Introduction to .NET

Florin Olariu
“Alexandru Ioan Cuza”, University of Iași
Department of Computer Science

Entity Framework Core – part 2

Agenda

- Entity Framework Core – Short recap
- Entity Framework Core - Roadmap
- Entity Framework Core – Features
- Entity Framework Core – Database providers support
- Entity Framework Core - Building connection strings
- Entity Framework Core - Managing migrations
- Entity Framework Core - Managing navigation properties
- Entity Framework Core - Using loading patterns in EF Core

Entity Framework Core - Short recap

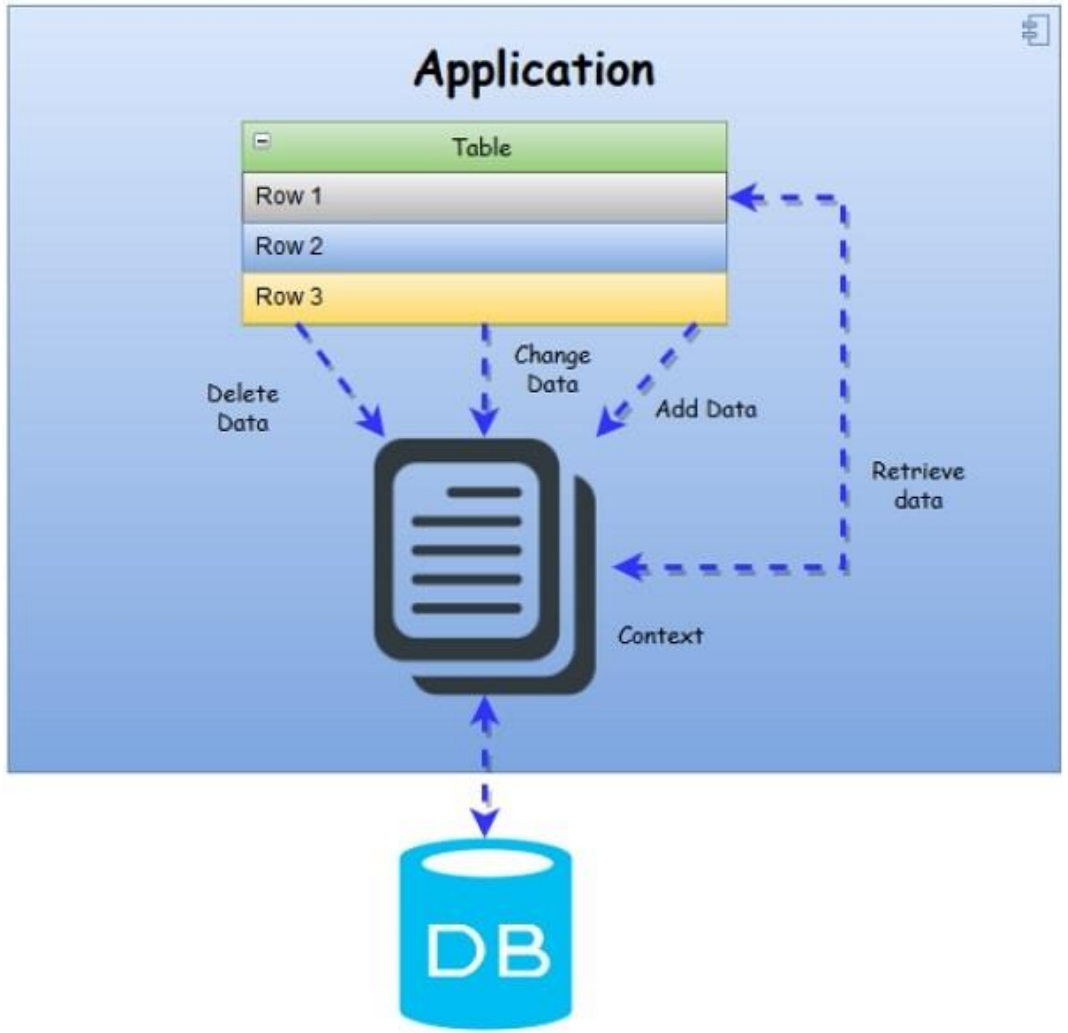
Entity Framework Core - Short recap

- Is a lightweight and extensible version of Entity Framework

Entity Framework Core - Short recap

- Is a lightweight and extensible version of Entity Framework
- It is based on ORM (Object Relational Mapper) which give us the possibility to work with databases using .NET objects

Entity Framework Core - Short recap



Entity Framework Core - Roadmap

Entity Framework Core - Roadmap

Schedule

The schedule of EF Core 2.2 will align with the schedule of .NET Core and ASP.NET Core 2.2. See the ASP.NET Core 2.2 Roadmap announcement.

