.Open();
    using (SqlCommand command = new SqlCommand(
        "SELECT * FROM Pets WHERE Type='dog'", con))
    {
        using (SqlDataReader reader = command.ExecuteReader())
        {
            while (reader.Read())
            {
                var firstDog = (new Pet
                {
                    Id = reader.GetInt32(0),
                    Name = reader.GetString(1),
                    Type = reader.GetString(2),
                    Age = reader.GetInt32(3)
                });
            }
        }
    }
}
```


MUNI FI

MicroORMs ukážky

```
// Dapper
using (var db = new SqlConnection(connectionString))
{
    var firstDog = db.Query<Pet>
        ("SELECT * FROM Pets WHERE Type=@type", new { type = "dog" });
}

// PetaPoco
using(var db = new PetaPoco.Database("PetsConnectionString"))
{
    var firstDog = db.Query<Pet>
        ("SELECT * FROM Pets WHERE Type=@type", new { type = "dog" });
}

//Simple.Data
dynamic sDdb = Simple.Data.Database.OpenConnection(connectionString);

dynamic sDfirstDog = sDdb.Pets.FindByType("dog");
```


MUNI FI

Rekapitulácia

1. Návrhové vzory
   - Repository a Unit of Work
2. Dotazovanie a modifikácia dát
   - Možnosti
   - Stavy entít
   - Vyhodnotenie dotazu
   - LINQ to Entities
   - Práca s navigation properties
3. MicroORMs
   - Dapper
   - PetaPoco
   - Simple.Data

MUNI FI