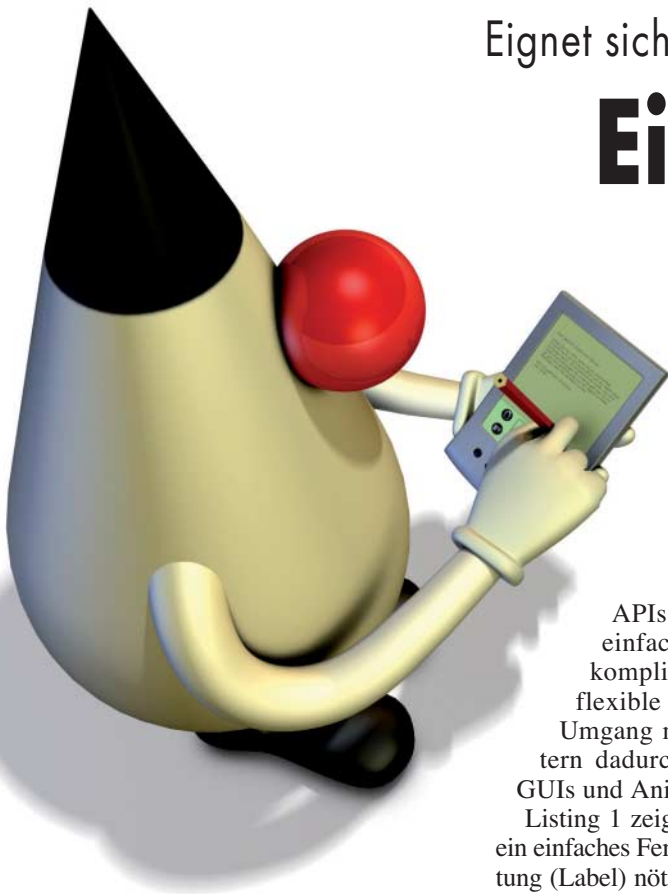


Eignet sich Suns neue GUI-Sprache für Einsteiger?

Ein erster Schritt

Mike Mannion

Mit JavaFX Script will Sun Adobes Flash und Microsofts Silverlight Paroli bieten. Erste Erfahrungen mit der Skriptsprache zeichnen ein widersprüchliches Bild.



Als wichtigste Neuheiten auf der diesjährigen JavaOne stellte Sun JavaFX Mobile (siehe Abbildung 1) und das unter der GPL erhältliche JavaFX Script (JFX) heraus. JFX soll das Erscheinungsbild mobiler Anwendungen verbessern – aber nicht nur ihres. Geplant ist der Einsatz auf allen erdenklichen Plattformen: vom Desktop bis zum Auto.

Dieser Bericht beruht auf den Erfahrungen mit JFX bei der Entwicklung von Canoos Demo-Anwendung „Music Pinboard“. Er betrachtet sowohl die Entwickler- als auch die Endbenutzerperspektive.

JFX konzentriert sich auf den gleichen Funktionsbereich wie Flex/Flash von Adobe und Microsofts Silverlight. Es ist Suns Versuch, die wachsenden Ansprüche an multimediale Fähigkeiten und Ergonomie zu befriedigen. Anders als JavaFX Mobile ist es zurzeit nur als Alpha-Version erhältlich. Seine weitere Entwicklung findet im java.net-Projekt „OpenJFX“ statt. Diese Open-Source-Strategie soll die Akzeptanz und den Einsatz in produktiven Anwendungen beschleunigen.

Im Grunde genommen leistet JFX zurzeit nicht mehr als Swing und Java-2D, denn es baut direkt auf diesen

APIs auf. Allerdings vereinfachen seine relativ unkomplizierte Syntax und der flexible Programmaufbau den Umgang mit ihnen und erleichtern dadurch die Erstellung von GUIs und Animationen deutlich.

