



JavaFX

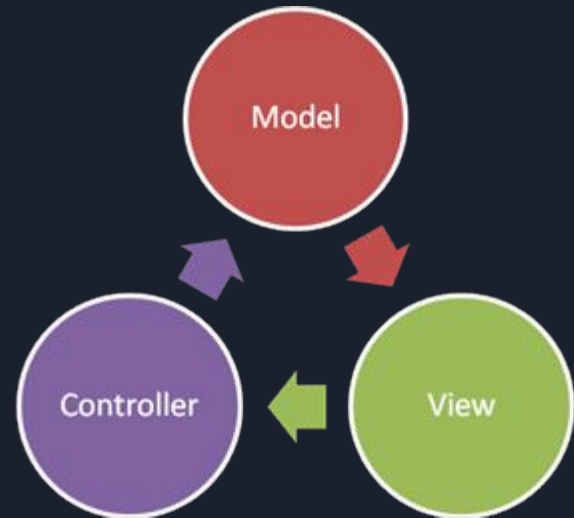
wprowadzenie



Java ma wbudowane pakiety, które pomagają nam stworzyć aplikację z interfejsem graficznym. Dopiero od Javy 8, pakiety JavyFX są dołączone do JDK. Dzięki tym pakietom jesteśmy w stanie w łatwy i szybki sposób stworzyć w pełni funkcjonalne rozwiązanie.

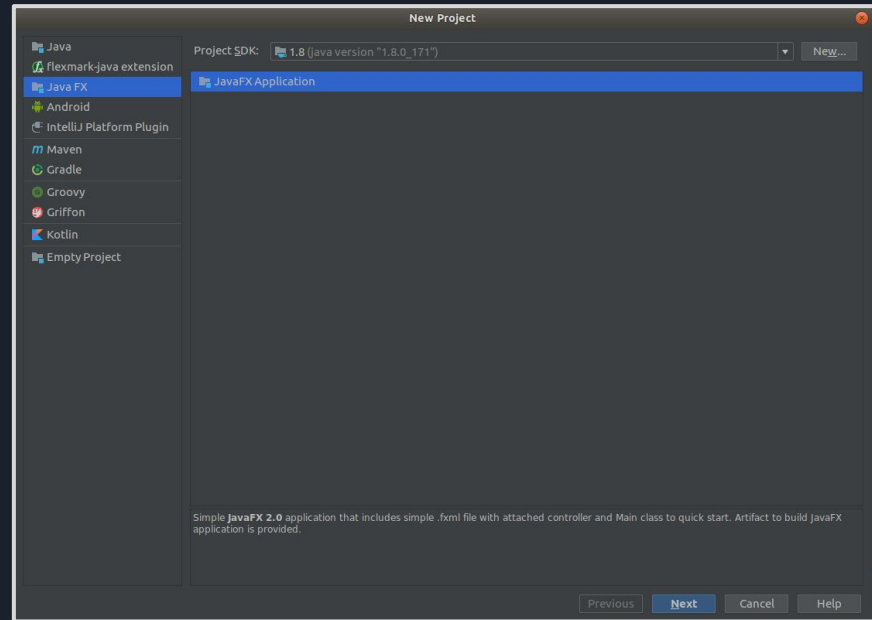
Model Widok Kontroler

Większość aplikacji z graficznym interfejsem dzieli zależności na model, widok i kontroler. Dzięki temu nasza aplikacja jest modularna i porównywalnie niskim kosztem możemy zmienić to jak nasza aplikacja wygląda, bądź się zachowuje. Dzięki takiemu rozdzieleniu GUI jest **prawie niezależne** od kodu który jest rdzeniem i logiką naszej aplikacji.



Tworzenie prostej aplikacji JavaFX

Przy tworzeniu aplikacji typu JavaFX można posłużyć się konkretną opcją w **menu tworzenia nowego projektu**.

