MV*

GUI Komponenten und Wiederverwendung

Hochschule **Rosenheim**
University of Applied Sciences

Agenda

1. Was sind GUI Komponenten? (20 Minuten)
2. Architektur von GUI Komponenten (20 Minuten)
3. GUI Komponenten mit JavaFX (20 Minuten)

Hochschule **Rosenheim**
University of Applied Sciences

Was sind GUI Komponenten?

- Voll funktionsfähiger und wiederverwendbarer Baustein
- Kann in verschiedenen Kontexten eingesetzt werden
- Basiert selbst auf wiederverwendbaren Bausteinen
- Beispiele
  - Ein einfaches Steuerelemement (z.B. Button)
  - Ein zusammengesetztes Steuerelement (z.B. Datumsauswahl
  - Ein Dialog-Bereich zur Ansicht und Eingabe von Adressdaten
  - Ein kompletter Dialog
  - Eine Abfolge von Dialogen

Hochschule Rosenheim
University of Applied Sciences

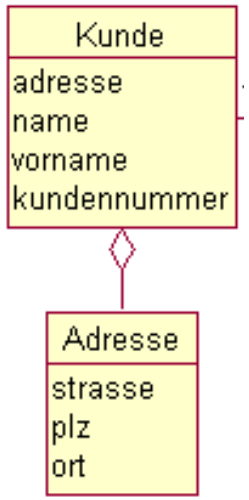
Eigenschaften von GUI Komponenten

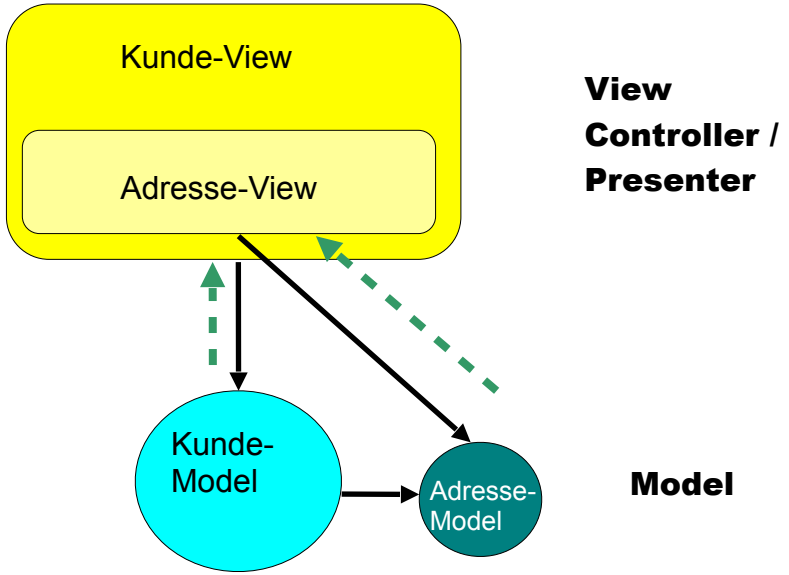
- Mehrfach Instanzierbar
  - Jede Komponente benötigt eigene Daten (Model, Presenter, Controller)
- Eventbasiert
  - Komponenten benachrichtigen Ihre Nutzer mittels Event-Schnittstellen
- Kontextfrei
  - Eine Komponente darf nicht das Vaterelement bzw. den umgebenden Kontext kennen.
- Wiederverwendbar
  - Eine Komponente kann in beliebigen Dialogen eingesetzt werden

