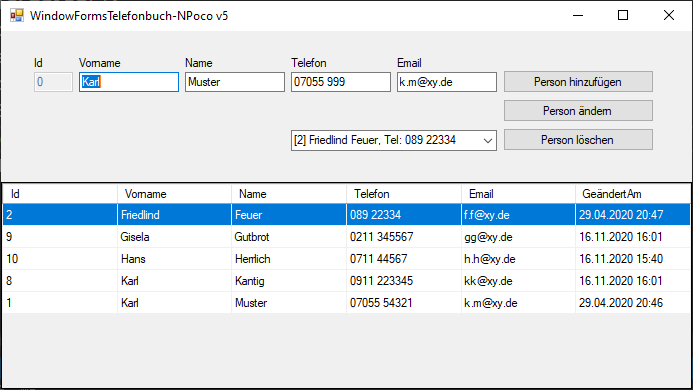
Aufgabe WindowsFormsTelefonbuchNPoco5

In diesem Projekt soll ein einfaches Telefonbuch mit dem Objekt-Relationalen Mapper NPoco realisiert werden. Die Personen werden nicht mit ADO.NET, sondern mit NPoco in eine MariaDb gespeichert.



* Beschreiben Sie die funktionalen **Anforderungen** als Use-Cases in einem UML- Anforderungsdiagramm.
* Zum **Design** der Anwendung stellen Sie die zu verwendende Datenklasse in einem UML-**Klassendiagramm** dar. Die Datenklasse enthalte Attribute für Id, Vorname, Name, Telefon, Email und ein GeändertAm Feld.
* Legen Sie eine **Datenbank** telefonbuch auf MariaDB an.   
  Erzeugen Sie eine **Tabelle** person mit den **Spalten**   
  id (INT AUTOINCREMENT), vorname (VARCHAR), name (VARCHAR), telefon (VARCHAR), email (VARCHAR) und geändertam (DATETIME) an.

Schreiben Sie dazu ein **SQL-Script** oder verwenden den MariaDB-Client HeidiSql.

* Starten Sie ein neues Windows-Forms-Projekt namens **WindowsFormsTelefonbuchNPoco**.
* Zwischen der zu verwendenen NPoco Bibliothek gibt es Unterschiede zwischen Version 4.0.2 und 5.0.0 z.B. bezüglich der Initialisierung der Klasse Database. Die weiteren Hinweise beziehen sich auf Version 5.0.0.
* Installieren Sie **NPoco** über die Paketmanager-Konsole oder die Paketmanager-Verwaltung.

Extras -> Nuget-Paket-Manager -> Paketmanager-Konsole PM> install-package npoco  
oder:

Extras -> NuGet-Paket-Manager -> NuGet-Pakete verwalten

* Stellen Sie sicher, dass der MySql-Connector oder der komplette MySql-server installiert sind.
* Fügen Sie den **Verweis** auf MySql.Data hinzu.
* Fügen Sie in Form1 den **namespace** NPoco, System.Configuration und MySql.Data.MySqlClient hinzu.
* Gestalten Sie die **Oberfläche**. Orientieren Sie sich dabei an der obigen Darstellung.
* Fügen Sie ein **DataGridView** hinzu, um die Liste anzeigen zu können.
* Für Debug-Ausgaben können Sie eine **Listbox** vorsehen.
* Implementieren Sie die Person **Klasse** mit den Microsoft Properties (get,set) Id, Vorname, Name, Telefon, Email und GeändertAm und den passenden Typen.
* Fügen Sie zusätzlich ein Feld namens Instance vom Typ Person und der Annotation [NPoco.Ignore] hinzu. Dies kann später zur Laufzeit eine Referenz auf die Person enthalten.

[NPoco.Ignore]

public Person Instance { get; set; }

* Legen Sie eine **generische Liste** **lstPerson** für die Personen an.
* Fügen Sie in der Datei **App.config** den **Connectionstring** ein.  
  Passen Sie den Parameter Pwd und Database an Ihre Umgebung an.

