

Short Look at JavaFX

(Björn Müller, CaptainCasa, 20110816)

JavaFX 2.0 Beta is out since a couple of days - so I spent half a day of time for taking a short look at. This is the summary...

Installation

The Installation is very simple. I downloaded and installed the JavaFX runtime first. Then I installed NetBeans 7 and the corresponding JavaFX plugin. I did not use Eclipse (as I usually do) just because currently the official plugins are available for NetBeans only.

No problems occurred during installation.

First Project

In NetBeans I opened a new project (type "Java FX Application"). As result both the project and some first screen implementation is generated.

Programming the first Screen

The definition of screens is simple + follows the typical way of UI programming. You have to assemble some tree of components (nodes) and specify their properties and event reactors.

First I built a simple form:

```
package fx1;

public class FX1 extends Application
{
    public static void main(String[] args)
    {
        Application.launch(FX1.class, args);
    }

    @Override
    public void start(Stage primaryStage)
    {
        primaryStage.setTitle("Hello world");
        Group root = new Group();
        Scene scene = new Scene(root, 1024, 768, Color.WHITE);

        final VBox vbox = new VBox();
        vbox.setSpacing(10);
        vbox.setPadding(new Insets(10,10,10,10));

        {
            // HBox
            HBox hbox = new HBox();
            hbox.setStyle("-fx-background-color:#e0e0e0");
            hbox.setSpacing(10);
            hbox.setPadding(new Insets(10,10,10,10));
            hbox.setAlignment(Pos.CENTER);

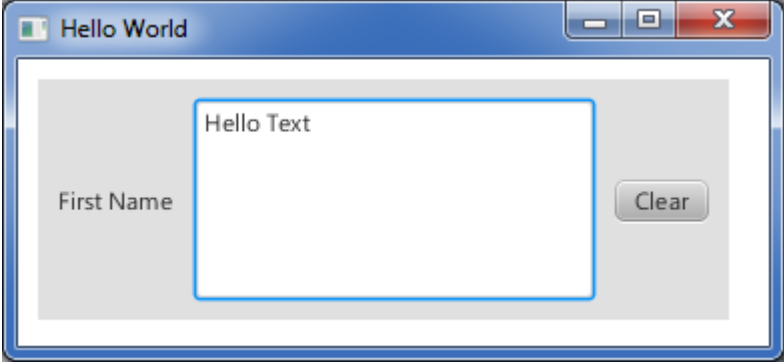
            // label
            Label label = new Label("First Name");
            hbox.getChildren().add(label);

            // field
            final TextBox field = new TextBox("Hello Text");
            field.setPrefwidth(200);
```

```
        field.setPrefHeight(100);
        hbox.getChildren().add(field);

        // button
        Button btn = new Button();
        btn.setText("Clear");
        btn.setOnAction(new EventHandler<ActionEvent>()
        {
            public void handle(ActionEvent event)
            {
                field.setText("");
            }
        });
        hbox.getChildren().add(btn);
    }
}
root.getChildren().add(vbox);
primaryStage.setScene(scene);
primaryStage.setVisible(true);
}
}
```

So, all this is quite simple and the result quite boring ;-)



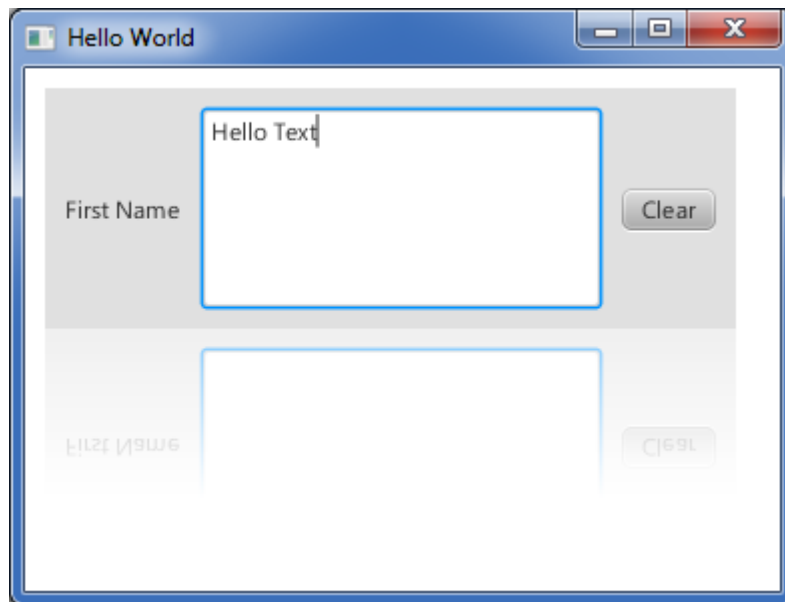
But: it's already nicer than some first Java Swing screens... and you may already have seen inside the code: via "setStyle(...)" you can apply CSS style parameters, so the styling issue is much nicer apply-able than with Swing components.

