Översikt

- Introduktion.
- Code-First.

.easec

Lektion 1: Introduktion

- Entity Framework.
- Object Relational Mapping.
- Arkitektur för Entity Framework.
- Entity Data Model (EDM).
- LINQ to Entities och Entity SQL.
- ObjectService.
- Entity Client Data Provider.

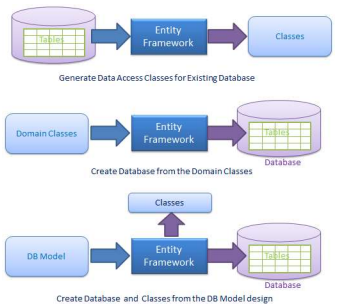
.easec

Entity Framework

- Entity Framework är en utökning av ADO.NET, som ger utvecklare en mekanism för tillgång och lagring av information i en databas.
- Är OpenSource.
- Är användbart i tre scenarios:
  - Om du redan har en databas och vill utgå från denna.
  - Om du fokusera på domänklasser, sedan utifrån dessa skapa databas.
  - Om du vill designa schema för databas i designer, sedan skapa databas och klasser utifrån design.

.eosec

Entity Framework (forts.)



.eosec

Versioner

- Version 5.
- Version 6.
- Version 7.
- Core.

.eosec

Object Relational Mapping

- Entity Framework är ett Object Relational Mapping (ORM) baserat ramverk.
- ORM är ett verktyg för att lagra objekt i en relationsdatabas, såsom MS SQL Server, automatiskt utan alltför mycket programmering.
- ORM inkluderar tre delar:
  - Domain class objects.
  - Relational database objects.
  - Information om knytningar mellan domain objects till objekt i databasen (tables, views och stored procedures).

.eavec

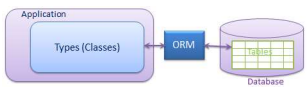
Object Relational Mapping (forts.)

- ORM ger möjlighet till att separera design av databas från design av domain classes.
- Gör bättre hantering av applikationen och möjlighet till utökning.
- Automatiserar CRUD-operationer.

.eavec

Object Relational Mapping (forts.)

- Ett typiskt ORM-verktyg genererar klasser genom interaktion mellan databas och applikation:

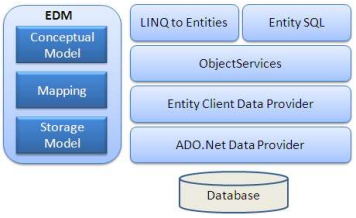


- Det finns ett antal ORM-baserade ramverk för .NET, förutom Entity Framework: DataObjects.Net, Nhibernate, OpenAccess, Subsonic.

.eavec

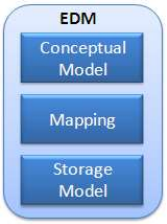
Arkitektur för Entity Framework

- Arkitekturen består av ett antal komponenter:



Entity Data Model (EDM)

- EDM består av tre delar: Conceptual Model, Mapping och Storage Model.
- Conceptual Model, består av klasser i modellen och dess relationer. Är oberoende av design för databastabell.
- Storage Model, är designmodell för databas med tabeller, views, stored procedures, relationer och nycklar.
- Mapping består av information om hur conceptual model knyts till storage model.



LINQ to Entities och Entity SQL

- LINQ to Entities, är ett språk som används för queries mot objektmodel. Kommer att returnera entities, dessa är definierade i conceptual model.
- Entity SQL, är ett annat språk för queries, lite svårare att lära sig.

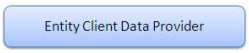
ObjectService

- ObjectService, är den huvudsakliga åtkomstpunkt för att få tillgång till information från databas och för att sedan returnera denna information.
- Är ansvarig för konvertering information som returneras från data provider till objektstruktur för entity.

- Object Service:
  - Tjänst som ger möjlighet att arbeta med Common Language Runtime (CLR) objekt.

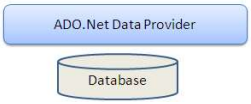
Entity Client Data Provider

- Entity Client Data Providers, är ansvarig för att konvertera LINQ to Entities eller Entity SQL queries till SQL query, som SQL-server förstår.
- Kommunicerar med ADO.Net data provider, som i sin tur skickar och tar emot information från databasen.



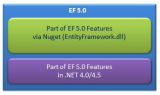
ADO.Net Data Provider

- ADO.Net Data Provider, kommunicerar med databasen genom att använda standard ADO.Net.



