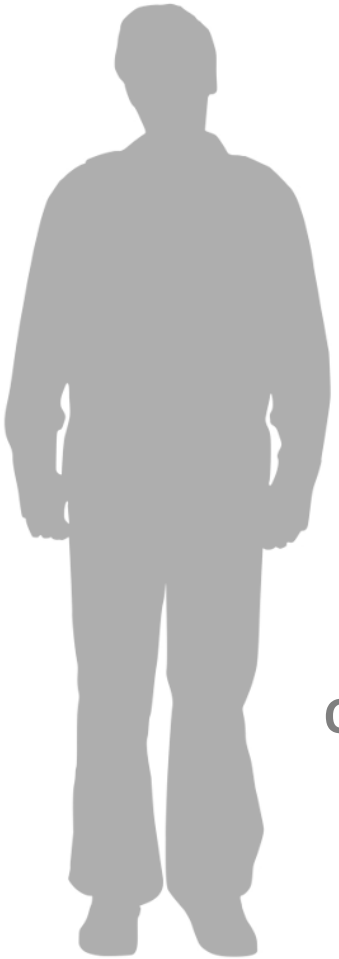
GroovyFX entfesselt JavaFX

OOP 2012

Stefan Glase

am 26.01.2012

Stefan Glase, OPITZ CONSULTING



Software-Entwickler

Java EE, Spring, Groovy, Grails

Trainer und Coach

Sprecher und Autor

OOP, GearConf, DOAG, CamelCaseConf, JUGs

OPITZ CONSULTING

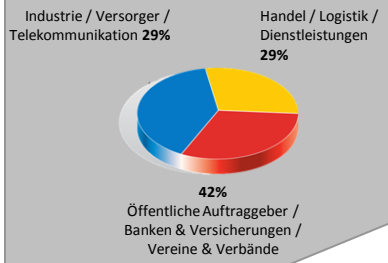
Menschen. Innovationen. Lösungen.

Märkte

- Java
- SOA
- ORACLE
- BI/DWH
- Outtasking

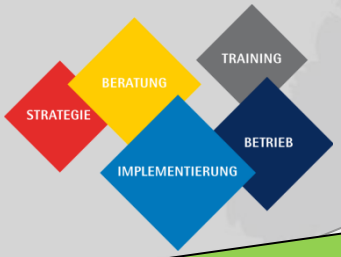
Kunden

- Branchen-übergreifend
- Über 600 Kunden



Leistungs-angebot

- IT-Strategie
- Beratung
- Implementierung
- Betrieb
- Training



Fakten

- Gründung 1990
- 400 Mitarbeiter
- 8 Standorte in D/PL

Besuchen Sie OPITZ CONSULTING am Stand 6.3!

Agenda und Ziele

- Groovy Basics
- Was ist GroovyFX?
- GroovyFX an Beispielen
  - Properties
  - Building DSL
  - Animationen
  - Events
- Fazit

GROOVY BASICS

Was ist Groovy?

- Dynamische Sprache für die Java Virtual Machine (JVM)
- Nahtlose Integration existierender Java Klassen und Bibliotheken

- Vereinfachtes Testen dank Power Asserts und Mocking
- Ausdrucksstarker Code durch kompaktere Syntax, Support für domänenspezifische Sprachen (DSLs), Closures

Hello World mit Groovy

```
class Greeter {
    def name
    def greet() { "Hello $name!" }
}

helloGroovy = new Greeter(name: 'Groovy')
println helloGroovy.greet()
```

Groovy im Web ausprobieren

http://groovyconsole.appspot.com/

Objekte erstellen mit Groovy

```
class Person {
    Long id
    String firstName
    String lastName
}

def person = new Person(
    id: 1,
    firstName: 'Fred',
    lastName: 'Feuerstein'
)

assert person.id == 1
assert person.firstName == 'Fred'
assert person.lastName == 'Feuerstein'
```

