**Reverse Engineering & Forward Engineering mit EF Core**

**Begriffsdefinitionen**

|  |  |
| --- | --- |
| Begriff | Beschreibung |
| Forward Engineering | Entwicklung eines Systems aus Modell bzw. Code heraus |
| Reverse Engineering | Rückgewinnung von Strukturinformationen aus einem existierenden System |
| In EF Core | Forward = Migrationen → DB, Reverse = DB → Code mit Scaffold-DbContext |

**Reverse Engineering mit EF Core**

1. Konsolenanwednung EFCNorthwind erstellen
2. Nuget-Paket Microsoft.EntityFrameworkCore.SqlServer hinzufügen
3. Nuget-Paket Microsoft.EntityFrameworkCore.Design hinzufügen
4. Projektmappe neu erstellen (rebuild Solution)
5. Ansicht -> Terminal ansteuern und darin den nächsten Punkt ausführen (vorher aber INSTANZNAME auf den eigenen anpassen).
6. dotnet ef dbcontext **scaffold** "Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=TableReservationSystem;Data Source=INSTANZNAME;TrustServerCertificate=true" Microsoft.EntityFrameworkCore.SqlServer -o Modelle

**Ergebnis:**

Eine DbContext-Klasse

Alle Entitäten als C#-Klassen mit Navigationseigenschaften

**Forward Engineering mit EF Core (“Code First”)**

**Ablauf: Code First mit Migrationen**

* Sie schreiben die Modellklassen selbst (Entitäten und DbContext)
* EF Core erzeugt daraus die Datenbank mittels Migrationen
* Code und DB sind synchronisiert über den Migrationsverlauf

**Wesentliche Schritte:**

1. Modell erstellen
2. Migration anlegen (dotnet ef migrations add)
3. Migration anwenden (dotnet ef database update)

**Übung – Forward Engineering („Code First”)**

Nutzen Sie den Code aus der vorherigen Reverse- Engineering-Übung zum Erstellen einer neuen Datenbank **EFCNorthwindVers2**

1. Ändern Sie Sie im Connectionstring (EfcnorthwindContext.cs) EFCNorthwind auf **EFCNorthwindVers2** um
2. Ansicht -> Terminal ansteuern und darin die nächsten beiden Punkte ausführen
3. dotnet ef migrations add InitialCreate
4. dotnet ef database update

**Weiterführende Übung**

1. Ändere in EfcnorthwindContext.cs alle Spalten vom Typ money in Spalten von Type decimal
2. Erstelle eine neue Migration: dotnet ef migrations add MoneyAufDecimal
3. Führe diese Migration aus: dotnet ef database update
4. Überprüfe das Ergebnis in SQL Server Management Studio

**Vergleichstabelle:**

|  |  |  |
| --- | --- | --- |
| Merkmal | Forward Engineering | Reverse Engineering |
| Startpunkt | Code | Datenbank |
| Flexibilität im Design | Hoch | Geringer |
| Synchronisation | Migrationen | Manuell / nicht automatisiert |
| Geeignet für | Neue Projekte | Bestehende Datenbanken |
| Änderungen verfolgen | Über Migrations | Schwer nachvollziehbar |