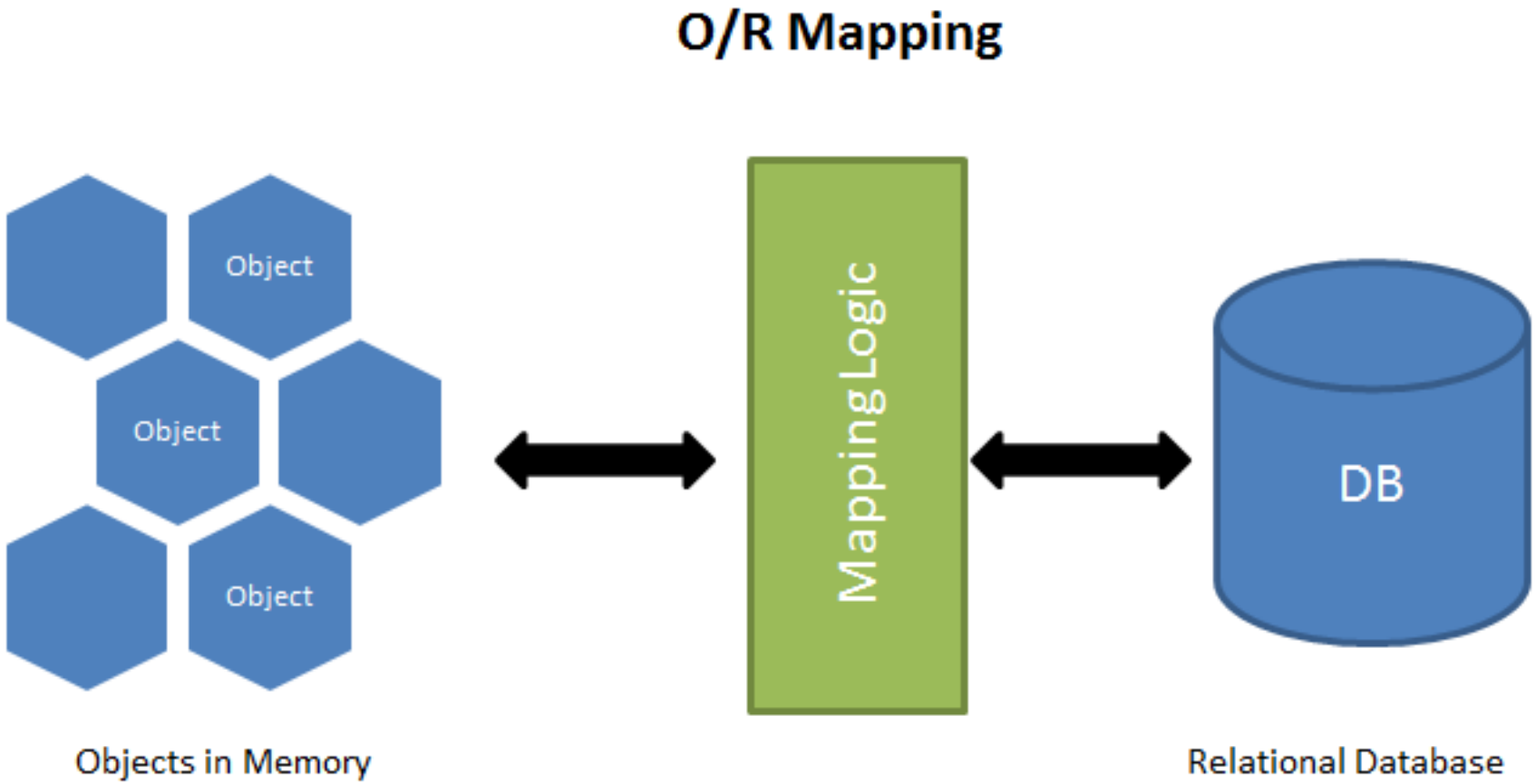
Entity Framework

Aleksandar Lukić

Šta je Entity Framework?