Our current plan is to have three previews before we ship RTM near the end of 2018:

- August – Preview 1
- September - Preview 2
- October - Preview 3
- Before end-of-year – RTM

As usual, this post reflects our current plan, but things may change as we make progress.

Entity Framework Core – Features

Entity Framework Core – Features

- **Modeling**

Entity Framework Core – Features

- **Modeling**
  - Fluent API

Entity Framework Core – Features

- **Modeling**
  - Fluent API
  - Data annotations

Entity Framework Core – Features

- **Modeling**
  - Fluent API
  - Data annotations
  - Shadow properties

Entity Framework Core – Features

- **Modeling**
  - Fluent API
  - Data annotations
  - Shadow properties

```
var blogs = context.Blogs
    .OrderBy(b => EF.Property<DateTime>(b, "LastUpdated"));
```

Entity Framework Core – Features

- **Modeling**
  - Fluent API
  - Data annotations
  - Shadow properties
  - It maintains the relation between entities with help of navigation and foreign key properties

Entity Framework Core – Features

- **Modeling**
  - Fluent API
  - Data annotations
  - Shadow properties
  - It maintains the relation between entities with help of navigation and foreign key properties
- **Change Tracking**

Entity Framework Core – Features

- **Modeling**
  - Fluent API
  - Data annotations
  - Shadow properties
  - It maintains the relation between entities with help of navigation and foreign key properties
- **Change Tracking**
- **SaveChanges**

Entity Framework Core – Features

- **Modeling**
  - Fluent API
  - Data annotations
  - Shadow properties
  - It maintains the relation between entities with help of navigation and foreign key properties
- **Change Tracking**
- **SaveChanges**
- **Model validation**

Entity Framework Core – Features

- **Modeling**
  - Fluent API
  - Data annotations
  - Shadow properties
  - It maintains the relation between entities with help of navigation and foreign key properties
- **Change Tracking**
- **SaveChanges**
- **Model validation**
- **Query**

Entity Framework Core – Features

- https://www.codemag.com/article/1807071/Entity-Framework-Core-2.1-Heck-Yes-It%E2%80%99s-Production-Ready

Entity Framework Core – Database providers support

Entity Framework Core – Database providers support

- This version of Entity Framework Core 2.1 supports:

Entity Framework Core – Database providers support

- This version of Entity Framework Core 2.1 supports:
  - SQL Server

Entity Framework Core – Database providers support

- This version of Entity Framework Core 2.1 supports:
  - SQL Server
  - SQL Lite

Entity Framework Core – Database providers support

- This version of Entity Framework Core 2.1 supports:
  - SQL Server
  - SQL Lite
  - Postgres

Entity Framework Core – Database providers support

- This version of Entity Framework Core 2.1 supports:
  - SQL Server
  - SQL Lite
  - Postgres
  - SQL Compact

Entity Framework Core – Database providers support

- This version of Entity Framework Core 2.1 supports:
  - SQL Server
  - SQL Lite
  - Postgres
  - SQL Compact
  - MySQL (Official and Pomelo)

Entity Framework Core – Database providers support

- This version of Entity Framework Core 2.1 supports:
  - SQL Server
  - SQL Lite
  - Postgres
  - SQL Compact
  - MySQL (Official and Pomelo)
  - In Memory (for testing)

Entity Framework Core – Database providers support

- This version of Entity Framework Core 2.1 supports:
  - SQL Server
  - SQL Lite
  - Postgres
  - SQL Compact
  - MySQL (Official and Pomelo)
  - In Memory (for testing)
  - Devart(MySQL, Oracle, PostgreSQL, SqlLite, DB2, Cloud apps)

Entity Framework Core – Database providers support

- This version of Entity Framework Core 2.1 supports:
  - SQL Server
  - SQL Lite
  - Postgres
  - SQL Compact
  - MySQL (Official and Pomelo)
  - In Memory (for testing)
  - Devart(MySQL, Oracle, PostgreSQL, SqlLite, DB2, Cloud apps)
  - MyCat

