Java FX

Uvod

Opšte

- Biblioteka za rad sa grafičkim komponentama
- Naslednik starijih tehnologija Swing-a i AWT-a
  - Bolje razdvojena logika od prezentacije
  - Mogućnost korišćenja tema i CSS-a
  - Ujednačen izgled na različitim platformama i uređajima
- Podržana počev od Jave Verzije 7

Instalacija i HelloFX

1. Napraviti klasičan Java projekat
2. U Build Path-u dodati novi spoljni **jar**:

   <JavaFX SDK Home directory>\rt\lib\ext\jfxrt.jar
3. Naslediti klasu javafx.application.Application
4. U main metodi ove klase pokrenuti aplikaciju pozivom na:

   Application.launch(args)
5. Dodati grafičke komponente

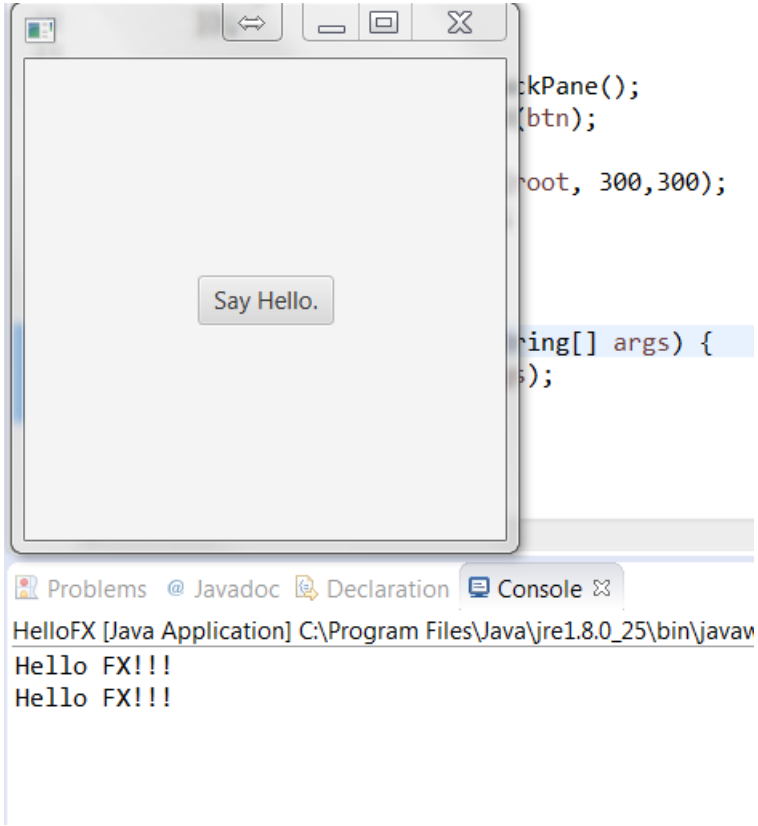
Evo kako to izgleda:

```
package fxtest;

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

public class HelloFX extends Application {

        @Override
        public void start(Stage stage) throws Exception {

        }

        public static void main(String[] args) {
                Application.launch(args);
        }
}
```