Listing 1 zeigt, wie wenig Code für ein einfaches Fenster mit einer Beschriftung (Label) nötig ist. Für nicht-triviale Aufgaben teilt man die GUI-Spezifikation auf, wofür JFX verschiedene Möglichkeiten anbietet. Wie dieses Beispiel zeigt, spiegelt die Struktur eines JFX-Programms die der dargestellten Oberfläche wider. Dadurch verläuft die GUI-Entwicklung mit JFX intuitiver als mit Java.

Verschiedene Bedürfnisse zweier Gruppen

Aber um zu entscheiden, ob die API für Nichtprogrammierer geeignet ist, muss man mehr als die Programmstruktur berücksichtigen. Damit sie Akzeptanz gewinnt, muss eine Technik die Bedürfnisse von mindestens zwei Parteien abdecken: der Endbenutzer und der Entwickler. Von einer modernen Internet-Anwendung erwartet der Endbenutzer:

- eine Benutzerschnittstelle mit ähnlichen Funktionen wie eine Desktop-Anwendung,
- Bedienalternativen, zum Beispiel Maus, Tastatur oder Touchscreen,
- ein klares Layout und eine intuitive Anordnung von Informationen,
- direkten Zugriff auf Funktionen, das heißt Verzicht auf geschachtelte Menüs,
- angebrachte und attraktive Animationen von Text und Bildern,

- minimale Verzögerung beim Start der Anwendung, das heißt insbesondere keine zusätzlichen Downloads wie Plugins, JREs und dergleichen,
- kein Blockieren der Benutzerschnittstelle, wenn die Anwendung im Hintergrund arbeitet.

Anhand der Beispielanwendung „Music Pinboard“ wollte ein kleines Team der Firma Canoo herausfinden, inwiefern JFX die Bedürfnisse von Anwendern und Entwicklern erfüllt. Statt ein komplett neues Look & Feel zu erfinden, ließen sie sich von der Website musictonic.com (Abbildung 1) und von iTunes inspirieren (Abbildung 2).

Listing 1: Hello World mittels JFX

```
import javafx.ui.*;
Frame {
  title: "JavaFX Script"
  width: 200
  height: 100
  content: Label {
    text: "Hello iX World!"
  }
  visible: true
};
```

-TRACT

- Mit der an Java angebotenen Skriptsprache JFX Script will Sun eine eigene Technik zur Erstellung leistungsfähiger Webanwendungen etablieren.
- Mit JFX lassen sich GUIs schneller und flexibler erstellen als mit Swing und ähnlichen Java-APIs.
- Sun sieht zwar Nichtprogrammierer als Zielgruppe für JFX, doch noch fehlen für diese geeignete Werkzeuge.

Abbildung 3 zeigt das Resultat von acht Tagen Entwicklungsarbeit. Nach Eingabe eines Künstlernamens greift „Music Pinboard“ auf verschiedene Webservices zu und stellt die dort gefundenen Informationen in einer einheitlichen Übersicht dar. Daten ähnlicher Künstler, Musikalben, Videos und Fotos erfragt es bei Last.fm, Amazon, Youtube und Flickr. Die mit Java-Webstart gebündelte Anwendung steht unter www.musicpinboard.com zum Download bereit.

Die ersten drei Benutzeranforderungen erfüllt das Programm recht gut. Installation und Start entsprechen jedoch nicht den Erwartungen. Erstens ist das Paket 5,4 MByte groß, sodass das Herunterladen lange dauert, zweitens benötigt es Java 6, das eventuell erst zu installieren ist, und drittens läuft das Programm nur auf Windows. Bevor diese Nachteile zu einem voreiligen Urteil über JFX führen, gilt es Folgendes zu berücksichtigen:

– Der Kern der Anwendung mit dem Modell und dem GUI ist lediglich 90 KByte groß. Die aktuelle JFX-Laufzeitumgebung besteht aus 2,2 MByte. Zukünftig könnte diese Umgebung auf dem Zielgerät vorhanden sein. Für die restlichen 3 MByte zeichnen die eingesetzten Webservice-APIs verantwortlich. Mit zusätzlichem Entwicklungsaufwand ließe sich dafür eine schlankere Lösung finden.