- Entity Framework je Objektno Relacioni Maper (ORM) - alat koji olakšava mapiranje objekata iz aplikacije na tabele i kolone u bazi podataka
- EF kreira konekcije ka bazi, izvršava komande i pretvara rezultete tih komandi u objekte
- EF takođe prati izmene nad objektima i na zahtev čuva te izmene u bazi
- Koristi ADO.NET za pristup bazi

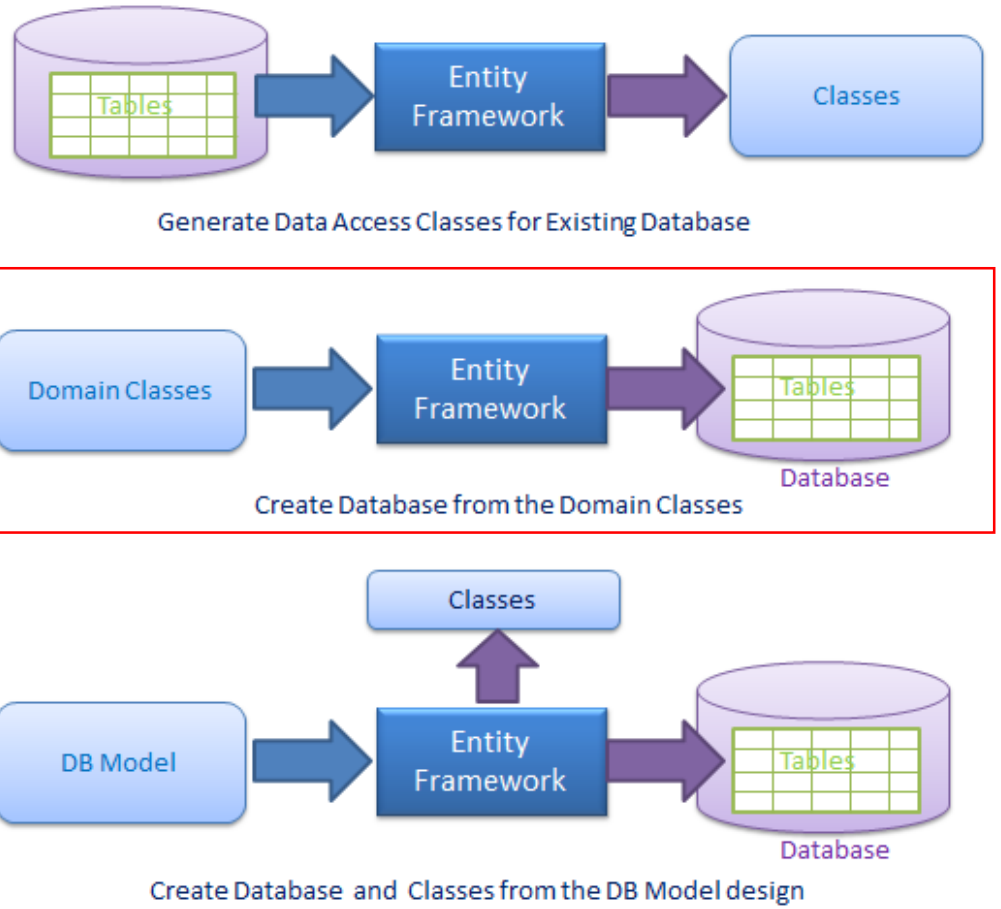
ORM


O/R Mapping
Object
Object
Object
Mapping Logic
DB
Objects in Memory
Relational Database

Procesi rada u EF

- Code-First: Prvo kreiramo klase, pa na osnovu njih kreiramo bazu podataka
- Database-First: Iz postojeće baze podataka, kreiramo klase u programu
- Model-First: Nekim alatom za modelovanje kreiramo model klasa i na osnovu njega se generišu klase i baza podataka

Procesi rada u EF



Code-First

- Počinjemo pisanjem naših domenskih klasa
- Prilikom pokretanja aplikacije EF kreira bazu (ukoliko ne postoji) i mapira domenske klase na tabele u bazi u skladu sa code-first konvencijama
- Konvencije možemo menjati putem Data anotacija i Fluent API-ja

DbContext

- DbContext je polazna klasa u EF koja se vezuje za jednu bazu podataka
- Ove klasa je nadležna za:
  - **Opis Entiteta:** Sadrži skup entiteta u vidu kolekcije DBSet<> klasa
  - **Marijalizaciju upita:** konvertuje LINQ-to-Entities upite u SQL upite i šalje ih u bazu
  - **Change Tracking:** Vodi računa o izmenama nad entitetima koje su nastale nakon njihovog učitavanja iz baze
  - **Perzistencija podataka:** Kreira CUD operacije u zavisnosti od stanja entiteta
  - **Kešira podatke:** Inicijalno radi first level caching. Za vreme svog životnog veka kešira sve učitane entitete iz baze
  - Rukuje vezama među tabelama
  - **Materijalizacija objekata:** Konvertuje raw tabelarne podatke u entitete

