MVVM mit JavaFX

Manuel Mauky & *Max Wielsch*
@manuel_mauky @simawiel

07.04.2016

JUG Görlitz

Saxonia Systems
So geht Software.

Saxonia Systems

So geht Software.

- Individual-Softwareentwicklung
- Dresden, München, Berlin, Hamburg, Leipzig, **Görlitz**
- 200+ Employees

JavaFX

JavaFX

- Toolkit für Java UIs
- Teil der JRE
- Offizieller Ersatz für Swing

Windows

Mac OS

Linux

JavaFX Features

Media Engine

Hardware Acceleration

Charts

Timelines

Effects & Animations

Properties

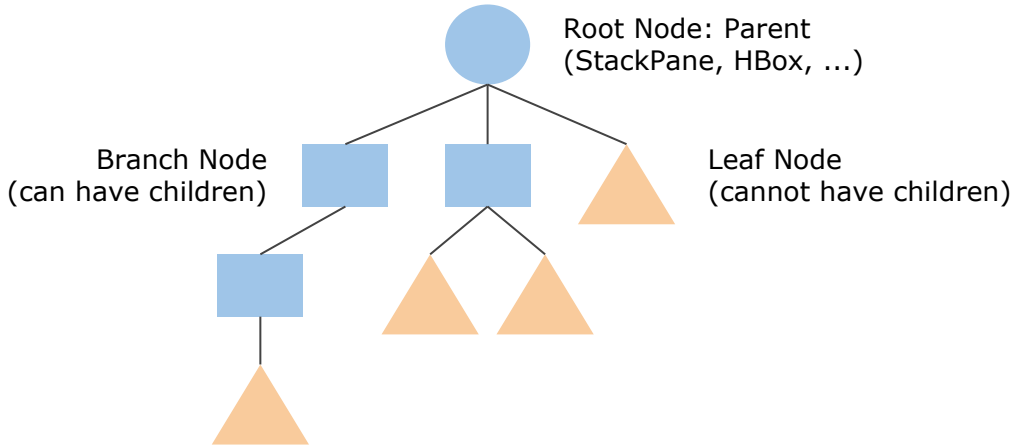
Web View

FXML + CSS

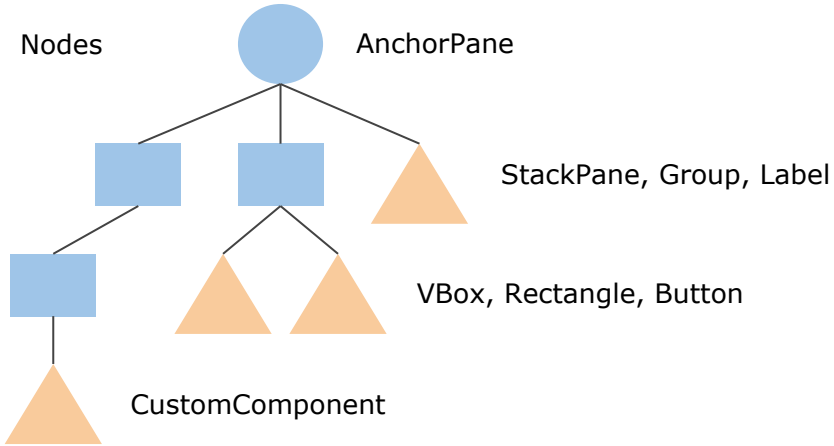
Multi Touch

Packaging

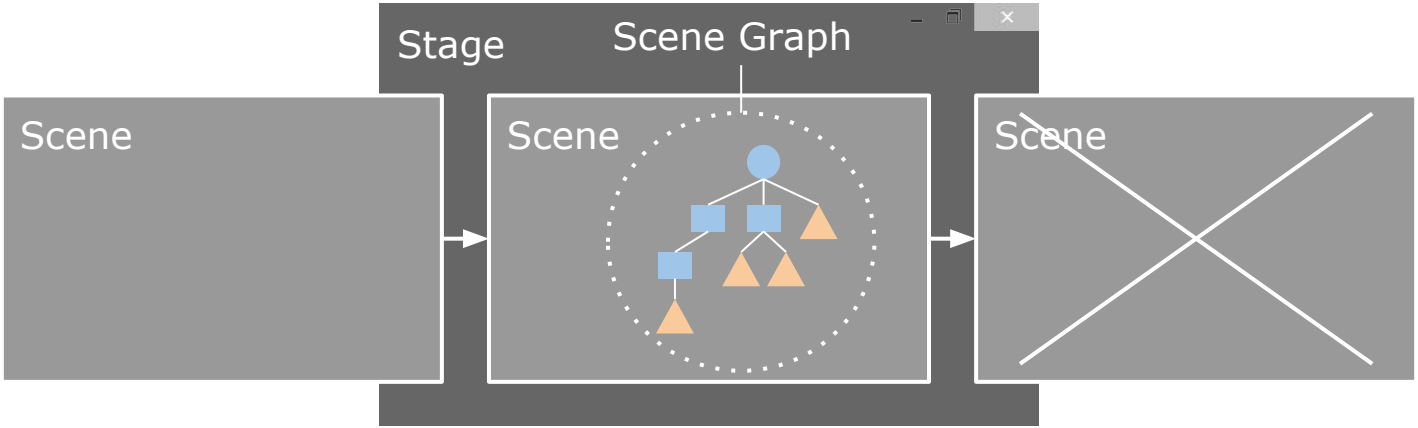
Scene Graph


Root Node: Parent
(StackPane, HBox, ...)
Branch Node
(can have children)
Leaf Node
(cannot have children)

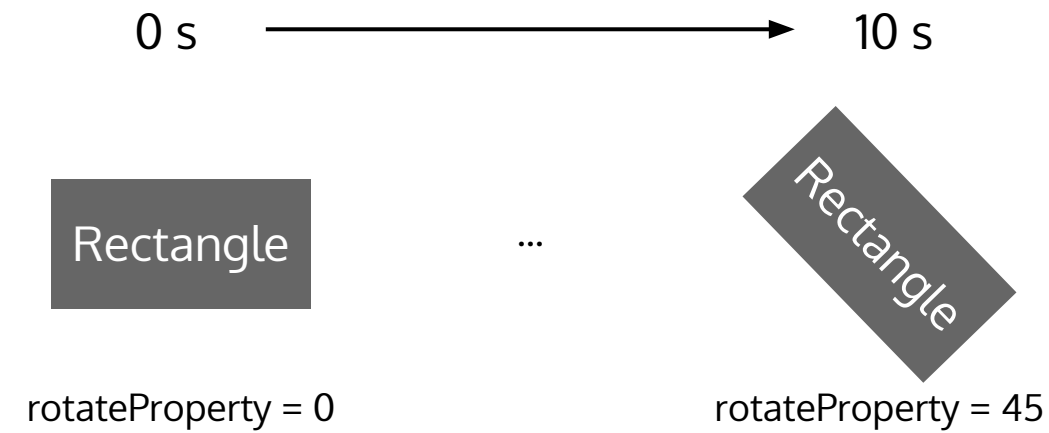
Scene Graph


Nodes
AnchorPane
StackPane, Group, Label
VBox, Rectangle, Button
CustomComponent

Scene Graph



Bsp.: Transitions




```
RotateTransition rotateTransition = new RotateTransition(
                                                Duration.seconds(10),
                                                reactangle);

rotateTransition.setByAngle(45);
rotateTransition.play();
```

