



NET2 - 13

Blandat: Enum, try/catch och mer EF

Förra gången

- LINQ
- Övningarna
 - EF-övningarna
 - LINQ-övningar
 - Lösningar: <https://www.w3resource.com/csharp-exercises/linq/index.php>

Idag

- Upprop
- Kursinventering
- Blandat
 - Enum
 - Try/Catch
 - Svenska tecken
- EF-Demo
 - Code First
 - Database First
 - Dapper
 - ...I samma projekt

Kursen

- Databasens grunder
- Utvecklingsmiljö: SSMS
- SQL Server
- SQL-språket
- Relationsdatabaser
- Datamodellering
- Normalisering
- Transaktioner (Vecka 2)
- Object-Relational Mapping (ORM)
- LINQ
- Asynkrona funktioner (Nästa lektion)
- Dapper
- Entity Framework
- Molnet: Azure och AWS (Nästa vecka)
- NoSQL: MongoDB (Vecka 2)

Enum

- En enum är en speciell "klass" som innehåller konstanter (oföränderliga/skrivskyddade variabler).
- Enum är en förkortning för "enumerations", vilket betyder "specifikt listad".
- En enum kan läggas fritt utanför en klass (innanför namespace), eller inne i valfri klass.
- Om du vill skapa en enum använder du nyckelordet enum (i stället för class eller interface) och avgränsar objekten med ett kommatecken

Exempel

```
enum Size
{
    Small,
    Medium,
    Large
}
```

Enum

- I din kod skriver du sedan:

```
Size size = Size.Medium;
```

- Detta används sedan såhär:

```
if(size == Size.Medium)
{
    Console.WriteLine("Du har valt medium");
}
```

Enum

- Vi kan också få ut ett numeriskt värde ur en enum:
- `enum Size`
- `{`
- `Small, // Alla Enums börjar på 0`
- `Medium, // 1`
- `Large // 2`
- `}`
- `int sizeInt = (int)Size.Large;`
- `Console.WriteLine(sizeInt); // Skriver ut 2`

Enums

- Vi kan dock ange egna siffror i vår enum:
- `enum Size`
- `{`
- `Small = 42,`
- `Medium = 46,`
- `Large = 50`
- `}`
- `int sizeInt = (int)Size.Large;`
- `Console.WriteLine(sizeInt); // Skriver ut 50`

Enum

- Det går också bra att bara ange ett startvärde:

```
public enum Months
{
    Januari = 1,
    Februari,
    Mars,
    April,
    Maj,
    Juni,
    Juli
}
```

Enum

- Loopa igenom en Enum

```
foreach (int i in Enum.GetValues(typeof(Months)))  
{  
    Console.WriteLine($"{Enum.GetName(typeof(Months), i)} = {i}");  
}
```

Enum | switch

```
enum Level
{
    Low, Medium, High
}

public static void ShowLevel()
{
    Level myVar = Level.Medium;
    switch (myVar)
    {
        case Level.Low:
            Console.WriteLine("Low level");
            break;
        case Level.Medium:
            Console.WriteLine("Medium level");
            break;
        case Level.High:
            Console.WriteLine("High level");
            break;
    }
}
```

Try/catch

- Ibland vill man inte att programkoden ska krascha då appen körs.
- Då kan man lägga till följande, som gör att koden inte stannar.

```
try
{
    // Kod som eventuellt kan krascha
}
catch(Exception e)
{
    Console.WriteLine("Koden smällde!");
    Console.WriteLine("Felmeddelandet: " + e.Message );
}
```

Code Along

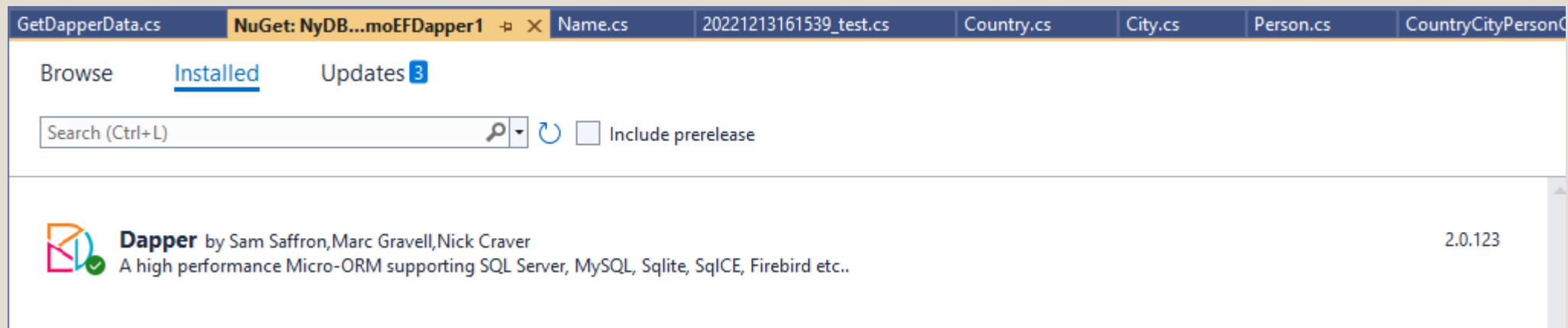
- Enums
- Try/catch

Svenska tecken

- Eftersom vi ibland har data som innehåller våra Svenska tecken ÅÄÖ så kommer LINQ att sortera dessa som A, A och O.
- Genom att lägga in följande rad i början av vår kod så löser vi detta:
- `Thread.CurrentThread.CurrentCulture = new System.Globalization.CultureInfo("sv-SE");`

Dapper

- Dapper och EF trivs bra ihop, och det går att kombinera dem.



