*UNIVERSITÄT KOBLENZ-LANDAU FACHBEREICH INFORMATIK*

# *.NET Framework / WPF / XAML*

Proseminar: Ausgewählte Themen der Medientechnik

**Grégory Catellani**
**Matrikelnummer: 205210618**

***Mai 2007***

# Inhaltsverzeichnis

# Das .NET Framework Modell

## Warum wurde das „Longhorn“ Application Model (heute .NET 3.0) entwickelt?

Aus Microsofts Sicht, gibt es zwei vollkommen unterschiedliche Ausrichtungen für Software. Es gibt, Windows Applikationen und Web Applikationen.

Programme für Windows nutzen dessen Features und Vorteile der auf dem PC installierten Hardware. Nachteile solcher Programme ist ihre Abhängigkeit zu dem Betriebssystem, was die Ausbreitung von für Windows geschriebenen Programmen ausbremst.

Auf der anderen Seite bringen Web Applikationen den Vorteil, dass Web Browser die Applikation progressiv downloaden. Das heißt, sie downloaden nur die Daten, die zur Anzeige der gerade aufgerufenen Seite notwendig sind. Bei heutigen Breitbandanschlüssen geht dies sehr bequem und schnell. Außerdem braucht keine Software fest auf dem Rechner installiert zu werden.

Es ist ziemlich einfach für Web Applikationen anhand von markup Sprachen Inhalte zu erstellen. Die Anforderungen an die Ersteller von web-basierten Applikationen sind daher geringer.

Web-Applikationen bringen aber auch einige Nachteile mit sich. So können sie zum Beispiel nicht offline betrieben werden. Außerdem kann es bei einigen Programmen, die für manche Operationen erst eine Anfrage an und dann eine Antwort vom Server erwarten, zu Latenzproblemen und zähflüssigen Abläufen kommen. Zusätzlich sind die schlechten Interaktionsmöglichkeiten von klassischen markup Sprachen zu erwähnen.

Bisher musste man sich entscheiden, ob man eher für die eine oder die andere Technologie entwickeln wollte.

Um diese Lücke zu schließen, wurde das Longhorn Application Model entwickelt. Es erlaubt die Vorteile beider Welten zu kombinieren, um so leistungsfähige, attraktive und hochgradig interaktive Oberflächen per Webbrowser zur Verfügung zu stellen.

Das Longhorn Application Model, erlaubt es einem Entwickler, Oberflächen deklarativ, per Markup zu erstellen. Es erlaubt reichhaltige Windows Komponenten sowie Event Handler ins User Interface zu integrieren. Texte, Graphiken und Videos lassen sich problemlos in einem intelligenten Layout und in den Einstellungen des Betrachters darstellen.

Das Modell erlaubt die Programmentwicklung in zwei übersichtliche und getrennte Teile aufzuspalten. So kann der eigentliche Programmcode abgekapselt von der graphischen

Oberfläche entwickelt werden. Änderungen an dem einen, beeinflussen so nicht den anderen Teil. Zusätzlich sollen beide Teile untereinander interagieren können.

Anhand einer .NET Sprache, wie C# oder VB.net, kann man den Programmcode erstellen. Diesen kann man dann per XAML, das auf einer XML Basis beruht, direkt anhand von entsprechenden Tags in eine Web-Oberfläche integrieren.

Ein Vorteil hierbei ist die gute Wartungsmöglichkeit der einzelnen Teile. Außerdem können solche Programme lokal oder im Web Browser auf Distanz aufgerufen werden. Dabei wird dann nur das nötigste zur Darstellung der Oberfläche aus dem Internet geladen. Alle Programmabläufe laufen auf dem Webserver. Lokal installiert, entfallen natürlich die lästigen Latenzprobleme.

Das Model stellt eine vereinte Architektur für Dokumente-, Applikationen- und Multimediainhalte dar.

## Eigenschaften des „Longhorn Application Model“

Eine Web-Applikation, die nach diesem Modell erstellt wurde, ist viel simpler und viel mächtiger, als eine auf DHTML (oder ähnlichem Werkzeug) Basis. Diese Mächtigkeit resultiert aus dem Fakt, dass man auf nahezu alle Ressourcen des .NET 3.0 Frameworks zugreifen kann. Möglich sind so:

- Reichhaltiger Multimedia Inhalt (Videos, Vektorgraphiken…),
- Mächtige Bedienelemente, wie man sie aus Desktop-Applikationen kennt (Menüs…),
- Darstellung von Dokumenten (XPS…),
- Gut strukturiertes Layout (wie bei Desktop Applikationen).

