JavaFX: New Kid on RIA Block

Sang Shin
Sun Microsystems
javapassion.com
Topics

- Things you can build with JavaFX
- JavaFX script overview
- Declarative GUI building
- Scene graph
- Animation
- Media
- Deployment
- JavaFX Mobile
- Web services
Things You Can Build with JavaFX
Demo:

JavaFX Sample Apps from javafx.com
JavaFX Script
Overview
Features of JavaFX Script

• Declarative, statically-typed scripting language
• Facilitates rapid GUI development
• Runs on Virtual Machine for the Java™ platform
• Deployment options same as Java programs
• Fully utilizes Java class libraries behind the scenes
• Cool, interesting language features for building RIA apps
  > Object literal, Sequence, Data binding, Animation, Media, etc.
Class Definition

- Address class definition: The Address class declares street, city, state, and zip fields all of type String

```java
class Address {
    var street: String;
    var city: String;
    var state: String;
    var zip: String;
}
```
Object Literal – Creating object

• In the JavaFX Script programming language, an object instance can be created with an object literal (like in JavaScript, unlike in Java)
• Example: The first word (Address) specifies the type of object, class, that you are creating.

    Address {
        street: "1 Main Street"; // separated by semi colons
        city: "Santa Clara";
        state: "CA";
        zip: "95050";
    }
Nesting an Object inside Another Object

- Nesting `Address` object inside `Customer` object

```java
def customer = Customer {
  firstName: "John";
  lastName: "Doe";
  phoneNum: "(408) 555-1212";
  address: Address {
    street: "1 Main Street";
    city: "Santa Clara";
    state: "CA";
    zip: "95050";
  }
}
```
 Binding

• Cause and effect – responding to changes
• *bind* operator allows dynamic content to be expressed declaratively
• Dependency based evaluation of any expression
• Automated by the JavaFX runtime rather than manually wired by the programmer
• Eliminates the listener pattern
Binding to a Simple Expression

```javascript
var x = 0;

// Bind variable x to variable y. Whenever the value of x changes,
// the value of variable y automatically changes as well.
var y = bind x + 10;

x = 1;
println("----y after x is changed to 1 = {y}"); // y now equals 11

x = 47;
println("----y after x is changed to 47 = {y}"); // y now equals 57
```
Using Declarative Syntax (for Creating GUI)
Example of JavaFX Application

```java
import javafx.scene.paint.Color;
import javafx.scene.Scene;
import javafx.scene.shape.Circle;
import javafx.stage.Stage;

Stage {
    title: "My circle"
    width: 100
    height: 100
    scene: Scene {
        content: [
            Circle {
                centerX: 50,
                centerY: 50
                radius: 40
                fill: Color.RED
            }
        ]
    }
}
```
Why Declarative Syntax for Building GUI?

• Because the structure of declared objects in the code reflects the visual structure of the scene graph, and this enables you to understand and maintain the code easily.

• The order of elements you declare in the code matches the order in which they appear in the application.
Demo:

Building “HelloWorld” JavaFX Application

http://www.javapassion.com/handsonlabs/javafx_lang/#Exercise_1
Scene Graph
What is Scene Graph?

- Scene Graph enables declarative GUI programming
- The scene graph is a tree-like data structure which defines a hierarchy of graphical objects in a scene.
- A single element in the scene graph is called a node
- You can apply the following GUI properties to any node in Scene Graph tree
  - Effect
  - Animation
  - Transformation
  - User input
Scene Graph: Group

Group {
  transforms: Translate {
    x: 15, y: 15
  }
  content: [
    Text {
      x: 10, y: 50
      font: Font: {
        size: 50
      }
      content: "Hello World"
    }
    Circle {
      centerX: 100, centerY: 100
      radius: 40
      fill: Color.BLACK
    }
  ]
}
Effects
How Effect Works

• Any Effect instance can be applied to a scene graph Node by setting the `Node.effect` variable.

• All Effect classes extend the abstract `javafx.scene.effect.Effect` base class.

