

프로그래밍 언어 활용 강의안

(GUI JAVAFX)

JavaFX 개요

자바 UI 변천사

AWT(Abstract Window Toolkit)

운영 체제가 제공하는 네이티브 UI 컴포넌트 이용
운영 체제에 따라 UI의 모양 서로 달랐고, 종류도 제한적

Swing

모든 운영체제상에서 동일한 UI 갖도록
사용자는 애니메이션 추가된 시각적 운영 체제의 네이티브 UI 더 선호
네이티브 UI로 보여지도록 자신의 UI 재정비
실행 성능이 느려지고, 메모리 더 많이 사용

JavaFX

가볍고 풍부한 UI 제공
레이아웃, 스타일, 애플리케이션 로직 분리 개발
자바7 업데이트6버전부터 JavaFX2.2를 JDK와 JRE에 포함

<https://docs.oracle.com/javase/8/javafx/api/toc.htm>

JavaFX 개요

JavaFX 애플리케이션 구성하는 파일단위 구성요소

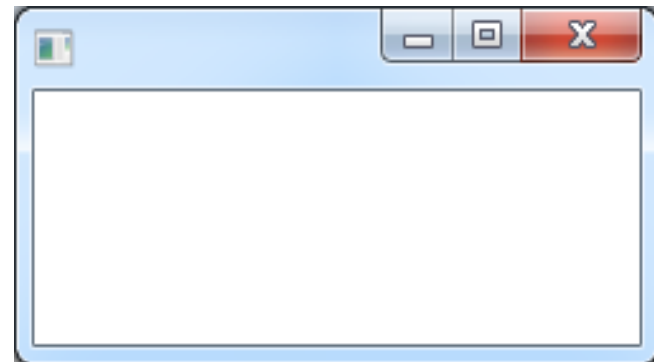


JavaFX 애플리케이션 개발 시작

메인 클래스

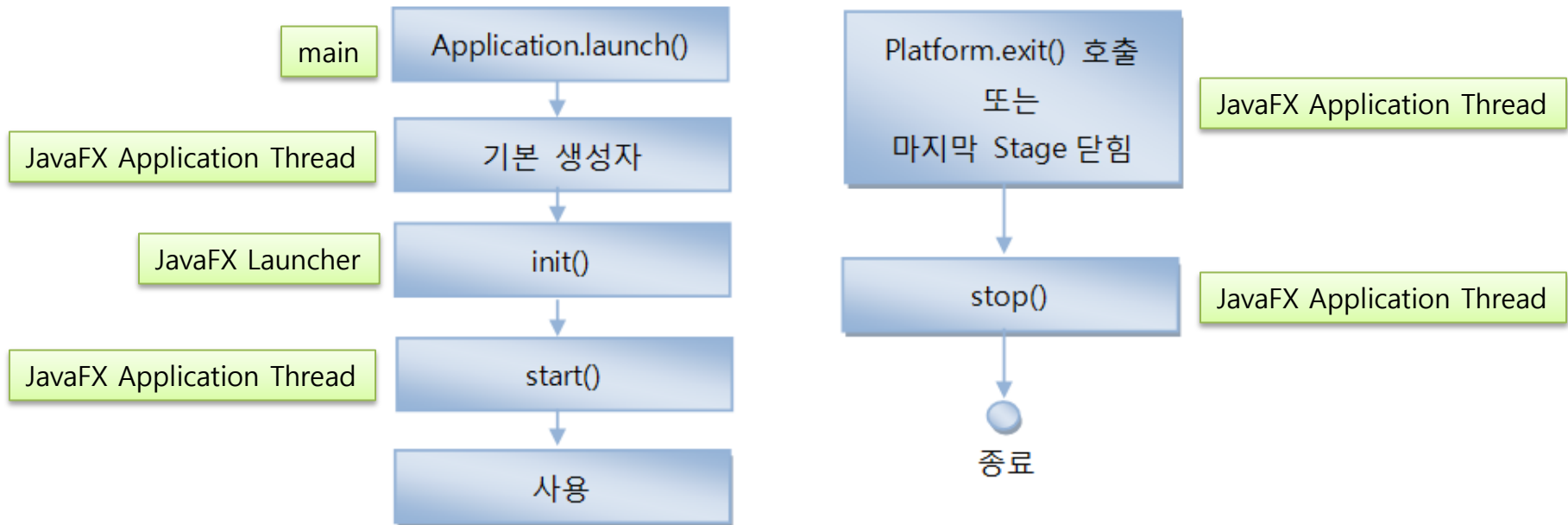
[AppMain.java] 메인 클래스

```
1 public class AppMain extends Application {  
2     @Override  
3     public void start(Stage primaryStage) throws Exception {  
4         primaryStage.show(); //윈도우 보여주기  
5     }  
6  
7     public static void main(String[] args) {  
8         launch(args); //AppMain 객체 생성 및 메인 윈도우 생성  
9     }  
10 }
```



JavaFX 애플리케이션 개발 시작

JavaFX 라이프 사이클



JavaFX 애플리케이션 개발 시작

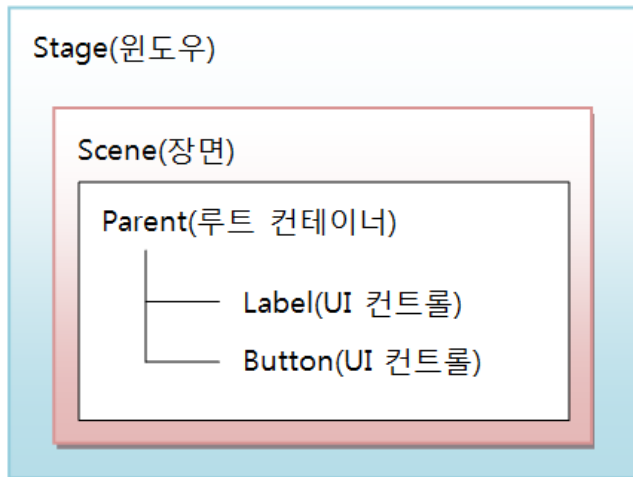
메인 클래스 실행 매개값 얻기

init() 메소드에서 다음 두 가지 방법으로 매개값 얻기

```
Parameters params = getParameters();  
List<String> list = params.getRaw();           //①  
Map<String, String> map = params.getNamed();   //②
```

무대와 장면

무대(Stage)는 윈도우 하나에 하나의 장면 가질 수 있음
장면은 javafx.scene.Scene 으로 표현



FXML 레이아웃

레이아웃

Scene에 포함된 다양한 컨트롤들을 배치하는 것

프로그램적 레이아웃

자바 코드로만 개발

간단하게 쉽게 만들 것 - 코드를 잘 정리하지 않으면 난해한 프로그램

개발자가 **직접** 작성 - 디자이너와 협력 개발 어려움

개발 완료 후 간단한 레이아웃 변경이나 스타일 변경이라도 자바 소스 수정 후 재 컴파일

FXML 레이아웃

FXML 레이아웃

FXML은 XML 기반의 마크업 언어

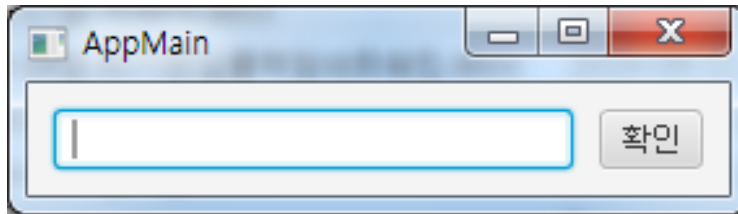
JavaFX UI 레이아웃 자바 코드에서 분리 - 태그로 선언하는 방법 제공

