

# Java

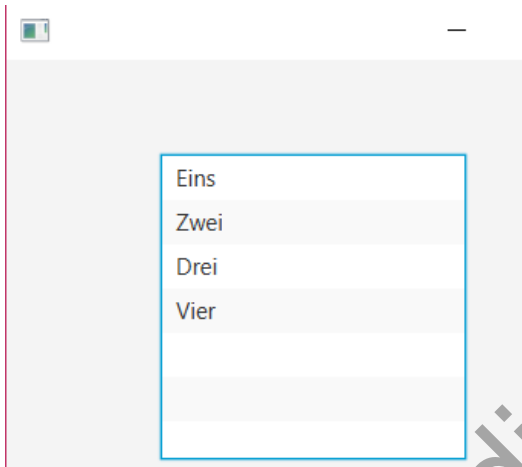
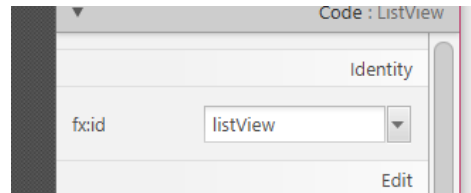
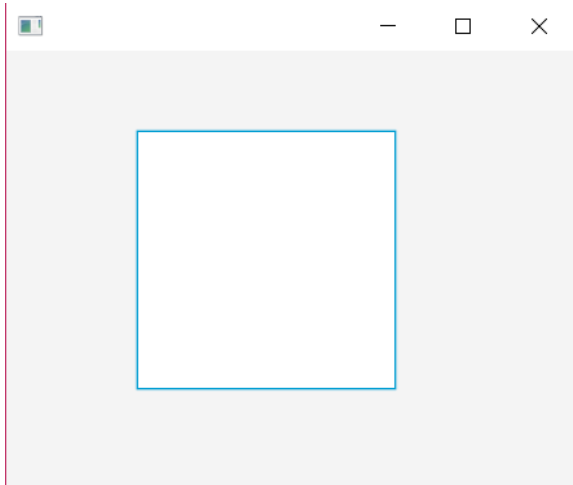
## UI II Controls



# ListView

- Die ListView ist sehr einfach zu verwenden. Sie stellt Daten in einer Liste dar.
- Als Modell verwaltet sie wie viele andere Controls eine ObservableList von Objekten.
  - Diese können wir entweder mit der Methode setItems übergeben, oder wir holen uns eine anfangs leere Liste
  - von der ListView durch Aufruf der Methode getItems und befüllen sie mit unseren Datenobjekten.