DbSet<TEntity>

- Mapira se na tabele u bazi
- Koristi se za CRUD operacije (**C**reate, **R**ead, **U**pdate, **D**elete)
- Pored osnovnih CRUD operacija sadži i operacije koje utiču na ponašanje DbContext-a

DbContext - primer

- U konstruktoru roditeljske klase **base()**
  - zadaje se naziv connection stringa, sa “name=“
  - zadaje se naziv baze, bez “name=”

```
public class ProductContext: DbContext
{
    public DbSet<Product> Products { get; set; }
    public DbSet<ProductCategory> ProductCategories { get; set; }

    public ProductContext():base("name=ProductContext")
    {

    }
}
```

Connection string

- Podatke o bazi dajemo u vidu connection stringa u App.Config ili Web.config datotekama

```
<connectionStrings>
    <add name="BookingAppContext"
    connectionString="Data Source=(localdb)\MSSQLLocalDB;
    Initial Catalog=BookingDB;
    Integrated Security=True;"
    providerName="System.Data.SqlClient" />
</connectionStrings>
```

DbContext - primer

- Kreiranje tabela u bazi i pristup njima omogućen je kroz polja DbContext klase. Ova polja su tipa DbSet<> i kao parametar im prosleđujemo našu domensku klasu. U ovom primeru Product i ProductCategory.

```
public class ProductContext: DbContext
{
    public DbSet<Product> Products { get; set; }
    public DbSet<ProductCategory> ProductCategories { get; set; }

    public ProductContext():base("name=ProductContext")
    {

    }
}
```

CRUD operacije

- Pomoću objekta DbContext klase pristupamo tabelama u bazi

```
ProductContext db = new ProductContext();
var products = db.Products;
```

- Učitavanje svih elemenata u listu radimo pomoću ToList() metode

```
products.ToList()
```

- Elemente pretražujemo po identifikatoru pomoću Find() metode

```
Product product = db.Products.Find(id);
```

- Nove elemente dodajemo putem Add() metode

```
db.Products.Add(product);
```

- Elemente brišemo putem Remove() metode

```
db.Products.Remove(product);
```

CRUD operacije

- Izmena elemenata je malo kompleksnija
- Pošto EF kešira (čuva u memoriji) sve objekte koje je kreirao u toku svog rada, izmena elementa bi zahtevala pronalaženje tog elementa u memoriji i nakon toga njegovu izmenu
- Šta ako do sada nismo učitali neki objekat ?
  - Morali bi smo da ga učitamo putem Find() metode
- Kako bi izbegli ovaj način rada EF nudi još jednu metodu

```
db.Entry(product).State = EntityState.Modified;
```

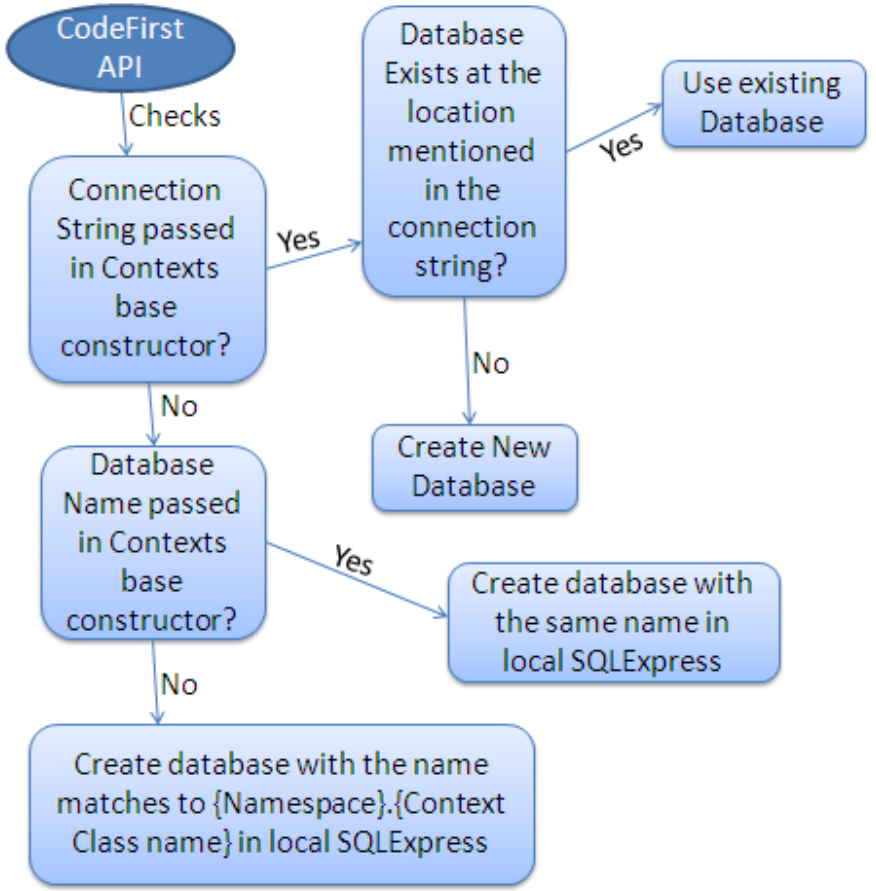
  - Na ovaj način kažemo EF - u da želimo da izmenimo neki objekat
  - Uslov: Identifikator tog objekta mora biti postojeći identikikator u bazi

