Entity Framework

Mål med lektionen!

- Veta kursmålen.

- Ha kännedom om några av de grundläggande begreppen.

Vem är **jag**?

- Mitt namn är Björn Jönsson och jobbar på Tahoe Solutions, ni når mig via **mail**: bjorn.jonsson@tahoesolutions.se
- Jag är Certified ScrumMaster sedan 2008 och har programmerat i +10 år. Jobbat mycket med C# och .Net, SQL, JavaScript och SharePoint.

Vilka moment **omfattar** kursen?

- Ramverket Entity Framework
- ADO.NET med Entity Framework som datakälla
- Mapping, Enitys, Simple Types, Complex types, Enums
- Olika sätt att skapa modeller på
  - Modell till database (depricated)
  - Databas till modell
  - Class first (inblick)
- LINQ Queries
  - Query syntax
  - Lamda syntax
- Uppdatering av klasser / databas och modell på ett säkert sätt
- Entity Framework med ASP.NET

Målet med kursen

Mål som den studerande ska ha uppnått efter avslutad kurs

Den studerande ska ha kunskaper och färdigheter i:

- Hantering av ramerket Entity Framework
- LINQ Queries och uppdatering av klasser/databas/modell

Vad krävs för **godkänt**?

- Har erhållit lägst betyget godkänt på tentamen
- Har klarat av alla obligatoriska moment
- Ska kunna skapa modeller genom codefirst, database first och model to database
- Kan formulera LINQ Queries
- kan använda Entity Framework tillsammans med ett webbprojekt
- Under praktiska tentamen klarar av att skapa ett webbprojekt som är styrt med entity framework

Vad krävs för **väl godkänt**?

- Uppnått kunskapskraven för betyget godkänt
- Har erhållit lägst betyget **väl godkänd** på tentamen
- Kan skriva och använda LINQ Queries obehindrat
- Kan hantera förändringar i modellen på ett säkert sätt
- Under praktisk tentamen behärska 2 olika sätt att skapa Entity Framework på

Vad skall vi **göra** på denna kursen?

- Anteckna gärna och framför allt fråga om det behövs!
- Följ gärna med i Slidarna på datorn

En **liten** innehållsförteckning

- Lektion 1 (Grundläggande begrepp, Model)
- Lektion 2 (Grundläggande begrepp, Model, forts)
- Lektion 3 (Linq)
- Lektion 4 (Arbeta mot Entity Framework med Linq, CRUD)
- Tillfälle (halvdag labb, projekt (**OBLIGATORISK NÄRVARO**))
- Lektion 5 (LINQ)
- Lektion 6 ()

- Tenta (Fredagen 11:nov dvs v45)

Vad lektionen omfattar

- Grundläggande begrepp och förklaringar

Vilka problem vill vi **lösa**?

- **Vi arbetar med Webbapplikationer**
- Vi kommer att behöva **läsa, ändra och lägga till ny data**
  - **CRUD**
- Det **vanligaste** sättet när vi arbetar med **.NET** kommer att vara en **SQL databas**
- Vi behöver alltså ha en **kommunikation** mellan vår **webbapplikation** och **databasen**(en eller flera)

Hur **kommunicerar** *vi med vår* **db**?

- För det första behöver vi ha en **koppling** *till databasen*
- För den andra behöver vi **prata med databasen** (CRUD)
- Koppling kan ske direkt till databasen men vanligare är ADO.NET som är ett O/RM verktyg
- Sen kan vi med hjälp av LINQ fråga databasen

```
IOrderedQueryable<Employee> hireQuery =
    from emp in db.Employees
    orderby emp.HireDate
    select emp;

foreach (Employee empObj in hireQuery)
{
    Console.WriteLine("EmpID = {0}, Date Hired = {1}",
        empObj.EmployeeID, empObj.HireDate);
}
```

Vad är **O/RM**?

- ORM är ett verktyg för att lagra data **från domänobjekt till relationsdatabaser** som MS SQL Server, på ett automatiserat sätt utan mycket programmering.