Wie man sieht, kann man Web-Applikationen entwickeln, die nahezu die gleichen Merkmale wie Desktop-Applikationen besitzen. Der einzige Unterschied hierbei ist, dass die Applikation auf einem Server läuft, und der Client so nur die benötigten Oberflächenelemente runterladen muss. Außerdem bietet dies mehr Sicherheit für den Client, da er keine Software lokal installieren muss, und Web-Applikationen allgemein mit weniger Rechten laufen. So kriegt man keine Bestätigungsnachfragen zu Gesicht, man muss nicht neustarten und man hat keine DLL-Hölle auf dem Rechner. Man braucht noch nicht mal Administrator-Rechte um Web-Applikationen zu bedienen.

# Das Graphics Device Interface

## Was ist das Graphics Device Interface?

Das Graphics Device Interface (kurz GDI) ist eine Komponente des Betriebssystems Microsoft Windows, der Pre-Vista Ära.

Es dient als Programmierschnittstelle zu den logischen Grafikgeräten (Grafikkarte, Drucker) und kapselt die Komplexität der Hardware ab.

Die meisten graphischen Operationen (wie Bitmaps und Farben) werden bei der GDI per Software, also von der CPU, berechnet. Benötigt ein Programm Informationen über gerätespezifische Eigenschaften, (Bildschirmauflösung, Bildschirmtyp) kann es sie vom Device Context beziehen.

Durch dieses Abkapselungsprinzip wurde die Ausgabe auf dem Bildschirm aber auch langsamer. So wurde, für Anwendungen die eine schnellere Grafikschnittstelle benötigen (wie Spiele), DirectX (bzw. im Speziellen DirectDraw), geschaffen.

# Das .NET Framework

## Was ist das .NET Framework?

Das .NET Framework ist eine Softwarekomponente, die einem Windows Betriebssystem hinzugefügt werden kann. Es enthält eine Reihe vorgefertigter Lösungen für allgemeine Programmieraufgaben und handhabt die Ausführung von Programmen, die speziell für dieses Framework entwickelt wurden.

Die vorgefertigten Programme decken einen breiten Bereich an Programmbedürfnissen ab. So zum Beispiel:

- (Graphische) Oberflächen,
- Datenzugriffe,
- Datenbanken,
- Kryptographie,
- Web Applikationen,
- Numerische Algorithmen,
- Netzwerk Verbindungen.

Die Basisfunktionen des Frameworks können von Entwicklern benutzt und durch eigens geschriebene Klasen erweitert werden. Dabei unterstützt das Framework verschiedenste Sprachen. Die geläufigsten sind C# und VB.NET.

.NET-Applikationen laufen, ähnlich wie bei Java, in einer virtuellen Maschine. Bei Microsoft heißt diese Common Language Runtime (CLR). Sie ist auch ein Teil des .NET-Frameworks und wird mit dem Paket geliefert. Die CLR erlaubt es Programmierern durch ihre Virtualisierung, Programme ohne spezielle Rücksicht auf eine gewisse Hardware zu schreiben. Außerdem liefert die CLR noch einige andere wichtige Merkmale. So etwa:

- Schutzmechanismen,[1]
- Speichermanagement,
- Ausnahmebehandlungen.

[1] Bei Windows Vista werden Internet Zone, Herkunftsseite, Publisher und Modulname an die „Code Access Security“ (kurz CAS) geliefert und von dieser ausgewertet. Zusätzlich läuft der Webbrowser bei Windows Vista in einer Sandbox, aus der keine Programme ausbrechen können.

Das Framework soll es insgesamt leichter machen, Programme zu schreiben. Gleichzeitig soll es dazu beitragen, die Verwundbarkeit von Programm und Computer gegenüber Bedrohungen zu minimieren.

