

Entity Framework

Kurzer Einblick in das Entity Framework, sowie das Datenmodell des KöTaf Projekts

Autor:	Florian Wasielewski
Letzte Änderung:	21.07.13
Version:	1.03

Inhalt

Abbildungsverzeichnis	2
Abkürzungsverzeichnis	3
Definition	4
KöTaf-Datenmodell	4
Tabellen / Attribute	5
Attribute hinzufügen	5
Tabelle hinzufügen	7
Einfache LINQ-Abfragen	8
Quellen	10

Abbildungsverzeichnis

Abbildung 1: Datenmodell des KöTaf-Projekts	4
Abbildung 2: Attribut hinzufügen.....	5
Abbildung 3: Eigenschaft hinzufügen	6
Abbildung 4: Eigenschaftsdetails.....	6
Abbildung 5: SQL-Statements erzeugen.....	6

Abkürzungsverzeichnis

Abkürzung	Beschreibung
EF	Entity Framework
ADO.NET	Datenbankschnittstelle des .NET Framework

Definition

Das [ADO.NET](#) Entity Framework ([EF](#)) ist ein OpenSource Objekt-relationaler Mapper für das NET Framework.

KöTaf-Datenmodell

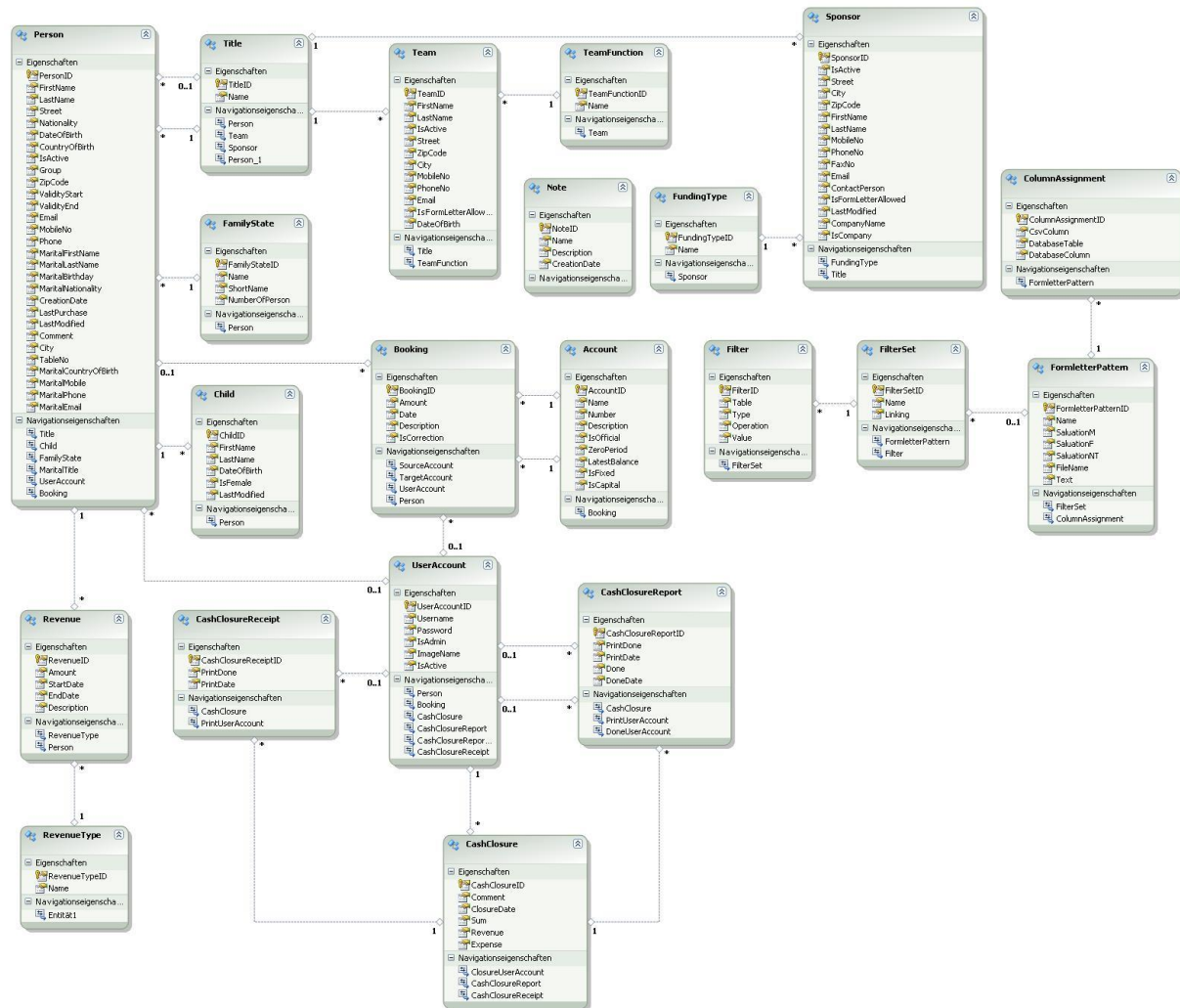


Abbildung 1: Datenmodell des KöTaf-Projekts

Unter „KöTaf.DataModel/_MODEL/EntityDesignerDiagram.jpg“ können Sie das Datenmodell auch in groß betrachten.

Tabellen / Attribute

Falls neue Tabellen oder Spalten im [EF](#) eingefügt werden müssen, so muss man wie folgt vorgehen.

- 1.) Im Projekt KöTaf.DataModel die Datei „TafelModel.edmx“ öffnen (Rechtsklick auf die EDMX-Datei und „Öffnen mit“ → ADO.NET Entity Data Model Designer)

Attribute hinzufügen

In der jeweiligen Entität auf die rechte Maustaste klicken und Hinzufügen → Skalareigenschaft auswählen.

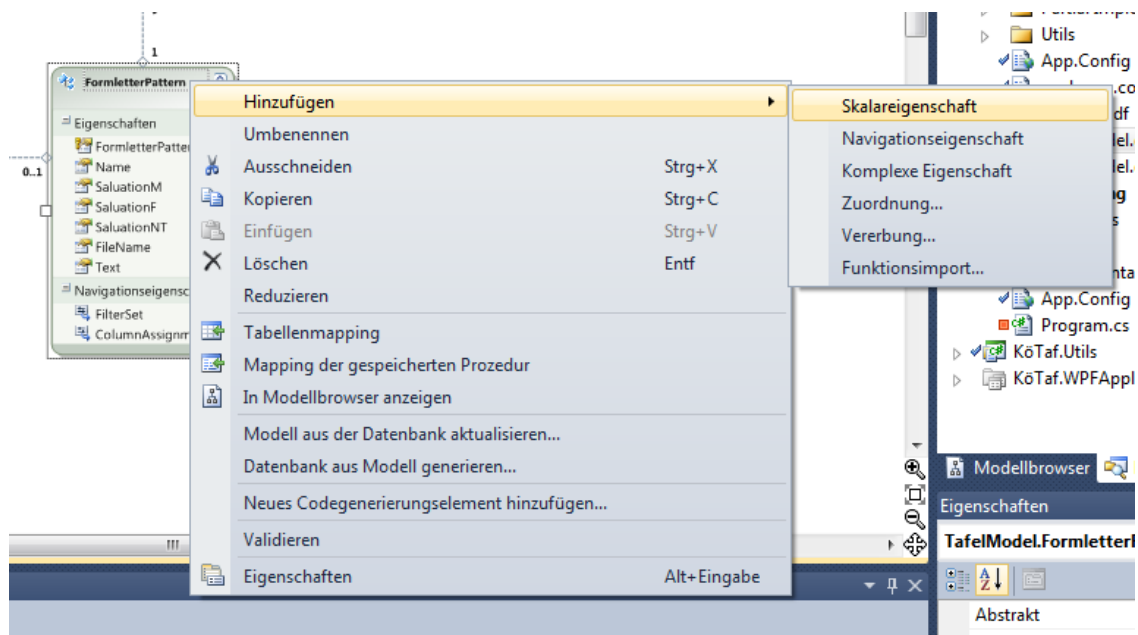


Abbildung 2: Attribut hinzufügen

Entity Framework 4

Im nächsten Schritt fügt das EntityFramework ein neues Attribut mit den Namen „Property“ ein.