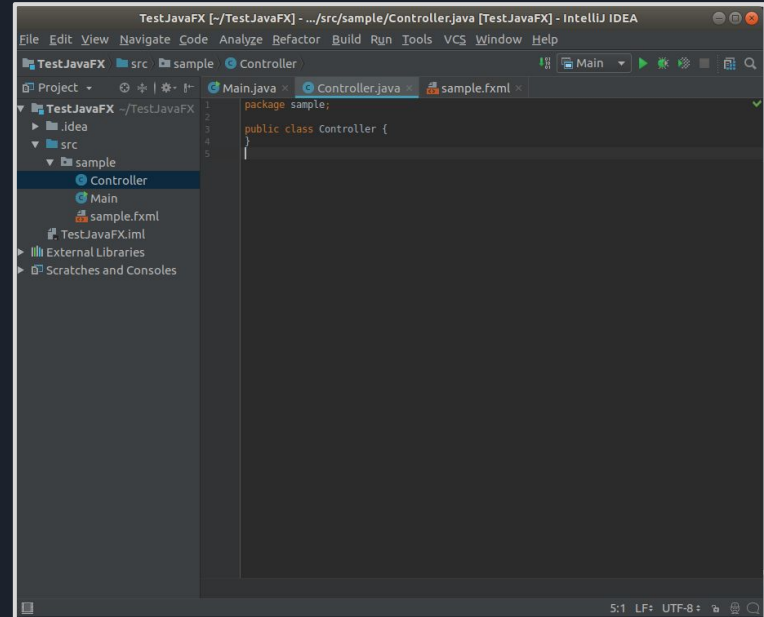


Pliki zawarte w przykładowym projekcie JavaFX

Jak widać na zrzucie ekranu, domyślnie IntelliJ tworzy nam trzy pliki.

- Controller.java
- Main.java
- sample.fxml

Jest to odwzorowanie MWK. W tym przypadku za **model** odpowiada przykładowa klasa **Main**, za **kontroler** klasa **Controller**, a za **widok** plik **sample.fxml**.

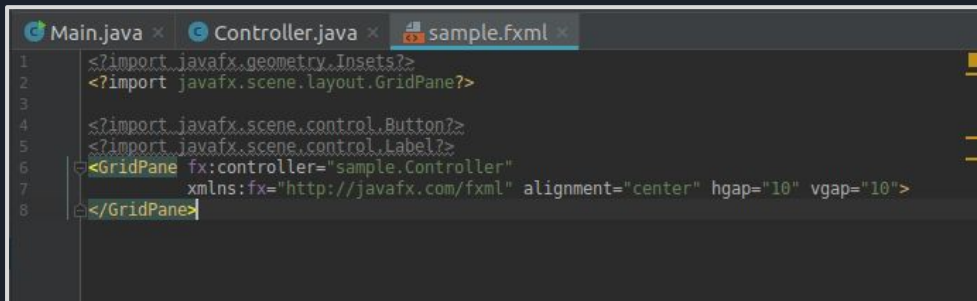


Widok

Plik sample.fxml

W domyślnie wygenerowanym pliku .fxml możemy zauważyć elementy **opisywania interfejsu graficznego**.

Przykładowym kontenerem jest tu **GridPane**, czyli można powiedzieć, że panel w postaci siatki. Jak widać w atrybutach tagu możemy zauważyć, że **podpinamy klasę**, która będzie kontrolować zachowanie interfejsu przez **fx:controller**.



```
1 <?import javafx.geometry.Insets?>
2 <?import javafx.scene.layout.GridPane?>
3
4 <?import javafx.scene.control.Button?>
5 <?import javafx.scene.control.Label?>
6 <GridPane fx:controller="sample.Controller"
7         xmlns:fx="http://javafx.com/fxml" alignment="center" hgap="10" vgap="10">
8 </GridPane>
```

Kolejnym istotnym atrybutem jest **xmlns:fx** który jest **wymagany** i określa przestrzeń nazw dla całego pliku. Reszta atrybutów opisuje rozmieszczenie całej siatki i odległości między kolumnami i wierszami poszczególnych komórek siatki.

