
02 – DB-Programmierung

ADO.NET und EF

Agenda

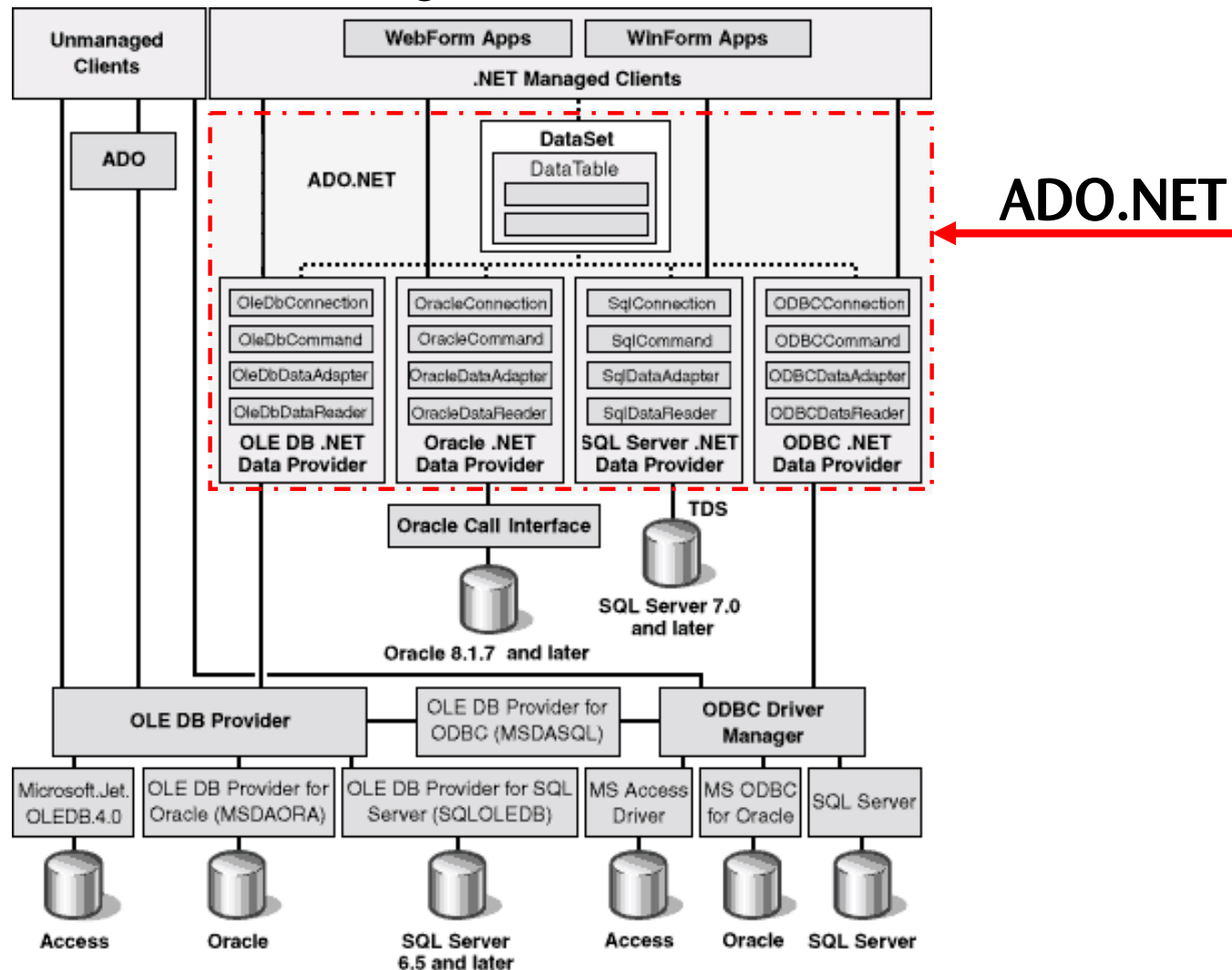
- Einführung in ADO.NET
- Architektur
- DataProvider
- Database Connection
- Command
- DataReader
- DataAdapter
- DataSet
- Entity Framework

Was ist ADO.NET?

ADO.NET ist eine Klassenbibliothek für den Zugriff auf Datenbankdienste für:

- Datenbankabfragen
- Datenmanipulation
- Datenbank-Commands erstellen und parametrieren
- Stored-Procedures der Datenbank ausführen
- Transaktionen
- Data-Binding (Verbinden von GUI-Komponenten mit Daten)
- Aber auch XML-Dateien als Datenquelle und -senke

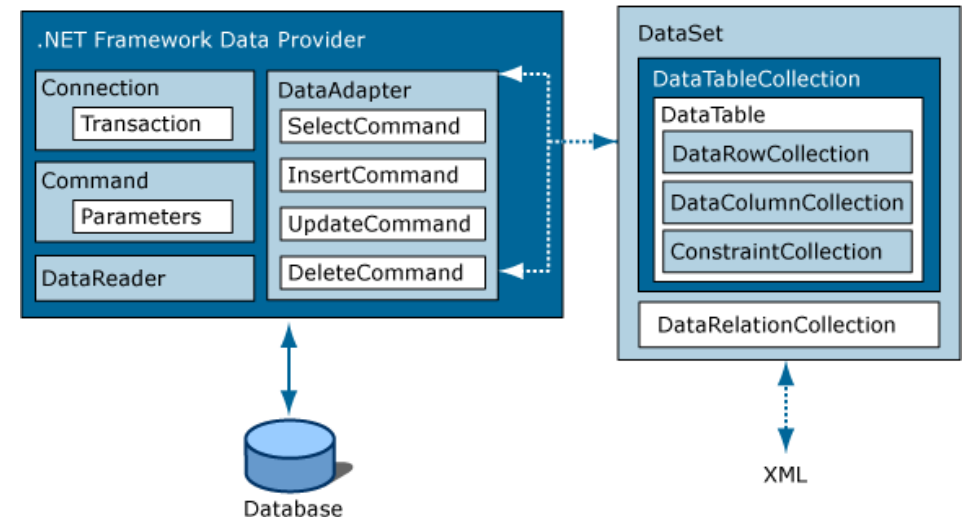
Übersicht – Datenbankzugriff



Übersicht – ADO-Komponenten

■ Data Provider

- Beinhaltet Objekte und Methoden, um eine Verbindung zu einer Datenbank aufzubauen, Befehle darauf auszuführen und Daten persistent zu modifizieren.
- Die enthaltenen Daten können dann entweder direkt verarbeitet oder in ADO.NET-Datasets zur Weiterverarbeitung verpackt werden.



■ DataSet

- Das DataSet bildet die Datenstruktur als entkoppeltes relationales Modell im Speicher ab, unabhängig von der ursprünglich zu Grunde liegenden Datenquelle.
- Es bildet die Gesamtheit der zur Verfügung stehenden Daten in Form von Spalten und Zeilen ab, inklusive aller Constraints und Beziehungen zwischen den Daten.

ADO.NET-DataProvider

- Ein Datenprovider stellt Klassen für den Zugriff auf einen bestimmten Datenspeichertyp zur Verfügung.
- Jeder .NET-Datenprovider implementiert dabei die gleichen Basisklassen, beispielsweise `Connection`, `Command` und `DataAdapter`.
- Die folgenden Datenprovider sind Teil des .NET Frameworks:
 - `SqlClient Provider`
 - `OleDb Provider`
 - `Odbc Provider`
 - `Oracle Provider`
 - `EntityClient Provider`
 - `SQLite`

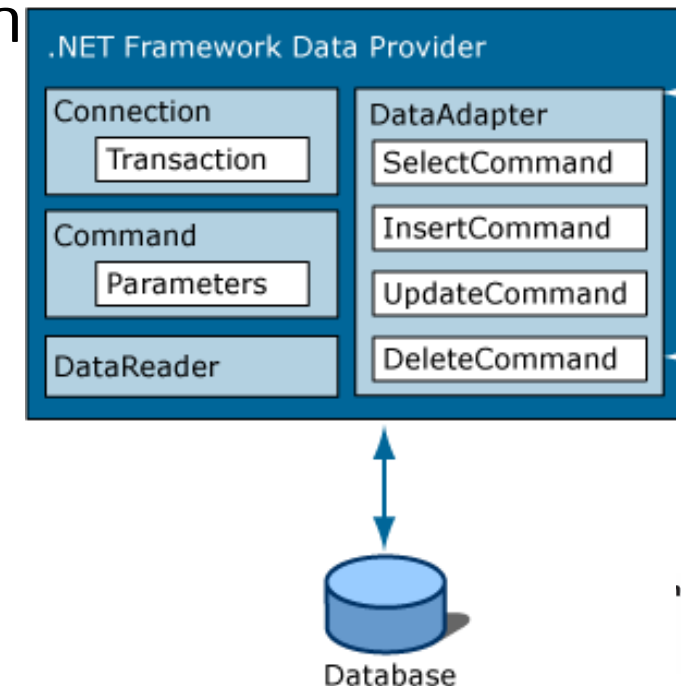
Übersicht

DataProvider für	Beschreibung
SQL Server	Bietet einen Datenzugriff für Microsoft SQL Server Version 7.0 oder höher. Namensraum: System.Data.SqlClient
OLE DB	Bietet einen Datenzugriff für Datenquellen via OLE DB. Namensraum: System.Data.OleDb
ODBC	Bietet einen Datenzugriff für Datenquellen via ODBC. Namensraum: System.Data.Odbc
Oracle	Bietet einen Datenzugriff für Oracle Datenbanken. Unterstützt Oracle Client Software 8.1.7 und neuere Versionen. Namensraum: System.Data.OracleClient
EntityClient Provider	Bietet einen Datenzugriff für Entity Data Model (EDM) Applikationen. Namensraum: System.Data.EntityClient

Hauptklassen eines DataProviders

- Alle DataProvider implementieren folgende Klassen

Klasse	Beschreibung
Connection	Stellt eine Verbindung zur Datenquelle (Datenbank) her.
Command	Führt einen Befehl (z.B. SQL oder Stored Procedure) auf der Datenbank aus. Basisklasse ist DbCommand.
DataReader	Stellt einen Read-Only Stream zur Datenquelle her. Basisklasse ist DbDataReader.
DataAdapter	Bewirtschaftet ein Dataset und aktualisiert die Datenbank. Basisklasse ist DbDataAdapter.



Database Connection instanziiieren

- Ohne Verbindung zur Datenbank läuft nichts
- Der Datenbankzugriff läuft immer gleich ab:
 - Herstellen einer Verbindung zur Datenbank
 - Absetzen von Kommandos
 - Ermitteln der Ergebnisse bei Abfragen (SELECT)
 - Schliessen der Verbindung
- Ein neues `SqlConnection`-Objekt instanziiieren:

```
using System.Data.SqlClient; // Namensraum
...
SqlConnection con = new SqlConnection();
```

- Das `Connection`-Objekt kann dann beliebig wieder verwendet werden (Connection-Pooling).

Database Connection

Database Connection instanziiieren

- Eine Datenbankverbindung zum DB-Server wird mittels `ConnectionString` hergestellt.
- Beispiel: Angabe des `ConnectionString` im Konstruktor

```
string connString = "Data Source=(local);" +  
                    "Initial Catalog=Northwind;" +  
                    "Trusted_Connection=yes");  
  
SqlConnection con = new SqlConnection(connString);
```

- Beispiel: Eigenschaft `ConnectionString` setzen

```
SqlConnection con = new SqlConnection();  
con.ConnectionString = "Data Source=(local);" +  
                      "Initial Catalog=Northwind;" +  
                      "Integrated Security=sspi";
```

Database Connection

Der Connection String

Key	Beschreibung
Connect Timeout, Connection Timeout	Zeitdauer [sec], die auf eine Verbindung zum Server gewartet werden soll, bevor der Versuch abgebrochen und ein Fehler generiert wird. Defaultwert = 15 Sec.
Data Source, Server, Address, Addr, Network Address	Entweder der Name oder die Netzwerkadresse der Instanz des SQL Servers, mit dem eine Verbindung hergestellt werden soll. Beispiele: Data Source=local (lokal) Data Source=Hostname\Instanzname (Netzwerk) Data Source=.\SQLEXPRESS (SQL-Express Edition)
Initial Catalog, Database	Hier wird der Name der Datenbank angegeben.
Integrated Security, Trusted Connection	Bei false werden die Benutzer-ID und das Kennwort für die Verbindung angegeben. Bei true werden die aktuellen Anmeldeinformationen des Windows-Kontos für die Authentifizierung verwendet. Gültige Werte sind true, false, yes, no und SSPI, das äquivalent zu true ist. (SSPI = Security Support Provider Interface, Windows Security API)

Database Connection

Der Connection String

Key	Beschreibung
Packet Size	Setzt die Größe der Netzwerkpakete in Bytes, die zum Kommunizieren mit einer Instanz von SQL Server verwendet werden. Defaultgröße ist 8192 Bytes, kann aber zwischen 512 und 32.767 Byte variieren.
Password, Pwd	Das Passwort fürs SQL Server-Konto.
User Id	Der SQL Server-Account.
Workstation ID	Der Name des Client Computers, welcher sich zum SQL Server verbinden möchte.

Database Connection

Der Connection String

- Beispiel für «MS Access»-Datenbank:

```
string connStr = @"Provider=Microsoft.Jet.OLEDB.4.0;"  
                + "Data Source=C:\Test\PersonenVerwaltung.mdb"  
                + ";Persist Security Info=False";
```

Datenbankzugriff

■ Beispiel: Einfacher Datenbankzugriff auf SQL Server

```
using System.Data.SqlClient;

string connString = "Data Source=(local);" +
                    "Initial Catalog=Northwind;" +
                    "Trusted_Connection=yes";

SqlConnection sqlConn = new SqlConnection(connString);
sqlConn.Open();

// ... Datenbank Zugriff ...

sqlConn.Close();
```

Datenbankzugriff

■ Beispiel: Einfacher Datenbankzugriff mit using-Block

```
string connString = "Data Source=(local);" +  
                    "Initial Catalog=Northwind;" +  
                    "Trusted_Connection=yes");  
  
using(SqlConnection con = new SqlConnection(connString))  
{  
    sqlConn.Open();  
    // ... Datenbank Zugriff ...  
}
```

- Beim using-Block werden die Methoden `Close()` und `Dispose()` in jedem Fall aufgerufen und alle Ressourcen freigegeben, auch im Exception-Fall
-

Database Connection Management

- Wie lange soll eine Datenbankverbindung offen sein?
 - Grundsätzlich sollte eine Verbindung so schnell wie möglich wieder geschlossen werden, um die dafür beanspruchten Ressourcen eines Datenbankservers möglichst gering zu halten.