Entity Framework Core - Building connection strings

Entity Framework Core - Building connection strings

```
4 references
public class EntityModelContext : DbContext
{
    2 references
    public DbSet<TestTable> TestTables { get; set; }
    0 references
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        optionsBuilder.UseSqlServer(@"Server=(localdb)\mssqllocaldb;Database=EntityModel;Trusted_Connection=True;");
    }
}
```

Entity Framework Core - Building connection strings

```
{
  "ConnectionStrings": {
    "EntityDatabase": "Server=(localdb)\\mssqllocaldb;Database=EntityModel;Trusted_Connection=True;"
  }
}
```

Entity Framework Core - Building connection strings

- For Postgre SQL :

```
{
  "ConnectionStrings": {
    "EntityModel": "User ID=admin;Password=1234%;Host=localhost;Port=5432;Database=damienbod;Pooling=true;"
  }
}
```

Entity Framework Core - Building connection strings

- For SQL local DB

```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
    optionsBuilder.UseSqlServer(@"Server=(localdb)\mssqllocaldb;Database=EntityModel;Trusted_Connection=True;");
}
```

Entity Framework Core - Managing migrations

Entity Framework Core - Managing migrations

- Enable-Migrations

```
PM> enable-migrations
Enable-Migrations is obsolete. Use Add-Migration to start using Migrations.
PM>
```

Entity Framework Core - Managing migrations

- Enable-Migrations
- Generating & Running Migrations

Entity Framework Core - Managing migrations

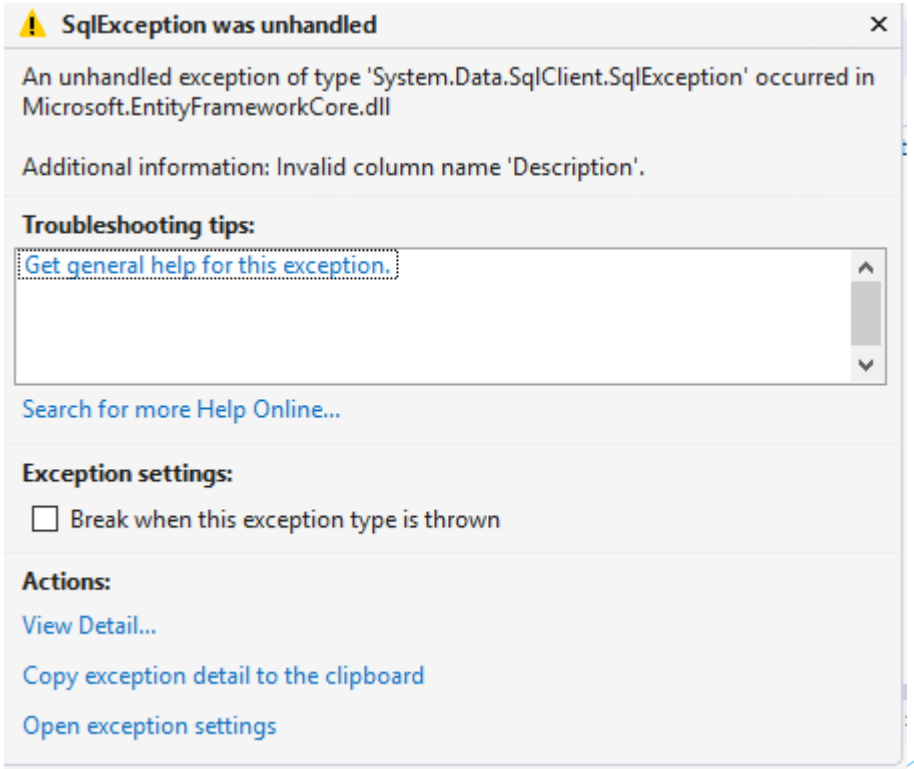
- Enable-Migrations
- Generating & Running Migrations
  - Add-Migration

Entity Framework Core - Managing migrations

- Enable-Migrations
- Generating & Running Migrations
  - Add-Migration
  - Update-Database

Entity Framework Core - Managing migrations

- Enable-Migrations
- Generating & Running Migrations



Entity Framework Core - Managing migrations

- Enable-Migrations
- Generating & Running Migrations

```
1 reference
public partial class Test2 : Migration
{
    1 reference
    protected override void Up(MigrationBuilder migrationBuilder)
    {
        migrationBuilder.AddColumn<string>(
            name: "Description",
            table: "TestTables",
            nullable: true);
    }

    1 reference
    protected override void Down(MigrationBuilder migrationBuilder)
    {
        migrationBuilder.DropColumn(
            name: "Description",
            table: "TestTables");
    }
}
```