## Designziele und prinzipielle Eigenschaften des .NET Frameworks

Das .NET Framework wurde mit verschieden Konzeptideen produziert:

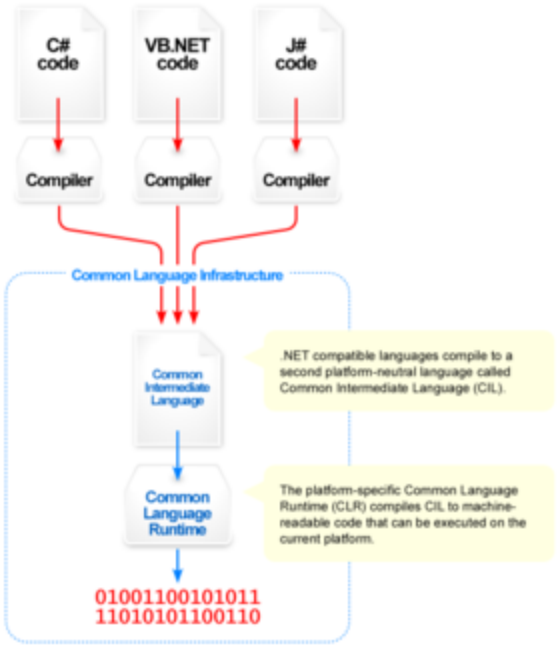
- **Interoperabilität**:

  Das Framework erlaubt es, mit Hilfe von speziellen Funktionen auf Programme, die außerhalb des .NET Frameworks laufen, zuzugreifen. Des Weiteren können diese so gesteuert werden. So ist die Kompatibilität und Interoperabilität zu älteren Programmen gewährleistet.

- **Common Runtime Engine:**

  .NET Programmiersprachen werden beim Kompiliervorgang, ähnlich Javas Bytecode, in eine Zwischensprache kompiliert. Diese wird Common Intermediate Language (CIL) und im speziellen bei Microsoft, Microsoft Intermediate Language (MIL), genannt. In Microsofts Implementation dieser Zwischensprache, wird sie bei der Ausführung nicht interpretiert, sondern anhand der just-in-time compilation (JIT)[2] in Maschinensprache umgewandelt. Die Kombination dieser zwei Methoden wird Common Language Infrastructure genannt (CLI). Microsofts Auslegung der CIL wird Common Language Runtime (CLR) genannt.

[2] Das just-in-time Kompilierverfahren bringt gegenüber einer Interpretation einige Vorteile. So müssen etwa, Schleifeninhalte bei mehrfacher Ausführung, nicht mehrmals interpretiert werden. Außerdem ist die direkte Ausführung von Maschinencode schneller.




Visuelle Übersicht der CLI

- **Sprachenunabhängigkeit:**

  Das .NET Framework stellt ein Common Type System (CTS) vor, das alle Datentypen und Programmierkonstrukte, die von der CLR unterstützt werden, spezifiziert. Außerdem schreibt es vor, welche Sprachen untereinander interagieren können und welche nicht. Durch dieses Merkmal unterstützt das .NET Framework die Entwicklung von Software in verschiedenen Programmiersprachen.

- **Basisklassen-Bibliothek:**

  Die Base Class Library (BCL), auch manchmal Framework Class Library (FCL) genannt, beinhaltet Klassen die von jeder .NET-Sprache benutzt werden können.

- **Vereinfachte Entfaltung:**

  Typische lokale Installationen von Programmen müssen sorgfältig durchgeführt werden um andere Programme nicht zu beeinflussen und immer schärfere Sicherheitsvorschriften zu erfüllen. Das .NET Framework beinhaltet Designmerkmale und Werkzeuge, die helfen, genau diese Anforderungen zu erfüllen.

- **Sicherheit:**

  .NET erlaubt es Code mit verschiedenen Vertrauensstufen laufen zu lassen und benötigt keine separate Sandbox.

