DB-Zugriff mit C# und .Net

Allgemeine Betrachtung

Die Programmierung in C# setzt auf das Microsoft DotNet-Framework auf. Datenbankzugriffe finden im .Net Framework mit Hilfe von Ado.net (**A**ctive **D**ata **O**bjects) statt. Dabei handelt es sich um eine Sammlung von Klassen die alle Bereiche vom einfachen Datenzugriff über das automatische Erstellen von SQL-Statements bis zur grafischen Darstellung der Ergebnisse abdecken.

Die grundlegenden Klassen für den Datenbankzugriff, die sogenannten Data Provider, befinden sich im Namespace System.Data.* und erlauben den Zugriff über ODBC, OleDB und eine native MSSQL-Server Schnittstelle. Darüber hinaus werden Basisklassen bereitgestellt von denen man eigene Klassen ableiten kann um spezielle Data Provider zu entwickeln.

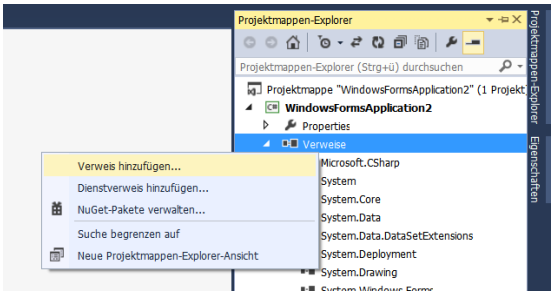
Mit dem Zugriff auf ein DBMS (Datenbank-Management-System) wird die erstellte Anwendung zum Teil abhängig vom angebundenen DBMS. Um Anwendungen mit DB-Zugriff trotzdem weitgehend unabhängig vom verwendeten DBMS zu machen, werden alle Vorgänge, die sich auf ein bestimmtes DBMS beziehen in Klassen ausgelagert, die dann bei Migration auf ein anderes DBMS leicht angepasst oder ausgetauscht werden können.

Grundsätzlich gilt folgender Ablauf für einen Datenbankzugriff

1. Herstellen der DB-Verbindung (Erstellen eines MySqlConnection Objekts)
2. Erstellen eines MySql-Command-Objekts auf der DB-Verbindung mit dem gewünschten SQL-Statement.
3. Senden des Statement-Objekts und Entgegennehmen der Ergebnistabelle (MySqlDataReader-Objekt)
4. Auswerten des Ergebnisses (Auslesen der Ergebnisdatensätze aus dem MySqlDataReader-Objekt.)
5. Schließen der DB-Verbindung

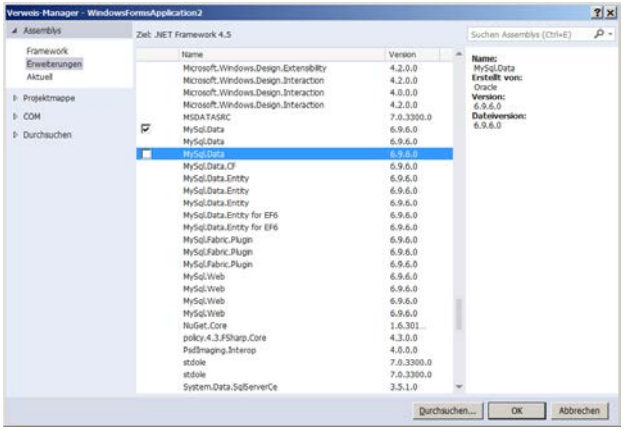
DB-Zugriff auf eine MySql-Datenbank über C# und Dot-Net

1. MySQL-DotNet-Treiber besorgen
   Laden Sie den neusten **stabilen** .Net-Treiber für Ihren DB-Server herunter (z.B. dev.mysql.com/downloads/connector/net/) und **installieren** Sie diesen.



2. MySQL-DotNet-Treiber einbinden
   Klicken Sie mit der rechten Maustaste im Projektmappen-explorer von Visual Studio auf ‚Verweise/Verweis hinzufügen'

   Wählen Sie MySql.Data mit der höchsten Versionsnummer.



3. Neue Klasse ‚MySQLConnector' anlegen
   Legen Sie die Klasse , MySQLConnector' an, die alle Informationen verwalten soll, die ganz speziell auf MySQL zugeschnitten sind.