<connectionStrings>

<add name="mariadb" connectionString="Server=localhost;Uid=root;Pwd=pa$$w0rd;Database=telefonbuch;Port=3306"

providerName="MySql.Data.MySqlClient" />

</connectionStrings>

* Fügen Sie zu Anfang der Klasse Form1 ein Property für den ConnectionString ein.  
  string ConnectionString { get; } = ConfigurationManager.ConnectionStrings["mariadb"].ConnectionString;
* Implementieren Sie die private Methode **DatenLaden**() mit NPoco, z.B.:

private void DatenLaden()

{

try

{

using (MySqlConnection connection = new MySqlConnection(ConnectionString))

using (IDatabase db = new Database(connection))

{

db.Connection.Open();

lstPerson.Clear();

lstPerson = db.Fetch<Person>("order by name,vorname");

}

}

catch (Exception ex)

{

System.Diagnostics.Debug.WriteLine(ex);

}

}

* Implementieren Sie die private Methode AnzeigeAktualisieren(), z.B:

private void AnzeigeAktualisieren()

{

// dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

// dataGridView1.MultiSelect = false;

// dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

// Instance setzen

foreach (Person p in lstPerson)

{

p.Instance = p;

}

dataGridView1.DataSource = null;

cboPerson.DataSource = null;

dataGridView1.DataSource = lstPerson;

cboPerson.DataSource = lstPerson;

dataGridView1.Columns["Instance"].Visible = false;

dataGridView1.RowHeadersVisible = false;

}

* Fügen Sie eine Ereignisbehandlungs-Methode für Load hinzu (Doppelklick auf Form)
* Implementieren Sie Form1\_Load, z.B.:   
    
   private void Form1\_Load(object sender, EventArgs e)

{

DatenLaden();

AnzeigeAktualisieren();

}

* Fügen Sie eine Ereignisbehandlungs-Methode **CellClick** für das dataGridView1 hinzu und implementieren diese, d.h. aktualisieren Sie die Textboxen.

private void dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)

{

Person p = dataGridView1.CurrentRow.Cells["Instance"].Value as Person;

if (p != null)

{

txtId.Text = p.Id.ToString();

txtVorname.Text = p.Vorname;

txtName.Text = p.Name;

txtTelefon.Text = p.Telefon;

txtEmail.Text = p.Email;

//txtGeändertAm.Text = p.GeändertAm.ToString("yyyy-MM-dd HH:mm");

cboPerson.SelectedItem = p;

}

}

* Implementieren Sie PersonHinzufügen, PersonÄndern und PersonLöschen und rufen diese in den entsprechenden Ereignisbehandlungsmethoden der Buttons auf.

private void PersonHinzufügen()

{

try

{

Person p = new Person();

p.Id = 0; // wird von Datenbank vergeben

p.Vorname = txtVorname.Text;

p.Name = txtName.Text;

p.Telefon = txtTelefon.Text;

p.Email = txtEmail.Text;

p.GeändertAm = DateTime.Now;

using (MySqlConnection connection = new MySqlConnection(ConnectionString))

using (IDatabase db = new Database(connection))

{

db.Connection.Open();

db.Insert(p);

}

}

catch (Exception ex)

{

System.Diagnostics.Debug.WriteLine(ex);

}

}

private void PersonÄndern()

{

try

{

Person p = new Person();

p.Id = Convert.ToInt32(txtId.Text); // id übernehmen

p.Vorname = txtVorname.Text;

p.Name = txtName.Text;

p.Telefon = txtTelefon.Text;

p.Email = txtEmail.Text;

p.GeändertAm = DateTime.Now;

using (MySqlConnection connection = new MySqlConnection(ConnectionString))

using (IDatabase db = new Database(connection))

{

db.Connection.Open();

db.Update(p);

}

}

catch (Exception ex)

{

System.Diagnostics.Debug.WriteLine(ex);

}

}

private void PersonLöschen()  
 {

try

{

Person p = cboPerson.SelectedItem as Person;

if (p != null)

{

using (MySqlConnection connection = new MySqlConnection (ConnectionString))

using (IDatabase db = new Database(connection))

{

db.Connection.Open();

db.Delete(p);

}

}

}

catch (Exception ex)

{

System.Diagnostics.Debug.WriteLine(ex);

}

}