# ListView

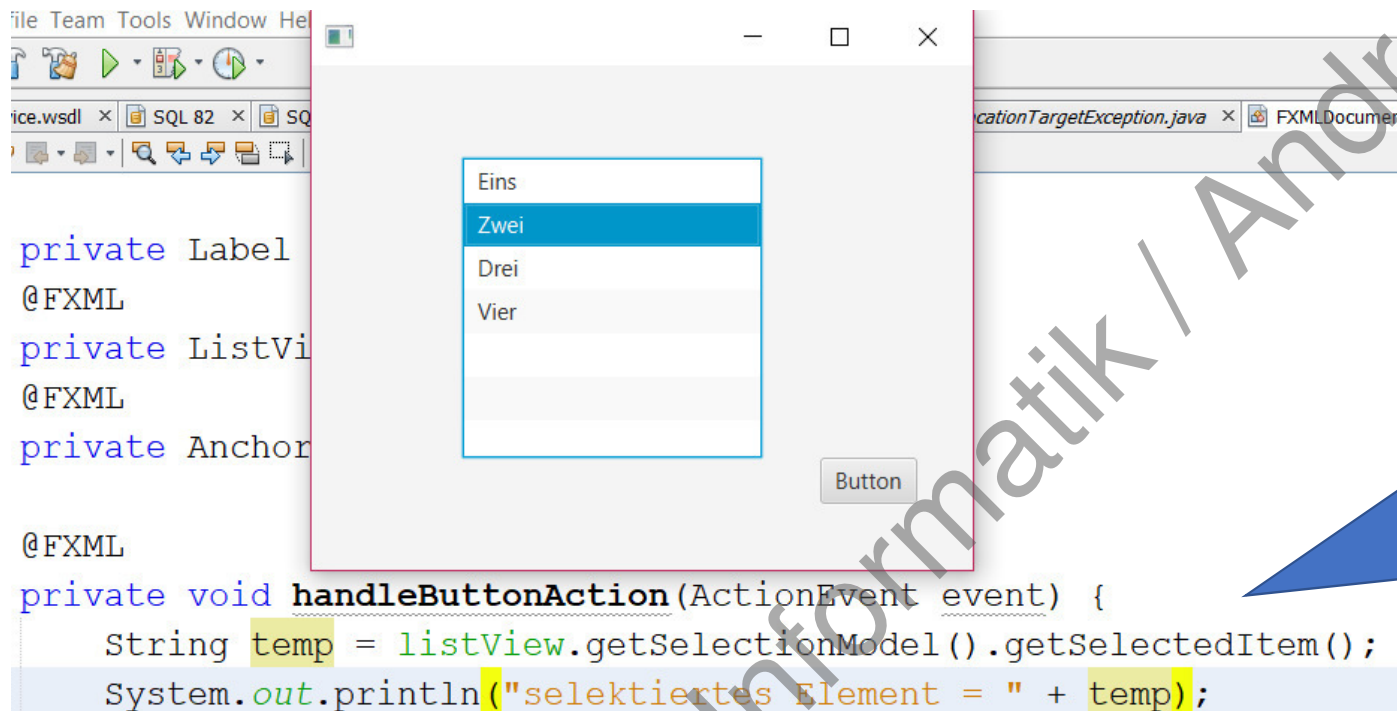


```
@Override  
public void initialize(URL url, ResourceBundle rb) {  
    listView.getItems().addAll("Eins", "Zwei", "Drei", "Vier");  
}
```

von der ListView  
durch Aufruf der  
Methode `getItems`  
und befüllen sie mit  
unseren  
Datenobjekten

Projekt -> Vorzeigen  
(FXMLDocument.fxml;  
FXMLDocumentController)

# ListView

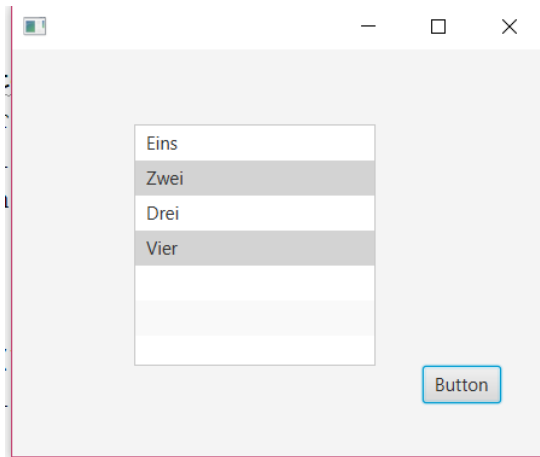


Die selektierten Werte werden von einem SelectionModel verwaltet. Standardmäßig wird hierfür ein MultiSelectionModel verwendet, das sowohl Einzel- als auch Mehrfachauswahl unterstützt. Die Voreinstellung ist die Einzelauswahl.

run-single:

selektiertes Element = Zwei

Projekt -> Vorzeigen  
(FXMLDocument.fxml;  
FXMLDocumentController)



## ListView

```
@FXML
private void handleButtonAction(ActionEvent event) {
    ObservableList<String> list = listView.getSelectionModel().getSelectedItem();
    for(String temp : list) {
        System.out.println(temp);
    }
}

@Override
public void initialize(URL url, ResourceBundle rb) {
    listView.getSelectionModel().setSelectionMode(SelectionMode.MULTIPLE);
    listView.getItems().addAll("Eins", "Zwei", "Drei", "Vier");
}
```

Für eine Mehrfachauswahl  
müssen Sie den Mode  
umstellen:

Zwei  
Vier

Projekt -> Vorzeigen  
(FXMLDocument.fxml;  
FXMLDocumentController)

# ListView

- Die ListView verwendet zur Darstellung der einzelnen Zeilen die JavaFX-Cell-API.
- Die Klasse Cell leitet von Labeled ab, weil in Listen meistens Text dargestellt wird. Deshalb ist auch die ListCell, die in der ListView per Default verwendet wird, so programmiert, dass sie auf dem Item die toString-Methode aufruft und das Resultat mit setText darstellt.
- Gibt die toString-Methode keinen vom User interpretierbaren Text zurück oder wollen wir den Text modifizieren, so können wir eine eigene Subklasse von ListCell erzeugen.

Person -> Franz Geiger  
Person -> Frieda Musterfrau

Identity  
fxid listViewPerson

# ListView

```
@Override  
public void initialize(URL url, ResourceBundle rb) {  
    listView.getSelectionModel().setSelectionMode(SelectionMode.MULTIPLE);  
    listView.getItems().addAll("Eins", "Zwei", "Drei", "Vier");  
    listViewPerson.getItems().addAll(new Person("Franz", "Geiger", "franz.jpg")  
    , new Person("Frieda", "Musterfrau", "frieda.jpg"));  
    listViewPerson.setCellFactory((param) -> {  
        return new PersonCell();  
    });  
}
```

```
public class PersonCell extends ListCell<Person>{  
    // ImageView iv;  
    @Override  
    protected void updateItem(Person item, boolean empty) {  
        super.updateItem(item, empty);  
        if (item != null & !empty) {  
            try {  
                iv = new ImageView(new Image(new FileInputStream(item.getBildName())));  
            } catch (FileNotFoundException ex) {  
                Logger.getLogger(PersonCell.class.getName()).log(Level.SEVERE, null, ex);  
            }  
            setText("Person -> " + item.getVorname() + " " + item.getNachname());  
            setGraphic(iv);  
        }  
    }  
}
```

vorzeigen  
FXMLDocument.fxml  
FXMLDocumentController.java  
PersonCell.java  
Vorzeigen.java

Projekt -> Vorzeigen  
(FXMLDocument.fxml;  
FXMLDocumentController)

Person -> Franz Geiger
Person -> Frieda Musterfrau

Identity	
fcid	listViewPerson

# ListView

```
@Override
public void initialize(URL url, ResourceBundle rb) {
    listView.getSelectionModel().setSelectionMode(SelectionMode.MULTIPLE);
    listView.getItems().addAll("Eins", "Zwei", "Drei", "Vier");
    listViewPerson.getItems().addAll(new Person("Franz", "Geiger", "franz.jpg"),
    , new Person("Frieda", "Musterfrau", "frieda.jpg"));
    listViewPerson.setCellFactory((param) -> {
        return new PersonCell();
    });
}
```

```
public class Person {
    String vorname;
    String nachname;
    String bildName;
```

Ein Objekt welches den Vornamen, den Nachnamen und den Namen eines Bildes im Hauptverzeichnis des Projektes benennt.

vorzeigen  
FXMLDocument.fxml  
FXMLDocumentController.java  
**PersonCell.java**  
Vorzeigen.java

Projekt -> Vorzeigen  
(FXMLDocument.fxml;  
FXMLDocumentController)



Person -> Franz Geiger  
Person -> Frieda Musterfrau

Identity  
fxid listViewPerson

# ListView

```
@Override  
public void initialize(URL url, ResourceBundle rb) {  
    listView.getSelectionModel().setSelectionMode(SelectionMode.MULTIPLE);  
    listView.getItems().addAll("Eins", "Zwei", "Drei", "Vier");  
    listViewPerson.getItems().addAll(new Person("Franz", "Geiger", "franz.jpg"),  
    , new Person("Frieda", "Musterfrau", "frieda.jpg"));  
    listViewPerson.setCellFactory((param) -> {  
        return new PersonCell();  
    });  
}
```

```
public class PersonCell extends ListCell<Person>{  
    // ImageView iv;  
    @Override  
    protected void updateItem(Person item, boolean empty) {  
        super.updateItem(item, empty);  
        if (item != null & !empty) {  
            try {  
                iv = new ImageView(new Image(new FileInputStream(item.getBildName())));  
            } catch (FileNotFoundException ex) {  
                Logger.getLogger(PersonCell.class.getName()).log(Level.SEVERE, null, ex);  
            }  
            setText("Person -> " + item.getVorname() + " " + item.getNachname());  
            setGraphic(iv);  
        }  
    }  
}
```

