Wykład Java

Paweł Jadanowski

24 maja 2017

Spis treści

JavaFX wstęp

JavaFX to platforma kliencka oraz pakiet do tworzenia graficznego interfejsu użytkownika następnej generacji.
JavaFX udostępnia bardzo potężną, łatwą w użyciu i elastyczną platformę ułatwiającą tworzenie nowoczesnych i bardzo atrakcyjnych wizualnie interfejsów użytkownika.

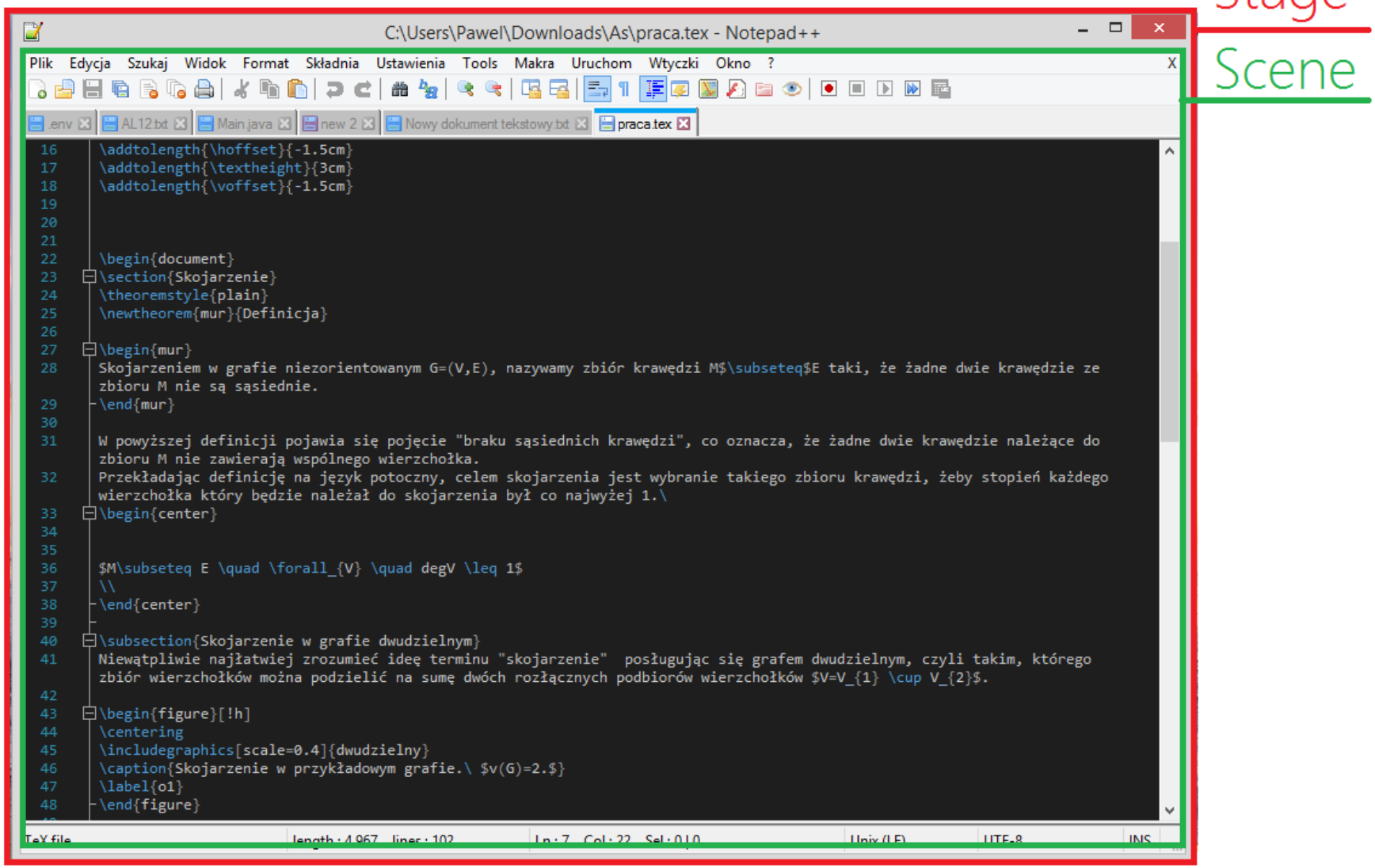
Podstawowe pojęcia

Ogólnie rzecz biorąc, platforma JavaFX ma wszystkie dobre cechy pakietu Swing. Na przykład została ona napisana bezpośrednio w Javie. Oprócz tego zapewnia wsparcie dla architektury MVC. Przeważająca większość pojęć związanych z tworzeniem graficznego interfejsu użytkownika w pakiecie Swing odnosi się także do JavaFX. Niemniej jednak pomiędzy oboma rozwiązaniami występują znaczące różnice.