CRUD operacije

- Kako bismo rezultat izvršavanja CUD operacija primenili na bazu pozivamo SaveChanges() metodu

```
db.SaveChanges();
```

Inicijalizacija baze



Code-First konvencije

- Mapiranje klasa na tabele u bazi:
  1. Za svaku domensku klasu koja se nalazi u DbSet<> klasi, kreiraće se nova tabela u bazi
  2. Za klase koje su sadržane u klasama iz 1, biće kreirane tabele u bazi
  3. Za podklase klasa iz 1, biće kreirane tabele u bazi

Code-First konvencije

Primarni ključ	EF kreira primarni ključ od atributa klase koji se zove Id ili <naziv klase>Id (nije case sensitive). Atribut može biti bilo kog tipa. Ukoliko je atribut tipa int ili GUID, biće kreiran kao Identity kolona.
Veze	EF kreira veze na osnovu navigacionih atributa. Navigacioni atributi mogu biti objekti ili kolekcije objekata. EF za odgovarajuće navigacione atribute kreira strani ključ sa nazivom <NazivKlase>_<NazivPrimarnogKljuča>
Strani ključ	Pre nego što se napravi strani ključ sa nazivom <NazivKlase>_<NazivPrimarnogKljuča>, EF u istoj klasi pored navigacionog atributa pokuša da pronađe atribut sa nazivom istim kao i navigacionog atributa ali sa nastavkom Id. Ukoliko pronađe takav atribut, naziv stranog ključa će biti isti kao i naziv tog atributa.

Code-First konvencije

Null kolona	EF kreira nullable kolone za sve objekte i primitivne tipove koji mogu imati null vrednost.
Not Null kolona	EF kreira NotNull kolone za primarne ključeve, i non-nullable primitivne tipove.
Mapiranje atributa na kolone	Po konvenciji svi atributi će biti mapirani na kolone u bazi, koristiti atribut [NotMapped] kako bi sprečili kreiranje kolone
Kaskadno brisanje	Inicijalno omogućeno za sve tipove veza.

Code-First konvencije

C# DataType	Related DB Column DataType	PK Column DataType & Length
int	int	int, Identity column increment by 1
string	nvarchar(Max)	nvarchar(128)
decimal	decimal(18,2)	decimal(18,2)
float	real	real
byte[]	varbinary(Max)	varbinary(128)
datetime	datetime	datetime
bool	bit	bit
byte	tinyint	tinyint
short	smallint	smallint
long	bigint	bigint
double	float	float
char	No mapping	No mapping
sbyte	No mapping (throws exception)	No mapping
object	No mapping	No mapping

Data anotacije

Attribute	Description
Key	Označava atribut kao EntityKey koji će biti mapiran na primarni ključ tabele
Required	Označava da atribut mora imati vrednost
MinLength	Zadaje minimalnu dužinu stringu ili nizu
MaxLength	Zadaje maksimalnu dužinu stringu ili nizu
StringLength	Zadaje minimalnu i maksimalnu dužinu stringa

[Key]

- Označava da će atribut klase biti primarni ključ tabele

```
[Key]
public string Name { get; set; }
```