Wir bauen eine Zelle welche dieses Aussehen beschreibt. Die ListCell muss bei der ListView registriert werden.

vorzeigen  
FXMLDocument.fxml  
FXMLDocumentController.java  
PersonCell.java  
Vorzeigen.java

Projekt -> Vorzeigen  
(FXMLDocument.fxml;  
FXMLDocumentController)

# ListView

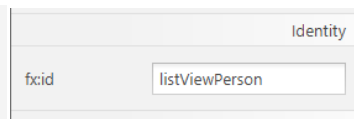
```
@Override
public void initialize(URL url, ResourceBundle rb) {
    listView.getSelectionModel().setSelectionMode(SelectionMode.MULTIPLE);
    listView.getItems().addAll("Eins", "Zwei", "Drei", "Vier");
    listViewPerson.getItems().addAll(new Person("Franz", "Geiger", "franz.jpg"),
    , new Person("Frieda", "Musterfrau", "frieda.jpg"));
    listViewPerson.setCellFactory((param) -> {
        return new PersonCell();
    });
}
```

```
public class PersonCell extends ListCell<Person>{
    // ImageView iv;
    @Override
    protected void updateItem(Person item, boolean empty) {
        super.updateItem(item, empty);
        if (item != null & !empty) {
            try {
                iv = new ImageView(new Image(new FileInputStream(item.getBildName())));
            } catch (FileNotFoundException ex) {
                Logger.getLogger(PersonCell.class.getName()).log(Level.SEVERE, null, ex);
            }
            setText("Person -> " + item.getVorname() + " " + item.getNachname());
            setTextFill(item.getVorname().equals("Franz") ? Color.RED : Color.BLUE);
            //
            setTooltip(new Tooltip("Dies ist die Person " + item.getVorname() + " " + item.getNachname()));
        }
    }
}
```

Es kann auch die Schrift modifiziert werden oder ein Tooltip zugeordnet werden

vorzeigen  
FXMLDocument.fxml  
FXMLDocumentController.java  
PersonCell.java  
Vorzeigen.java

Projekt -> Vorzeigen  
(FXMLDocument.fxml;  
FXMLDocumentController)



# ListView

```
@Override
public void initialize(URL url, ResourceBundle rb) {
    listView.getSelectionModel().setSelectionMode(SelectionMode.MULTIPLE);
    listView.getItems().addAll("Eins", "Zwei", "Drei", "Vier");
    listViewPerson.getItems().addAll(new Person("Franz", "Geiger", "franz.jpg"),
    , new Person("Frieda", "Musterfrau", "frieda.jpg"));
    listViewPerson.setCellFactory((param) -> {
        return new PersonCell();
    });
}
```

```
public class PersonCell extends ListCell<Person> {
    ImageView iv;
    @Override
    protected void updateItem(Person item, boolean empty) {
        super.updateItem(item, empty);
        if (item != null & !empty) {
            try {
                iv = new ImageView(new Image(new FileInputStream(item.getBildName())));
            } catch (FileNotFoundException ex) {
                Logger.getLogger(PersonCell.class.getName()).log(Level.SEVERE, null, ex);
            }
            setText("Person -> " + item.getVorname() + " " + item.getNachname());
            setTextFill(item.getVorname().equals("Franz") ? Color.RED : Color.BLUE);
            setToolTip(new ToolTip("Dies ist die Person " + item.getVorname() + " " + item.getNachname()));
            setGraphic(iv);
        }
    }
}
```