SqlCommand

- Das `SqlCommand`-Objekt repräsentiert einen SQL-Befehl oder eine gespeicherte Prozedur
- In der Eigenschaft `CommandText` wird die SQL-Anweisung bzw. die gespeicherte Prozedur festgelegt. Die Ausführung wird mit einer der `Execute`-Methoden gestartet.

SQL - Exkurs

■ Daten abfragen mit SELECT

```
SELECT <Ausdrucksliste> FROM <Tabelle>  
WHERE <Bedingung>
```

■ Beispiel - Alle Mitarbeiter der Abteilung Verkauf abfragen:

```
SELECT Name, Vorname, TelefonNr FROM Mitarbeiter  
WHERE Abteilung = 'Verkauf'
```

SQL - Exkurs

■ Daten einfügen mit INSERT

```
INSERT INTO <Tabelle>  
(Spalten) VALUES (Werte)
```

■ Beispiel – Ein Mitarbeiter einfügen:

```
INSERT INTO Mitarbeiter  
(Name, Vorname, TelefonNr, Abteilung)  
VALUES ('Meier', 'Peter', '567', 'Verkauf')
```

SQL - Exkurs

■ Daten aktualisieren mit UPDATE

```
UPDATE <Tabelle>  
SET <Spalte1=Wert1, Spalte2=Wert2, ...>  
WHERE <Bedingung>
```

■ Beispiel – Die Telefonnummer eines Mitarbeiters aktualisieren:

```
UPDATE Mitarbeiter  
SET TelefonNr = '989'  
WHERE (Name = 'Meier' AND Vorname = 'Peter')
```

SQL - Exkurs

■ Daten löschen mit DELETE

```
DELETE FROM <Tabelle>  
WHERE <Bedingung>
```

■ Beispiel – Einen Mitarbeitereintrag löschen:

```
DELETE FROM Mitarbeiter  
WHERE (Name = 'Meier' AND Vorname = 'Peter')
```

Ein Commando-String konstruieren

- 1) Command Objekt instanziiieren
- 2) Parameter und Werte hinzufügen
- Beispiel – Ein INSERT Commando erstellen:

```
private OleDbCommand GetInsertBekannteCommand(Person pers)
{
    string insertStr = @"INSERT INTO Bekannte" +
        "(Name, Vorname, Tel) VALUES (@Name, @Vorname, @Tel)";
    OleDbCommand cmd = new OleDbCommand(insertStr);
    cmd.Parameters.AddWithValue("@Name", pers.Name);
    cmd.Parameters.AddWithValue("@Vorname", pers.Vorname);
    cmd.Parameters.AddWithValue("@Tel", pers.Telefon);
    return cmd;
}
```

Command
String

Platzhalter
für Parameter

Parameter mit
Wert addieren

SQL Command ausführen

Methode	Beschreibung
ExecuteNonQuery	Führt ein SQL-Kommando auf die DbConnection aus und liefert die Anzahl betroffenen Zeilen als Rückgabewert.
ExecuteScalar	Führt ein SQL-Kommando auf die DbConnection aus und liefert den Wert der ersten Spalte in der ersten Zeile als Rückgabewert. Alle weiteren Spalten und Zeilen werden ignoriert.

```
using (OleDbConnection conn = new OleDbConnection(GetConnString()))
{
    // Command Objekt erstellen
    OleDbCommand cmd = GetInsertBekannteCommand(newPerson);

    conn.Open();                // DB Connection oeffnen
    cmd.Connection = conn;      // und dem Command zuweisen
    cmd.ExecuteNonQuery();      // Command ausführen
                                // oder cmd.ExecuteScalar();
}
```

Daten lesen mit Command und DataReader

- Der DataReader wird verwendet, um mehrere Zeilen aus der Datenbank zu lesen

Methode	Beschreibung
ExecuteReader	Führt ein SQL-Kommando auf die DbConnection aus und liefert ein SqlDataReader Objekt als Rückgabewert.

- Beispiel für SqlDataReader:

```
SqlCommand command = new SqlCommand(queryString, connection);  
SqlDataReader reader = command.ExecuteReader();
```


Daten lesen mit Command und DataReader

Beispiel Code mit OleDbDataReader:

```
public static void ReadData(string connString, string queryString)
{
    using (OleDbConnection conn = new OleDbConnection(connString))
    {
        OleDbCommand command = new OleDbCommand(queryString, conn);
        conn.Open();
        OleDbDataReader reader = command.ExecuteReader();

        while (reader.Read())           // Alle Zeilen auslesen
        {
            Console.WriteLine(reader[0].ToString()); // 1. Spalte ausgeben
        }
        reader.Close();
    }
}
```

Auf Null-Werte überprüfen

- Aus der Datenbank können `null`-Werte kommen. Diese können abgefragt werden:
- Beispiel: Abfrage der Spalte 0

```
int index = 0;
int zahl;
if (!dbReader.IsDBNull(index))
{
    zahl = dbReader.GetInt32(index);
}
else
{
    zahl = 0;
}
```

Übung 2.1



Personalverwaltung (1/3)

- Schreibe eine einfache «Windows Forms»-Applikation, um Mitarbeiterdaten zu verwalten
- Die Mitarbeiterdaten sind in der Datenbanktabelle „Persons“ gespeichert

Persons : Table

	ID	Name	Vorname	Strasse	PLZ	Ort	Telefon
	1	Meier	Bernd	Wilstrasse 2	8932	Beppiswil	0343213341
	2	Müller	Heino	Feldweg 5	3224	Zürich	4334211324
	3	Walder	Monika	Florastrasse 51	4312	Augst	432432432
	5	Moser	Hans	Dorfstrasse 2	4563	Allschwil	
	6	Wenger	Guido	Badenerstrasse 234	8001	Zürich	
	7	Manser	Urs	Wiesenstrasse 4	8952	Schlieren	
	8	Fender	Karl	Bertastrasse 5	8000	Zürich	
►	lumber)						

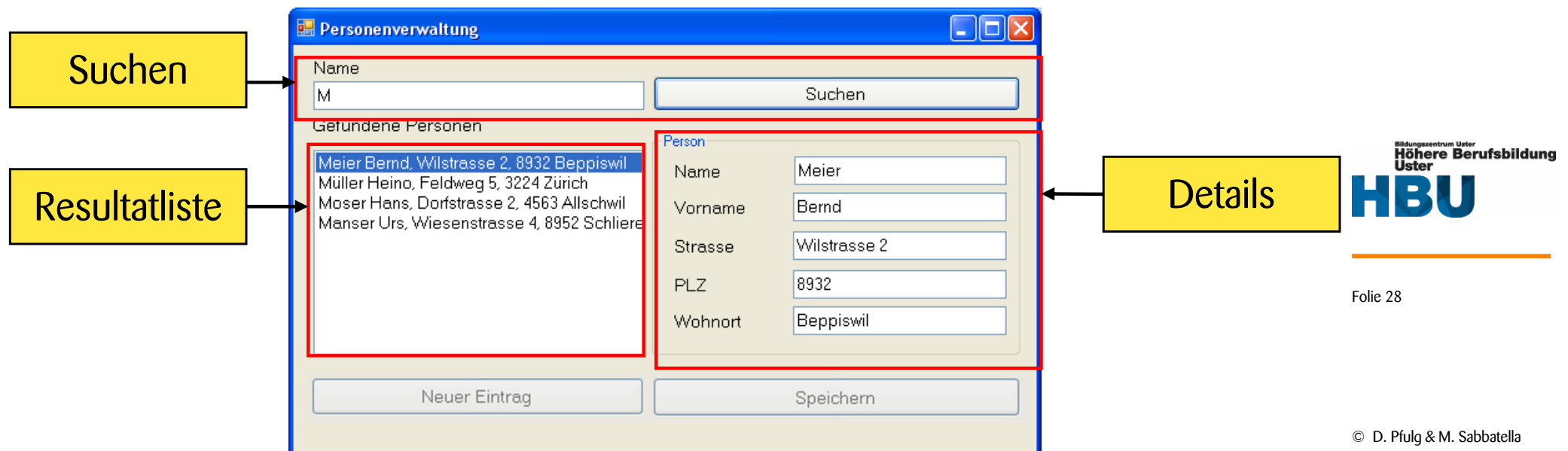
Record: 8 of 8

Übung 2.1



Personalverwaltung (2/3)

- A) Implementiere eine einfache Suche. Die Resultate sollen in einer Liste dargestellt werden.
- B) Wird ein Listeneintrag selektiert, so soll er auf der Detailansicht erscheinen.



Übung 2.1



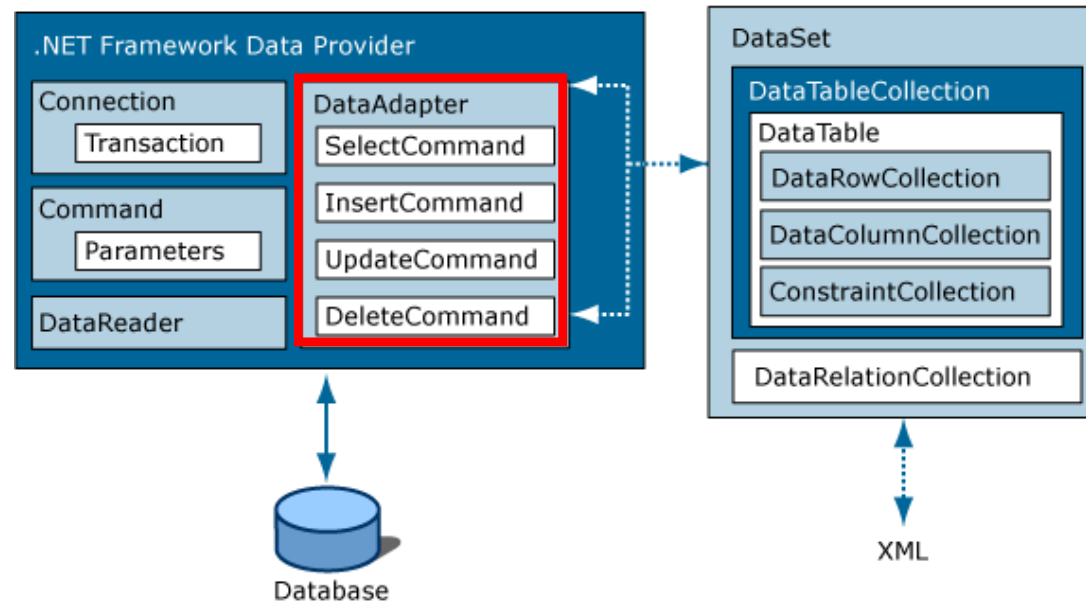
Personalverwaltung (3/3)

Erweitere die Applikation Personalverwaltung so, dass

- C) die Daten geändert werden können und beim Klicken auf den Button „Speichern“ die Daten in der Datenbank aktualisiert werden.
- D) ein neuer Eintrag erstellt oder ein bestehender gelöscht werden kann.

DataAdapter

DataAdapter



Klasse	Beschreibung
DataAdapter	Repräsentiert ein Set von SQL Kommandos und eine Datenbank Verbindung, welche benutzt werden, um ein Dataset zu füllen, sowie die Datenbank zu aktualisieren. Die Basisklasse ist DbDataAdapter.

Die SQL-Kommando-Eigenschaften

- Der DataAdapter kann mit vier SqlCommands parametrisiert werden.

Property	Beschreibung
DeleteCommand	Repräsentiert das Kommando, um Daten in der Datenquelle zu löschen. (SQL Kommando oder Stored Procedure)
InsertCommand	Repräsentiert das Kommando, um Daten in der Datenquelle einzufügen. (SQL Kommando oder Stored Procedure)
SelectCommand	Repräsentiert das Kommando, um Daten von der Datenquelle abzufragen. (SQL Kommando oder Stored Procedure)
UpdateCommand	Repräsentiert das Kommando, um Daten in der Datenquelle zu aktualisieren. (SQL Kommando oder Stored Procedure)

Ein DataAdapter instanziiieren

- Beispielcode SqlDataAdapter instanziiieren und Select-Befehl zuweisen:

```
string queryString = "SELECT * FROM Products";  
SqlDataAdapter adapter = new SqlDataAdapter();  
adapter.SelectCommand = new SqlCommand(queryString, connection);
```


Methoden, um ein DataSet zu füllen

Methode	Beschreibung
Fill(DataSet)	Aktualisiert oder fügt neue Zeilen im DataSet ein.
Fill(DataTable)	Aktualisiert oder fügt neue Zeilen im DataSet ein, abhängig vom spezifizierten Tabellennamen.