– Grundsätzlich setzt JFX zwar nur Java 5 voraus, Java 6 bietet jedoch eine bessere grafische Wiedergabe. Das ansehnliche Resultat bezahlt der Anwender mit dem zeitaufwendigen Herunterladen der passenden JRE, die es bislang noch nicht für alle Plattformen gibt.

– Die Abhängigkeit von Windows hat einen einfachen Grund: Zur Video-wiedergabe dient eine native Bibliothek des Projekts JDesktop Integration Components (JDIC). Zwar unterstützt JDIC auch andere Plattformen, allerdings hat sich das Canoo-Team im Rahmen des Projekts auf Windows konzentriert. Enthielte JFX bereits eine plattformunabhängige Video-API, fiel diese Einschränkung weg.

– Java-Webstart ist nicht besonders benutzerfreundlich. Beim Start erscheint ein Fenster, das den Benutzer nach seinem Vertrauen in das Anwendungszertifikat fragt. Die meisten Anwender dürfte diese Frage verwirren oder gar abschrecken.

Aus Sicht des Nutzers stellen die lange Zeit für das erstmalige Herunterladen und die Notwendigkeit, Java 6

In Apples iTunes zeigt das Karussell die Cover gespielter Alben (Abb. 2).



Die mittlerweile geschlossene Site musictonic.com fasste Informationen über Musiker zusammen (Abb. 1).

bereitzuhalten, die größten Hürden dar. Beides sind keine JFX-spezifischen Phänomene, sondern der uralte Grund für den ständig schrumpfenden Marktanteil von Java Applets. Wären allerdings die benötigten JRE- sowie die JFX-Dateien bereits auf dem Zielgerät vorhanden und ließe sich das Zertifikat automatisch verifizieren, wäre der Abstand zur Konkurrenz geringer.

Was Entwickler sich von JFX wünschen

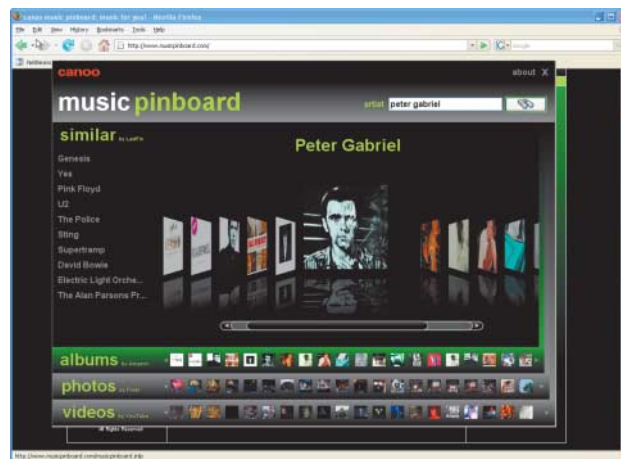
Neben den erwähnten Anwenderforderungen erwarten Entwickler von der eingesetzten Technik:

- klare, möglichst einfache Syntax,
- eine deutliche Trennung von Code-Bereichen („separation of concerns“), etwa zwischen Modell und Sicht,
- einwandfreie Anbindung an Informationsquellen wie Webservices,
- Wiederverwendbarkeit von Code, einschließlich Java und JFX,
- hilfreiche Entwicklungswerkzeuge.

Nach acht Tagen Entwicklung kombiniert die JFX-Anwendung musictonic und iTunes (Abb. 3).

Eines der Hauptmerkmale von JFX ist seine deklarative Syntax. Anders als Sprachen wie Java, die expliziten Code für das Aktualisieren von Variablen und Properties benötigen, reicht hier die Deklaration der Beziehungen zwischen diesen Entitäten. Das Laufzeitsystem erledigt anschließend das Synchronisieren (ein Beispiel folgt). JFX bietet gegenüber Java einen einfacheren Umgang mit Listen: Zur Deklaration dient ein spezielles Konstrukt von eckigen Klammern; *insert* fügt Elemente hinzu, und *delete* entfernt sie.

