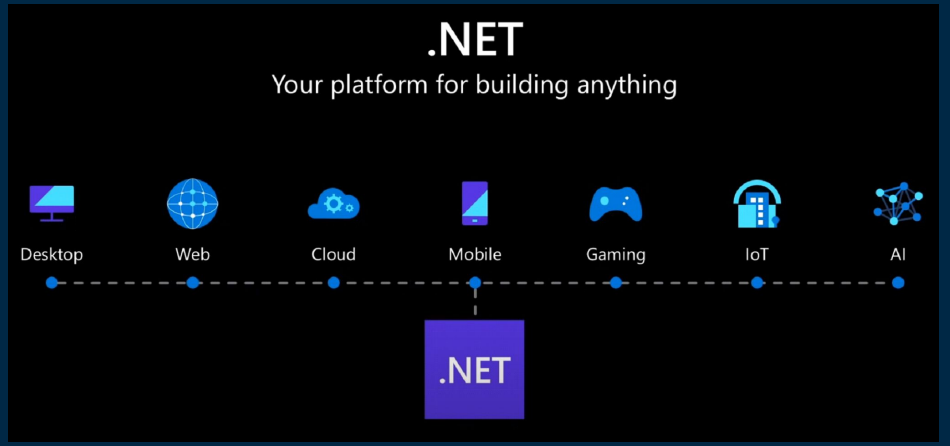
.NET Entity Framework VS NHibernate

Palo Litauszki
Andrej Urbanek
Lívia Cigániková
Adam Trebichalský

.NET

- Release 2016,
- Bezplatný, open-source a manažovaný softvér,
- Operačné systémy: Windows, Linux a macOS,
- Nástupca .NET Framework (release: 2002),
- Umožňuje vytvárať rôzne typy aplikácií.
- Skladá sa z:
  - .NET Framework (Windows od r. 2002)
  - Mono / Xamarin (Linux od r. 2004)
  - .NET Core (Windows, Linux, MacOS od r. 2016



Objektovo-relačné mapovanie

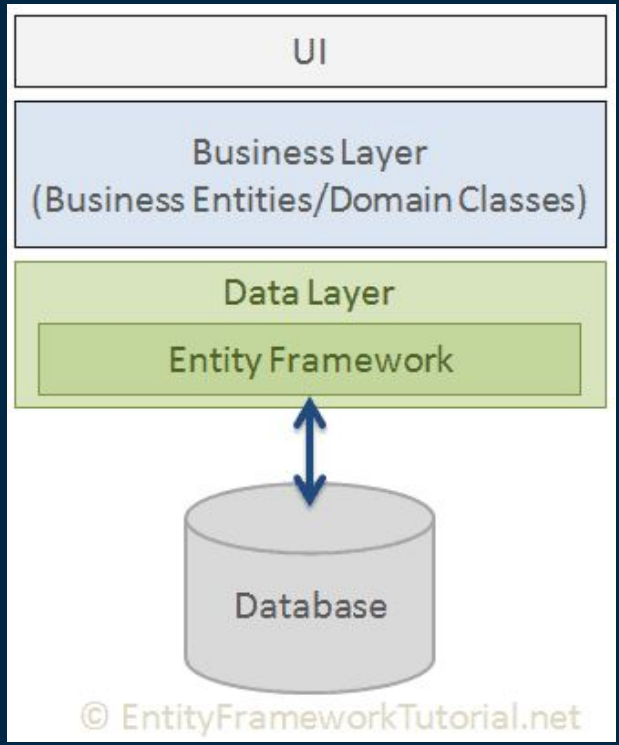
- Mapovanie objektov na riadky v relačných tabuľkách,
- Ľahší prístup k údajom,
- Aktualizovanie cez objektový model,
- Interagovanie s databázou programovacím jazykom namiesto SQL príkazov.

```
SELECT * FROM users WHERE email = 'test@test.com';
```

```
var orm = require('generic-orm-libarry');
var user = orm("users").where({ email: 'test@test.com' });
```

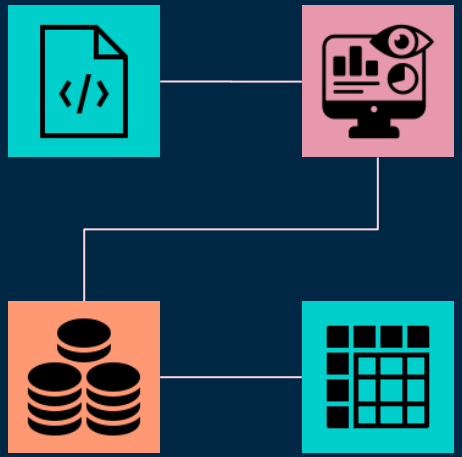
.NET Entity Framework

- Lightweight, rozšíriteľný, open-source ORM
- Cross-platformový (Windows OS, Linux OS, MacOS)
- Mapuje entity na objekty špecifické pre ich konkrétnu doménu
- Súčasť ADO.NET
- “Object–relational impedance mismatch”
- Vyššia úroveň abstrakcie