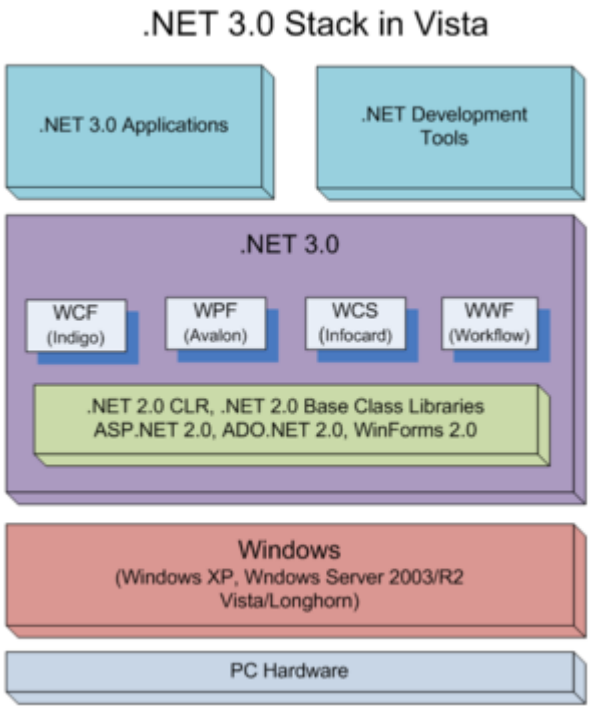
- **Plattformunabhängigkeit:**

  Das .NET Framework ist so designt, dass es Plattformunabhänigkeit unterstützt (Stichwort Silverligth, oder früher Windows Presentation Foundation Everywhere (WPF/E)). So sollen jegliche für .NET geschriebenen Programme auf egal welcher Plattform laufen. Bis heute hat Microsoft das vollständige Framework nur in das hauseigene Windows Betriebssysteme integriert. Teile des Frameworks wurden für andere Betriebssysteme portiert. Doch sind diese Teilimplementationen weder reichhaltig, noch werden sie bisher oft verwendet.

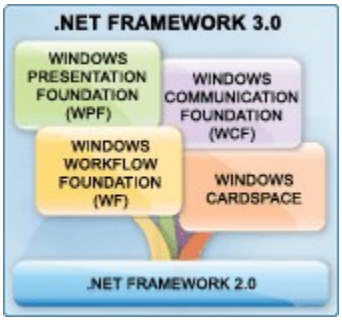
## Das .NET 3.0 Framework

Das .NET 3.0 Framework, früher WinFX oder auch Longhorn Application Model genannt, enthält eine neue Palette von managed code[3] APIs (Application programming interface), die ein fester Bestandteil von Windows Vista und dem Windows Server „Longhorn“ Betriebssystem sind.

Im Wesentlichen baut das .NET 3.0 Framework auf der älteren Version 2.0 auf, doch beinhaltet es auch einige Neuerungen und Erweiterungen. Hier eine Auflistung und kurze Beschreibung der vier wichtigsten neuen Komponenten:




[3] Managed Code sind in Microsoft Terminologie Computerbefehle, die von CLI-kompatiblen virtuellen Maschinen (wie Microsoft .NET Framework Common Language Runtime) ausgeführt werden können.




- **Windows Presentation Foundation (WPF)**

  Die WPF, früher unter dem Codenamen Avalon entwickelt, ist ein neues User Interface Subsystem, das auf einer XML-API-Basis sowie Vektorgraphiken gestützt ist. Es bedient sich der 3D Computergraphik Hardware (GPU einer Grafikkarte) und Direct3D Technologien.

- **Windows Communication Foundation (WCF)**

  Die WCF, früher unter dem Codenamen Indigo bekannt, ist ein dienstorientiertes Nachrichtensystem, das es unterschiedlichen Programmen erlaubt lokal oder über Distanz zu interagieren.

- **Windows Workflow Foundation (WWF)**

  Die WWF erlaubt das Einrichten von Prozessautomatisierung und integrierten Transaktionen anhand von Workflows[4].

- **Windows CardSpace (WCS)**

  Das WCS, während der Entwicklungszeit auf den Codenamen InfoCard getauft, ist eine Softwarekomponente die die digitale Identität einer Person sicher aufbewahrt, sowie ein vereinheitlichtes Interface für die Auswahl einer Identität bei einer Transaktion bereitstellt.

[4] Workflow: das Durchlaufen einzelner Etappen eines Arbeitsprozesses durch Dokumente oder Anwendungen

## .NET vs. Java

Die CLR und C# haben viele Ähnlichkeiten mit Sun´s JVM und Java. Gemeinsam haben sie, unter anderem, folgende wichtige Punkte:

- Beide basieren auf einer virtuellen Maschine, womit sie Details über den Computer, auf dem sie laufen, verstecken.
- Beide benutzen ihren eigenen ByteCode (CIL gegen Java bytecode) als Zwischenschritt.
- Bei .NET wird der ByteCode JIT-kompiliert. Der Java Bytecode kann sowohl JIT-kompiliert als auch interpretiert werden.
- Beide bieten reichhaltige Klassenbibliotheken, die viele Programmvoraussetzungen erfüllen.
- Beide adressieren eine Menge Sicherheitsangelegenheiten.
- Der Namensraum des .NET Framework ähnelt stark den Plattformpaketen der Java EE API-Spezifikation, sowohl im Stil, wie auch beim Aufruf.
- Beide sind größtenteils Plattformunabhängig

Während das .NET Framework aber noch nicht vollständig auf allen Plattformen integriert ist, läuft Sun´s Java ohne Weiteres auf den unterschiedlichsten Plattformen.

