Entwicklung mit JavaFX

Für die Entwicklung von Client-Web-Anwendungen wurde JavaFX von Oracle als strategische Technologie innerhalb der Java Plattform für den Desktop vorgestellt. JavaFX 2 erfährt eine Renaissance mit nachhaltiger Verstärkung der zuständigen Entwickler-Teams und klar definierter Roadmap für die kommenden Versionen mit dem Ziel JavaFX 3.0 im JDK 8 zu paketieren und auszuliefern. Die aktuelle Version JavaFX 2.1 wird bereits mit dem JDK 7u4 für MS Windows und Mac OS X ausgeliefert.

Die eigens für JavaFX 1.0 geschaffene Skriptsprache ist mit JavaFX 2.0 vollständig verschwunden und man setzt jetzt auf eine einheitliche Java-Entwicklung mit neuer JavaFX Bibliothek für den Desktop. Das Augenmerk liegt dabei auf verbesserter Browser-Integration, dem Styling vom UI und auf Animationen. Die verfügbaren API´s sind vereinfacht worden sodass sie den Java-Entwicklungsprozess mit seinen Werkzeugen besser unterstützen. Damit gestaltet sich der Einsatz von JavaFX deskriptiv und programmatisch und führt zu einer schnelleren Umsetzung bei der Entwickltung von Desktop-Anwendungen. JavaFX 2.0 liefert einige UI-Controls (Charts, Tabellen, Menüs) und ein API für selbst erstellte Controls mit der Maßgabe weitere folgen zu lassen um den konkreten Bedarf an ausgereifter Desktop-Funktionalität zu decken und langfristig Swing abzulösen.

Java als Programmiermodell für JavaFX 2

Die Einführung von Java API's für JavaFX bietet mit Generics, Annotationen, und Multithreading eine gewohnte Programmiermodellunterstützung mit vorhandenen Java-Sprachmerkmalen an. Beim Design der API's versuchte man eine bewußte Alternative zu dynamisch typisierten Sprachen zu schaffen. Die Funktionalität von JavaFX wird dem Entwickler über die API's direkt angeboten um vorhandene Java-Werkzeuge für Code Refactoring, Debugging und Profiling auch für die Erstellung von JavaFX-Anwendungen nutzen zu können.

Architektur für schnelle Grafikverarbeitung

Die neue JavaFX Grafik-Engine unterstützt moderne Grafikprozessoren (Graphics Processing Units) mit der Hardware-beschleunigten Grafik-Pipeline über Prism für schnelles Rendering von 2D- und 3D-Elementen (siehe Abbildung 1). Web-Seiten die in einer JavaFX Anwendung als Web-Komponente eingebunden sind werden mittels Web-Kit HTML Rendering Technologie und Prism dargestellt. Prism spielt mit dem Glass Windowing-Toolkit zusammen und macht die Darstellung umfangreicher grafischer Oberflächen schneller und wird in getrennten Threads für die Anwendung und das Rendering im Hintergrund synchronisiert ohne das es der Entwickler bemerkt. Ein JavaFX Plug-In ermöglicht das Laden von Prism-basierten Applets im Browser. Die neue JavaFX Media Engine unterstützt den schnellen Ablauf von Web Multimedia Content in der Playback-Funktion mit dem GStreamer Multimedia Framework.