.NET Entity Framework – výhody

Poskytuje automaticky generovaný kód



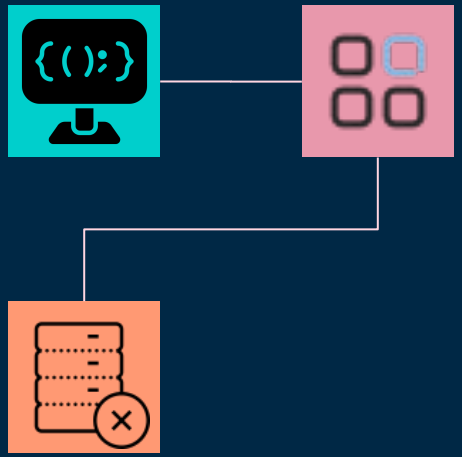
Umožňuje vývojárom vizuálne navrhovať modely a mapovať databázu

Umožňuje mapovanie viacerých koncepčných modelov do jednej schémy úložiska

Ľahké mapovanie business objektov (pomocou tabuliek a drag&drop)

.NET Entity Framework – nevýhody

Komplikovaná syntax



Logická schéma databázy nie je schopná využívať niektoré časti aplikácie

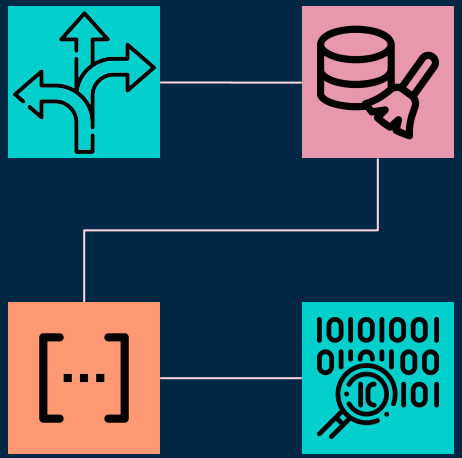
Nie je k dispozícii pre každý RDMS

NHibernate

- Release 2003
- Open source objektovo-relačný mapovač pre .NET. framework
- Bol dostupný ešte pred prvým uvedením Entity Frameworku
- Časť Hibernate z Javy
- Umožňuje mapovať doménové objekty do tradičných databáz

NHibernate – výhody

Flexibilné a veľmi bohaté možnosti mapovania.



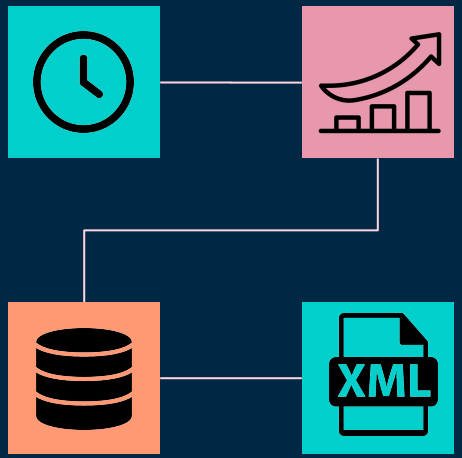
Podpora cacheovania druhej úrovne.

Podpora 6 druhov kolekcií (list,map,bag,set,array...)

NH podporuje primitive types a aj komponenty.

NHibernate – nevýhody

Dlhší čas spustenia v dôsledku prípravy metadát.



Vysoký učiaca krivka bez predchádzajúcej skúsenosti.

Vysoké nároky na DB schému.

Vyžaduje sa určitá konfigurácia XML súborov.

Vlastnosti Entity Framework a NHibernate

LINQ

01

- Zásadná súčasť frameworku - spôsob dopytovania do databázy
- Elegantný a rýchly prístup pre písanie queries nad objektami
- Súbor technológií založených na integrácii dotazovacích schopností priamo do jazyka C#

ENTITY SQL A HQL

02

- NHibernate aj EF definujú vlastné query jazyky, ktoré je možné použiť v situáciách, kedy LINQ alebo iné prístupy nestačia.
- Ide veľmi zjednodušene o dialekty jazyka SQL
- NHibernate - HQL - písanie zložitých queries, napríklad s kombináciami inner, outer a cross joinu
- EF - Entity SQ - možno použiť pre zložitejšie queries

Vlastnosti Entity Framework a NHibernate

LAZY FETCHING

03

- V NHibernate výrazne konfigurovateľný, vrátane načítania jednotlivých stĺpcov
- V EF je možné ho zapnúť/vypnúť globálne na celý model

DB PODPORA

04

- NHibernate - MS SQL Server, Oracle, DB2, MySQL, SQLite, PostgreSQL a Firebird
- EF - MS SQL, iné vyžadujú dodatočné pripojenia na ich spracovanie.