First Wow - Some Reflection Effect

Next step was to add a reflection effect. The code for doing so is quite simple:

```
// reflection
Reflection reflection = new Reflection();
reflection.setFraction(0.7);
hbox.setEffect(reflection);
```

The result is:



The reflection is updated automatically: when writing some text into the field then the text is mirrored as result.

Second Wow - Some Animation

Then I added some animation. There is a path animation (object is moved) and a rotation animation:

```
private void startPathAnimation(Node node)
{
    Path path = new Path();
    Bounds nodeBounds = node.getLayoutBounds();
    double startX = nodeBounds.getMinX() + nodeBounds.getWidth() / 2;
    double startY = nodeBounds.getMinY() + nodeBounds.getHeight() / 2;
    path.getElements().add(new MoveTo(startX, startY));
    path.getElements().add(new CubicCurveTo(380, 0, 380, 120, 200, 120));
    path.getElements().add(new CubicCurveTo(0, 120, 0, 240, 380, 240));
    path.getElements().add(new CubicCurveTo(0, 0, 0, 0, startX, startY));
    PathTransition pathTransition = new PathTransition();
    pathTransition.setDuration(Duration.valueOf(4000));
    pathTransition.setPath(path);
    pathTransition.setNode(node);
    //
    pathTransition.setOrientation(PathTransition.OrientationType.ORTHOGONAL_TO_TANGEN
T);
    //
    pathTransition.setCycleCount(Timeline.INDEFINITE);
    pathTransition.setCycleCount(0);
    pathTransition.setAutoReverse(false);
    pathTransition.play();
}

private void startRotateAnimation(Node node)
{
    RotateTransition rotateTransition = new
RotateTransition(Duration.valueOf(1000), node);
    rotateTransition.setByAngle(360f);
    rotateTransition.setCycleCount(0);
    rotateTransition.setAutoReverse(false);
    rotateTransition.play();
}

private void startTwoAnimations(Node node)
{
    startPathAnimation(node);
    startRotateAnimation(node);
}
}
```

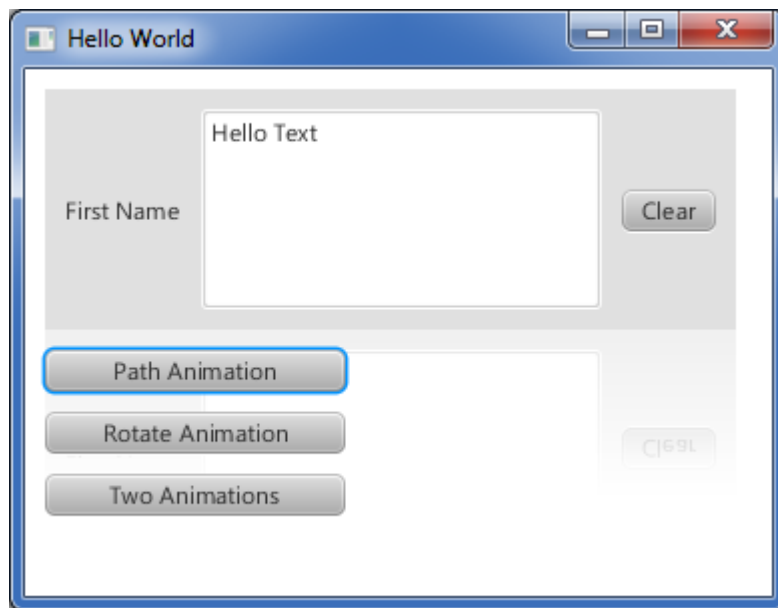
The nice point: animations are referenced to a certain node - a node being one simple component, or - as in our example - being a complex component, containing other

components.

