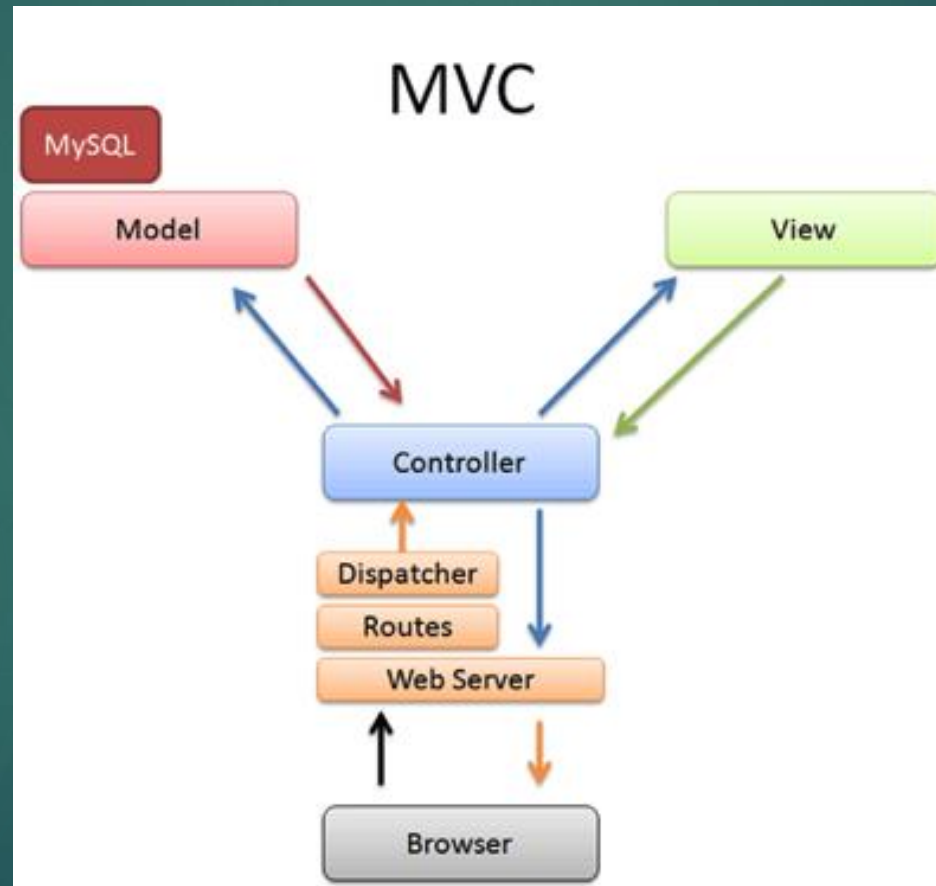


MVC and JavaFX

MVC – Model, View, Controller

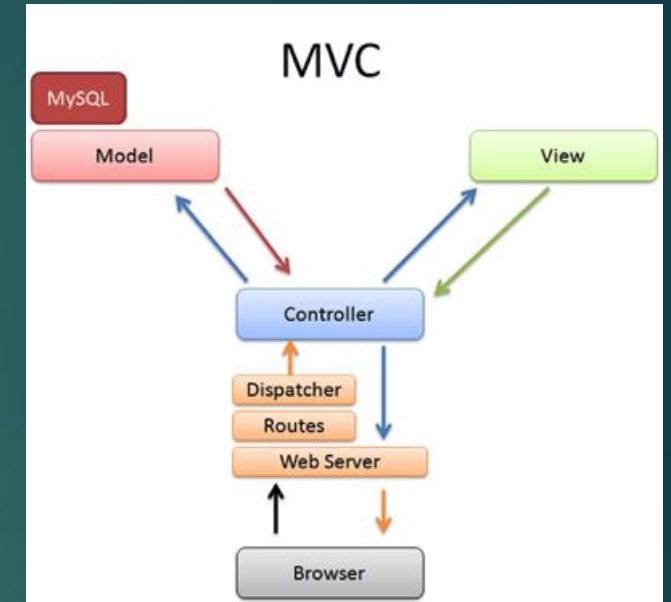
- ▶ Controller – receives requests
 - ▶ Requests could involve
 - ▶ Request to change some information
 - ▶ Retrieve some information
- ▶ Model – interacts with data storage (such as a database, file, etc.)
 - ▶ Handles all data storage, manipulation, and retrieval
- ▶ View – defines what is display to the user