Entity Framework Core - Managing migrations

- Enable-Migrations
- Generating & Running Migrations

```
1 reference
protected override void Up(MigrationBuilder migrationBuilder)
{
    migrationBuilder.AddColumn<string>(
        name: "Description",
        table: "TestTables",
        nullable: true);
    migrationBuilder.Sql("UPDATE TestTable SET Description = Name");
}
```

Entity Framework Core - Managing navigation properties

Entity Framework Core - Managing navigation properties

- One to One => 1:1

Entity Framework Core - Managing navigation properties

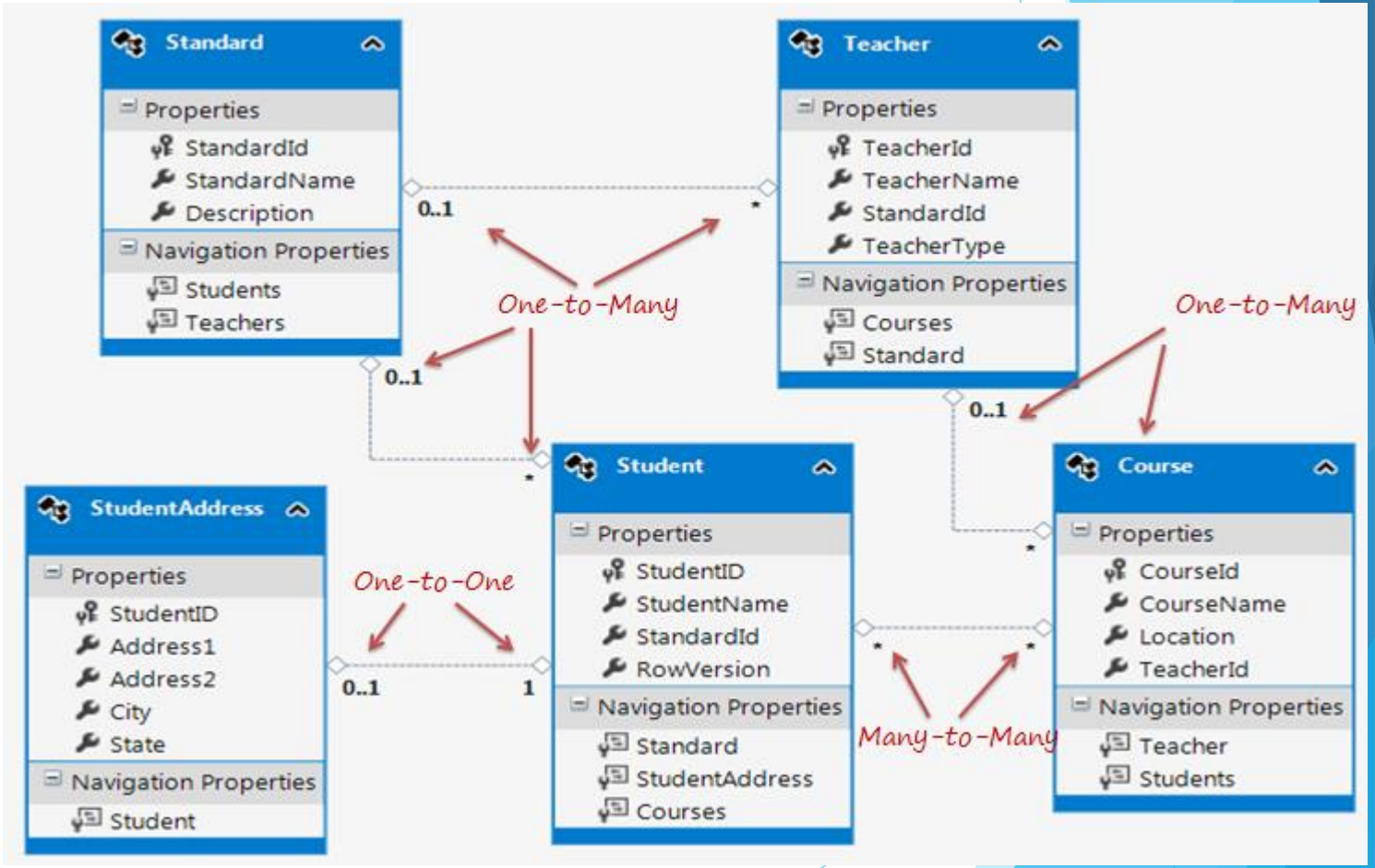
- One to One => 1:1
- One to Many => 1: many

