* JAVA FX 개요

1. JAVA FX는 사용자 애플리케이션을 개발하기 위한 그래픽과 미디어의 패키지를 말한다. **JAVA FX는 자바7버전부터 JDK에 포함되어 있기 때문에 별도의 SDK 설치 필요 없이도 바로 사용할 수 있다.**

2. 현재 JAVA FX는 자바 표준 UI 라이브러리였던 스윙보다 더 가벼워지고 더 강력해졌기 때문에 스윙을 대체하는 새로운 라이브러리로 자리매김하고 있다. JAVA FX는 데스크탑 UI 애플리케이션을 개발할 수 있도록 가볍고 풍부한 UI를 제공하고 있다.

* AWT 와 Swing, JAVA FX 비교

1. AWT

가. 자바 언어가 탄생한 1995년에는 인터넷이 활성화되지 않았기 때문에 대부분 사용자 프로그램은 운영체제가 제공하는 UI 컴포넌트를 이용해서 개발되었다. 그래서 JAVA 1.0 에 포함된 AWT(Abstract Window Toolkit)는 운영 체제가 제공하는 UI 컴포넌트를 이용하는 자바 라이브러리였다.

나. 그렇다 보니 자바 프로그램이 실행되는 운영 체제에 따라 UI 모양이 서로 달랐고, 종류도 제한적이였다.

2. Swing

가. AWT 다음 주자로 Swing 이 나왔다. 스윙의 목적은 운영 체제가 제공하는 UI를 사용하지 말자는 것이었다. 즉 모든 운영체제상에서 동일한 ui를 갖도록 스윙 자신만의 UI를 갖도록 하자는 것인데 자바에서 제공하는 것을 사용하다 보니 모든 운영체제에서 동일한 모양의 UI 구현이 가능하게 되었다.

나. 스윙은 AWT 보다 디자인이 세련되었다.

3. JAVA FX

가. JAVA FX는 어도비의 플래쉬와 MS사의 실버라이트의 대항마로 만들어 졌다. 플래쉬가 웹 광고용으로만 사용되다가 멀티미디어 스트리밍 시장을 점령하게 되자, MS는 이에 대항하기 위해 실버라이트를 내놓았다.

나. 썬마이크로시스템사도 이에 뒤질세라 2007년에 JAVA FX 1.0 이라는 새로운 기술을 소개했다. JAVA FX 1.0은 JAVA FX 스크립트언어로 개발된 프로그램을 자바가상머신에서 실행되는 구조로 되어 있었다. 이 언어는 애니메이션과 시각적인 효과를 내는데 최적화된 언어였으나 프로그래머들은 새로운 언어를 또 익혀야 하기 때문에 불평과 함께 멀리 했다.

다. 그래서 2011년 오라클사는 자바 언어외에 별도의 언어가 필요 없는 새로운 버전인 JAVA FX 2.0을 발표했다.자바 7업데이트 6버전부터 JDK와 JRE에 JAVA FX 2.2를 포함시켜 자바의 한 식구가 되었다.

라. 자바 8에서는 복잡한 버전 번호를 간략화해서 JAVA FX 8로 명칭을 변경하였다. JAVA FX 프로그램은 UI 생성, 이벤트 처리, 멀티미디어 재생, 웹 뷰 등과 같은 기능은 JAVA FX API로 개발하고 그 이외의 기능은 자바 표준 API를 활용해서 개발할 수 있다.

마. JAVA FX는 화면 레이아웃과 스타일, 프로그램 로직을 분리할 수 있기 때문에 디자이너와 프로그래머들이 협력해서 JAVA FX 애플리케이션을 개발할 수 있는 구조를 가지고 있다. 자바 코드와 분리해서 CSS로 외관을 작성할 수 있기 때문에 디자이너가 CSS로 작업할 동안 프로그래머들은 로직에 더 많은 시간을 할당할 수 있다. 자바코드에서 레이아웃(UI)과 로직을 분리하고 싶다면 레이아웃은 FXML 파일로 작성하고, 로직은 자바코드로 작성하면 된다.