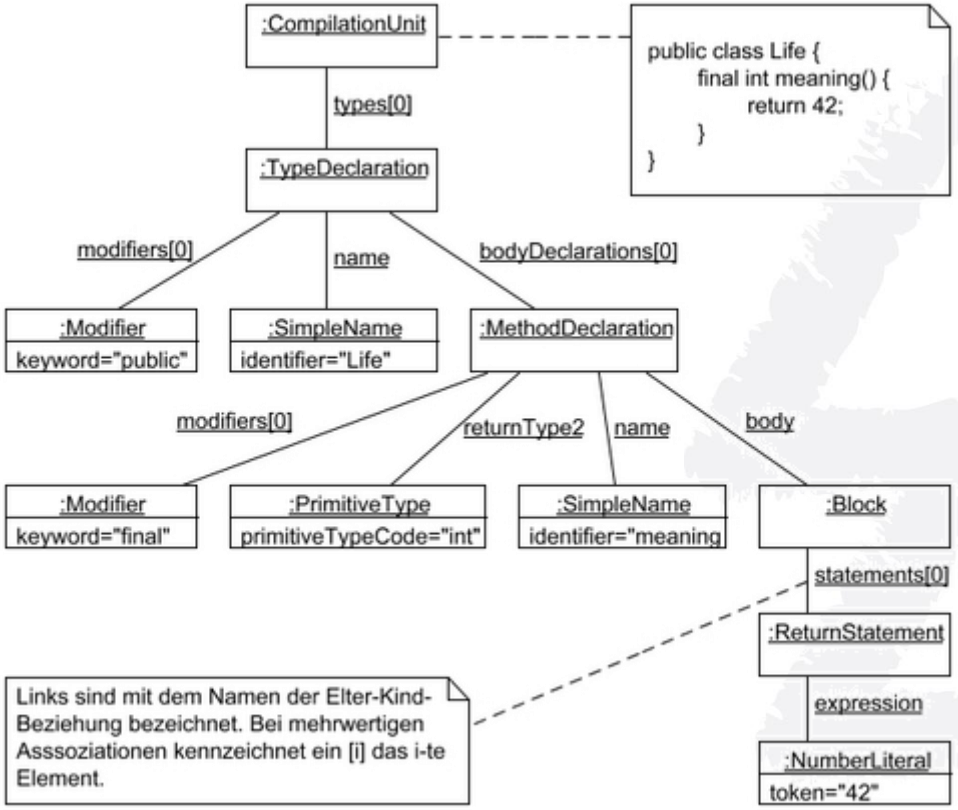
AST-Transformationen mit Groovy

```
@groovy.transform.ToString
class Person {
    Long id
    String firstName
    String lastName
}

def person = new Person().with {
    id = 1
    firstName = 'Fred'
    lastName = 'Feuerstein'
    delegate
}

assert 'Person(1, Fred, Feuerstein)' == person.toString()
```

Abstract Syntax Tree?



http://wiki.fernuni-hagen.de/eclipse/index.php/Abstract_Syntax_Tree_(AST)

Operationen auf Collections mit Groovy

```
class Person {
    Long id
    String firstName
    String lastName
}

def people = [
    new Person(id: 1, firstName: 'Fred', lastName: 'Feuerstein'),
    new Person(id: 2, firstName: 'Wilma', lastName: 'Feuerstein'),
    new Person(id: 3, firstName: 'Betty', lastName: 'Geröllheimer'),
    new Person(id: 4, firstName: 'Barney', lastName: 'Geröllheimer'),
    new Person(id: 5, firstName: 'Bam-Bam', lastName: 'Geröllheimer')]

assert ['Fred', 'Wilma'] ==
       people.findAll{it.lastName == 'Feuerstein'}.firstName
assert ['Feuerstein':2, 'Geröllheimer':3] == people.countBy{it.lastName}

people.findAll{it.lastName == 'Geröllheimer'}
       .each{ println "Hello $it.firstName!"}
```

Vereinfachtes File-Handling mit Groovy

```
def file = new File('myTemp.file')

file.text = """Good day
Guten Tag
Buenos Dias"""

file.eachLine { line, i -> println "$i: $line" }

println file.text
```

WAS IST GROOVYFX?

Was ist GroovyFX?