Ähnlich wie Java kennt JFX Klassen, die von JFX- und Java-Klassen erben können. Allerdings gibt es in einer JFX-Klasse keinen Konstruktor, sondern einen „Trigger“, den die Laufzeitumgebung beim Erzeugen von Objekten oder beim Ändern von Attributen auslöst, und der beliebige Aktionen durchführen kann.



Listing 2a: Die Modellklasse (Ausschnitt)

```
public class MusicPinboardModel {
    attribute searchText: String;
    attribute artist: String;
    attribute selectedServiceModel: AbstractServiceModel;
    attribute selectedIndex: Number;
    attribute playbackModel: PlaybackModel;
    :
}
```

Listing 2b: Die Sichtklasse (Ausschnitt)

```
:
model = MusicPinboardModel {};
:
Carousel {
    imageURLs: bind model.selectedServiceModel.data
    selectedIndex: bind model.selectedIndex
    titleText: bind model.artist
    playbackModel: bind model.playbackModel
}
```

bind-Anweisungen koppeln Attribute in Sicht- und Modellklasse.

Wie die Entwicklung des „Music Pinboard“ zeigte, ist JFX keine schwierige Sprache. Man muss sich jedoch aufgrund der deklarativen Eigenschaften eine neue Denkweise angewöhnen. Dennoch beklagen sich bereits viele Java-Programmierer, JFX enthalte unnötige Abweichungen von Java-Konventionen. So lesen sich *not*, *and* und *or* zwar besser als die Java-Entsprechungen *!*, *&&* und *//*. Da die ersten JFX-Entwickler aus der Java-Welt stammen, bürdet ihnen die neue Notation jedoch unnötiges Umdenken auf und verlangsamt das Lernen.

Details dank JFX später klären

Unabhängig von Einzelheiten der Syntax bleibt es für die Wart- und Testbarkeit einer Anwendung entscheidend, ob die Sprache eine klare und logische Aufteilung der einzelnen Funktionsbereiche ermöglicht. Die vielen Interfaces, Ereignisklassen und das Hinzufügen beziehungsweise Entfernen von Listeners in Modellen erschweren Swing- und SWT-Programmierern das Leben. JFX' deklarative Syntax ermöglicht es ihnen hingegen, im Nachhinein die Attribute einer Sicht an die eines Modells zu „binden“. Ändern sich die Attribute im Modell, aktualisiert die Laufzeitumgebung daran gebundene Eigenschaften des GUI automatisch. So bleibt die Trennung von Modell- und Sicht-Code in JFX elegant und übersichtlich

(siehe Listing 2a und 2b). Die Produktivität steigt, und das Programm lässt sich besser warten.

Übertreibt man den Einsatz animierter Grafikelemente, wirkt eine Anwendung rasch unprofessionell, doch können sie an den richtigen Stellen den Gesamteindruck durchaus verbessern. JFX ermöglicht animierte GUI-Elemente durch das asynchrone und regelmäßige Aktualisieren von Variablen. An sie gebundene Objektattribute modifiziert die Laufzeitumgebung analog, sodass die Form des Objekts gemäß den Veränderungen der Attribute variiert. Listing 3 illustriert das Vorgehen: Mit *dur* und *bind* lassen sich Attribute in beliebigen Schritten und mit beliebigem Tempo modifizieren. Man kann GUI-Elemente einschweben, drehen, skalieren, ein- und ausblenden und so weiter.

Alles in allem ist es mit JFX recht einfach, animierte Benutzerschnittstellen zu konstruieren. Allerdings hängt das Ergebnis je nach Größe und Komplexität der Animation stark von der Leistung der Laufzeitumgebung ab. Das zugrunde liegende Java wird nie so flott sein wie native 3D- oder Animationsbibliotheken.