- O/RM innehåller tre huvuddelar
  - **Domän klassobjekt**,
  - **Relationsdatabasobjekt**
  - **Mapping information**

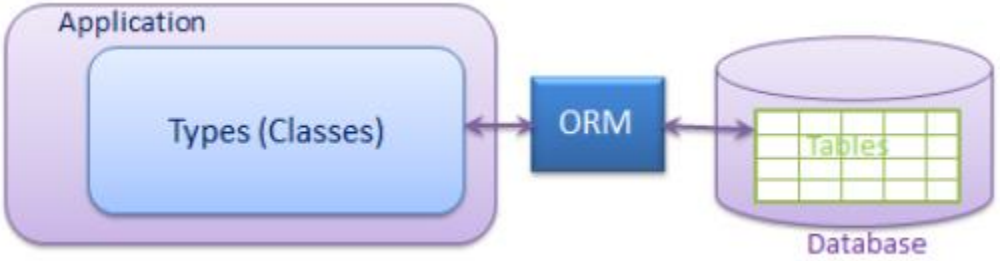
  om hur domänobjekt mappas till relationsdatabasobjekt (tabeller, vyer och storedprocedures).

- ORM låter oss hålla vår **databas design skild från vår domän klass design**.

- Detta gör programmet **lätt att underhålla och bygga ut**.
  - Det **automatiserar också standard CRUD operationer** (Skapa, läsa, uppdatera och radera) så att *utvecklaren inte behöver skriva det manuellt.*

Vad är **O/RM**? (forts).

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



Vad är Entity Framework?

- Skriva och hantera ADO.Net kod för dataåtkomst det är jobbigt och ensidigt.
- Microsoft har försett oss med ett **O/RM ramverk som kallas "Entity Framework"** för att **automatisera databasrelaterade aktiviteter** för ditt program.
- Microsoft har gett följande **definition** av Entity Framework:

  Microsoft ADO.NET Entity Framework är ett objekt / Relational Mapping (ORM) ramverk som gör det möjligt för utvecklare att arbeta med relationsdata som domänspecifika föremål, vilket tar bort det största behovet av att skriva dataåtkomst kod som utvecklare behöver vanligtvis skriva. I och med användningen av Entity Framework, så skapar utvecklare frågor med LINQ, som sedan hämtar och manipulerar data som starkt typade objekt. Entity Framework's ORM implementering tillhandahåller tjänster som ändringsspårning, identitetsupplösning, lazy loading, och fråge översättning så att utvecklarna kan fokusera på sin applikationsspecifika affärslogik snarare än dataåtkomst grunderna.

  Entity ramverk är ett objekt / Relational Mapping (O / RM) ramverk. Det är en förbättring av ADO.NET som ger utvecklare en automatiserad mekanism för åtkomst & lagring av data i databasen.

Vad är Entity Framework? (forts).

Entity ramverket är användbart i tre scenarier.

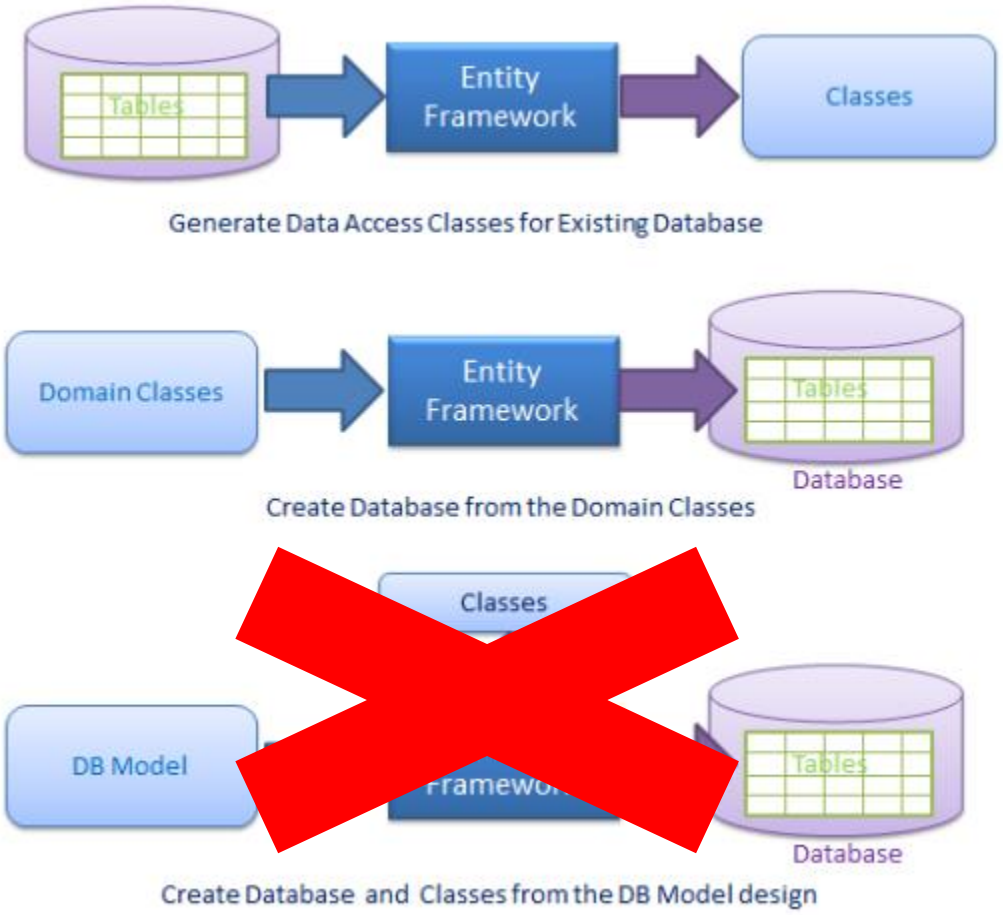
- **Först**, om du redan har befintlig databas eller om du vill designa din databas före dina andra delar av programmet.

