Конспект второй пары по JavaFX: Продвинутые контролы, динамическое создание UI и работа с файлами

Подсказка лектору

Ноябрь 2025

1. Общие принципы урока

Цель: Продолжить изучение JavaFX: динамическое создание UI в коде (без FXML), дополнительные контролы (Spinner, TextArea, Canvas, TabPane, DatePicker, ColorPicker, ComboBox), обработка событий, стилизация, диалоги файлов (FileChooser), многопоточность (Platform.runLater) и базовая работа с файлами. Фокус на практике: 30--40% теории, 60--70% демо в IntelliJ.

Аудитория: Студенты, знакомые с базовым JavaFX и FXML.

Формат: Краткая теория, демонстрации в IntelliJ. Каждый шаг — показ на экране с комментариями в коде. Использовать встроенные комментарии для ключевых понятий.

Распределение времени: 80 мин.

2. Динамическое создание UI в коде (0--15 мин)

0--5 мин: Введение в динамическое создание.
JavaFX позволяет создавать UI программно, без FXML. Это полезно для динамических элементов (например, генерируемых на основе данных). Сравнить с FXML: код более гибкий, но менее визуальный.

5--15 мин: Примеры создания и стилизации.
Демо: создать `Label` динамически и добавить обработчик события.

```
import javafx.scene.control.Label;
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;
import javafx.scene.layout.Background;
import javafx.scene.layout.BackgroundFill;
import javafx.scene.layout.CornerRadii;
import javafx.geometry.Insets;

// Вметоде  start иликонтроллере
Label result = new Label("Результат");
result.setOnMouseClicked(event -> {
    System.out.println("Клик␣по␣лейблу!");
});
result.setStyle("-fx-background-color:␣aqua;");
```

```
// Альтернативастилячерез   Background
result.setBackground(new Background(
    new BackgroundFill(
        new Color(0.9, 0.9, 0.9, 1),
        CornerRadii.EMPTY,
        Insets.EMPTY
    )
));
```

Добавить в layout, например, VBox.

3. Дополнительные контролы: Spinner, TextArea, Canvas (15--35 мин)

15--20 мин: Spinner и TextArea.

Spinner — для ввода чисел с кнопками увеличения/уменьшения. TextArea — многострочный текст.

```
import javafx.scene.control.Spinner;
import javafx.scene.control.TextArea;

// Spinner дляцелыхчиселот    0 до 100
Spinner<Integer> spinner = new Spinner<>(0, 100, 50);
spinner.setOnAction(event -> {
    System.out.println("Значение:␣" + spinner.getValue());
});

// TextArea
TextArea textArea = new TextArea("Введите␣текст...");
textArea.setPrefRowCount(5);
textArea.setWrapText(true);
```

20--35 мин: Canvas и GraphicsContext.

Canvas — для рисования графики. GraphicsContext — контекст для операций.

```
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.paint.Color;

// Создаем Canvas
Canvas canvas = new Canvas(200, 200);
GraphicsContext gc = canvas.getGraphicsContext2D();
gc.setFill(Color.BLACK);
gc.setStroke(Color.BLACK);
gc.strokeLine(20, 20, 40, 40);
gc.fillOval(0, 0, 20, 20);

// Дополнительныйпример : линияикруг
gc.strokeLine(10, 10, 100, 100);
gc.fillOval(50, 50, 30, 30);
```

Демо: добавить в Scene и показать рисование.

4. Продвинутые layouts и контролы: TabPane, DatePic ColorPicker, ComboBox (35--55 мин)

35--45 мин: TabPane.

TabPane — для вкладок. Каждая Tab содержит контент.

```
import javafx.scene.control.TabPane;
import javafx.scene.control.Tab;
import javafx.scene.layout.VBox;
import javafx.scene.layout.GridPane;
import javafx.scene.control.Label;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.control.ComboBox;
import javafx.scene.control.TableView;
import javafx.scene.control.TableColumn;

// Создаем TabPane
TabPane tabPane = new TabPane();

// Перваявкладка
Tab tab1 = new Tab("Основная");
tab1.setClosable(false);
VBox tab1Content = new VBox(10);
tab1Content.getChildren().addAll(
    new Label("Это␣основная␣вкладка"),
    new Button("Кнопка␣1"),
    new TextField("Текстовое␣поле")
);
tab1.setContent(tab1Content);

// Втораявкладка
Tab tab2 = new Tab("Настройки");
tab2.setClosable(false);
GridPane tab2Content = new GridPane();
tab2Content.setVgap(10);
tab2Content.setHgap(10);
tab2Content.add(new Label("Настройка␣1:"), 0, 0);
tab2Content.add(new TextField("Значение"), 1, 0);
tab2Content.add(new Label("Настройка␣2:"), 0, 1);
tab2Content.add(new ComboBox<String>(), 1, 1);
tab2.setContent(tab2Content);

// Третьявкладкастаблицей
Tab tab3 = new Tab("Данные");
TableView<String> tableView = new TableView<>();
TableColumn<String, String> column = new TableColumn<>("Данные");
tableView.getColumns().add(column);
tab3.setContent(tableView);

// Добавляемвкладки
tabPane.getTabs().addAll(tab1, tab2, tab3);
```

45--50 мин: DatePicker и ColorPicker.