Entity framework



Microsoft.EntityFrameworkCore by Microsoft

Entity Framework Core is a modern object-database mapper for .NET. It supports LINQ queries, change tracking, updates, and schema migrations. EF Core works with SQL Server, Azure SQL Database, SQLite, Azure Cosmos DB, MySQL, PostgreSQL, and other databases through a provider plugin API.

6.0.12

7.0.1



Microsoft.EntityFrameworkCore.SqlServer by Microsoft

Microsoft SQL Server database provider for Entity Framework Core.

6.0.12

7.0.1



Microsoft.EntityFrameworkCore.Tools by Microsoft

Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.

6.0.12

7.0.1

EF

- Database First:

- Packet Manager: Scaffold-DbContext
"Server=.\SQLEXPRESS;Database=**DBNamnet**;Trusted_Connection=True; "
Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models
- Använd dbContext och alla klasser/models som skapas, i din kod.

- Code First

- Programmera som vanligt, inklusive klasser/models
- Skapa en klass: MinDbContext.cs (exempel nästa sida)
- Packet Manager: Add-Migration **NuSkaparViDb**
- **Sen skriver vi: Update-database (Databasen skapas)**
- Databaserna skapas, och du lägger till **db.SaveChanges()** i din kod, där det behövs.

- Extra parametrar:

- Add-Migration InitialCreate **-Context MyDbContext -OutputDir Migrations\MyDbContext**

Exempel på MyDbContext.cs

```
internal class MyDbContext : DbContext
{
    public DbSet<Product> Products { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        optionsBuilder.UseSqlServer(@"Server=.\SQLEXPRESS; Database=Webshop;
Trusted_Connection=True;TrustServerCertificate=True;");
    }
}
```

Förra kursen: Properties: Virtual och override

```
internal class BasKlassen
```

```
{
```

```
    public virtual string Name { get; set; };
```

```
}
```

```
internal class SubKlassen : BasKlassen
```

```
{
```

```
    public override string Name { get; set; };
```

```
}
```

Relationer med EF

- En databas som har tabeller med relationer till varandra kan definieras i kod.
- Databas med Countries och Cities relaterar till varandra via CountryId.
- I Cities anges det genom en virtuell property, som EF använder:
 - `public virtual Country? Country { get; set; }`
- I Countries skapar vi en konstruktor, som skapar en kollektion (HashSet), som kan innehålla alla cities som hör till varje land.
- `public Country()`
- `{`
- `Cities = new HashSet<City>();`
- `}`
- Vi skapar också en property med en kollektion som innehåller alla städer.
 - `public virtual ICollection<City> Cities { get; set; }`
- Resten tar EF hand om, och overridear Country och Cities

Exempel City.cs

```
public partial class City
{
    public int Id { get; set; }
    public string? Name { get; set; }
    public int? CountryId { get; set; }

    public virtual Country? Country { get; set; }
}
```

Exempel Country.cs

```
public partial class Country
{
    public Country()
    {
        Cities = new HashSet<City>();
    }

    public int Id { get; set; }
    public string? Name { get; set; }

    public virtual ICollection<City> Cities { get; set; }
}
```

Code-along

- Kombinera database-first och code-first + Dapper
 - Ladda ner och kör CountryCity.sql

Länkar

- [C# Enums \(w3schools.com\)](https://www.w3schools.com/csharp/csharp_enums.asp)
- [How to loop through all enum values in C# \(c-sharpcorner.com\)](https://c-sharpcorner.com/2015/05/27/how-to-loop-through-all-enum-values-in-csharp/)
- [C# Exceptions \(Try..Catch\) \(w3schools.com\)](https://www.w3schools.com/csharp/csharp_exceptions.asp)
- [Navigation Property With Code First \(Navigation Property In EF\) \(c-sharpcorner.com\)](https://c-sharpcorner.com/2015/05/27/navigation-property-with-code-first-navigation-property-in-ef/)
- [Introduction to Relationships \(learnentityframeworkcore.com\)](https://learnentityframeworkcore.com/introduction-to-relationships/)
- [Entity Framework - Navigation Property Basics with Code First \(staticvoid.co.nz\)](https://staticvoid.co.nz/entity-framework/navigation-property-basics-with-code-first/)
- [entity framework - Why Navigation Properties are virtual by default in EF - Stack Overflow](https://stackoverflow.com/questions/2140080/entity-framework-why-navigation-properties-are-virtual-by-default-in-ef)

Övningar/Uppgift

- EF-övningarna
- LINQ-övningarna
- IDAG: Inlämningsuppgifterna