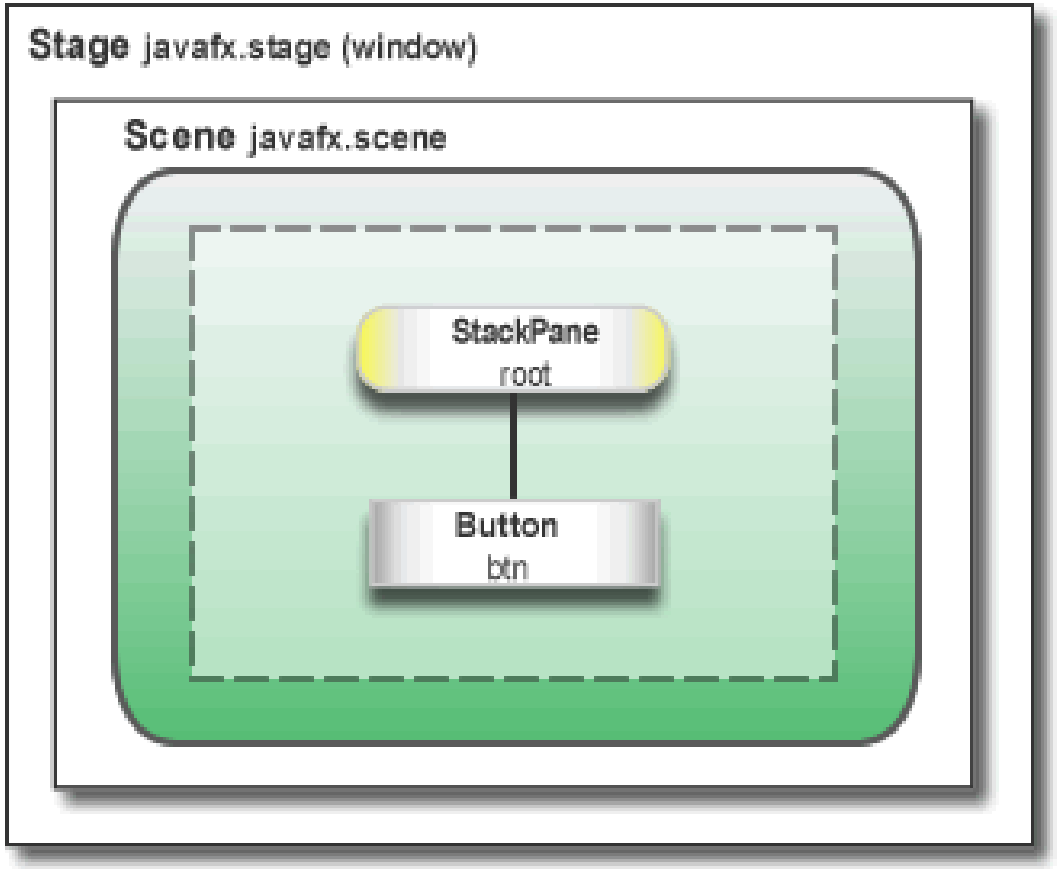
Hello FX

```
public class HelloFX extends Application {

	@Override
	public void start(Stage stage) throws Exception {
		Button btn=new Button("Say Hello.");
		//Reagovanje na događaj
		btn.setOnAction(new EventHandler<ActionEvent>() {
			@Override
			public void handle(ActionEvent arg0) {
				System.out.println("Hello FX!!!");
			}
		});
		//Kontejner za komponente
		StackPane root=new StackPane();
		root.getChildren().add(btn); //Dodavanje u kontejner
		/*Priprema scene sa zadatim dimenzijama i datim kontejnerom
		komponenti */
		Scene scene=new Scene(root, 300,300);
		//Postavljanje scene na pozornicu i prikazivanje
		stage.setScene(scene);
		stage.show();
	}

	public static void main(String[] args) {
		Application.launch(args);
	}
}
```