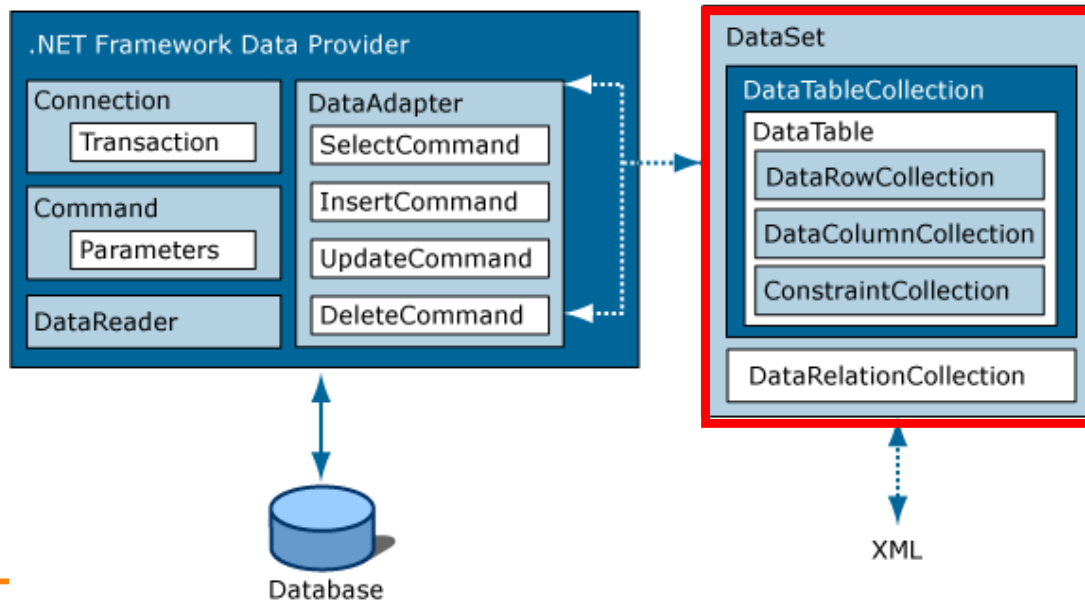
Ein DataSet abfüllen

■ Beispielcode mit SqlDataAdapter:

```
private static DataSet SelectRows(DataSet dataset,
    string connectionString, string queryString)
{
    using (SqlConnection connection =
        new SqlConnection(connectionString))
    {
        SqlDataAdapter adapter = new SqlDataAdapter();
        adapter.SelectCommand = new SqlCommand(
            queryString, connection);
        adapter.Fill(dataset);
        return dataset;
    }
}
```

Dataset Überblick

- Ein DataSet stellt einen offline Daten-Cache im Speicher dar. Die gespeicherten Daten werden in 1-n Tabellen verwaltet. Innerhalb eines DataSets kann eine ganze Datenbankstruktur aufgebaut werden.
- Ein DataSet kann in eine XML-Datei geschrieben, resp. aus einer XML-Datei abgefüllt werden.
- Ein DataSet ermöglicht eine automatische Datenbindung mit Windows Forms Controls.



DataSet

Namensraum

```
using System.Data
```

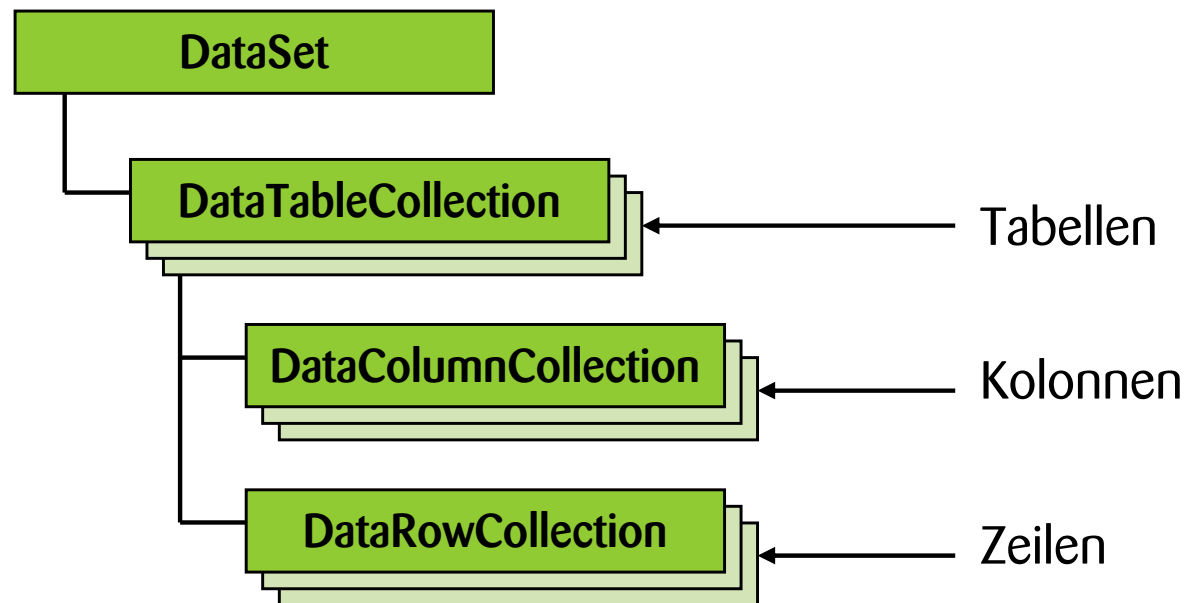
DataSet instanziiieren

```
DataSet dsPersons = new DataSet();
```

DataSet füllen

```
string strSQL = "SELECT * FROM Persons";  
SqlDataAdapter daPersons = new SqlDataAdapter(strSQL, con);  
DataSet dsPersons = new DataSet();  
daPersons.Fill(dsPersons);
```

DataSet-Struktur



Data-binding

- Ein DataSet ermöglicht eine automatische Datenbindung mit Windows Forms Controls, z.B. DataGridView. Die Änderungen werden über das Control direkt an die Tabellen im DataSet weitergegeben. Dazu wird die Eigenschaft `DataSource` verwendet.

Schritte zur Verwaltung von DataSets

- Eine typische multi-tier Applikation enthält die folgenden Schritte, um ein DataSet zu erstellen, verändern sowie die Originaldaten zu aktualisieren:

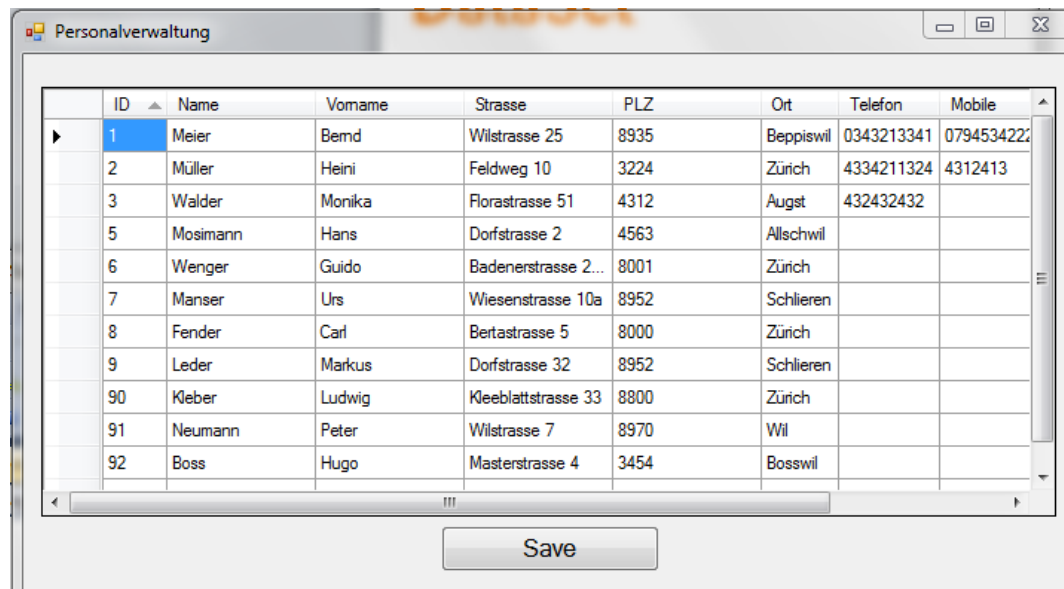
Schritt	Beschreibung
1.	Instanziiere und fülle jedes DataTable-Objekt mit Daten aus der Datenquelle mit einem DataAdapter.
2.	Verändere die Daten in den DataTable-Objekten durch hinzufügen, aktualisieren oder löschen von DataRow-Objekten.
3.	Rufe die Methode GetChanges auf, damit ein "Update"-DataSet erstellt wird welches nur die Änderungen enthält.
4.	Rufe die Update-Methode des DataAdapters auf mit dem "Update"-DataSet als Argument.
5.	Rufe die Merge-Methode auf, um die Änderungen im "Update" DataSet mit dem Original-DataSet zu vereinigen.
6.	Rufe die Methode AcceptChanges des DataSet auf, um die Änderungen zu akzeptieren oder die Methode RejectChanges, um die Änderungen zu verwerfen.

Übung 2.2



Personalverwaltung mit DataSet und Databinding

- Schreibe eine zur Übung 2.1 analoge «Windows Forms»-Applikation, um Mitarbeiterdaten in einem GridView zu verwalten
- Implementiere die Lösung mit einem DataSet und einem GridView. Ein Save-Button soll die Daten in der Datenbank aktualisieren.



Codebeispiel zu Übung 2.2: GridView, DataSet und Binding (1/2)

```
public partial class PersonalForm : Form
{
    OleDbDataAdapter dAdapter;
    DataTable personTable;

    public PersonalForm()
    {
        InitializeComponent();
        ReadData();
    }

    private string GetConnString()
    {
        string connStr = @"Provider=Microsoft.Jet.OLEDB.4.0;Data" +
            @"Source=C:\Users\sab\Documents\60 HFU\2010\10 Exercises\13 ADO.NET\PersonenVerwaltung.mdb;" +
            @"Persist Security Info=False";
        return connStr;
    }
}
```

Codebeispiel zu Übung 2.2: GridView, DataSet und Binding (2/2)

```
private void ReadData()
{
    string query = "SELECT * FROM Persons";
    //create an OleDbDataAdapter to execute the query
    dAdapter = new OleDbDataAdapter(query, GetConnString());

    //create a command builder
    OleDbCommandBuilder cBuilder = new OleDbCommandBuilder(dAdapter);

    //create a DataTable to hold the query results
    personTable = new DataTable();
    |
    //fill the DataTable
    dAdapter.Fill(personTable);

    //BindingSource to sync DataTable and DataGridView
    BindingSource bSource = new BindingSource();

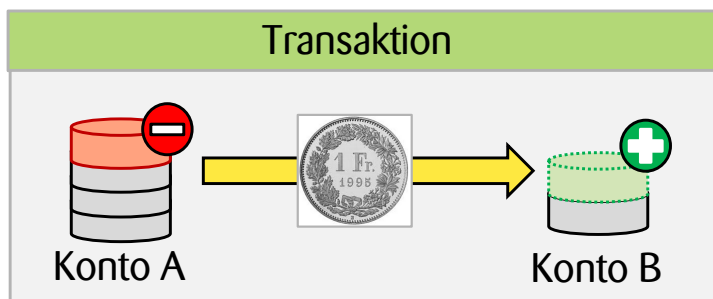
    //set the BindingSource DataSource
    bSource.DataSource = personTable;

    //set the DataGridView DataSource
    dgPersons.DataSource = bSource;
}

private void btnSave_Click(object sender, EventArgs e)
{
    dAdapter.Update(personTable);
}
}
```

Was ist eine DB-Transaktion?

- Transaktionen werden benötigt, wenn 2 bis n SQL-Befehle miteinander erfolgreich ausgeführt werden sollen. Falls ein Befehl fehlschlägt, sollen alle anderen rückgängig gemacht werden. Ein klassisches Beispiel ist die Transaktion im Finanzumfeld. Dort spricht man von einer Finanztransaktion.
- Beispiel ist die Überweisung eines Betrags von einem Konto A auf ein Konto B. Für eine korrekte Buchführung müssen beide Operationen erfolgreich ausgeführt werden. Wenn eine Operation fehlschlägt, ist die gesamte Transaktion ungültig. Demzufolge müssen alle Operationen rückgängig gemacht werden.



```
!-- TRANSACTION START
UPDATE Account SET Balance = Balance - 1 WHERE Customer = 'A';
UPDATE Account SET Balance = Balance + 1 WHERE Customer = 'B';
!-- END
```

SQL

Transaktion muss ACID sein

- **Atomic** - Alle Befehle müssen erfolgreich sein oder keiner darf ausgeführt werden
- **Consistent** – Eine Gruppe von SQL-Befehlen müssen die Datenbank von einem gültigen Startstatus in einen gültigen Endstatus bringen
- **Isolated** – Eine Gruppe von SQL-Befehlen muss unabhängig von anderen gleichzeitig ausgeführten sein.
- **Durable** – Nachdem eine Gruppe von SQL-Befehlen erfolgreich ausgeführt wurde, müssen die Änderungen permanent sein. => Transaction Commit

Datenbanktransaktionen

Codebeispiel

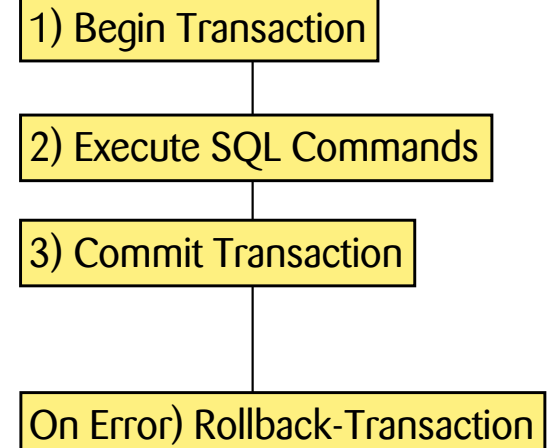
```
public void TransactionWithTryCatch(SqlDataAdapter da, SqlConnection connection, DataSet ds)
{
    SqlCommand command = new SqlCommand();
    command.CommandText = "INSERT INTO"; // SQL Kommando ...

    connection.Open();
    SqlTransaction transaction = connection.BeginTransaction();
    command.Transaction = transaction;

    try
    {
        da.Update(ds);
        command.Transaction.Commit(); // Transaktion abschliessen
    }
    catch (Exception exception)
    {
        try
        {
            transaction.Rollback(); // Transaktion rückgängig machen
        }
        catch (SqlException ex) // Transaktion Rollback fehlgeschlagen
        {
            if (transaction.Connection != null)
            {
                Console.WriteLine("An exception of type " + ex.GetType() +
                    " was encountered while attempting to roll back the transaction.");
            }
        }

        Console.WriteLine(exception.ToString()); // Transaktion fehlgeschlagen
    }

    connection.Close();
}
```



Datenbanktransaktionen

Codebeispiel mit using {}

```
void TransactionWithUsing()
{
    using (SqlConnection connection = new SqlConnection(ConfigurationManager.AppSettings["MyDB"]))
    {
        try
        {
            string sqlCommandText = "INSERT INTO"; // SQL Kommando ...
            SqlCommand command = new SqlCommand(sqlCommandText, connection);
            connection.Open();
            using (SqlTransaction transaction = connection.BeginTransaction())
            {
                command.ExecuteNonQuery();
                transaction.Commit();
            }
        }
        catch (SqlException exception)
        {
            Console.WriteLine(exception.ToString()); // Transaktion fehlgeschlagen
        }
    }
}
```

1) Begin Transaction

2) Execute SQL Commands

3) Commit Transaction

On Error: Rollback Transaction

Datenbanktransaktionen

Transaktion – Isolationsstufen (Isolation Levels)

```
public enum IsolationLevel
```

Isolation Level	Beschreibung
Chaos	Ausstehende Änderungen von stärker isolierten Transaktion können nicht überschrieben werden.
ReadCommitted	Daten können während der Transaktion nicht gelesen, jedoch geändert werden.
ReadUncommitted	Daten können während der Transaktion gelesen und geändert werden.
RepeatableRead	Daten lesen, aber nicht während der Transaktion geändert werden.
Serializable	Daten können während der Transaktion gelesen, aber nicht geändert werden. Ebenfalls können keine neuen Daten hinzugefügt werden.
Snapshot	Daten können gelesen werden. Bevor eine Transaktion Daten ändert, wird überprüft, ob eine andere Transaktion die Daten geändert hat, nachdem sie ursprünglich gelesen wurden. Falls die Daten aktualisiert wurden, wird ein Fehler ausgelöst. Dies kann eine Transaktion auf den zuvor übernommenen Wert der Daten zurücksetzen.
Unspecified	Die Isolationsstufe ist nicht spezifiziert/angegeben.