웹 애플리케이션 및 안드로이드(Android) 앱 개발법과 유사

디자이너와 협업 가능

간단한 레이아웃 변경이나 스타일 변경 시 자바 소스 수정 X

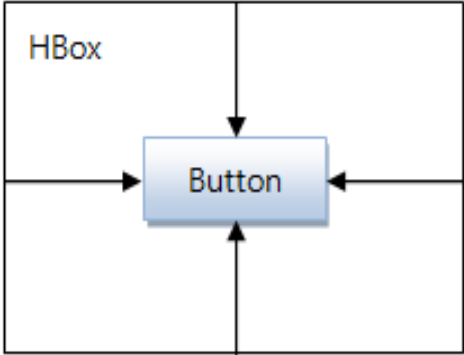
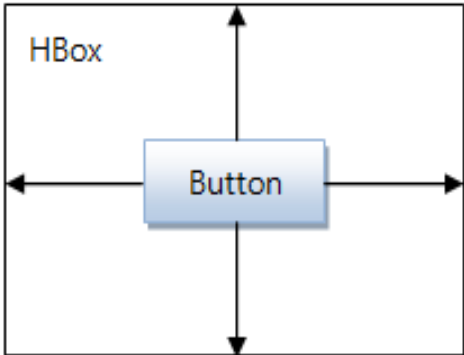
레이아웃이 비슷한 장면(Scene)들간에 재사용 가능



FXML 레이아웃

레이아웃 여백: 마진(margin)과 패딩(padding)

패딩은 안쪽 여백, 마진은 바깥 여백

구분	HBox 의 패딩	Button 의 마진
개념		
자바 코드	<pre>HBox hbox = new HBox(); hbox.setPadding(new Insets(50));</pre>	<pre>Button button = new Button(); HBox.setMargin(button, new Insets(50));</pre>
FXML 태그	<pre><HBox> <padding> <Insets topRightBottomLeft="50"/> </padding> </HBox></pre>	<pre><Button> <HBox.margin> <Insets topRightBottomLeft="50"/> </HBox.margin> </Button></pre>

FXML 레이아웃

FXML 작성 규칙

FXML 태그는 자바 코드로 변환되어 실행

자바 코드와 매핑 관계 존재

매핑 관계 잘 이해하면 JavaFX API 참조해 FXML 태그 쉽게 작성

패키지 선언 (**위치 중요!**) – 해당 클래스가 존재하지 않으면 에러

자바 코드	FXML 태그
<code>import javafx.scene.layout.HBox;</code>	<code><?import javafx.scene.layout.HBox?></code>
<code>import javafx.scene.control.*;</code>	<code><?import javafx.scene.control.*?></code>

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

```
<?import javafx.scene.layout.HBox?>
```

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

```
<루트컨테이너 xmlns:fx="http://javafx.com/fxml" >
```

```
...
```

```
</루트컨테이너>
```

FXML 레이아웃

❖ 태그 선언

FXML 태그는 < 와 > 사이에 태그 이름 작성
반드시 시작 태그가 있으면 끝 태그도 있어야
없으면 javax.xml.stream.XMLStreamException 예외 발생

자바 코드	FXML
<pre>Button button = new Button(); button.setText("확인");</pre>	<pre><Button > <text>확인</text> </Button></pre>

❖ 속성 선언

속성값은 큰따옴표(") 또는 작은따옴표(')로 반드시 감싸야
javax.xml.stream.XMLStreamException 예외 발생

자바 코드	FXML (Setter 태그)	FXML (Setter 속성)
<pre>Button button = new Button(); button.setText("확인");</pre>	<pre><Button > <text>확인</text> </Button></pre>	<pre><Button text="확인"/></pre>

FXML 레이아웃

객체 선언

Setter 메소드가 기본 타입과 String 타입이 아닌 다른 타입의 객체를 매개값으로 갖는다면 ?

속성으로 작성할 수 없고, 태그로 작성해야

<클래스 속성="값">

자바 코드	FXML
<pre>HBox hbox = new HBox(); hbox.setPadding(new Insets(10,10,10,10));</pre>	<pre><HBox> <padding> <Insets top="10" right="10" bottom="10" left="10"/> </padding> </HBox></pre>

<클래스 fx:value="값">

자바 코드	FXML
<pre>String.valueOf("Hello, World!"); Integer.valueOf("1"); Double.valueOf("1.0"); Boolean.valueOf("false");</pre>	<pre><String fx:value="Hello, World!"/> <Integer fx:value="1"/> <Double fx:value="1.0"/> <Boolean fx:value="false"/></pre>

FXML 레이아웃

객체 선언

<클래스 fx:constant="상수">

자바 코드	FXML
<pre>Button button = new Button(); button.setMaxWidth(Double.MAX_VALUE);</pre>	<pre><Button> <maxWidth> <Double fx:constant="MAX_VALUE"/> </maxWidth> </Button></pre>

<클래스 fx:factory="정적메소드">

자바 코드	FXML
<pre>ComboBox combo = new ComboBox(); combo.setItems(FXCollections.observableArrayList("공개", "비공개"));</pre>	<pre><ComboBox> <items> <FXCollections fx:factory="observableArrayList"> <String fx:value="공개"/> <String fx:value="비공개"/> </FXCollections> </items> </ComboBox></pre>