Hochschule **Rosenheim**
University of Applied Sciences

MV* bei GUI Komponenten

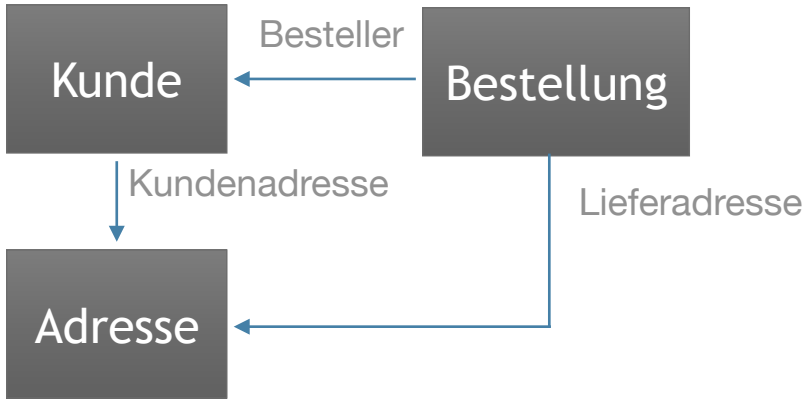
- Bei zusammengesetzten GUI Komponenten sind auch die Modelle zusammengesetzt

