

Entity Framework Core Code First Grundlagen

Ingo Köster

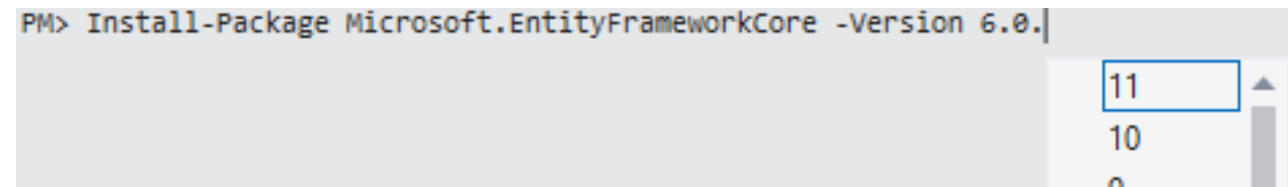
Diplom Informatiker (FH)

Datenbankanbieter und EF installieren

- › Zuerst wird der oder die gewünschten Datenbankanbieter per NuGet installiert
- › Hier wird nur der SQL Server verwendet
- › Über „Extras“ > „NuGet-Paket-Manager“ > „Paket-Manager-Konsole“ auswählen
- › Datenbankanbieter Paket
 - › `Microsoft.EntityFrameworkCore.SqlServer`
- › Entity Framework Paket
 - › `Microsoft.EntityFrameworkCore`

NuGet Pakete installieren

- › Installiert wird mit dem Befehl `Install-Package`
- › `Install-Package` installiert immer die neuste Version eines Paketes
- › Die passende Version für .NET 6 wird wie folgt installiert
 - › `Install-Package Microsoft.EntityFrameworkCore -Version 6.0.11`
- › Wildcards z.B. mit `*` werden nicht unterstützt, jedoch eine Autovervollständigung mittels der Tab-Taste



```
PM> Install-Package Microsoft.EntityFrameworkCore -Version 6.0.
```

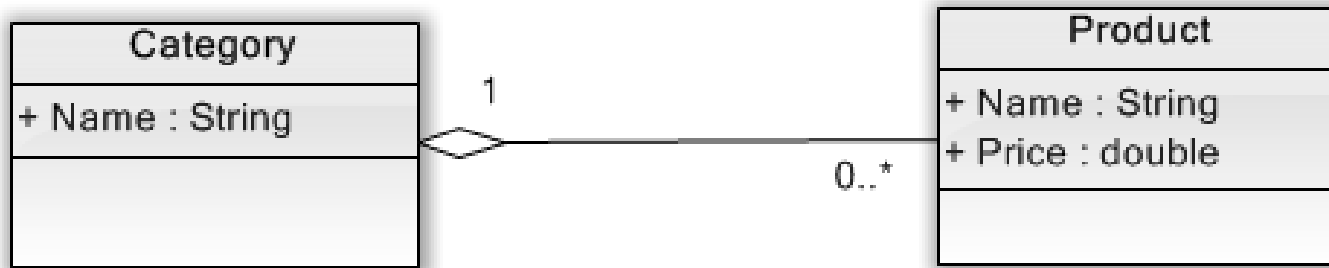
11
10
9

Datenbankanbieter (Auswahl)

