24. JavaFX (현대적 GUI 프레임워크)

Stage, Scene, Node 개념

JavaFX는 GUI를 구성하는 **계층적 구조**를 가지고 있다.
전체 구조는 이렇게 생각하면 된다:

```
Stage (무대)
 └ Scene (장면)
     └ Node (구성 요소들: 버튼, 텍스트, 레이아웃 등)
```

---

1 `Stage`: 윈도우(창) 그 자체

- `Stage`는 화면에 나타나는 **최상위 컨테이너**, 즉 **창(Window)** 역할
- JavaFX 애플리케이션은 기본적으로 하나의 **Primary Stage**를 가지고 시작됨
- 추가 Stage를 생성해서 여러 창을 띄울 수도 있음

```
@Override
public void start(Stage primaryStage) {
    primaryStage.setTitle("JavaFX 데모");
    ...
    primaryStage.show();
}
```

주요 메서드:

- `setTitle(String)` : 창 제목 설정
- `setScene(Scene)` : Scene 연결
- `show()` : 창 보이기
- `close()` : 창 닫기

---

2 `Scene`: 무대 위의 한 장면

- `Scene`은 `Stage`에 올라가는 **하나의 장면** 또는 **UI 트리의 루트**
- UI 컴포넌트들이 **루트 노드로부터 계층적으로** 배치되는 단위
- 하나의 `Stage`에는 하나의 `Scene`만 설정 가능하지만, 필요에 따라 교체 가능

```
Scene scene = new Scene(rootNode, 400, 300);
primaryStage.setScene(scene);
```

- `rootNode`: 보통 `Pane`, `VBox`, `HBox`, `GridPane` 같은 레이아웃 노드
- `Scene` 크기 설정 가능: `new Scene(root, width, height)`

3️⃣ `Node`: 실제 UI 요소

- `Node`는 **화면에 표시되는 모든 요소의 공통 부모 클래스**
- 버튼, 텍스트, 이미지, 레이아웃, 도형 등 모든 시각적 요소가 `Node`의 하위 클래스임

주요 하위 클래스

종류	클래스 예시
컨트롤(Control)	`Button`, `TextField`, `CheckBox` 등
레이아웃(Layout)	`VBox`, `HBox`, `StackPane`, `GridPane` 등
도형(Shape)	`Circle`, `Rectangle`, `Line` 등
미디어	`MediaView`, `ImageView` 등
텍스트	`Label`, `Text`

예시 구조

```
VBox root = new VBox();              // Node
Button btn = new Button("클릭해!"); // Node
root.getChildren().add(btn);       // Node 트리 구성

Scene scene = new Scene(root, 300, 200); // Scene 생성
stage.setScene(scene);                   // Stage에 부착
stage.show();
```

🧠 요약 구조

```
Stage
 └── Scene
      └── Node
           ├── Layout (VBox, HBox 등)
           │     └── Node (Button, Label 등)
           └── Shape, Media, Text ...
```

✅ 핵심 요약

개념	역할
Stage	하나의 윈도우 창
Scene	창 안의 장면(화면 레이아웃)
Node	버튼, 텍스트, 이미지 등 화면 요소들

💡 JavaFX는 이 구조를 기반으로 **FXML**, **CSS**, **이벤트 핸들링**과도 자연스럽게 연동된다.

FXML 기반 설계

FXML은 JavaFX에서 **UI 설계를 XML로 표현**할 수 있도록 해주는 기술이다. 이를 통해 **디자이너와 개발자의 역할을 분리**하고, 코드보다 선언적으로 UI를 구성할 수 있다.

✅ FXML이란?

- **FXML (FXML Markup Language)**은 JavaFX UI를 정의하는 **XML 기반 언어**
- HTML처럼 계층적으로 UI 요소를 구성함
- `.fxml` 파일에 UI를 작성하고, Java 코드에서 이를 불러와 동작을 정의함

1️⃣ FXML 기본 예시

`MainView.fxml`

```
<?xml version="1.0" encoding="UTF-8"?>

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

<VBox xmlns="http://javafx.com/javafx"
      xmlns:fx="http://javafx.com/fxml"
      fx:controller="com.example.MainController"
      spacing="10" alignment="CENTER" prefWidth="300" prefHeight="200">

    <Label text="이름을 입력하세요:" />
    <TextField fx:id="nameField" />
    <Button text="인사하기" onAction="#handleGreet"/>
</VBox>
```

