

Aufgabe WindowsFormsTelefonbuchXml

In dieser Aufgabe soll ein einfaches Telefonbuch objektorientiert realisiert werden. Neue Personen sollen hinzugefügt und Personen gelöscht werden können. Beim Schließen sollen die Daten in eine XML-Datei serialisiert werden. Beim Start soll daraus der letzte Zustand geladen werden.

Orientieren Sie sich an der folgenden Darstellung.

Vorname	Name	Telefon	Email	AngelegtAm
Karl	Muster1	07055 999	k.m@xyz.de	12.11.2021 09:13
Karl	Muster2	07055 999	k.m@xyz.de	12.11.2021 09:13
Karl	Muster3	07055 999	k.m@xyz.de	12.11.2021 10:13

Hinweise:

- Entwerfen Sie die **Oberfläche**:
 - Fügen Sie ein Panel ein mit Dock=Top
 - Fügen Sie ein DataGridView ein mit Dock=Fill
 - Fügen Sie vier TextBoxen ein: txtVorname, txtName, txtTelefon, txtEmail
 - Platzieren Sie vier Labels: label1,... Text: Vorname,...
 - Fügen Sie Buttons ein: btnPersonHinzufügen, btnPersonLöschen, btnBeenden
 - Text: Person hinzufügen, Person löschen, Beenden

Setzen Sie die Eigenschaft StartPosition von Form1 ggf. auf CenterScreen

Setzen Sie die Eigenschaft Text von Form1.

- Deklarieren Sie eine **Klasse** Person entsprechend der folgenden Darstellung (UML Klassendiagramm).
Verwenden Sie für die Attribute die Microsoft Automatic Properties mit {get; set;}
- Implementieren Sie einen **Konstruktor** für die Klasse **Person** mit 4 Parametern.



- Sehen Sie am Anfang der Klasse **Form1** eine generische **Liste** `List<Person>` zur Speicherung der Personen während der Laufzeit vor.
(Array: `Person[]` arPerson, generische Liste: `List<Person>` lstPerson)
- Implementieren Sie die **Ereignisbehandlungs-Methode** für den Button "Person hinzufügen":
Generieren Sie eine neue Person (`Person person = new Person(...)`). Fügen Sie die Person zur Liste mit `lstPerson.Add(person)` hinzu. Rufen Sie danach die Methode `PersonenAnzeigen()` auf.

```

private void btnPersonHinzufügen_Click(object sender, EventArgs e)
{
    // neue Person anlegen mit Werten aus den Textboxen
    Person person = new Person(txtVorname.Text,
                               txtName.Text,
                               txtTelefon.Text,
                               txtEmail.Text);

    // Person zur Liste hinzufügen
    lstPerson.Add(person);

    PersonenAnzeigen();
}
  
```

- Implementieren Sie die private Methode **PersonenAnzeigen()**. Hierbei soll die aktuelle Liste der Personen im DataGridView angezeigt werden. Dazu wird die einfachste Art einer Bindung der Daten an das DataGridView-Steuerelement über DataSource verwendet.

```

dataGridView1.DataSource = null;
dataGridView1.DataSource = lstPerson;
  
```

- Testen Sie das **Anzeigen** und **Hinzufügen** von Personen.
- Setzen Sie die folgenden **Eigenschaften** für das **DataGridView** entweder dynamisch im **Quellcode** oder als statische Anfangswerte im **Designer**.

```
public Form1()
{
    InitializeComponent();
    GridInitialisieren();
}

private void GridInitialisieren()
{
    // Einstellungen für DataGridView
    dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;
    dataGridView1.RowHeadersVisible = false;
    dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
    dataGridView1.MultiSelect = false;
}
```

Im Folgenden soll die **Liste** der Personen **serialisiert** und in eine **XML**-Datei geschrieben werden.

- Fügen Sie zu Anfang der Datei **Form1.cs** die Namespaces **using System.XML** und **using System.Xml.Serialization** hinzu. Die Namespaces werden damit bekannt gemacht.
- Machen Sie die Klasse **Person** **public** und fügen oberhalb der Klasse die Annotation **[Serializable]** ein.
- Stellen Sie sicher, dass die Klasse auch einen öffentlichen **Standard-Konstruktor** mit 0 Parametern enthält.

```
public Person() { }
```

- Deklarieren Sie in **Form1** eine Variable **xmlPath** und initialisieren diese z.B. mit dem Dateinamen "person.xml".
- Übernehmen Sie die folgenden zwei **Methoden** in die Klasse **Form1**.

```

private void PersonenSpeichernXml()
{
    try
    {
        // beim Schließen der Anwendung schreibe aktuelle Liste in die XML-Datei
        using (XmlWriter xmlWriter = XmlWriter.Create(xmlPath))
        {
            XmlSerializer xmlSerializer = new XmlSerializer(typeof(List<Person>));
            xmlSerializer.Serialize(xmlWriter, lstPerson);
        }
    }
    catch (Exception ex)
    {
        System.Diagnostics.Debug.WriteLine(ex);
    }
}