FXML 레이아웃

FXML 로딩과 Scene 생성

FXML 로딩

- FXML 파일을 읽어 들여 선언된 내용을 객체화하는 것

- FXMLLoader 의 load() 메소드 이용

- load()가 리턴하는 실제 객체

- FXML 파일에서 루트 태그로 선언된 컨테이너

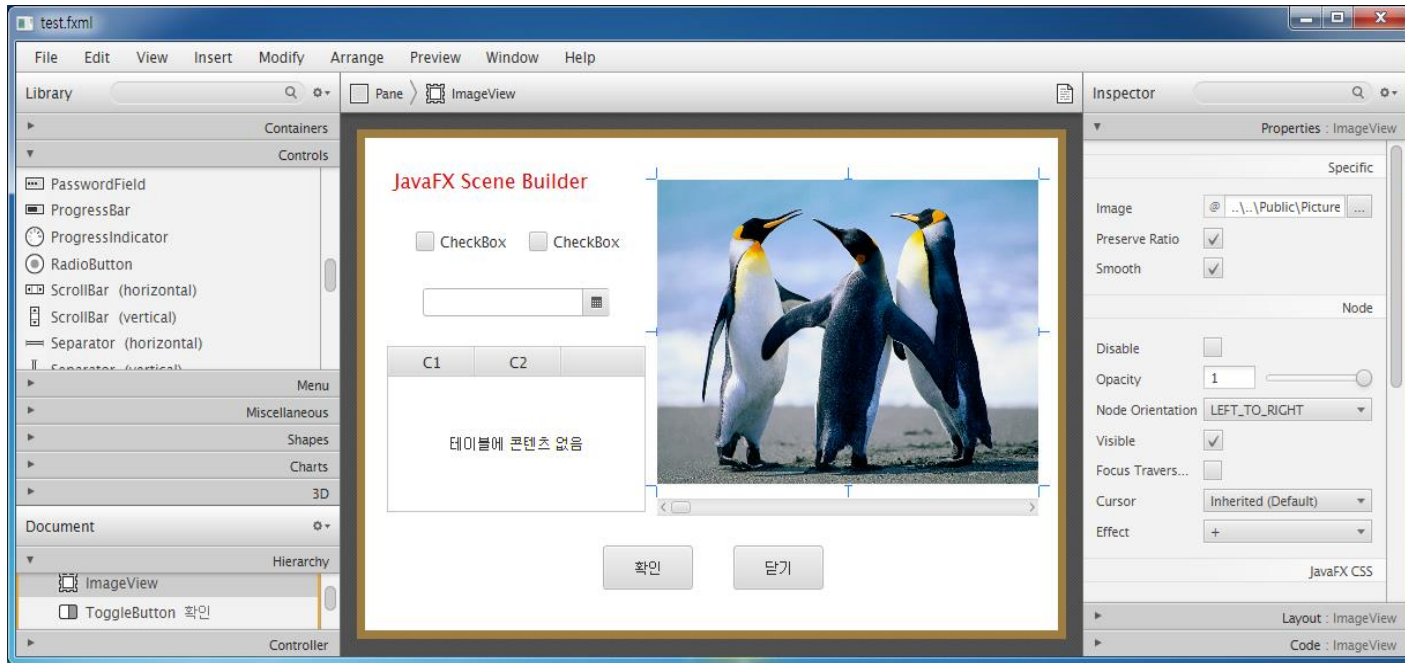
Scene 객체 생성

- FXML 로딩 후 얻은 루트 컨테이너는 Scene을 생성할 때 매개값으로 사용

FXML 레이아웃

JavaFX Scene Builder

드래그 앤 드롭 방식의 WYSIWYG 디자인 툴
자동으로 FXML 파일 생성



설치방법

오라클에서 다운로드 후 설치

E(fx)clipse 플러그인 설치하면 더 편리하게 사용가능

JavaFX 컨테이너

컨테이너

레이아웃 작성시 다양한 컨트롤들을 쉽게 배치하도록 해주는 역할
javafx.scene.layout 패키지에 속함

컨테이너의 종류

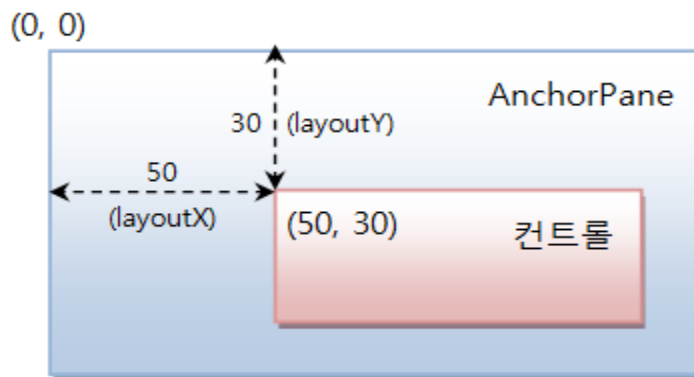
컨테이너	설명
AnchorPane	컨트롤을 좌표를 이용해서 배치하는 레이아웃
BorderPane	위, 아래, 오른쪽, 왼쪽, 중앙에 컨트롤을 배치하는 레이아웃
FlowPane	행으로 배치하되 공간이 부족하면 새로운 행에 배치하는 레이아웃
GridPane	그리드로 배치하되 셀의 크기가 고정적이지 않는 레이아웃
StackPane	컨트롤을 겹쳐 배치하는 레이아웃
TilePane	그리드로 배치하되 고정된 셀의 크기를 갖는 레이아웃
HBox	수평으로 배치하는 레이아웃
VBox	수직으로 배치하는 레이아웃

JavaFX 컨테이너

AnchorPane 컨테이너

JavaFX Scene Builder 사용해 디자인
눈으로 거리 확인해 컨트롤 드롭

AnchorPane 컨테이너는 AnchorPane 의 좌상단(0, 0)을 기준으로 컨트롤을 좌표로 배치한다.
컨트롤 좌표는 좌상단 (layoutX, layoutY) 값을 말하는데 (0,0) 에서 떨어진 거리이다.



JavaFX 컨테이너

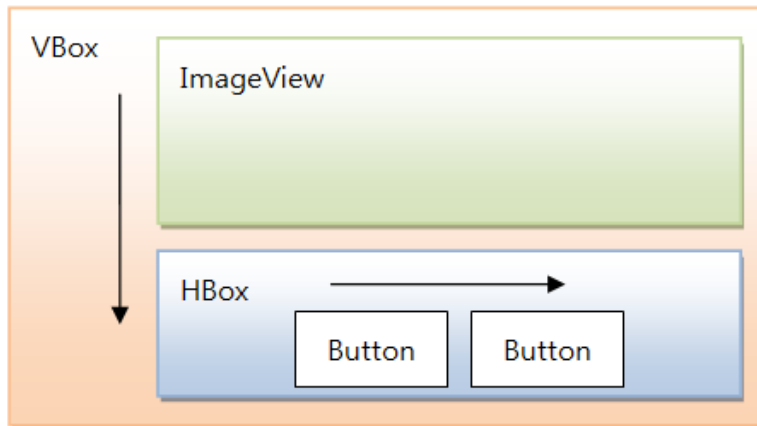
HBox와 VBox 컨테이너

수평과 수직으로 컨트롤을 배치하는 컨테이너

자식 컨트롤의 크기 재조정에 쓰임

HBox는 컨트롤의 높이 확장하고, 컨트롤의 폭은 유지

VBox는 컨트롤의 폭 확장하고 컨트롤의 높이는 유지



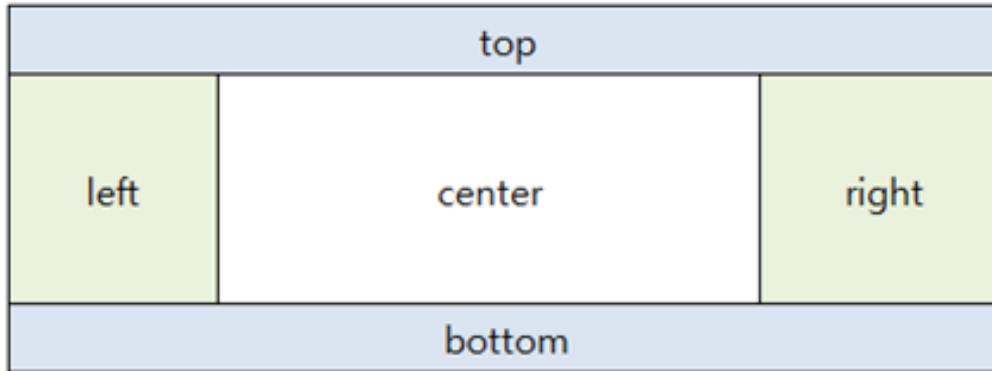
JavaFX 컨테이너

BorderPane 컨테이너

top, bottom, left, right, center 셀에 컨트롤 배치하는 컨테이너

각 셀에는 하나의 컨트롤 또는 컨테이너만 배치

top, bottom, left, right에 배치하지 않으면 center에 배치된 컨트롤이 사방으로 자동 확장

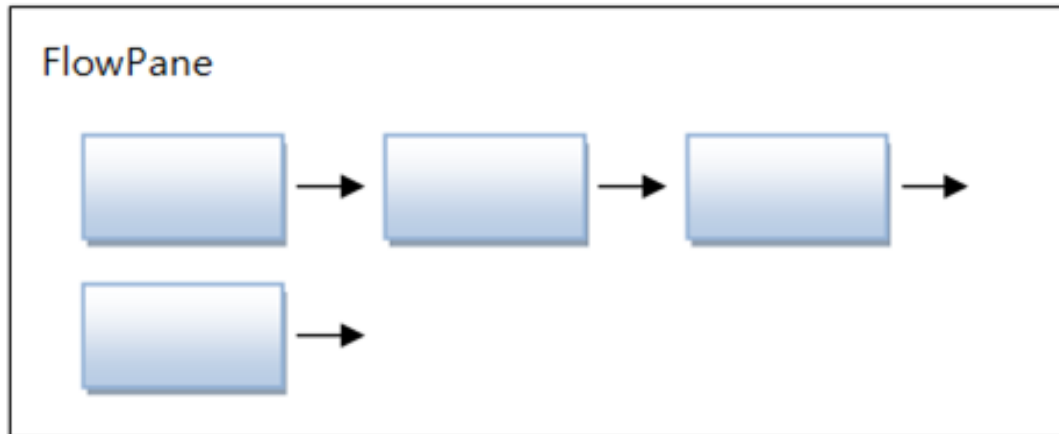


JavaFX 컨테이너

FlowPane 컨테이너

행으로 컨트롤 배치

공간 부족하면 새로운 행에 배치하는 컨테이너
윈도우 창을 늘였다 줄여보면 쉽게 이해 가능



JavaFX 컨테이너

TilePane 컨테이너

그리드로 컨트롤 배치

고정된 셀(타일) 크기 갖는 컨테이너

오른쪽에 컨트롤 배치할 공간 부족하면 새로운 행에 컨트롤 배치



JavaFX 컨테이너

GridPane 컨테이너

그리드로 컨트롤 배치

셀의 크기가 고정적이지 않고 유동적인 컨테이너

셀 병합 - 다양한 입력 폼 화면 만들 때 매우 유용



JavaFX 컨테이너

StackPane 컨테이너

컨트롤을 겹쳐 배치하는 컨테이너 (카드 레이아웃)