Entity Framework Core - Managing navigation properties

- One to One => 1:1
- One to Many => 1: many
- Many to Many => many : many

Entity Framework Core - Managing navigation properties

- One to One => 1:1
- One to Many => 1: many
- Many to Many => many : many

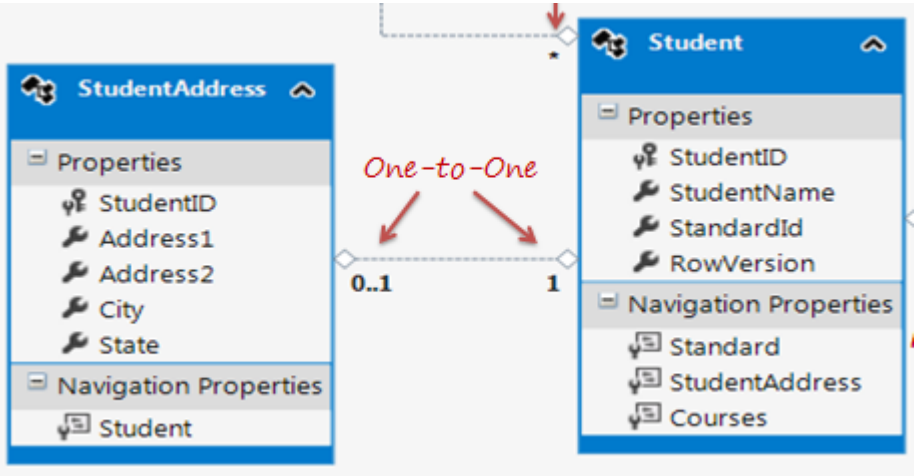


Entity Framework Core - Managing navigation properties

- One to One relationship

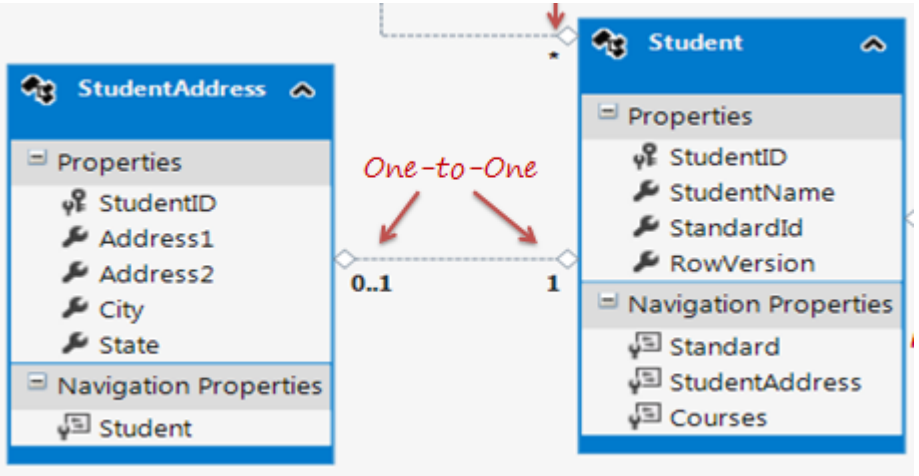
Entity Framework Core - Managing navigation properties