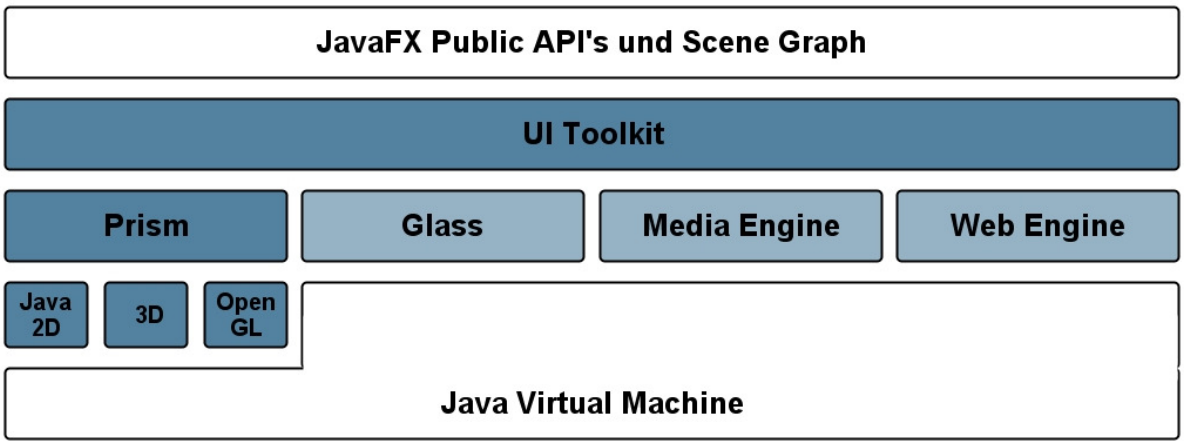

Abbildung 1: JavaFX Architektur

Entwicklungsumgebungen und Scene Builder

Die Verwendung von JavaFX in der Entwicklungsumgebung NetBeans mit MS Windows XP Betriebssystem erfolgt durch Konfiguration mit dem Java Plattform Manager über die Definition einer Java Plattform Version (JDK 6 oder JDK 7) und Einbindung der *JavaFX 2.0 Runtime* mit der Datei `fxrt.jar` aus dem Verzeichnis `\JavaFX 2.0 Runtime\lib` (Siehe Abbildung 2).

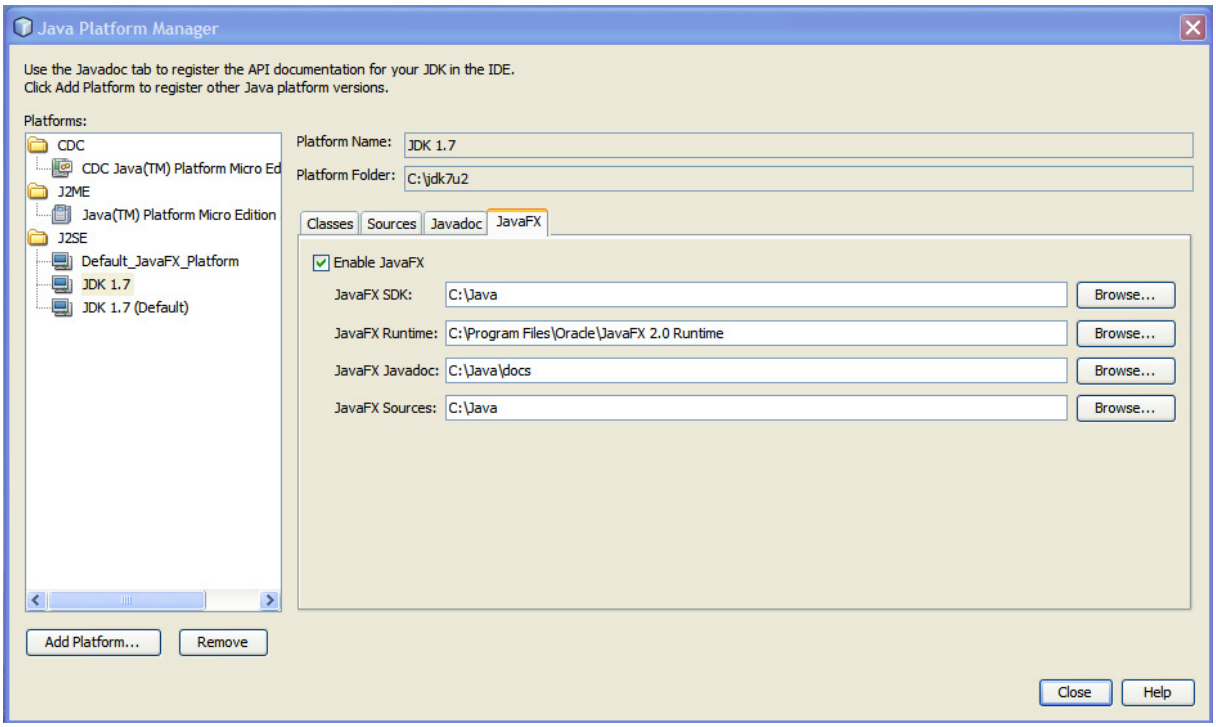

Abbildung 2: NetBeans Java Plattform Manager mit JavaFX

Mit nahezu jeder Java Entwicklungsumgebung kann man JavaFX verwenden. Zur Auswahl stehen NetBeans, Eclipse IDE über das JavaFX-Eclipse-Integrationsprojekt e(fx)clipse, IntelliJ IDEA, JDeveloper/ADF. JavaFX-Unterstützung existiert auch für