Nakon pokretanja

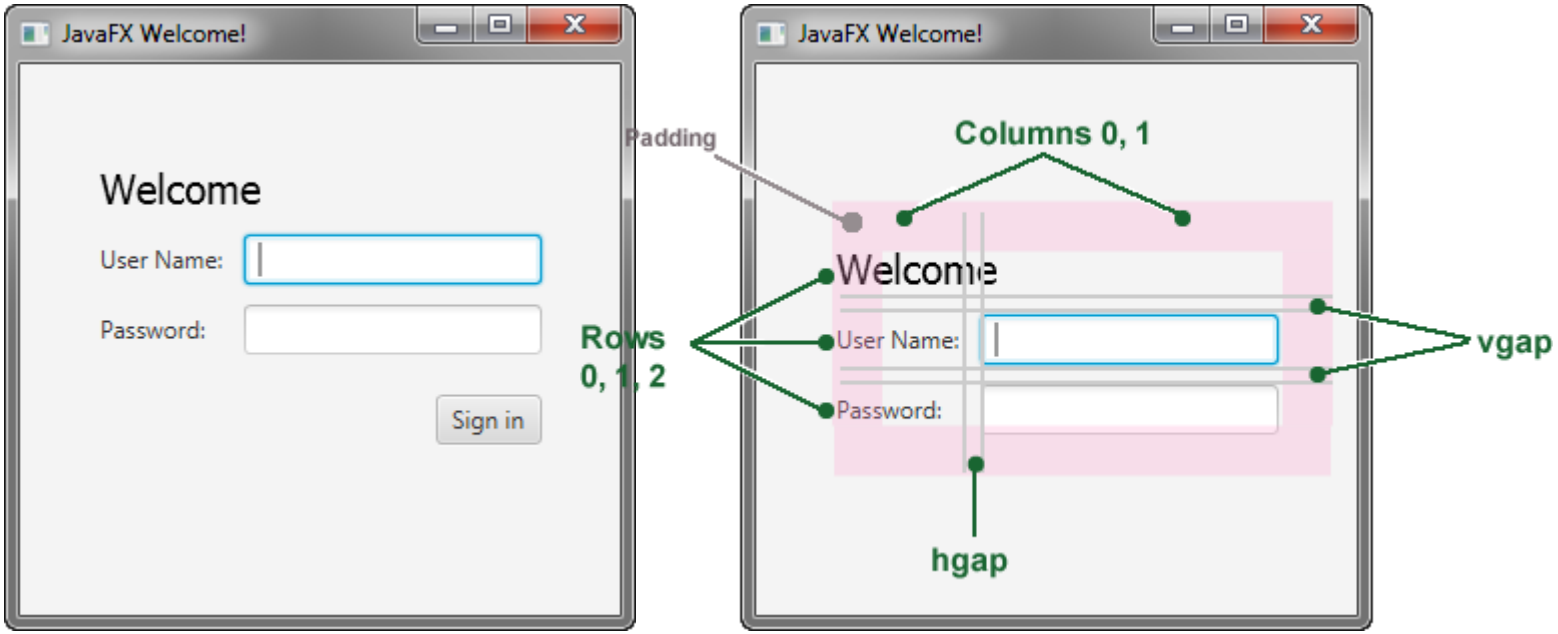


Scene i pozornica



Kreiranje formi za unos

- Forme za unos obično imaju tabelarnu organizaciju komponenti
- Npr. hoćemo ovako nešto:



Organizatori komponenti - Layouts

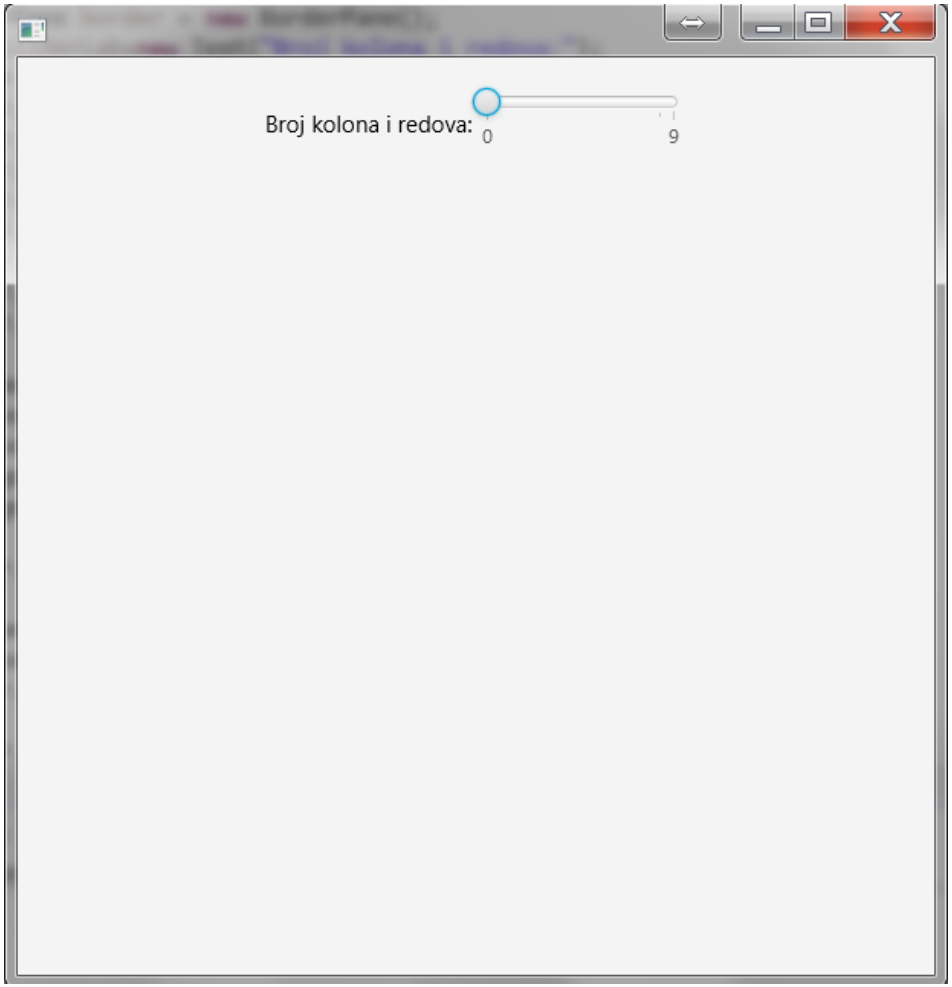
- Postoje različiti organizatori komponenti:
  - StackPane - jedan iznad drugog razvučeno
  - **GridPane – tabelarna organizacija**
  - HBox, VBox – sve u jednom redu (jednoj koloni)
  - FlowPane – sa leva na desno, odozgo na dole
  - TilePane – isto kao Flow, samo podjednake dimenzije
  - BorderPane – 5 regiona (gornji, donji, levi, desni, centar)