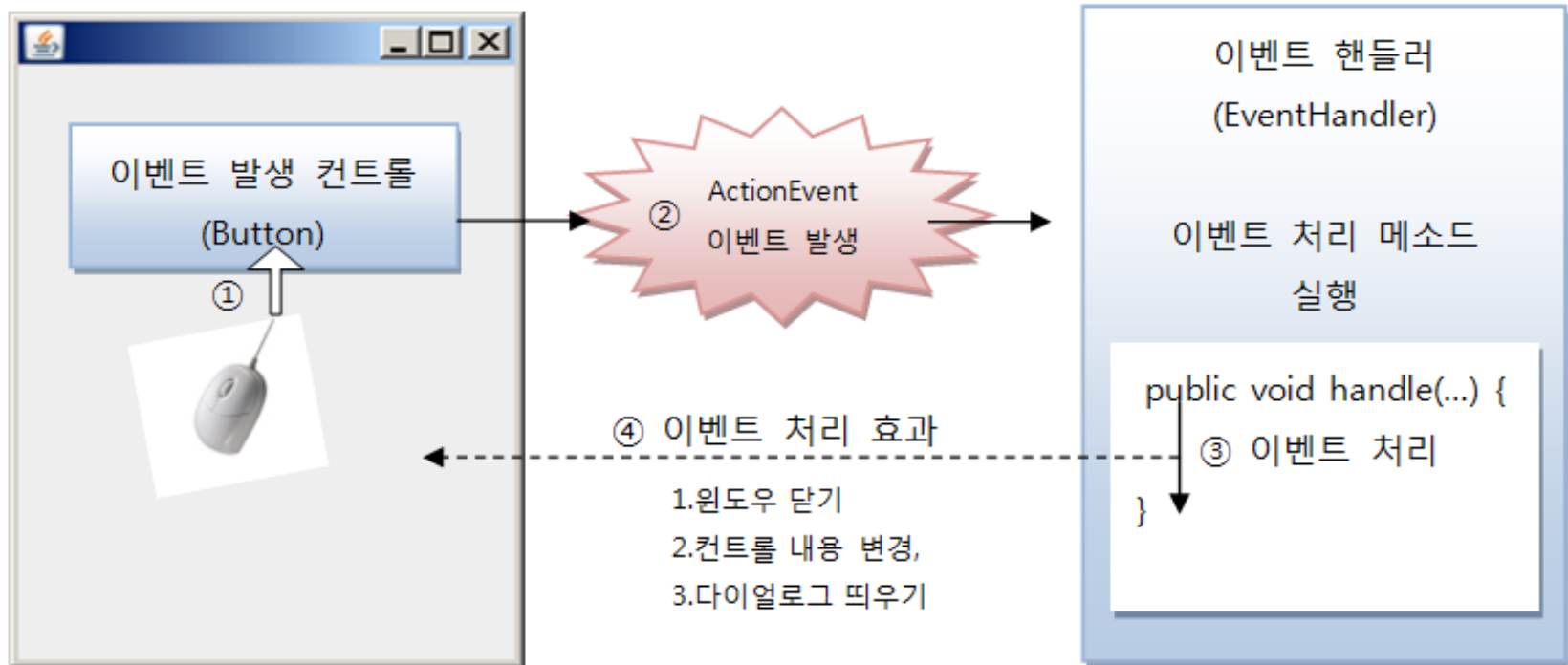
위에 있는 컨트롤이 투명이라면 밑에 있는 컨트롤이 겹쳐 보임



JavaFX 이벤트 처리

이벤트 핸들러

이벤트 발생 컨트롤과 이벤트 핸들러를 분리하는 위임형 방식



JavaFX 이벤트 처리

FXML 컨트롤러

FXML 파일당 별도의 컨트롤러(Controller) 지정해 이벤트 처리
FXML 레이아웃과 이벤트 처리 코드 완전히 분리

fx:controller 속성과 컨트롤러 클래스
UI 컨트롤에서 발생하는 이벤트를 컨트롤러가 처리

fx:id 속성과 @FXML 컨트롤 주입
컨트롤러의 @FXML 어노테이션이 적용된 필드에 자동 주입.
fx:id 속성값과 필드명은 동일해야 함

EventHandler 생성 및 등록
컨트롤에서 발생하는 이벤트 처리 (메소드 매핑)

JavaFX 속성 감시와 바인딩

속성 감시

컨트롤의 속성값 변화 감시하는 ChangeListener 등록 가능
속성값에 변화가 생기면 ChangeListener의 changed() 호출
JavaFX 컨트롤 속성의 구성

- Setter

- Getter

- Property 객체 리턴하는 메소드

JavaFX 속성 감시와 바인딩

속성 바인딩

두 컨트롤의 속성을 서로 연결하는 것

바인드 된 속성들은 하나가 변경되면 자동적으로 다른 하나도 변경

단방향 바인드 – `bind()`

양방향 바인드 – `bindBidirectional()`

언바인드 – `unbind()`, 바인드 해제

JavaFX 속성 감시와 바인딩

Bindings 클래스



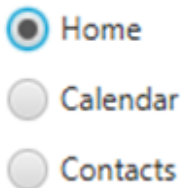
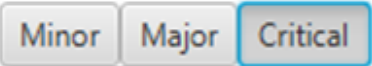
속성 연산하거나, 다른 타입으로 변환 후 바인딩하는 기능 제공

메소드	설명
add, subtract, multiply, divide	속성값을 덧셈, 뺄셈, 곱셈, 나눗셈 연산을 수행하고 바인딩함
max, min	속성값과 어떤 수를 비교해서 최대, 최소값을 얻고 바인딩함
greaterThan, greaterThanOrEqualTo	속성값이 어떤 값보다 큰지, 같거나 큰지를 조사해서 true/false 로 변환하여 바인딩함
lessThan, lessThanOrEqualTo	속성값이 어떤 값보다 적거나, 같거나 적은지를 조사해서 true/false 로 변환하여 바인딩함
equal, notEquals	속성값이 어떤 값과 같은지, 다른지를 조사해서 true/false 로 변환하여 바인딩함
equalIgnoreCase, notEqualIgnoreCase	대소문자와 상관없이 속성값이 어떤 문자열과 같은지, 다른지를 조사해서 true/false 로 변환하여 바인딩함
isEmpty, isEmpty	속성값이 비어있는지, 아닌지를 조사해서 true/false 로 변환하여 바인딩함
isNull, isNotNull	속성값이 null 또는 not null 인지를 조사해서 true/false 로 변환하여 바인딩함
length	속성값이 문자열일 경우 문자수를 얻어 바인딩함
size	속성 타입이 배열, List, Map, Set 일 경우 요소수를 얻어 바인딩함
and, or	속성값이 boolean 일 경우, 논리곱, 논리합을 얻어 바인딩함
not	속성값이 boolean 일 경우, 반대값으로 바인딩함
convert	속성값을 문자열로 변환해서 바인딩함
valueAt	속성이 List, Map 일 경우 해당 인덱스 또는 키의 값을 얻어 바인딩함

JavaFX 컨트롤




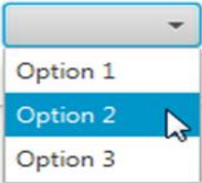
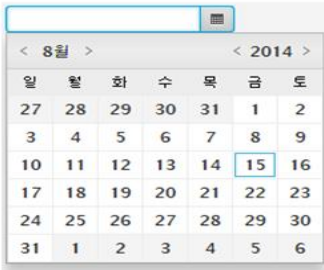