Sie können auch zusätzlich zum Text oder anstatt des Textes einen beliebigen Node zur Darstellung verwenden, etwa eine `ImageView`, der eine `Bitmap` darstellt. Dazu rufen Sie einfach die Methode `setGraphic` der Superklasse `Labeled` auf:

Projekt -> Vorzeigen  
(`FXMLDocument.fxml`;  
`FXMLDocumentController`)


Identity

fcid    listViewPerson

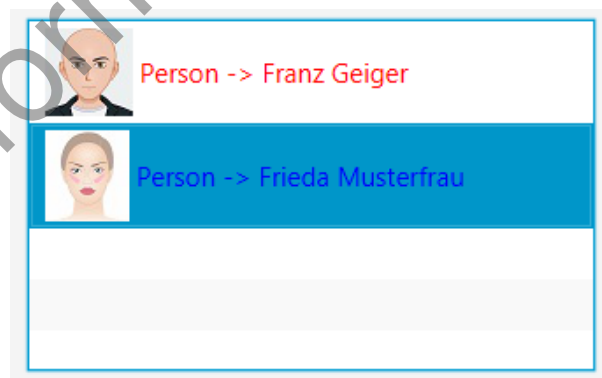
## ListView



Initial state of the ListView. It contains two items: 'Person -> Franz Geiger' and 'Person -> Frieda Musterfrau'. Each item has a small avatar icon to its left.



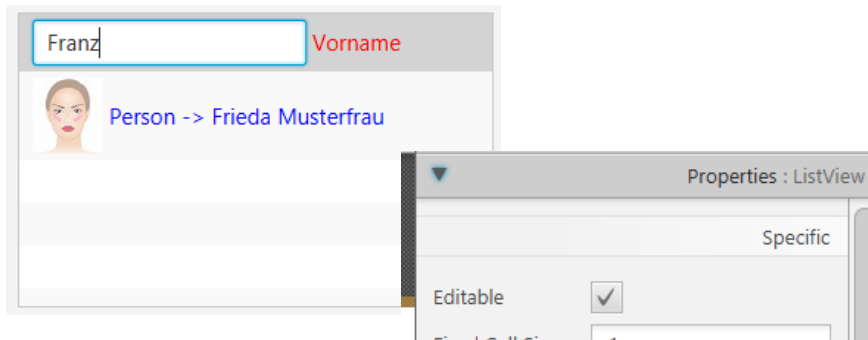
Intermediate state of the ListView. The first item, 'Person -> Franz Geiger', is selected and highlighted. A text field labeled 'Vorname' is visible above the selected item, containing the text 'Franz'.



Final state of the ListView. The first item, 'Person -> Franz Geiger', is selected and highlighted. The text field 'Vorname' is no longer visible.

Wird auf die Zelle geklickt, so wird ein TextFeld angezeigt. Wird die Zelle verlassen, so wird die Eingabe in das Objekt geschrieben.

Projekt -> Vorzeigen  
(FXMLDocument.fxml;  
FXMLDocumentController)



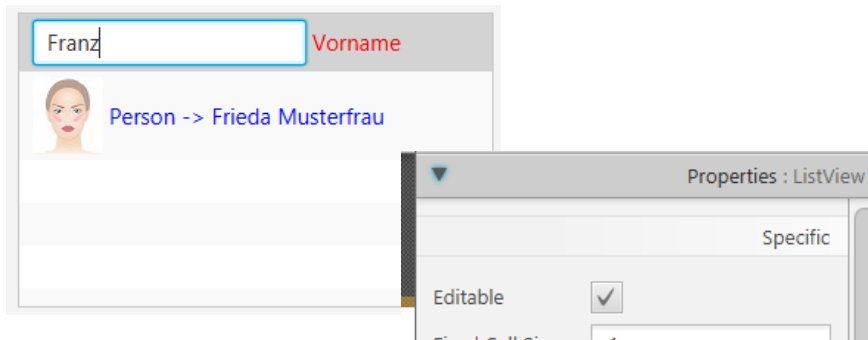
## ListView

```
public class PersonCell extends ListCell<Person> {  
    ImageView iv;  
    @Override  
    protected void updateItem(Person item, boolean empty)  
    Person item;  
    public PersonCell() { ...3 lines }  
    private TextField textFieldVorname;  
    @Override  
    public void startEdit() {  
        super.startEdit(); //To change body of generated  
        setText("Vorname");  
        setGraphic(textFieldVorname);  
    }  
}
```

Auch wenn Sie die ListView editierbar machen möchten, ist das die Aufgabe der Klasse Cell.