- One to One relationship



Entity Framework Core - Managing navigation properties

- One to One relationship



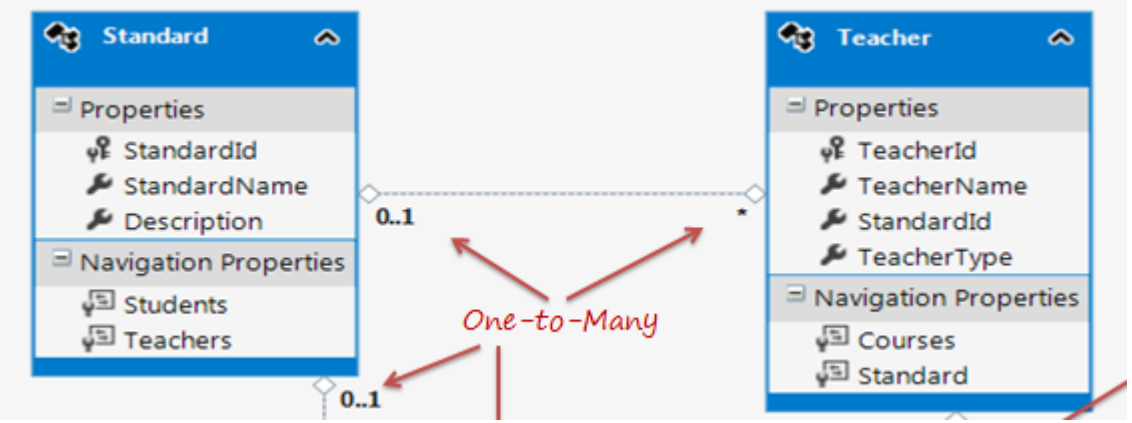

```
public class Student
{
    public Student()
    {
        this.Courses = new HashSet<Course>();
    }

    public int StudentID { get; set; }
    public string StudentName { get; set; }
    public Nullable<int> StandardId { get; set; }
    public byte[] RowVersion { get; set; }

    public virtual Standard Standard { get; set; }
    public virtual StudentAddress StudentAddress { get; set; }
    public virtual ICollection<Course> Courses { get; set; }
}

public class StudentAddress
{
    public int StudentID { get; set; }
    public string Address1 { get; set; }
    public string Address2 { get; set; }
    public string City { get; set; }
    public string State { get; set; }

    public virtual Student Student { get; set; }
}
```

Entity Framework Core - Managing navigation properties

- One-to-Many relationship

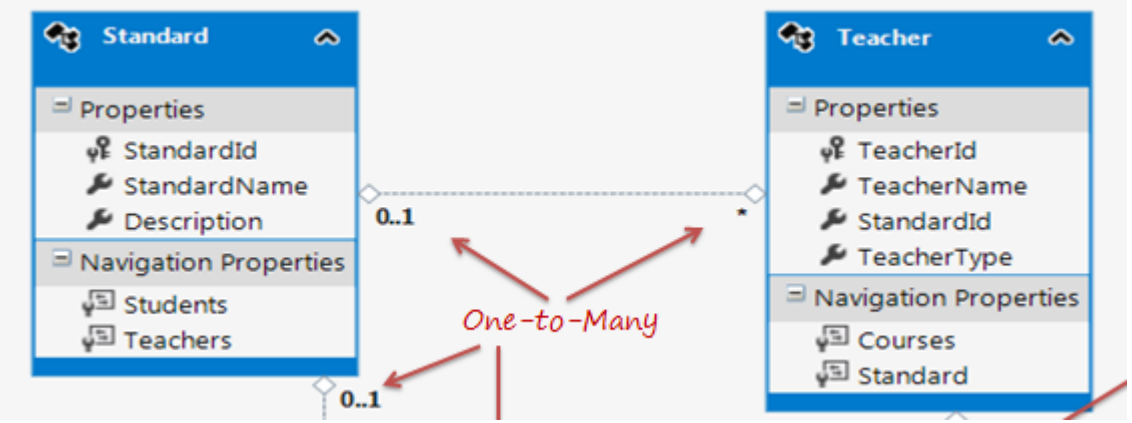
Entity Framework Core - Managing navigation properties

- One-to-Many relationship



Entity Framework Core - Managing navigation properties

- One-to-Many relationship



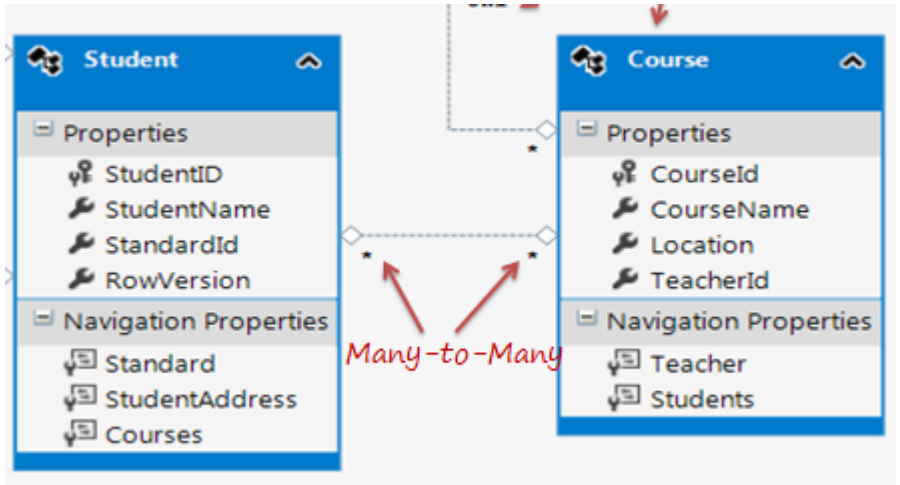

```
public class Standard
{
    public Standard()
    {
        this.Students = new HashSet<Student>();
        this.Teachers = new HashSet<Teacher>();
    }

    public int StandardId { get; set; }
    public string StandardName { get; set; }
    public string Description { get; set; }

    public virtual ICollection<Student> Students { get; set; }
    public virtual ICollection<Teacher> Teachers { get; set; }
}

public class Teacher
{
    public Teacher()
    {
        this.Courses = new HashSet<Course>();
    }

    public int TeacherId { get; set; }
    public string TeacherName { get; set; }
    public Nullable<int> StandardId { get; set; }
    public Nullable<int> TeacherType { get; set; }

    public virtual ICollection<Course> Courses { get; set; }

    public virtual Standard Standard { get; set; }
}
```

