Dokumentation für ‚object-relational mapping' (ORM) Framework

Das ORM-Framework funktioniert mittels Code-First Ansatz. Dieser Ansatz ermöglicht es Metadaten in den Code einzubinden und erleichtert es Änderungen am Modell durchzuführen.

Einrichtung des Frameworks

1. SQLite installieren und Datenbank erzeugen
2. Datenbank-Schema für die Anwendung erstellen
    a. Mittels sql-query in SQLite
    b. Mittels Erzeugen von Klassen -> CreateTable(of Class)
3. Connection-String in Program.vb → Main Methode adaptieren
   z.B.: `Connection = New SQLiteConnection("Data Source=Filename.db;")`
4. Das Framework ist nun einsatzbereit

Code-Datenbank Mapping

Das Framework mappt die Entitäten (Klassen) mit den Tabellen der Datenbank; Damit dieser Vorgang funktioniert wurden folgende Annotationen implementiert:

Table-Attribute (`TableName As String`)

Die Table-Attribute Annotation mappt die Klasse mit der dazugehörigen Tabelle der Datenbank. Bei Erstellen einer Klasse, sollte die Annotation folgenderweise <TableAttr)> über die Tabelle gesetzt werden. Falls der Klassenname nicht mit dem Tabellenname übereinstimmt, kann mit folgender Schreibweise <TableAttr(TableName:="TABLE_NAME")> der Tabellenname festgelegt werden.

Column-Attribute (`ColumnName As String, ColumnType As Type, IsNullable As Boolean`)

Die Column-Attribute Annotation mappt die Eigenschaft (Property) der Klasse als Spalte in der Datenbank. Diese Annotation wird mit folgender Schreibweise <ColumnAttr> gesetzt. Wenn jedoch der Name der Eigenschaft nicht mit dem Spaltennamen der Tabelle übereinstimmt, kann dieser geändert werden: <ColumnAttr(ColumnName:="COLUMN_NAME")>. Falls notwendig kann der ColumnType extra gesetzt werden. Außerdem ist es möglich IsNullable auf ‚true' zu setzen, damit dieser Wert auch NULL in der Datenbank enthalten kann. Der Default-Wert von IsNullable ist ‚false'.

PrimaryKey-Attribute

Die PrimaryKey-Attribute Annotation markiert die Eigenschaft (Property) als Primärschlüssel.

ForeignKey-Attribute

Die ForeignKey-Attribute Annotation markiert die Eigenschaft (Property) als Fremdschlüssel. Bei Verbindungen wie 1:1, n:1, 1:n muss nur die Annotation gesetzt werden.

Wenn es sich aber um eine M:N Beziehung handelt muss die Assignment-Table bestimmt werden.

```
<FKAttr(AssignmentTable:="table_name ", ColumnName:="column_name",
RemoteColumnName:="remote_column_name")>
```

Ignore-Attribute

Die Ignore-Attribute-Annotation können gesetzt werden, wenn das Objekt nicht für die Datenbank zu berücksichtigen ist und ignoriert werden kann.

Table Management

Wenn das Klassen-Model steht und alle Annotationen richtig gesetzt worden sind kann das Datenbank-Schema erzeugt werden.

Erzeugen von Tabellen aus Klassen

Das Erzeugen einer Tabelle mittels einer Klasse funktioniert sehr einfach. Durch Aufrufen der ‚CreateTable' Methode wird automatisch eine Tabelle in der Datenbank erzeugt.

```
CreateTable(Of klassen_name)()
```

Löschen von existierenden Tabellen

Nicht nur das Erzeugen, sondern auch das Löschen einer Tabelle funktioniert einwandfrei. Durch Aufrufen der ‚DropTable' Methode wird die Tabelle der angegebenen Klasse entfernt.

```
DropTable(Of klassen_name)()
```

Erzeugen von Indizes auf eine Tabelle

Zum Erstellen eines Index wird die Methode ‚CreateIndex' aufgerufen.

```
CreateIndex(index_name, table_name)
```

Funktionalitäten

Erzeugen von Objekten

Um ein Objekt einer Klasse in der Datenbank zu speichern, kann die Methode ‚SaveObject' aufgerufen werden.

```
SaveObject(object_name)
```

Lesen von Objekten

Um ein Objekt von einer bestimmten Klasse aus der Datenbank zu lesen, verwendet man die ‚GetObjectType' Methode. Um erfolgreich ein Objekt aus der Datenbank zu laden wird der ‚primary-key' des Objektes benötigt.

```
Dim obj = [GetObjectType](Of Klassen_name)(pk as String)
```

Anpassen von Objekten

Um eine Änderung an einem Objekt durchzuführen, wird dieses zuerst aus der Datenbank gelesen. Anschließend können die Werte des Objekts wie oben beschrieben, geändert und gespeichert werden.

```
Dim obj = [GetObjectType](Of klassen_name)(pk as String)
obj.property = new_value
SaveObject(obj)
```

Löschen von Objekten

Um ein Objekt aus der Datenbank zu löschen, muss es zuerst aus der Datenbank gelesen und zwischengespeichert werden. Anschließend kann das Objekt mit der ‚RemoveObj' Methode gelöscht werden.

```
Dim obj = [GetObjectType](Of Klassen_name)( pk as String)
RemoveObj(obj)
```

Query Sprache für Filtern von Objekten

Um Objekte aus der Datenbank zu filtern kann der QueryBuilder verwendet werden. Folgende Operationen und Konnektoren stehen zur Verfügung:

- GreaterThan
- GreaterThanOrEqual
- LowerThan
- LowerThanOrEqual
- Equals
- Like
- In
- Not
- Or
- And
- Group
- End Group

Beispiel:

Ausgabe aller Trainer mit einem Mindestgehalt von 36000€.

```
Dim trainer As QueryBuilder(Of Trainer) = [Select](Of
Trainer)().GreaterThanOrEqual("Gehalt", 36000)

Console.WriteLine("Alle Trainer mit einem Mindestgehalt von 36000: ")

        For Each item In trainer
            Console.WriteLine(item.FirstName + " " + item.Name)
        Next
```

Sperren und Freigeben von Objekten in der Datenbank

Um ein Objekt in der Datenbank zu sperren, damit es zu keinen Komplikationen kommt, kann die Methode ‘LockDBObject‘ verwendet werden.

```
LockDBObject(obj)
```

Um das gesperrte Objekt wieder freizulassen, ruft man die Methode ‚ReleaseDBObject‘ auf.

```
ReleaseDBObject(obj)
```