2️⃣ 연결된 Controller 클래스

`MainController.java`

```
package com.example;

import javafx.fxml.FXML;
import javafx.scene.control.*;

public class MainController {

    @FXML
    private TextField nameField;

    @FXML
```

```
    private void handleGreet() {
        String name = nameField.getText();
        System.out.println("안녕하세요, " + name + "!");
    }
}
```

3️⃣ Application에서 FXML 로드

```
@Override
public void start(Stage primaryStage) throws Exception {
    Parent root = FXMLLoader.load(getClass().getResource("MainView.fxml"));
    Scene scene = new Scene(root);
    primaryStage.setTitle("FXML 예제");
    primaryStage.setScene(scene);
    primaryStage.show();
}
```

🧩 구조 요약

요소	설명
`VBox`, `HBox` 등	UI의 레이아웃 루트 노드
`fx:id`	컨트롤러 클래스에서 접근할 변수 이름
`onAction`	버튼 클릭 시 호출할 메서드 이름
`fx:controller`	연결된 Java 컨트롤러 클래스

📌 주요 FXML 어노테이션

어노테이션	설명
`@FXML`	FXML에서 선언한 컴포넌트를 Java 필드/메서드로 연결할 때 사용
`fx:id`	FXML 내에서 컴포넌트에 ID를 지정
`onAction`	버튼 등 이벤트 발생 시 호출할 메서드 지정
`fx:controller`	해당 FXML을 제어하는 Java 컨트롤러 클래스 지정

💡 장점

- **UI와 로직을 분리**하여 코드 가독성 및 유지보수성 향상
- **디자이너-개발자 분업**에 유리함
- Scene Builder 같은 도구로 **시각적 편집 가능**

🛠️ 도구: Scene Builder

- `.fxml` 파일을 **GUI 기반으로 편집**할 수 있는 도구
- 드래그 앤 드롭으로 컨트롤 추가
- 자동으로 `fx:id`, `onAction` 등을 지정해줌

✅ 정리

구성 요소	역할
`.fxml` 파일	UI 구조를 선언적으로 기술
Controller	이벤트 처리 및 로직 연결
`FXMLLoader`	FXML 파일을 로드하여 Node 생성
`@FXML`	Java 코드와 FXML 연결

이벤트 처리, CSS 적용

JavaFX는 HTML/CSS에서 영감을 받아 **스타일을 CSS로 분리하여** 표현 가능하다.

✅ 기본 CSS 파일 예시 (`style.css`)

```
.button {
    -fx-background-color: #4CAF50;
    -fx-text-fill: white;
    -fx-font-size: 14px;
}

.text-field {
    -fx-border-color: #333;
}
```

✅ CSS 적용 방법

1️⃣ Java 코드로 적용

```
scene.getStylesheets().add(getClass().getResource("style.css").toExternalForm());
```

2️⃣ FXML에서 직접 지정

```
<AnchorPane stylesheets="@style.css">
    ...
</AnchorPane>
```

또는 개별 노드에 `styleClass` 지정:

```
<Button text="Click" styleClass="button"/>
<TextField fx:id="inputField" styleClass="text-field"/>
```

✅ 주요 JavaFX CSS 속성

속성 이름	설명
`-fx-background-color`	배경 색
`-fx-text-fill`	글자 색
`-fx-border-color`	테두리 색
`-fx-font-size`	글자 크기
`-fx-padding`	내부 여백
`-fx-alignment`	정렬

> JavaFX CSS는 W3C CSS와 유사하지만, 속성명이 `-fx-`로 시작함

✅ 스타일 클래스 vs 인라인 스타일

방법	설명	예시
`styleClass`	CSS 클래스 이름 부여	`node.getStyleClass().add("my-class");`
`setStyle()`	인라인 스타일 지정	`button.setStyle("-fx-background-color: red;");`

✅ Scene Builder에서 스타일 적용

- CSS 파일을 등록할 수 있음
- 각 컴포넌트에 `styleClass` 지정 가능
- 실시간 미리보기 가능

✅ 정리

항목	설명
이벤트 등록	FXML, 람다, EventHandler 클래스 등
이벤트 흐름	Capturing → Target → Bubbling
CSS 적용 방식	Java 코드, FXML, Scene Builder
CSS 대상	`styleClass`, `setStyle()` 등
CSS 파일 구조	`.css` 파일 별도 작성, Scene에 주입