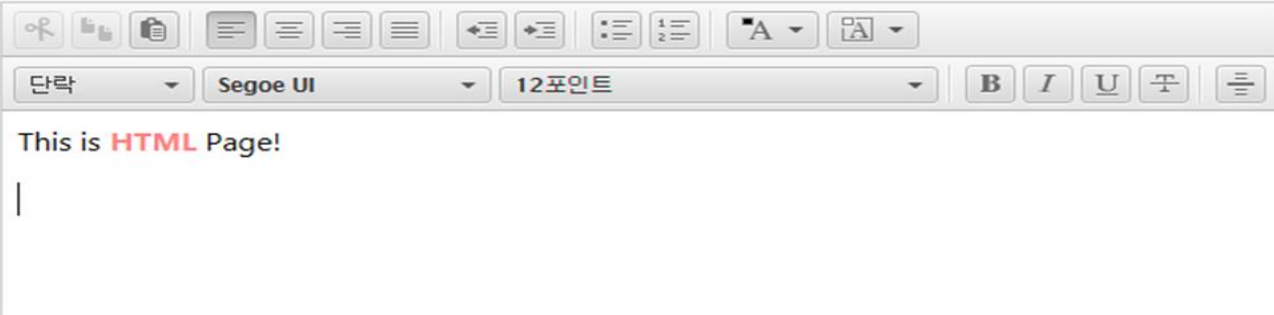
버튼 컨트롤

마우스로 클릭 가능한 컨트롤로 ButtonBase 상속하는 하위 컨트롤
버튼 컨트롤의 종류

Button	CheckBox	RadioButton	ToggleButton
			

JavaFX 컨트롤

입력 컨트롤의 종류

Label & TextField	PasswordField	TextArea
		
ComboBox	DatePicker	ColorPicker
		
HTMLEditor		
		

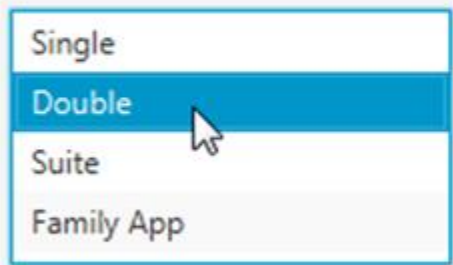

JavaFX 컨트롤

뷰 컨트롤

목록 형태로 보여주는 ListView

테이블 형태로 보여주는 TableView

이미지를 보여주는 ImageView

ListView	TableView	ImageView												
	<table><tr><th>First Name</th><th>Last Name</th></tr><tr><td>Jacob</td><td>Smith</td></tr><tr><td>Isabella</td><td>Johnson</td></tr><tr><td>Ethan</td><td>Williams</td></tr><tr><td>Emma</td><td>Jones</td></tr><tr><td>Michael</td><td>Brown</td></tr></table>	First Name	Last Name	Jacob	Smith	Isabella	Johnson	Ethan	Williams	Emma	Jones	Michael	Brown	
First Name	Last Name													
Jacob	Smith													
Isabella	Johnson													
Ethan	Williams													
Emma	Jones													
Michael	Brown													

JavaFX 컨트롤

미디어 컨트롤

비디오를 재생할 수 있는 MediaView 컨트롤

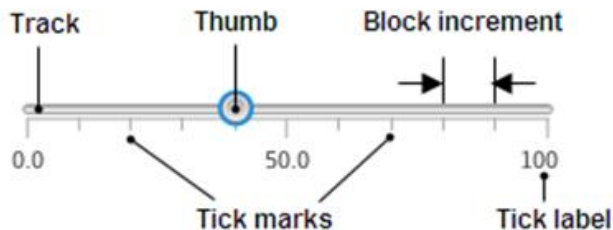
볼륨 조절 및 재생 위치 조절을 위한 Slider 컨트롤

현재 진행 상태 보여주는 ProgressBar, ProgressIndicator

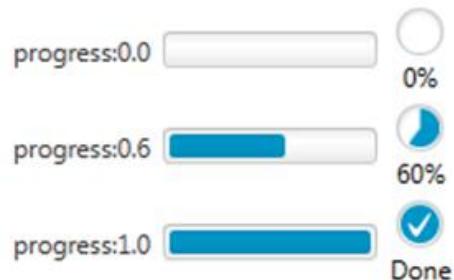
MediaView



Slider



ProgressBar 와 ProgressIndicator



JavaFX 컨트롤

차트 컨트롤

javafx.scene.chart 패키지에 포함

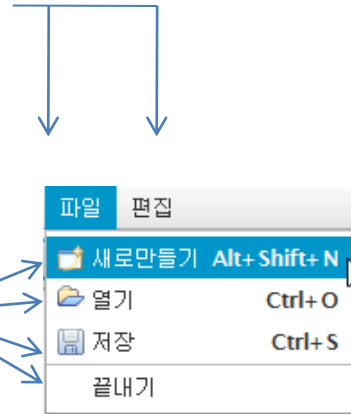


JavaFX 메뉴바와 툴바

메뉴바

MenuBar에는 Menu들이 배치
Menu에는 메뉴 아이템 추가

MenuItem,
CheckMenuItem,
RadioMenuItem,
CustomMenuItem,
SeparatorMenuItem
Menu(서브 메뉴)

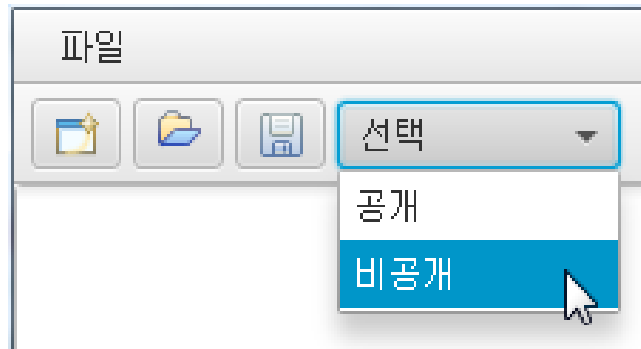


계층적인 작업 선택 기능 구현에 주로 쓰임

JavaFX 메뉴바와 툴바

툴바

빠르게 작업을 선택하고 싶을 때 사용
Toolbar 컨트롤은 UI 컨트롤이면서 컨테이너
Button이 추가되지만, ComboBox와 같은 다른 컨트롤도 배치



JavaFX 다이얼로그

다이얼로그(Dialog)

주 윈도우에서 알림 또는 사용자의 입력 위해 실행되는 서브 윈도우 자체적으로 실행될 수 없고, 주 윈도우(소유자 윈도우)에 의해서 실행

모달과 모달리스

모달 다이얼로그는 다이얼로그를 닫기 전까지 소유자 윈도우 사용 불가
모달리스 다이얼로그는 소유자 윈도우 계속 사용 가능

JavaFX에서 제공하는 다이얼로그 종류

파일을 선택하는 FileChooser

디렉토리를 선택하는 DirectoryChooser

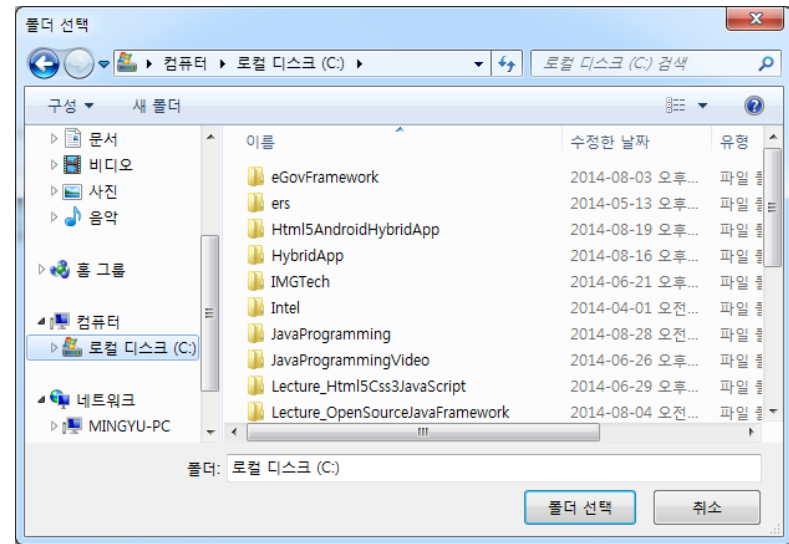
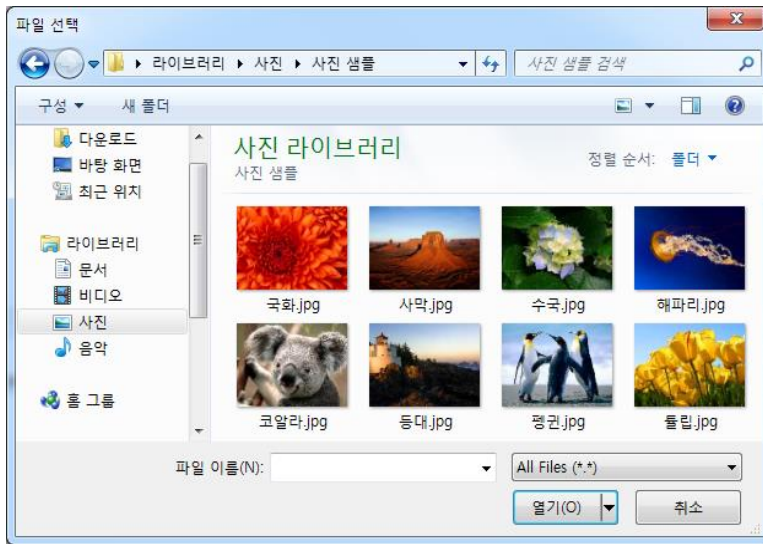
팝업창을 띄우는 Popup

javafx.stage 패키지에 모두 포함

JavaFX 다이얼로그

FileChooser, DirectoryChooser

XXXChooser 는 컨트롤이 아니라 FXML 에서 선언 불가
모달 다이얼로그 - 버튼 클릭하기 전에는 소유자 윈도우 사용 불가



JavaFX 다이얼로그

Popup

투명한 컨테이너 제공하는 모달리스 다이얼로그

윈도우의 기본 장식(아이콘, 제목, 최소화 및 복원 버튼, 닫기 버튼) 없음

용도

컨트롤의 툴팁(tooltip), 메시지 통지(notification), 드롭 다운 박스(drop down boxes)