바. JAVA FX 프로그램 구성요소

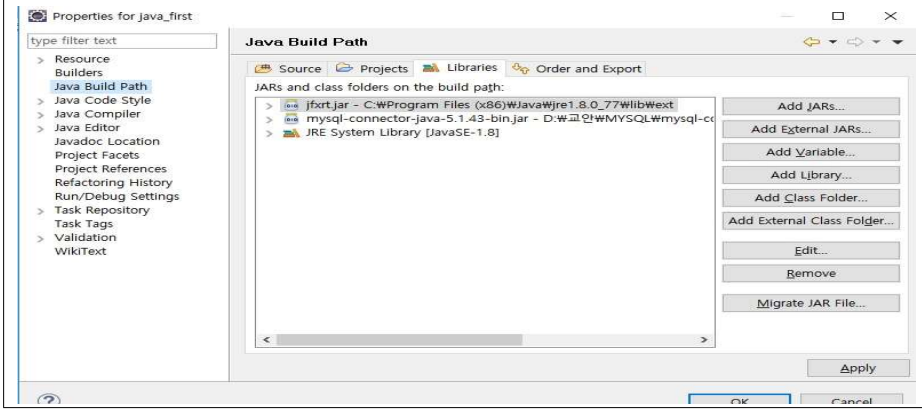
[레이아웃]	[외관 및 스타일]	[로직]	[리소스=>자원]
자바 코드 파일 또는 FXML	CSS 파일	자바언어	이미지,동영상 파일

* JavaFX : 이클립스 연동하는 법

자바로 GUI 나 그래픽, 멀티미디어 기반환경의 어플리케이션을 구축하는데 사용해왔던 SWING 에 이어 JavaFX 가 주목을 받고 있는 듯 하다. 과거에는 별도로 라이브러리 다운 받아서 설치하고 했었지만 JDK 1.7 버젼 이후 (현재 1.8xx) 는 JavaFX 가 포함되어 배포되고 있다. 과거에 이클립스 사용자들이 market place 에서 플러그인 설치하고 sdk 지정하는 번거로움은 많이 사라졌다.

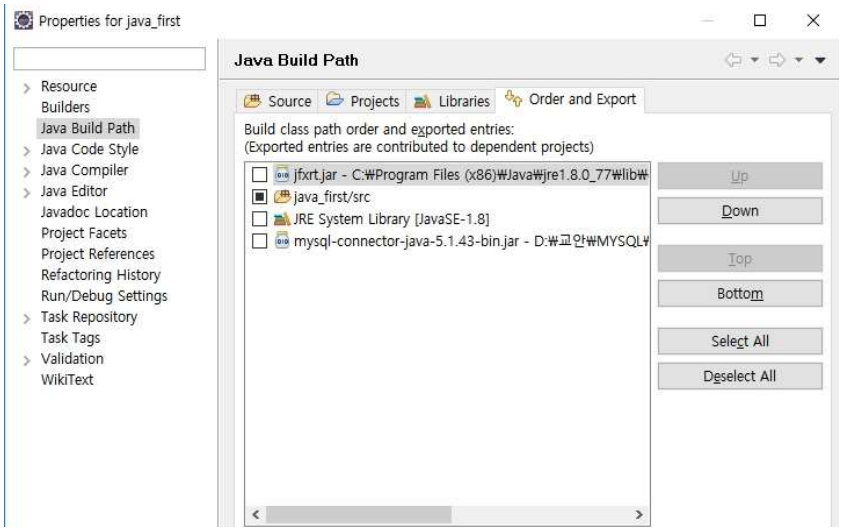
1. 우선 JDK 와 이클립스를 설치한다.(이는 생략)

2. 프로젝트 속성 (프로젝트 선택후에 우클릭-속성 혹은 Alt+Enter) 에서
   - 좌측에서 Java Build Path 선택
   - 가운데에서 Libraries 탭을 선택
   - 우측에서 Add External JARs... 선택



JAVA FX 라이브러리 jfxrt.jar를 추가한다.