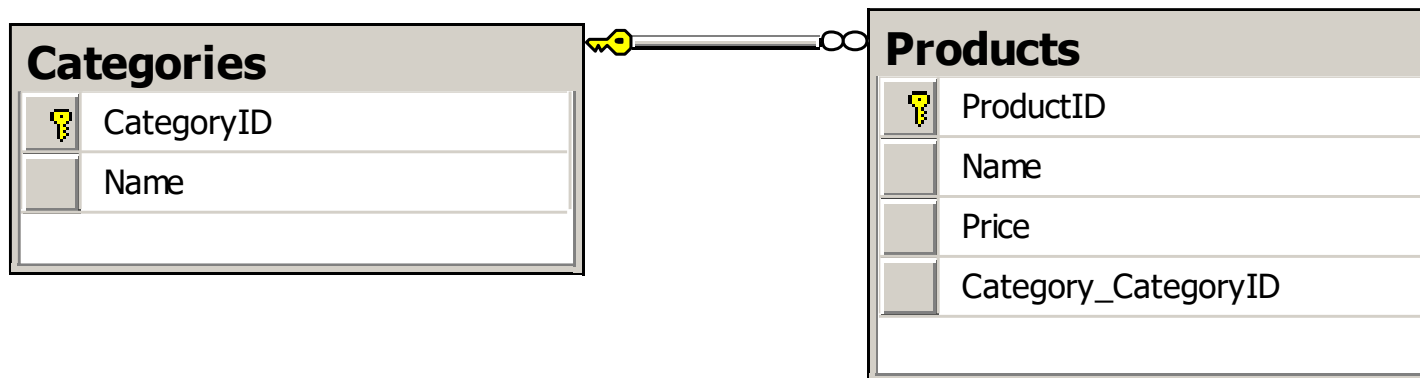
Datenbank	NuGet-Paket
SQL Server 2008 oder höher	<code>Microsoft.EntityFrameworkCore.SqlServer</code>
SQLite 3.7 oder höher	<code>Microsoft.EntityFrameworkCore.Sqlite</code>
EF Core-In-Memory-Datenbank	<code>Microsoft.EntityFrameworkCore.InMemory</code>
PostgreSQL	<code>Npgsql.EntityFrameworkCore.PostgreSQL</code>
MySQL, MariaDB	<code>Pomelo.EntityFrameworkCore.MySql</code>
MySQL	<code>MySql.Data.EntityFrameworkCore</code>
Oracle 9.2.0.4 oder höher	<code>Devart.Data.Oracle.EFCore</code>
PostgreSQL 8.0 oder höher	<code>Devart.Data.PostgreSql.EFCore</code>
Microsoft Access-Dateien	<code>EntityFrameworkCore.Jet</code>

Beispiel Model für die Folien

› UML Diagramm



› Die Tabellen, welche aus den Klassen erstellt werden sollen



Model

- › Im nächsten Schritt werden die notwendigen Klassen des Modells erzeugt
- › Für Klassen des Datenmodells gelten folgende Regeln:
 - › Klasse muss öffentlich sein
 - › Klasse muss ein Property für den Primary Key bereitstellen
 - › Primary Key wird über Namenskonventionen oder Attribut festgelegt
 - › Namenskonvention: ID oder TypnameID
 - › Attribut [Key]

Model (forts.)

- › Regeln für Klassen des Datenmodells (forts.):
 - › Foreign Keys können, müssen aber nicht explizit aufgeführt werden
 - › Navigationseigenschaften die automatisch geladen werden sollen, müssen virtuell sein
 - › Ab EF Version 2.1
 - › Private Properties werden nicht in der Datenbank gespeichert
 - › Ungeeignete Datentypen (z.B. ulong) werden nicht in der Datenbank gespeichert

Quelltext des Models

```
public class Category
{
    public int CategoryID { get; set; }
    public String Name { get; set; }
    :
    public virtual ICollection<Product> Products { get; set; }
}
```

```
public class Product
{
    public int ProductID { get; set; }
    public string Name { get; set; }
    public double Price { get; set; }

    [Required]
    public virtual Category Category { get; set; }
}
```


Der Datenkontext

- › Für den Datenbankzugriff wird eine sog. Kontext Klasse benötigt
- › Diese erbt von DbContext
- › Stellt als Properties DbSet zur Verfügung über welche auf die Objekte (Daten in der Datenbank) zugegriffen werden kann
- › Es können eine Vielzahl von Methoden, welche aus DbContext stammen überschrieben werden, um Einstellungen zu verwalten

Kontext Klasse

```
public class ProductContext : DbContext
{
    public DbSet<Category> Categories { get; set; }
    public DbSet<Product> Products { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
    {
        optionsBuilder.UseSqlServer(@"Server=.\SQLEXPRESS;Database=Prod
uctsDB;Integrated Security=True;");
    }
}
```

