Kurzreferenz („Cheat Sheet“) ADO.NET Entity Framework Code First - Modellerstellung

Autoren: FH-Prof. Manfred Steyer & Dr. Holger Schwichtenberg

www.IT-Visions.de
Dr. Holger Schwichtenberg

Kontextklasse (nur Basisklasse DbContext erlaubt)

```
public class HotelContext : DbContext
{
    public HotelContext():
            base("ConnectionString") {
        […] // optionale Initialisierung der DB
}
    public DbSet<Hotel> Hotels { get; set; }
    public DbSet<Region> Regionen {get;set;}

    protected override void OnModelCreating(
            DbModelBuilder modelBuilder)
    {
        […] // Fluent-API-Code
    }
}
```

Entitätsklassen mit Annotationen

Navigationseigenschaften müssen virtual sein, andere können virtual sein.
Alle müssen virtual sein für Change Tracking über Runtime Proxies!

```
[Table("RegionenTab")]
public class Region
{
    [Key,
     Column("Region_Code", Order=1)]
    [DatabaseGenerated(
         DatabaseGeneratedOption.None)]
    public int RegionCode { get; set; }
    [StringLength(27),
     Required,
     Column("Bez", Order = 3)]
    public string Bezeichnung { get; set; }
    public virtual
       ICollection<Hotel> Hotels {get;set;}
    public virtual
       ICollection<Hotel> TopRanked { … }
    [ConcurrencyCheck]
    [Column("Version",
     Order = 2, TypeName="bigint")]
    public int Version { get; set; }
}

public class Hotel
{
    public virtual
       int RegionCode { get; set; }

    [StringLength(50), MinLength(5)]
    public string Bezeichnung { get; set; }

    [InverseProperty("TopRanked")]
    public virtual
       Region TopRankedInRegion {get;set;}

    [Timestamp]
    public byte[]
       LetzteModifikation { get; set; }

    [NotMapped]
    public double TouristenPreis {
           get { return Preis * 2; } }

    [ForeignKey("RegionCode"),
     InverseProperty("Hotels")]
    public virtual Region Region {get;set;}
}
```

Schema Migration PowerShell Commandlets

- Enable.Migrations
- Add-Migration xy
- Update-Database -Script
- Update-Database
- Update-Database –TargetMigration xy
- Versionskonfliktlösung: Add-Migration xyMerge –IgnoreChanges

Optionale Datenbank-Initialisierung zur Laufzeit

- System.Data.Entity.CreateDatabaseIfNotExists
- System.Data.Entity.DropCreateDatabaseAlways
- System.Data.Entity.DropCreateDatabaseIfModelChanges

```
Database.SetInitializer(new
  DropCreateDatabaseAlways<HotelContext>());
```

Konventionen

Primärschlüssel

- „Id“ oder Klassenname + „Id“

```
public class Region
{
  public int RegionId { get; set; }
  public string Bezeichnung { get; set; }
  public virtual
     ICollection<Hotel> Hotels { get; set; }
}
```

Komplexe Typen

- haben keinen PK UND
- verweisen auf keine Entitäten UND
- keine Referenzierung über Auflistung in anderen Entitäten

```
public class Adresse
{
    public string Strasse { get; set; }
    public string Plz { get; set; }
    public string Ort { get; set; }
}
```

Fremdschlüssel-Mappings

- FK = Name des PKs der referenzierten Entität
- FK = Name der referenzierten Entität + PK
- FK = Name der Navigationseigenschaft + PK

```
public class Hotel
{
    public virtual Region Region {get;set;}
    public int RegionId {get;set;}
}

public class Region
{
    public int RegionId {get;set;}
    […]
    public virtual
       ICollection<Hotel> Hotels {get;set;}
}
```

Bidirektionale Beziehungen

- Es verweist jeweils nur eine Navigationseigenschaft auf das Gegenüber

```
public class Hotel
{
    public int HotelId { get; set; }
    […]
    public virtual Region Region {get;set;}
}

public class Region
{
    public int RegionId { get; set; }
    […]
    public virtual
       ICollection<Hotel> Hotels {get;set;}
}
```

M:N-Beziehungen

- Zwischentabelle wird von EF eingerichtet
- Name = Tabelle1 + Tabelle 2

```
public class Hotel
{
    […]
    public virtual ICollection<Merkmal> Merkmale { … }
}

public class Merkmal
{
    […]
    public virtual ICollection<Hotel> Hotels {get;set;}
}
```

Vererbung

- Standardstrategie: Table-per-Hierarchy (TPH), wenn Basisklasse ein DbSet in Kontextklasse hat.