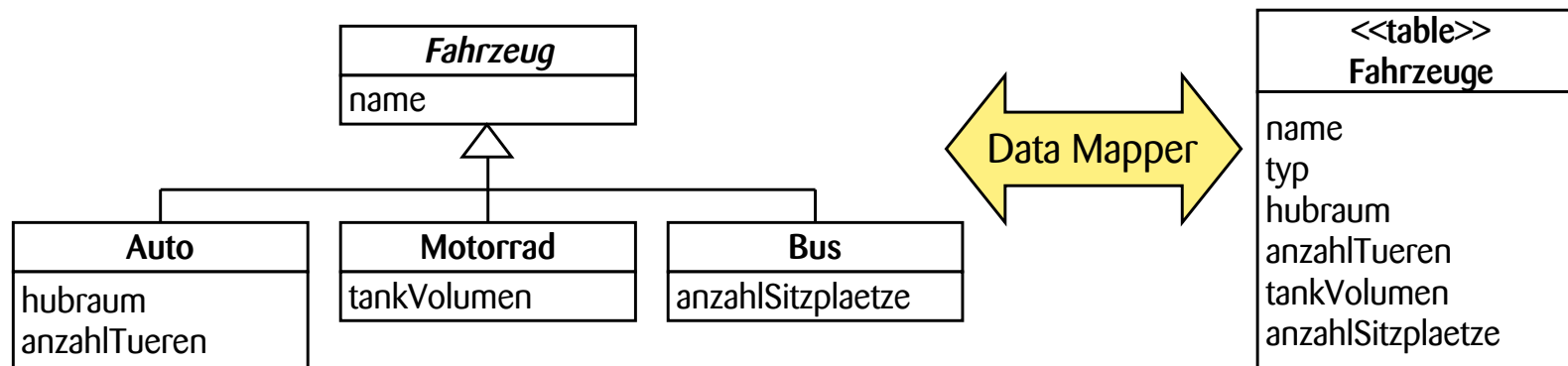
```
connection.Open();  
SqlTransaction transaction = connection.BeginTransaction(IsolationLevel.ReadCommitted);  
command.Transaction = transaction;
```

Entity Framework

Grundlagen

■ O/R-Mapper

- Ein O/R-Mapper unterstützt die Transformation von Objekten (Applikation) nach relational gespeicherten Daten (Datenbanktabellen) und umgekehrt.
- O/R-Mapper verbessern die Effizienz bei der Entwicklung von Applikationen in Verbindung mit Datenbanken



Abbildungsszenarien O/R

■ Objektidentität

- Entitäten im Speicher müssen den Datensätzen in der DB eindeutig zugeordnet werden können
- Kombinierte Schlüssel verkomplizieren das Abbilden
- GUID können als künstliche Schlüssel verwendet werden, vergrößert aber die Index-Bildung
- Wo/wie wird der Schlüssel generiert?
 - Datenbank:
 - Identity-Feld bei SQL Server wird erst beim Einfügen in die DB generiert
 - ID mit einer Sequenz generieren
 - Eigene Transaktion

Abbildungsszenarien O/R

■ Beziehungen

- Zwischen Objekten:
 - 1:1 sind unidirektional, speichern der Objektreferenz
 - 1:n werden mit Collections gebildet
 - n:m werden durch zwei 1:n-Beziehungen gebildet
- In der DB:
 - 1:1 sind mit Primär- und Fremdschlüssel und Constraints abgebildet
 - 1:n sind mit Primär- und Fremdschlüssel abgebildet
 - n:m werden durch eine zusätzliche Mapping-Tabelle mit zwei Fremdschlüsseln gebildet

Abbildungsszenarien O/R

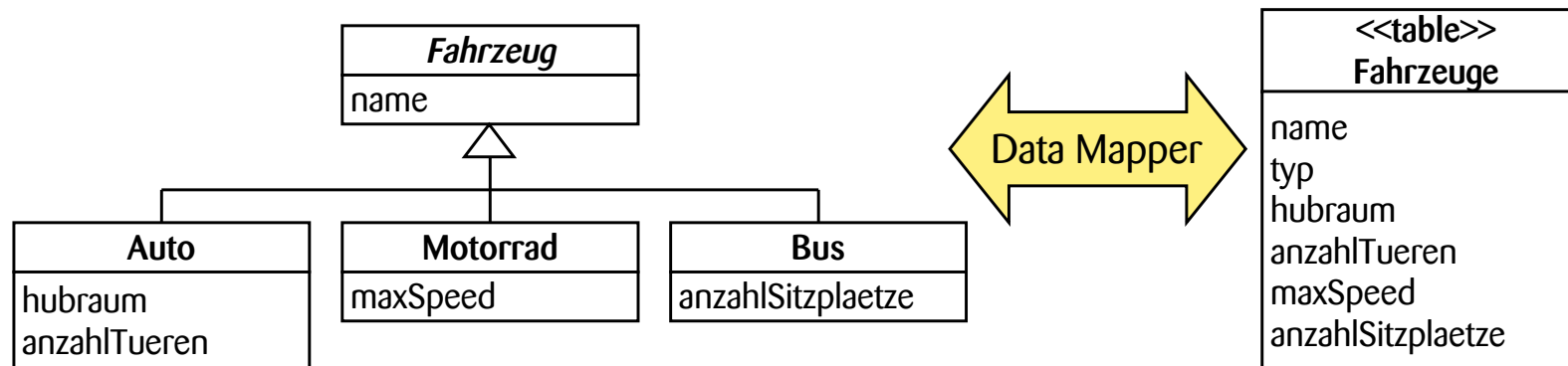
■ Vererbung

- Im relationalen Datenmodell sind keine Vererbungen vorgesehen!
- Folgende Strategien ermöglichen das Abbilden:
 - Vererbungshierarchie in einer Tabelle
 - Vererbung mit einer Tabelle pro Klasse
 - Vererbung mit einer Tabelle pro konkreter Klasse

Entity Framework

Abbildungsszenarien O/R

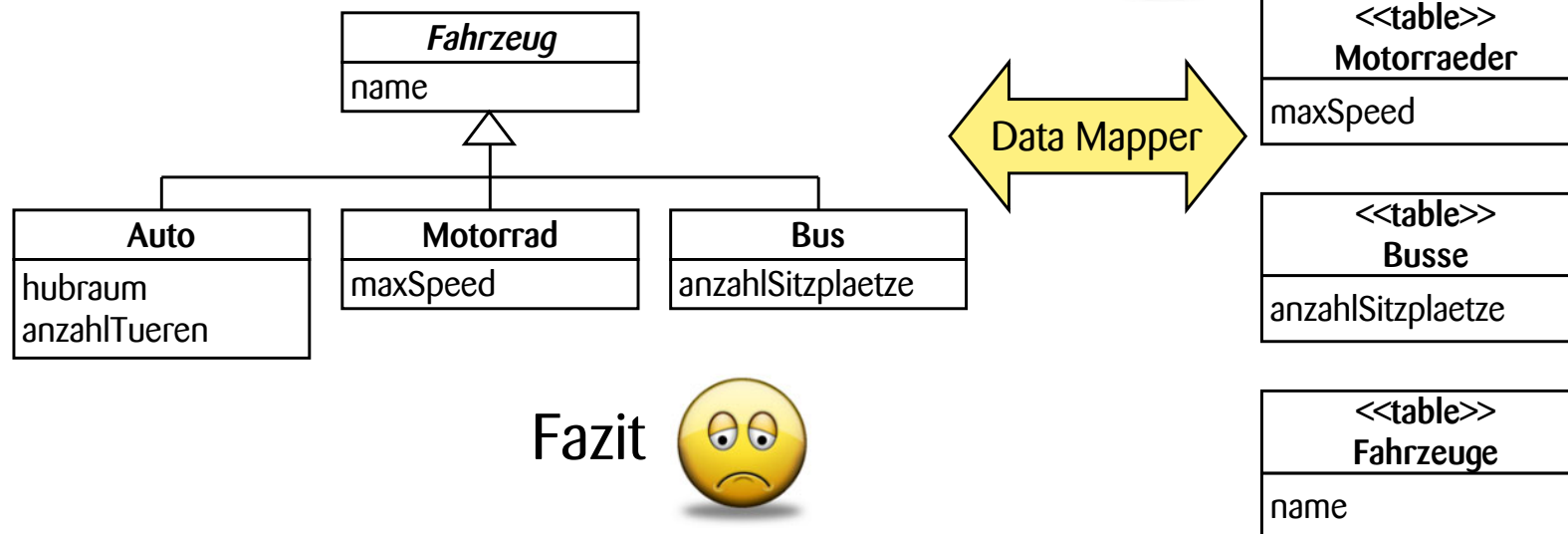
- Vererbungshierarchie in einer Tabelle (Single Table Inheritance)
 - Alle Objekte werden in einer Tabelle abgebildet
 - „Typ-Feld“ definiert den konkreten Objekttyp
 - Nur die entsprechenden Felder des Objekttyps werden gefüllt
 - Beim Laden wird kein JOIN benötigt -> effizient 😎



Entity Framework

Abbildungsszenarien O/R

- Vererbung mit einer Tabelle pro Klasse (Class Table Inheritance)
 - Für jede Klasse eine Tabelle in der Datenbank
 - Keine ungenutzten Spalten
 - Beim Laden müssen mit JOIN die Daten aus mehreren Tabellen zusammengesetzt werden
 - > Ab 3 Tabellen Einbussen bei der Performance 😞



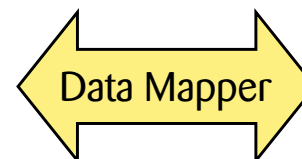
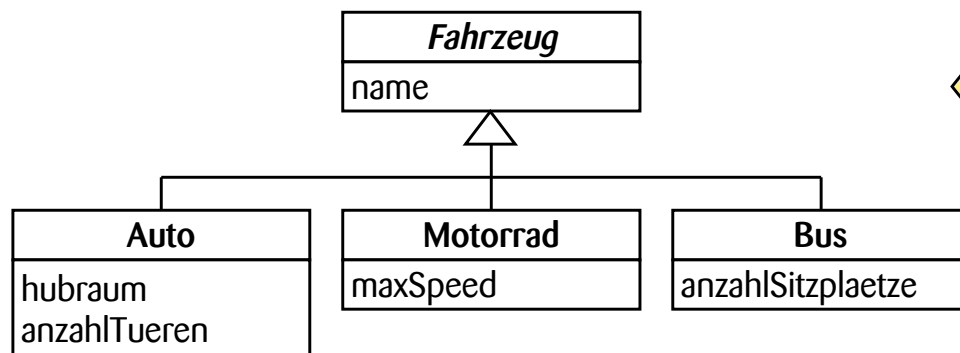
Fazit



Entity Framework

Abbildungsszenarien O/R

- Vererbung mit einer Tabelle pro konkreter Klasse (Concrete Table Inheritance)
 - Für jede konkrete Klasse eine Tabelle in der Datenbank
 - Keine ungenutzten Spalten
 - Alle Eigenschaften der konkreten Klasse und Superklassen als Felder
 - Beim Laden müssen keine JOIN verwendet werden
 - Bei Abfragen auf die Basisklasse sind UNIONS notwendig (z.B. alle Namen von Autos und Motorraeder ermitteln)



<<table>> Autos
name
hubraum
anzahlTueren

<<table>> Motorraeder
name
maxSpeed

<<table>> Busse
name
anzahlSitzplaetze

Entity Framework

Abbildungsszenarien O/R

- Arbeitseinheit (Unit of Work)
 - Design Pattern „Unit of Work“ (Ref: [Patterns of Enterprise Application Architecture, Martin Fowler](#))
 - Übernimmt die Verwaltung der Datenmanipulation, die Aktualisierung in der Datenbank und die Handhabung der Nebenläufigkeit (Concurrency)
 - Verwaltet eine Liste von geänderten Objekten in einer Business-Transaktion

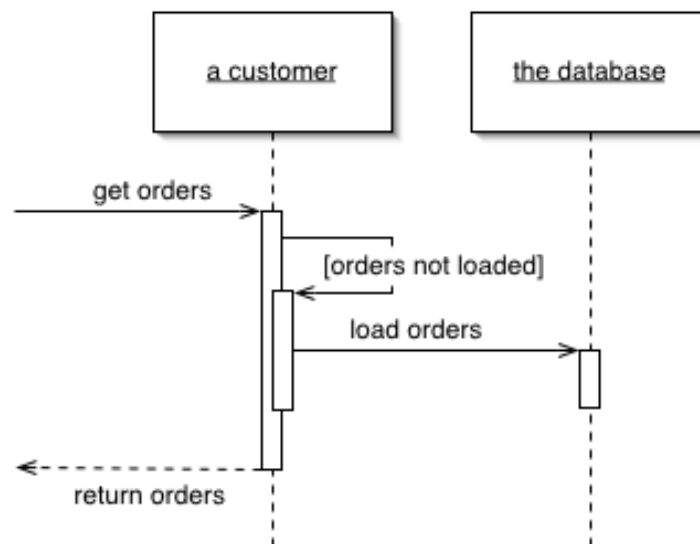
Unit of Work
registerNew(object) registerDirty(object) registerClean(object) registerDeleted(object) commit rollback

Entity Framework

Abbildungsszenarien O/R

■ Laden bei Bedarf (Lazy Loading)

- Nicht die ganze Datenbank auslesen, nur was wirklich benötigt wird.
- Design Pattern „Lazy Load“ (Ref: [Patterns of Enterprise Application Architecture, Martin Fowler](#))
- Ein Objekt, welches nicht alle Daten enthält, aber weiss, wie es die laden kann.