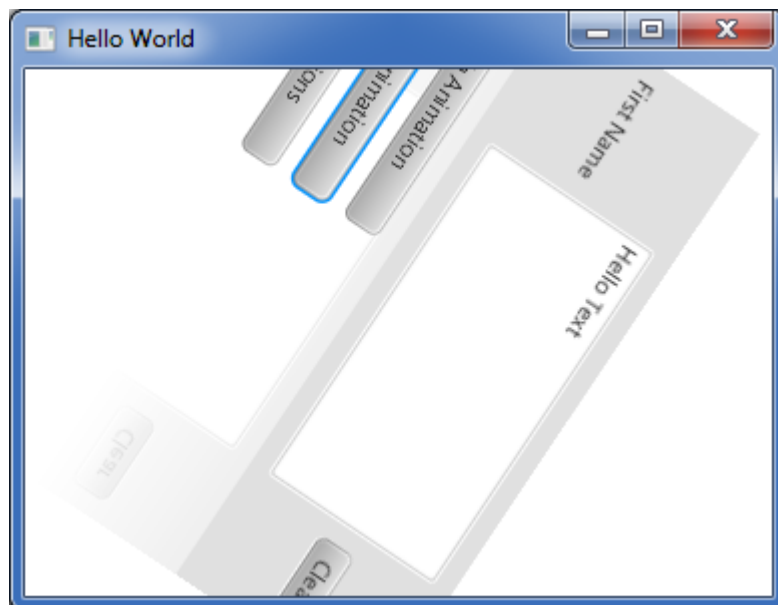
I added three buttons in order to start the animation. The code for one button is:

```
// button
Button btn = new Button();
btn.setText("Rotate Animation");
btn.setPrefWidth(150);
btn.setOnAction(new EventHandler<ActionEvent>()
{
    public void handle(ActionEvent event)
    {
        startRotateAnimation(vbox);
    }
});
vbox.getChildren().add(btn);
```

The result is:



During the animation...



...the whole node is animated automatically. The components (e.g. the field component) stays active during the animation - i.e. the user can still key in some text.

The animation speed is excellent on my machine.

Deploying into the Web

The JavaFX application can be started from the browser as applet or as webstart application. Corresponding HTML and JNLP files are generated by NetBeans automatically.

I posted the mini application from above onto our demo server, please follow the link:

<http://www.captaincasademo.com/fx1/FX1.html>

(You need to install the JavaFX runtime on your client first!)