Trennung von Darstellung und Verhalten

```
<BorderPane id=“border“>
       <top>
          <Label text=“Page Title“/>
       </top>
       <center>
          <Label text = “Some Data Here“/>
       </center>
</BorderPane>
```

Test.fxml

Structure

```
@FXML
private final BorderPane border;

public MainView() {
  border.setDisabled(true);
}
```

Test.java

Verhalten

```
#border {
    -fx-background-color:white;
}

.Label {
    -fx-text-fill: black;
}
```

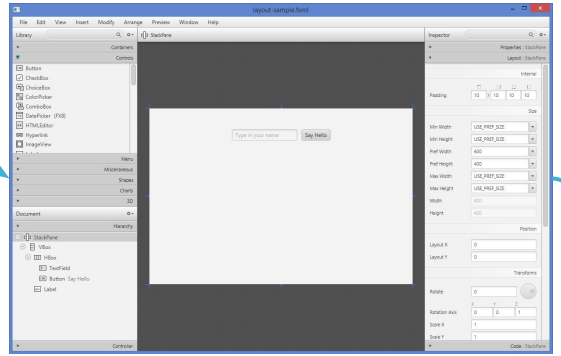
Test.css

Darstellung

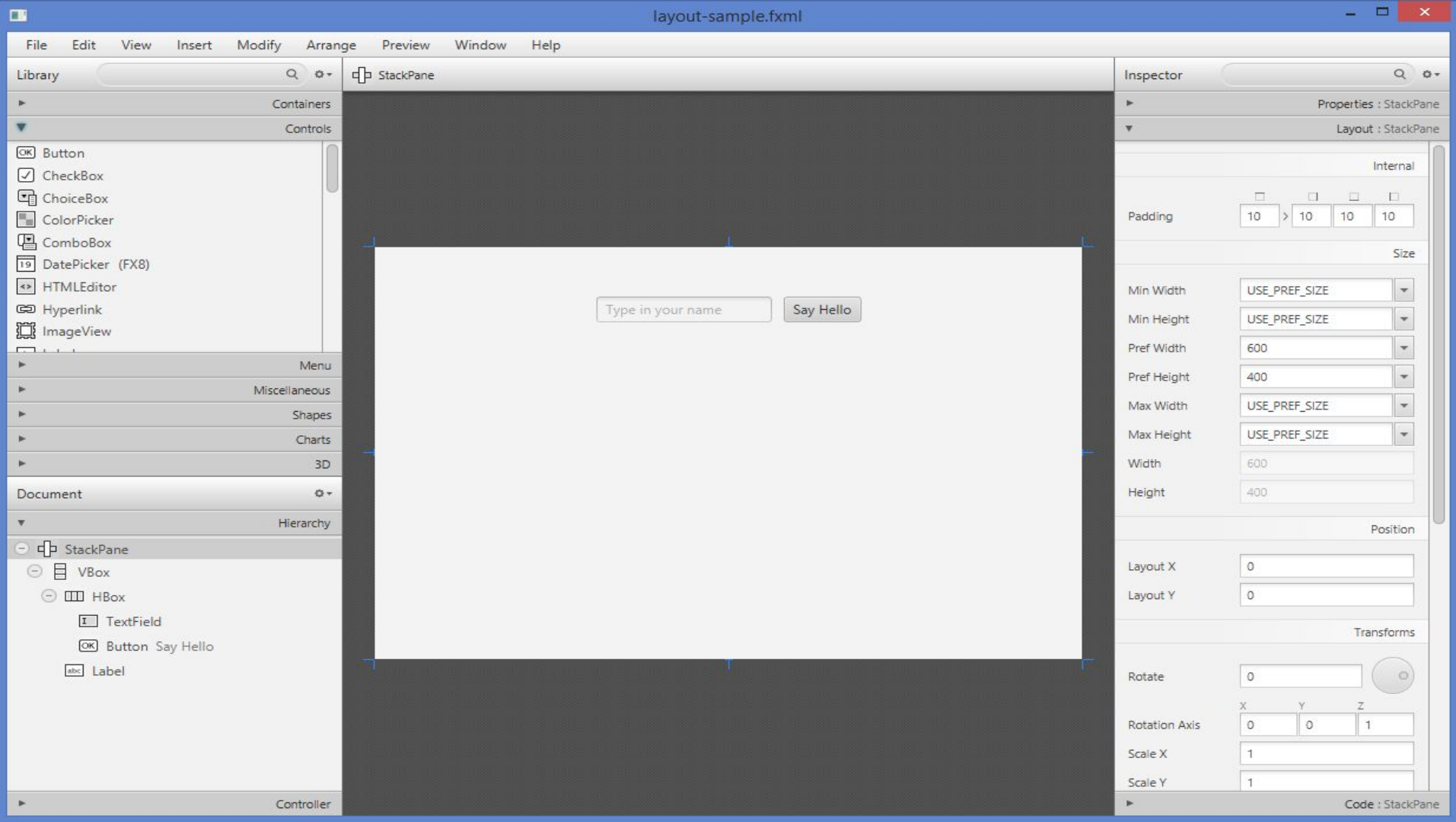
Saubere Trennung dieser Aspekte

Gluon Scene Builder

```
<BorderPane id="border">
        <top>
           <Label text="Page Title"/>
        </top>
        <center>
           <Label text = "Some Data Here"/>
        </center>
</BorderPane>
```



```
<BorderPane id="border">
        <top>
           <Label text="Page Title"/>
        </top>
        <center>
           <Label text = "Some Data Here"/>
        </center>
</BorderPane>
```