Data anotacije

Attribute	Description
Table	Zadaje naziv tabele na koji će se mapirati klasa
Column	Zadaje naziv i tip kolone na koje će se mapirati atribut
Index	Kreira indeks od kolone
ForeignKey	Definiše strani ključ za navigacioni atribut
NotMapped	Sprečava mapiranje atributa na kolonu u tabeli
DatabaseGenerated	Koristi se za kreiranje Identity i Computed kolona u tabeli
InverseProperty	Koristi se kada imamo više veza između istih klasa

[Table]

- Zadaje naziv tabele u bazi na koju se mapira ova klasa
- Opciono se može zadati i šema kojoj tabela pripada

```
[Table("Products", Schema = "dbo")]
public class Product
```

[Column]

- Zadaje naziv koloni tabele
- Opciono se može zadati redosled kolone u tabeli
- Opciono se može zadati i tip kolone

```
[Column("ProductPrice")]
public decimal Price { get; set; }
```

```
[Column("ProductPrice", TypeName = "decimal", Order = 2)]
public decimal Price { get; set; }
```

[ForeignKey]

- Definiše strani ključ za navigacioni atribut
- Naredna dva primera su ekvivalentna

```
[ForeignKey("ProductCategory")]
public int ProductCategoryId { get; set; }

public ProductCategory ProductCategory { get; set; }
```

```
public int ProductCategoryId { get; set; }

[ForeignKey("ProductCategoryId")]
public ProductCategory ProductCategory { get; set; }
```

Migracije

- Prilikom razvoja novih aplikacija, model podataka se često menja
- Kada izmenimo model, njegova reprezentacija u bazi će se razlikovati od novog stanja modela
- Migracije koristimo kako bi izjednačili model sa njegovom reprezentacijom u bazi
- Migracije nam omogućavaju verzionisanje šeme baze (kao Git)
- One pretvaraju naše izmene u modelu u odgovarajuće upite, pomoću kojih će se izmeniti šema baze

Migracije

- Migracije su opisane **DbMigration** klasom
- Svaka migracija ima **Up()** i **Down()** metode

```
public partial class EmptyMigration : DbMigration
{
    public override void Up()
    {
    }

    public override void Down()
    {
    }
}
```

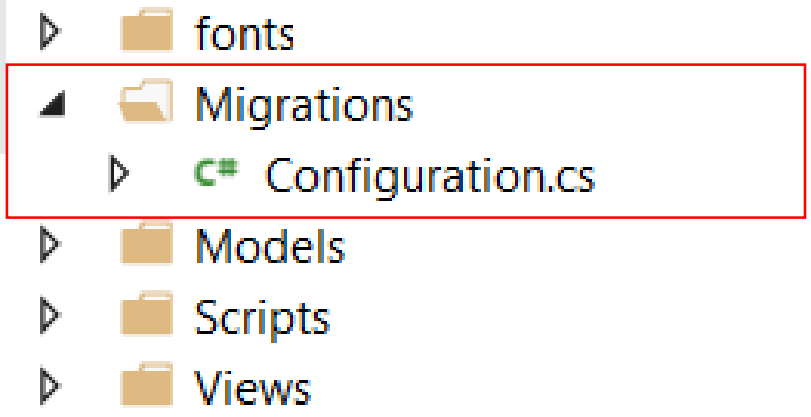
Migracije

- **Up()** metda se poziva kada radimo update baze (kreiramo novu verziju baze)
- **Down()** metoda se poziva kada radimo downgrade baze (vraćamo se na stariju verziju baze)

Rad sa migracijama

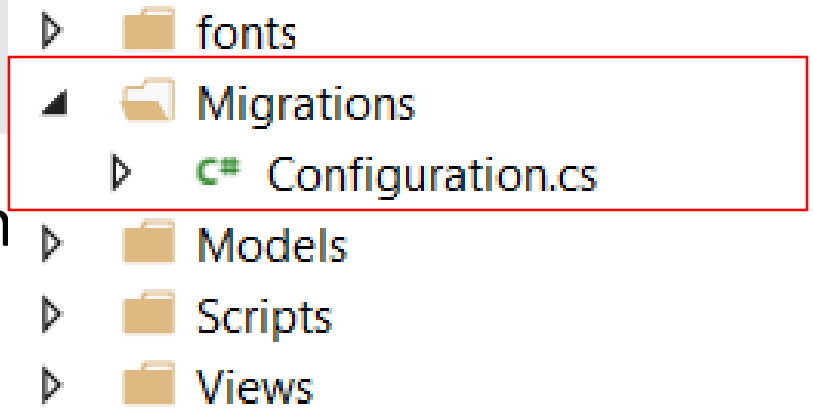
- **Korak 1** – Pre nego što pokrenemo aplikaciju moramo omogućiti migracije
- **Korak 2** – Otvoriti Package Manager Console u Tools → NuGet Package Manger → Package Manger Console.
- **Korak 3** – Uneti komandu **enable-migrations**

```
PM> enable-migrations
Checking if the context targets an existing database...
Code First Migrations enabled for project EntityFramework.
```



