Entity Framework

Mål med lektionen!

- Förstå grunderna i LINQ

Vad lektionen omfattar

- Genomgång av Lab 2
- Kort repetition av pelarna i ett EF Projekt
  - Extension methods
  - Deferred Execution
- LINQ & LINQ med Lambda

Hjälp(Helper) metoder

- Förutom våra metoder som innehåller själva **grundläggande logiken** för vår applikation finns det även **annan typ av kod** *som vi behöver*
- Ett exempel på sån kod är **hjälper metoder**
- Hjälp-metoder är metoder som **ger oss funktionalitet/logik** som är *ofta återkommande*.
  - Detta kan **t ex** var *en beräkning* eller *en omvandling*
- Läggs ofta i en **Statisk klass** så att vi sedan kan **kalla på metoderna** i vår applikation **utan** *att skapa en instans av den*

Demo

Hjälp(Helper) metoder

Extension metoder

- Vi kan skapa *hjälp metoder* som **kopplas till en specifik typ**
- Vi behöver då i vår kod **inte ha med** den **statiska klassens namn** utan kan **kalla på den direkt** *från den typen den implementerats på*
- T ex. **istället** för att skiva **StringUtils.ToDouble(myString)** kan vi skriva **myString.ToDouble()**
- För att det ska fungera måste vi ha med *den statiska klassens namn* i ett **using statement** i vår klass om extension klassen inte är med i vårt namespace

Extension metoder *(forts)*

- Vår klass ska vara en **statisk klass**
  - Så att vi kan kalla på den utan att skapa en instans av den
- **Metoden** vi *skriver måste också* vara **statisk**
- Vi använder ***this*-nyckelordet** för att göra en metod till en extension metod.
- Metoden kommer att bli en extension-metod till den **första typen** *vi deklarerar i vår metod signatur*

```
public static class PersonExtensions
{
    public static int CalculateAge(this Person person) {
```

Demo

Extension metoder

IEnumerable

- IEnumerable är den som gör att vi kan göra en foreach-loop
  - Genom den metoden som heter GetEnumerator
- ICollection som ärver från IEnumerable ger ytterliggare funktionaliet
- IList som ärver av ICollection ger oss t ex Add metoden
- **List** -> **IList** -> **ICollection** -> **IEnumerable**
- **Linq** ger oss en uppsättning av **extension metoder** som vi kan *använda på IEnumerable*

Varför LINQ?

- Vi vill kunna **ställa frågor** till våra **Objekt/Listor/Arrays** för att få ut data
  - Objekten kan t ex vara en lista av strängar, en lista med Objekt eller kanske enititeter från Entity Framework
- För att göra detta kam man klassiskt skriv en **foreach/eller for-loop.**
- Vill vi ha vissa villkor får vi använda **if satser**
  - T ex *Bara personer vars namn startar på ”A”*
- Vill vi sedan t ex **sortera** vår resultat så får vi skriva *ytterligare kod*

Demo

Söka fram information från en Lista

LINQ

- Linq hälper oss att ställa de här frågorna genom att ge oss en **SQL liknande syntax**
- Vi slipper alltså skriva flera loopar/if-satser och mer kod för att sortera.
- **Vår kod** *blir enklare att läsa, tydligare, effektivare för oss som utvecklare och lättare att underhålla*.
- Linq är i princip en **uppsättning av Extension metoder** som vi kan **använda på IEnumerable**.
- För att kunna använda det behöver vi **referera in System.Linq namespacet**

Demo

Linq

LINQ med Lambda

- **Grundtanken** när Linq skapades var att koden vi skriver ska vara så nära som möjligt som **sättet vi pratar,** utrycker saker på.
  - **T ex** *från listan personer ger mig alla personer som börjar på A*

- Grunden i Linq gör att vi kommer närmare det men vi kan komma *ännu närmare* med **Linq och lambda**

- **Ex persons.Where(p => p.FirstName.StartsWith("A"))**

- Detta kan vi *jämföra* med **en foreach loop**

```
List<Person> personsStartingWithanA = new List<Person>();

foreach (Person person in persons)
{
    if (person.FirstName.StartsWith("A")) {
        personsStartingWithanA.Add(person);
    }
```

Demo

Linq med Lambda

Deferred Execution

- När vi skiver vår Linq fråga skriver vi vad som ska utföras. Vi har dock **INTE kört frågan ännu**.
- Frågan **körs** först när vi *använder* frågan
- Detta spelar stor roll i vad vi får tillbaka. Vi måste vara **försiktiga** så att vi får *tillbaka de resultat vi förväntar oss*
- Att frågorna körs först när vi kallar på dem kallas för **Deferred Execution**

Linq/Lambda de vanligaste **operatorerna**

- **.Where**
- **.OrderBy**
  - Sorts the elements in the collection based on specified fields in ascending or decending order.
- **.Skip**
  - Skips elements up to a specified position starting from the first element in a sequence.
- **.Take**
  - Takes elements up to a specified position starting from the first element in a sequence.
- **.First**
  - Returns the first element of a collection, or the first element that satisfies a condition.
- **.Last**
  - Returns the last element of a collection, or the last element that satisfies a condition
- **.Contains**
  - The Contains operator checks whether a specified element exists in the collection or not and returns a boolean

Lab 3