The primary goal of GroovyFX is to **make JavaFX development simpler and more concise** than what it takes in Java. This is done via numerous built-in features that Groovy provides, including the Tree Structured Language supported through **Groovy's Builder framework** that makes declaring a JavaFX SceneGraph more **closely resemble the actual SceneGraph itself**. This is done through GroovyFX's SceneGraphBuilder object, that supports all the Controls, Shapes, Effects, and other JavaFX objects, as well as support for using **Groovy closures for event handling**.

http://fxexperience.com/2011/07/interview-with-the-developers-behind-groovyfx/

Ein paar Fakten

- Open-Source-Projekt
  - Alpha 1.0 Stadium
  - http://svn.codehaus.org/gmod/groovyfx/trunk
  - https://github.com/groovyfx-project/groovyfx
- Project-Lead
  - Jim Clarke, Dean Iverson, Dierk König
- Lizenz
  - Apache License, Version 2.0

Hello World mit GroovyFX

```
package samples

import groovyx.javafx.SceneGraphBuilder
import groovyx.javafx.GroovyFX
import javafx.scene.text.Font
import javafx.scene.text.FontWeight

def myFont = Font.font('Verdana', FontWeight.BOLD, 64)

GroovyFX.start {
    def sgb = new SceneGraphBuilder()
    sgb.stage(title: 'Hello OOP', width: 600, height: 280, visible: true) {
        scene(fill: white) {
            text(text: 'Hello OOP', x: 120, y: 140, font: myFont, fill: red)
        }
    }
}
```

Zum Ausprobieren: gradle HelloWorldSample

Hello World mit GroovyFX

Zum Ausprobieren: gradle HelloWorldSample

IDE Support

- .gdsl-Extension für IntelliJ IDEA 9+
  - http://jetbrains.dzone.com/articles/custom-groovy-dsl-support
- „GroovyFX-DSL-Deskriptor“ groovyfx.gdsl:

groovyfx.gdsl

```
package idesupport
/**
 * @author dean
 */

title = [title: 'java.lang.String']
visible = [visible: 'java.lang.Boolean']
position2d = [x: 'java.lang.Double', y: 'java.lang.Double']
size2d = [width: 'java.lang.Double', height: 'java.lang.Double']
fill = [fill: 'javafx.scene.paint.Paint']

def sgbContext = context(ctype: 'groovyx.javafx.SceneGraphBuilder')

contributor(sgbContext) {
```

GroovyFX an Beispielen

PROPERTIES MIT GROOVYFX

Properties in JavaFX

```
public class PersonInJava {
    private StringProperty firstName;
    private StringProperty lastName;

    public void setFirstName(String firstName) {
        firstNameProperty().set(firstName);
    }

    public String getFirstName() {
        return firstNameProperty().get();
    }

    public StringProperty firstNameProperty() {
        if (firstName == null)
            firstName = new SimpleStringProperty(this, "firstName");
        return firstName;
    }

    // [...]
}
```

Properties in GroovyFX

```
import groovyx.javafx.beans.FXBindable
import groovy.transform.Canonical

@Canonical
class Person {
    @FXBindable String firstName = 'Max'
    @FXBindable String lastName = 'Mustermann'
}
```

- **@Canonical**
  - **Groovy AST-Transformation für Tupel-Konstruktor, equals(), hashCode() und toString()**
- **@FXBindable**
  - **GroovyFX AST-Transformation für get<Eigenschaft>(), set<Eigenschaft>() und get<Eigenschaft>Property()**

Properties Beispiel

```
@Canonical
class Person {
    @FXBindable String firstName = 'Max'
    @FXBindable String lastName = 'Mustermann'
}
def person = new Person('Stefan', 'Glase')
println person.firstName

GroovyFX.start {
    new SceneGraphBuilder().stage(title: 'Properties', show: true) {
        scene(width: 300) {
            vbox {
                textField(text: bind(person.firstNameProperty()))
                textField(text: bind(person.lastNameProperty()))
            }
        }
    }
}
```