Rad sa migracijama

```
PM> enable-migrations
Checking if the context targets an existing database...
Code First Migrations enabled for project EntityFramework.
```



- Ova komanda kreira Migrations folder u kom

će se smeštati migracije

- Takođe se kreira klasa Configuration
  - Ima metodu Seed u kojoj možemo inicijalno dodati podatke u bazu

Rad sa migracijama

- **Korak 4** – Dodati migraciju pomoću komande **add-migration “<NazivMigracije>”.**
- **Korak 5 –** Primeniti migracije na bazu komandom **update-database**

Rad sa migracijama - primer

- Dodali smo domensku klasu Product

```
public class Product
{
    public int Id { get; set; }
    public string Name { get; set; }
    public decimal Price { get; set; }
}
```

- Želimo da naša baza podataka sadrži tabelu Products, koju opisuje ova klasa
- Kreiramo migraciju sa **add-migration “AddedProduct”** komandom
  - Naziv migracije je proizvoljan, ali poželjno je da ima neku semantiku

Rad sa migracijama - primer

```
public partial class AddedProduct : DbMigration
{
    public override void Up()
    {
        CreateTable(
            "dbo.Products",
            c => new
                {
                    Id = c.Int(nullable: false, identity: true),
                    Name = c.String(),
                    Price = c.Decimal(nullable: false, precision: 18, scale: 2),
                })
            .PrimaryKey(t => t.Id);

    }

    public override void Down()
    {
        DropTable("dbo.Products");
    }
}
```

Rad sa migracijama - primer

- Pošto smo dodali novu klasu, u **Up()** metodi kreira se nova tabela u bazi putem **CreateTable()** funkcije
- Atributi klase su mapirani na kolone u bazi u skladu sa konvencijama

```
CreateTable(
    "dbo.Products",
    c => new
        {
            Id = c.Int(nullable: false, identity: true),
            Name = c.String(),
            Price = c.Decimal(nullable: false, precision: 18, scale: 2),
        })
    .PrimaryKey(t => t.Id);
```

Rad sa migracijama - primer

- Pošto smo dodali novu klasu, ne želimo da se ona nađe u starijim verzijama baze, pa se zbog toga u **Down()** metodi poziva metoda **DropTable()**, kojom se briše tabela iz baze

```
public override void Down()
{
    DropTable("dbo.Products");
}
```

Dodatno

- Fluent API - http://www.entityframeworktutorial.net/code-first/fluent-api-in-code-first.aspx
- Eager vs Lazy vs Explicit loading: https://msdn.microsoft.com/en-us/library/jj574232(v=vs.113).aspx

Proverite vaše znanje

1. Šta je i čemu služi Entity Framework (EF)?
2. Šta su ORM maperi ?
3. Koji procesi rada postoje u EF ?
4. Šta je Code-First pristup ?
5. Čemu služi klasa DbContext ?
6. Na koji način vezujemo DbContext klasu sa connection string-om ?
7. Čemu služi klasa DbSet ?
8. Za koje će se sve klase kreirati tabela u bazi ?
9. Na koje sve načine možemo definisati primarni ključ ?
10. Na osnovu čega se prave veze među tabelama ?

Proverite vaše znanje

11. Na koji način kreiramo not null kolone ?
12. Na koji način možemo menjati konvencije kreiranja tabele ?
13. Na koji način možemo menjati konvencije kreiranja kolona ? Šta sve možemo menjati ?
14. Čemu služi [ForeignKey] atribut ?
15. Šta su i čemu služe migracije ?
16. Koje su prednosti korišćenja migracija ?
17. Šta je Eager, a šta Lazy loading ?