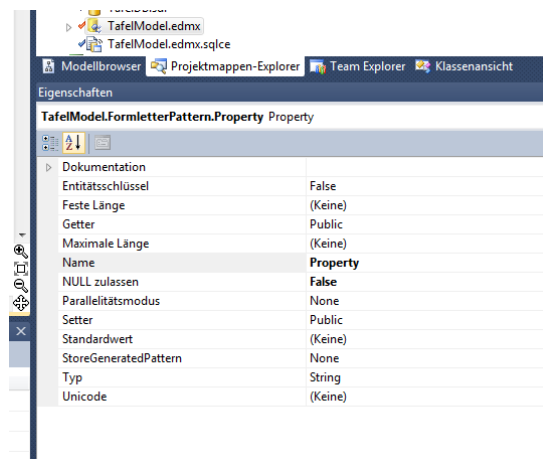
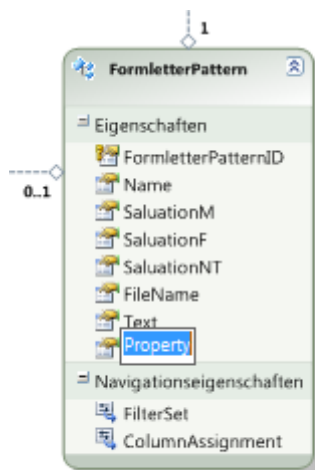


Abbildung 4: Eigenschaftsdetails

Abbildung 3: Eigenschaft hinzufügen

In Abbildung 4 sind folgende Eigenschaften daher auch interessant

- **Name**
Der Name ist der spätere Spaltenname in der Datenbank
- **NULL zulassen**
Gibt an ob die Spalte NULL-Werte enthalten kann
- **Typ**
Den Spaltentyp, wie z.B. String, DateTime, etc.

Im nächsten Schritt wird die Datenbank aktualisiert. Dazu schreibt das EntityFramework selbstständig SQL-Statements. Um die SQL-Statements zu generieren, klickt man auf die rechte Maustaste irgendwo auf der weißen Fläche und wählt „Datenbank aus Modell generieren...“ aus und danach auf fertigstellen (siehe Abbildung 5).

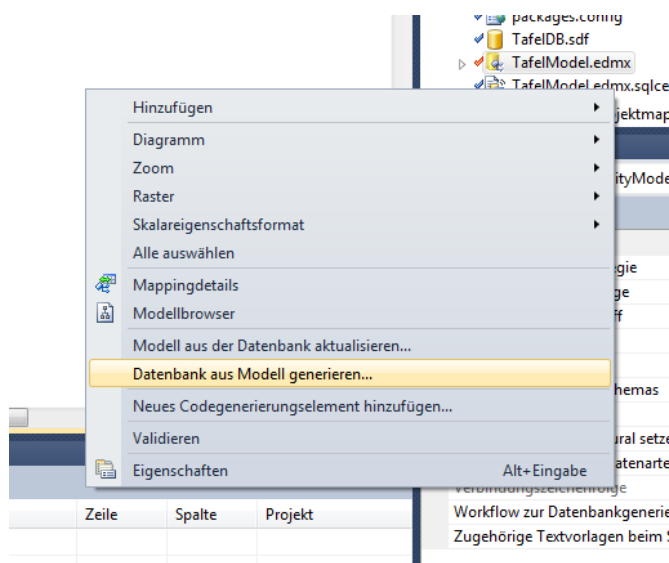
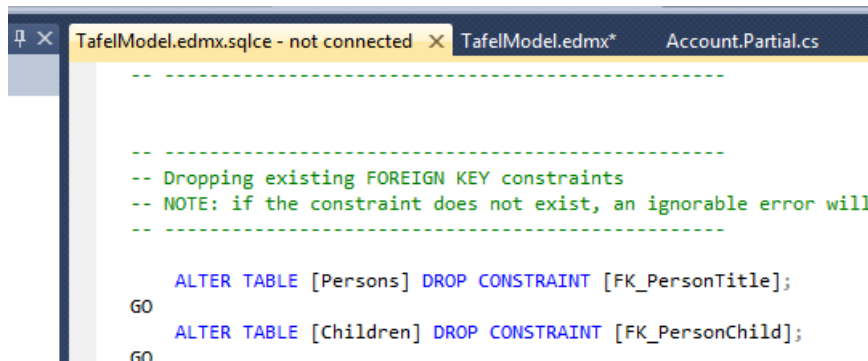