3. jfxrt.jar 추가한뒤 꼭 잊지 말아야 할것은 Order and Export 탭에서 jfxrt.jar 의 우선순위를 최상단으로 올려주어야 한다.



* 최초 JAVA FX로 윈도우 창 만들기 소스 실습
소스파일명:Fx_01.java

```
import javafx.application.Application;
import javafx.stage.Stage;

public class Fx01 extends Application {
//Application 추상클래스로부터상속
@Override
public void start(Stage primaryStage)
throws Exception {
//start()메서드를 오버라이딩
primaryStage.show();//윈도우보여주기
}
public static void main(String[] args) {
launch(args);//메인Application을생성하고윈도우를생성하고
//start() 호출
/* 1. JAVA FX는윈도우무대를javafx.stage.Stage로표현
*/
}//main()
}
```

* 무대(Stage)와 장면(Scene) 만들기

소스파일명:Fx_02.java

```
import javafx.application.Application;
import javafx.application.Platform;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import javafx.stage.Stage;

public class Fx_02 extends Application {
	@Override
	public void start(Stage primaryStage)
			throws Exception {
		//무대는 한 번에 하나의 장면(Scene)을 가질수 있다.
		VBox root=new VBox();//컴포넌트를 수직으로 배치하는 레이아웃
		//객체 생성
		root.setPrefWidth(380);//VBox 폭
		root.setPrefHeight(150);//높이
		root.setAlignment(Pos.CENTER);
		//컨트롤 중앙정렬
		root.setSpacing(20);
		//컨트롤 수직간격

		Label label=new Label();
		label.setText("Hello. JAVA FX");
		label.setFont(new Font(50));

		Button button=new Button();
		button.setText("확인");
		button.setOnAction(event->Platform.exit());
		//버튼이벤트를 람다식으로 처리. 확인버튼 클릭하면
		//윈도우창 종료

		root.getChildren().add(label);
		//VBox레이아웃 자식노드로 레이블 추가
		root.getChildren().add(button);
```

```
                Scene scene=new Scene(root);
                //레이아웃을 가지고 장면 객체 생성

                primaryStage.setTitle("프로그램적 FX연습");
                //윈도우 제목 설정
                primaryStage.setScene(scene);
                //장면을 윈도우 무대에 올림
                primaryStage.show();
                //윈도우 화면 보여주기
        }//윈도우 무대를 Stage로 표현
        public static void main(String[] args) {
       launch(args);
        }//main()
}
```

* JAVA FX 레이아웃 만드는 2가지 방법

1. 첫 번째 방법은 자바 코드로 작성하는 프로그램적 레이아웃 방법이다. 자바 코드에 익숙한 개발자들이 선호하는 방식으로 컨트롤 배치, 스타일 지정, 이벤트 처리 등을 모두 자바 코드로 작성한다. 레이아웃이 복잡해 지면 프로그램적 방법은 코드가 복잡해져 난해한 프로그램이 될 확률이 높다. 또한 모든 것을 개발자가 직접 작성해야 하므로 디자이너와 협력해서 개발하는 것도 어렵다. 개발이 완료 된후에 간단한 레이아웃 변경이나 스타일 변경이 필요하면 자바 소스를 수정하고 재 컴파일 해야 한다. 지금까지는 이 방법을 사용하여 실습을 하였다.

2. 두 번째 방법은 FXML로 작성하는 레이아웃 방법이다. FXML은 XML 기반의 언어로 UI 레이아웃을 자바코드에서 분리해서 태그로 해서 작성한다. 안드로이드에서도 XML로 레이아웃을 작성하고, 자바로 이벤트 처리와 로직코드를 작성한다. FXML 태그로 레이아웃을 정의하기 때문에 태그에 익숙한 디자이너와 협업이 가능하다. 또한 개발이 완료된 후 간단한 레이아웃변경도 쉽게 할 수 있다. 개발기간이 단축되는 효과가 발생한다.