Typical Internet MVC architecture



MVC is like a 3-layer Architecture

- ▶ View -> Presentation layer
- ▶ Model -> Data Layer
- ▶ Controller receives requests
 - ▶ Gets relevant Data from the Model
 - ▶ Sends Data to View for display
- ▶ Business Logic may be handled in the model or Controller,
depending on implementation
 - ▶ In JavaFX, business logic tends to be in Model



MVC Design

- ▶ Modularity
 - ▶ We have cohesion by separating out handling requests, managing the data, and generating the user output
- ▶ Modifiable
 - ▶ We can add more pages, model, etc. over time
- ▶ MVC not unique to web applications
 - ▶ Android has XML layouts for the View, Activities for Controllers, SQLite or Firebase for Model
 - ▶ Desktop Applications *can* be implement in MVC way (JavaFX)

MVC Popularity

- ▶ There are a number of popular Java MVC frameworks for building applications



- ▶ Many other popular web app frameworks are MVC



* - node.js doesn't *have* to be MVC, but that is a very common use case

MVC as a common internet architecture

- ▶ Controllers can handle requests as they come in
 - ▶ Supports asynchronous web-applications
- ▶ Client sends requests to web-server
 - ▶ Web-server sends request to controller
 - ▶ Controller then interprets the request
 - ▶ Tells the model what model transactions to perform
 - ▶ Tells the view what view to generate
 - ▶ Models performs the transactions like database queries
 - ▶ View generates an HTML web page for the user to see

MVC in JavaFX

- ▶ Define our interface using .FXML
 - ▶ The .FXML file describes the layout of the interface
- ▶ Our Controller is **tightly coupled** with the .FXML file:
 - ▶ The base Pane of the .FXML file explicitly states its controller Class

```
<VBox alignment="CENTER" spacing="1.0" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1"
    fx:controller="edu.virginia.cs.javafx.NumberController">
```

- ▶ Instance variables of the controller are tied to individual widgets via fx:id

```
<TextField id="numberBox" fx:id="entryTextArea" ... >
```

```
@FXML
private TextField entryTextArea;
```

- ▶ @FXML tag means “this variable is tied to something in view”

Event Handling with FXML

- ▶ onKeyPressed defines the EventHandler function

```
<TextField id="numberBox" fx:id="entryTextArea" onKeyPressed="#onTextEntryEnter"
           prefHeight="37.0" prefWidth="100.0" text="" />
```

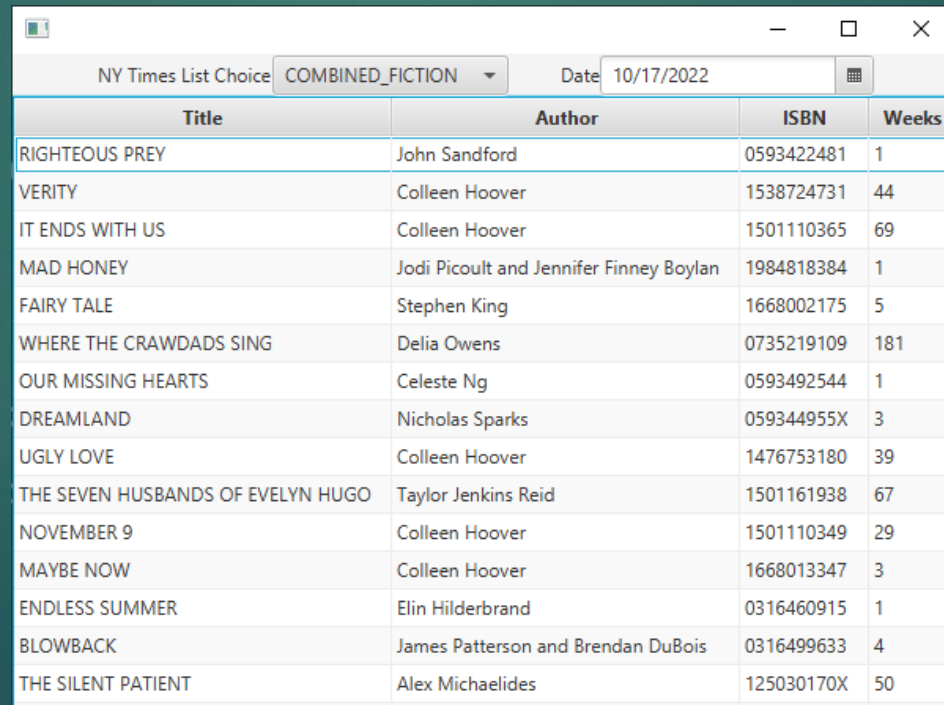
```
@FXML
protected void onTextEntryEnter(KeyEvent event) {
    if (event.getCode().equals(KeyCode.ENTER)) {
        try {
            entryTextArea.setText(entryTextArea.getText().strip());
            errorLabel.setText("");
            int value = Integer.parseInt(entryTextArea.getText());
            model.addNumber(value);
            updateAllLabelsForCurrentNumber();
        } catch (NumberFormatException e) {
            errorLabel.setText("Error: Enter a valid int");
        }
    }
}
```