Forma za unos - kod

```
GridPane grid=new GridPane();
grid.setAlignment(Pos.CENTER);
Text scenetitle = new Text("Welcome");
scenetitle.setFont(Font.font("Tahoma", FontWeight.NORMAL, 20));
grid.add(scenetitle, 0, 0, 2, 1); //0-ta kolona, 0-ti red
//2 označava da se komponenta prostire preko 2 kolone
//1 označava da se prostire po 1 redu što je podrazumevano

Label userName = new Label("User Name:");
grid.add(userName, 0, 1);
TextField userTextField = new TextField();
grid.add(userTextField, 1, 1);
Label pw = new Label("Password:");
grid.add(pw, 0, 2);
PasswordField pwBox = new PasswordField();
grid.add(pwBox, 1, 2);
```

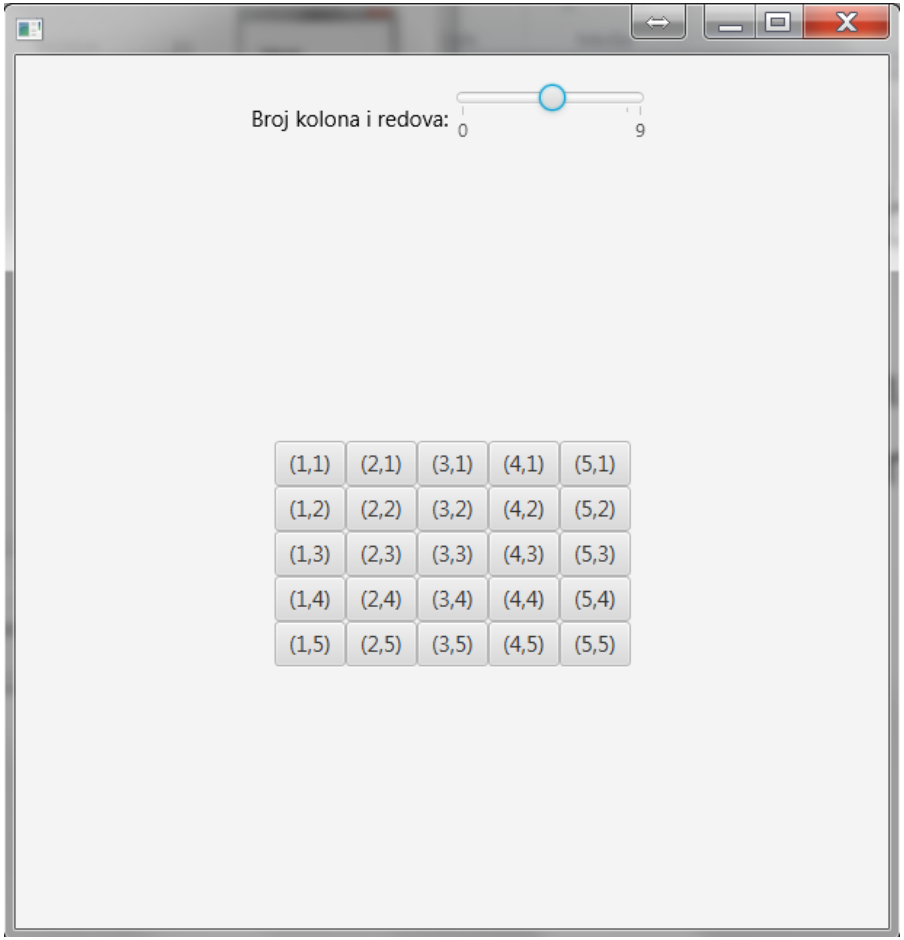
Zadatak za vežbu

- Napraviti interfejs sa sledećim inicijalnim izgledom (prozor 600 x 600)



Zadatak za vežbu

- Vrednost slajdera definiše dimenziju matrice dugmića (max=9). Nakon promene se dešava ovo sa slike.
- Potrebno je upotrebiti Border, Grid i Flow Pane (videti gde koji)...