Vlastnosti Entity Framework a NHibernate

ORM MAPOVANIE

05

- Mapovanie pomocou XML
- Attribute mapping
- Mapovanie kódom
- Convention based mapping (Automapping)

DOKUMENTÁCIA

06

- NHibernate - nepatrí medzi najrozsiahlejšie, chýba popis niektorych funkcionalít
- EF - kvalitná , s veľkým množstvom informácií a návodov

Praktická časť

- Príprava
  - Visual Studio[1], .NET[2], Docker[3]
- Inštalácia (Windows, Visual Studio)
  - Vytvorenie a pripojenie projektu k databázovému serveru / súboru
  - Obnovenie verzií
  - Spustenie existujúcich riešení, následný troubleshooting
- Inštalácia (Linux Debian)
  - *sudo apt install dotnet*
  - Nové prostredie / link na hlavné .dll
  - Lokálna databáza v separátnom Docker kontajneri
- Testovanie
  - Rýchlosť komplikovaných queries, porovnanie štruktúry

```
FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build-env
WORKDIR /App

# Copy everything
COPY . ./
# Restore as distinct layers
RUN dotnet restore
# Build and publish a release
RUN dotnet publish -c Release -o out

# Build runtime image
FROM mcr.microsoft.com/dotnet/aspnet:6.0
WORKDIR /App
COPY --from=build-env /App/out .
ENTRYPOINT ["dotnet", "DotNet.Docker.dll"]
```

1 https://visualstudio.microsoft.com/, 2 https://dotnet.microsoft.com/en-us/, 3 https://www.docker.com/

Návrh riešenia

- Dostupné riešenia
  - Porovnanie rýchlosti pre zložité queries - https://github.com/dmcliver/NHibernateVsEf
  - Použitie .NET atribút namiesto XML - https://github.com/nhibernate/NHibernate.Mapping.Attributes
  - Alternatíva ku ORM modelu EF -https://github.com/codexguy/CodexMicroORM
  - Porovnanie celkového výkonu https://github.com/leotx/performance-test

Z týchto riešení sme analyzovali časť ich implementácií a pre konkrétne porovnanie vo výsledkoch sme vybrali riešenie z https://github.com/dmcliver/NHibernateVsEf.

Test výkonu

```
SELECT
[Project4].[C2] AS [C1],
[Project4].[Name] AS [Name],
[Project4].[C1] AS [C2]
FROM ( SELECT
        [GroupBy1].[A1] AS [C1],
        [GroupBy1].[K1] AS [Name],
        1 AS [C2]
        FROM ( SELECT
                [Extent1].[Name] AS [K1],
                COUNT(1) AS [A1]
                FROM  [dbo].[Artist] AS [Extent1]
                INNER JOIN [dbo].[Track] AS [Extent2] ON  EXISTS (SELECT
                        1 AS [C1]
                        FROM    ( SELECT 1 AS X ) AS [SingleRowTable1]
                        LEFT OUTER JOIN  (SELECT
                                [Extent3].[Id] AS [Id]
                                FROM [dbo].[Artist] AS [Extent3]
                                WHERE [Extent2].[ArtistId] = [Extent3].[Id] )
    AS [Project1] ON 1 = 1
                        LEFT OUTER JOIN  (SELECT
                                [Extent4].[Id] AS [Id]
                                FROM [dbo].[Artist] AS [Extent4]
                                WHERE [Extent2].[ArtistId] = [Extent4].[Id] )
    AS [Project2] ON 1 = 1
                        WHERE [Extent1].[Id] = [Project1].[Id]
                )
                GROUP BY [Extent1].[Name]
        )  AS [GroupBy1]
)  AS [Project4]
ORDER BY [Project4].[C1] DESC
```

Obr. 7: Vzorový SELECT pre EF ,(vzor je upravený pre zachovanie jednoduchosti a prehľadnosti
)

```
SELECT TOP (1)  a.Name as ArtistName, count(t.Id) as TrackCount
FROM Track t
INNER JOIN Artist a ON t.ArtistId = a.Id
GROUP BY a.Name
ORDER BY COUNT(t.Id) DESC
```

Obr. 8: Vzorový SELECT pre NHibernate

Číslo pokusu	*Čas spustenia*	*NHibernate*	*EF*
1	3ms	98	96
2	4.5ms	251	245
3	9ms	449	440

Tabuľka 2: Tabuľka pokusu pre SELECT z výpisov 7 a 8

*Databáza bola naplnená vzorkami z http://ocelma.net/MusicRecommendationDataset/lastfm-1K.html

Zhodnotenie

EF

- pomerne ľahko sa s ním pracuje
- vhodný pri použití MS SQL Server
- má menšiu krivku učenia
- lepší pre menšie projekty
- rýchlo sa vyvíja, väčšinu slabín je možné s určitým navýšením pracnosti úspešne riešiť

NHibernate

- dlhú dobu používaný mnohými vývojármi
- vhodné v prípade inej DB než MS SQL Server
- veľké projekty, kde je zásadný výkon a škálovateľnosť

Otázka na skúšku

Aké sú rôzne typy mapovania pre ORM?