Kontroler klasa Controller

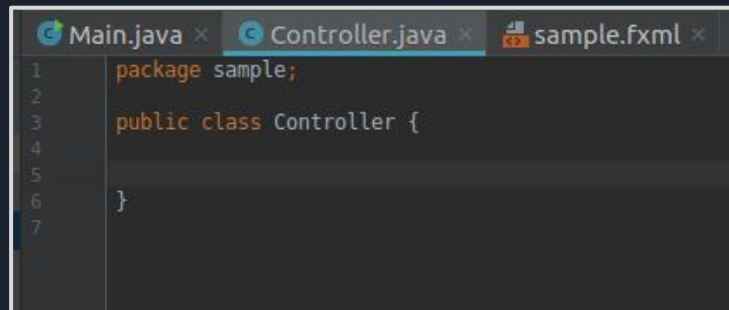
Na początku pusta klasa Controller. Odpowiada ona za jak już wcześniej zaznaczono podpinanie konkretnych zależności do interfejsu graficznego. Można to głównie osiągnąć dzięki adnotacji **@FXML** która łączy nam **konkretny znacznik** (musi być opatrzony takim samym **fx:id** jak nazwa pola w kontrolerze) z **polem w klasie Controller**.

```
import javafx.fxml.FXML;
import javafx.scene.control.Button;

public class Controller {

    @FXML
    Button someButton;

}
```



```
Main.java x Controller.java x sample.fxml x
1 package sample;
2
3 public class Controller {
4
5
6 }
7
```

Jak możemy zauważyć, w atrybutach przycisku występują **odniesienia do GridPane**, tak właśnie ustawia się **ułożenie przycisku na panelu siatki**. Ustalając numer kolumny i wiersza.

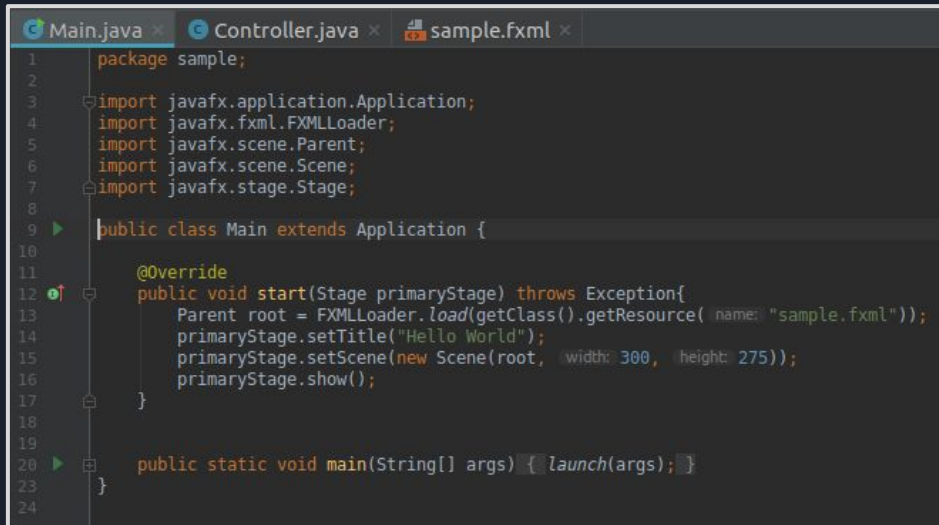
```
<GridPane fx:controller="sample.Controller"
  xmlns:fx="http://javafx.com/fxml" alignment="center" hgap="10" vgap="10">

  <Button fx:id="someButton" text="Some button" GridPane.columnIndex="0" GridPane.rowIndex="0"
  |
```

