.net Windows.Forms/WPF

Thomas Reinwart

Ist Windows Forms Entwicklung aufgrund von WPF (XAML) tot?

Antwort: noch nicht

Ein Windows Forms Projekt wird weiterhin in der Programierumgebung von Visual Studio 2008 unterstützt, die Projekte Windows.Forms und XAML existieren parallel. Auch das Ausführen von Windows.Forms Anwendungen wird noch lange möglich sein. Die Entwicklung neuer Controls wird sich aber Richtung XAML bewegen, die Anwendungen ebenfalls.

Die .net WPF (Windows presentation foundation) kommt für Desktop (Win32) Applikationen mit der .net Version 3.5 zum Einsatz und ist als Ablöse von Windows.Forms konzipiert. Die Grundlage von WPF ist XAML. (eXtensible Application Markup Language)

Geschichte

- .net 1.x: Code und Design in einem File
- .net 2.0: Partial Class, Code und Design in Files getrennt. (Designer.cs)
- .net 3.5: Code und Design in Files getrennt. (XAML offen)

XAML wird inzwischen von vielen Microsoft Produkten (Bsp. Expression Studio), aber auch von Drittherstellern unterstützt. (Bsp. Adobe Illustrator.)

WPF ist nicht mehr auf der auf Win32 API aufgesetzt (GDI+ / Pixel), sondern auf Vektor Grafik. Die Entwicklung eigener Windows.Forms basierender Controls hat immer der Ableitung eines existierenden Controls begonnen. Dann wurde die OnPaint Mehode überschrieben, das Debuggen was mühsam, beim Breakpoint wurde der Bildschirm weiß und man konnte das Ergebnis nicht sehen usw. Bei XAML gibt es dieses Problem nicht mehr.

Ein Beispiel

In Windows.Forms konnte ein Button ein Bild haben, das am Button ausgerichtet werden konnte. Bei XAML kann der Button nicht nur ein Bild einbinden, sondern „irgendwas". Die ContentControls in XAML (Button, CheckBox, …) können einen Content einbinden, das kann ein Video sein oder ein weiterer Button (ein Button im Button).

Drehen wir das Rad der Zeit etwa 10 Jahre zurück. Damals waren die verbreiteten Windows-Entwicklungsumgebungen von Microsoft VB und C++ (MFC), einige andere wie Delphi, auf nicht Windows Systemen C++, Pascal usw. Für jede der Sprachen gab es eine eigene Syntax, jede Sprache zeichnete sich durch unterschiedliche Methoden und Parametern für eine Funktion einer graphischen Darstellung (z.B. eine Kreis zeichnen) aus.

Durch die Einführung des .net Frameworks wurden VB und C++ „zusammengefasst", mit VB.net / C# / C++ /… entstanden die .net Sprachen, die alle eine gemeinsame Basis – die des .net Frameworks haben. Mit welcher .net Sprache ich auch entwickle, der Syntax ist zwar unterschiedlich, die Grundlage des Frameworks dieselbe.

In .net gibt die Trennung zwischen dem eigenen Code und der Beschreibung der Oberfläche. Bei der Windows.Forms Programmierung ist dies das Designer.cs Files. In dieser Datei wird in der jeweiligen Syntax der zuvor im Projekt gewählten .net Sprache die Oberflache, also die verwendeten Control, deren Position, Eigenschaften etc. beschrieben. Dies wird immer im Constructor in der `InitializeComponent()`-Methode initialisiert. Diese Beschreibung passt immer nur zur gewählten .net Sprache, ich kann VB.net Designer und C# Code File nicht mischen. Außerdem kann dieses Designer File auch nur vom .net Framework verarbeitet werden.

XAML kann man sich nun so vorstellen, dass die Beschreibung nochmals verbessert wurde, nämlich so, dass es unabhängig von der Sprache und der Plattform ist.

XAML wurde mit der WPF *(Windows presentation foundation)* mit dem .net framework 3.0 erstmals verfügbar. XAML basiert auf einem XML Standard. Mit XAML ist es möglich, Oberflächenelemente zu beschreiben. Damit ist eine Trennung zwischen Oberflächengestaltung und Implementierung möglich, also zwischen Designer und Entwickler. Für jeden gibt es die passenden Tools, für den Entwickler Visual Studio 2008 (mit dem sich XAML ebenfalls ändern lässt), für den Designer passenden Tools aus Microsoft Expression Studio. (Web, Design, Blend, Media).

Wie kann das in der Praxis aussehen?

Der Softwareentwickler erzeugt ein simples GUI mit XAML und implementiert den .net Code dahinter. Das XAML Files wird von Designer nachbearbeitet und optisch ansprechender gestaltet.

Für den Entwickler gibt es die bekannte Sammlung von Controls im Visual Studio 2008, nur das diese eben XAML Code erzeugen. Die Controls können in das neue Fenster hineingezogen werden, die Positionierung ist dabei möglich. Oder man kann parallel dazu im Split Fenster (Design und Code) arbeiten, oder nur im Code Fenster. Der Abgleich erfolgt automatisch. Für den optischen Feinschliff verwendet der Designer nun die Tools vom Expression Studio.

Durch die normierte Sprache XAML lässt sich eine Oberfläche plattformübergreifend und geräteunabhängig zu beschreiben. Für den Entwickler bedeutet dies eine Veränderung, denn die bisherigen Windows.Forms-Anwendungen können (überhaupt möglich?) nicht in XAML konvertiert werden. In einer Anwendung kann es jedoch parallel Windows.Forms und XAML Code geben, in getrennten Controls.

Vorteile von XAML

- Trennung von Design und Code (mittels Tools und Personen)
- Vektorgraphik anstatt Pixel
- Die Oberfläche kann zur Laufzeit verändert werden
- Einfache Handhabung von Styles
- Userexperience

Soll ich nun weiters Windows.Forms entwickeln oder gleich XAML?

Die Richtung geht nach XAML. Es gibt auch die Möglichkeit der Interopability, dabei kann Windows.Forms und XAML in einem Projekt gleichzeitig verwendet werden. Damit kann man schrittweise umsteigen.

Code Sample

```
<Window x:Class="Sample1.Window1"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="Window1" Height="300" Width="300">
  <Grid>
    <Button Height="23" Name="button1" VerticalAlignment="Top" Margin="102,26,100,0">Button</Button>
    <Label Height="23" Margin="0,26,0,0" Name="label1" VerticalAlignment="Top"
      HorizontalAlignment="Left" Width="102">Press Button</Label>
  </Grid>
</Window>
```