Dann müssen Sie die Methode startEdit überschreiben und dort mit setGraphic den Slider als Editor anzeigen:

Projekt -> Vorzeigen  
(FXMLDocument.fxml;  
FXMLDocumentController)



## ListView

```
public class PersonCell extends ListCell<Person> {
    @Override
    public void cancelEdit() {
        super.cancelEdit(); //To change body of generated methods, click here
        String temp = textFieldVorname.getText();
        item.setVorname(temp);
        setText("Person -> " + item.getVorname() + " "
            + item.getNachname());
        setTextFill(item.getVorname().equals("Franz")
            ? Color.RED : Color.BLUE);
        setTooltip(new Tooltip("Dies ist die Person "
            + item.getVorname() + " " + item.getNachname()));
        setGraphic(iv);
    }
}
```

Wird die Zelle verlassen, so wird der eingeegebene Wert in das Personenobjekt gespeichert und die ursprüngliche Darstellung wieder hergestellt.

Projekt -> Vorzeigen  
(FXMLDocument.fxml;  
FXMLDocumentController)

# ChoiceBox

- Die ChoiceBox ist eine einfache Komponente, die dafür gedacht ist, eine Auswahl aus einer Liste von Optionen zu treffen. Eine Mehrfachauswahl wird dabei nicht unterstützt.
- Die ChoiceBox verwaltet dabei eine ObservableList von Items, die wir mit `getItems` erhalten und dann befüllen können:

```
ChoiceBox cb = new ChoiceBox();  
for (int i = 0; i < 20; i++) {  
    cb.getItems().add("Option "+i);  
}
```

# ChoiceBox

@FXML

```
private ChoiceBox<String> choiceBox;
```

@Override

```
public void initialize(URL url, ResourceBundle rb) {  
    for (int i = 0; i < 5; i++) {  
        choiceBox.getItems().add("Object" + i);  
    }  
    choiceBox.setValue("Object3");  
}
```

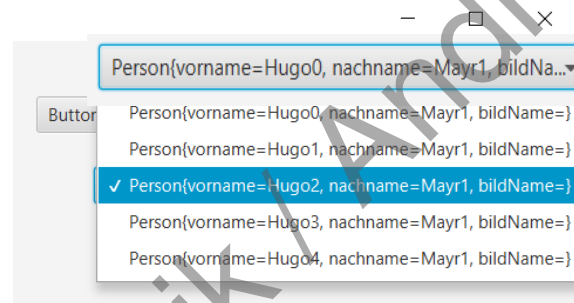
Es erfolgt die Initialisierung über die ObservableList. Mit setValue kann die Vorselektion in der ChoiceBox definiert werden.

Projekt -> Vorzeigen  
(FXMLDocument.fxml;  
FXMLDocumentController)



# ChoiceBox

```
public class Person {  
    String vorname;  
    String nachname;  
    String bildName;  
}
```



Es können natürlich auch komplexe Objekte hinterlegt werden. Es wird zur Darstellung die toString-Methode verwendet.