Im Gegensatz zu Java, wurde .NET von vorneherein entwickelt, um mehrere Programmiersprachen zu unterstützen. Die Java Plattform hingegen kennt nur Java und wurde darauf ausgerichtet auf den unterschiedlichsten Plattformen gemäß dem Motto, „Write once, run anywhere“ zu laufen.

## Kritik zum .NET Framework

- Durch die Einführung des .NET Frameworks, wurde die alte Visual Basic Sprache durch die neue Visual Basic.NET ersetzt, was für einige Unruhen im Lager der Transitionsgegner sorgte.
- Es gibt zwischen den einzelnen .NET Versionen Inkompatibilitäten. Diese halten sich aber weitestgehend in Grenzen und sind gut kommentiert.
- Programme, die in einer virtuellen Maschine, wie Microsoft CLR oder Sun´s JVM laufen, neigen dazu, mehr Ressourcen zu verbrauchen, als Programme, die direkten Zugriff auf die Computerressourcen haben. Allerdings gibt es auch einige Programme, die in einem .NET Umfeld schneller laufen als ihre nicht .NET Vorversionen. Dies könnte auf effektivere Funktionen des .NET Frameworks hindeuten, oder aber auch durch das JIT-Compiling oder andere Aspekte der CLR zurückzuführen sein.

# Die Windows Presentation Foundation

(Codename: Avalon)

## Einführung – WPF

Die WPF, früher unter dem Codenamen Avalon entwickelt, ist das neue User Interface Subsystem vom .NET 3.0 Framework (früher WinFx genannt), das auf einer XML-API-Basis sowie Vektorgraphiken gestützt ist. Es bedient sich der 3D Computergraphik Hardware (GPU einer Grafikkarte) und Direct3D Technologien. Es ist bei Vista vorinstalliert und wird es auch bei Longhorn Server sein. Auf früheren Windows Systemen (XP und 2003) kann man das .NET 3.0 Framework aber problemlos nachinstallieren.

Die WPF bringt ein konsistentes Programmiermodell zur Applikations-Erstellung mit sich und bietet eine klare Trennung zwischen graphischer Oberfläche und Programmlogik. Eine WPF Applikation kann sowohl lokal auf dem Desktop oder aber auch in einem Webbrowser laufen. Außerdem erlaubt es bessere Kontrolle, Design und Entwicklung für Windowsprogramme. Die WPF zielt darauf, eine grosse Anzahl an Diensten gleichzeitig anzubieten.  Diese sind unter Anderem: Graphische Oberfächen, zwei- und dreidimensionale Zeichnungen, feste und anpassungsfähige Dokumente, erweiterte Typographie, Vektorgraphiken, Rastergraphiken, Animationen, Datenverbindungen, Audio und Video.

Die Plattformunabhängigkeit soll durch Microsoft Silverligth (früher WPF/E) sichergestellt werden, indem Programme auf einem Host laufen und nur die benötigten Inhalte runtergeladen und im Webbrowser dargestellt werden.

## Eigenschaften der WPF

Auflistung einiger der wichtigsten WPF Eigenschaften:

- **Graphische Dienste:**
  - Alle graphischen Elemente, darunter auch Windows Desktop-Gegenstände, laufen über Direct3D.
    - Dies soll eine vereinheitlichte Art der Darstellungsweise darstellen und gleichzeitig erweiterte graphische Möglichkeiten erlauben.
    - Die graphische Darstellung über Direct3D ermöglicht es, die GPU der Grafikkarte zu nutzen und die CPU zu entlasten.
    - Die WPF ermöglich den Einsatz vektor-basierter Graphiken, die eine unendliche verlustfreie Skalierung erlauben.
    - Zudem erlaubt die WPF, 3D-Modell Rendering und Interaktion in 2D-Applikationen.
  - Spezialeffekte
    - Die WPF ermöglicht Bitmap Effekte leider nur in Software. GPU Eingenschaften wie Pixel Shader können nicht genutzt werden.
    - Spezialeffekte wie Schattenwurf und Unschärfe sind eingebaut.
    - Andere Effekte, wie etwa Spiegelungen, können leicht implementiert werden.