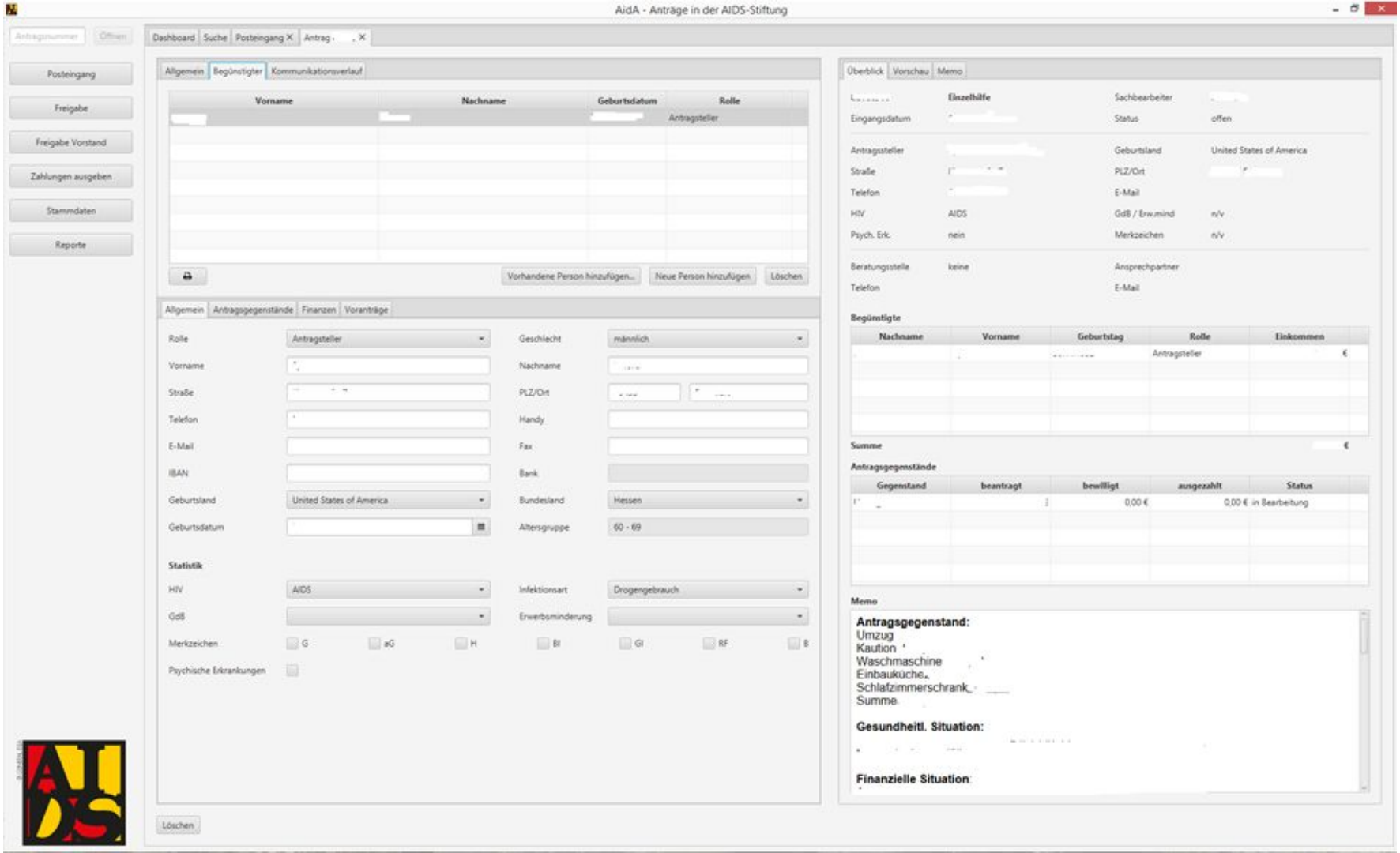

layout-sample.fxml
File
Edit
View
Insert
Modify
Arrange
Preview
Window
Help
Library
Containers
Controls
Button
CheckBox
ChoiceBox
ColorPicker
ComboBox
DatePicker (FX8)
HTMLEditor
Hyperlink
ImageView
Menu
Miscellaneous
Shapes
Charts
3D
Document
Hierarchy
StackPane
VBox
HBox
TextField
Button Say Hello
Label
Controller
StackPane
Type in your name
Say Hello
Inspector
Properties : StackPane
Layout : StackPane
Internal
Padding
10
10
10
10
Size
Min Width
USE_PREF_SIZE
Min Height
USE_PREF_SIZE
Pref Width
600
Pref Height
400
Max Width
USE_PREF_SIZE
Max Height
USE_PREF_SIZE
Width
600
Height
400
Position
Layout X
0
Layout Y
0
Transforms
Rotate
0
X
Y
Z
Rotation Axis
0
0
1
Scale X
1
Scale Y
1
Code : StackPane

JavaFX @ saxsys


ETEO


e·t·e·o Board
ETEO


1. Sprint - Spielwiese
Time
Effort
ETEO Open
ETEO In Progress
ETEO In Review
ETEOB-1
ETEOB-2
ETEOB-3
ETEOB-4
ETEOB-5
Site
Call for bids
Excavation work
Lawn
Jungle gym
P3 ETEOB-207
P3 ETEOB-209
P3 ETEOB-37
Seesaw
Carousel
Benches
Slide
eteoBoard Premium -
Software inclusive Hardware InFocus Mondopad


AidA - Anträge in der AIDS-Stiftung
Antragsnummer
Öffnen
Dashboard
Suche
Posteingang ×
Antrag ×
Posteingang
Freigabe
Freigabe Vorstand
Zahlungen ausgeben
Stammdaten
Reporte
Allgemein
Begünstigter
Kommunikationsverlauf
Vorname
Nachname
Geburtsdatum
Rolle
Antragsteller
Vorhandene Person hinzufügen...
Neue Person hinzufügen
Löschen
Allgemein
Antragsgegenstände
Finanzen
Voranträge
Rolle
Antragsteller
Geschlecht
männlich
Vorname
Nachname
Straße
PLZ/Ort
Telefon
Handy
E-Mail
Fax
IBAN
Bank
Geburtsland
United States of America
Bundesland
Hessen
Geburtsdatum
Altersgruppe
60 - 69
Statistik
HIV
AIDS
Infektionsart
Drogengebrauch
GdB
Erwerbsminderung
Merkzeichen
G
aG
H
Bl
Gl
RF
B
Psychische Erkrankungen
Löschen
Überblick
Vorschau
Memo
Einzelhilfe
Sachbearbeiter
Eingangsdatum
Status
offen
Antragsteller
Geburtsland
United States of America
Straße
PLZ/Ort
Telefon
E-Mail
HIV
AIDS
GdB / Erw.mind
n/v
Psych. Erk.
nein
Merkzeichen
n/v
Beratungsstelle
keine
Ansprechpartner
Telefon
E-Mail
Begünstigte
Nachname
Vorname
Geburtstag
Rolle
Einkommen
Antragsteller
Summe
Antragsgegenstände
Gegenstand
beantragt
bewilligt
ausgezahlt
Status
0,00 €
0,00 €
in Bearbeitung
Memo
Antragsgegenstand:
Umzug
Kaution
Waschmaschine
Einbauküche
Schlafzimmerschrank
Summe
Gesundheitl. Situation:
Finanzielle Situation:

APPLIKATIONS STRUKTUR

Applikation

Applikations Modell

Präsentation

Applikations Modell

Aussehen

Verhalten

MODEL VIEW VIEWMODEL

Model View ViewModel

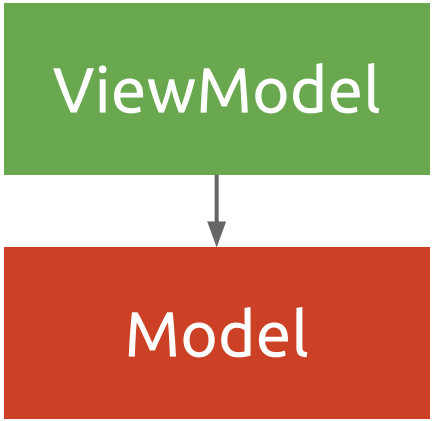
- Basiert auf Presentation Model Pattern
- 2005 von Microsoft publiziert
- WPF (.NET), JavaScript (knockout.js), ...

Model

- Applikationsmodell
- Unabhängig von UI
- Backend-System usw.

