JAVA 2

JavaFX 02

Teme

- MVC
- *Controller*
- Povezivanje *View-Controller*
- *@FXML* anotacija
- Povezivanje metoda sa GUI elementima
- *Observable Collections*
- *Collection Utilities*
- *TableView*
- *Menu*

MVC

- JavaFX temelji se na MVC (*Model View Controller*) arhitekturi
- odvajanje slojeva aplikacije koji predstavljaju podatke (*Model*), grafičko sučelje (*View*) i poslovnu logiku aplikacije (*Controller*)
- model podataka se obično sastoji od klasa koje predstavljaju domenske podatke
- grafičko sučelje se temelji na FXML datoteci koja definira izgled grafičkog sučelja ili klasama pomoću kojih se dizajnira grafičko sučelje kroz naredbe u Java kodu

MVC

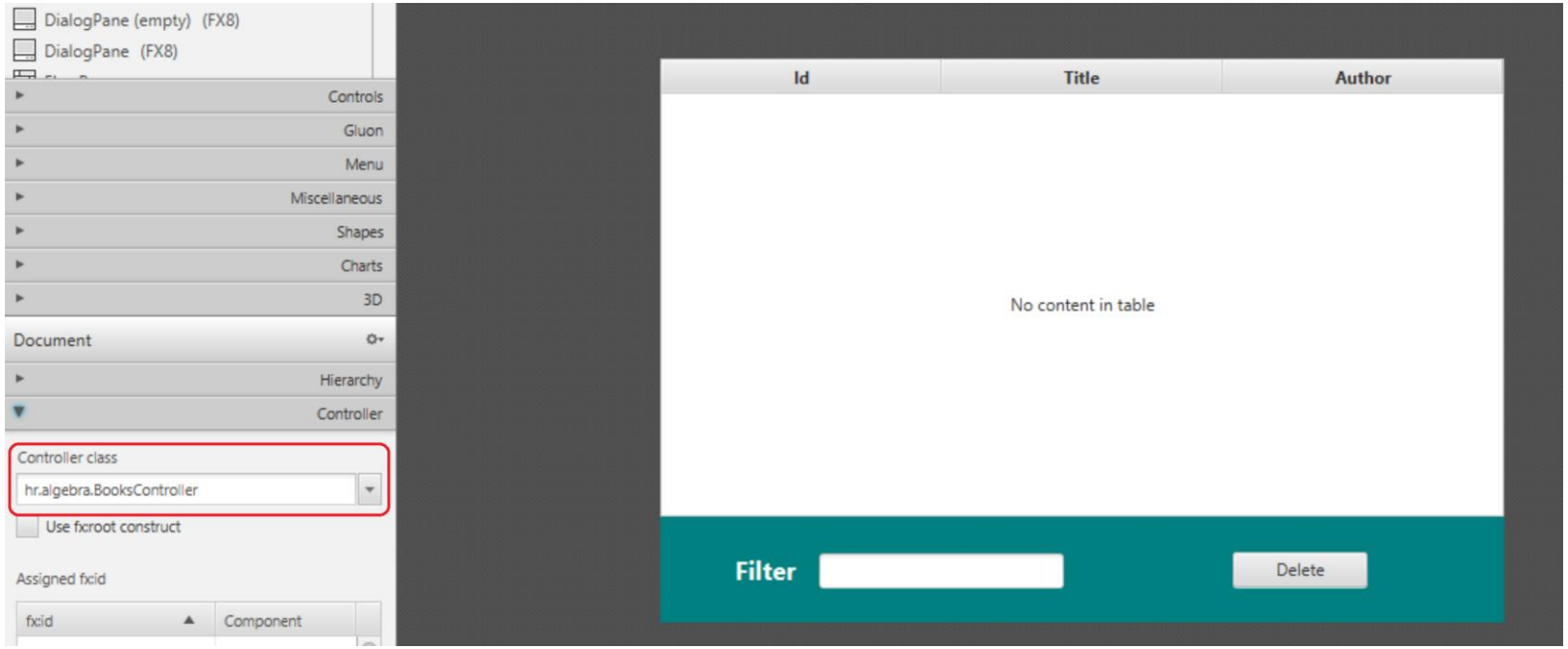
- poslovna logika – implementirana unutar klasa koje povezuju elemente grafičkog sučelja (npr. *button*) s metodama koje se moraju izvoditi tijekom interakcije korisnika s tim elementima (npr. *button click*)
- za ostvarivanje veze između grafičkog sučelja i poslovne logike, elementima grafičkog sučelja je potrebno dodijeliti jedinstvene identifikatore *fx:id*