Wie ihre Entsprechungen in Java funktionieren *package* und *import*, die JFX- und Java-Klassen importieren. Die Fähigkeit, existierende Klassen zu verwenden – direkt oder durch Vererbung – beschleunigte bei „Music Pinboard“ die Entwicklung erheblich.

Plug-ins für IDEs mit Hindernissen

Obwohl JFX erst im Alphastadium ist, gibt es bereits Netbeans- und Eclipse-Plug-ins dafür. Für die Realisierung des „Music Pinboard“ kam in erster Linie Eclipse unter Windows XP zum Einsatz. Diese Plug-ins bieten vor allem Code-Vervollständigung und Hinweise beim Verstoß gegen Sprachregeln. Allerdings kommen diese Nachrichten häufig zu kryptisch daher und nutzen dem JFX-Anfänger nur wenig.

Ein Versuch mit Mac OS X wurde schnell abgebrochen, denn das JFX-Plug-in brachte die dortige Eclipse-Umgebung regelmäßig zum Stillstand. Ganz stabil ist das Eclipse-Plug-in auch auf Windows nicht. Gelegentlich laufen die Programme nicht, melden aber keine Fehler. Das erfordert ein Aufräumen des Projekts per „Clean“

oder im schlimmsten Fall einen Neustart der Entwicklungsumgebung. Trotzdem hilft das Plug-in, Schwachstellen schneller zu identifizieren, obwohl es die genaue Ursache nicht verrät.

Sollen, wie von Sun gewünscht, Nichtprogrammierer eine Programmiersprache wie JFX benutzen, brauchen sie geeignete Werkzeuge. Unter dem Namen JFXBuilder bietet Reportmill ein solches grafisches Entwurfstool an. Nach dem Gestalten des GUI mit dem Point-and-Click-Designer kann man die Animationen durchspielen und den generierten JFX-Code anschauen. Eine denkbare Arbeitsweise bei anspruchsvolleren Anwendungen wäre, dieses Werkzeug für die Generierung des ersten Entwurfs anzuwenden und nachher mit Java-Code zu kombinieren. JFXBuilder ist neu und noch eingeschränkt im Funktionsumfang. Trotzdem zeigt er, in welche Richtung weitere Entwicklungen gehen müssen, wenn man Webautoren und -designer für JFX interessieren möchte.

Listing 3: Animationen per JFX

```
Canvas {
    var duration = 2500
    var fade = [0,0,0,0,1,1,0] dur (1.5 * duration) linear
    var xPos1 = [230,150] dur duration easeboth
    var xPos2 = [230,306] dur duration easeboth
    :
    background: black
    content: [
        View {
            transform: bind [translate(xPos1, 120),
                scale(2, 2)]
            content: SimpleLabel {
                text: "music"
                foreground: Color {
                    red: 255
                    green: 255
                    blue: 255
                }
                opacity: bind fade
            }
        },
        View {
            transform: bind [translate(xPos2, 200),
                scale(1.5, 1.5)]
            content: SimpleLabel {
                text: "pinboard"
                foreground: Color {
                    red: yellowgreen.red
                    green: yellowgreen.green
                    blue: yellowgreen.blue
                }
                opacity: bind fade
            }
        }
    ]
    :
}
```

Listing 4: Groovy sagt Hallo

```
<is!>
import groovy.swing.SwingBuilder
swing = new SwingBuilder()
frame = swing.frame(title:'Groovy') {
    panel {
        label 'Hello iX World!'
    }
}
frame.pack()
frame.show()
```

Die Frage liegt nahe, weshalb Sun nicht Groovy als Basis für eine einfachere GUI-Erstellung gewählt hat. Was das offizielle JavaFX-Projekt auf diese Frage antwortet, überzeugt nicht: Groovy und andere Sprachen seien generisch und stellten keine für die Optimierung des Entwurfsprozesses geeignete Abstraktion zur Verfügung. Außerdem seien sie explizit für Programmierer und nicht für Webautoren entworfen.

