

Einführung in die professionelle Softwareentwicklung unter .NET mit Delphi Prism

mit Olaf Monien

In seinem Vortrag gab Olaf Monien einen Einblick in Prism. Man konnte sehen, dass es gegenüber der bisher „gewohnten“ Delphi-Programmierung doch gewaltige Unterschiede gibt. Und das fängt schon bei der Namensgebung an. Die Prism-Entwickler haben sich offensichtlich mehr nach .NET und C# orientiert, als am Win32 Delphi.

Fangen wir mal bei den für ein Programm gebrauchten Dateien an. Die früheren DFM-Dateien sind zwar immer noch Textdateien, aber haben nunmehr das XAML-Format. Das ist quasi ein Spezial-XML-Format für WPF-Programmoberflächen. Das lässt in etwa ahnen, wohin die Reise geht. Es wurde dann noch gezeigt, was man mit WPF machen kann. Und zwar wurde eine Listbox programmiert, die grafisch das Sonnensystem darstellt (mit Planetenbahnen usw.). Auf Anhieb käme wohl keiner auf die Idee, dass sich dahinter eine schlichte Listbox verbirgt. Jetzt hätte man z.B. die Möglichkeit, ein ButtonClick-Ereignis zu programmieren, das Informationen zu einzelnen Planeten liefert. Um die zu erhalten müsste man dann nicht mehr eine imaginäre Zeile der Listbox treffen, sondern die Grafik des entsprechenden Planeten.

Wer schon mal versucht hat, eine DFM-Datei von Hand zu editieren oder gar eine zu erstellen, der weiß, dass es sehr mühsam ist (ohne Delphi), eine funktionsfähige richtig hinzukriegen. Man sollte da lieber die Finger davonlassen und insofern spielt das neue Format kaum eine Rolle. Es hat aber einen nicht unerheblichen Vorteil. Statt wie vorher die DFM in die EXE mit einzulinken, kann man in Prism diese XAML-Dateien mit ausliefern. Das bedeutet im Gegensatz zu Delphi Win32 Anwendungen, dass man Layout-Änderungen auch ohne Delphi vornehmen kann. Man braucht also nicht den wichtigeren PAS-Quelltext mitsamt der Logik und den DFMs mitauszuliefern und noch den Compiler um lediglich am Layout etwas ändern zu können. Will ein erfahrener Enduser lediglich seine Buttons etc. anders anordnen, dann kann man ihm diese Möglichkeit anbieten. Ich stufe das als eindeutigen Vorteil ein.

Einige Datei-Endungen haben sich auch geändert, aber das ist nicht so wichtig. Viel wichtiger ist, was sich im Bereich der Sprache Pascal geändert hat, insbesondere bei den Schlüsselwörtern. Und da gibt es einiges, an das sich langjährige Delphi-Entwickler wohl erst mal gewöhnen müssen.

Da unter .NET ein völlig anderes Speichermodell verwendet wird, als für Win32, ein Beispiel anhand des üblichen Create/Free und wie das unter Prism aussieht. In Delphi Win32 ist es ja ungefähr so üblich :

Man erzeugt irgendetwas mit *Foo := Tfoo.Create*; und räumt das ganze dann irgendwo mit *Foo.Free* wieder weg. Natürlich kommt es vor, dass man letzteres vergisst und mit der Zeit wird der Speicher dann immer geringer. Bei .NET Programmen handelt es sich aber um „managed“ Code. D.h. .NET sorgt selbst bei einem vergessenen *.Free* dafür, dass das in der Praxis keine Auswirkungen hat (aber Achtung ! Ausnahmen bestätigen die Regel !). Das geht sogar so weit, dass es kein Free mehr gibt! Aber trotz des Speichermanagers muss man .NET wohl erst mal sagen, wenn man etwas vom Speicher abhaben will. Es wird also schon das *.Create* gebraucht. Das gibt es aber auch nicht ! Aber nicht weiter schlimm. Das heißt jetzt lediglich *new*. Das obige Create müsste also unter Prism so aussehen : *new Foo*;

Was an der Prism-Syntax noch anders ist : *Procedure* und *Function* haben z.B. ausgedient (sind allerdings wegen Abwärtskompatibilität noch vorhanden). Es gibt nur noch das Schlüsselwort *method*. Vermutlich sind die Wörter *Procedure* oder *Function* im .NET noch nicht anders vorbelegt. Bei *Create* wird's wohl anders aussehen, deshalb fällt das eben weg.

Meiner Ansicht nach eine Unsitte ist, die Variablen – Deklaration nicht an gewohnter Stelle machen zu müssen, sondern das auch an einer Stelle geschehen kann, wo man gerade Lust dazu hat. Dieses „Feature“ nennen sie „Inline-Variablen-Deklaration“. In dem Vortrag war das schon insofern nützlich und auch egal wo die Variablendeklaration steht. Nun denn, wer und wozu braucht man das ?

Strings sind grundsätzlich nullbasierend (PCchar lässt grüßen). Arrays sind ähnlich, wie von der VCL gewöhnt. Units gibt's auch nicht wie gewohnt, sondern die fallen unter *Namespace*.

Mein Fazit : schön, dass es mit Prism auch ein Pascal für .NET gibt. Das ist immer noch besser, als gezwungen zu sein C# oder VB verwenden zu müssen, sofern es überhaupt .NET sein muss. Man ist wohl trotzdem gezwungen auf den .NET-Zug aufzuspringen. Da stehen schon einige mächtige Klassen zur Verfügung und für Win32 wird es irgendwann eben zu spät sein. Es heißt eben abwarten, die Entwicklung genau zu verfolgen um den Anschluss nicht zu verpassen.