Popup의 내용은 자바 코드로 작성하거나, FXML 파일로 작성

Popup은 최상위 윈도우

소유자 윈도우를 닫거나, `hide()`를 호출하면 닫힘

`setAutoHide(true)`: 다른 윈도우로 포커스 이동시 Popup은 자동 닫힘

JavaFX 다이얼로그

커스텀 다이얼로그

다양한 내용의 다이얼로그를 만들고 싶다면 Stage로 직접 생성
StageStyle 열거 상수와 윈도우 스타일

StageStyle 열거 상수	설명
DECORATED	일반적인 윈도우 스타일. 배경이 흰색, 제목줄에 장식(아이콘, 타이틀, 축소, 복원, 닫기 버튼 장식)이 있음
UNDECORATED	배경이 흰색, 제목줄 없음
TRANSPARENT	배경이 투명, 제목줄 없음
UNIFIED	DECORATED 와 동일하나, 내용물의 경계선이 없음
UTILITY	배경이 흰색이고, 제목줄에 타이틀, 종료 버튼만 있음

JavaFX 다이얼로그

컨트롤러에서 primaryStage 사용

- 컨트롤러에서 다이얼로그 실행

- 소유자 윈도우로 primaryStage 필요

컨트롤러에서 primaryStage 얻는 방법

- 메인 클래스에서 전달하는 방법

- 컨테이너 또는 컨트롤로부터 얻는 방법

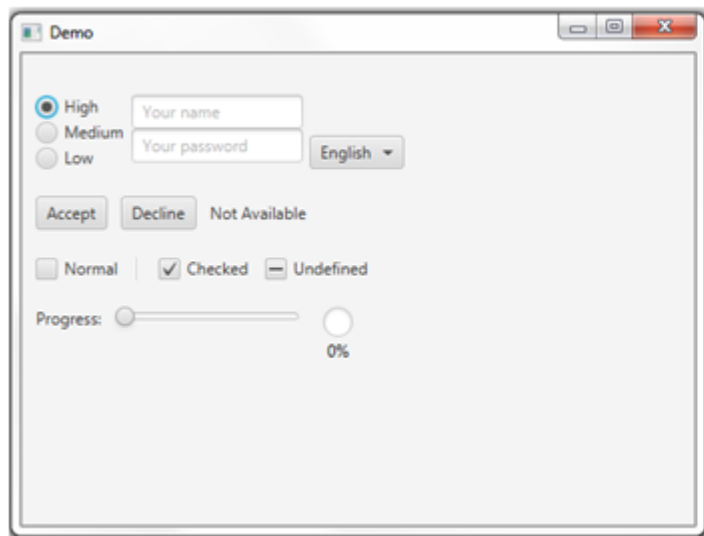
- initialize() 메소드 안에서는 사용 불가

JavaFX CSS 스타일

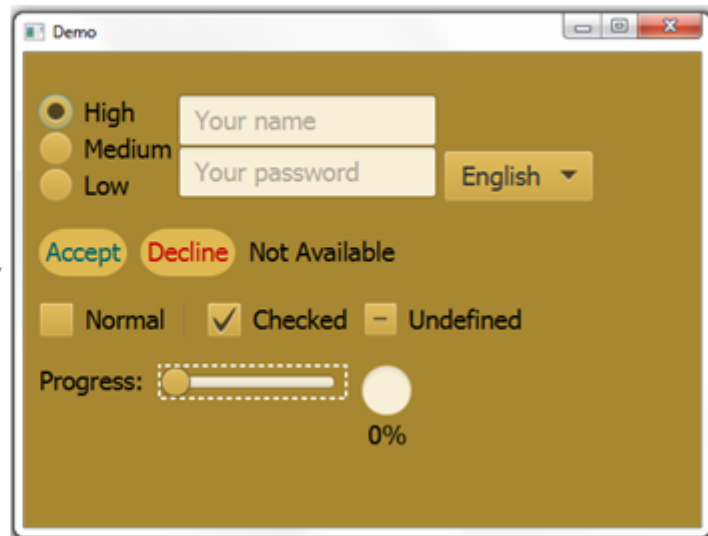
JavaFX 애플리케이션

FXML(레이아웃) + CSS(스타일) + 자바(컨트롤러, 로직)

[기본 CSS]



[커스텀 CSS]



JavaFX CSS

W3C CSS 버전 2.1 스펙(<http://www.w3.org/TR/CSS21/>) 준수

FXML 인라인 스타일 또는 외부 CSS 파일로 작성 가능

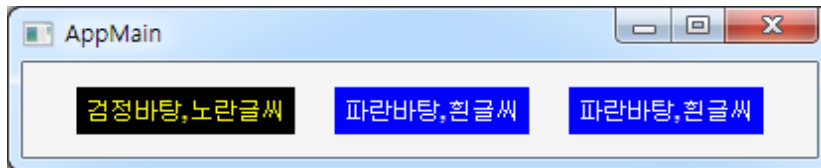
W3C CSS 속성명 앞에 "-fx-" 붙임

JavaFX CSS 스타일

인라인(inline) 스타일

컨테이너 또는 컨트롤의 style 속성값으로 직접 CSS 정의
쉽고, 빠르게 모양과 색상 변경

```
<Label id="label1" text="검정바탕,노란글씨"  
  style="-fx-background-color: black; -fx-text-fill: yellow; -fx-padding: 5;"/>  
<Label text="파란바탕,흰글씨"  
  style="-fx-background-color: blue; -fx-text-fill: white; -fx-padding: 5;"/>
```



JavaFX CSS 스타일

외부 CSS 파일

인라인 스타일 문제점

동일한 스타일을 적용하는 컨트롤 많을수록 중복 코드가 많이 늘어남
FXML과 CSS가 뒤섞여 추후 유지 보수가 어려움

선택자:

외부 CSS 파일

스타일 적용할 컨테이너와 컨트롤 선택해주는 선택자 필요

JavaFX CSS 스타일

선택자의 종류

Type 선택자: `Type { 속성:값; 속성:값; ... }`

id 선택자: `#id { 속성:값; 속성:값; ... }`

class 선택자: `.class { 속성:값; 속성:값; ... }`

Type 선택자와 class 선택자 조합

상태별 선택자

상태	상태별 선택자
입력 가능한 상태	선택자:focusd { 속성:값; 속성:값; ... }
마우스가 컨트롤 위에 있는 상태	선택자:hover { 속성:값; 속성:값; ... }
마우스로 컨트롤을 누른 상태	선택자:pressed { 속성:값; 속성:값; ... }