• Each Effect subclass exposes a small number of variables that control the visual appearance of the Node.
Effects:
DropShadow
Example: DropShadow class

- *DropShadow* class provides 5 variables
  - color: The shadow Color
    - default: Color.BLACK
  - offsetX: The shadow offset in the x direction, in pixels.
    - default: 0.0
  - offsetY: The shadow offset in the y direction, in pixels.
    - default: 0.0
  - radius: The radius of the shadow blur kernel.
    - default: 10.0, max: 63.0
  - spread: The spread of the shadow.
    - default: 0.0, max: 1.0, min: 0.0
Example: DropShadow

Text {
    effect: DropShadow {
        offsetY: 3
        color: Color.color(0.4, 0.4, 0.4)
    };
    ...
},
Circle {
    effect: DropShadow {
        offsetY: 4
    },
    ...
}
Example: DropShadow

Text {
  effect: DropShadow {
    offsetY: 3
    color: Color.GREEN
    radius: 20.0
  }
};
...
},
Circle {
  effect: DropShadow {
    offsetX: 10
    offsetY: 20
    color: Color.BLUE
    radius: 30.0
  }
}
...
Example: DropShadow with Binding

• Apply a DropShadow effect to a rounded Rectangle and control its appearance through the magic of the bind operator.

```javascript
Rectangle {
    effect: DropShadow {
        radius: bind radius
    }
    x: 50 y: 30 width: 150 height: 100
    arcWidth: 40 arcHeight: 40
    fill: Color.RED
}
```
Demo:

- DropShadow,
  javapassion.com/handsonlabs/javafx_guibasics/index.html#7.1
- DropShadow with Binding,
  javapassion.com/handsonlabs/javafx_customnode
- EffectsPlayground
  http://javafx.com/samples/EffectsPlayground/index.html
JavaFX
GUI Basics II: Interaction, Transformation, Binding, Drag and drop, Swing components
Topics

• Interaction
• Transformation
• Binding
• Drag and drop
• Swing components
Interactions
Handling Events

• All nodes have either mouse or keyboard events
  > Override the appropriate method
• Mouse events – `onMouseXXXX()`
  > XXXX = Entered, Exited, Pressed, Dragged, Moved, Clicked, Released, WheelMoved
• Keyboard events – `onKeyboardXXXX()`
  > XXXX = Pressed, Released, Typed
• Indicate interactivity by changing cursor
  > Set the cursor attribute
Example: Handling Events

• Mouse events change the color of an rectangle

```javascript
var rectangle:Rectangle = Rectangle {
    x: 20, y: 10
    width: 150, height: 70
    arcWidth: 50, arcHeight: 50
    fill: Color.LIGHTBLUE
    stroke: Color.ROYALBLUE
    strokeWidth: 3
    onMouseEntered: function( e: MouseEvent ):Void {
        rectangle.fill = Color.WHITESMOKE;
    }
    onMouseExited: function( e: MouseEvent ):Void {
        rectangle.fill = Color.LIGHTBLUE;
    }
}
```
Demo: Simple Sketch

gui2_interaction_Sketch_basic_3steps
http://www.javapassion.com/handsonlabs/javafx_guibasics2/#1.3
Transformation
Transformation class

• Provides functions to perform
  > Rotating
  > Scaling
  > Shearing
  > Translation
Rotate Transformation

- Rotates coordinates around an anchor point

```java
Stage {
    title: "Welcome to JavaFX!"
    scene: Scene {
        width: 200
        height: 200
        content: [
            Rectangle {
                transforms: Rotate {
                    angle: bind angle
                    pivotX: 10
                    pivotY: 10
                }
            }
        ]
    }
}
```
Translation Transformation

• Translates coordinates by the specified factors

Stage {
  title: "Transformation - Translate"
  width: 100  height: 100
  scene: Scene {
    content: [
      Rectangle {
        transforms: Translate {
          x: bind translateX
          y: bind translateY
        }
        x: 10
        y: 10
        width: 20
        height: 20
        fill: Color.GREEN
        onMouseEntered: function( e: MouseEvent ):Void {
          translateY = 20;
          translateY = 30
        }
      }
    }
  }
}
Demo: Transformation

www.javapassion.com/handsonlabs/javafx_guibasics2/#Exercise_2
Bindings with GUI Objects
Binding with GUI objects