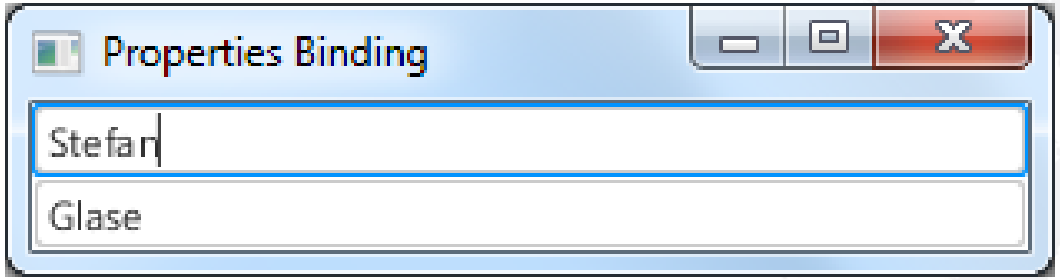
Optionale Defaults

Lesender und schreibender Zugriff auf Werte

Property Binding

Zum Ausprobieren: gradle PropertyBindingSample

Properties Beispiel

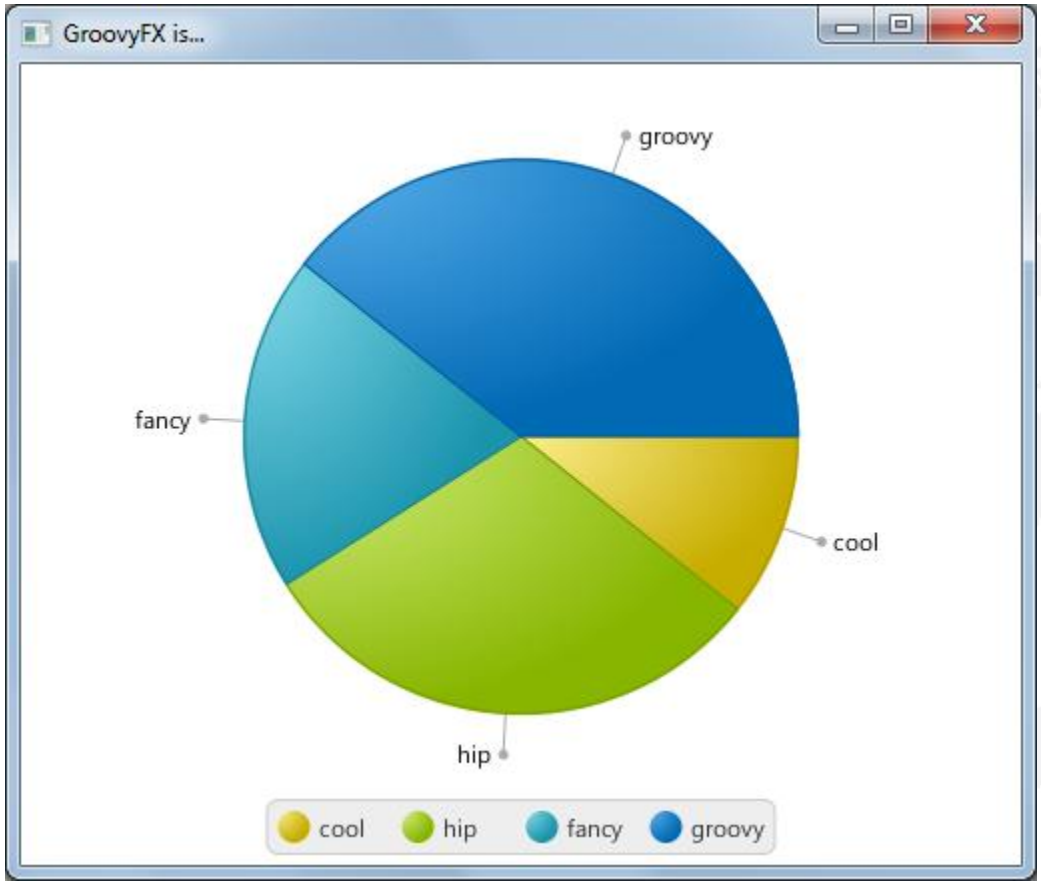


GroovyFX an Beispielen

BUILDING-DSL MIT GROOVYFX

Building-DSL mit GroovyFX

am Beispiel eines PieChart



Zum Ausprobieren: gradle PieChartSample

Erforderlicher Source-Code

```
package samples

import groovyx.javafx.GroovyFX
import groovyx.javafx.SceneGraphBuilder

def groovyFx = [cool: 12, hip: 34, fancy: 22, groovy: 44]

GroovyFX.start {
    def sgb = new SceneGraphBuilder()
    sgb.stage(title: 'GroovyFX is...', visible: true) {
        scene {
            pieChart(data: groovyFx)
        }
    }
}
```