Integration Swing - JavaFX

Then I tested the JavaFX integration into Swing. Basicall it is possible to run one or several JavaFX applications within a special Swing container (JFXPanel):

```
package fx1;

import com.sun.deploy.uikit.impl.fx.FXPluginToolkit;
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.event.ActionListener;
import javafx.application.Application;
import javafx.application.Platform;
import javafx.embed.swing.JFXPanel;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextBox;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.text.Font;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JTextField;

public class SwingIntegrationApp extends JFrame
{
    JTextField m_tfSWING;
    JButton m_bSWING;
    TextBox m_tbFX;
    Button m_bFX;

    public SwingIntegrationApp()
    {
        initComponents();
    }

    public static void main(String args[])
    {
        EventQueue.invokeLater(new Runnable()
        {
            public void run()
            {
                new SwingIntegrationApp().setVisible(true);
            }
        });
    }

    private void initComponents()
    {
        getContentPane().setBackground(Color.WHITE);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400,300);
        setLayout(null);
        {
            m_tfSWING = new JTextField("This is a Swing text field.");
            m_tfSWING.setBounds(10,250,200,25);
            getContentPane().add(m_tfSWING);
            m_bSWING = new JButton("Transfer Text");
            m_bSWING.setBounds(220,250,100,25);
```

```

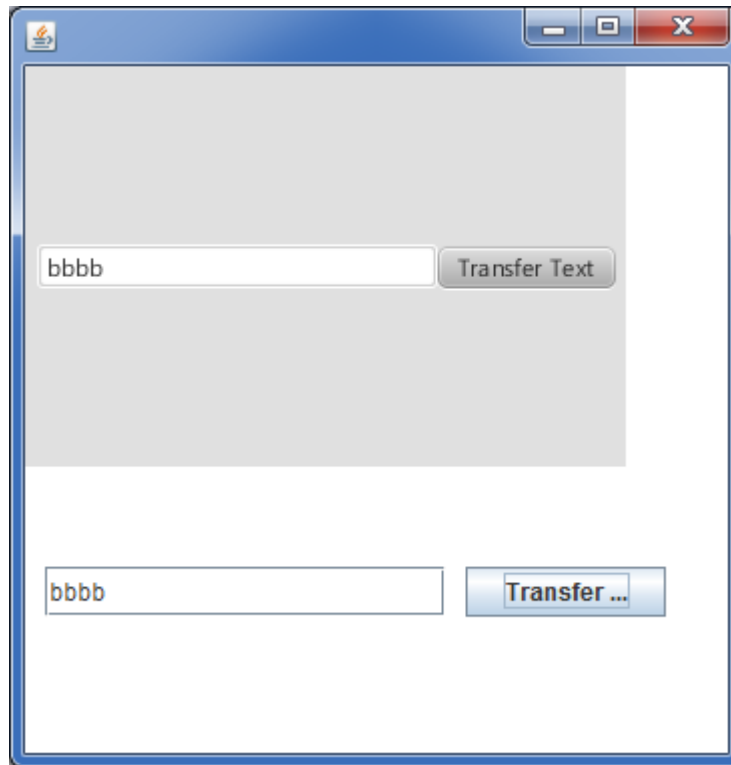
        m_bSWING.addActionListener(new ActionListener()
        {
            public void actionPerformed(java.awt.event.ActionEvent e)
            {
                Platform.runLater(new Runnable()
                {
                    public void run()
                    {
                        m_tbFX.setText(m_tfSWING.getText());
                    }
                });
            }
        });
    getContentPane().add(m_bSWING);
}
}

final JFXPanel fx = new JFXPanel();
fx.setBounds(0,0,300,200);
getContentPane().add(fx);
Platform.runLater(new Runnable()
{
    public void run()
    {
        Scene scene = createScene();
        fx.setScene(scene);
    }
});
}
}

private Scene createScene()
{
    m_tbFX = new TextBox("This is a JavaFX Textbox");
    m_tbFX.setPrefwidth(200);
    m_bFX = new Button("Transfer Text");
    m_bFX.setOnAction(new EventHandler<ActionEvent>()
    {
        public void handle(ActionEvent event)
        {
            EventQueue.invokeLater(new Runnable()
            {
                public void run()
                {
                    m_tfSWING.setText(m_tbFX.getText());
                }
            });
        }
    });
    HBox hbox = new HBox();
    hbox.setAlignment(Pos.CENTER);
    hbox.getChildren().add(m_tbFX);
    hbox.getChildren().add(m_bFX);
    hbox.setStyle("-fx-background-color:#e0e0e0");
    Scene scene = new Scene(hbox);
    return scene;
}
}
}

```

The result is:



You see: the integration is quite simple. You have to live with different threads in the client when interacting between Swing components and JavaFX components (and vice versa). This means you need to package method calls into Runnable-objects and need to execute these objects in the thread of the corresponding environment.

Tabbing-Issue

The focus management seems not yet to be synchronized between both environments: it is not yet possible (at least not by default) to tab through the components with the keyboard...

Browser Integration

There is a native Web-component in JavaFX (embedded browser) - so you need not to go through complex scenarios anymore to integrate HTML pages (as with Swing). The component internally uses the WebKit engine (open source web browser engine which is also used in the Safari browser).

Summary

I believe, from just technical point of view, Java FX 2.0 is the right release to dive in: I never liked the special scripting language and am happy that all is back to pure Java. And I never liked the non-integration between the Swing and the FX line - which is now solved in a much better way (again: all in Java).

There are new media components (as the embedded browser) that also are missing in Swing, and which just are required by today's application. (Of course there were workarounds in Swing, but this was always painful.)

Compared to Swing, there are (currently) no significant new basic components like field,

checkbox, etc. yet. But: what you can do with the components is a level higher than what you are able to do with Swing components:

- Apply styling in a much easier CSS-driven way
- Transform, rotate them, with still keeping them functionally working (Swing: transformations only on images)
- Animate them in a very easy way (Swing-transformations always quite painful, esp. from performance point of view)

There were no problems with stability during this short test. From installation over first examples to running the result inside the browser - everything worked without problems.

