DevTrain

Autor: Bernhard Elbl

Klassen serializen in C#

Serialisieren - Häufig hört man jetzt dieses Wort. Aber was ist Serialisieren eigentlich? Wenn Sie eine Klasse serialisieren, wird die Klasse zu einem Data-Stream. Das heisst, dass Sie die Klasse z.B. als Datei abspeichern können. Diese Datei können Sie dann wieder deserialisieren. So wird die Datei wieder zur Klasse, die Sie in Ihrer Anwendung verwenden können. In dieser Datei können alle Daten gespeichert werden; Strings, Arrays...eigentlich alle Objekte. In wenigen Zeilen Code können Sie eine komplette Klasse mit allen aktuellen Eigenschaften abspeichern und wieder laden. Serialisieren war schon mit C++ möglich und ist also nichts neues, mit der .NET Technologie wirft das "Serializen" aber ganz neue Möglichkeiten auf, und zwar deswegen, weil es so einfach ist wie noch nie zuvor.

Dieser Artikel soll Ihnen verdeutlichen, was "Serializen" eigentlich ist und geht weniger in die tiefen Abgründe und Seitengassen dieser Technologie.

Hier ein simples Windows-Form Beispiel:

1. Wir binden 2 Namespaces ein.

```
// stellt den XmlSerializer zur Verfügung, dieser kann eine Klasse in Xml serializen.
using System.Xml.Serialization;
// stellt Möglichkeiten für Datei-Zugriff bereit
using System.IO;
```

2. Jetzt erstellen wir eine kleine Klasse mit den Attribut [Serializable()]. Das Attribut gibt uns die Möglichkeit zur Serialisierung der Klasse MyUserInfo bereit.

```
[Serializable()]
public class MyUserInfo
{
 public string FirstName = string.Empty;
 public string LastName = string.Empty;
 public int Age = 0;
}
```

3. Jetzt platzieren wir 3 Textboxen und 2 Button auf unsere Form.
Button1 serialisiert die Klasse UserInfo und speichert sie auf C: ab.

```
private void button1_Click(object sender, System.EventArgs e)
{
 // Klasse UserInfo mit Werten füllen
 MyUserInfo uinfo = new MyUserInfo();
 uinfo.FirstName = textBox1.Text;
 uinfo.LastName = textBox2.Text;
 uinfo.Age = System.Convert.ToInt32(textBox3.Text);
 // Serializer erstellen, der den Object-Type MyUserInfo aufnimmt
 XmlSerializer serializer = new XmlSerializer(typeof(MyUserInfo));
```

```
 // Stream-Object und den Object-Stream als Datei abzuspeichern.
 StreamWriter writer = File.CreateText("C:\myMyUserInfo.xml");
 // Serialisieren
 serializer.Serialize(writer, uinfo);
 // Datei schliessen
 writer.Close();
}
```

Die Datei myMyUserInfo.xml hat folgenden Inhalt.

```
<?xml version="1.0" encoding="utf-8"?>
<MyUserInfo xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <FirstName>Bernhard</FirstName>
  <LastName>Elbl</LastName>
  <Age>22</Age>
</MyUserInfo>
```

Button2 deserialisiert die eben erstellte Datei und erzeugt wieder eine Klasse von Typ UserInfo und zeigt die Eigenschaften der Klasse UserInfo in den TextBoxen dar.

```
private void button2_Click(object sender, System.EventArgs e)
{
 // Serializer-Object erstellen, der Objecte von Type MyUserInfo behandeln hat.
 XmlSerializer serializer = new XmlSerializer(typeof(MyUserInfo));
 // StreamReader kann Dateien auslesen.
 StreamReader reader = File.OpenText("C:\myMyUserInfo.xml");

 // Klasse wird deserialisiert
 MyUserInfo uinfo = (MyUserInfo) serializer.Deserialize(reader);

 // Eigenschaften anzeigen
 textBox1.Text = uinfo.FirstName;
 textBox2.Text = uinfo.LastName;
 textBox3.Text = uinfo.Age.ToString();
}
```

Sie können auch wesentlich komplexere Klassen mit eigenen Unterklassen abspeichern.

Schlusswort:
Sie brauchen sich nicht mehr drum zu kümmern, **WIE** Sie Informationen abspeichern, sondern nur noch **WAS** sie abspeichern wollen. Serialisierung macht es sehr einfach Informationen abzuspeichern und kann zum Teil Datenbanken ersetzen.

Erfasst am: 03.09.2002 - Artikel-URL: http://www.devtrain.de/news.aspx?artnr=798