„Datenquelle“

Komponente

Was steckt dahinter? Eine Factory!

```
class PieChartFactory extends NodeFactory {
    @Override
    Object newInstance(FactoryBuilderSupport builder, Object name, Object value, Map attributes) {
        if (FactoryBuilderSupport.checkValueIsType(value, name, PieChart)) {
            return value
        } else {
            return createChart(attributes)
        }
    }
    private PieChart createChart(Map attributes) {
        def chart = new PieChart()
        def data = attributes.remove("data")
        if (data) {
            chart.data = createPieChartData(data)
        }
        return chart
    }
    private ObservableList<PieChart.Data> createPieChartData(data) {
        if (data instanceof ObservableList) {
            return data
        } else if (data instanceof Map) {
            def dataList = ((Map) data).collect {String k, Double v -> new PieChart.Data(k, v)}
            return FXCollections.observableArrayList(dataList)
        } else {
            throw new RuntimeException("Could not recognize the pie chart data '$data'. Try an ObservableList " +
                    "of PieChart.Data objects or a Map of strings to values: ['Label 1': 75, 'Label 2': 25]")
        }
    }
}
```

Handelt es sich bereits um ein PieChart?

In welchem Format liegen die Daten vor?

Was steckt dahinter? Ein Builder!

```
public class SceneGraphBuilder extends FactoryBuilderSupport {

    [SNIP]

    public def registerCharts() {
        registerFactory('pieChart', new PieChartFactory())
        registerFactory('lineChart', new XYChartFactory(LineChart))
        registerFactory('areaChart', new XYChartFactory(AreaChart))
        registerFactory('bubbleChart', new XYChartFactory(BubbleChart))
        registerFactory('barChart', new XYChartFactory(BarChart))
        registerFactory('scatterChart', new XYChartFactory(ScatterChart))
        registerFactory('numberAxis', new AxisFactory(NumberAxis))
        registerFactory('categoryAxis', new AxisFactory(CategoryAxis))
        registerFactory('series', new XYSeriesFactory())
    }

    [SNIP]

}
```

Registrierung aller unterstützten Charts

GroovyFX an Beispielen

ANIMATIONEN MIT GROOVYFX

TimelineBuilder

```
class TimelineBuilder extends FactoryBuilderSupport {
    // local fields
    private static final Logger LOG = Logger.getLogger(TimelineBuilder.name)
    private static boolean headless = false

    public static final String DELEGATE_PROPERTY_OBJECT_ID = "_delegateProperty:id";
    public static final String DEFAULT_DELEGATE_PROPERTY_OBJECT_ID = "id";

    public TimelineBuilder(boolean init = true) {
        super(init)
        this[DELEGATE_PROPERTY_OBJECT_ID] = DEFAULT_DELEGATE_PROPERTY_OBJECT_ID
        ExpandoMetaClass.enableGlobally()
        Number.metaClass.getM = { -> new Duration(delegate*1000.0*60.0)};
        Number.metaClass.getS = { -> new Duration(delegate*1000.0)};
        Number.metaClass.getMs = { -> new Duration(delegate)};
        Number.metaClass.getH = { -> new Duration(delegate*1000.0*60.0*60.0)};

    }

    public def registerTimelines() {
        registerFactory("timeline", new TimelineFactory())

        registerFactory("at", new KeyFrameFactory())
        registerFactory("action", new KeyFrameActionFactory())
        registerFactory("change", new KeyValueFactory())
        registerFactory("to", new KeyValueSubFactory())
        registerFactory("tween", new KeyValueSubFactory())

    }

    def propertyMissing(String name) {
        switch(name.toLowerCase()) {
            case "ease_both":
            case "easeboth":
```