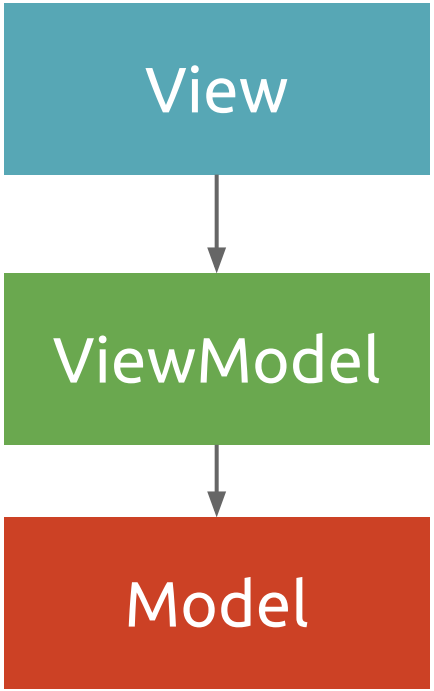
ViewModel

- UI-Zustand
- Präsentationslogik
- Kommunikation mit Backend
- Aufbereitung von Modell-Daten

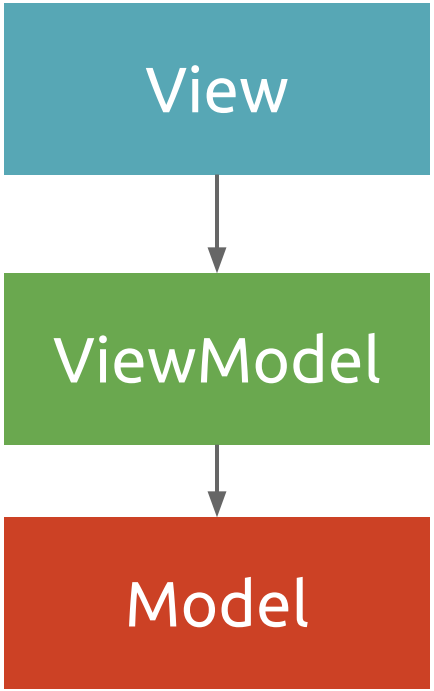


View

- Daten aus ViewModel anzeigen
- Nutzereingaben an ViewModel weitergeben
- UI-Zustand im ViewModel aktualisieren

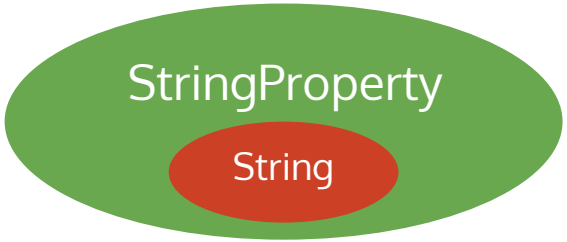


VIEW UND VIEWMODEL SYNCHRON HALTEN?


View
ViewModel
Model

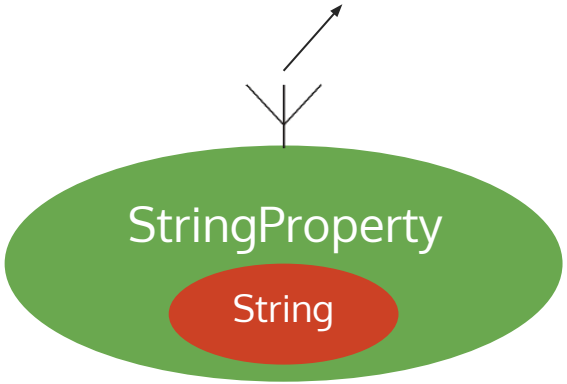
Data Binding und Properties

Data Binding und Properties

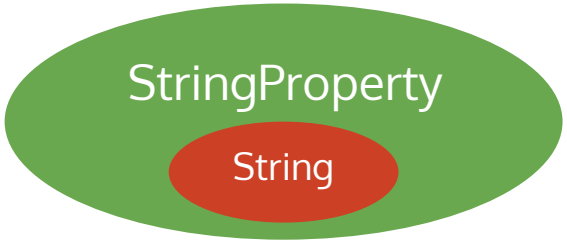


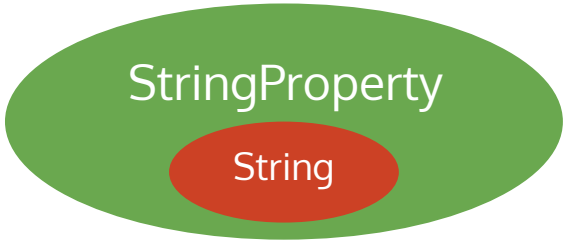
Data Binding und Properties

notifications about changes
(events)

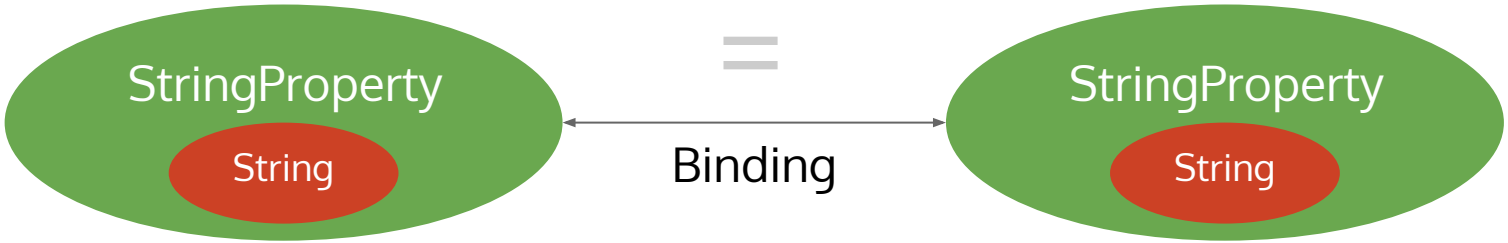


Data Binding und Properties





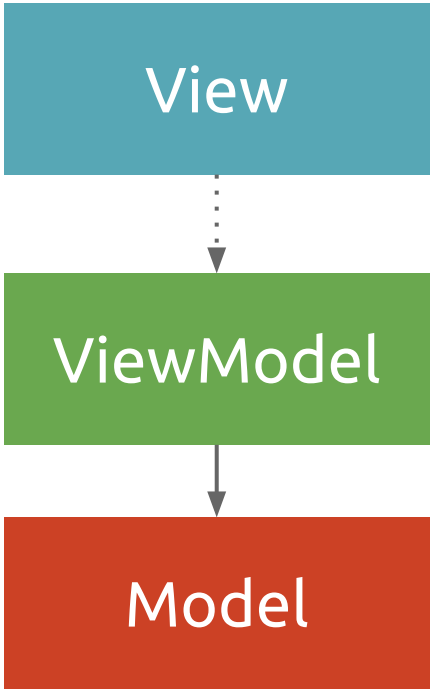
Data Binding und Properties



```
StringProperty a = new SimpleStringProperty();
StringProperty b = new SimpleStringProperty();

a.bindBidirectional(b);

a.setValue(“Hallo”);
System.out.println(b.getValue()); // “Hallo”
b.setValue(“World”);
System.out.println(a.getValue()); // “World”
```

```
IntegerProperty
DoubleProperty
StringProperty
BooleanProperty
ListProperty<T>
ObjectProperty<T>
MapProperty<K,V>
```

Data Binding in MVVM




```
welcomeLabel.textProperty().bind(
    viewModel.welcomeMessageProperty());

welcomeMessageProperty.set(
     “Hallo “
     + user.getFirstName() + “ ”
     + user.getLastName());
```

