Anmerkungen zum Code

Allgemein

- Nachdem das Programm zum ersten Mal ausgeführt wurde, sollte die Zeile 18 mit dem „InsertNewObject.Show();“ auskommentiert werden, weil sonst bei jedem Mal die Daten wieder eingefügt werden und das bei den Kursen dann zu Duplikaten führt (weil diese einen autoincremented PK haben und daher werden diese nicht als Duplikate gesehen, im Gegensatz zum Teacher, bei dem der PK angegeben wird), was zu etwas Verwirrung führen kann :D.

Datenbankerstellung

- Da ich bei meinem Code ein wenig an EF orientiert habe, funktionieren die Queries etc. anhand von DataSets (selbst geschriebene Klasse). Das heißt wenn die Klassen, die mithilfe des Or-Mappers abgebildet werden sollen, muss statt einer Liste ein DataSet erstellt werden. Listen werden nicht unterstützt.

```
[InverseProperty("course")]
1-Verweis
public virtual DataSet<Student> students { get; set; }
```

- Alle nicht simplen Typen, also DataSets und Objekte, müssen mit virtual angeführt werden.

```
[InverseProperty("HeadCourses")]
[field("ProfessorId")]
3 Verweise
public virtual Teacher Head { get; set; }

[InverseProperty("course")]
1-Verweis
public virtual DataSet<Student> students { get; set; }
```

- Mit der field Property kann der Name der Fk Spalte eines Objektes bestimmt werden. Das heißt beispielsweise die Verbindung zwischen der folgenden Klasse und dem Teacher Head funktioniert über eine Spalte ProfessorId. Wird dieser Name nicht angegeben wird einfach der Objektname benutzt.

```
[InverseProperty("HeadCourses")]
[field("ProfessorId")]
3 Verweise
public virtual Teacher Head { get; set; }
```

- Durch das Setzen vom InverseProperty Attribute kann bestimmt werden welche Property zu welcher Property in der anderen Klasse gehört. Dieses muss für alle 1_N, N_1, M_N Beziehungen gesetzt werden.
- Beim Setzen des PK Attribute kann ein Boolean als Parameter übergeben werden, anhand dessen kann man bestimmen, ob dieser Primary key autoincremented werden soll oder nicht. Das funktioniert natürlich nur bei Primary keys mit dem Typen Integer. Der Default Wert ist false;

```
/// <summary>Gets or sets the class ID.</summary>
[pk(true)]
public int ID { get; set; }
```

Einfügen

- Um ein Objekt zu speichern müssen dessen Referenzen bereits in der Datenbank liegen, das heißt diese müssen immer zuerst gespeichert werden und anschließend kann das eigentliche Projekt mithilfe der Save Funktion gespeichert werden.

```
//Save all refernces of course object
Teacher t = new Teacher()
{
    ID = "T1",
    Name = "Lieb",
    FirstName = "Anna",
    BirthDate = new DateTime(),
    HireDate = new DateTime(2015, 1, 1),
    Gender = Gender.FEMALE,
    Salary = 3000,
    HeadCourses = null
};
myDbContext.Save(ref t);
```

```
//Save the courses with its referneces
Course c = new Course
{
    Name = "Turnen",
    Active = true,
    Head = t,
    students = students,
    ClassRoomTeacher = classTeachers
};
myDbContext.Save(ref c);
```

- Wenn ein Objekt ein DataSet enthält (weil ich wie bereits erwähnt Datasets statt Listen in den Objekten benutze) muss dieses DataSet zunächst in der Datenbank gespeichert werden. Dafür muss lediglich eine **Liste** erstellt werden, der Objekte hinzugefügt werden und anschließend muss diese gespeichert werden. Diese List kann man dann der Dataset property zuweisen.
- Damit das Speichern funktioniert, muss das übergebene Objekt mit ref übergeben werden (siehe Bild oben).

- Der Save Funktion können sowohl einfache Objekte als auch ganze Listen übergeben werden.

```
Course c = new Course...;
myDbContext.Save(ref c);
```

```
List<Teacher> classTeachers = new List<Teacher>
{
    new Teacher()...,
    new Teacher()...
};
myDbContext.Save(ref classTeachers);
```

Update

- Mithilfe der Save Funktion kann ebenfalls ein Objekt geupdatet werden, hierbei gelten die gleichen Sachen wie beim Einfügen.

Vererbung

- Ich habe leider zu spät gemerkt, dass in den Folien steht, dass wir die Vererbung nicht mit „Table per concrete Type“ umsetzen sollten und konnte das deswegen nicht mehr ändern, weil ich sonst meine Select, Insert und Update Funktionen ändern musste.
- Vererbung wird nun aber dennoch unterstützt, aber dafür muss die Klasse von der geerbt wird, als abstract deklariert werden und es muss für die Basisklasse ein DataSet im DbContext vorhanden sein. (siehe Show/TestInheritanceSupport.cs).

```
public DataSet<Teacher> Teachers { get { return Get<Teacher>(); } }
0 Verweise
public DataSet<Student> Students { get { return Get<Student>(); } }
3 Verweise
public DataSet<Person> Persons { get { return Get<Person>(); } }

7 Verweise
public abstract class Person
```

Objekt locken

- Ein Objekt wird nach dem Aufrufen der Lock Funktion gelockt und nach 5 Minuten wird automatisch wieder das Lock aufgehoben,