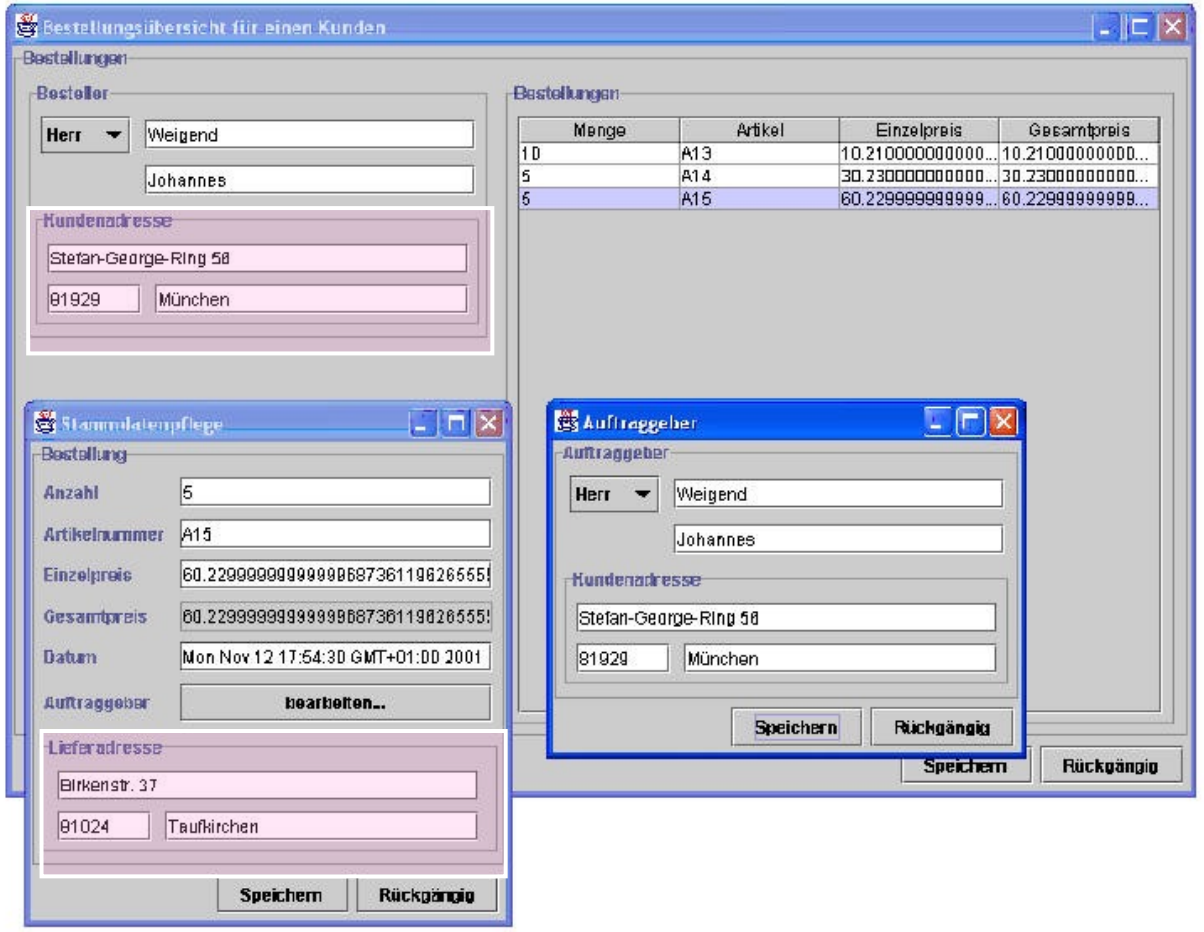




Hochschule Rosenheim
University of Applied Sciences

Wiederverwendung des Kundenbereichs im Kontext einer Bestellung



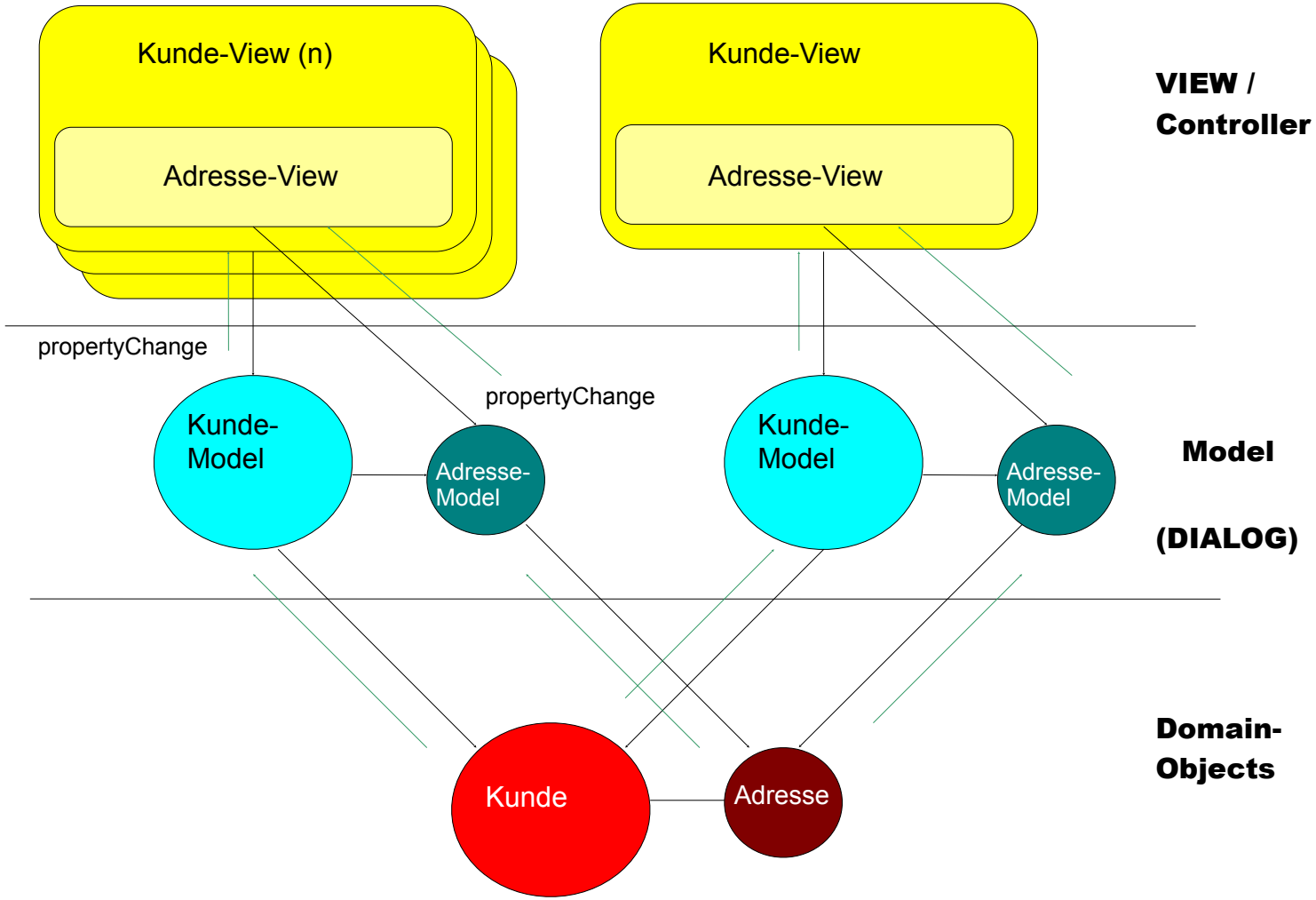


Das Presentation Model Muster

- MV* macht keine Aussagen wie das Model aussieht
- Eine direkte Verwendung von Business Objekten als UI Modelle sollte man absehen
  - UI Modelle bilden die Sicht der Oberfläche ab. Businessmodelle bilden die Sicht des fachlichen Datenmodells ab und können davon abweichen
  - UI Modelle sind meist um technische Eigenschaften angereichert. Diese sind UI spezifisch und werden in der Anwendungslogik nicht benutzt.
  - UI Modelle können zeitweise inkonsistent sein. Businessobjekte niemals.
- Das Presentation Model Muster trennt beide Welten
  - http://martinfowler.com/eaaDev/PresentationModel.html

Hochschule Rosenheim
University of Applied Sciences

Präsentation-Model vs. Domain-Model



Agenda

1. Vergabe der Studienarbeiten (20 Minuten)
2. Architektur von GUI Komponenten (20 Minuten)
3. GUI Komponenten mit JavaFX (20 Minuten)

Hochschule **Rosenheim**
University of Applied Sciences

GUI Komponenten mit JavaFX

- Jede Komponente besteht aus View (FXML), Controller/ Presenter und Model
- Komponenten lassen sich mit JAVA Code und/oder FXML bauen
- FXML
  - Ein View kann einen anderen View inkludieren (Mittels fx:include)
  - Der Kind-Controller kann injiziert werden (nicht umgekehrt!)

Hochschule **Rosenheim**
University of Applied Sciences

<fx:include>

The `<fx:include>` tag creates an object from FXML markup defined in another file. It is used as follows:

```
<fx:include source="filename"/>
```