Controller

- omogućava povezivanje elemenata grafičkog sučelja s Java programskim kodom koji predstavlja poslovnu logiku
- sadrži varijable koje po tipu i nazivu odgovaraju elementima grafičkog sučelja, kako bi se mogle koristiti njihove značajke i atributi
- naziv klase *Controller* mora biti povezan s grafičkim sučeljem korištenjem FXML konfiguracije
- preporuča se da svaki ekran osim pripadajuće FXML datoteke ima i svoju klasu tipa *Controller*
- svaka Controller klasa treba imati svoju *initialize(URL, ResourceBundle)* metodu označenu s *@FXML* anotacijom koja sadrži svu logiku potrebnu za inicijalizaciju osnovnih postavki elemenata na grafičkom sučelju

Povezivanje View-Controller

- *root* ekrana se povezuje s pripadajućim *Controllerom* korištenjem opcije u *Scene Builderu* ili unutar FXML datoteke:

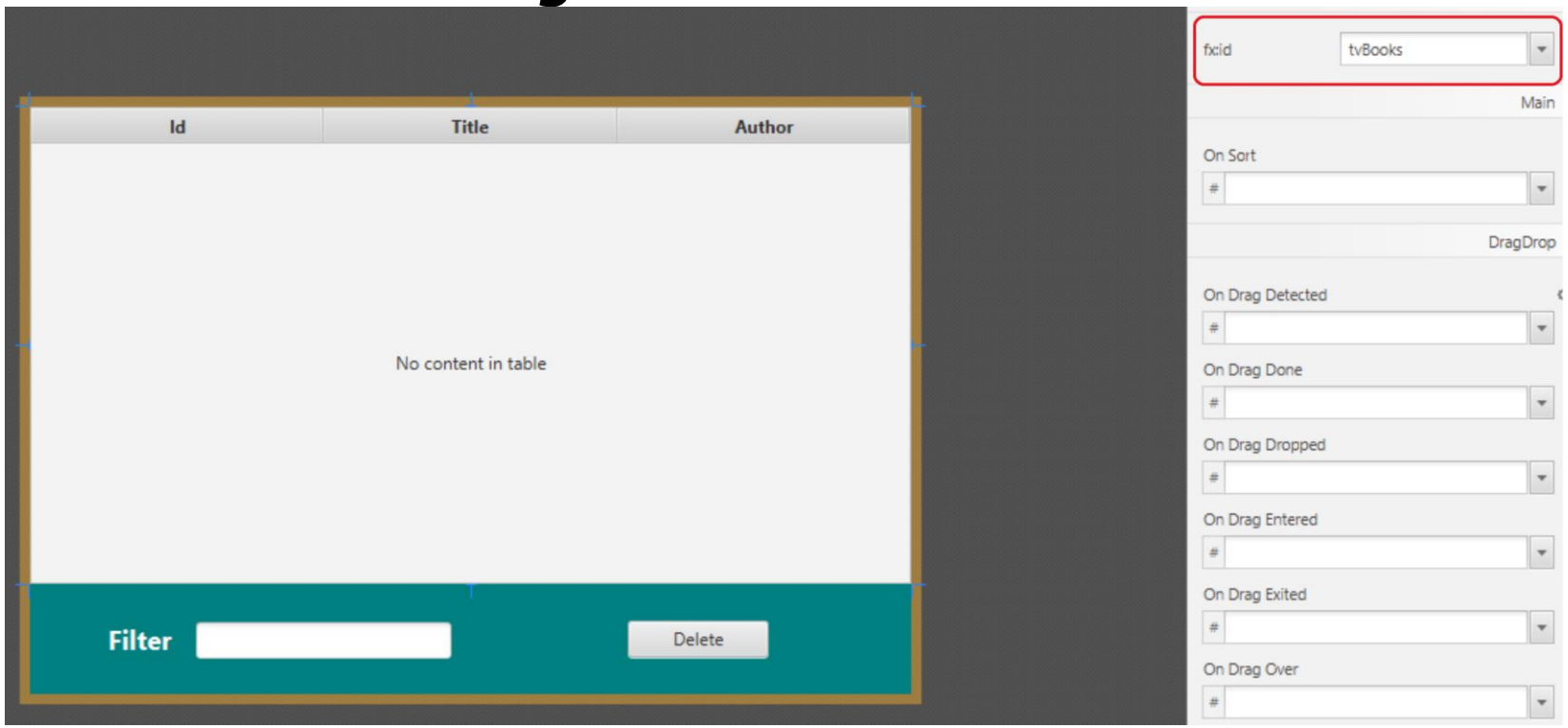



```
<BorderPane fx:controller="{className} ">
```