@FXML

```
private ChoiceBox<Person> choiceBox;
```

@Override

```
public void initialize(URL url, ResourceBundle rb) {  
    for (int i = 0; i < 5; i++) {  
        choiceBox.getItems().add(new Person("Hugo" + i, "Mayr" + i, ""));  
    }  
}
```

Projekt -> Vorzeigen  
(FXMLDocument.fxml;  
FXMLDocumentController)

# ChoiceBox

```
@Override
public void initialize(URL url, ResourceBundle rb) {
    for (int i = 0; i < 5; i++) {
        choiceBox.getItems().add(new Person("Hugo" + i, "Mayr" + i, ""));
    }
    choiceBox.getSelectionModel().selectedIndexProperty().addListener(
        ((observable, oldValue, newValue) -> {
            System.out.println(choiceBox.getItems().get(newValue.intValue()));
            System.out.println(oldValue + choiceBox.getValue());
        }));
}
```

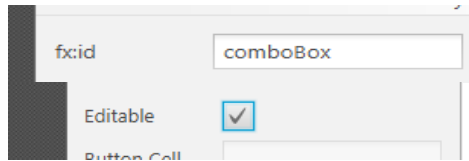
Index des zuvor und des jetzt selektierten Elementes.

Soll die Auswahl später ausgelesen werden, können wir das mit der Methode `getValue()` tun. Wenn Sie dynamisch auf eine Änderung reagieren möchten, so können Sie auch einen Listener auf dem `SelectionMode` der `ChoiceBox` registrieren:

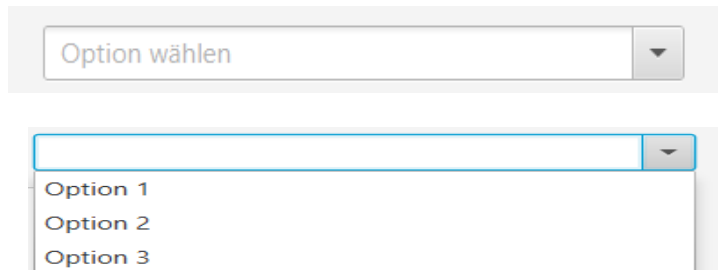
Projekt -> Vorzeigen  
(FXMLDocument.fxml;  
FXMLDocumentController)

# ComboBox

- Die ComboBox verfolgt einen ganz ähnlichen Ansatz wie die ChoiceBox, ist aber deutlich mächtiger und besser konfigurierbar.
- Sie kann auch alles, was eine ChoiceBox kann; deshalb können Sie im vorigen Beispiel die ChoiceBox ganz einfach durch eine ComboBox ersetzen:
- Im Unterschied zur ChoiceBox kann die ComboBox auch editierbar sein. Wir sollten zu diesem Zweck allerdings ein einfacheres Beispiel wählen, denn für die Eingabe einer Person ist die Logik recht komplex, um den eingegebenen String zu validieren und zu parsen, und es ist aufwendig. Hier würde man eher ein eigenes Formular verwenden, um die Eingabe einer neuen Person für den Benutzer komfortabler zu machen. Verwenden wir also ein einfacheres Beispiel mit Strings als Datenobjekten.



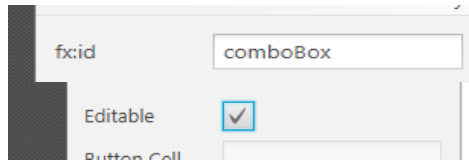
# ComboBox



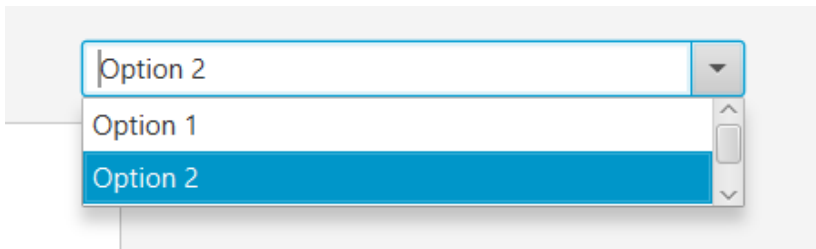
```
@FXML
private ComboBox<String> comboBox;
@FXML
private void handleButtonAction(ActionEvent event) { ...6 li
@Override
public void initialize(URL url, ResourceBundle rb) {
    comboBox.setPromptText("Option wählen");
    comboBox.setEditable(true);
    comboBox.getItems().addAll("Option 1", "Option 2", "Option 3");
}
```

Setzen des Eingabeaufforderungstextes. Es können auch Werte eingegeben werden welche noch nicht in der Liste sind.