Pogledati detalje o upotrebi slajdera na http://docs.oracle.com/javafx/2/ui_controls/slider.htm

M – Model (Logika)
V – View (Prezentacija)
C – Control (Kontrola)

- Recimo da treba da se napravi program za traženje najboljeg puta na mapi grada.
- Hoćemo da imamo sledeće mogućnosti:
  - Da obeležimo na mapi početnu i krajnju lokaciju
  - Da odaberemo prevozno sredstvo
  - Da se izvrši algoritam traženja najboljeg puta
  - Da nam se trajektorija lepo prikaže na mapi
  - Da se to čuva u nekoj istoriji naših pretraga
  - Da sve ovo možemo da radimo sa različitih uređaja: desktop, mobilni telefon, tablet pc itd.

M – Model (Logika)
V – View (Prezentacija)
C – Control (Kontrola)

- Recimo da treba da se napravi program za traženje najboljeg puta na mapi grada.
- Hoćemo da imamo sledeće mogućnosti:
  - (C) Da obeležimo na mapi početnu i krajnju lokaciju
  - (C) Da odaberemo prevozno sredstvo
  - (M) Da se izvrši algoritam traženja najboljeg puta
  - (V) Da nam se trajektorija lepo prikaže na mapi
  - (M) Da se to čuva u nekoj istoriji naših pretraga
  - (V) Da sve ovo možemo da radimo sa različitih uređaja: desktop, mobilni telefon, tablet pc itd.

FXML

- Problem sa prethodnim kodovima:
  - Kod za logiku isprepleten sa kodom za kontrolu i prezentaciju (definisanje komponenti, boja, dimenzija, logika za reagovanje na događaje itd.)

- FXML je kod koji se odnosi SAMO na prezentaciju, i piše se odvojeno od .java datoteke

Pre nego što nastavimo, instalacija

- Podrška za FXML u 2 varijante:
  1. Eclipse verzija koja to već sve sadrži

     preuzeti sa: http://www.eclipse.org/efxclipse/install.html
  2. Ili dodavanje Plugin-a u već postojećoj Eclipsi:

     a) Otvoriti postojeću Eclipsu

     b) Help→Install new software

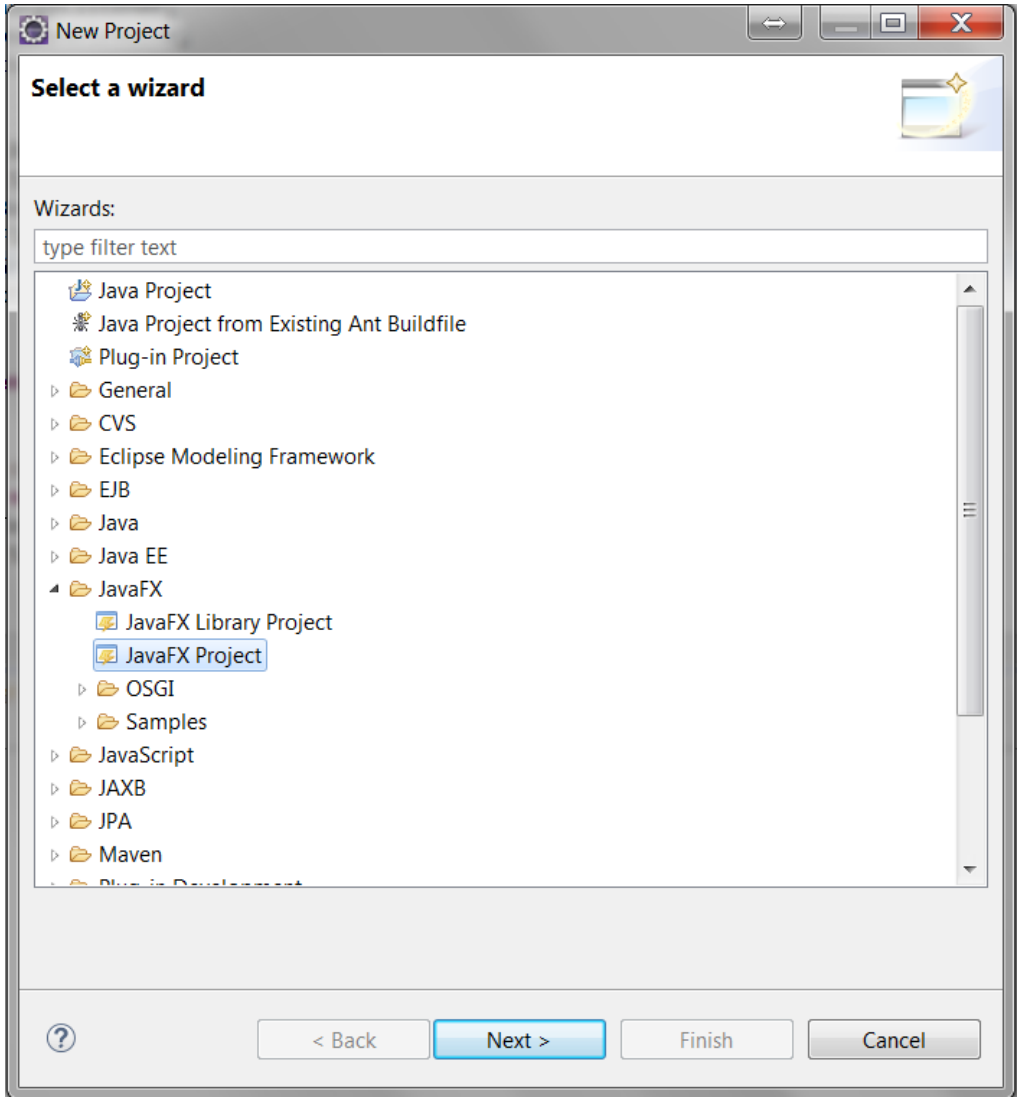
     c) U prostor za lokaciju sajta ukucati:

     http://download.eclipse.org/efxclipse/updates-released/1.2.0/site

     d) Pa dodati sajt (Add..) – i potom nastaviti sa instalacijom

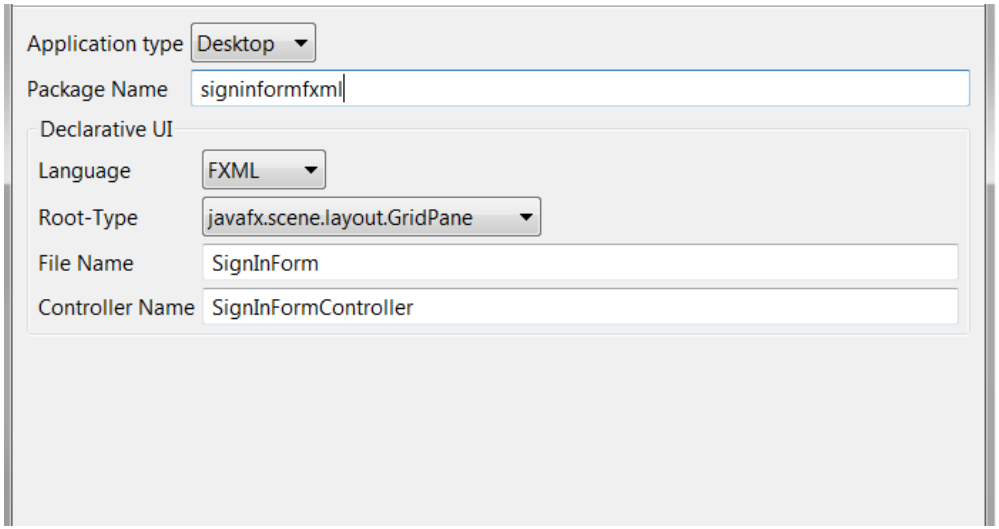
Kreiranje JavaFX projekta

- Nakon instalacije i resetovanja Eclipse, trebala bi da se pojavi sledeća opcija.
- Odaberite JavaFX Project, pa next, next...