Animationen mit GroovyFX

```
GroovyFX.start {
    def sgb = new SceneGraphBuilder(it)
    sgb.stage(title: "Move!", width: 400, height: 400, visible: true) {
        scene(fill: greenyellow) {
            redCircle = sgb.circle(radius: 25) { fill(red) }
        }
        timeline(cycleCount: indefinite, autoReverse: true) {
            at(800.ms) {
                change(redCircle, 'layoutX') to 200 tween easeboth
                change(redCircle, 'layoutY') to 200 tween easeboth
                change(redCircle, 'scaleX') to 4
                change(redCircle, 'scaleY') to 4
            }
        }.play()
    }
}
```

Animationen mit GroovyFX

Einfache Syntax zur Beschreibung von Animationen

```
GroovyFX.start {
    def sgb = new SceneGraphBuilder(it)
    sgb.stage(title: "Move!", width: 4[illegible]ght: 400, visible: true) {
        scene(fill: greenyellow) {
            redCircle = sgb.circl[illegible]dius: 25) { fill(red) }
        }
        timeline(cycleCount: indefinite, autoReverse: true) {
            at(800.ms) {
                change(redCircle, 'layoutX') to 200 tween easeboth
                change(redCircle, 'layoutY') to 200 tween easeboth
                change(redCircle, 'scaleX') to 4
                change(redCircle, 'scaleY') to 4
            }
        }.play()
    }
}
```

Animationen mit GroovyFX

```
GroovyFX.start {
    def sgb = new SceneGraphBuilder(it)
    sgb.stage(title: "Move!", width: 400, height: 400, visible: true) {
        scene(fill: greenyellow) {
            redCircle = sgb.circle(radius: 25) { fill(red) }
        }
        timeline(cycleCount: indefinite, autoReverse: true) {
            at(800.ms) {
                change(redCircle, 'layoutX') to 200 tween easeboth
                change(redCircle, 'layoutY') to 200 tween easeboth
                change(redCircle, 'scaleX') to 4
                change(redCircle, 'scaleY') to 4
            }
        }.play()
    }
}
```