@FXML anotacija

- služi za označavanje elemenata u *Controlleru* koji se koriste unutar FXML datoteke
- može se koristiti kod varijabli koje predstavljaju grafičke elemente i metoda koje se pozivaju prilikom interakcije s nekim od grafičkih elemenata
- na primjer, ako se elementu *TableView* unutar FXML datoteke postavi za *fx:id* identifikator *tvBooks*, taj element je moguće povezati s varijablom unutar *Controller* klase korištenjem anotacije *@FXML* (nazivi se moraju podudarati)

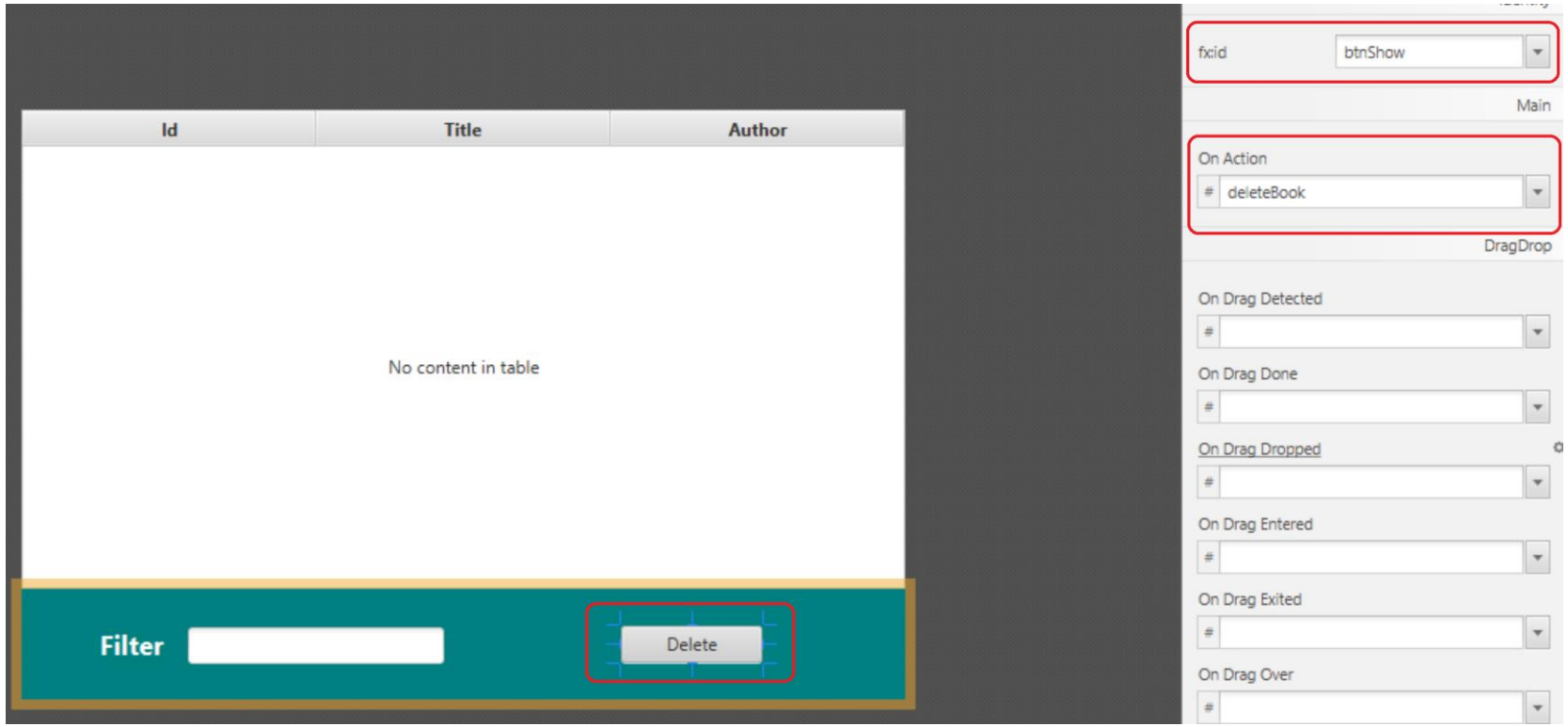
@FXML anotacija




```
public class BooksController implements Initializable {
    @FXML
    private TableView<Book> tvBooks;
}
```