where *filename* is the name of the FXML file to include. Values that begin with a leading slash character are treated as relative to the classpath. Values with no leading slash are considered relative to the path of the current document.

For example, given the following markup:

```
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<VBox xmlns:fx="http://javafx.com/fxml">
    <children>
        <fx:include source="my_button.fxml"/>
    </children>
</VBox>
```

If `my_button.fxml` contains the following:

```
<?import javafx.scene.control.*?>
<Button text="My Button"/>
```

the resulting scene graph would contain a `VBox` as a root object with a single `Button` as a child node.

Note the use of the "fx" namespace prefix. This is a reserved prefix that defines a number of elements and attributes that are used for internal processing of an FXML source file. It is generally declared on the root element of a FXML document. Other features provided by the "fx" namespace are described in the following sections.

`<fx:include>` also supports attributes for specifying the name of the resource bundle that should be used to localize the included content, as well as the character set used to encode the source file. Resource resolution is discussed in a later section.

Nested Controllers

Controller instances for nested FXML documents loaded via the `<fx:include>` element are mapped directly to member fields of the including controller. This allows a developer to easily access functionality defined by an include (such as a dialog window presented by an application's main window controller). For example, given the following code:

main_window_content.fxml

```
<VBox fx:controller="com.foo.MainController">
   <fx:include fx:id="dialog" source="dialog.fxml"/>
   ...
</VBox>
```