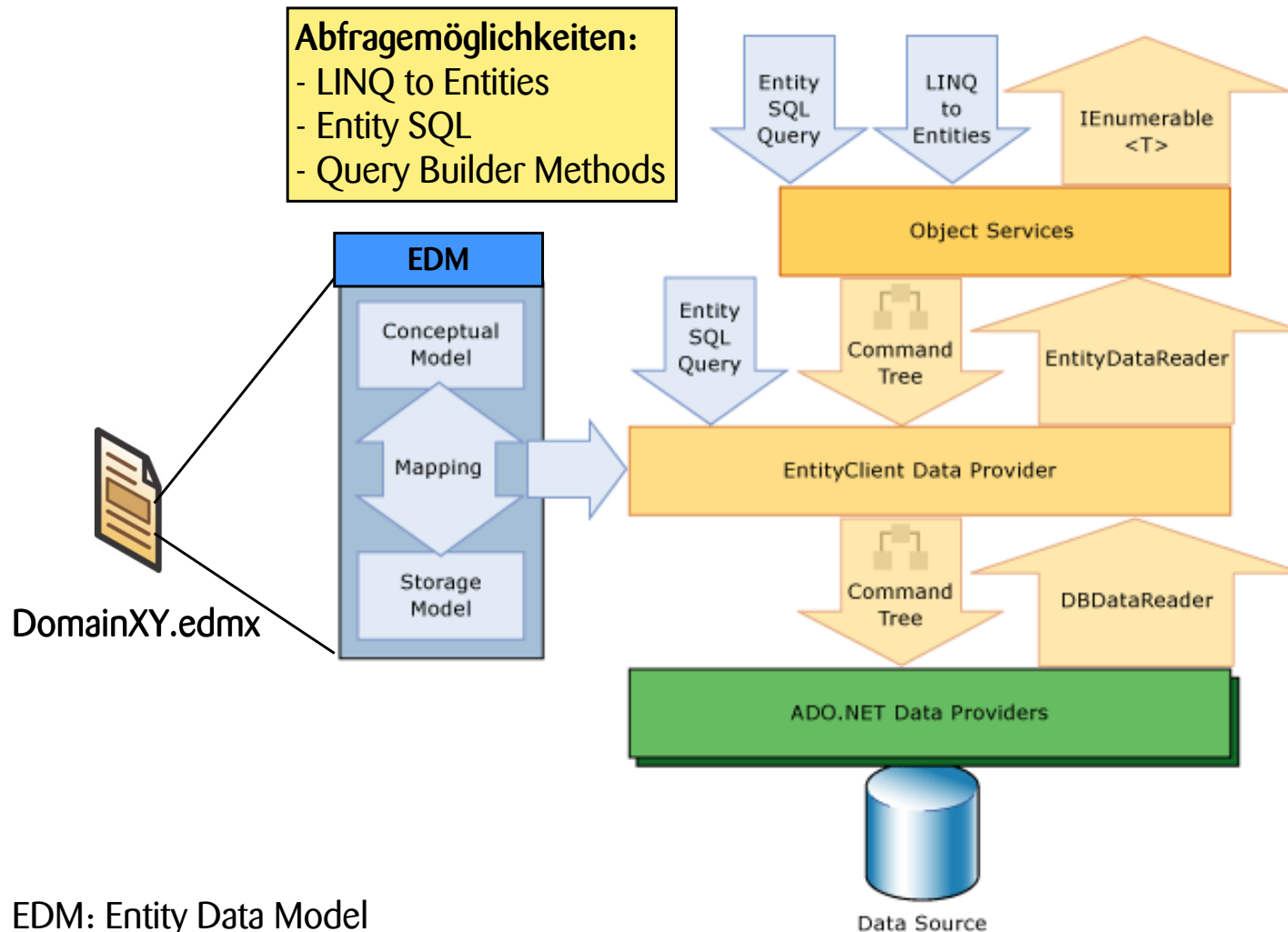
Abbildungsszenarien O/R

■ Nebenläufigkeit (Concurrency)

- Situation:
Alain und Bruno lesen die gleiche Zeile aus der Tabelle *Customers*. Beide ändern die Daten und versuchen ihre Version in die Datenbank zu schreiben. Welche Änderungen sollen gespeichert werden? Alains? Brunos? Keine? Eine Kombination?
- Strategien, um dieses Problem zu lösen:
 - Pessimistic Locking
 - Datensatz in der Datenbank sperren solange die Objekte im Speicher sind
 - Optimistic Locking
 - Datensatz im Speicher ändern und hoffen, dass niemand in der Zwischenzeit ihn ändert.
 - Datensatz in der Datenbank nur beim Aktualisieren sperren

Entity Framework

Architekturübersicht



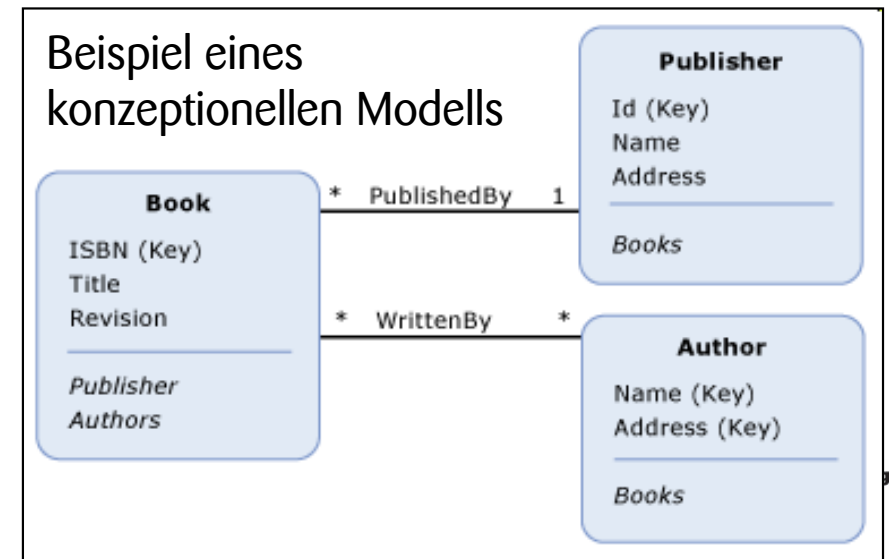
Architekturkomponenten

- LINQ-to-Entities
 - Sprache, um Abfragen aufs Objekt Model auszuführen.
Resultat sind Entitäten, welche im konzeptionellen Model definiert sind
- Entity SQL
 - Ist eine weitere Abfragesprache
- Object Service
 - Verarbeitet die Abfragen und liefert die Resultate als Objekte
- Entity Client Data Provider
 - Transformiert LINQ-to-Entities und Entity Queries in SQL Abfragen für die Datenbank
- ADO.NET Data Provider
 - Kommuniziert mit der DB via Standard ADO.NET-Framework

Entity Framework

Architektur – Modellbeschreibung

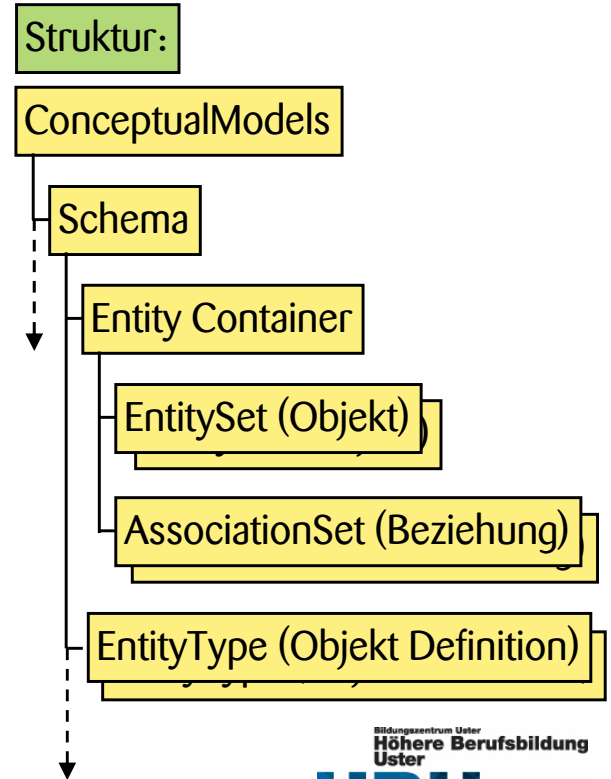
- Conceptual schema definition language (CSDL)
 - Die CSDL definiert das konzeptionale Modell (conceptual model) und ist die Umsetzung des Entity Data Model.
Dateierweiterung: .csdl
- Store schema definition language (SSDL)
 - Die SSDL definiert das Daten Modell (storage model).
Dateierweiterung: .ssdl
- Mapping specification language (MSL)
 - Die MSL definiert die Transformation (mapping) zwischen dem Datenmodell und dem konzeptionellen Modell.
Dateierweiterung : .msl



Entity Framework

Conceptual schema def. lang. (CSDL) - Ausschnitt

```
<!-- CSDL content -->
<edmx:ConceptualModels>
  <Schema Namespace="SchoolModel" Alias="Self" xmlns:annotation="http://schemas.microsoft.com/ado/2009/02/edm/annotation"
    <EntityContainer Name="SchoolEntities" annotation:LazyLoadingEnabled="true">
      <EntitySet Name="Courses" EntityType="SchoolModel.Course" />
      <EntitySet Name="Departments" EntityType="SchoolModel.Department" />
      <EntitySet Name="OfficeAssignments" EntityType="SchoolModel.OfficeAssignment" />
      <EntitySet Name="OnlineCourses" EntityType="SchoolModel.OnlineCourse" />
      <EntitySet Name="OnsiteCourses" EntityType="SchoolModel.OnsiteCourse" />
      <EntitySet Name="People" EntityType="SchoolModel.Person" />
      <EntitySet Name="StudentGrades" EntityType="SchoolModel.StudentGrade" />
      <EntitySet Name="sysdiagrams" EntityType="SchoolModel.sysdiagram" />
      <AssociationSet Name="FK_Course_Department" Association="SchoolModel.FK_Course_Department">
        <End Role="Department" EntitySet="Departments" />
        <End Role="Course" EntitySet="Courses" />
      </AssociationSet>
      <AssociationSet Name="FK_OnlineCourse_Course" Association="SchoolModel.FK_OnlineCourse_Course">
      <AssociationSet Name="FK_OnsiteCourse_Course" Association="SchoolModel.FK_OnsiteCourse_Course">
      <AssociationSet Name="FK_StudentGrade_Course" Association="SchoolModel.FK_StudentGrade_Course">
      <AssociationSet Name="FK_OfficeAssignment_Person" Association="SchoolModel.FK_OfficeAssignment_Person">
      <AssociationSet Name="FK_StudentGrade_Student" Association="SchoolModel.FK_StudentGrade_Student">
      <AssociationSet Name="CourseInstructor" Association="SchoolModel.CourseInstructor">
    </EntityContainer>
    <EntityType Name="Course">
      <Key>
        <PropertyRef Name="CourseID" />
      </Key>
      <Property Name="CourseID" Type="Int32" Nullable="false" />
      <Property Name="Title" Type="String" Nullable="false" MaxLength="100" Unicode="true" FixedLength="false" />
      <Property Name="Credits" Type="Int32" Nullable="false" />
      <Property Name="DepartmentID" Type="Int32" Nullable="false" />
      <NavigationProperty Name="Department" Relationship="SchoolModel.FK_Course_Department" FromRole="Course" ToRole="Dej
      <NavigationProperty Name="OnlineCourse" Relationship="SchoolModel.FK_OnlineCourse_Course" FromRole="Course" ToRole:
      <NavigationProperty Name="OnsiteCourse" Relationship="SchoolModel.FK_OnsiteCourse_Course" FromRole="Course" ToRole:
      <NavigationProperty Name="StudentGrades" Relationship="SchoolModel.FK_StudentGrade_Course" FromRole="Course" ToRol
      <NavigationProperty Name="People" Relationship="SchoolModel.CourseInstructor" FromRole="Course" ToRole="Person" />
    </EntityType>
```