“Hallo Max Wielsch”

firstName — *“Max”*

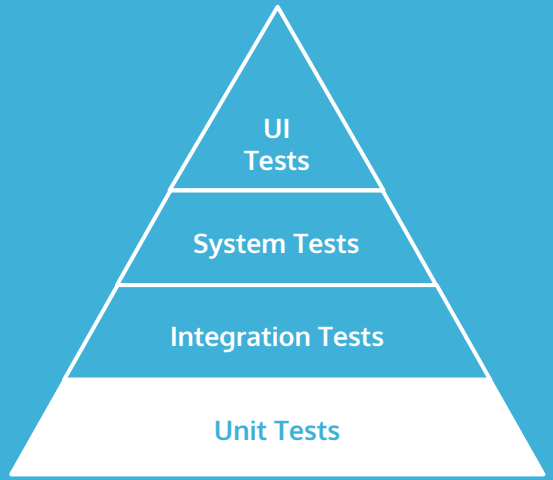
lastName — *“Wielsch”*

Nutzen

- gesamte Präsentationslogik ist nur im ViewModel
- ViewModel ist mit Unit-Tests testbar

→ Hohe Testbarkeit von Frontend-Code

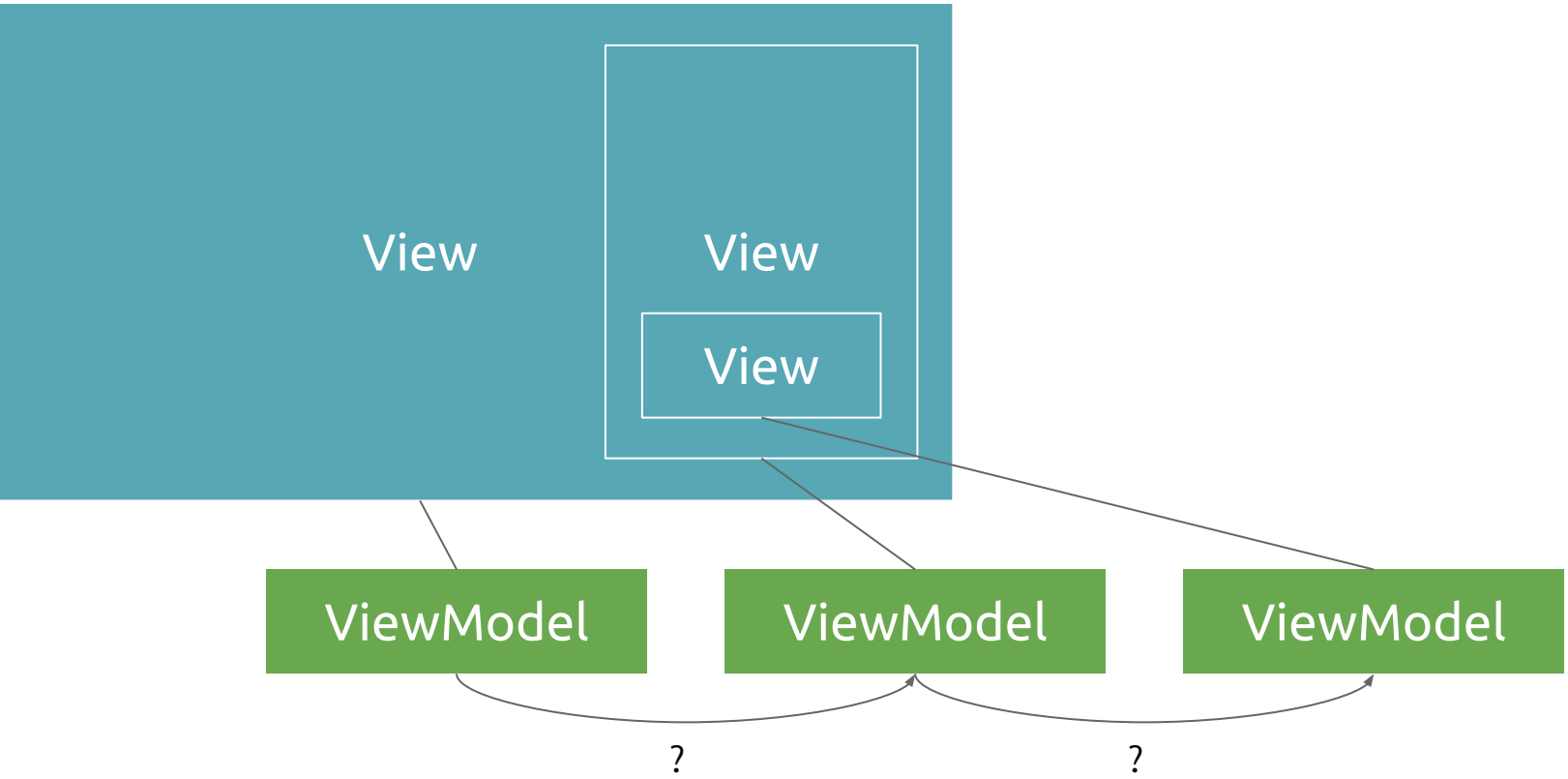
→ Test-Driven-Development im der UI



LIVE CODING TDD

Example: Todo List

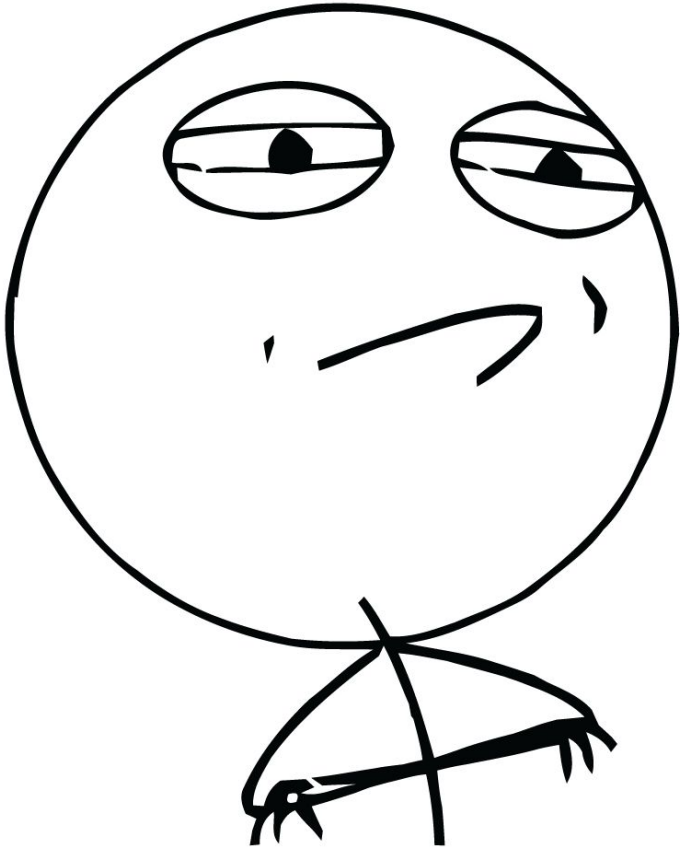
View Hierarchien



Herausvorderungen mit MVVM

- Kommunikation zwischen VMs in komplexeren Anwendungen
- Erfordert Disziplin, Sichtbarkeiten nicht zu verletzen
- Mehr Code für Indirektionsschicht nötig