- The power of binding really shows when it is used with GUI objects
  - GUI objects can automatically change their shape, characteristics, and behavior
Demo: Transformation

gui2_binding_DropShadow_Mouse
www.javapassion.com/handsonlabs/javafx_guibasics2/#Exercise_3
Animation Support in JavaFX
Animation Support in JavaFX

• Built in the language syntax
  > Can animate any variable

• Native support for time
  > *Duration* class
  > Time literals – *1ms, 1s, 1m, 1h*
  > Eg. *var runFor = 500ms*
Two Types of Animation in JavaFX

• Transition
  > “Precanned” animation
  > Single purpose

• Animation
  > More flexible but more code
Transitions
Transitions

• Predefined animations to perform a specific task
  > Position, rotation, opacity, etc.

• Out of the box transitions
  > RotateTransition – rotation
  > FadeTransition – opacity
  > TranslateTransition – move a node along a straight line
  > PathTransition – move an object along a defined path
  > ScaleTransition – grows or shrinks a node
Using Transitions

• Need to specify which node the transition is performed on
  > Nodes – geometric shapes, images, text, Swing components

• Other attributes
  > Duration – how long to perform the animation
  > Rate – the speed and direction
  > Interpolator – the acceleration and deceleration of the animation

• Can execute a function at the end of the animation
  > Assign a function to action attribute
RotationTransition

```javascript
var rotTransition = RotateTransition {
    duration: 3s
    node: node
    byAngle: 180
    repeatCount: 4
    autoReverse: true
}

var princess: ImageView = ImageView {
    image: Image {
        url: "{__DIR__}princess.png"
    }
    onMouseClicked: function( e: MouseEvent ): Void {
        rotTransition.play();
    }
}
```
Path Transition

```javascript
var earth: ImageView = ImageView {
    x: sx y: sy
    image: Image { url: "{__DIR__}earth.png" }
}
def path = [
    MoveTo { x: sx y: sy}
    ArcTo { x: 0 y: 200
        radiusX: 50 radiusY: 50 sweepFlag: true
    }
];
var aniPath: PathTransition = PathTransition {
    node: earth
    path: AnimationPath.createFromPath(
        Path {elements: path })
    duration: 1500ms
}

aniPath.playFromStart();
```
Demo:

Transitions

http://www.javapassion.com/handsonlabs/javafx_animation/#Exercise_1
http://www.javapassion.com/handsonlabs/javafx_animation/#Exercise_2
KeyFrame based Animation
Key Frame based Animation

• What is Key Frame based animation?
  > A declarative model in which programmer describes the animated state transitions of each "scene" by declaring "snapshots" (key frames) of state at certain points in time

• Two basic varieties of key frame animation
  > Discrete - Set of discrete key frames
  > Interpolated - Special interpolation functions calculate the states that occur between animation frames

• Animation controls
  > Start, stop, pause, and resume
Programming Model of Key Frame Animation

- Animations occur along a timeline, represented by a `javafx.animation.Timeline` object.
- Each timeline contains two or more key frames, represented by `javafx.animation.KeyFrame` objects.
- Each timeline supports:
  - Animation attributes
    - `autoReverse`, `repeatCount`, `toggle`, etc.
  - Playback controls
    - `start()`, `stop()`, `pause()`, and `resume()`
Example: Interpolatorator Based

// The value of tx changes from 0 to 700
// in 10 second period in linear fashion
var t = Timeline {
  keyFrames : [
    KeyFrame {
      time: 0s
      values: [ tx => 0 ]
      action: function() { ... }
    },
    KeyFrame {
      time: 10s
      values: [
        tx => 700 tween Interpolatorator.LINEAR
      ]
    }
  ]
}
t.start();
Example – Defining Key Frames

Timeline {
    keyFrames: [
        KeyFrame {
            time: 0s
            values: [ radius => 30 ]
        }
        KeyFrame {
            time: 5s
            values: [
                radius => 300 tween Interpolator.LINEAR
            ]
        }
    ]
}