Designing a Controller in JavaFX

- ▶ A Controller should **ONLY** handle:
 - ▶ Handling actions from the View
 - ▶ Updating the View
 - ▶ Call functions in the model when needed
- ▶ No meaningful or complicated business logic should be handled in the FXML Controller class
 - ▶ This is because Controller is already tightly coupled with the UI

MVC vs Three Layer

- ▶ Consider our Three Layer Architecture for NYTimes Best Seller
 - ▶ <https://github.com/sde-coursepack/ThreeLayerBooks/tree/gui>



The screenshot shows a web application window titled "NY Times List Choice". It features a dropdown menu set to "COMBINED_FICTION" and a date input field set to "10/17/2022". Below these controls is a table with four columns: Title, Author, ISBN, and Weeks. The table lists 17 books, with "THE SILENT PATIENT" by Alex Michaelides at the bottom, having a value of 50 in the Weeks column.

Title	Author	ISBN	Weeks
RIGHTEOUS PREY	John Sandford	0593422481	1
VERITY	Colleen Hoover	1538724731	44
IT ENDS WITH US	Colleen Hoover	1501110365	69
MAD HONEY	Jodi Picoult and Jennifer Finney Boylan	1984818384	1
FAIRY TALE	Stephen King	1668002175	5
WHERE THE CRAWDADS SING	Delia Owens	0735219109	181
OUR MISSING HEARTS	Celeste Ng	0593492544	1
DREAMLAND	Nicholas Sparks	059344955X	3
UGLY LOVE	Colleen Hoover	1476753180	39
THE SEVEN HUSBANDS OF EVELYN HUGO	Taylor Jenkins Reid	1501161938	67
NOVEMBER 9	Colleen Hoover	1501110349	29
MAYBE NOW	Colleen Hoover	1668013347	3
ENDLESS SUMMER	Elin Hilderbrand	0316460915	1
BLOWBACK	James Patterson and Brendan DuBois	0316499633	4
THE SILENT PATIENT	Alex Michaelides	125030170X	50

MVC vs Three Layer

- ▶ In this case, the View:
 - ▶ nytimes.fxml
- ▶ And the Controller
 - ▶ GUIController.java
- ▶ Act as the presentation layer
 - ▶ Tightly coupled, Controller updates what is displayed
 - ▶ View receives users interactions
- ▶ The Model encompasses both of the:
 - ▶ Business Logic
 - ▶ Data Layer

```
<?import javafx.scene.control.*?>
<?import javafx.scene.control.cell.*?>
<?import javafx.scene.layout.*?>

<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0"
  <center>
    <TableView fx:id="tableView" prefHeight="200.0" prefWidth="200.0" BorderPane.alignment="CENTER">
      <columns>
        <TableColumn text="Title"><cellValueFactory><PropertyValueFactory property="title" /></cellValueFactory>
        <TableColumn text="Author"><cellValueFactory><PropertyValueFactory property="authorName" /></cellValueFactory>
        <TableColumn text="ISBN"><cellValueFactory><PropertyValueFactory property="isbn" /></cellValueFactory>
        <TableColumn text="Weeks"><cellValueFactory><PropertyValueFactory property="weeksOnList" /></cellValueFactory>
      </columns>
    </TableView>
  </center>
  <top>
    <HBox alignment="CENTER" prefHeight="27.0" prefWidth="600.0" BorderPane.alignment="CENTER">
      <children>
        <Label text="NY Times List Choice" />
        <ChoiceBox fx:id="listSelector" onAction="#updateList" prefWidth="150.0" />
        <Separator opacity="0.0" prefHeight="0.0" prefWidth="33.0" />
        <Label text="Date" />
        <DatePicker fx:id="datePicker" onAction="#updateDate" />
      </children>
    </HBox>
  </top>
</BorderPane>
```

