

# GUI Übung 6 – GUI Komponenten

[illegible]

## Beschreibung

Ziel der Übung ist es, eine Oberfläche bestehend aus Eingabeformular und Tabelle zu erzeugen. Die Oberfläche aus Übung 4 soll dazu als Komponente verwendet und eingebettet werden.

## Aufgabe 1 – Komponenten

Erstellen Sie die oben abgebildete grafische Oberfläche nach dem MVP-Pattern. Verwenden Sie dazu die Komponente „LocationForm“ aus der letzten Übung und erstellen Sie eine neue Komponente „LocationsMain“ die das Eingabefeld und eine Tabelle enthält. Gehen Sie dabei wie folgt vor:

- Verwenden Sie das Ergebnis aus Übung 4. Erstellen Sie ein neues Unterpaket „locationform“ und verschieben Sie Anteile des Formulars dorthin. Passen Sie ggf. die Pfade an.
- Erstellen Sie eine neue FXML Datei (`LocationsMain.fxml`) mit Controller (`LocationsMainPresenter.java`).
- Fügen Sie in `LocationsMain.fxml` (mit Scene Builder) eine vorerst leere Tabelle ein.
- Verwenden Sie in `LocationsMain.fxml` die `LocationForm` mittels `fx:include`.
- Die Gesamtansicht ist jetzt hierarchisch geschachtelt (Vorlesung 2). Verwenden Sie einen geeigneten Container für das Layout.
- Laden Sie in Ihrer Applikation die Datei `LocationsMain.fxml` und starten Sie die Anwendung.

## Aufgabe 2 – Injection & Kommunikation

Reagieren Sie in der Elternkomponenten `LocationsMain` auf Aktionen in der Kindkomponente `LocationForm`.

Gehen Sie dabei wie folgt vor:

- Erweitern Sie den `LocationsMain-Controller/Presenter` um eine Methode „`saveLocation(LocationModel location)`“.
- Injizieren Sie sich den `LocationFormPresenter` in den `LocationsMainPresenter`.  
Hinweis: Wie in der Vorlesung besprochen (fx:id nicht vergessen!).
- Holen Sie sich im `LocationsMainPresenter` den Zugriff auf den Button „Save“ des `LocationFormPresenter`.
- Registrieren Sie einen Event Handler für den Button und führen Sie die entsprechende Methode (`saveLocation`) im `LocationsMainPresenter` aus.

## Aufgabe 3 – Listener

Diskutieren Sie, warum es nicht ideal ist in Main auf einen Button der Form zuzugreifen.

Implementieren Sie eine Lösungsalternative.

## Aufgabe 4 – Tabelle (optional)

Halten Sie in `LocationsMain` eine Liste von `Locations` (Veranstaltungen) und fügen Sie beim Save-Location ein neues Element hinzu. Zeigen sie die `Locations` (Veranstaltungen) mit den wichtigsten Attributen in der Tabelle an.

### TableView

- Über `<TableView>` und `<columns>` können Sie in der `LocationsMain.xml` Spalten definieren
- Um das automatische Befüllen (über Binding) zu ermöglichen liefert JavaFX eine „`cellValueFactory`“ und eine „`PropertyValueFactory`“ in welcher Sie das entsprechende Attribut ausgeben können

- Hierfür benötigen Sie außerdem den Import:

```
<?import javafx.scene.control.cell.PropertyValueFactory?>
```

Beispiel:

```
<columns>
  <TableColumn prefWidth="75.0" text="Id" >
    <cellValueFactory><PropertyValueFactory
property="id"/></cellValueFactory>
  </TableColumn>
</columns>
```

Hier wird das Attribut "id" ausgegeben.

## Data Binding und Listeners

- Legen Sie im LocationsMainPresenter eine SimpleListProperty an. Diese hält die gespeicherten Locations und kann an die Tabelle übergeben werden.
- Setzen sie das SimpleListProperty auf der Tabelle über `.itemsProperty().bind`
- Anstatt den Klick auf den Save Button im LocationsMainPresenter zu verwalten könnten Sie beispielsweise das Listener Pattern im LocationsFormPresenter implementieren.
  - Legen Sie ein interface SaveListener an, das eine Methode `onSave(LocationModel location)` bietet
  - Legen Sie im LocationsFormPresenter eine Liste an SaveListener an.
  - In der onSave Methode des LocationsFormPresenter können Sie nun auf allen gesetzten SaveListener die Methode onSave aufrufen und das aktuelle LocationModel übergeben.
  - Im LocationsMainPresenter können Sie nun dem LocationsFormPresenter einen neuen SaveListener hinzufügen. Dieser nimmt das gespeicherte LocationModel Objekt und fügt es dem SimpleListProperty hinzu.
  - Somit sollte sich die Tabelle automatisch updaten.