* FXML로 작성하는 레이아웃 만들기

FXML 파일명: root.fxml

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

<?import javafx.scene.layout.HBox?>
<?import javafx.geometry.Insets?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.control.Button?>
<!-- FXML 임포트 -->
<HBox xmlns:fx="http://javafx.com/fxml/1">
```

```
<!-- HBox는 수평으로 배치하는 레이아웃 -->
        <padding>
         <!-- 안쪽 여백 설정 -->
         <Insets top="10" right="10" bottom="10" left="10" />
         <!-- 시계방향으로 위,오른쪽,아래쪽,왼쪽 안여백 설정 -->
        </padding>
        <spacing>10</spacing>
        <!-- 컨트롤 간의 수평간격 -->

        <children>
        <!-- 자식컨트롤 추가 -->
         <TextField>
          <!-- 한줄 입력박스 컨트롤 -->
          <prefWidth>200</prefWidth>
          <!-- 입력박스 폭 -->
         </TextField>

         <Button>
          <text>확인</text>
          <!-- 버튼 위 캡션문자열 -->
         </Button>
        </children>
</HBox>
```

* FXML 로딩과 Scene 생성

소스파일명:Fx_03.java

```
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class Fx_03 extends Application {

        @Override
        public void start(Stage primaryStage)
                        throws Exception {
                Parent root=FXMLLoader.load(getClass().getResource("root.fxml"));
                //getClass()는 현재클래스, FXML파일을 로드한다.
                Scene scene=new Scene(root);
```

```
                primaryStage.setTitle("FXML 레이아웃");
                primaryStage.setScene(scene);//무대위
                //장면설정
                primaryStage.show();//윈도우 표시
        }
        public static void main(String[] args) {
                launch(args);
        }
}
```

* FXML 컨트롤 이벤트 처리

FXML 파일명: root2.fxml

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

<?import javafx.scene.layout.HBox?>
<?import javafx.scene.control.Button?>

<HBox xmlns:fx="http://javafx.com/fxml/1"
 fx:controller="fx.RootController"
 prefHeight="300.0" prefWidth="500.0"
 alignment="CENTER" spacing="20.0" >
 <!-- FXML파일당 별도의 컨트롤러를 지정해서 이벤트를 처리,
 FXML 레이아웃에서 이벤트 처리 코드를 완전히 분리 -->
        <children>
         <Button  fx:id="btn1" text="버튼1" />
         <Button  fx:id="btn2" text="버튼2" onAction="#handleBtn2Action" />
<!-- 컨트롤에서 이벤트핸들러를 등록하지 않고, onAction="#handleBtn2Action"
을 추가하면 2번째 버튼 이벤트 발생시 컨트롤클래스에 정의된 handleBtn2Action
()메서드를 호출한다. onAction="#메서드명“ 이 온다. -->
         <Button  fx:id="btn3" text="버튼3" />
         <!-- 컨트롤 클래스의 @FXML 어노테이션이 적용된 참조변수명(필드명)
         과 FXML의 fx:id속성값과 일치해야 한다.  -->
        </children>
</HBox>
```

이벤트 처리 컨트롤 클래스명:RootController.java

```
package fx;

import java.net.URL;
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;

public class RootController implements Initializable {
	@FXML private Button btn1;
	@FXML private Button btn2;
	@FXML private Button btn3;

	@Override
	public void initialize(URL location, ResourceBundle res) {
		btn1.setOnAction(new EventHandler<ActionEvent>() {

			@Override
			public void handle(ActionEvent event) {
				handleBtn1Action(event);

			}
		});//첫번째 버튼 이벤트 등록(내부무명클래스)

		//btn2.setOnAction(event->handleBtn2Action(event));
		btn3.setOnAction(event->handleBtn3Action(event));//각 버튼 이벤
트 등록
	}//initialize()

	public void handleBtn1Action(ActionEvent event){
		System.out.println("버튼 1 클릭");
	}//첫번째 이벤트 처리
	public void handleBtn2Action(ActionEvent event){
		System.out.println("버튼 2 클릭");
	}//두번째 이벤트 처리
	public void handleBtn3Action(ActionEvent event){
		System.out.println("버튼 3 클릭");
```

```
        }//세번째 이벤트 처리
}//Root class
```