`DatePicker` — выбор даты. `ColorPicker` — выбор цвета.

FXML-пример для DatePicker и ColorPicker:

```
<?import javafx.scene.control.DatePicker?>
<?import javafx.scene.control.ColorPicker?>

<DatePicker fx:id="datePicker" onAction="#onDateChanged"/>
<ColorPicker fx:id="colorPicker" value="#FF0000" onAction="#
   onColorChanged"/>

<DatePicker GridPane.rowIndex="5" GridPane.columnIndex="0">
    <value>
        <LocalDate fx:factory="now"/>
    </value>
</DatePicker>
```

В коде (для задания начального значения в контроллере, реализующем `Initializable`):

```
import javafx.scene.control.DatePicker;
import javafx.scene.control.ColorPicker;

// DatePicker
DatePicker datePicker = new DatePicker();
datePicker.setOnAction(event -> {
    System.out.println("Выбрана␣дата:␣" + datePicker.getValue());
});

// ColorPicker
ColorPicker colorPicker = new ColorPicker(Color.RED);
colorPicker.setOnAction(event -> {
    System.out.println("Выбран␣цвет:␣" + colorPicker.getValue());
});
```

50--55 мин: ComboBox.

`ComboBox` — выпадающий список.

FXML-пример:

```
<?import javafx.scene.control.ComboBox?>
<?import javafx.collections.FXCollections?>

<ComboBox fx:id="comboBox" promptText="Выберите..." onAction="#
   onComboBoxChanged">
    <items>
        <FXCollections fx:factory="observableArrayList">
            <String fx:value="Вариант␣1"/>
            <String fx:value="Вариант␣2"/>
            <String fx:value="Вариант␣3"/>
        </FXCollections>
    </items>
</ComboBox>
```

В коде:

```
import javafx.scene.control.ComboBox;
```

```
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;

// Создаем ComboBox
ComboBox<String> comboBox = new ComboBox<>();
comboBox.setPromptText("Выберите...");
ObservableList<String> options = FXCollections.observableArrayList(
    "Вариант␣1", "Вариант␣2", "Вариант␣3"
);
comboBox.setItems(options);
comboBox.setOnAction(event -> {
    System.out.println("Выбрано:␣" + comboBox.getValue());
});
```

5. Диалоги и работа с файлами (55--70 мин)

55--65 мин: FileChooser.

`FileChooser` — диалог выбора файла.

```
import javafx.stage.FileChooser;
import javafx.stage.Window;
import java.io.File;

// Вметоде , например, вобработчикекнопки
FileChooser fileChooser = new FileChooser();
fileChooser.getExtensionFilters().add(
    new FileChooser.ExtensionFilter("Image␣Files", "*.png", "*.jpg", "*.
       gif")
);
File selectedFile = fileChooser.showOpenDialog(number1.getScene().
   getWindow()); // number1 —любой  Node всцене
if (selectedFile != null && selectedFile.isFile()) {
    String path = selectedFile.getPath();
    System.out.println("Выбран␣файл:␣" + path);
}
```

Демо: открыть файл и вывести путь.

65--70 мин: Чтение файлов с BufferedReader.

Использовать `java.nio` для чтения.

```
import java.io.BufferedReader;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;

// Примерчтенияфайла
try (BufferedReader reader = Files.newBufferedReader(Path.of("file.txt"))
   ) {
    String line;
    while ((line = reader.readLine()) != null) {
        System.out.println(line);
    }
} catch (IOException e) {
```

```
    e.printStackTrace();
}
```

6. Многопоточность в JavaFX (70--80 мин)

70--80 мин: Platform.runLater.

JavaFX UI обновляется только в главном потоке. Для других потоков использовать `Platform.runLater`.

```
import javafx.application.Platform;

// Вдополнительномпотоке
Platform.runLater(() -> {
    // Обновление UI, например:
    resultLabel.setText("Обновлено из другого потока");
});
```

Демо: симулировать фоновую задачу (`Thread`) и обновить UI.