Rich Client Applications with JavaFX SDK

Lee Chuk Munn
Staff Engineer
Overview of the JavaFX SDK
Overview
JavaFX Script Programming Language

• Declarative, statically-typed scripting language
• Facilitates rapid GUI development
• Many cool, interesting language features
• Runs on Virtual Machine for the Java™ platform
• Deployment options same as Java programs
• Fully utilizes Java class libraries behind the scenes
• For content designers and Media engineers
Example of JavaFX Application

```java
import javafx.application.*;
import javafx.scene.shape.*;
import javafx.scene.paint.*;

Stage {
    scene: Scene {
        content: [
            Circle {
                centerX: 50
                centerY: 50
                radius: 50
                fill: Color.RED
            }
        ]
    }
}
```
(Some) Language Features
Basic Data Types

• Garden variety type
  > String
  > Number/Integer – byte, short, int, long, BigInteger
  > Boolean
  > Void

• Durations – 1ms, 2s, 4m, 8h

• Sequences – more later

• Functions

  var doExit = function():Void {
    System.exit(0);
  };

Sequences

- **Sequences**

  ```javascript
  var time:Duration[] = [60ms, 60s, 60m, 60h];
  var days = [1..31]; //Fill days with the values
  ```

- **Insert, delete and reverse**

  ```javascript
  insert 5s into time;
  delete 31 from days;
  var revDays = reverse days;
  ```

- **Slice via range and filters**

  ```javascript
  var oddDays = days[n|(n mod 2) == 1];
  var firstThree = time[0..<3]; // [60ms, 60s, 60m]
  ```
Data Binding

• Binding associates the RHS to a variable or field
  > Changes to the RHS will cause the value to be reapplied to the field

```javascript
var r = 10;
var area = bind r * r * Math.PI;
r = 5;
area == 78.5714 //true
```
Graphical Objects

*The fun stuff
Base Graphical Objects

• Graphical objects
  > Text, geometric shapes, images, Swing components
  > Subclass of Node

• Some common attributes in nodes
  > Transformation – translate, shear, rotate, scale
  > Clip – displaying only part of the node based on a geometric shape
  > Effect – type of effect, eg. blurring, shadowing, to apply
  > Events – mouse, keyboard
  > Opacity – setting the translucency
  > List is not exhaustive
Scene Graph

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
    }
  ]
}
Example – Text

- Displaying text

```javascript
Text {
  effect: DropShadow {
    offsetX: -10
    offsetY: -10
  }
  font: Font {
    name: "DirtyBakersDozen"
    size: 50
  }
  fill: Color.ROYALBLUE
  stroke: Color.BLUE, strokeWidth: 3
  x: 15, y: 80
  content: "Hello World"
}
```
Geometric Shapes

• Arc, ellipse, line, polygon, circle, rectangle

• Very similar to text

Circle {
  centerX: 70, centerY: 70
  radius: 50
  fill: LinearGradient {
    startX: 0.0, startY: 0.0
    endX: 1.0, endY: 1.0
    stops: [
      Stop { color: Color.RED, offset: 0.2 }
      Stop { color: Color.BLUE, offset: 0.6 }
      Stop { color: Color.GREEN, offset: 0.8 }
    ]
  }
  stroke: Color.BLACK
  strokeWidth: 3
  strokeDashArray: [ 7 ] strokeDashOffset: 2
Example – ShapeIntersect, ShapeSubtract

ShapeIntersect {
  a: [rect]
  b: [diamond]
}

ShapeSubtract {
  a: [rect]
  b: [diamond]
}

ShapeSubtract {
  a: [rect
  b: [diamond ]
}
Example – Path

Path {
    fill: Color.LIGHTGRAY
    stroke: Color.GRAY
    strokeWidth: 3
    elements: [
        MoveTo { x: 15 y: 15 },
        ArcTo { x: 50 y: 10 radiusX: 20
            radiusY: 20 sweepFlag: true},
        ArcTo { x: 70 y: 20 radiusX: 20
            radiusY: 20 sweepFlag: true},
        ArcTo { x: 50 y: 60 radiusX: 20
            radiusY: 20 sweepFlag: true},
        ArcTo { x: 20 y: 50 radiusX: 10
            radiusY: 5 sweepFlag: false},
        ArcTo { x: 15 y: 15 radiusX: 10
            radiusY: 10 sweepFlag: true},
    ]
}
Images

```
ImageView {
  clip: Rectangle {
    y: 30 x: 50
    width: 350 height: 100
  }
  image: Image { url: "file:///..."}
}
```
Some Effects Supported In JavaFX

effect: SepiaTone { level: 0.5 }

effect: Glow { level: 0.7 }

effect: GaussianBlur {
  input: SepiaTone {
    level: 0.5
  }
  radius: 10.0
}

effect: Reflection {
  fraction: 0.7
}
Lighting Effect

effect: Lighting{
    surfaceScale: 7
    light: DistantLight{
        azimuth: 90
        elevation: 30
    }
}

effect: Lighting{
    surfaceScale: 7
    light: SpotLight {
        x: 0 y: 0 z: 50
        pointsAtX: 10
        pointsAtY: 10
        pointsAtZ: 0
    }
}