Model

klasa startowa aplikacji: Main

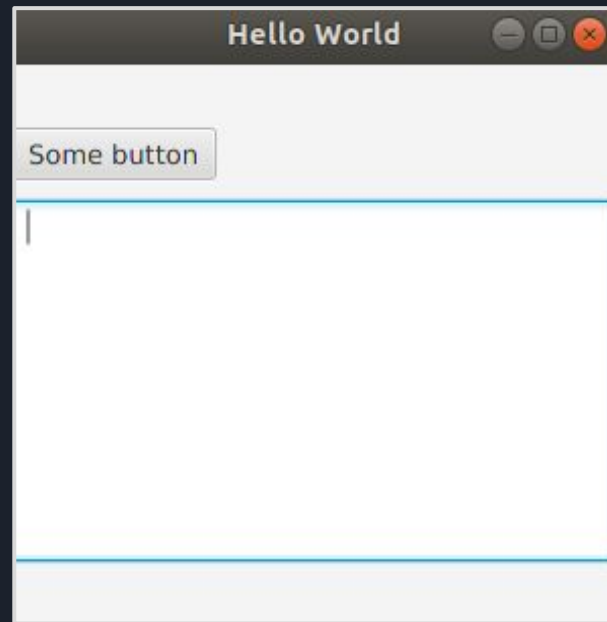
Od razu można zauważyć, że klasa główna z metodą main zdecydowanie różni się od tej, z aplikacji konsolowej. W tym wypadku mamy jeszcze **metodę start**, która pozwala nam na ustawienie pliku, w którym opisany jest **widok**, **nazwy sceny** i innych jej parametrów. Jeśli chcemy zmienić **rozmiar okna**, czy **nazwę na pasku aplikacji**, to właśnie tutaj.



```
1 package sample;
2
3 import javafx.application.Application;
4 import javafx.fxml.FXMLLoader;
5 import javafx.scene.Parent;
6 import javafx.scene.Scene;
7 import javafx.stage.Stage;
8
9 public class Main extends Application {
10
11     @Override
12     public void start(Stage primaryStage) throws Exception{
13         Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));
14         primaryStage.setTitle("Hello World");
15         primaryStage.setScene(new Scene(root, width: 300, height: 275));
16         primaryStage.show();
17     }
18
19
20     public static void main(String[] args){ launch(args); }
21
22 }
23
24 }
```


Okno aplikacji

Po prawej można zobaczyć efekt przykładowej aplikacji. W oknie programu widzimy prosty przycisk i pole tekstowe typu `TextArea`.





Podpinanie metody pod przycisk

Aby podpiąć metodę pod dany przycisk, należy dodać temu przyciskowi atrybut np. `onAction` który następnie wskaże nam nazwę metody w kontrolerze, którą chcemy uruchomić. Należy pamiętać aby nazwę w pliku `FXML` poprzedzić znakiem `#`.

```
<Button fx:id="someButton" text="Some button" GridPane.columnIndex="0" GridPane.rowIndex="0" onAction="#doAction"/>
```

Zawartość pliku `sample.fxml`

```
@FXML
private void doAction() {
}
```

Zawartość pliku `Controller.java`

Różne elementy kontrolujące

Na podanych zrzutach ekranu można zobaczyć przykłady różnych obiektów, które mogą kontrolować informacje, bądź je przetrzymywać. Koniecznie należy zwrócić uwagę na fakt, by używać **pakietów javafx**, ponieważ istnieją również pakiety awt posiadające podobne klasy, takie jak **Button**.

```
import javafx.fxml.FXML;
import javafx.scene.control.*;
import javafx.scene.text.Text;
```

```
@FXML
private Button someButton;
@FXML
private RadioButton radioButton;
@FXML
private Text text;
@FXML
private Label label;
@FXML
private TextField textField;
@FXML
private TextArea textArea;
@FXML
private ListView<String> listView;
```