Entity Framework Core - Managing navigation properties

- Many-to-Many relationship

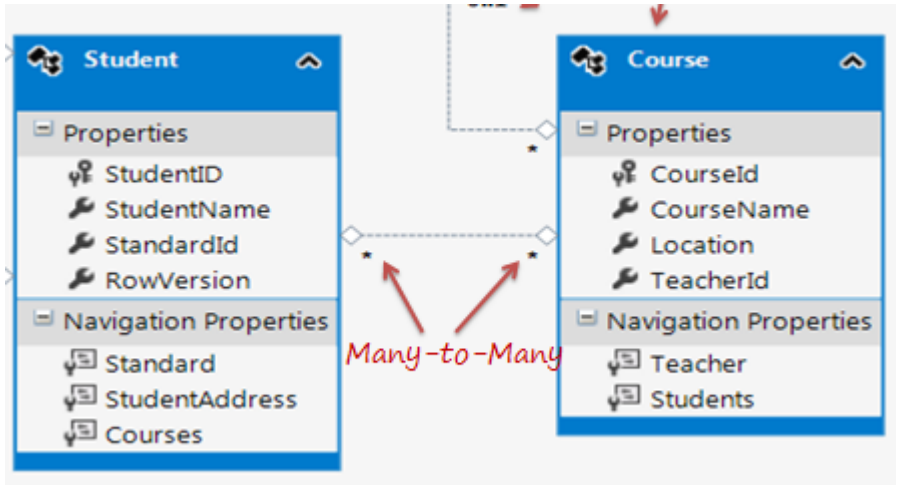
Entity Framework Core - Managing navigation properties

- Many-to-Many relationship



Entity Framework Core - Managing navigation properties

- Many-to-Many relationship




```
public class Student
{
    public Student()
    {
        this.Courses = new HashSet<Course>();
    }

    public int StudentID { get; set; }
    public string StudentName { get; set; }
    public Nullable<int> StandardId { get; set; }
    public byte[] RowVersion { get; set; }

    public virtual Standard Standard { get; set; }
    public virtual StudentAddress StudentAddress { get; set; }
    public virtual ICollection<Course> Courses { get; set; }
}

public class Course
{
    public Course()
    {
        this.Students = new HashSet<Student>();
    }

    public int CourseId { get; set; }
    public string CourseName { get; set; }
    public System.Data.Entity.Spatial.DbGeography Location { get; set; }
    public Nullable<int> TeacherId { get; set; }

    public virtual Teacher Teacher { get; set; }
    public virtual ICollection<Student> Students { get; set; }
}
```

Entity Framework Core - Using loading patterns in EF Core

Entity Framework Core - Using loading patterns in EF Core

- Lazy loading

Entity Framework Core - Using loading patterns in EF Core

- Lazy loading
- Eager loading

Entity Framework Core - Using loading patterns in EF Core

- Lazy loading
- Eager loading
- Explicit loading

Entity Framework Core - Using loading patterns in EF Core

- Lazy loading

Entity Framework Core - Using loading patterns in EF Core

- Lazy loading

```
var db = new Northwind();
db.Database.Log = new Action<string>(
message = >
{
    WriteLine( message);
});

var query = db.Categories;
foreach (var item in query)
{
    WriteLine( item.CategoryName);
}
```