- **Ausbreitung:**
  - Die WPF erlaubt mehr als nur traditionelle standalone Applikationen. Sie können sowohl lokal oder durch XAML Browser Applikationen im Webbrowser implementiert und dargestellt werden:
    - Lokale Programme sind lokal installierte Programme. Sie werden als voll vertrauenswürdig eingestuft und haben Zugriff auf alle Computerressourcen.
    - XAML Browser Applikationen sind auf einem Server gehostet und werden über einen Webbrowser angezeigt. Sie laufen in einer partiell vertrauten Sandkastenumgebung und haben keinen direkten Zugriff auf Computerressourcen. Obwohl es den Anschein hat, sie würden sofort im Browserprozess laufen, laufen sie aber in Wirklichkeit in einer out of process exe, die nichts mit dem Browser gemeinsam hat. Solche Programme laufen ohne Installationsabfrage.

- **Interoperabilität:**
  - Die WPF enthält Möglichkeiten auf nicht .NET-sprachigen Code zuzugreifen. So lassen sich auch alte Programme in das neue .NET Gewand kleiden.

- **Media Dienste:**
  - Die WPF bringt eine Anzahl Grundformen für 2D Graphiken mit sich.
  - Die 3D-Fähigkeiten der WPF sind eine Teilmenge der Direct3D-Möglichkeiten. Dafür bringt die WPF aber ein bessere Integratonsmöglichkeiten für Graphische Oberflächen, Dokumente und Medien mit sich. Dies erlaubt 3D-UIs, 3D-Dokumente, sowie 3D-Medien.
  - Die meisten gebräuchlichen Bildformate werden unterstützt. Zusätzliche Unterstützung für andere Formate kann ein Entwickler anhand der Windows Imaging Component (WIC) erstellen.
  - WPF unterstützt als Videoformate: WMV, MPEG sowie einige AVI Dateien.
  - Die WPF ermöglicht Text Rendering anhand von ClearType. Außerdem unterstützt sie OpenType Fonts.

- Datenbindungen:
  - Die WPF enthält einen Satz verschiedenster Datendienste, die es Entwicklern ermöglichen soll Daten in ihre Applikationen einzubinden und zu manipulieren.
  - Das alleinige Einbinden von Daten bestimmt nicht deren Darstellung. Dies kann anhand von vorgefertigten Templates vom Entwickler bewerkstelligt werden.
- Benutzerobefläche:
  - Die WPF beeinhaltet eine Reihe von Kontrollelementen für graphische Benutzerumgebungen, wie Buttons und Menüs.
  - Ein mächtiges Konzept der WPF ist die logische Trennung der Kontrollstruktur und deren graphischen Darstellung.
    - So kann eine graphische Vorlage überschrieben werden um die visuelle Darstellung einer Applikation zu verändern, ohne die eigentliche Programmlogik zu verändern. Dies funktioniert auch andersherum.

# Microsoft Silverlight

(Codename: Windows Presentation Foundation Everywhere)

## Was ist Silverlight?

Bei Silverlight handelt es sich um eine stark abgespeckte plattformunabhängige Version der WPF auf XAML und Javascript Basis. Silverlight erlaubt Flash-ähnliche Web- und Mobilapplikationen mit exakt dem selben Code wie .NET Applikationen für Windows. 3D Fähigkeiten sind zwar nicht enthalten, XPS, vektorgestützte Zeichnungen und Hardware Beschleunigung aber schon.

Silverlight wird als Add-on für den Internet Explorer 6 und höher, Mozilla Firefox, sowie Safari angeboten.

Im Moment (Mai 2007) ist die aktuellste Version von Silverlight, die im Rahmen der Entwicklerkonferenz Mix07 in Las Vegas veröffentlichte Beta Version 1.0. Die finale Version wird Mitte 2007 erwartet.

Die Version 2.0 von Silverlight soll dann eine Mini-Implementation der Common Language Runtime (CLR) mit eingeschränktem Funktionsumfang enthalten. Diese ermöglicht die Ausführung von VB.NET und C# Code in einer Sandbox ohne lokale Dateizugriffe.

Silverlight steht in direkter Konkurrenz zum Adobe Flash-Player, Ajax und Co. Im Gegensatz zur Konkurrenz, reizt Silverlight aber nur unter Windows Betriebssystemen alle Funktionen aus.