```

```

private void PersonenLadenXml()
{
    if (!File.Exists(xmlPath))
        return;

    try
    {
        // beim Laden der Form hole Daten aus der XML-Datei
        using (XmlReader xmlReader = XmlReader.Create(xmlPath))
        {
            XmlSerializer xmlSerializer = new XmlSerializer(typeof(List<Person>));
            lstPerson.Clear();
            lstPerson = xmlSerializer.Deserialize(xmlReader) as List<Person>;
        }
    }
    catch (Exception ex)
    {
        System.Diagnostics.Debug.WriteLine(ex);
    }
}

```

- Generieren Sie die **Ereignisbehandlungs-Methoden** für das Ereignis **Load** und das Ereignis **FormClosing**. Markieren Sie dazu die **Form1** im Entwurfsfenster. Wählen Sie die **Ereignisse** im Eigenschaften-Fenster und klicken doppelt rechts vom Ereignis **Load** und danach rechts vom Ereignis **FormClosing**
- Implementieren Sie die Methoden, indem Sie bei **FormClosing**, die Methode **PersonenSpeichernXml()** aufrufen. Bei **Load** rufen Sie **PersonenLadenXml()** und **PersonenAnzeigen()** auf.

```

// Daten automatisch am Ende speichern
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    PersonenSpeichernXml();
    PersonenSpeichernJson();
}

```

- Fügen sie einen **Beenden**-Button hinzu (btnBeenden).
In der Ereignisbehandlungs-Methode rufen Sie die Close()-Methode der Form auf.
- **Testen** Sie das **Laden** und **Speichern** der Personen.
- Zeigen Sie in den TextBoxen die Werte der gerade im **DataGridView** ausgewählten **Person** an.
Erstellen Sie dazu die Ereignisbehandlungsmethode für das Ereignis CellClick (nicht CellContentClick) des DataGridView Steuerelements.

Markieren Sie dazu das DataGridView Steuerelement dataGridView1.

Öffnen Sie im Eigenschaftenfenster die **Ereignisse** (Blitz-Symbol) und klicken Sie doppelt auf das leere Feld rechts neben dem Ereignis CellClick.

Der Designer erstellt dabei die **Ereignisbehandlungsmethode** und springt zur Code-Ansicht.
Füllen Sie die Methode mit den erforderlichen Anweisungen.

```
// Im Grid selektierte Person anzeigen
private void dataGridView1_CellClick(object sender,
                                     DataGridViewCellEventArgs e)
{
    Person person = dataGridView1.CurrentRow.DataBoundItem as Person;
    if (person == null)
        return;
    PersonAnzeigen(person);
}

private void PersonAnzeigen(Person person)
{
    txtVorname.Text = person.Vorname;
    txtName.Text = person.Name;
    txtEmail.Text = person.Email;
    txtTelefon.Text = person.Telefon;
}
```

- Markieren Sie verschiedene Zeilen im **DataGridView** und prüfen, ob die entsprechenden Werte in den Textboxen richtig angezeigt werden.
- Implementieren Sie die Methode für den Button "**Person löschen**", z.B.

```
// Referenz auf die im GridView selektierte Person
Person person = dataGridView1.CurrentRow.DataBoundItem as Person;
if (person == null)
    return;

// Person aus der Liste löschen
lstPerson.Remove(person);
PersonenAnzeigen();
```

Erweiterung:

- Alternativ lassen sich die Daten auch in eine **Json**-Datei speichern.
- Fügen Sie die folgenden zwei Methoden zur Klasse **Form1** hinzu.
Deklarieren Sie in Form1 eine weitere Variable `jsonPath = "person.json"`;

```
// Daten mit dem JsonSerializer in eine json-Datei speichern
private void PersonenSpeichernJson()
{
    try
    {
        string json = JsonSerializer.Serialize<List<Person>> (lstPerson,
            new JsonSerializerOptions() { WriteIndented = true });
        File.WriteAllText(jsonPath, json);
    }
    catch (Exception ex)
    {
        System.Diagnostics.Debug.WriteLine(ex);
    }
}

// Daten mit dem DeSerializer aus einer Json Datei laden
private void PersonenLadenJson()
{
    if (!File.Exists(xmlPath))
        return;

    try
    {
        string json = File.ReadAllText(jsonPath);
        lstPerson = JsonSerializer.Deserialize<List<Person>>(json);
    }
    catch (Exception ex)
    {
        System.Diagnostics.Debug.WriteLine(ex);
    }
}
```