Metody obiektów kontrolujących w JavaFX

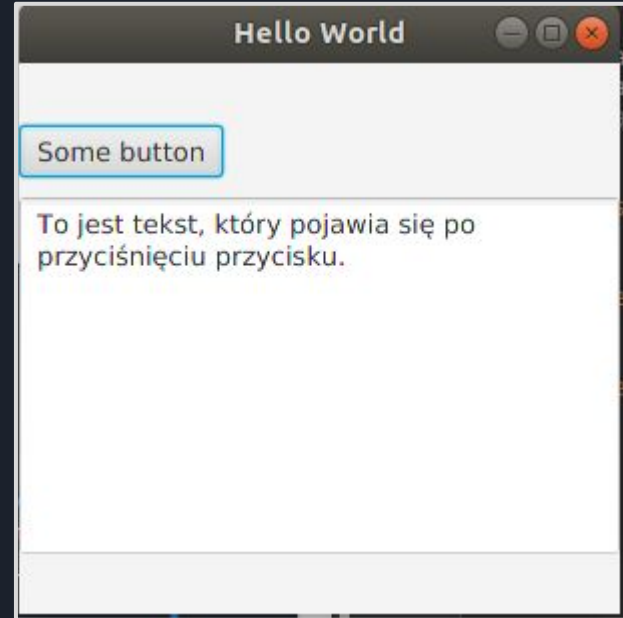
Dzięki użyciu adnotacji **@FXML** nasze pole bądź metoda łączą się z widokiem opisanym w pliku fxml. W ten sposób używając metod w klasie Controller możemy wykonać konkretną akcję po przyciśnięciu przycisku w interfejsie graficznym.

W tym celu wystarczy użyć jakiejś metody na obiekcie klasy przykładowo **Text**.

```
@FXML
private void doAction() {
    text.setStrikethrough(true);
}
```

Zadanie pierwsze

Zachęcam wszystkich zainteresowanych, do **przetestowania** przekazanych funkcjonalności i podjęcie próby utworzenia bardzo prostej aplikacji, która **po przyciśnięciu przycisku, wyświetla tekst na ekranie**.

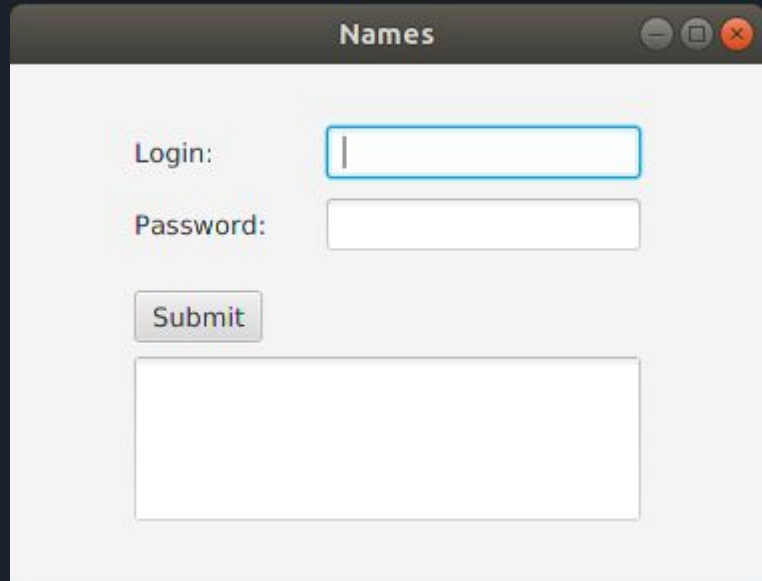


Podstawowy formularz

Po prawej można zobaczyć przykład podstawowego formularza. Po wpisaniu loginu i prawidłowego hasła, a następnie kliknięciu przycisku submit, dostajemy odpowiedź do pola tekstowego poniżej. W tym wypadku możecie wpisać swoje nazwisko (bez polskich znaków) i hasło, którym jest "java".

Program można pobrać w całości z repozytorium:

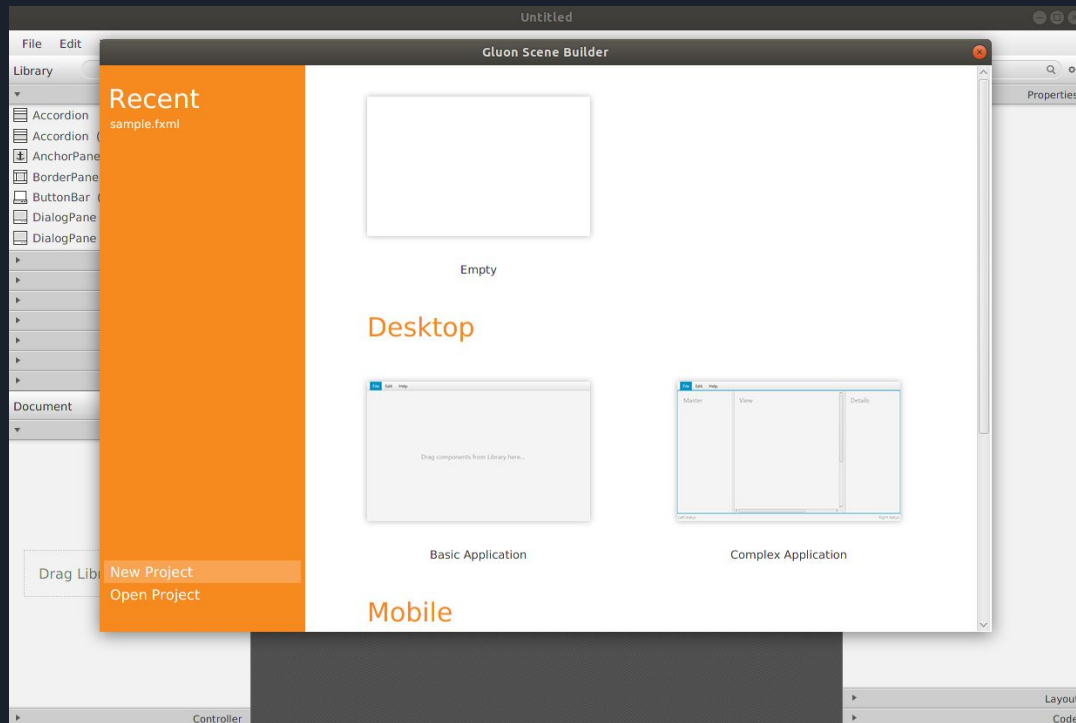
<https://github.com/AdrianHarenczyk/JavaFXWorkshops>



The image shows a JavaFX window titled "Names" with standard macOS window controls (minimize, maximize, close). Inside the window, there is a login form. It consists of two labels, "Login:" and "Password:", each followed by a text input field. The "Login:" field has a blue border and a cursor. Below these fields is a "Submit" button. At the bottom of the window is a large, empty text area for the response.

SceneBuilder

Program, który być może sprawdziłeś, był stworzony po części dzięki programowi SceneBuilder. Pomaga on tworzyć interfejs graficzny dla użytkownika. Polecam pobranie go bezpośrednio z linku: [SceneBuilder Download](#)





Zadanie drugie

Spróbuj odtworzyć poprzednią aplikację. Jeśli wolisz, możesz przeglądnąć kod rozwiązania, w celu lepszego zrozumienia go. Warto używać **Scene Builder'a** do szybkiego i łatwego budowania interfejsu graficznego, ale ważne, by nie zapominać jaki kod za tym stoi.



Podsumowanie

Na sam koniec tego wprowadzenia wysyłam parę linków, które mogą przydać się w dalszej nauce JavaFX, bądź dostarczyć ciekawych dodatkowych elementów nie zawartych standardowo w pakiecie.

- [ControlsFX](#) - projekt zawierający nowe elementy interfejsu.
- [FormsFX](#) - API pomagające tworzyć formularze.
- [Spring Boot z JavaFX](#) - tutorial jak używać razem tych dwóch technologii.
- [Simple Timer](#) - artykuł o tym, jak stworzyć prosty timer.
- [Calendar Heatmap](#) - projekt utworzenia prostej mapy informującej o ilości kontrybucji.
- [TilesFX](#) - repozytorium zawierające różne ciekawe elementy interfejsu użytkownika.
- [JavaFX dla Raspberry PI](#) - aplikacja stworzona w celu uruchomienia na Raspberry PI.
- [FXDocs](#) - repozytorium mające na celu tworzenie dokumentacji PDF do JavyFX.
- [Udemy Java Masterclass Course](#) - kurs do Javy w tym JavyFX, pozwala na dobre wejście do tematu i naukę krok po kroku. Cena zwykle jest promocyjna i oscyluje w granicach 35 zł.