CHALLENGE ACCEPTED



... ist ein Application-Framework, das benötigte Komponenten für die Verwendung des Model-View-ViewModel-Patterns mit JavaFX zur Verfügung stellt.

https://github.com/sialcasa/mvvmFX

Basis Klassen für MVVM

Erweiterter FXML Loader

Dependency-Injection-Unterstützung

ResourceBundles

ModelWrapper

Notifications

Commands

Validation

Scopes

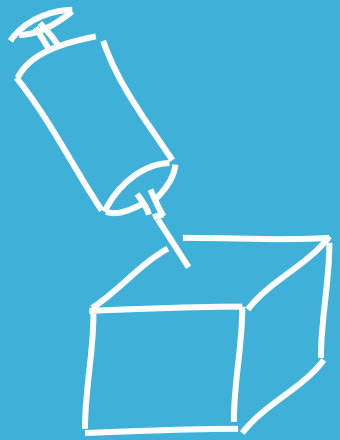
Base Classes / Interfaces

```
class TodolistViewModel implements ViewModel {
    …
}

class TodolistView implements FxmlView<TodolistViewModel> {

    @InjectViewModel
    private TodolistViewModel viewModel;
}
```

```
URL url = getClass().getResource(“/de/saxsys/MyView.fxml”);
FXMLLoader loader = new FXMLLoader(url);
loader.load();
loader.getRoot(); // loaded node
loader.getController(); // controller class
```

```
URL url = getClass().getResource(“/de/saxsys/MyView.fxml”);
FXMLLoader loader = new FXMLLoader(url);
loader.load();
loader.getRoot(); // loaded node
loader.getController(); // controller class
```

```
ViewTuple tuple = FluentViewLoader.fxmlView(MyView.class).load();
tuple.getView(); // loaded node (View)
tuple.getCodeBehind(); // controller class (View)
tuple.getViewModel(); // ViewModel
```



DEPENDENCY INJECTION

```
class MyViewModel implements ViewModel {

    private Service service = new ServiceImpl(); // ?



}
```

```
class MyViewModel implements ViewModel {

    private Service service = new ServiceImpl(); // ?

    @Inject
    private Service service;

}
```

Dependency Injection

- Inversion of Control
- Keine statische Abhängigkeit zu einer spezifischen Implementierung, nur zu Interfaces
- Mock-Implementierungen für Unit-Tests nutzen
- Lebenszyklus von Instanzen konfigurieren

Dependency Injection Support

```
<dependency>
    <groupId>de.saxsys</groupId>
    <artifactId>mvvmfx-cdi</artifactId>
</dependency>v
```

```
<dependency>
    <groupId>de.saxsys</groupId>
    <artifactId>mvvmfx-guice</artifactId>
</dependency>
```

oder

```
MvvmFX.setCustomDependencyInjector(...);
```

Dependency Injection Support

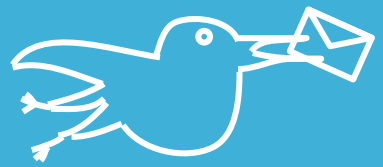
```
public class MyFxApp extends Application { … }
```



```
public class MyFxApp extends MvvmfxCdiApplication { … }
```

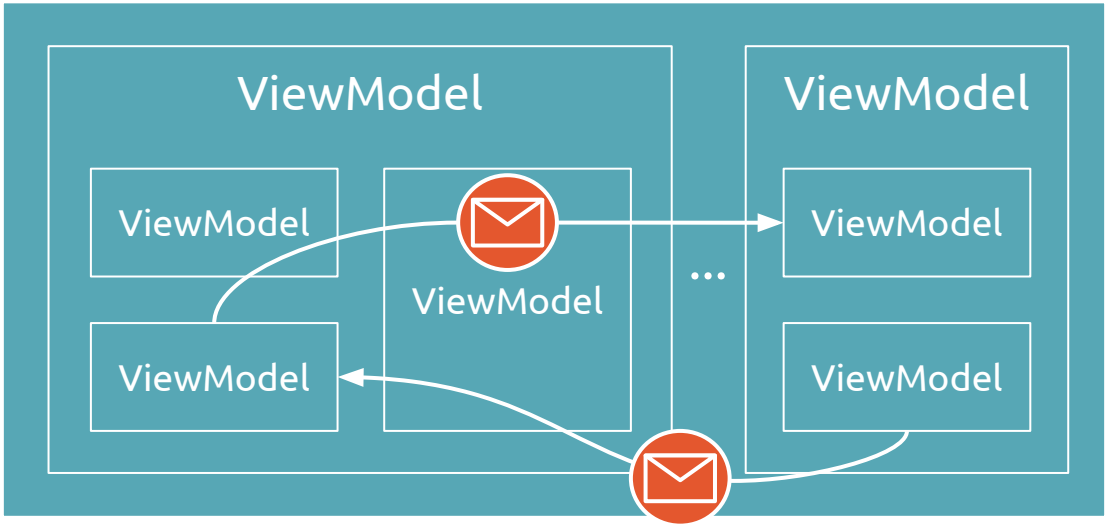
oder

```
public class MyFxApp extends MvvmfxGuiceApplication { … }
```



NOTIFICATIONS

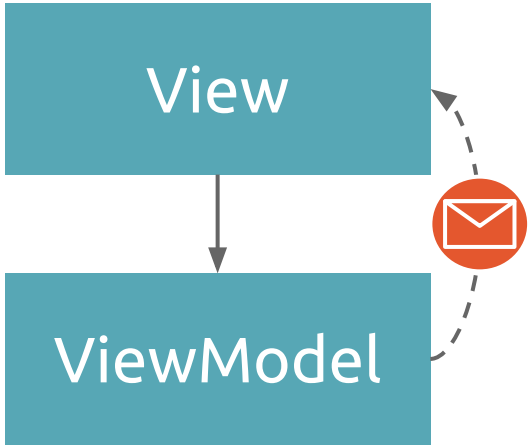
Notifications



Notifications

```
notificationCenter.subscribe(“messageKey”, (k,v) -> …);

notificationCenter.publish(“messageKey”);
```

Notifications



Notifications

```
public class MyView implements FxmlView<MyViewModel> {
    @InjectViewModel
    private MyViewModel viewModel;
    …
    viewModel.subscribe(“messageKey”, (k,v) ->  doSomething());
    …
}
public class MyViewModel implements ViewModel {
    …
    publish(“messageKey”);
    …
}
```

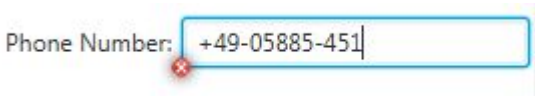
VALIDATION

Validation

Validation logic

```
boolean isPhoneNumberValid(String input) {
    return Pattern.compile("\\+?[0-9\\s]{3,20}")
            .matcher(input).matches();
}
```

Visualization



Validation

ControlsFX ValidationSupport

```
TextField textField = ...;

ValidationSupport validationSupport = new ValidationSupport();

validationSupport.registerValidator(textField,
      Validator.createRegexValidator("Wrong Number", "\\+?[0-9\\s]{3,20}", Severity.ERROR));
```