JFX ist eine domänenspezifische Sprache für GUI-Erstellung, die dem Groovy SwingBuilder ähnelt (siehe Listing 4). Groovy verfügt über dieselben Abstraktionen wie JFX, und Webautoren arbeiten mit beiden Sprachen auf dieselbe Art. Dass Groovy darüber hinaus eine vollständige Programmiersprache ist, kann allenfalls ein Vorteil sein. Das heißt, der heutige Mangel an direkter Unterstützung von JFX-artigen Konzepten sowie der im Normalfall positiv zu wertende generische Ansatz sind keine Argumente für eine neue Sprache.

Argumente für und gegen Groovy

Pragmatische Einwände gegen den Einsatz von Groovy wären, dass seine Laufzeitumgebung für mobile Geräte zu groß und die meisten seiner gelieferten Fähigkeiten überflüssig sind. Groovy müsste, wie das JDK, vorinstalliert werden.

Die wahren Vorteile von JFX gegenüber Groovy liegen im verhältnismäßig geringen Speicherbedarf (zusätzlich zu JavaME/SE, Swing und Java-2D) sowie seinem Schwerpunkt auf GUI-Interaktion und Animation, was besonders elegante Speziallösungen ermöglicht.

Ob mit Java, Flash oder Silverlight realisiert, die Technik hinter einer Anwendung dürfte den meisten End-

benutzern gleichgültig sein – Hauptsache, sie erfüllt ihre Erwartungen, läuft einwandfrei und ohne Zusatzoperationen. Programme wie „Music Pinboard“ zeigen, dass selbst die Alpha-Ausgabe von JFX die Gestaltung eines modernen interaktiven und benutzerfreundlichen Programms ermöglicht. Jedoch behindern die altbekannten Mängel der Verbreitung von Java-Anwendungen seinen schnellen Start.

Fazit

Aus Entwicklersicht hat die neue Technik ebenfalls einige Vor- und Nachteile. Zwar lassen sich mit JFX Benutzerschnittstellen deutlich schneller als mit Swing oder ähnlichen GUI-Werkzeugen erstellen. Die neue Skriptsprache ist nicht kompliziert, aber hinsichtlich Syntax und Konsistenz bleiben einige Wünsche offen. Erfüllten die JFX-Ersteller diese, beschleunigten sie den Lernprozess und die Akzeptanz bei Entwicklern deutlich. Obwohl die Animations- und Effektkonstrukte relativ primitiv sind, ließen sie sich dank flexibler Code-Gestaltung und Objektorientierung zu mächtigen Effektbibliotheken ausbauen. Solange es keine produktionsreifen grafischen Werkzeuge gibt, können Grafiker und Webautoren JFX ohne Java- und allgemeine technische Kenntnisse nicht nutzen.

Damit JFX beziehungsweise JavaFX Mobile nicht den gleichen Abgang wie Applets erlebt, müssen Sun, seine Partner und das OpenJFX-Projekt noch viel leisten. (ck)

MIKE MANNION

beschäftigt sich unter anderem mit Java- und Webtechniken und ist Senior IT-Consultant bei der Canoo Engineering AG.

Onlineresourcen

JavaFX Script	sun.com/javafx/
JaxaFX Mobile	sun.com/javafxmobile/
Music-Pinboard-Blog	www.canoo.com/blog/2007/06/15/javafx-script-canoo-music-pinboard/
Music Pinboard download	www.musicpinboard.com
Chris Oliver (JFX Erfinder)	blogs.sun.com/chrisoliver/category/JavaFX
Projekt OpenJFX	openjfx.dev.java.net/
JFX IDE Plug-ins	openjfx.dev.java.net/javafx-eclipse-plugin-install.html Openjfx.dev.java.net/javafx-nb55-plugin-install.html
JFXBuilder	www.reportmill.com/jfx/
Groovy	groovy.codehaus.org/