- **För det andra**, du vill fokusera på dina domänklasser och sedan skapa databasen från dina domänklasser.

- **Tredje**, du vill designa ditt databasschema med den visuella designern och sedan skapa databasen och klasser.

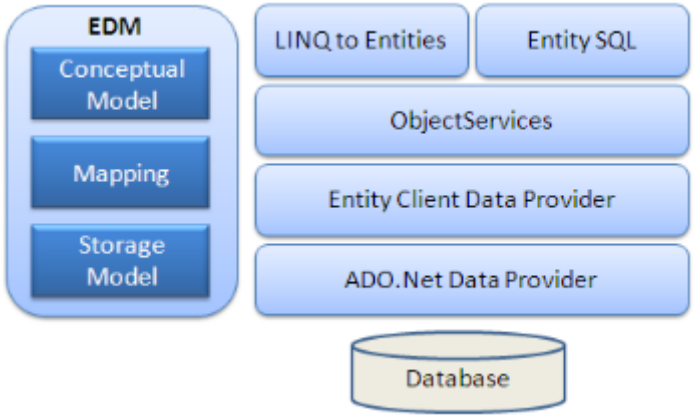
  Figur på nästa sida illustrerar ovanstående scenarier.

Vad är Entity Framework? (forts).



Entity Framework **arkitekturen**

Följande bild visar den övergripande arkitekturen för Entity Framework.



Entity Framework **arkitekturen** *(forts)*

EDM (Entity Data Model): EDM består tre huvudsakliga delar-Konceptuell modell, Mappnings och lagrings modell.

Konceptuell modell: Den konceptuella modellen innehåller modellklasser och deras relationer. Detta kommer att vara oberoende från din databas tabell design.

Lagrings Modell: Lagrings modell är den databasdesign modell som inkluderar tabeller, vyer, stored procedures och deras relationer och nycklar.

Mapping: Mappning består av information om hur den konceptuella modellen mappas till lagrings modellen.

LINQ till Entities: LINQ till Entiteter är ett frågespråk som används för att skriva frågor mot objektmodellen. Den returnerar entiteter, som definieras i den konceptuella modellen. Du kan använda dina LINQ kunskaper här.

Entity SQL: Entity SQL är ett annan frågespråk precis som LINQ till entiteter. Men det är lite svårare än L2E och även utvecklaren kommer att behöva lära sig detta separat.

Objekt service: Objekt service är en ingångspunkt för åtkomst av data från databasen och att returnera den tillbaka. Objekttjänst ansvarar för materialisering, vilket är processen att omvandla data som returneras från entitet klient dataprovidern (nästa lager) till en entitet som objektstruktur.

Entitet Klient Data Provider: Huvudansvaret för detta lager är att konvertera L2E eller Entity SQL frågor i en SQL fråga som förstås av den underliggande databasen. Den kommunicerar med ADO.Net dataprovider som i sin tur skickar eller hämtar data från databasen.

ADO.Net Data Provider: Detta lager kommunicerar med databasen med standard ADO.Net.

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 modell

Några **väsentliga delar** för vår Modell *(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 Modell 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.Contex.tt**
    - **Klassen** som **ärver** *från DBContext*
    - *Är den som ger oss access till entiteterna (klasserna)*
  - **Xxxx.tt**
    - **Klasserna** som **genererats** *från databasen*

Demo

Delarna av ett projekt med Entity Framework

Lab 1