Entity Framework

Mål med lektionen!

- Lära oss interagera med Entity Framework i kod.

Vad lektionen omfattar

- Kort repetition av pelarna i ett EF Projekt

- CRUD mot Entity Framework
- Hur vi kan skriva över EF:s default Save metod

Extension metoder *(repetition)*

- 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

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*

Extension metoder *(forts) (repetition)*

- 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) {
```

LINQ med Lambda *(repetition)*

- **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);
    }
```

Deferred Execution *(repetition)*

- 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**

Separation of Concers

- En v*iktig del* av **Objektorientering**

- Vi vill skriva **snygg, lätthantering kod** som vi enkelt kan *uppdatera, läsa, ändra osv.*

- En klass är själv ansvarig för sig själv
- En metod gör **bara en sak**
- **Vi skiljer på UI och programlogik**

- Detta vi vi **tillämpa** som bra programmerare *oavsett om vi skriver ett litet eller ett stort program*

Arbeta mot EF modellen i kod

- **1.** Det första vi måste göra är **öppna** upp vårt **context**.
  - Det är här som vi arbetar mot vår databas
  - Saker som sker inom vårat context hjälper EF oss att hålla koll på.

- **2.** I vårt context så kan vi utföra olika arbeten
  - T ex: Lägga till, Ta bort och Ändra

- **3.** När vi gjort vad vi ska så kallar vi på SaveChanges() – som sparar ner det arbete som vi gjort

De vanligaste kommandona

- **.Add()**
  - Tar ett objekt som inparameter
- **.Remove()**
  - Tar ett objekt som inparameter
- **.Find()**
  - Tar ett id(int)som inparameter
- **.SaveChanges()**
  - Sparar ner de ändringar vi gjort

Demo

Arbeta mot EF modellen

Skriva över SaveChanges()

- **SaveChanges()** metoden kommer att **spara ner** vårt *arbete vi gjort mot Enity Framework*

- Metoden **kommer** dock att **misslyckas** om vi **försöker spara ner saker till databasen som inte Db:n kan spara ner** t ex pga felaktigt format
  - T ex ett DateTime.MinValue(). *.NETs min value är mycket lägre än vad som tillåts i en MS SQL Databas.*

- Vi kan då **skriva över SaveChanges()-metoden** för att tillföra *egen validering/funktionalitet mm.*

Demo

Skriva över SaveChanges()

Lab 4