Projekt -> Vorzeigen  
(FXMLDocument.fxml;  
FXMLDocumentController)



# ComboBox



```
comboBox.setVisibleRowCount(2);  
comboBox.getSelectionModel().selectedIndexProperty().addListener(  
    ((observable, oldValue, newValue) -> {  
        System.out.println(comboBox.getItems().get(newValue.intValue()));  
        System.out.println("'" + comboBox.getValue());  
    }));
```

Sie können festlegen, wie viele Zeilen angezeigt werden sollen.

Wird die maximale Zeilenzahl überschritten, erscheint automatisch ein ScrollBar.

Sonst funktioniert die ComboBox wie die ChoiceBox

Projekt -> Vorzeigen  
(FXMLDocument.fxml;  
FXMLDocumentController)

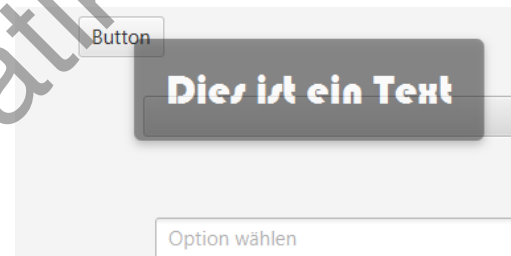
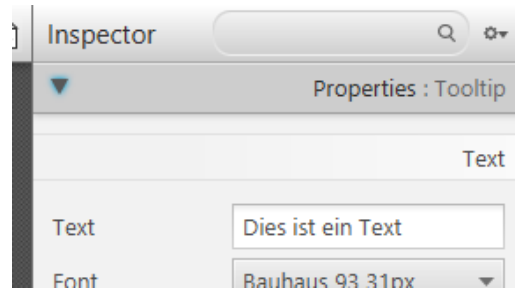
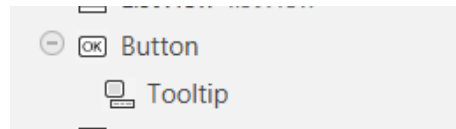
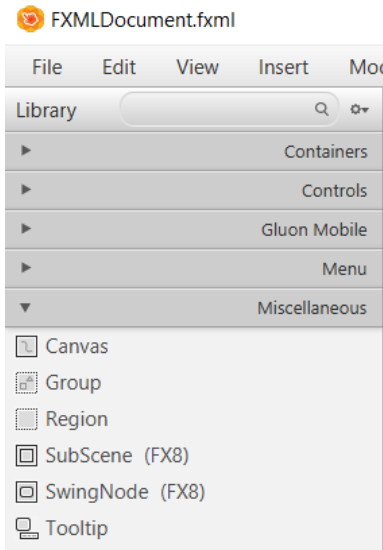
# Tooltip

- Ein Tooltip ist ein Popup, welches zum Anzeigen als Information bezgl. des Elementes dient.
- Mit seiner Hilfe werden typischerweise Hinweise für die Verwendung eines Controls gegeben. Deshalb hat die Basisklasse Control auch die Methode `setTooltip` und wir können auf allen Controls einen Tooltip registrieren, der bei Bedarf angezeigt wird. Das geht ganz einfach über die Methode `setTooltip`.
- Für Nodes, die keine `setTooltip`-Methode haben, können wir die statische Methode `install` der Klasse `Tooltip` verwenden, der wir den Node und die `Tooltip`-Instanz übergeben.



```
Button btn = new Button();
btn.setText("Dummy Button");
btn.setTooltip(new Tooltip("Dieser Button tut nichts.));
StackPane root = new StackPane();
root.getChildren().add(btn);
Tooltip tooltip = new Tooltip("Ein ToolTip für die StackPane");
Tooltip.install(root, tooltip);
```

# Tooltip im Scene Builder



Projekt -> Vorzeigen  
(FXMLDocument.fxml;  
FXMLDocumentController)

# Are there any Questions?



Exercise:

B\_Uebung

C\_Uebung

Medizin-Informatik / Andreas Pilger