Groovy über das GroovyFX API und für Scala mit ScalaFX (Scala Bindings for JavaFX 2). Durch Einführung der neuen XML-basierten deklarativen Markup Sprache FXML kann das Benutzer-Interface der JavaFX-Anwendung unabhängig von der jeweils verwendeten Entwicklungsumgebung aufgebaut werden, ohne IDE-spezifische Fragmente nachpflegen zu müssen. Bei Layout-Änderungen ist das vorteilhaft da keine erneute Code-Kompilierung notwendig ist.

Der neue JavaFX Scene Builder 1.0 als Beta-Version (Siehe Abbildung 3) ist ein visuelles Layout-Werkzeug zum Design von JavaFX Anwendungsoberflächen ohne Kodieren zu müssen. Im Scene Builder Editor können UI-Komponenten per drag und drop im Arbeitsfeld der Oberfläche platziert werden, deren Eigenschaften verändert und Style-Sheets zugeordnet werden. Für das gewählte Layout wird automatisch das passende FXML erzeugt und in einer FXML-Datei abgelegt, die in ein Java-Projekt einer Entwicklungsumgebung integriert wird und die JavaFX-Oberfläche mit der Applikationslogik verbunden wird. Mit dem eingebauten Scene Builder Preview kann im Menü über *Preview in Window* die Oberfäche von `IssueTracking.fxml` auf dem Desktop sichtbar gemacht und gemäß ihrer Funktionalität verwendet werden.