Sätta upp miljö

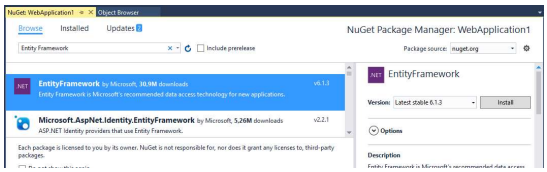
- Entity Framework 5.0 API distribuerades från två platser, Nuget-paket och i .NET framework.
- .NET Framework 4.0/4.5 inkluderade EF core API, EntityFramework.dll distribuerades via NuGet.



- EF 6.0 inkluderar EntityFramework.dll och är inte beroende av .NET Framework.

.eosec

Sätta upp miljö (forts.)



.eosec

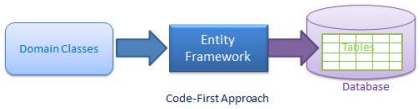
Lektion 2: Code-First

- Introduktion.
- Arbetsflöde.
- Bakomliggande funktioner.
- Initiera databas.
- Konventioner.

.eosec

Introduktion

- Code-First introducerades med Entity Framework 4.1.
- Är användbart vid Domain Driven Design.
- När Code-First används, kan du fokuserar på design av domain och arbeta med de klasser som behövs, för att därefter skapa databas.



Arbetsflöde

- Som utvecklare skriver du först C# klasser och context klasser.
- När applikation skapas, kommer API att skapa databasen (om den inte finns) och kommer att knyta dina klasser till databasen.
- Du kan också konfigurera dina klasser att skriva över standardinställningar, genom att använda DataAnnotation attribut eller fluent API.

Arbetsflöde (forts.)

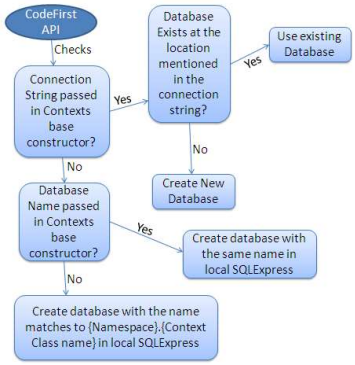
1. Skriv domain klasser och context klass.
2. Konfigurera eventuella ytterligare knytningar.
3. F5 för att köra applikationen.
4. API skapar nya databas eller knyta existerade databas till domain klass.
5. Lägga till (seed) standard- eller testvärde till databasen.
6. Applikationen startas.

Bakomliggande funktioner

- Entity Framework har ett context som hanterar interaktion mellan klasser och databas.
- Code-First lägger till model builder som inspekterar de klasser som context hanterar, använder sedan ett set av regler och konventioner för att räkna ut hur dessa klasser och relationer beskriver en model, men även hur model skall knytas till din databas.
- Allt detta händer i runtime, model visas aldrig, då den existerar i minnet.
- Model används också för att uppdatera databas när model ändras, detta görs genom funktionen Code First Migrations.

.eo/ee

Initiera databas



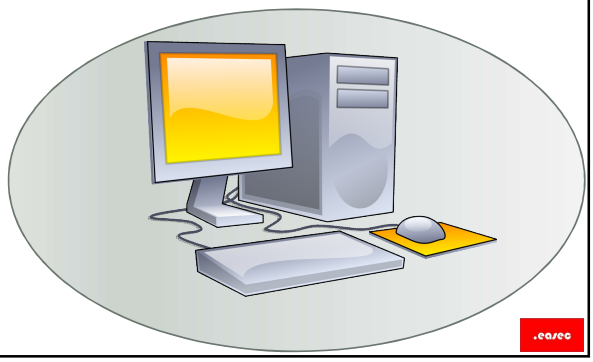
.eo/ee

Konventioner

- Konventioner är ett antal standardregler för att automatiskt skapa en conceptual model, som är baserad på definitioner i domain klass, när du arbetar med Code-First.
- Konventionerna finns definierade i System.Data.Entity.ModelConfiguration.Conventions.
- Några konventioner:
  - Type Discovery
  - Relationsship Convention
  - Complex type Convention
  - Primary Key Convention.
  - Foreign key Convention.
  - Default Code-First Conventions.

.eo/ee

Övning Microsoft Step-by-step



Repetitionsfrågor