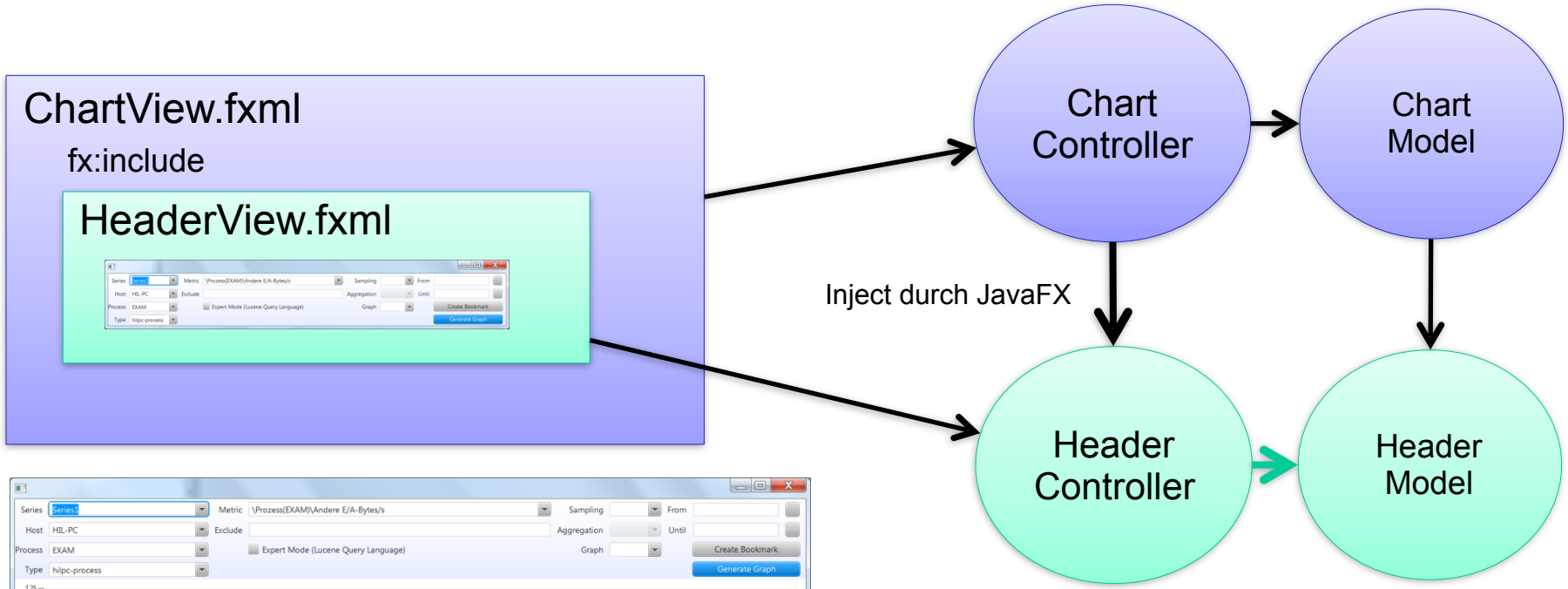
MainController.java

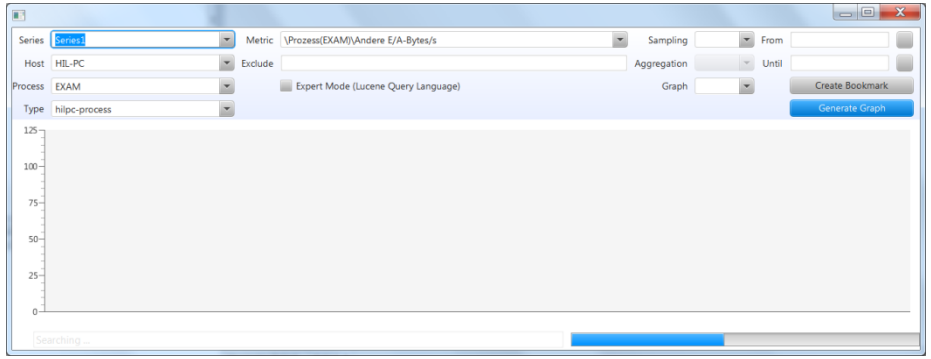
```
public class MainController extends Controller {
    @FXML private Window dialog;
    @FXML private DialogController dialogController;

    ...
}
```

when the controller's `initialize()` method is called, the `dialog` field will contain the root element loaded from the "dialog.fxml" include, and the `dialogController` field will contain the include's controller. The main controller can then invoke methods on the included controller, to populate and show the dialog, for example.

Demo: Schachtelung von Oberflächen





Hochschule Rosenheim
University of Applied Sciences