Key value
radius = 30
radius = 300

How the value changes over time

0s 1s 2s 3s 4s 5s 6s
Keyframes
**at() (Shorthand notation)**

```javascript
var t = Timeline {
    ...
    keyFrames: [
        KeyFrame {
            time: 0ms
            values: [ radius => 30 ]
        },
        KeyFrame {
            time: 500ms
            values: [
                radius => 300 tween Interpolator.LINEAR
            ]
        }
    ]
};
```

```javascript
keyFrames: [
    at(0ms) { radius => 30 }
    at(500ms) {
        radius => 300 Interpolator.LINEAR
    }
]
```
Animation through Binding

```javascript
var opa = 0.0;
var street1:ImageView = ImageView {
    image: Image { url: "{__DIR__}street1.jpg" } 
    opacity: bind opa
    onMouseClicked: function( e: MouseEvent ):Void {
        timeline.play();
    }
}

var timeline:Timeline = Timeline {
    keyFrames: [
        KeyFrame {
            time: 0s
            values: [ opa => 0.0,]
        },
        KeyFrame {
            time: 1s
            values: [ opa => 1.0 tween Interpolator.LINEAR,]
        }
    ]
}
```
Custom Node
Custom Node

• Primary means of Scene Graph encapsulation
• Use it when extending existing GUI class is not enough for your task
• Simply override the `create()` method, which returns a Node object
Simple CustomNode

- Extend CustomNode class and override create()

class Bars extends CustomNode {
  override function create():Node {
    return Group {
      content: for(x in [0..4]) {
        Rectangle {
          y: indexof x * 20
          width: 100
          height: 10
          fill: Color.RED
        }
      }
    }
  }
}

// Bars object literal
Bars { }
Demo: Building “Under the Sea” Step by Step

http://www.javapassion.com/handsonlabs/javafx_animation/index.html#4.1
Media
FX Media API Overview

- Simple model-view-controller design
- All classes in `javafx.scene.media` package.
- `MediaView` takes all cool features of SceneGraph node, such as effects, transforming, clip, opacity, etc...
Example of Creating a Media Player

```javascript
var video:Media = Media {
    source: "http://..."
};

var player:MediaPlayer = MediaPlayer {
    media: video
    rate: 1.0
    volume: 0.4
};

var view:MediaView = MediaView {
    mediaPlayer: player
    x: 200
    y: 200
};

Stage {
    title: "Media Player"
    width: 700
    height: 700
    scene: Scene {
        content: [view]
    }
}
```
Demo:
Media

http://javafx.com/samples/SimpleVideoPlayer/index.html
Deployment
Deployment Options

• JavaFX 1.0 applications can be deployed using the two standard Java deployment technologies
  > Java Plugin: A tool used for deploying Java applets that run inside a web browser
  > Java Web Start: A tool used for deploying stand-alone Java applications on the desktop, using JNLP (Java Network Launching Protocol).

• Or using mobile emulation
  > JavaFX Mobile Emulator: A tool provided with the JavaFX SDK, which displays your applications as they would look on a typical mobile device.
Execution Models

- Standard, Web Start, Applet, Mobile emulator
Demo:

Draggable Applet
JavaFX Mobile
Demo: JavaFX Mobile

http://javafx.com/samples/SimpleVideoPlayerMobile/index.html
What is Java Store?

- Online store for Java and JavaFX applications
  > Distribution channel delivering Java and JavaFX applications to 1Billion Java enabled desktop and devices
- Provide an easy means for developers and ISVs to monetize their Java and JavaFX applications
Java Store is Written in JavaFX

Solar System
This application shows the solar system with planet orbiting around the Sun. The animations and transformations available in the JavaFX platform make it easy to build graphical simulations. This example is an orrery, a simulation of the pl...
Java Store End-to-End Architecture

- Developer Registration
- Publishing
- Testing
- Content Management
- Payment Settlement

WAREHOUSE

Java Store Desktop
Mobile Store SP/OEM Branded
TV Store

STORES

MARKETPLACE

END USER

DEVELOPERS

WEB DEVELOPERS
How People Get Java Store?

> store.java.com
> Java auto-update
> JRE/JDK downloads
> Java.sun.com
Demo: Java Store

Solar System
This application shows the solar system with planet orbiting around the Sun. The animations and transformations available in the JavaFX platform make it easy to build graphical simulations. This example is an orrery, a simulation of the pl...

Developers: want to submit an app?
Demo: WidgetFx
JavaFX Technology Overview