Abbildung 5: SQL-Statements erzeugen



```
TafelModel.edmx.sqlce - not connected X TafelModel.edmx* Account.Partial.cs P
--
-- Dropping existing FOREIGN KEY constraints
-- NOTE: if the constraint does not exist, an ignorable error will
--
ALTER TABLE [Persons] DROP CONSTRAINT [FK_PersonTitle];
GO
ALTER TABLE [Children] DROP CONSTRAINT [FK_PersonChild];
GO
```

Danach wird eine Datei namens `TafelModel.edmx.sqlce` generiert und automatisch geöffnet. Die geschriebenen SQL Statements können jetzt in die „TafelDB.sdf“ eingefügt werden.

WICHTIG:

ALLE ÄNDERUNGEN, SOWIE VORHANDENE EINTRÄGE WERDEN IN DER DATENBANK KOMPLETT ÜBERSCHRIEBEN.

Tabelle hinzufügen

Um eine neue Tabelle dem Datenmodell hinzu zufügen, klickt man rechts auf einer weißen Fläche im „ADO.NET Entity Data Model Designer“ und wählt Hinzufügen → Entität aus und folgt ebenfalls den Schritten [Attribute hinzufügen](#).

Einfache LINQ-Abfragen

Möchte man eine einfache Abfrage oder neue Methode für das EntityFramework generieren, so geht man in Projekt auf den Ordner „[Partial Implementations](#)“ und wählt eine Klasse aus.

Hierzu ein simples Beispiel um den vollständigen Namen einer Person auszugeben, anhand der PersonID:

```
public static string GetFullNameByPersonID(int personID)
{
    using (TafelModelContainer db = new TafelModelContainer())
    {
        var pObj = db.Persons.Single(p => p.PersonID == personID);

        return string.Format("{0} {1}", pObj.FirstName, pObj.LastName);
    }
}
```

WICHTIG ZU WISSEN

Wenn die PersonID aus irgendeinem Grund nicht in der Datenbank vorhanden ist, so schmeißt ADO.NET eine Exception bei der Methode „Single(...)“. Um das zu verhindern kann „SingleOrDefault(...)“ verwendet werden.

```
db.Persons.Single(p => p.PersonID == personID)
```

Möchte man eine Liste aller Personen zurückgeben, so kann man sich an folgendem Beispiel orientieren:

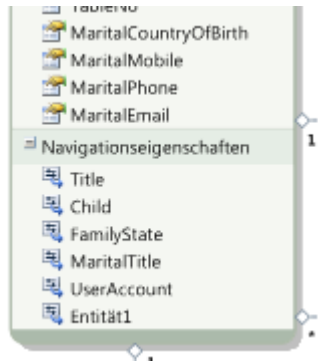
```
public static IEnumerable<Person> GetPersons(int? personID=null)
{
    using (TafelModelContainer db = new TafelModelContainer())
    {
        var p = db.Persons
            .Include("Child")
            .Include("Title")
            .Include("UserAccount")
            .Include("FamilyState")
            .AsQueryable();

        if (personID.HasValue)
            p = p.Where(p => p.PersonID == personID.Value);

        // Erst mit ToList() wird SQL Command ausgeführt!
        return p
            .OrderBy(p => p.TableNo)
            .OrderBy(p => p.Group)
            .ToList();
    }
}
```

1.) `db.Persons.Include("ABC")`

Möchte man später auf andere Tabellen bei dem eigentlichen Objekt zugreifen so verwendet man Include. Includes sind die Navigationseigenschaften bei den Entitäten:



2.) `p.Where(p => p.PersonID == personID.Value)`

Filterkriterium; Eine Person anhand ihrer PersonID filtern

3.) `p.OrderBy(p => p.TableNo)`

Personenliste aufsteigend sortieren

4.) `p.ToList()`

Führt den eigentlichen SELECT in der Datenbank aus und holt die die Personen

Alle wichtigen Teile sind im Code gelb markiert

Quellen

URL	Beschreibung
http://en.wikipedia.org/wiki/Entity_Framework	Entity Framework