Validation

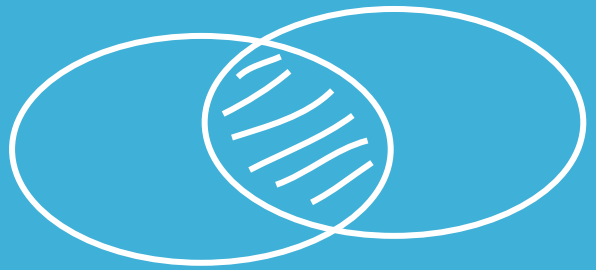
```
// ViewModel

private final Validator phoneValidator = ...

public ValidationStatus phoneValidation() {
   return phoneValidator.getValidationStatus();
}

// View

ValidationVisualizer validVisualizer= new ControlsFxVisualizer();
validVisualizer.initVisualization(viewModel.phoneValidation(), textField);
```

Validation

```
StringProperty phoneNumber = new SimpleStringProperty();

Predicate<String> predicate = input -> Pattern.compile("\\+?[0-9\\s]{3,20}")
                                                    .matcher(input).matches();

Validator phoneValidator = new FunctionBasedValidator<>(
     phoneNumber, predicate, ValidationMessage.error("Not a valid phone number");
```

Validation

```
Validator phoneValidator = new FunctionBasedValidator(...);
Validator emailValidator = new ObservableRuleBasedValidator(...);

Validator formValidator = new CompositeValidator();
formValidator.addValidators(phoneValidator, emailValidator);
```



SCOPES

Scopes

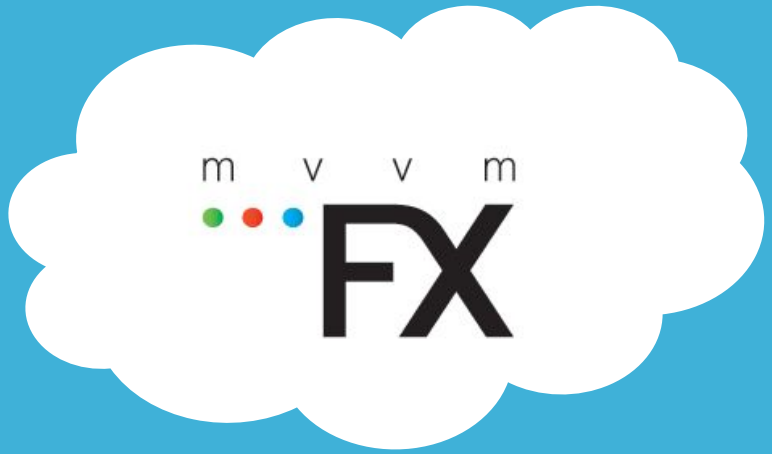
Master View Detail View

Scopes

```
public class MasterDetailScope implements Scope {
   private ObjectProperty<Person> selectedPerson;
   // getter, setter...
}
```

Scopes

```
public class MasterViewModel implements ViewModel {
   @InjectScope
   private MasterDetailScope scope;
   // access the scope to set the selected person
}

public class DetailViewModel implements ViewModel {
   @InjectScope
   private MasterDetailScope scope;
   // access the scope to get the selected person
}
```



www.mvvmfx.de

Source Code, Wiki, Tutorials: **https://github.com/sialcasa/mvvmFX**

Feedback, Bug Reports, Feature Requests erwünscht :-)

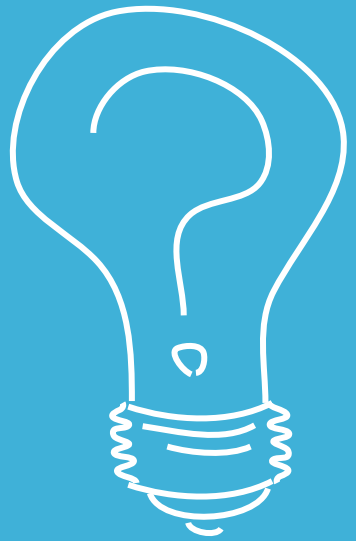
Q & A

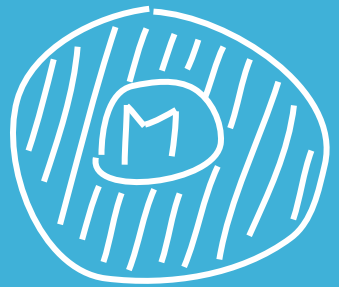
Max Wielsch

max.wielsch@saxsys.de
http://max-wielsch.blogspot.com
@simawiel

Manuel Mauky

manuel.mauky@saxsys.de
http://lestard.eu
@manuel_mauky





MODEL WRAPPER

ModelWrapper

Der ModelWrapper optimiert das Lesen und Schreiben von Modell-Daten.

Negative Example

```
public class ContactFormViewModel implements ViewModel {

   private StringProperty firstname = new SimpleStringProperty();
   private StringProperty lastname = new SimpleStringProperty();
   private StringProperty emailAddress = new SimpleStringProperty();
   private StringProperty phoneNumber = new SimpleStringProperty();

   public StringProperty firstnameProperty() {
      return firstname;
   }
   public StringProperty lastnameProperty() {
      return lastname;
   }
   public StringProperty emailAddressProperty() {
      return emailAddress;
   }
   public StringProperty phoneNumberProperty() {
      return phoneNumber;
   }
}
```

```
public class ContactFormViewModel implements ViewModel {
   // Properties and Property-Getter …

   @Inject
   private Repository repository;
   private person;

   public void showPerson(String id) {
      person = repository.find(id);

      firstname.setValue(person.getFirstName());
      lastname.setValue(person.getLastName());
      emailAddress.setValue(person.getEmailAddress());
      phoneNumber.setValue(person.getPhoneNumber());
   }
   public void save() {
      person.setFirstName(firstname.get());
      person.setLastName(lastname.get());
      person.setEmailAddress(emailAddress.get());
      person.setPhoneNumber(phoneNumber.get());

      repository.persist(person);
   }
}
```

```
public class ContactFormViewModel implements ViewModel {

   private ModelWrapper<Person> modelWrapper = new ModelWrapper<>();
   @Inject
   private Repository repository;

   public void showPerson(String id) {
      Person person = repository.find(id);
      modelWrapper.set(person);
      modelWrapper.reload();
   }
   public void save() {
      modelWrapper.commit();
      repository.persist(modelWrapper.get());
   }

   public StringProperty firstnameProperty() {
      return modelWrapper.field(Person::getFirstName, Person::setFirstName);
   }
   public StringProperty lastnameProperty() {
      return modelWrapper.field(Person::getLastName, Person::setLastName);
   }
   public StringProperty emailAddressProperty() {
      return modelWrapper.field(Person::getEmailAddress, Person::setEmailAddress);
   }
   public StringProperty phoneNumberProperty() {
      return modelWrapper.field(Person::getPhoneNumber, Person::setPhoneNumber);
   }
}
```

ADVANCED RESOURCE BUNDLES

Advanced (Cascading) ResourceBundles

Global ResourceBundle

```
title=”My Cool App”
ok_button=”OK”
cancel_button=”Cancel”
```

View1 ResourceBundle

```
title=”Other Window”
menu_help=”Help”
```

View2 ResourceBundle

```
title=”Settings Dialog”
```

Resources in View1

```
title=”Other Window”
menu_help=”Help”
ok_button=”OK”
cancel_button=”Cancel”
```

Resources in View2

```
title=”Settings Dialog”
ok_button=”OK”
cancel_button=”Cancel”
```

