Entity Framework

Mål med lektionen!

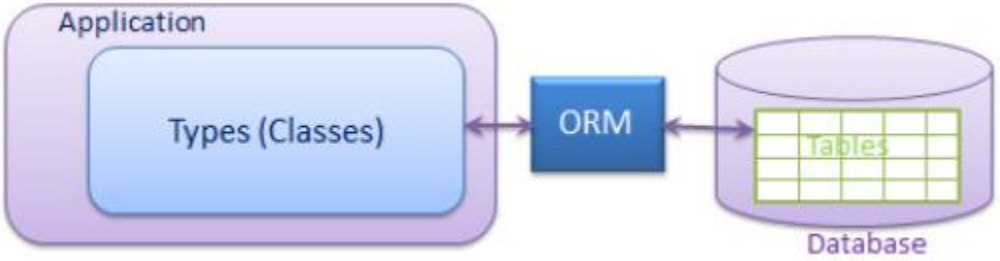
- Få ytterligare förståelse för vår modell

Vad lektionen omfattar

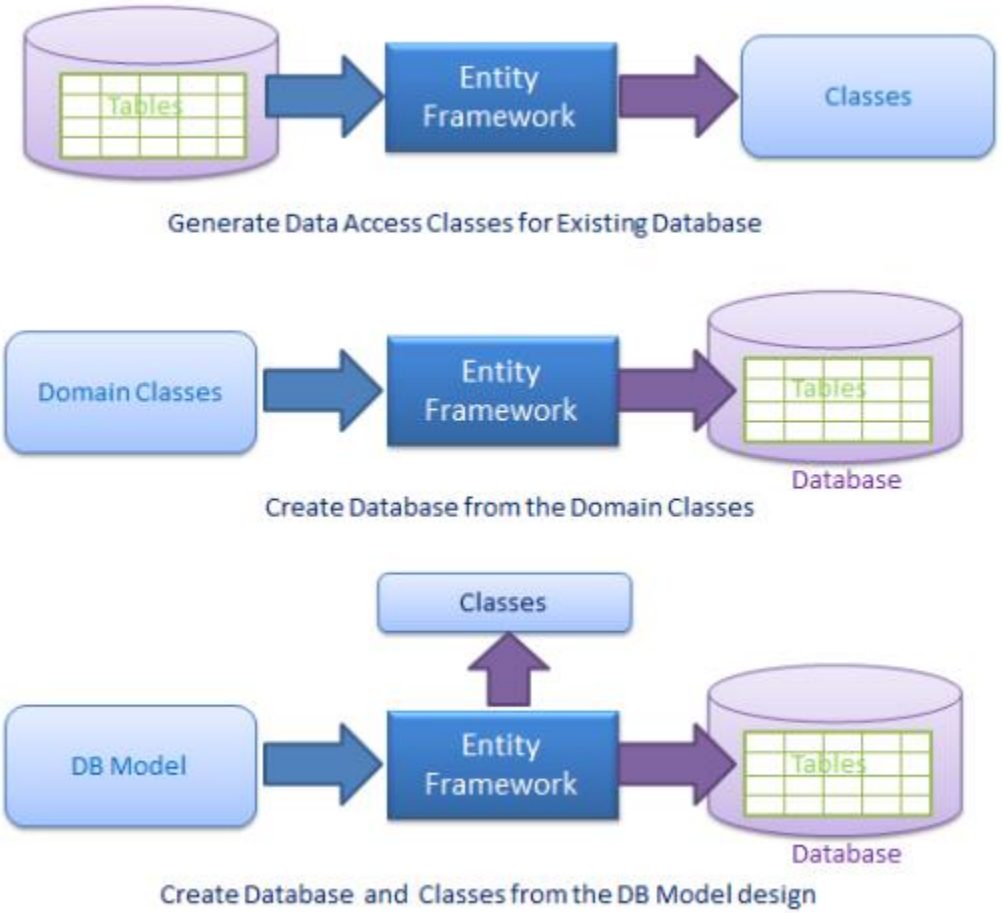
- Genomgång av Lab 1
- Repetition av grunderna i Entity Framework
- Se hur man kan få in ändringar från databasen till VS

Vad är **O/RM**? (forts).

Typisk ORM verktyg genererar klasser för databas interaktion för ditt program enligt nedan.



Vad är Entity Framework? (forts).



Olika **vägar** till Entity Framework?

- **Model to database** (depricated)
  - Vi bygger själva upp vår modell och utifrån den skapas databasen
- **Code first**
  - Vi skapar våra klasser och utfrån dem genereras databasen
- **Database first**
  - Utfrån en befintlig databas skapar vi vår model

Några **väsentliga delar** för vår Model *(forts)*

- **Refererens** till *Entity Framework*
- **Koppling** *till databasen*
- **Klasser** med **motsvarighet** *till vår db*
- En **Entitets klass** com **ärver från DbContext**

Några **väsentliga delar** för vår Model i VS *(forts)*

- **References**
  - Dessa kommer med när vi lägger till Entity Framework till vårt projekt i Visual Studio
- **App.config**
  - I filen som ligger på root-nivån i projektet sparas det som behövs för att ansluta till SQL servern.
- **Xxxx.edmx**
  - Vår model
  - **Xxxx.Context.tt**
    - **Klassen** som **ärver** *från DBContext*
    - *Är den som ger oss access till entiteterna (klasserna)*
  - **Xxxx.tt**
    - **Klasserna** som **genererats** *från databasen*

Hur hanterar vi **ändringar** i vår **Db**?

- Vi kan vilja **uppdatera** fält mm i vår Databas
- Dessa *ändringar* vill vi ska **reflekteras** i vår Databas
- Vi har **färdiga verktyg** i *Visual Studio* för **automatisera** detta
- Vi kan både **lägga till** *nya* tabeller och **uppdatera** *befintliga*

Hur hanterar vi **ändringar** i vår **Db** *(forts)*

- Kör alternativet ***Update model from database***
- **Bygg om projektet** för att ändringarna ska reflekteras i koden
- Om vi **ändrar typen** av *ett fält* i Databasen så kommer dessa **INTE** *att reflekteras i modellen*.
  - Fullt logiskt eftersom det kan ändra logiken i vårt program
  - Ex. Tänk t ex på ***string*** *som vi byter till* ***int***. Vi kan ha ha 50 olika ställen i vår kod som försöker använda just detta fältet som en sträng. **INGA** av dessa *skulle fungera om vi skulle göra den ändringen*.
  - Den ändringen måste vi göra **manuellt** *först i modellen* och *sedan uppdatera vår kod*

Demo

Uppdatera vår Db och få in ändringarna i vår model

Enums

- Är en distinkt typ i c# som består av ett **set(samling) av konstanter,** kallad enumerator listan
- Per **Default** så *startar* den på 0 och ökar med 1
- En Enum **kräver inte** att vi deklarerar en *siffra* **men vi kan** göra det.
- Den underliggande typen är per **default**: ***int***

Enums *(forts)*

- Exempel
  - enum Days {Sat, Sun, Mon, Tue, Wed, Thu, Fri};
  - enum Days {Sat=1, Sun, Mon, Tue, Wed, Thu, Fri};
  - int x = (int)Days.Sun;

```
int x = (int)Days.Sun;
int y = (int)Days.Fri;
Console.WriteLine("Sun = {0}", x);
Console.WriteLine("Fri = {0}", y);
```

Demo

Arbeta med Enums i EF Modellen

Lab 2