Entity Framework Core - Using loading patterns in EF Core

- Lazy loading

```
var db = new Northwind();
db.Database.Log = new Action<string>(
message = >
{
    WriteLine( message);
});

var query = db.Categories;
foreach (var item in query)
{
    WriteLine( item.CategoryName);
}
```

```
Beverages
Condiments
Confections
Dairy Products
Grains/Cereals
Meat/Poultry
Produce
```

Entity Framework Core - Using loading patterns in EF Core

- Lazy loading

```
var db = new Northwind();
db.Database.Log = new Action<string>(
message = >
{
    WriteLine( message);
});

var query = db.Categories;
foreach (var item in query)
{
    WriteLine( item.CategoryName);
}
```

```
Beverages
Condiments
Confections
Dairy Products
Grains/Cereals
Meat/Poultry
Produce
```

- Let’s change the message from loop in order to show the products.

Entity Framework Core - Using loading patterns in EF Core

- Lazy loading

```
var db = new Northwind();
db.Database.Log = new Action<string>(
message = >
{
    WriteLine(message);
});

var query = db.Categories;
foreach (var item in query)
{
    // WriteLine(item.CategoryName);
    WriteLine($"{ item.CategoryName} has
    {item.Products.Count} products.");
}
```

Entity Framework Core - Using loading patterns in EF Core

- Lazy loading

```
var db = new Northwind();
db.Database.Log = new Action<string>(
message = >
{
    WriteLine(message);
});

var query = db.Categories;
foreach (var item in query)
{
    // WriteLine(item.CategoryName);
    WriteLine($"{ item.CategoryName} has
    {item.Products.Count} products.");
}
```

```
Beverages has 12 products.
Condiments has 12 products.
Confections has 13 products.
Dairy Products has 10 products.
Grains/ Cereals has 7 products.
Meat/ Poultry has 6 products.
Produce has 5 products.
Seafood has 12 products.
```

Entity Framework Core - Using loading patterns in EF Core

- Eager loading

Entity Framework Core - Using loading patterns in EF Core

- Eager loading

```
using (var context = new BloggingContext())
{
    var blogs = context.Blogs
        .Include(blog => blog.Posts)
        .ToList();
}
```

Entity Framework Core - Using loading patterns in EF Core

- Eager loading

```
using (var context = new BloggingContext())
{
    var blogs = context.Blogs
        .Include(blog => blog.Posts)
            .ThenInclude(post => post.Author)
            .ThenInclude(author => author.Photo)
        .Include(blog => blog.Owner)
            .ThenInclude(owner => owner.Photo)
        .ToList();
}
```

Entity Framework Core - Using loading patterns in EF Core

- Explicit loading

Entity Framework Core - Using loading patterns in EF Core

- Explicit loading

```
using (var context = new BloggingContext())
{
    var blog = context.Blogs
        .Single(b => b.BlogId == 1);
    context.Entry(blog)
        .Collection(b => b.Posts)
        .Load();
    context.Entry(blog)
        .Reference(b => b.Owner)
        .Load();
}
```

Entity Framework Core - Using loading patterns in EF Core

- Tips

Entity Framework Core - Using loading patterns in EF Core

- Tips
  - Carefully consider which loading pattern is best for your code.

Entity Framework Core - Using loading patterns in EF Core

- Tips
  - Carefully consider which loading pattern is best for your code.
  - The default of lazy loading can literally make you into a lazy database developer!

One more thing…(1/2)

One more thing...(2/2)

Figure 3 Average Time in Milliseconds to Execute a Query and Populate an Object Based on 25 Iterations, Eliminating the Fastest and Slowest

*AsNoTracking queries	Relationship	LINQ to EF*	EF Raw SQL*	Dapper Raw SQL
All Designers (30K rows)	–	96	98	77
All Designers with Products (30K rows)	1 : *	251	107	91
All Designers with Clients (30K rows)	* : *	255	106	63
All Designers with Contact (30K rows)	1 : 1	322	122	116

Bibliography

- Price, Mark. C# 6 and .NET Core 1.0: Modern Cross-Platform Development
- https://msdn.microsoft.com/en-us/library/jj591621(v=vs.113).aspx
- http://www.c-sharpcorner.com/article/introduction-to-entity-framework-core/
- https://docs.efproject.net/en/latest/platforms/full-dotnet/new-db.html
- https://www.codemag.com/article/1807071/Entity-Framework-Core-2.1-Heck-Yes-It%E2%80%99s-Production-Ready
- http://www.entityframeworktutorial.net/efcore/entity-framework-core.aspx

Questions

- Do you have any other questions?

Thanks!

See you next time! ☺