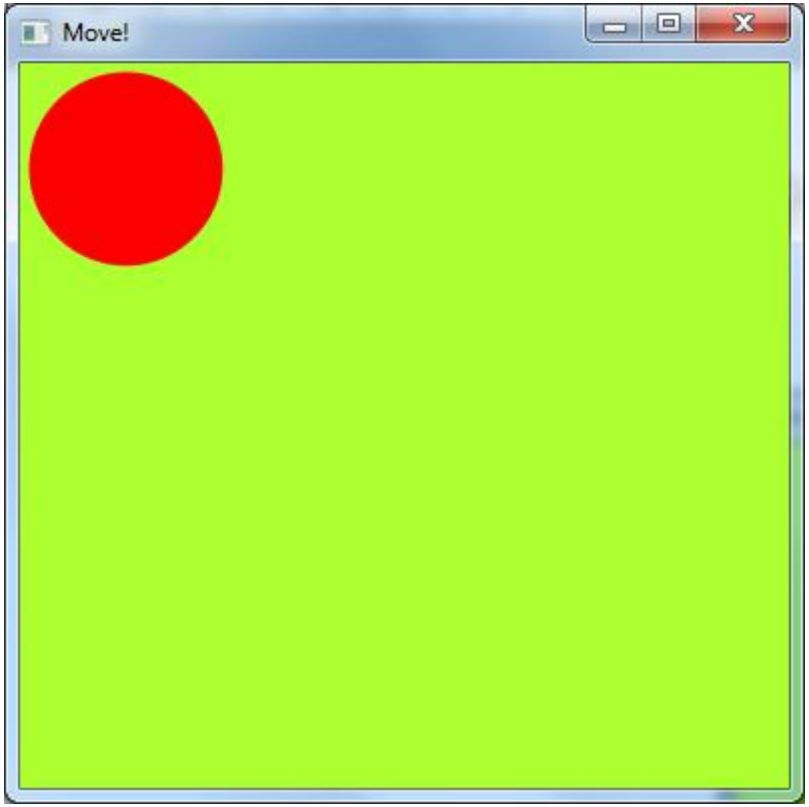
DSL zur Beschreibung der KeyFrames

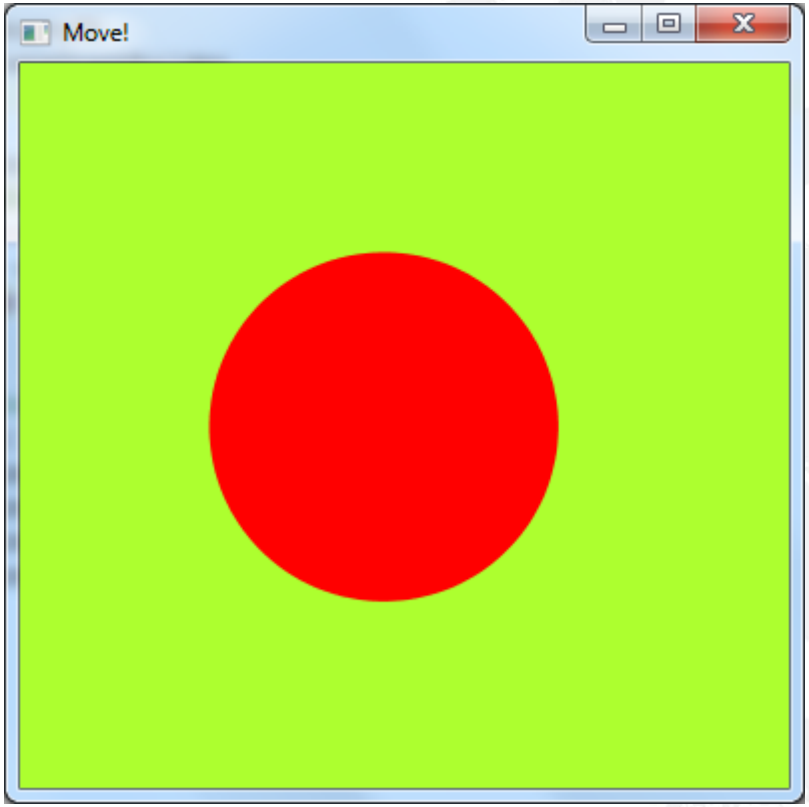
Animationen mit GroovyFX

```
GroovyFX.start {
    def sgb = new SceneGraphBuilder(it)
    sgb.stage(title: "Move!", width: 400, height: 400, visible: true) {
        scene(fill: greenyellow) {
            redCircle = sgb.circle(radius: 25) { fill(red) }
        }
        timeline(cycleCount: indefinite, autoReverse: true) {
            at(800.ms) {
                change(redCircle, 'layoutX') to 200 tween easeboth
                change(redCircle, 'layoutY') to 200 tween easeboth
                change(redCircle, 'scaleX') to 4
                change(redCircle, 'scaleY') to 4
            }
        }.play()
    }
}
```

Optionale Angabe der Interpolation

Animationen mit GroovyFx





Zum Ausprobieren: gradle MovingBallSample

GroovyFX an Beispielen

EVENTS MIT GROOVYFX

Events mit Groovy

```
public def registerInputListeners() {
    MouseHandlerFactory mf = new MouseHandlerFactory()
    KeyHandlerFactory kf = new KeyHandlerFactory()
    ActionHandlerFactory af = new ActionHandlerFactory()

    registerFactory('onMouseClicked', mf)
    registerFactory('onMouseDragged', mf)
    registerFactory('onMouseEntered', mf)
    registerFactory('onMouseExited', mf)
    registerFactory('onMouseMoved', mf)
    registerFactory('onMousePressed', mf)
    registerFactory('onMouseReleased', mf)
    registerFactory('onDragDetected', mf)
    registerFactory('onDragDone', mf)
    registerFactory('onDragEntered', mf)
    registerFactory('onDragExited', mf)
    registerFactory('onDragOver', mf)
    registerFactory('onDragDropped', mf)
    registerFactory('onMouseWheelMoved', mf) // only works for scene.

    registerFactory('onKeyPressed', kf)
    registerFactory('onKeyReleased', kf)
    registerFactory('onKeyTyped', kf)

    registerFactory('onAction', af)
}
```