Interactions
Example – Handling Events

Changing the color of the 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;
    }
}
```
Data Format Parser

- Includes a 'pull' parser that supports JSON and XML
- To use in 'event' mode
  > You will be notified of when the parser gets a token
  > Specify the format and input stream
  > Handle input as the parser returns tokens
- Can be use in 'linear' mode as well
  > Direct the parser with `forward()` and `seek()`
Example – PullParser in 'Event' Mode

```javascript
var jsonParser: PullParser = PullParser {
    documentType: PullParser.JSON
    onEvent: function(event: Event) {
        if (event != null) {
            if (event.type == PullParser.START_VALUE)
                System.out.println("start: name = {event.name}"")
            else if (event.type == PullParser.END_VALUE)
                System.out.println("\tend: name = {event.name}" )
        }
    }
}

jsonParser.input = //some input
jsonParser.parse();
```
Accessing REST Resources

- Includes an asynchronous HTTP request class
- Need to specify the location and the HTTP method
- Provides access lifecycle events
  - started, connecting, writing, reading, done, ...
  - onStarted, onConnection, onWriting, onReading, onDone, ...
- Invoke `enqueue()` to start request
Example – Making HTTP Request

```typescript
var getTwitter: HttpRequest = HttpRequest {
    method: HttpRequest.GET
    location: "http://twitter.com/statuses/public_timeline.json"
    onInput: function(is: InputStream) {
        try {
            jsonParser.input = is;
            jsonParser.parse();
        } finally {
            is.close();
        }
    }
}

getTwitter.enqueue();
```
Animations
Animation Support

• Timing key in animation
• Native support for time
  > Duration class
  > Time literals – 1ms, 1s, 1m, 1h
  > Eg. `var runFor = 500ms`
• Two types of animation support
  > Transition – canned animation
  > Keyframe based – more flexible but more code
Key Animation Concepts Illustrated

- Vary an **attribute** over **time**
- Specify **how** the value changes over **time**
- Bind these attribute to objects

How the value changes over time

Key value
radius = 30  radius = 300

Keyframes
0s  1s  2s  3s  4s  5s  6s
Example – Animating an Object

```actionscript
var _radius:Number = 30
var animate:Timeline = Timeline {
    rate: -1 //Default is 1
    keyFrames: [
        at(0s) { _radius => 30 }
        at(5s) { _radius => 300 tween Interpolator.LINEAR }
    ]
}
var circle:Circle = Circle {
    radius: bind _radius
    ...
    function onMouseClicked(e: MouseEvent): Void {
        animate.rate = if (animate.rate == -1) 1 else -1;
        animate.play();
    }
}
```

Changing the value of a bound object
Media
Formats, Codecs and Platform Support

• Cross platform video format support
  > Encode once, play anywhere
  > License codec from On2 Technologies

• Leveraging the native platform
  > Windows
    • Play windows media via DirectPlay
    • Flash via the ActiveX control
  > Mac – CoreAudio and CoreVideo
  > Solaris and Linux
    • Any audio/video supported by totem can be played
    • May have to recompile gstreamer on some Linux platform
Architectural Overview

• MVC architecture

• Main classes
  > Media – represents the media source
  > Tracks – audio, video and subtitles currently supported
  > Duration, size, etc
  > Metadata information
  > MediaPlayer – controls for playing media
  > MediaView – display for MediaPlayer
  > Access to events and exceptions
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]
    }
}
```
JavaFX Application
Top Level Container

- **Stage** is the top level container
  - Regular, Applet, JavaWeb Start, mobile
  - Defines the characteristics like title, size, location, handling exit, etc
  - Provides hooks for handling different types of deployment – eg. Applet

- **Stage contains** **Scene**
  - The 'panel' for displaying the content
Example of a Stage

Stage {
    title: "My app"
    extension: [
        AppletStageExtension { //Valid for applet only
            onDragStart: function(e: MouseEvent): Void {
                circle.fill = Color.LIGHTGREEN;
            }
            onDragFinished: function(e: MouseEvent): Void {
                circle.fill = Color.PINK;
            }
        }
    ]
    scene: Scene {
        width: 500 height: 300
        content: [ circle ]
    }
}
THANK YOU

Lee Chuk Munn
Staff Engineer
chuk-munn.lee@sun.com

SUN TECH DAYS 2008–2009
A Worldwide Developer Conference