ConnectionString

- › Der ConnectionString ist im Unterricht oft wie folgt aufgebaut
 - › Server=.\SQLEXPRESS;
 - › Database=<Der Datenbankname>;
 - › Integrated Security=True;
- › Name der Instanz, kann auch IP oder Hostname sein
- › Name der Datenbank des Servers, zu welcher verbunden werden soll
- › Zugangsdaten, Integrated Security bedeutet, dass die Windows-Anmeldeinformationen verwendet werden sollen

ConnectionString

- › Wird mit mehreren Tabellen gearbeitet, ist ggf. noch die Verwendung vom mehreren gleichzeitigen Verbindungen notwendig
 - › `Server=... _Connection=True;MultipleActiveResultSets=True;`
- › Der ConnectionString sollte idealerweise nicht im Quelltext gespeichert werden!
 - › Mangelt an Flexibilität und birgt Sicherheitsrisiken
- › Bei den Konsolen-Übungen ist die Verwendung im Quelltext noch okay, ab den ASP.NET-Übungen wird der ConnectionString in einer Datei gespeichert

Datenbank anlegen

- › Nachdem Model und Kontext angelegt wurden, kann mittels der Kontext Klasse und der zugehörigen Properties auf die Datenbank zugegriffen werden
- › Da zu Beginn der Entwicklung noch keine Datenbank mit Tabellen vorhanden ist, können diese vom Entity Framework erstellt werden
- › Dazu wird die Methode EnsureCreated des Kontext Objektes verwendet
 - › `ProductContext ctx = new ProductContext();`
 - › `ctx.Database.EnsureCreated();`
- › EnsureCreated kann auch im Konstruktor des Kontexts aufgerufen werden

EnsureCreated

- › EnsureCreated erstellt die Datenbank inklusive Tabellen, falls diese nicht existieren
- › In allen anderen Fällen führt EnsureCreated nichts durch
- › Mittels der sog. Migrations kann ebenfalls eine Datenbank erstellt werden
- › Für die Migrations werden weitere Kommandos benötigt
 - › Migrations bieten deutlich mehr Optionen für das Erstellen und das Verändern von Datenbanken, falls sich die Model-Klassen verändern

Daten einfügen

```
using (ProductContext ctx = new ProductContext()) {  
    Category support = new Category() { Name = "Support" };  
  
    Product backup = new Product() {  
        Name = "Sicherung eines SQL Servers",  
        Price = 65.99,  
        Category = support  
    };  
  
    ctx.Products.Add(backup);  
  
    ctx.SaveChanges();  
}
```

Daten lesen

› Produkte abfragen

```
foreach (var product in ctx.Products) {  
    Console.WriteLine("{0}, {1} Euro Kategorie: {2}", product.Name,  
        product.Price, product.Category.Name);  
}
```

› Kategorien abfragen

```
foreach (var category in ctx.Categories) {  
    Console.WriteLine("Kategorie: {0}", category.Name);  
    foreach (var product in category.Products)  
    {  
        Console.WriteLine("\t - {0}", product.Name);  
    }  
}
```


Namen der Tabellen

- › Wenn die Datenbank erstellt wird, erstellt EF Tabellen mit den Namen, die den DbSet-Eigenschaftennamen entsprechen
- › Propertynamen für Auflistungen stehen in der Regel im Plural
 - › Students oder Studenten anstelle von Student
- › Tabellennamen können angepasst werden, um den Singular zu verwenden:

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Course>().ToTable("Course");
    modelBuilder.Entity<Student>().ToTable("Student");
}
```

Spaltennamen

- › Der Name eines Properties wird auch für den Namen der Spalte in der Datenbank verwendet
- › Mit dem Column-Attribut kann für die Erstellung der Datenbank ein anderer Name für die Spalte angegeben werden
 - › `[Column("Firstname")]`
 - › `public string Vorname { get; set; }`

Ids manuell vergeben

- › Soll der Primärschlüssel per Code und nicht durch die Datenbank vergeben werden, ist ein Attribut für das Id Property notwendig

```
[DatabaseGenerated(DatabaseGeneratedOption.None)]  
public int Id { get; set; }
```