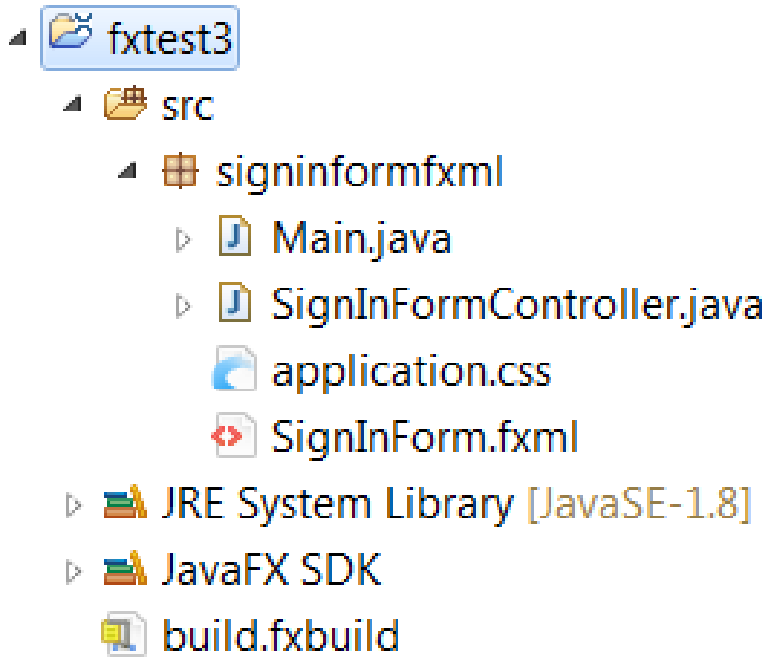
Kreiranje JavaFX paketa i FXML datoteke

- U nekom momentu kasnije, odaberite ovu opciju →
- I popunite osatle tražene informacije



Nakon kreiranja sledeći elementi

1. Main datoteka
   za pokretanje
2. SignInFormController
   za logiku i kontrolu
3. SingInForm.fxml
   za prezentacioni kod
4. application.css
   za „došminkavanje“



Sređivanje FXML datoteke

- Nazivi komponenti i polja slična kao u pristup iz java datoteke
- S tim što je sad sve u xml formatu:

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.layout.GridPane?>

<GridPane xmlns:fx="http://javafx.com/fxml/1"
fx:controller="signinformfxml.SignInFormController">
<!– Ovo je komentar-->
</GridPane>
```

Dodajemo sve importe zasad...

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

<?import java.net.*?>
<?import javafx.geometry.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.text.*?>

<GridPane xmlns:fx="http://javafx.com/fxml/1"
fx:controller="signinformfxml.SignInFormControl
ler">
<!--  U regiji između otvorenog i zatvorenog
GridPane taga upisujemo sadržaj -->
</GridPane>
```

Dodajemo komponente u GridPane

```
<Text text="Welcome" GridPane.columnIndex="0"
GridPane.rowIndex="0"
GridPane.columnSpan="2" />

<Label text="User Name:" GridPane.columnIndex="0"
GridPane.rowIndex="1" />

<TextField GridPane.columnIndex="1"
GridPane.rowIndex="1" />

<Label text="Password:" GridPane.columnIndex="0"
GridPane.rowIndex="2" />

<PasswordField fx:id="passwordField"
GridPane.columnIndex="1" GridPane.rowIndex="2" />
```

Dodajemo osluškivač na dugme

```
<HBox spacing="10" alignment="bottom_right"
GridPane.columnIndex="1"
GridPane.rowIndex="4">
<!-- ovako se povezujemo sa metodom iz kontroler klase -->
<Button text="Sign In" onAction="#handleSubmitButtonAction" />
</HBox>

<Text fx:id="actiontarget" GridPane.columnIndex="0"
GridPane.columnSpan="2" GridPane.halignment="RIGHT"
GridPane.rowIndex="6" />
```

Povezivanje sa događajem klika u java kodu

```
package signinformfxml;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.text.Text;

public class SignInFormController {
        //označavamo da je reč o FXML polju - prepoznavanje se vrši po nazivu
        @FXML private Text actiontarget;

//ovo označava da je reč o FXML metodi - prepoznavanje se vrši po nazivu
        @FXML
        protected void handleSubmitButtonAction(ActionEvent event) {
                actiontarget.setText("Sign in button pressed");
        }
}
```