www.IT-Visions.de
Dr. Holger Schwichtenberg

Kurzreferenz („Cheat Sheet“)
ADO.NET Entity Framework 4.1 Code Only

Autor: FH-Prof. Manfred Steyer
Experte für .NET, insbes. ORM und SOA

Datenbank und Context

DbContext

```
public class HotelContext : DbContext
{
    public HotelContext():
            base("HotelDb-CodeFirst") { }
    public DbSet<Hotel> Hotels { get; set; }
    public DbSet<Region> Regionen {get;set;}
}
```

Datenbank-Initialisierung

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

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

Datenbank-Initialisierung

```
class CustomDbConnectionFactory :
                IDbConnectionFactory {

   public DbConnection
      CreateConnection(
         string nameOrConnectionString) {
              return …
   }
}

Database.DefaultConnectionFactory =
          new CustomDbConnectionFactory();
```

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-Hirarchy (TPH)
- Name des Diskriminators: Discriminator

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

Attribute

```
[Table("RegionenTab")]
public class Region
{
    [Key,
     Column("Region_Code", Order=1)]
    [DatabaseGenerated(
         DatabaseGeneratedOption.None)]
    public int RegionCode { get; set; }
    [MaxLength(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; }

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

    [InverseProperty("TopRanked")]
```

Kurzreferenz („Cheat Sheet“)
ADO.NET Entity Framework 4.1 Code Only

Autor: FH-Prof. Manfred Steyer
Experte für .NET, insbes. ORM und SOA


```
    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;}
}
```

Fluent-API

OnModelCreating

```
public class HotelContext : DbContext
{
    public HotelContext() :
             base("HotelDb-CodeFirst-2") { }
    public DbSet<Hotel> Hotels { get; set; }
    public DbSet<Region> Regionen {get;set;}

    protected override
        void OnModelCreating(
            DbModelBuilder modelBuilder)
    {
         […]
    }
}
```

Konventionen entfernen

```
modelBuilder
  .Conventions
  .Remove<PluralizingTableNameConvention>();

modelBuilder
  .Conventions
  .Remove<ColumnTypeCasingConvention>();
```

Entitäten

```
modelBuilder
    .Entity<Region>().ToTable("Regionen");
```

Eigenschaften mappen

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

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

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

Concurrency-Checks

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

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

Komplexe Eigenschaften mappen

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

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

1:N-Beziehungen mappen

```
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"));
```

```
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");
        });
```

Vererbung

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

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

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

Konfiguration

```
public class KategorieEntityTypeConfiguration :
                EntityTypeConfiguration<Kategorie>
{
    public KategorieEntityTypeConfiguration()
    {
        this.ToTable("KategorieTable");
    }
}
[…]
// Externe Konfiguration hinzufügen
modelBuilder.Configurations.Add(
    new KategorieEntityTypeConfiguration());
```