Entity Framework

Store schema definition language (SSDL) - Ausschnitt

```
<!-- SSDL content -->
<edmx:StorageModels>
  <Schema Namespace="SchoolModel.Store" Alias="Self" Provider="System.Data.SqlClient" ProviderManifestToken=
  <EntityContainer Name="SchoolModelStoreContainer">
    <EntityType Name="Course" EntityType="SchoolModel.Store.Course" store:Type="Tables" Schema="dbo" />
    <EntityType Name="CourseInstructor" EntityType="SchoolModel.Store.CourseInstructor" store:Type="Tables"
    <EntityType Name="Department" EntityType="SchoolModel.Store.Department" store:Type="Tables" Schema="dbo
    <EntityType Name="OfficeAssignment" EntityType="SchoolModel.Store.OfficeAssignment" store:Type="Tables"
    <EntityType Name="OnlineCourse" EntityType="SchoolModel.Store.OnlineCourse" store:Type="Tables" Schema=
    <EntityType Name="OnsiteCourse" EntityType="SchoolModel.Store.OnsiteCourse" store:Type="Tables" Schema=
    <EntityType Name="Person" EntityType="SchoolModel.Store.Person" store:Type="Tables" Schema="dbo" />
    <EntityType Name="StudentGrade" EntityType="SchoolModel.Store.StudentGrade" store:Type="Tables" Schema=
    <EntityType Name="sysdiagrams" EntityType="SchoolModel.Store.sysdiagrams" store:Type="Tables" Schema="d
    <AssociationSet Name="FK_Course_Department" Association="SchoolModel.Store.FK_Course_Department">
      <End Role="Department" EntityType="Department" />
      <End Role="Course" EntityType="Course" />
    </AssociationSet>
    <AssociationSet Name="FK_CourseInstructor_Course" Association="SchoolModel.Store.FK_CourseInstructor_C
    <AssociationSet Name="FK_CourseInstructor_Person" Association="SchoolModel.Store.FK_CourseInstructor_P
    <AssociationSet Name="FK_OfficeAssignment_Person" Association="SchoolModel.Store.FK_OfficeAssignment_P
    <AssociationSet Name="FK_OnlineCourse_Course" Association="SchoolModel.Store.FK_OnlineCourse_Course">
    <AssociationSet Name="FK_OnsiteCourse_Course" Association="SchoolModel.Store.FK_OnsiteCourse_Course">
    <AssociationSet Name="FK_StudentGrade_Course" Association="SchoolModel.Store.FK_StudentGrade_Course">
    <AssociationSet Name="FK_StudentGrade_Student" Association="SchoolModel.Store.FK_StudentGrade_Student"
  </EntityContainer>
  <EntityType Name="Course">
    <Key>
      <PropertyRef Name="CourseID" />
    </Key>
    <Property Name="CourseID" Type="int" Nullable="false" />
    <Property Name="Title" Type="nvarchar" Nullable="false" MaxLength="100" />
    <Property Name="Credits" Type="int" Nullable="false" />
    <Property Name="DepartmentID" Type="int" Nullable="false" />
  </EntityType>
  <EntityType Name="CourseInstructor">
```

Struktur:

StorageModels

Schema

Entity Container

EntitySet (Tabelle)

AssociationSet (Beziehung)

EntityType (Tabelle-Felder)

Bildungszentrum Uster
Höhere Berufsbildung
Uster

HBU

Folie 62

© D. Pfulg & M. Sabbatella

Entity Framework

Mapping specification language (MSL) - Ausschnitt

```
<!-- C-S mapping content -->
<edmx:Mappings>
  <Mapping Space="C-S" xmlns="http://schemas.microsoft.com/ado/2008/09/mapping/cs">
    <EntityContainerMapping StorageEntityContainer="SchoolModelStoreContainer" CdmEntityContainer="SchoolEntities">
      <EntitySetMapping Name="Courses"><EntityTypeMapping TypeName="SchoolModel.Course"><MappingFragment StoreEntitySet="Course">
        <ScalarProperty Name="CourseID" ColumnName="CourseID" />
        <ScalarProperty Name="Title" ColumnName="Title" />
        <ScalarProperty Name="Credits" ColumnName="Credits" />
        <ScalarProperty Name="DepartmentID" ColumnName="DepartmentID" />
      </MappingFragment></EntityTypeMapping></EntitySetMapping>
      <EntitySetMapping Name="Departments"><EntityTypeMapping TypeName="SchoolModel.Department"><MappingFragment StoreEntitySet="Dep">
        <ScalarProperty Name="DepartmentID" ColumnName="DepartmentID" />
        <ScalarProperty Name="Name" ColumnName="Name" />
        <ScalarProperty Name="Budget" ColumnName="Budget" />
        <ScalarProperty Name="StartDate" ColumnName="StartDate" />
        <ScalarProperty Name="Administrator" ColumnName="Administrator" />
      </MappingFragment></EntityTypeMapping></EntitySetMapping>
      <EntitySetMapping Name="OfficeAssignments"><EntityTypeMapping TypeName="SchoolModel.OfficeAssignment"><MappingFragment StoreEr">
        <ScalarProperty Name="InstructorID" ColumnName="InstructorID" />
        <ScalarProperty Name="Location" ColumnName="Location" />
        <ScalarProperty Name="Timestamp" ColumnName="Timestamp" />
      </MappingFragment></EntityTypeMapping></EntitySetMapping>
      <EntitySetMapping Name="OnlineCourses"><EntityTypeMapping TypeName="SchoolModel.OnlineCourse"><MappingFragment StoreEntitySet="OnlineCourses">
      </EntitySetMapping>
      <EntitySetMapping Name="OnsiteCourses"><EntityTypeMapping TypeName="SchoolModel.OnsiteCourse"><MappingFragment StoreEntitySet="OnsiteCourses">
      </EntitySetMapping>
      <EntitySetMapping Name="People"><EntityTypeMapping TypeName="SchoolModel.Person"><MappingFragment StoreEntitySet="Person">
      </EntitySetMapping>
      <EntitySetMapping Name="StudentGrades"><EntityTypeMapping TypeName="SchoolModel.StudentGrade"><MappingFragment StoreEntitySet="StudentGrades">
      </EntitySetMapping>
      <EntitySetMapping Name="sysdiagrams"><EntityTypeMapping TypeName="SchoolModel.sysdiagram"><MappingFragment StoreEntitySet="sysdiagrams">
        <ScalarProperty Name="name" ColumnName="name" />
        <ScalarProperty Name="principal_id" ColumnName="principal_id" />
        <ScalarProperty Name="diagram_id" ColumnName="diagram_id" />
        <ScalarProperty Name="version" ColumnName="version" />
        <ScalarProperty Name="definition" ColumnName="definition" />
      </MappingFragment></EntityTypeMapping></EntitySetMapping>
      <AssociationSetMapping Name="CourseInstructor" TypeName="SchoolModel.CourseInstructor" StoreEntitySet="CourseInstructor">
        <EndProperty Name="Course">
          <ScalarProperty Name="CourseID" ColumnName="CourseID" />
        </EndProperty>
        <EndProperty Name="Person">
          <ScalarProperty Name="PersonID" ColumnName="PersonID" />
        </EndProperty>
      </AssociationSetMapping>
    </EntityContainerMapping>
  </Mapping>
</edmx:Mappings>
```

Struktur:

Mappings

Mapping

EntityContainerMapping

EntitySetMapping
(Tabelle – Objekt)

AssociationSetMapping
(BeziehungsMapping)

Entity Framework – Data Model Designer

WinFormEntityFramework - Microsoft Visual Studio

File Edit View Project Build Debug Team Data Tools Test Analyze Window Help

Debug OnCourseIDChanging

EntityObject [from metadata] School.Designer.cs School.edmx CourseViewer.cs CreateSchoolDatabase.sql CourseViewer.cs [Design]

Data Model Designer

```
graph TD
    Department -- "1" --> "*" Course
    Course -- "1" --> "0..1" OnlineCourse
    Course -- "1" --> "0..1" OnsiteCourse
    Course -- "1" --> "*" Person
    Person -- "1" --> "0..1" OfficeAssignment
    OfficeAssignment -- "1" --> "*" StudentGrade
    StudentGrade -- "1" --> "*" Course
```

Model Browser

Type here to search

- School.edmx
 - SchoolModel
 - Entity Types
 - Course
 - Department
 - OfficeAssignment
 - OnlineCourse
 - OnsiteCourse
 - Person
 - StudentGrade
 - sysdiagram
 - Complex Types
 - Associations
 - CourseInstructor
 - FK_Course_Department
 - FK_OfficeAssignment_Person
 - FK_OnlineCourse_Course
 - FK_OnsiteCourse_Course
 - FK_StudentGrade_Course
 - FK_StudentGrade_Student
 - EntityContainer: SchoolEntities
 - Entity Sets
 - Association Sets
 - Function Imports
 - SchoolModel.Store
 - Tables / Views
 - Stored Procedures
 - Constraints

Mapping Details - Department

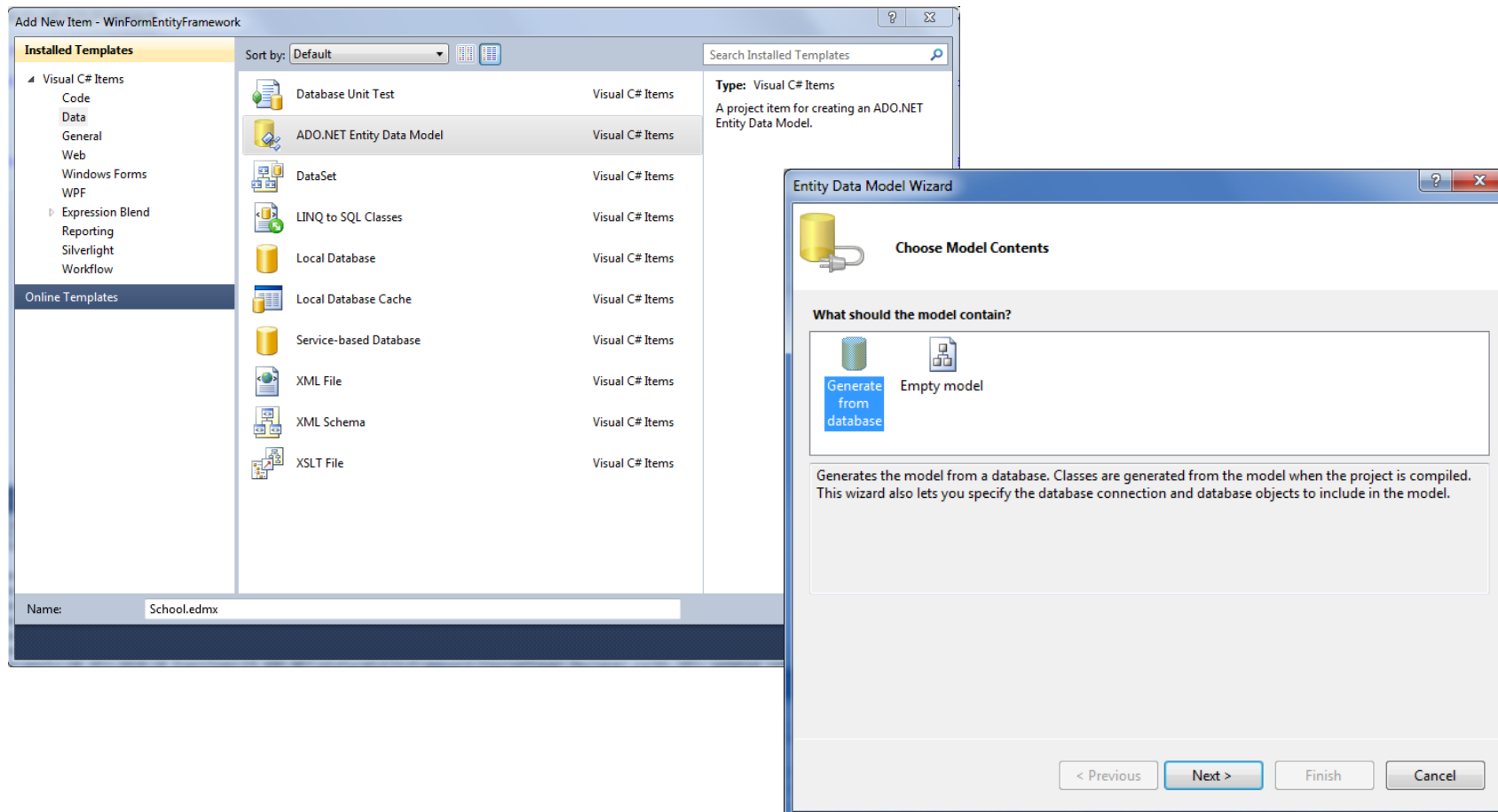
Column	Operator	Value / Property
Tables		
Maps to Department		
<Add a Condition>		
Column Mappings		
DepartmentID : int	↔	DepartmentID : Int32
Name : nvarchar	↔	Name : String
Budget : money	↔	Budget : Decimal
StartDate : datetime	↔	StartDate : DateTime
Administrator : int	↔	Administrator : Int32
<Add a Table or View>		

Mapping Details Table <-> Object

Entity Framework (Tutorial)

Entity Model Tutorial

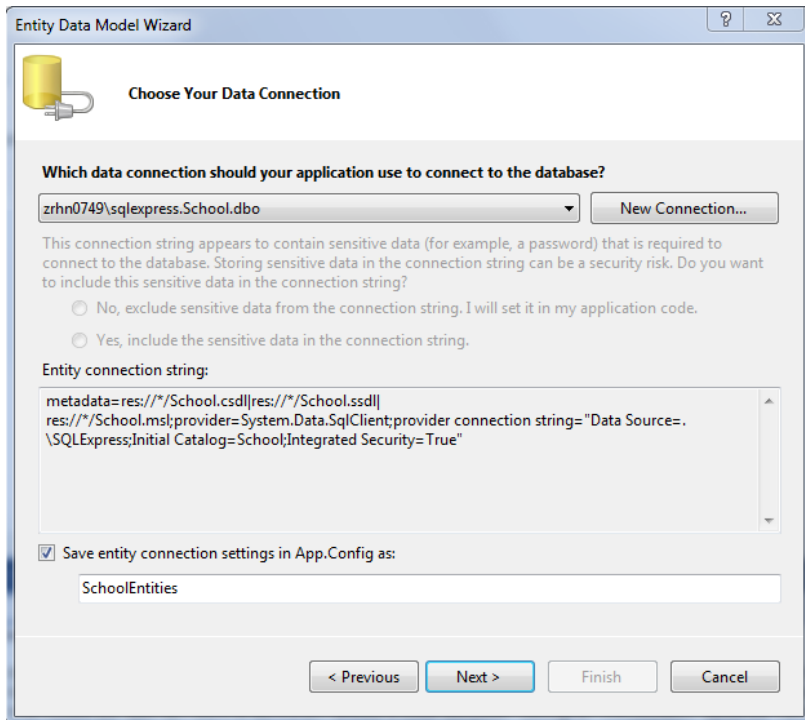
- ADO.NET Entity Data Model zum Projekt hinzufügen



Entity Framework (Tutorial)

Entity Data Model Tutorial

■ Database Connection und Objektauswahl



Entity Data Model Wizard

Choose Your Data Connection

Which data connection should your application use to connect to the database?

zrh0749\sql express.School.dbo New Connection...