Appendix

This is the full code of the mini demo application:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package fx1;

import javafx.animation.PathTransition;
import javafx.animation.RotateTransition;
import javafx.animation.Timeline;
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Bounds;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Group;
import javafx.scene.Node;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.Textbox;
import javafx.scene.effect.Reflection;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;
import javafx.scene.shape.CubicCurveTo;
import javafx.scene.shape.MoveTo;
import javafx.scene.shape.Path;
import javafx.scene.transform.Transform;
import javafx.scene.transform.Translate;
import javafx.scene.web.WebEngine;
import javafx.scene.web.WebView;
import javafx.stage.Stage;
import javafx.util.Duration;

/**
 *
 * @author bmu
 */
public class FX1 extends Application
{
    public static void main(String[] args)
    {
        Application.launch(FX1.class, args);
    }

    @Override
    public void start(Stage primaryStage)
    {
        primaryStage.setTitle("Hello world");
        Group root = new Group();
        Scene scene = new Scene(root, 1024, 768, Color.WHITE);

        final VBox vbox = new VBox();
        vbox.setSpacing(10);
        vbox.setPadding(new Insets(10,10,10,10));

        {
            {
                // HBox
                HBox hbox = new HBox();
                hbox.setStyle("-fx-background-color:#e0e0e0");
                hbox.setSpacing(10);
                hbox.setPadding(new Insets(10,10,10,10));
                hbox.setAlignment(Pos.CENTER);

                // label
                Label label = new Label("First Name");
                hbox.getChildren().add(label);

                // field
            }
        }
    }
}
```

```

final TextBox field = new TextBox("Hello Text");
field.setPrefWidth(200);
field.setPrefHeight(100);
hbox.getChildren().add(field);

// button
Button btn = new Button();
btn.setText("Clear");
btn.setOnAction(new EventHandler<ActionEvent>()
{
    public void handle(ActionEvent event)
    {
        field.setText("");
    }
});
hbox.getChildren().add(btn);

// reflection
Reflection reflection = new Reflection();
reflection.setFraction(0.7);
hbox.setEffect(reflection);

vbox.getChildren().add(hbox);
}
{
    // button
    Button btn = new Button();
    btn.setText("Path Animation");
    btn.setPrefWidth(150);
    btn.setOnAction(new EventHandler<ActionEvent>()
    {
        public void handle(ActionEvent event)
        {
            startPathAnimation(vbox);
        }
    });
    vbox.getChildren().add(btn);
}
{
    // button
    Button btn = new Button();
    btn.setText("Rotate Animation");
    btn.setPrefWidth(150);
    btn.setOnAction(new EventHandler<ActionEvent>()
    {
        public void handle(ActionEvent event)
        {
            startRotateAnimation(vbox);
        }
    });
    vbox.getChildren().add(btn);
}
{
    // button
    Button btn = new Button();
    btn.setText("Two Animations");
    btn.setPrefWidth(150);
    btn.setOnAction(new EventHandler<ActionEvent>()
    {
        public void handle(ActionEvent event)
        {
            startTwoAnimations(vbox);
        }
    });
    vbox.getChildren().add(btn);
}
}

// transformations
// vbox.getTransforms().add(Transform.shear(-0.1,0));

root.getChildren().add(vbox);

primaryStage.setScene(scene);
primaryStage.setVisible(true);

```

```

}

private void startPathAnimation(Node node)
{
    Path path = new Path();
    Bounds nodeBounds = node.getLayoutBounds();
    double startX = nodeBounds.getMinX() + nodeBounds.getWidth() / 2;
    double startY = nodeBounds.getMinY() + nodeBounds.getHeight() / 2;
    path.getElements().add(new MoveTo(startX, startY));
    path.getElements().add(new CubicCurveTo(380, 0, 380, 120, 200, 120));
    path.getElements().add(new CubicCurveTo(0, 120, 0, 240, 380, 240));
    path.getElements().add(new CubicCurveTo(0, 0, 0, 0, startX, startY));
    PathTransition pathTransition = new PathTransition();
    pathTransition.setDuration(Duration.valueOf(4000));
    pathTransition.setPath(path);
    pathTransition.setNode(node);
    //
    pathTransition.setOrientation(PathTransition.OrientationType.ORTHOGONAL_TO_TANGEN
T);
    //
    pathTransition.setCycleCount(Timeline.INDEFINITE);
    pathTransition.setCycleCount(0);
    pathTransition.setAutoReverse(false);
    pathTransition.play();
}

private void startRotateAnimation(Node node)
{
    RotateTransition rotateTransition = new
RotateTransition(Duration.valueOf(1000), node);
    rotateTransition.setByAngle(360f);
    rotateTransition.setCycleCount(0);
    rotateTransition.setAutoReverse(false);
    rotateTransition.play();
}

private void startTwoAnimations(Node node)
{
    startPathAnimation(node);
    startRotateAnimation(node);
}
}

```