Povezivanje metoda s GUI elementima

- element grafičkog sučelja povezuje s odgovarajućom metodom unutar *Controllera*



Povezivanje metoda s GUI elementima

- *Controller* mora sadržavati metodu koja sadrži logiku za obradu događaja

```
@FXML
private void deleteBook() {
    int selectedIndex = tvBooks.getSelectionModel().getSelectedIndex();
    if (selectedIndex != -1) {
        Alert alert = new Alert(AlertType.CONFIRMATION);
        alert.setTitle("Confirmation");
        alert.setHeaderText("Delete book?");
        alert.setContentText("Really tired of '" + books.get(selectedIndex) + "'?");
        if (alert.showAndWait().get() == ButtonType.OK){
            books.remove(selectedIndex);
        }
    }
}
```

Observable Collections

- ako je potrebno prikazivati kolekciju podataka na grafičkom sučelju, JavaFX podržava *Observable* tipove kolekcija koje omogućavaju izravno praćenje promjena na grafičkom sučelju – *Observable Pattern*
- ***ObservableList*** – lista koja omogućava praćenje promjena na ekranu korištenjem *Listenera*
- ***ObservableMap*** – mapa koja omogućava praćenje promjena na ekranu korištenjem *Listenera*
- ***ListChangeListener*** i ***MapChangeListener*** – implementacije *Listener* klasa koje služe za primanje notifikacija koje generiraju objekti klasa *ObservableList* i *ObservableMap*

Collection Utilities

- ***FXCollections*** – sadrži istovjetne statičke metode onima u klasi *java.util.Collections*
- ***ListChangeListener.Change*** – instance koje predstavljaju promjene nad *ObservableList* kolekcijom
- ***MapChangeListener.Change*** - instance koje predstavljaju promjene nad *ObservableMap* kolekcijom

Collection Utilities

- Objekt klase ***ObservableList*** moguće je kreirati iz *java.util.List* zbirke na sljedeći način:

```
ObservableList<Book> books
        = FXCollections.observableArrayList(BooksRepository.getBooks());
```

- Ako je tu zbirku potrebno povezati s *TableView* komponentom, to je moguće obaviti na sljedeći način:

```
tvBooks.setItems(books);
```

TableView

- za tablično prikazivanje podataka
- *TableColumn* - prikazivanje podataka u pojedinim stupcima
- model – korištenje *{Type}Property* koji omogućuju *binding (unidirectional, bidirectional) i observable pattern*
- podržava ugrađena sortiranja odabirom kolona
- ako koristimo *FilteredList*, mjesto *ObservableList* ručno moramo obaviti sortiranje

```
FilteredList<Book> filteredBooks = books.filtered(b ->
        b.getTitle()
                .toLowerCase()
                .startsWith(tfFilter.getText().trim().toLowerCase()));
SortedList<Book> sortedBooks = new SortedList<>(filteredBooks);
tvBooks.setItems(sortedBooks);
sortedBooks.comparatorProperty().bind(tvBooks.comparatorProperty());
```

TableView

- *bindanje {Type}Property* stupca tablice s modelom
- moguće *editiranje* kolona

```
tcAuthor.setCellValueFactory(new PropertyValueFactory<>("author"));
tcAuthor.setCellFactory(TextFieldTableCell.forTableColumn());
tcAuthor.setOnEditCommit(new EventHandler<TableColumn.CellEditEvent<Book, String>>() {
    @Override
    public void handle(TableColumn.CellEditEvent<Book, String> event) {
        int selectedIndex = tvBooks.getSelectionModel().getSelectedIndex();
        books.get(selectedIndex).setAuthor(event.getNewValue());
    }
});
```

Menu

- izbornik za odabir opcija na grafičkom sučelju sastoji se od *MenuBar* elementa s pripadajućim *Menu* elementima koji sadrže *MenuItem* elemente
- *MenuItem* element – povezuje se s akcijom u *Controlleru,* jednako kao i drugi elementi sučelja

```
@FXML
private void exitApp(ActionEvent event) {
    Platform.exit();
}
```

Demo

- Project

Izvor:http://www.jnhsolutions.com/contact-us/request-a-demo/

Hvala na pažnji!