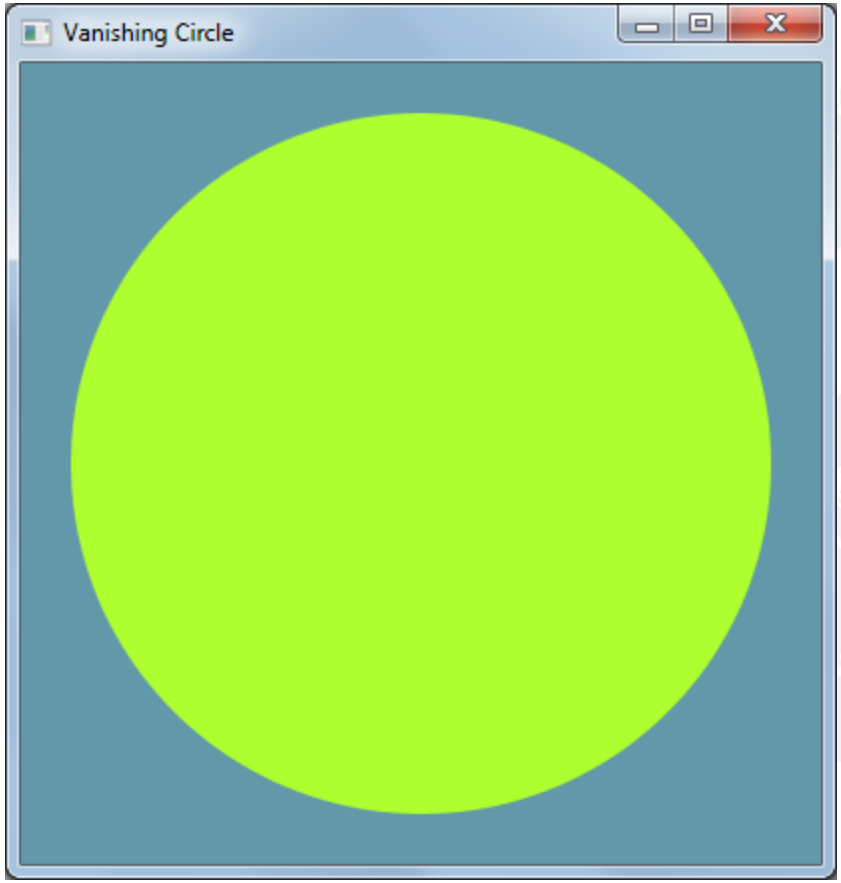
Registrierung der Events in SceneGraphBuilder

Events mit GroovyFX

```
GroovyFX.start {
    new SceneGraphBuilder().stage(title: 'Vanishing...', show: true) {
        scene(fill: groovyblue, width: 400, height: 400) {
            circle(centerX: 200, centerY: 200, radius: 175) {
                fill greenyellow
                onMouseClicked { e ->
                    timeline {
                        at(3.s) {
                            change e.source.radiusProperty() to 0
                        }
                    }.play()
                }
            }
        }
    }
}
```

Kompakte Event-Syntax als Groovy-Closure mit optionalem Zugriff auf das Event

Events mit Groovy



Zum Ausprobieren: gradle VanishingCircleSample

Fazit

“Our goal with GroovyFX is to make it fun and easy to write client Java applications. So join in!” -- Dean Iverson

Im Netz...

Groovy und GroovyFX

http://groovy.codehaus.org

http://groovy.codehaus.org/GroovyFX

http://fxexperience.com

Beispiele:

https://github.com/codescape/presentations

Twitter:

@stefanglase

Fragen und Antworten

Ihr Ansprechpartner

Stefan Glase, Senior Consultant

OPITZ CONSULTING Gummersbach GmbH
stefan.glase@opitz-consulting.com
Telefon: +49 2261 60 01-1093
Twitter: @stefanglase

youtube.com/opitzconsulting

slideshare.net/opitzconsulting

xing.com/group-51062.460375

twitter.com/OC_WIRE