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

☐ No, exclude sensitive data from the connection string. I will set it in my application code.

☐ Yes, include the sensitive data in the connection string.

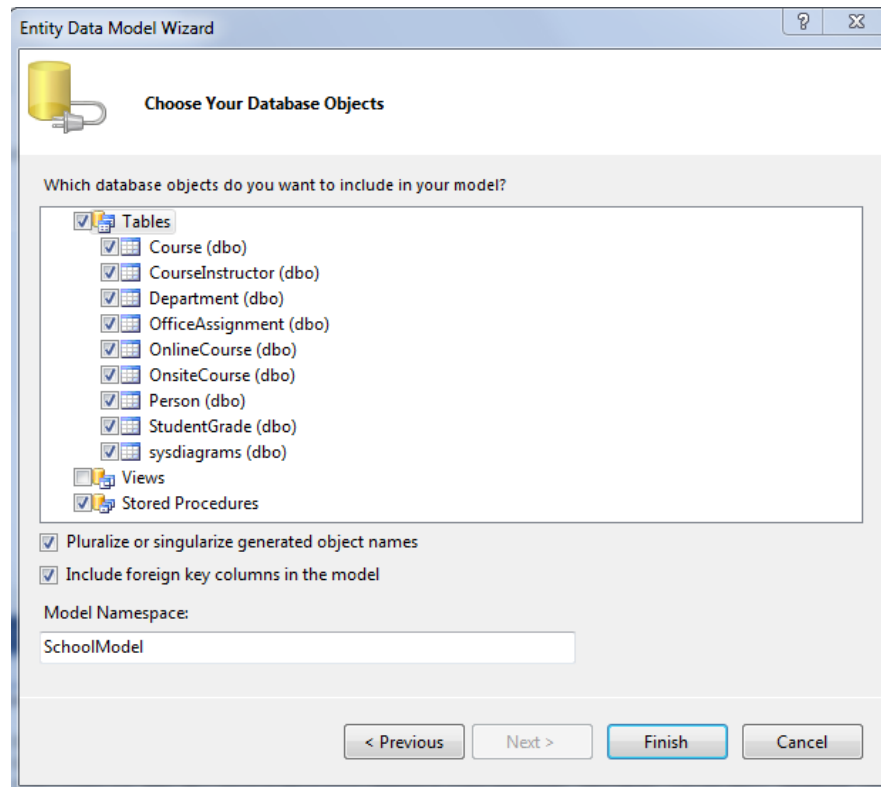
Entity connection string:

```
metadata=res://*/School.csdl|res://*/School.ssdl|
res://*/School.msl;provider=System.Data.SqlClient;provider connection string="Data Source=.
\SQLExpress;Initial Catalog=School;Integrated Security=True"
```

☒ Save entity connection settings in App.Config as:

SchoolEntities

< Previous Next > Finish Cancel



Entity Data Model Wizard

Choose Your Database Objects

Which database objects do you want to include in your model?

☒ Tables

- ☒ Course (dbo)
- ☒ CourseInstructor (dbo)
- ☒ Department (dbo)
- ☒ OfficeAssignment (dbo)
- ☒ OnlineCourse (dbo)
- ☒ OnsiteCourse (dbo)
- ☒ Person (dbo)
- ☒ StudentGrade (dbo)
- ☒ sysdiagrams (dbo)

☐ Views

☒ Stored Procedures

☒ Pluralize or singularize generated object names

☒ Include foreign key columns in the model

Model Namespace:

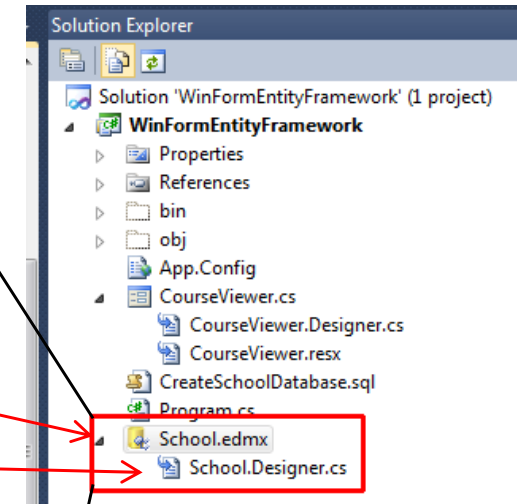
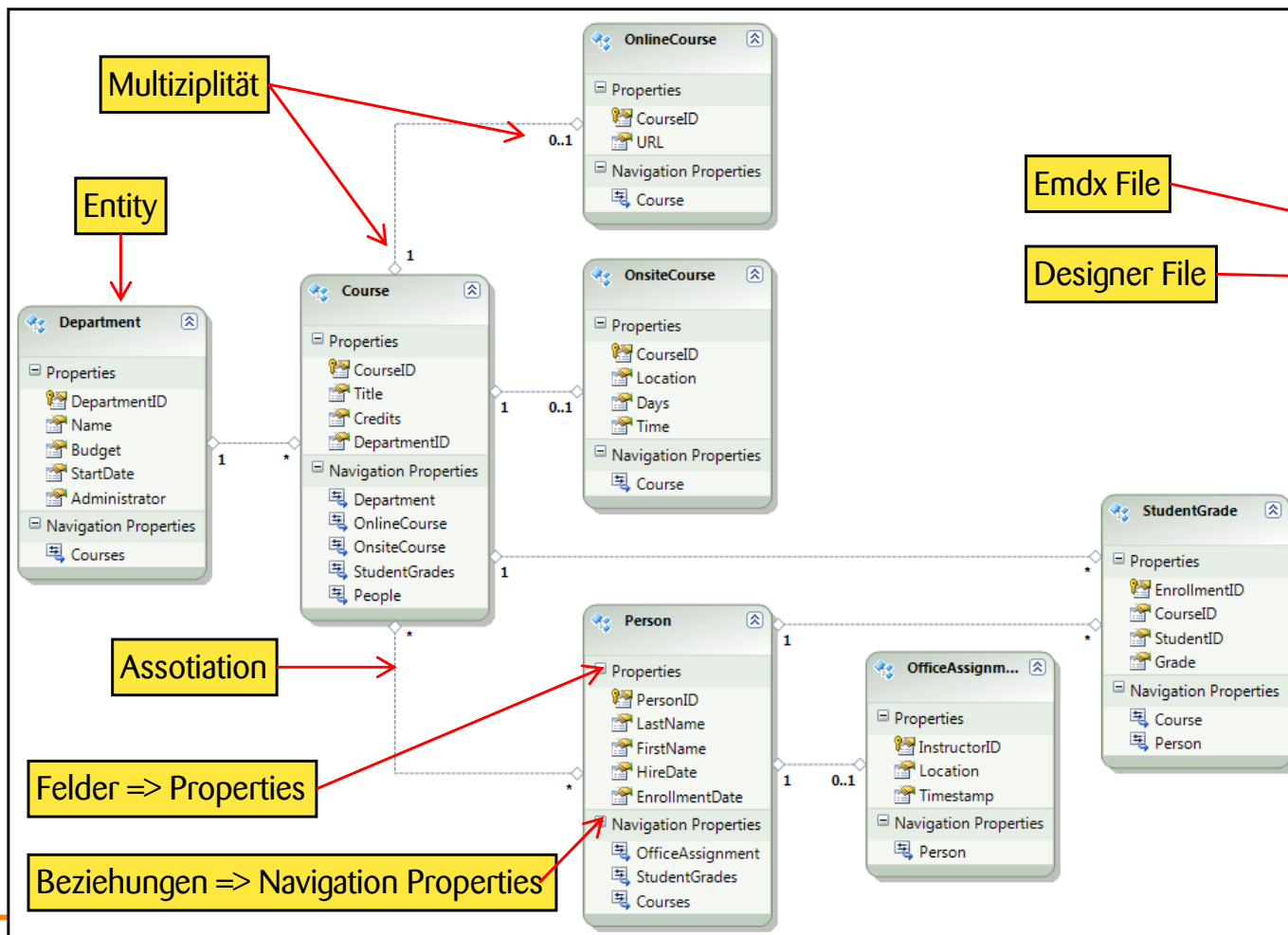
SchoolModel

< Previous Next > Finish Cancel

Entity Framework (Tutorial)

Generiertes Entity Data Model

- Ansicht im Entity Framework Designer

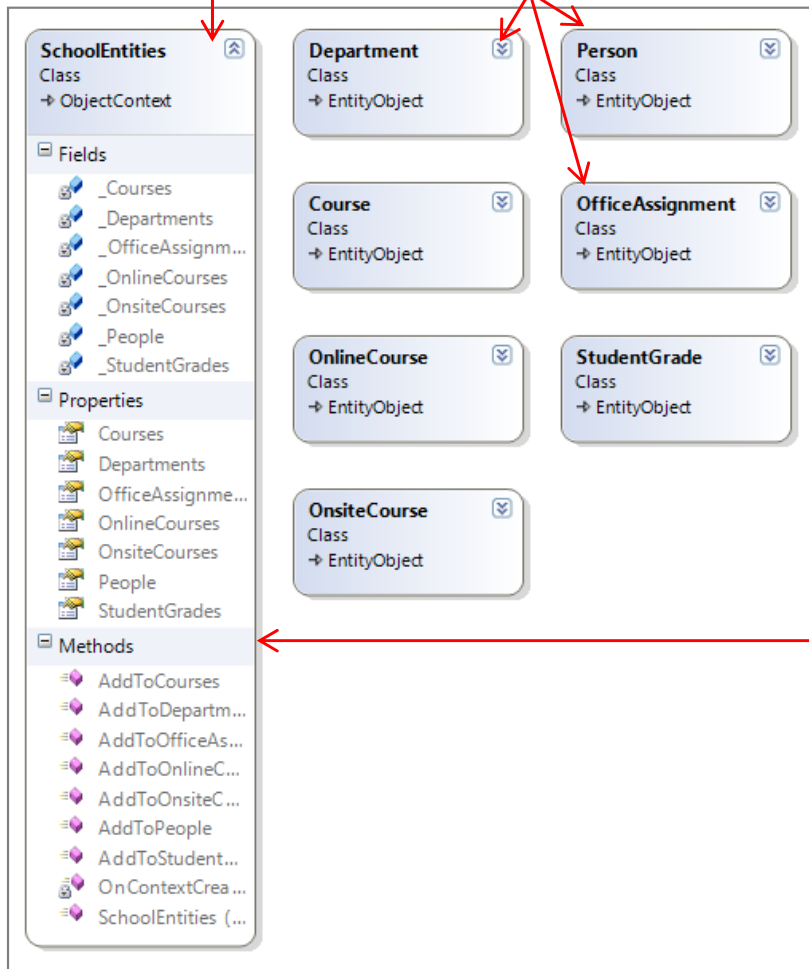


Entity Framework (Tutorial)

Generierte Klassen

Data Kontext Klasse

Entities



Data Kontext Klasse Code Ausschnitt

```
namespace WinFormEntityFramework
{
    #region Contexts

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    public partial class SchoolEntities :ObjectContext
    {
        #region Constructors

        /// <summary>
        /// Initializes a new SchoolEntities object using the connection string
        /// </summary>
        public SchoolEntities() : base("name=SchoolEntities", "SchoolEntities")
        {
            this.ContextOptions.LazyLoadingEnabled = true;
            OnContextCreated();
        }
    }
}
```

Entity Framework (Tutorial)

Entity Data Model Abfragen

■ Es bestehen drei Möglichkeiten

1. LINQ to Entities
2. Entity SQL
3. Native SQL

1. LINQ to Entities

```
using (SchoolEntities context = new SchoolEntities())
{
    var courses = from cs in context.Courses
                  where cs.CourseID == 1050
                  select cs;
    Course mathCourse = courses.FirstOrDefault<Course>();
    IList<Course> courseList = courses.ToList<Course>();

    string courseName = mathCourse.Title;
}
```

Entity Framework (Tutorial)

Entity Data Model Abfragen

2. Entity SQL

```
using (SchoolEntities context = new SchoolEntities())
{
    string sqlString = "SELECT VALUE cs FROM SchoolEntities.Courses" +
        " AS cs WHERE cs.CourseID = 1050";
    ObjectQuery<Course> course = context.CreateQuery<Course>(sqlString);
    Course firstCourse = course.FirstOrDefault<Course>();
}
```

3. Native SQL

```
using (SchoolEntities context = new SchoolEntities())
{
    //Insert Person mit ExecuteStoreCommand
    int InsertedRows = context.ExecuteStoreCommand(
        "INSERT INTO PERSON (LastName, FirstName) VALUES('Meier', 'Beno')");

    //Person abfragen mit ExecuteStoreQuery
    var pers = context.ExecuteStoreQuery<Person>(
        "SELECT * FROM PERSON WHERE LastName = 'Meier'", null).ToList();
    Person firstPerson = pers.FirstOrDefault();
}
```

Entity Framework (Tutorial)

Entity Data Model Abfragen

Achtung Lazy Loading!