CSS 파일 적용

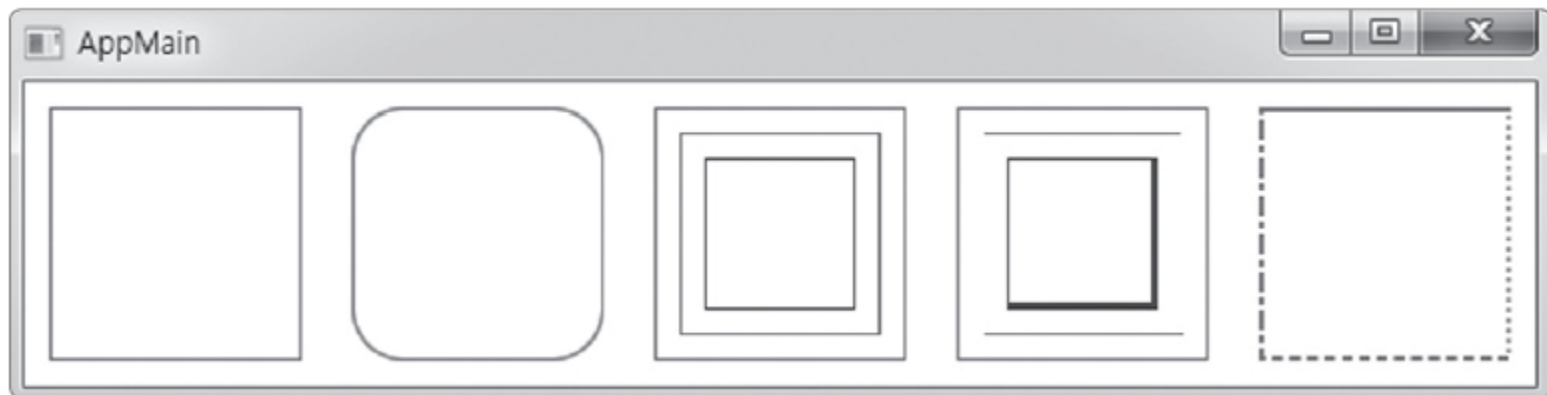
Scene에 추가하여 Scene 내부의 모든 컨테이너와 컨트롤에 적용

JavaFX 다이얼로그

border 속성

컨테이너 및 컨트롤의 경계선의 스타일 설정

속성	설명
-fx-border-color	경계선의 색상
-fx-border-insets	내부 경계선의 위치
-fx-border-radius	둥근 모서리를 위한 원의 반지름
-fx-border-style	경계선의 스타일(실선, 점선)
-fx-border-width	경계선의 굵기



JavaFX 다이얼로그

background 속성

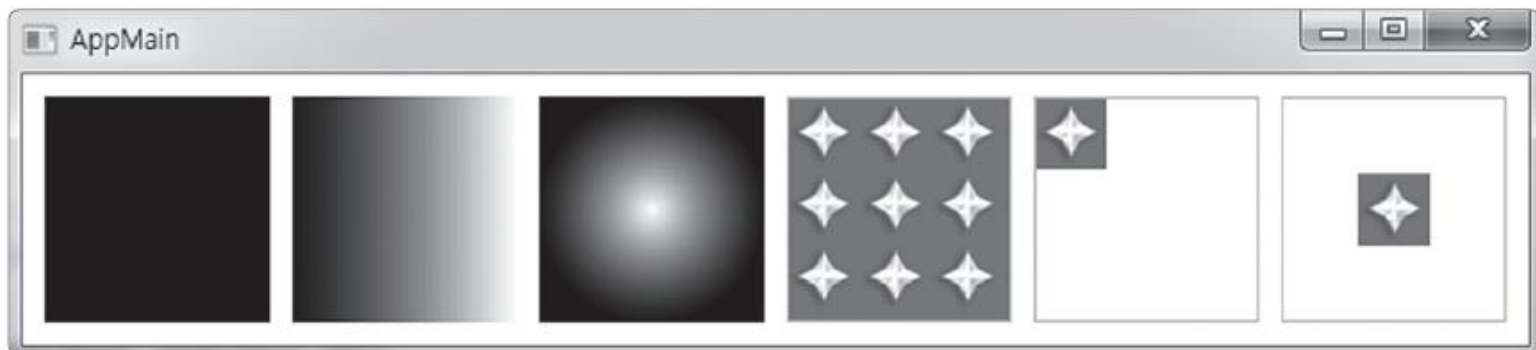
컨테이너 및 컨트롤의 배경 스타일을 설정

속성	설명
-fx-background-color	배경 색상
-fx-background-image	배경 이미지
-fx-background-position	배경 이미지 위치 (top, right, bottom, left, center)
-fx-background-repeat	이미지 반복 여부 (no-repeat: 반복하지 않음)

-fx-background-color

선형 그라디언트

원형 그라디언트

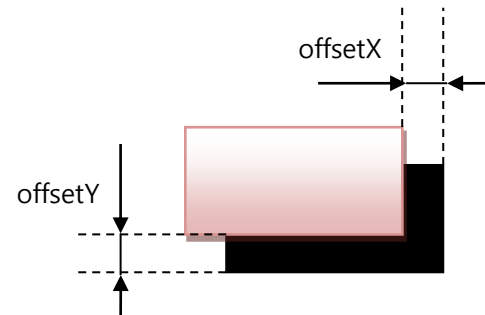
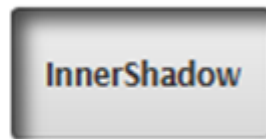
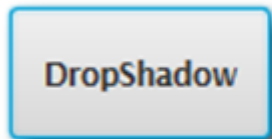


JavaFX 다이얼로그

font 속성

속성	설명
-fx-font-size	폰트 크기
-fx-font-family	폰트 종류
-fx-font-weight	폰트 굵기(bold)
-fx-text-fill	폰트 색상(단색, 선형 그라디언트, 원형그라디언트)

shadow 효과(-fx-effect)



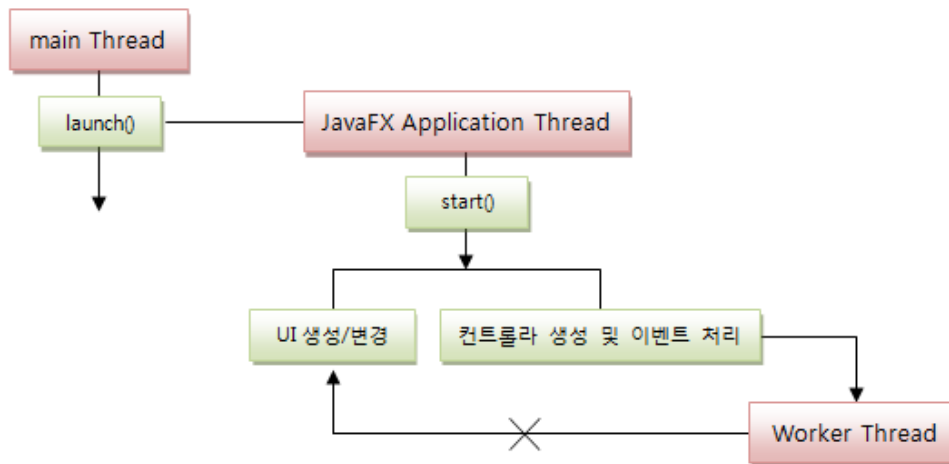
blur-type : gaussian, one-pass-box, three-pass-box, two-pass-box
radius: blur kernel의 반지름, 0.0~127.0 사이의 값 , 기본값 10
spread, choke: 그림자의 spread와 choke, 0.0~1.0 사이의 값. 기본값은 0.0
offsetX, offsetY: 그림자의 편차