* JAVA FX CSS 스타일

1. JAVA FX UI를 담당하는 컨테이너 및 컨트롤은 CSS(Cascading Style Sheets)를 적용해서 모양 및 색상을 변경할 수 있다. 이것은 HTML에 CSS를 적용하는 것과 유사하다.

2. CSS 적용하는 방법은 인라인 스타일과 외부 CSS 파일 적용법이 있다.

*** 먼저 인라인 스타일 법**

1. 이 방법은 컨테이너 또는 컨트롤에 스타일 속성값을 직접 정의하기 때문에 쉽고,빠르게 모양과 색상을 변경할 수 있다.

2. 하지만 동일한 스타일을 적용하는 컨테이너와 컨트롤이 많아질수록 중복 코드가 늘어나는 단점이 있다. 또한 레이아웃과 CSS가 뒤섞여 추후 유지보수가 어렵다.

FXML 파일명: root3.fxml

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

<?import javafx.scene.layout.HBox?>
<?import javafx.scene.control.Label?>

<HBox xmlns:fx="http://javafx.com/fxml/1"
  prefHeight="100" prefWidth="400" alignment="CENTER"
  spacing="20" >

  <children>
    <Label text="검정바탕,노란글씨"
    style="-fx-background-color:black;-fx-text-fill:yellow;
    -fx-padding:5;" />
    <!-- -fx-background-color속성은 배경색,
         -fx-text-fill속성은 글자색, -fx-padding속성은 안쪽여백 -->
    <Label text="파란바탕,흰글씨"
    style="-fx-background-color:blue; -fx-text-fill:white;
    -fx-padding:5;" />
    <Label text="파란바탕,흰글씨"
    style="-fx-background-color:blue; -fx-text-fill:white;
    -fx-padding:5;" />
  </children>
</HBox>
```

*** 외부 CSS 파일 적용법**

1. 인라인 스타일 방법보다는 중복 코드를 줄이고, 재사용성을 높이면서 유지보수도 편리하다. 그러므로 외부 CSS 파일을 작성해서 사용하는 것이 좋다.

FXML 파일명: root4.fxml

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

<HBox xmlns:fx="http://javafx.com/fxml/1"
prefHeight="100" prefWidth="400" alignment="CENTER"
spacing="20">

 <children>
  <Label id="id01" text="검정바탕,노란글씨" />
  <!-- id선택자는 유일해야 한다. 하나의 컨트롤만 선택할 수 있는 방법. -->
  <Label styleClass="idClass" text="파란바탕,흰글씨" />
  <!-- styleClass의 클래스 선택자는 중복이름을 가질수 있다.class 선택자는
  동시에 여러 컨트롤을 선택할 수 있다. -->
  <Label styleClass="idClass" text="파란바탕,흰글씨" />
 </children>
</HBox>
```

css 파일명: app.css

```
/*타입 선택자로 Label선택*/
Label{
        -fx-padding:5;
}
/* 아이디 선택자 */
#id01{
        -fx-background-color:black;
        -fx-text-fill:yellow; /*글자색을 노랑*/
}
/* 클래스 선택자 */
.idClass{
        -fx-background-color:blue;
        -fx-text-fill:white;
}
```

java 파일명: Fx_06.java

```
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class Fx_06 extends Application {

	@Override
	public void start(Stage stage) throws Exception {
		Parent
root=(Parent)FXMLLoader.load(getClass().getResource("root4.fxml"));
		Scene scene=new Scene(root);

scene.getStylesheets().add(getClass().getResource("app.css").toString());
		//장면에 css적용하기

		stage.setTitle("외부스타일 css적용");
		stage.setScene(scene);
		stage.show();
	}
	public static void main(String[] args) {
	launch(args);
	}
}
```