Advanced (Cascading) ResourceBundles

```
ResourceBundle bundle =
          ResourceBundle.getBundle("myapp");

MvvmFX.setGlobalResourceBundle(bundle);
```

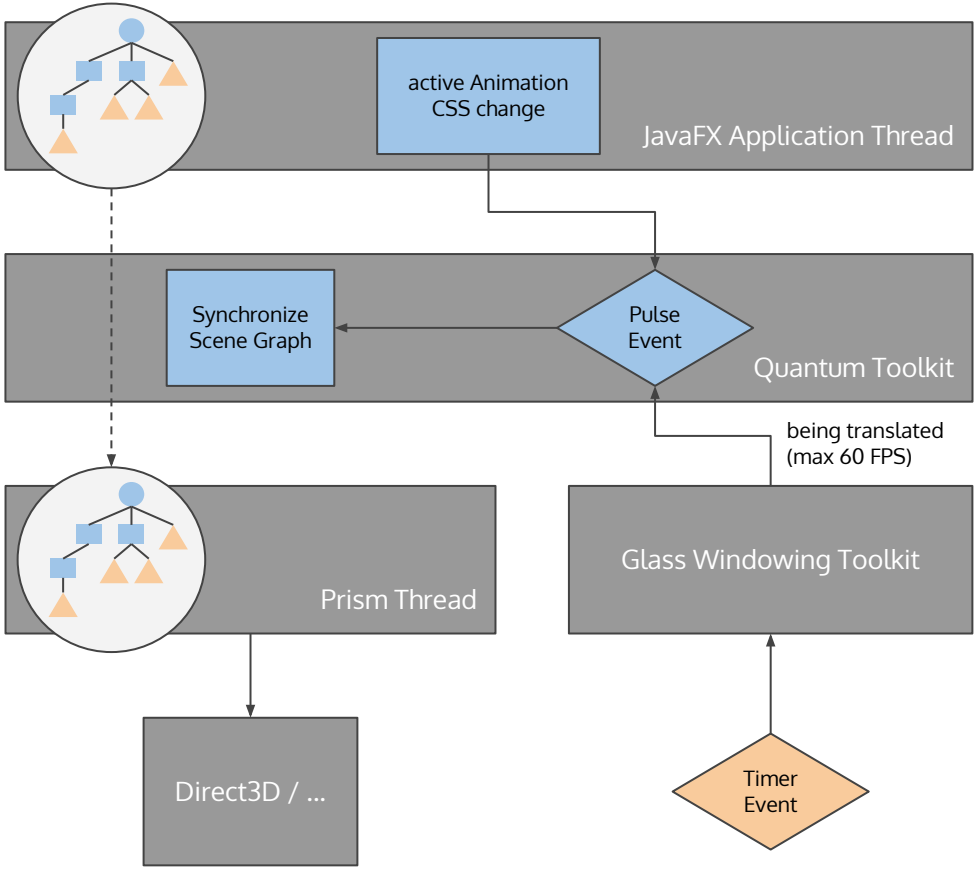
Advanced (Cascading) ResourceBundles

```
FluentViewLoader
    .fxmlView(MyView.class)
    .resourceBundle(bundle)
    .load();
```

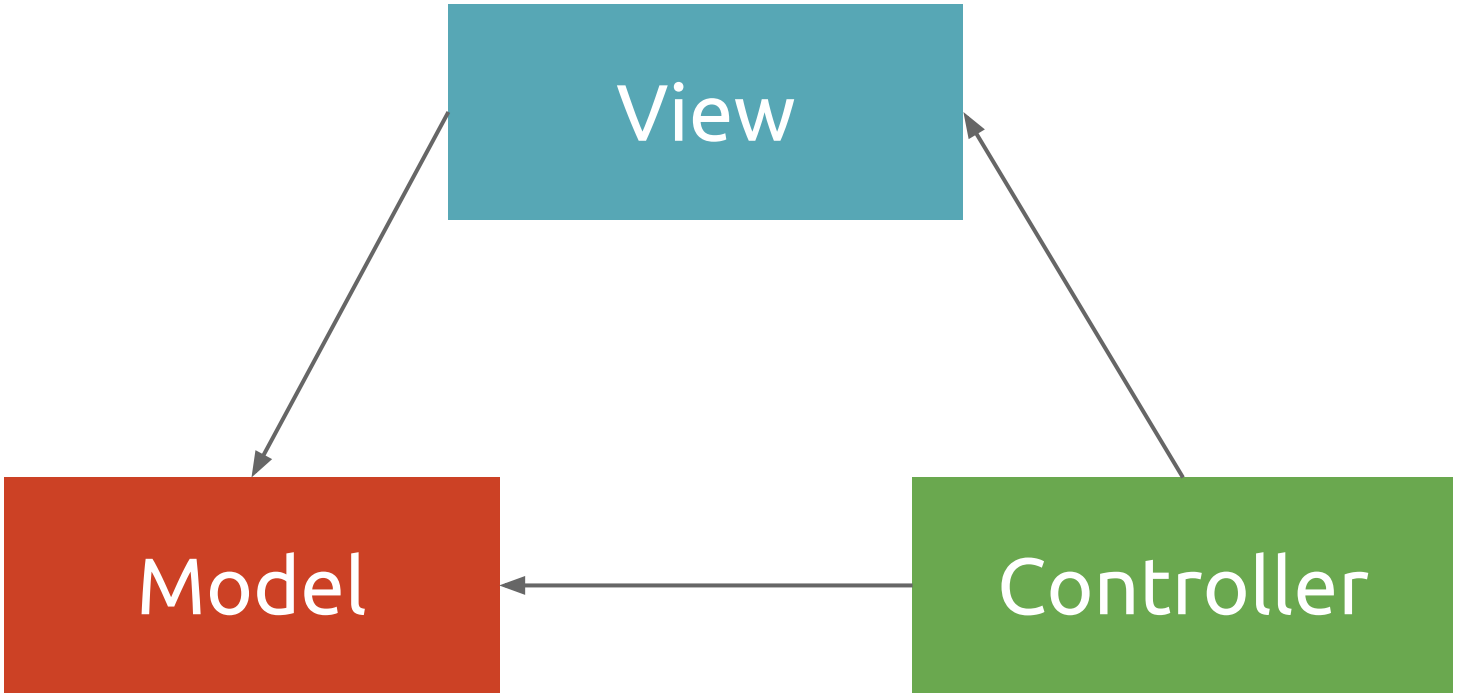
Advanced (Cascading) ResourceBundles

```
<fx:include source="../menu/MenuView.fxml"
        resources="menu"/>
```

Rendering


active Animation
CSS change
JavaFX Application Thread
Synchronize
Scene Graph
Pulse
Event
Quantum Toolkit
being translated
(max 60 FPS)
Prism Thread
Glass Windowing Toolkit
Direct3D / ...
Timer
Event

Model View Controller



Problems

- Kopplung zwischen Controller und View
- Schwer mit Unit-Tests zu testen
- Stattdessen notwendig: Integrationstests mit UI-Automatisierung
  - langsam
  - komplex
  - wartungsintensiv
  - mehr Anforderungen an Build-Infrastruktur (z. B. Jenkins benötigt grafische Oberfläche)
- Vermischung von verschiedenen Aspekten