Kurzreferenz („Cheat Sheet“) ADO.NET Entity Framework Code First - Modellerstellung

Autoren: FH-Prof. Manfred Steyer & Dr. Holger Schwichtenberg

- Name des Diskriminators: „Discriminator“
- Sonst: TPC

```
public class WellnessHotel: Hotel
{
    public int PoolsAnzahl { get; set; }
    public int SaunaAnzahl { get; set; }
}
```

Fluent-API (in OnModelCreating)

Konventionen hinzufügen und entfernen

```
modelBuilder.Conventions.Add<Konventionsklasse>();
modelBuilder.Conventions.Remove<Konventionsklasse>();
```

Typische Standard-Konventionen, die man entfernt:
- PluralizingTableNameConvention
- PluralizingEntitySetNameConvention
- ManyToManyCascadeDeleteConvention
- OneToManyCascadeDeleteConvention

Lightweight Conventions

```
// überall TPT
modelBuilder.Types()
.Configure(c => c.ToTable(c.ClrType.Name));

// überall Datetime2
modelBuilder.Properties<DateTime>()
.Configure(p => p.HasColumnType("datetime2"));
```

Entitäten abbilden

```
// TPH
modelBuilder.Entity<Hotel>()
    .Map<Hotel>(h => h.Requires("Type")
        .HasValue("Standard-Hotel"))
    .Map<WellnessHotel>(w => w.Requires("Type")
              .HasValue("Wellness-Hotel"));

// TPT
modelBuilder.Entity<Hotel>()
            .ToTable("Hotel");

modelBuilder.Entity<WellnessHotel>()
            .ToTable("WellnessHotel");

// TPC
modelBuilder.Entity<Hotel>().Map(m =>
  m.MapInheritedProperties().ToTable("Hotel"));
           modelBuilder.Entity<WellnessHotel>().Map(m =>
  m.MapInheritedProperties().ToTable("WellnessHotel"));
```

Eigenschaften abbilden

```
modelBuilder.Entity<Hotel>()
    .Ignore(h => h.TouristenPreis);

modelBuilder.Entity<Region>()
    .HasKey(r => r.RegionCode);

modelBuilder.Entity<Region>()
    .Property(r => r.RegionCode)
    .HasDatabaseGeneratedOption(
        DatabaseGeneratedOption.None);
```

Zusammengesetzte Schlüssel

```
modelBuilder.Entity<Rechnung>()
     .HasKey(t =>
              new { t.RechnungId, t.KundeId });
modelBuilder.Entity<Rechnungsposition>()
     .HasRequired(t => t.Rechnung)
     .WithMany(t => t.Positions)
     .HasForeignKey(d => new {
                d.RechnungId, d.KundeId });
```

Concurrency-Prüfung

```
modelBuilder.Entity<Hotel>()
    .Property(h => h.LetzteModifikation)
    .HasColumnType("timestamp")
    .IsConcurrencyToken()
    .HasDatabaseGeneratedOption(
        DatabaseGeneratedOption.Computed);

modelBuilder.Entity<Region>()
    .Property(r => r.Version)
    .IsConcurrencyToken();
```

Komplexe Eigenschaften

```
modelBuilder.ComplexType<Adresse>();

modelBuilder.Entity<Hotel>()
    .Property(p => p.Adresse.Strasse)
    .HasColumnName("Adresse_Strasse")
    .HasColumnType("varchar")
    .IsRequired();
```

1:N-Beziehungen

```
modelBuilder.Entity<Region>()
    .HasMany(r => r.Hotels)
          // Region hat viele Hotels
    .WithRequired(h => h.Region)
          // Hotel hat eine Region
    .HasForeignKey(h => h.RegionCode)
    .WillCascadeOnDelete(false);

modelBuilder.Entity<Hotel>()
    .HasOptional<Kategorie>(
      h => h.Kategorie)
        // Hotel hat 1 oder 0 Kategorie(n)
    .WithMany(k => k.Hotels)
        // Kategorie hat viele Hotels
    .Map(m => m.MapKey("KategorieId"));
```

N:M-Beziehung

```
modelBuilder.Entity<Hotel>()
    .HasMany<Merkmal>(h => h.Merkmale)
    .WithMany(m => m.Hotels)
    .Map(m => {
            m.MapLeftKey("HotelId");
            m.MapRightKey("MarkmalId");
            m.ToTable(
             "Hotel_Merkmal_Links");
        });
```

Externe Konfigurationsklasse

```
public class KategorieEntityTypeConfiguration :
                EntityTypeConfiguration<Kategorie>
{
    public KategorieEntityTypeConfiguration()
    {
        this.ToTable("KategorieTable");
    }
}

// Externe Konfiguration hinzufügen
modelBuilder.Configurations.Add(
    new KategorieEntityTypeConfiguration());
```

Stored Procedures

```
modelBuilder.Entity<Hotel>()
    .MapToStoredProcedures(sp => {
        sp.Update(conf => {
            conf.HasName("Insert_Hotel");
            conf.Parameter(
               h => h.Bezeichnung,
               "Bezeichnung",
               "Bezeichnung_Original");
            conf.Result(
               h => h.Bezeichnung,
               "Bezeichnung");
            conf.RowsAffectedParameter("cnt");
        });
        sp.Insert(conf => { […] });
        sp.Delete(conf => { […] });
    });
```

Literatur

Buch „Moderne Datenzugriffslösungen mit Entity Framework 6“

https://leanpub.com/entityframework [PDF, MOBI, EPUB]

http://www.amazon.de/dp/B016ON2TVE [DRUCK, MOBI]