MVC in Java FX setup

- ▶ Application “main” class
- ▶ Loads the initial .fxml interface and handles launching the application

```
public class GUIApplication extends Application {  
    private BestSellersService service;  
  
    @Override  
    public void start(Stage stage) throws Exception {  
        URL view = GUIApplication.class.getResource(  
            "/edu.virginia.cs.threelayer.presentation.nytimeslist.fxml");  
        FXMLLoader fxmlLoader = new FXMLLoader(view);  
        Scene scene = new Scene(fxmlLoader.load());  
        stage.setScene(scene);  
        stage.show();  
    }  
  
    public static void main(String[] args) {  
        launch(args);  
    }  
}
```

MVC in Java Controller

- ▶ Able to connect variables to fxml elements via the fx:id tag in fxml and the @FXML annotation in Java

```
public class GUIController {  
  
    private BestSellersService service;  
  
    @FXML  
    private TableView<Book> tableView;  
    @FXML  
    private ChoiceBox<ListName> listSelector;  
    @FXML  
    private DatePicker datePicker;  
  
    public void initialize() {  
        service = new BestSellersService();  
        initDatePicker();  
        initListSelector();  
        updateTable();  
    }  
}
```

FXML to Controller

- ▶ fx:controller identifies the Controller class
- ▶ fx:id identifies UI elements

```
public class GUIController {  
  
    private BestSellersService service;  
  
    @FXML  
    private TableView<Book> tableView;  
    @FXML  
    private ChoiceBox<ListName> listSelector;  
    @FXML  
    private DatePicker datePicker;  
  
    public void initialize() {  
        service = new BestSellersService();  
        initDatePicker();  
        initListSelector();  
        updateTable();  
    }  
}
```

```
<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml" fx:controller="edu.virginia.cs.cmc.edu.layer1.presentation.GUIController">  
    <center>  
        <TableView fx:id="tableView" prefHeight="200.0" prefWidth="200.0" BorderPane.alignment="CENTER">  
            <columns>  
                <TableColumn text="Title"><cellValueFactory><PropertyValueFactory property="title" /></cellValueFactory></TableColumn>  
                <TableColumn text="Author"><cellValueFactory><PropertyValueFactory property="author" /></cellValueFactory></TableColumn>  
                <TableColumn text="ISBN"><cellValueFactory><PropertyValueFactory property="isbn" /></cellValueFactory></TableColumn>  
                <TableColumn text="Weeks"><cellValueFactory><PropertyValueFactory property="weeks" /></cellValueFactory></TableColumn>  
            </columns>  
        </TableView>  
    </center>  
    <top>  
        <HBox alignment="CENTER" prefHeight="27.0" prefWidth="600.0" BorderPane.alignment="CENTER">  
            <children>  
                <Label text="NY Times List Choice" />  
                <ChoiceBox fx:id="listSelector" onAction="#updateList" prefWidth="150.0" />  
                <Separator opacity="0.0" prefHeight="0.0" prefWidth="33.0" />  
                <Label text="Date" />  
                <DatePicker fx:id="datePicker" onAction="#updateDate" />  
            </children>  
        </HBox>  
    </top>  
</BorderPane>
```


Modifying Fields with MVC

- Frequently, a method like **setValue** is used to set a starting value:

```
private void initDatePicker() {  
    LocalDate today = LocalDate.now();  
    datePicker.setValue(today);  
}
```

- ListView has **setItems** and **clear**, which we use to update the view.

```
private void updateTable() {  
    BestSellersList bookList = service.getHistoricBestSellerList(listSelector.getValue(), Date.from(  
        datePicker.getValue().atStartOfDay(ZoneId.systemDefault()).toInstant()));  
    ObservableList<Book> obsList = FXCollections.observableList(bookList.getAllBooksInOrderOfRank());  
    tableView.getItems().clear();  
    tableView.getItems().addAll(obsList);  
}
```