4. DB-Zugriffsparameter als Attribute der Klasse bereitstellen und kapseln

```
private String dBHost="localhost"; // kann auch IP-Adresse sein
private String dBUser="root";
private String dBPasswort="";
private String dBName="pcverwaltung";
```

   Durch Visual Studio kapseln lassen

5. Einen ConnectionStringBuilder erzeugen
   Die Verbindung zum MySQL-Server erfordert eine bestimmte Syntax. Um hier die Fehleranfälligkeit zu reduzieren, bietet der .Net-MySQL-Connector die Klasse MySqlConnectionStringBuilder an. Davon muss ein Objekt als Attribut der Klasse MySQLConnector erstellt werden.

```
private MySqlConnectionStringBuilder mysqlConnectionStringBuilder
                                    = new MySqlConnectionStringBuilder();
```

   Die Verwendung dieser Klasse erfordert die Einbindung der using-Direktive

```
using MySql.Data.MySqlClient;
```

6. Connection-String erzeugen in einer eigenen Read-Only-Property

```
public String ConnectionString
   {
       get
       {
           mysqlConnectionStringBuilder.Server = DBHost;
           mysqlConnectionStringBuilder.UserID = DBUser;
           mysqlConnectionStringBuilder.Password = DBPasswort;
           mysqlConnectionStringBuilder.Database = DBName;

           return mysqlConnectionStringBuilder.ConnectionString;
       }
   }
```

7. testConnection-Methode (Methode zum Testen der DB-Verbindung)

```
public String testConnection()
    {
        String msg = "DB-Verbindung erfolgreich";
        MySqlConnection connection = new MySqlConnection(ConnectionString);
        try
        {
            connection.Open();
            connection.Close();

        }
        catch (Exception ex)
        {
            msg = ex.GetBaseException().Message;
        }
        return msg;
    }
```

8. DB-Verbindung testen
   Im Menü von Form1 gibt es den Menüpunkt '? / DB-Verbindung testen'. Erzeugen Sie dort ein MySqlConnector-Objekt, rufen Sie die testConnection-Methode auf und zeigen Sie das Ergebnis als Pop-Up an.

```
MySQLConnector connector = new MySQLConnector();
MessageBox.Show(connector.testConnection());
```

DB-Zugriffsklasse einer Fachklasse

In einer DB-Zugriffsklasse, hier ‚DBAdapterPCVerwaltung', die auf die Datenbank **und** die Anwendung zugeschnitten ist, werden die SQL-Statements erzeugt, abgesetzt und die Ergebnisse (Tabelle in einem MySqlDataReader-Objekt) ausgewertet. Das bedeutet, dass für jede Abfrage eine eigene Methode hergestellt wird, die die Funktionalität im Namen trägt, falls nötig Parameter entgegennimmt und das Ergebnis als Objekt einer Klasse zurückgibt, die in der Anwendung weiterverwendet werden kann. In der Klasse muss ein Objekt der MySQLConnector-Klasse als Attribut erzeugt werden.

```
private MySQLConnector mysqlConnector = new MySQLConnector();
```

9. Zugriffsmethoden – hier eine Methode, die alle PCs aus der Datenbank ausliest

```
public List<PC> getAllePCs()
```

- Eine neue Liste anlegen

```
List<PC> allePCs = new List<PC>();
```

- SQL-Statement formulieren und SQL-Kommando Objekt erzeugen

```
String sql = "SELECT id, hdd, takt, ram " +
             "FROM tblPCs; ";
MySqlCommand sqlcommand=new MySqlCommand(sql);
```

- MySqlConnection-Objekt erzeugen, an das SQL-Kommando übergeben und die Datenbankverbindung öffnen

```
MySqlConnection connection = new MySqlConnection(mysqlConnector.ConnectionString);
sqlcommand.Connection = connection;
connection.Open();
```

- Ergebnistabelle vom Server abholen

```
MySqlDataReader dataReader = sqlcommand.ExecuteReader();
```

- Ergebnistabelle (MySqlDataReader) verarbeiten ähnlich wie beim Auslesen aus Dateien.

```
while (dataReader.Read())
{
    PC neuerPC = new PC();
    neuerPC.ID=dataReader.GetInt16("id");
    neuerPC.HDD = dataReader.GetDouble("hdd");
    neuerPC.RAM = dataReader.GetDouble("ram");
    neuerPC.Prozessorgeschwindigkeit = dataReader.GetDouble("takt");
    allePCs.Add(neuerPC);
}
```

- Offene Verbindung wieder schließen und allePCs in der Liste zurückgeben

```
connection.Close();
return allePCs;
```

10. Dort wo das Auslesen aus der Datenbank erfolgen soll, muss ein Objekt der Klasse DBAdapterPCVerwaltung erstellt werden und die Methode von der Oberfläche aufgerufen werden.

```
allePC=dbAdapterPCVerwaltung.getAllePCs();
```

11. Weitere Zugriffsarten sind Einfügen (Insert), Ändern (Update) und Löschen (Delete) hier soll als Beispiel die Einfügen Methode gezeigt werden.

```
public void insert(PC einPC)
{
   String dml="INSERT INTO tblpcs(hdd, takt, ram)"+
              "VALUES("+einPC.HDD+","
                       +einPC.Prozessorgeschwindigkeit+","
                       +einPC.RAM+");";
   MySqlConnection connection = new MySqlConnection(mysqlConnector.ConnectionString);
   connection.Open();
   MySqlCommand sqlcommand = new MySqlCommand(dml, connection);
   sqlcommand.ExecuteNonQuery();
   connection.Close();
}
```

Diese Methode kann nun in der Oberfläche verwendet werden. Wenn ein neuer PC eingefügt wurde, darf man nicht vergessen alle PCs neu aus der Datenbank auszulesen.