JavaFX 스레드 동시성

JavaFX UI 스레드 동시성

JavaFX UI는 스레드에 안전하지 않음

UI를 생성하고 변경하는 작업은 JavaFX Application Thread가 담당
다른 작업 스레드들은 UI를 생성하거나 변경할 수 없음



JavaFX Application Thread는 시간을 요하는 작업 하지 않도록
시간을 요하는 작업을 하게 되면 UI는 멈춰있는 상태

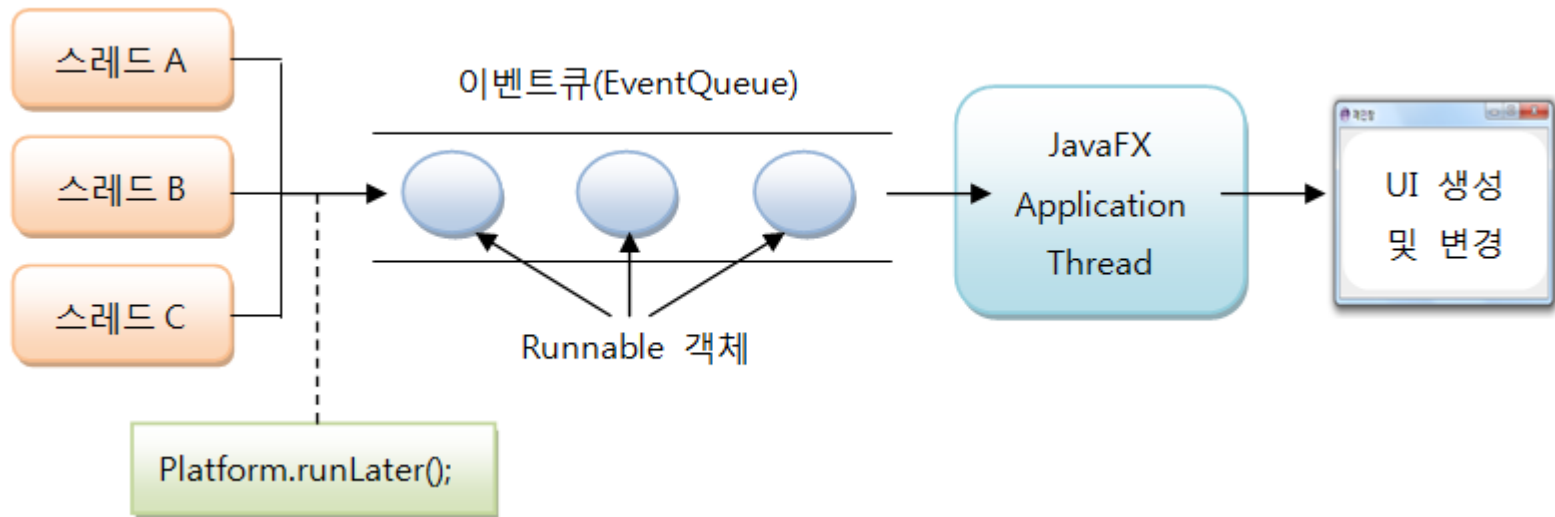
Ex) 파일 읽고 쓰기, 네트워크상에서 데이터를 주고 받을 경우
다른 작업 스레드 생성해 처리

JavaFX 스레드 동시성

Platform.runLater() 메소드

작업 스레드는 UI 를 변경할 수 없음

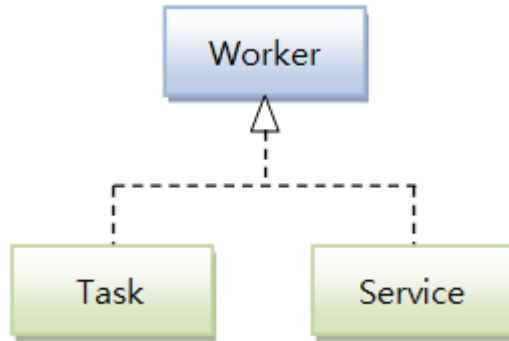
UI 변경 필요한 경우 Runnable 로 생성해 실행



JavaFX 스레드 동시성

Task 클래스

javafx.concurrent 패키지가 제공하는 스레드 동시성 API



Worker 인터페이스

작업 스레드가 UI 변경할 때 Task와 Service에서 공통적으로 사용할 수 있는 메소드 제공

Task 추상 클래스

JavaFX 애플리케이션에서 비동기 작업을 표현한 클래스

Service 추상 클래스

Task를 간편하게 시작, 취소, 재시작 할 수 있는 기능 제공하는 클래스

JavaFX 스레드 동시성

Task 생성

하나의 작업을 정의할 때는 Task 상속해 클래스 생성
그 후에 call () 메소드 재정의해 작업 결과 값 얻도록

Task 취소

Task가 처리되는 도중 취소 하려면 cancel() 메소드 호출
Task는 cancel() 메소드가 호출되었는지 검사해 작업 멈출 수 있도록

UI 변경

call() 메소드는 작업 스레드상에서 호출 - UI 변경 코드 작성 불가
updateProgress(), updateMessage() 메소드 호출
UI 속성 바인딩
Platform.runLater() 메소드 이용

JavaFX 스레드 동시성

작업 상태 별 콜백

작업이 처리 결과 따라 Task의 다음 세가지 메소드 중 하나 자동 콜백

콜백 메소드	설명
succeeded()	성공적으로 call() 메소드가 리턴되었을 때
cancelled()	cancel() 메소드로 작업이 취소되었을 때
failed()	예외가 발생되었을 때

Task 클래스를 작성할 때 재정의해서 애플리케이션 로직으로 재구성 가능

작업 결과가 있는 Task일 경우(call() 메소드가 리턴값 있을 경우)

succeeded() 메소드 재정의해 작업 결과 얻는 것 가능

V는 Task의 타입 파라미터에 지정된 타입

JavaFX Application Thread상에서 호출

안전하게 UI 변경 코드 작성 가능

JavaFX 스레드 동시성

Service 클래스

작업 스레드상에서 Task를 간편하게 시작, 취소, 재시작 기능 제공

Service 생성

Service를 상속받고 `createTask()` 메소드 재정의
`createTask()`는 작업 스레드가 실행할 Task를 생성해서 리턴

Service 시작, 취소 재시작

`start()`
`cancel()`
`restart()`

JavaFX Application Thread 상에서 호출

JavaFX 스레드 동시성

작업 상태 별 콜백

작업이 어떻게 처리됐는지 따라 Service의 다음 세 가지 메소드 중 하나가 자동 콜백

콜백 메소드	설명
succeeded()	성공적으로 call() 메소드가 리턴되었을 때
cancelled()	cancel() 메소드로 작업이 취소되었을 때
failed()	예외가 발생되었을 때

Service 클래스를 작성할 때 재정의해서 애플리케이션 로직으로 재구성

작업 결과가 있는 Task일 경우(call() 메소드가 리턴값이 있을 경우)
succeeded() 메소드를 재정의해 작업 결과 얻음
V는 Task의 타입 파라미터에 지정된 타입

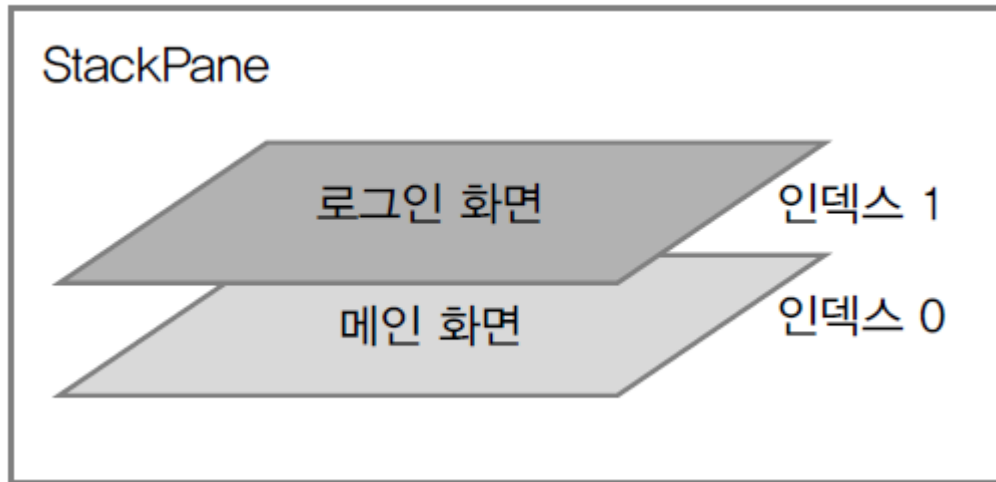
JavaFX Application Thread상에서 호출
안전하게 UI 변경 코드 작성 가능

화면 이동과 애니메이션

화면 이동

Stage에 새로운 Scene을 세팅하는 것

StackPane 을 루트 컨테이너로 사용하면 애니메이션도 사용 가능



화면 이동과 애니메이션

애니메이션

컨트롤 또는 컨테이너의 속성(Property) 변화를 주어진 시간 동안 진행함으로써 구현

애니메이션과 관련된 클래스

클래스	설명
Timeline	KeyFrame에 설정된 내용대로 애니메이션을 진행시키는 객체
KeyValue	타겟 속성(Property)과 종료값을 설정하는 객체
KeyFrame	애니메이션의 지속 시간과 KeyValue를 설정하는 객체 (지속 시간 동안 타겟 속성의 값을 종료값까지 변화시킴)