- Aus Performanzgründen werden nicht automatisch alle Daten der Fremdtabellen mit geladen => Lazy Loading
- Um das Laden zu erzwingen, muss die Methode `Include()` angewendet werden

```
using (SchoolEntities context = new SchoolEntities())
{
    Person person = (from p in context.People.Include("Courses") // Load includes Courses
                     where p.PersonID == 31
                     select p).FirstOrDefault();

    List<Course> courses = (from c in context.Courses
                           select c).ToList<Course>();

    person.Courses.Remove(courses.ElementAt(0));
    person.Courses.Add(courses.ElementAt(4));
    context.SaveChanges();
}
```

Entity Framework (Tutorial)

Änderungen speichern - Methode SaveChanges

- Änderungen werden mit der Methode SaveChanges() in der Datenbank gespeichert

```
using (SchoolEntities context = new SchoolEntities())
{
    var course = (from cs in context.Courses
                  where cs.CourseID == 1050
                  select cs).FirstOrDefault();
    course.Title = "Mathematics";
    int count = context.SaveChanges();
}
```

- Die Status werden verändert:
 - Geänderte Entitäten auf Status = Unchanged
 - Gelöschte Entitäten werden entfernt

Entity Framework (Tutorial)

INSERT

■ Objekt Person hinzufügen

- Neues Objekt instanziiieren
- Zum Data Context hinzufügen
- Änderungen im Data Context in Database speichern

```
Person pers = new Person()
{
    LastName = "Huber", FirstName = "Alex"
};

using (SchoolEntities context = new SchoolEntities())
{
    context.People.AddObject(pers);
    context.SaveChanges();
}
```

Entity Framework (Tutorial)

UPDATE

■ Objekt Person aktualisieren

```
using (SchoolEntities context = new SchoolEntities())
{
    var pers = (from p in context.People
                where p.PersonID == 37
                select p).FirstOrDefault();
    pers.FirstName = "Hugo";
    int count = context.SaveChanges();
}
```

Locals	
Name	Value
pers	{WinFormEntityFramework.Person}
base	{WinFormEntityFramework.Person}
base	{WinFormEntityFramework.Person}
EntityKey	"EntitySet= People;PersonID=37"
EntityState	Modified
Static members	
Non-Public members	
_EnrollmentDate	null
_FirstName	"Hugo"
_HireDate	null
_LastName	"Huber"
_PersonID	37
Courses	{System.Data.Objects.DataClasses.Entity}
EnrollmentDate	null
FirstName	"Hugo"

- Objekt selektieren
- Objekt ändern
- Änderungen im Data Context in Database speichern

Entity Framework (Tutorial)

DELETE

■ Objekt Person löschen

```
using (SchoolEntities context = new SchoolEntities())
{
    var pers = (from p in context.People
                where p.LastName == "Huber"
                select p).FirstOrDefault();

    context.People.DeleteObject(pers);
    context.SaveChanges();
}
```

pers	{WinFormEntityFramework.Person}
base	{WinFormEntityFramework.Person}
base	{WinFormEntityFramework.Person}
EntityKey	"EntitySet=People;PersonID=38"
EntityState	Deleted
Static members	
Non-Public members	
EnrollmentDate	null

- Objekt selektieren
- Objekt als gelöscht markieren
- Änderungen im Data Context in Database speichern

Entity Lifecycle – Entity Status

- Jede Entität hat einen Status, abhängig von der ausgeführten Operation
- Enum `System.Data.EntityState`
 - Added
 - Deleted
 - Modified
 - Unchanged
 - Detached
- Der Context speichert die Referenzen zu den Entitäten und deren Zustände

Entity Framework

Object State Manager

- Verwaltet eine In-Memory-Collection aller von der Datenbank gelesenen Entitäten – Data Context
- Verwaltet die Status der Entitäten im Data Context
- Alle Operationen, wie z.B. Add, Attach usw. werden durch den Object State Manager ausgeführt

```
using (SchoolEntities context = new SchoolEntities())  
{  
    ObjectStateManager oManager = context.ObjectStateManager;  
}
```

Entity Framework (Tutorial)

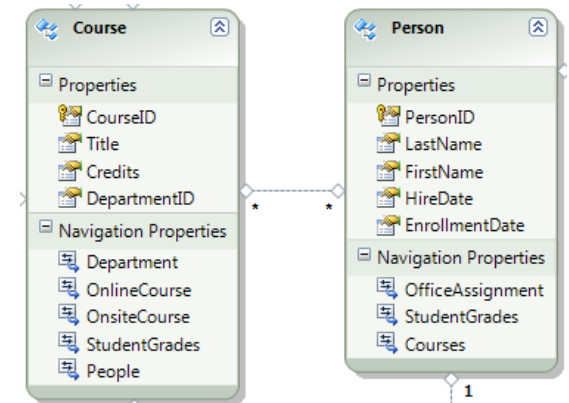
M:N-Relationen - Kurse zu einer Person hinzufügen

```
using (SchoolEntities context = new SchoolEntities())
{
    Person person = (from p in context.People.Include("Courses")
                     where p.PersonID == 31
                     select p).FirstOrDefault();

    List<Course> courses = (from c in context.Courses
                           select c).ToList<Course>();

    person.Courses.Clear();
    person.Courses.Add(courses.ElementAt(0));
    person.Courses.Add(courses.ElementAt(1));
    person.Courses.Add(courses.ElementAt(2));

    context.SaveChanges();
}
```



Course: Query(zrhn...\sqlexpress.School) X

	CourseID	Title
	1045	Calculus
	1050	Chemistry
	1061	Physics
	2021	Composition

Resultat in Datenbank

CourseInstructor: Q...\sqlexpress.School) X

	CourseID	PersonID
▶	1045	5
	1045	31
	1050	1
	1050	31
	1061	31

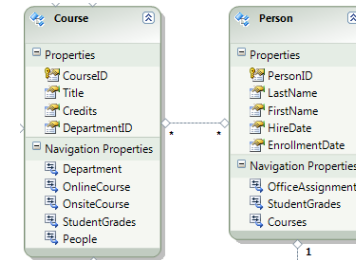
Entity Framework (Tutorial)

M:N-Relationen – Kurs einer Person ersetzen

```
using (SchoolEntities context = new SchoolEntities())
{
    Person person = (from p in context.People.Include("Courses")
                     where p.PersonID == 31
                     select p).FirstOrDefault();

    List<Course> courses = (from c in context.Courses
                           select c).ToList<Course>();

    person.Courses.Remove(courses.ElementAt(0));
    person.Courses.Add(courses.ElementAt(4));
    context.SaveChanges();
}
```



	CourseID	Title
	1045	Calculus
	1050	Chemistry
	1061	Physics
	2021	Composition
	2030	Poetry
	2042	Literature

Resultat in Datenbank

	CourseID	PersonID
	1045	5
	1050	1
	1050	31
	1061	31
	2021	27
	2030	4
	2030	31

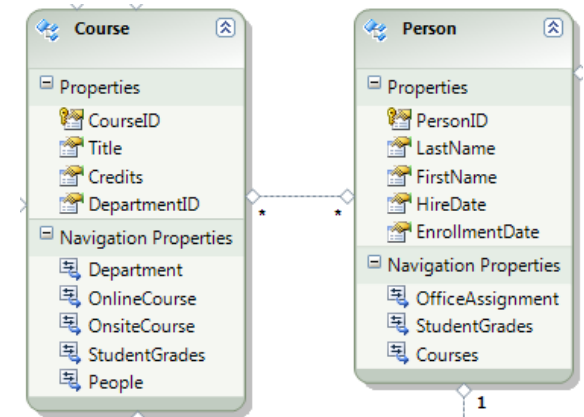
Entity Framework (Tutorial)

M:N-Relationen – Kurse einer Person löschen

```
using (SchoolEntities context = new SchoolEntities())
{
    Person person = (from p in context.People.Include("Courses")
                     where p.PersonID == 31
                     select p).FirstOrDefault();

    List<Course> courses = (from c in context.Courses
                           select c).ToList<Course>();

    person.Courses.Clear();
    context.SaveChanges();
}
```

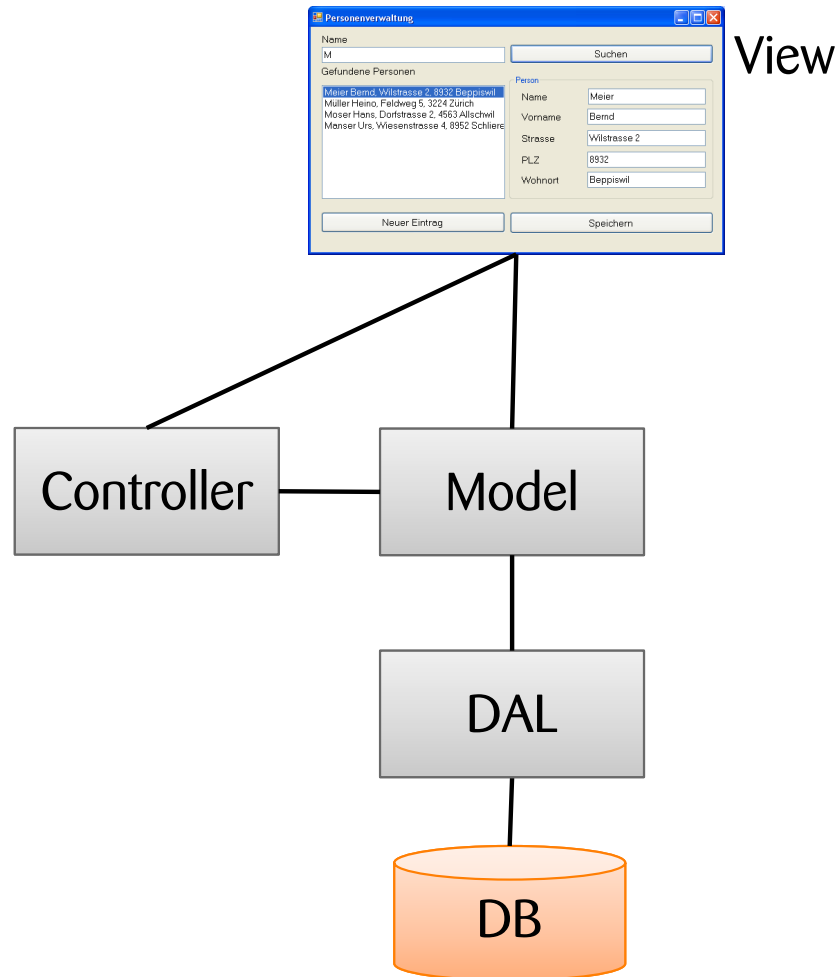


Resultat in Datenbank

	CourseID	PersonID
	1045	5
	1050	1
	2021	27
	2030	4
	2042	25
	4022	18
	4041	32
	4061	34
▶*	NULL	NULL

MVC - Pattern

High-level Architecture



MVC - Pattern

Architecture

Anzeige / Eingabe

- UpdateView
- EventHandler (Add, Search,...)

PersonManagementView

Gezeichnete Person:

- Meier Bernd, Wilstrasse 2, 8932 Beppiswil
- Müller Heinz, Feldweg 5, 3224 Zürich
- Moser Hans, Dorfstrasse 2, 4563 Allschwil
- Moser Urs, Wiesenstrasse 4, 8952 Schlieren

Person

Name: Meier

Vorname: Bernd

Strasse: Wilstrasse 2

PLZ: 8932

Wohnort: Beppiswil

Neuer Eintrag Speichern

View

Commands

Read

PersonManager

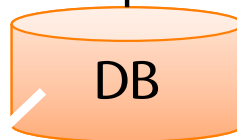
Controller

CRUD

PersonModel

Model

Data Access



PersonManagementDB

Daten

List<Person>
CRUD Operations

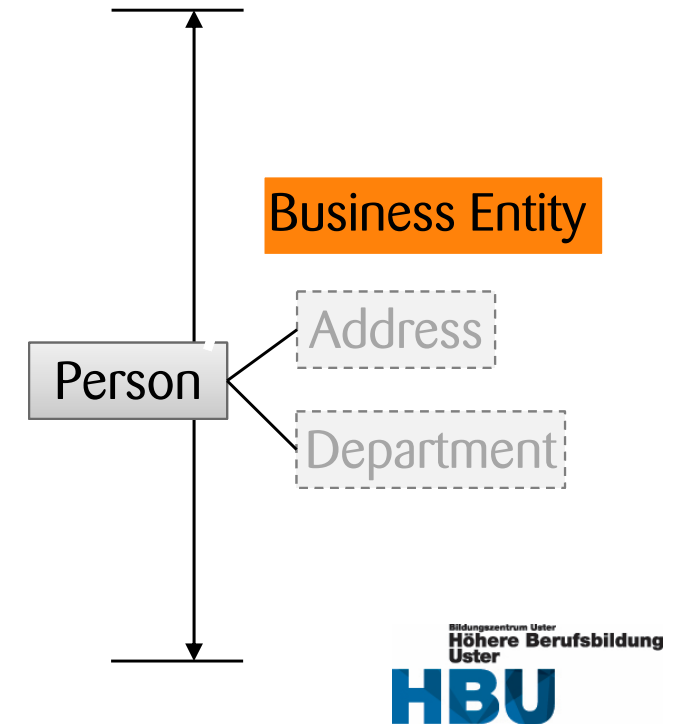
- Create
- Read
- Update
- Delete

Business Logik

- AddPerson
- SearchPerson
- UpdatePerson
- DeletePerson

Datenhaltung

Business Entity



Übung 2.3



Kursverwaltung - EF

- Schreibe eine «Windows Forms»-Applikation, um Kursdaten zu verwalten. Verwende die SchoolDB vom Entity Framework Tutorial.

<http://www.entityframeworktutorial.net/EntityFramework6/introduction.aspx>

- Implementiere:
 - a. Departments anzeigen
 - b. Kurse des selektierten Departments anzeigen
 - c. Alle Kurse anzeigen
 - d. Kurse zu einem Department zuordnen
 - e. Departments erfassen
 - f. Kurse erfassen