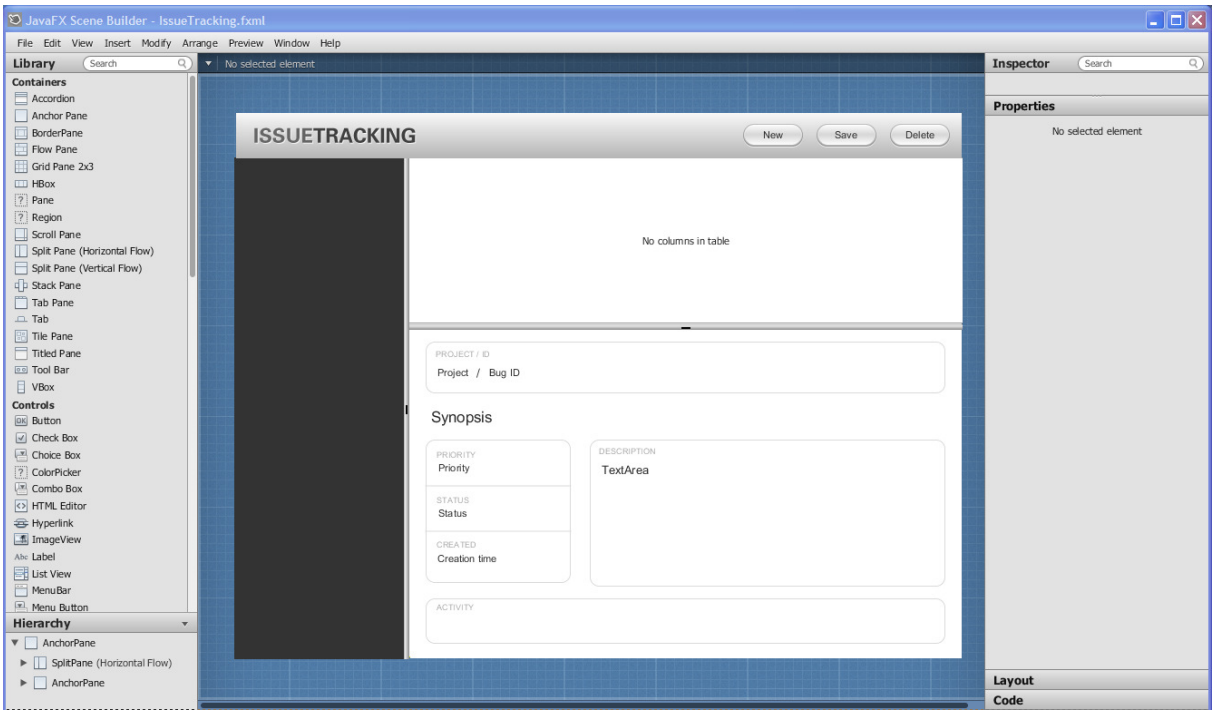

Abbildung 3: JavaFX Scene Builder 1.0

Wie im JavaFX Scene Graph in Abbildung 4 dargestellt, verwendet JavaFX zuerst eine Stage, um die Bühne zu eröffnen und darin eine Scene mit Parent- und Child-Nodes anzuordnen. Die Scene wird mit Layout und Controls verknüpft und in der JavaFX-Anwendung über ein Event-Framework gesteuert. Dies zeigt das Code-Beispiel *javafxapplication1* (Hello World) im Listing 1 bei der ein Button „Say Hello World“ in einer Oberfläche erscheint und beim Betätigen der gedrückte Zustand über ein *ActionEvent* vom Event Handler ausgewertet wird und im IDE-Output den String „Hello Alexander!“ ausgibt.

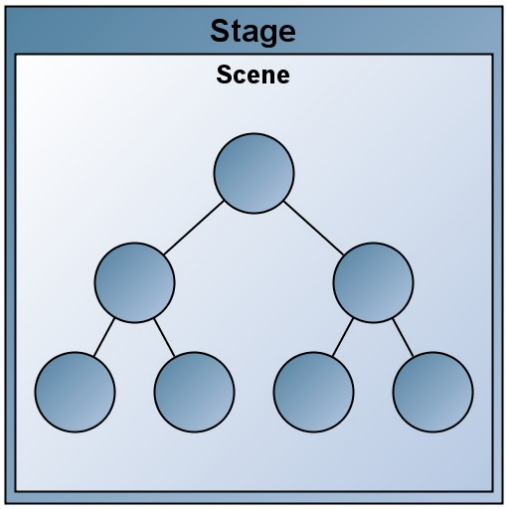

Abbildung 4: JavaFX Scene Graph

Listing 1

```
package javafxapplication1;

import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

public class JavaFXApplication1 extends Application {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Hello World!");
        Button btn = new Button();
        btn.setText("Say 'Hello World'");
        btn.setOnAction(new EventHandler<ActionEvent>() {

            @Override
            public void handle(ActionEvent event) {
                System.out.println("Hello Alexander!");
            }
        });

        StackPane root = new StackPane();
        root.getChildren().add(btn);
        primaryStage.setScene(new Scene(root, 300, 250));
        primaryStage.show();
    }
}
```

Verschiedene Beispiele sind per Download der Datei `javafx_samples-2_1_0.zip` als *JavaFX 2.1 Samples* für MS Windows und Mac OS X verfügbar und können mit der Entwicklungsumgebung verwendet werden. Als nützliche Informationsquelle erweist sich *FX Experience* (http://fxexperience.com/) mit technischen Erfahrungsberichten, Verfügbarkeit neuer UI-Controls und weitere Beispielen.

Open Source Projekt OpenJFX

Das Open Source Projekt *OpenJFX* (http://openjdk.java.net/projects/openjfx/) wurde im November 2011 durch die OpenJDK Community etabliert und ist eine wichtige Anlaufstelle für Entwickler die sich mit JavaFX beschäftigen. Das Ziel vom OpenJFX Projekt ist der Aufbau eines neuen Java Client Toolkits mit einer eigenständigen Java Spezifikation (JSR) die den JavaFX-Source-Code enthält. Die Beteiligung der Entwicklergemeinschaft ist von erheblicher Bedeutung für JavaFX um die Reife zu erhöhen und die Funktionalität mit weiteren UI-Controls anzureichern.