# XAML

## Was ist XAML?

XAML steht für eXtensible Application Markup Language. Es handelt sich dabei um Microsofts neue deklarative Sprache zur Erstellung von graphischen Benutzeroberflächen. XAML verfügt über eine leicht erweiterbare und lokalisierte Syntax zur Definition von UIs, getrennt von der eigentlichen Programmlogik. Man kann XAML am ehesten mit ASP.NET vergleichen, wo mit XHTML und ASP-Syntax Oberfläche und Code voneinander getrennt sind.

Der Vorteil von XAML und markup Sprachen im Allgemeinen ist ihre leichte Zugänglichkeit. So können Menschen ohne große Programmiererfahrung Oberflächen ohne besondere Schwierigkeiten erstellen, da der eigentliche Code der Applikation von der Oberfläche getrennt ist. Außerdem sind markup-gestützte Interfaces schneller zu erstellen, ihr Code ist wesentlich kürzer und einfacher zu modifizieren. Hier als Beispiel, die Erstellung eines Buttons in XAML und in Java:

- XAML:

```
<Button Click="OnClickHandler" Background="Black" Foreground="White"
Content="Test" />
```

- Java:

```
JButton button = new JButton("Test");
button.setBackground(Color.BLACK);
button.setForeground(Color.WHITE);
button.addActionListener(new ActionListener(){
        public void actionPerformed(Event e){…}
});
```

Neu ist die Möglichkeit zentral zu entwickeln, zentral zu verteilen und den Inhalt im Webbrowser anzuzeigen. Vor dem Aufkommen dieser XML-basierten GUI-Abstraktion und appletgesteuerter bzw. webservicebasierter Software waren derartige dezentrale Einsatzszenarios nur über das X Window System unter Unix möglich. Dabei musste aber jedes Programm für seine eigene GUI sorgen, das X Window System sorgte nur für die Darstellung und Kommunikation mit Maus und Tastatur, die an einem beliebigen, entfernten Rechner, stattfinden konnte.

Wenn XAML in der WPF benutzt wird, dann um reichhaltige graphische und multimediale Oberflächen zu erstellen. Die WPF, XAML und genauer genommen das Plug-In Silverlight, stehen hier in direkter Konkurrenz zu Adobe´s Flash Player.

XAML ist unter Windows Vista und Longhorn Server ohne Weiteres einsetzbar. Die Benutzeroberflächen von Vista und dem Ende des Jahres erscheinenden Longhorn Server wurden auf XAML Basis entwickelt. Unter Windows XP und Windows Server 2003 ist XAML per Zusatzsoftware nachrüstbar. Dank Silverlight kann man teilweise auch auf anderen Plattformen in den Genuss in XAML geschriebener Oberflächen kommen.

## Wie funktioniert XAML?

XAML steht in direkter Relation zur Windows Presentation Foundation. XAML Elemente repräsentieren Common Language Runtime (CLR) Klassen des .NET Frameworks. Der XAML-Tag <Button> zum Beispiel, ist sofort mit der Klasse System.Windows.Controls.Button verbunden. Da alle XAML Elemente CLR Klassen repräsentieren, kann man alles was man mit XAML macht, auch leicht in einer prozeduralen Sprache realisieren. Zu bemerken ist außerdem, dass nur als public deklarierte Klassen, welche des Weiteren  get- und set-Methoden enthalten, von XAML ansprechbar sind.

Eigens erstellte Klassen kann man anhand der Mapping und xmlns Konstrukte einbinden.

Beispiel:

```
<?Mapping XmlNamespace="animC" ClrNamespace="MSAvalon.Windows.Media.Animation"
  Assembly="Presentation Core" ?>
```

.NET Klassen werden anhand von sogenannten Tags angesprochen. Diese Technik findet man in nahezu jeder markup Sprache. Bei XAML ist es nun so, dass die Sprache auf XML Basis aufbaut. Daher lehnen sich Tags und Semantik an dieser Basis an. Zu beachten ist, dass jegliche Tags korrekt geschrieben sein, und zu einer existierenden Klasse gehören müssen. Ansonsten gibt es bei der JIT-Kompilierung Probleme, da XAML im Gegensatz zu HTML falsche Tags nicht einfach ignoriert.

Eigenschaften und Layout einzelner XAML Elemente kann man anhand von Parametern in den Tags verändern.

Da Entwickler oft auch gerne auf Ereignisse innerhalb der Oberfläche reagieren wollen, ist es möglich passende Eventhandler zu definieren. Dies kann anhand einer beliebigen .NET Sprache geschehen. Diesen Code kann man entweder – wie eigentlich vom Modell definiert – in eine externe Programmdatei schreiben, oder aber sofort in den XAML Quellcode einbetten. Dieser Code wird Inline-Code genannt und muss mit speziellen Tags versehen werden, damit er nicht fälschlicherweise als XAML Tag interpretiert wird. Der Tag <x:Code>, ist für die Einleitung des

entsprechenden Bereichs, sowie <![CDATA[…]]> um dem Interpreter zu sagen, dass es sich um Inline-Codehandelt, zuständig.

Beispiel:

```
<Canvas>
        <Button Name="button1" Click="Clicked">Click Me!</Button>
        <x:Code><![CDATA[
                Void Clicked(object sender, RoutedEventArgs e){
                        Button1.Content = "Hello World";
                }
        ]]></x:Code>
</Canvas>
```

Zu beachten ist allerdings, dass XAML-Dateien mit Inline-Code kompiliert werden müssen. Erwähnenswert ist auch die Tatsache, dass unter Windows XP und Server 2003, jegliche XAML-Datei kompiliert werden muss.

### Hierarchische Gliederung:

- Parent
    - Child

Ein oder mehrere Elemente können, abhängig von ihrer Ordnung, das Layout und Verhalten der Oberfläche beeinflussen. Jedes Element besitzt nur ein Parent, kann aber eine unbegrenzte Anzahl von Children haben. In allen XAML-Anwendungen ist das Rootobjekt typischerweise ein Panel.

## Ähnlichkeiten von XAML mit anderen markup Sprachen

XAML und HTML stellen eine Baumstruktur dar. Zwischen einem Tag können wieder unendlich viele weitere Tags stehen.

Wie bei XML oder HTML, besitzt XAML einen Root-Tag. Bei XAML nimmt man ein Vaterobjekt, z.B. ein <Window>, in das man weitere Kinder - also Elemente - einfügen kann. Für das Vaterobjekt muss das Root-Tag entsprechende Attribute xmlns und xmlns:x besitzen, die dem Parser wichtige Informationen liefern. Hier ein Codebeispiel für das Vaterobjekt Canvas.

Beispiel:

```
<Canvas>
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x=" http://schemas.microsoft.com/winfx/2006/xaml"
        …
</Canvas>
```

Beim Parsen einer XAML-Datei entsteht so ein Element-Tree, ähnlich dem DOM-Tree (Document Object Model)bei HTML.

XAML benutzt ähnlich Cascading Style Sheet (CSS) um Eigenschaften von Elementen zu beschreiben.

Beispiel:

Innerhalb eines XAML Dokumentes definiert man sein eigenes Style-Sheet für einen Button:

```
<Style def:Name="MyStyle">
        <Button Background="Red" FontSize="24" />
</Style>
```

Beim Erstellen eines Buttons verweist man dann nur noch auf sein selbst erstelltes Style-Sheet:

```
<Button Style="{MyStyle}">Styled</Button>
```

## Unterschiede zu anderen markup Sprachen

XAML erlaubt viel dynamischere und interaktivere Oberflächen, als andere herkömmliche markup Sprachen. Obwohl die in XAML geschriebene Benutzeroberfläche von der eigentlichen Programmlogik getrennt ist, können die zwei Komponenten doch sehr leicht untereinander interagieren. So sind Oberflächen mit High-End Content wie 3D-Element-Rendering, Schattenwurf, Spiegelungen, vektor-basierten Zeichnungen, und vieles mehr möglich.

Obwohl XAML keine prozedurale, sondern eine markup Sprache ist, kann und muss der Quelltext in bestimmten Fällen kompiliert, anstelle von geparsed werden. Außerdem lässt sich XAML auch in das Binary Application Markup Language Format .baml kompilieren und komprimieren. Sowohl XAML- als auch BAML-Dateien werden von der WPF interpretiert und am Bildschirm, genau wie HTML-Seiten, durch einen Browser dargestellt.

## Kritikpunkte zu XAML und der WPF im Allgemeinen

- Die Technologie ist proprietär. Die Weiterentwicklung liegt in der Hand von Microsoft.
- XAML fördert die Verwendung von WVG (Windows Vector Graphics), wodurch der vom W3C empfohlene Standard SVG (Standard Vector Graphics) geschwächt wird.

**Quellenangaben:**

MSDN - 2007 Microsoft Corporation - Chapter 1 The Longhorn Application Model

Deutsche und englische Wikipedia – Suchkriterien: .NET Framework, XAML, WPF