Klasy Stage oraz Scene

Podstawową metaforą zaimplementowaną przez JavaFX jest obszar roboczy (ang. Stage) . Jak to się dzieje w przypadku planu scenicznego, także i obszar roboczy zawiera scenę (ang. Scene). A zatem ogólnie rzecz biorąc, obszar roboczy definiuje przestrzeń, natomiast scena określa, co się w tej przestrzeni znajduje. Można to także opisać w inny sposób: obszar roboczy jest pojemnikiem dla sceny, natomiast scena — dla elementów, które się na nią składają. W efekcie wszystkie aplikacje JavaFX zawierają przynajmniej jeden obszar roboczy oraz jedną scenę. W JavaFX elementy te są reprezentowane przez klasy Stage oraz Scene. W celu utworzenia aplikacji JavaFX należy w najprostszym przypadku dodać do obiektu Stage przynajmniej jeden obiekt Scene.

Klasy Stage oraz Scene



Klasa Application i metody cyklu życia

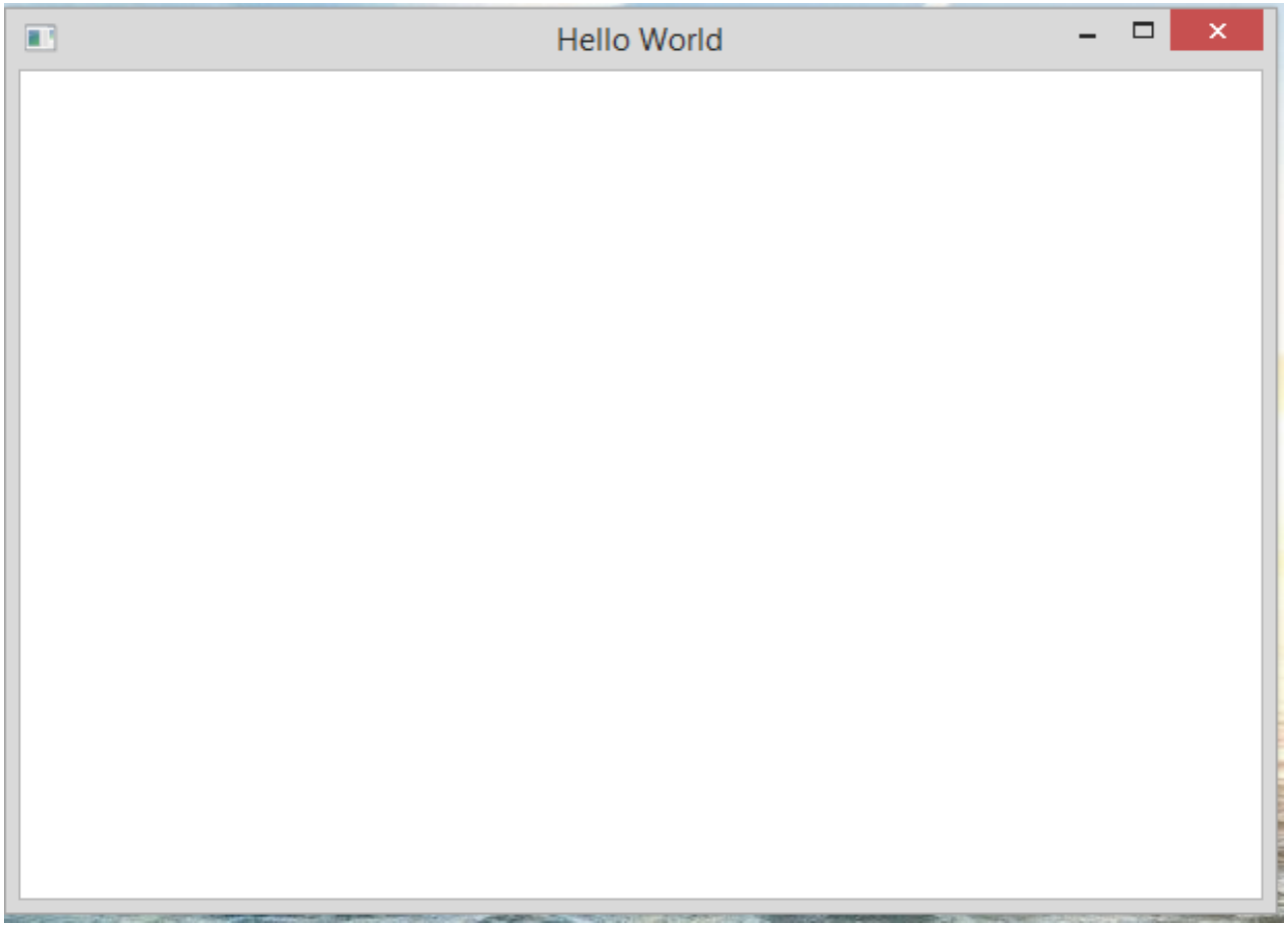
Każda aplikacja JavaFX musi być klasą dziedziczącą po klasie Application, umieszczoną w pakiecie javafx.application. Oznacza to, że klasy tworzonych przez nas aplikacji JavaFX będą rozszerzać klasę Application. Klasa ta definiuje trzy metody cyklu życia, które aplikacje mogą przesłaniać. Są nimi przedstawione poniżej metody:

```
1 init(), start() oraz stop()
```

Szkielet aplikacji JavaFX

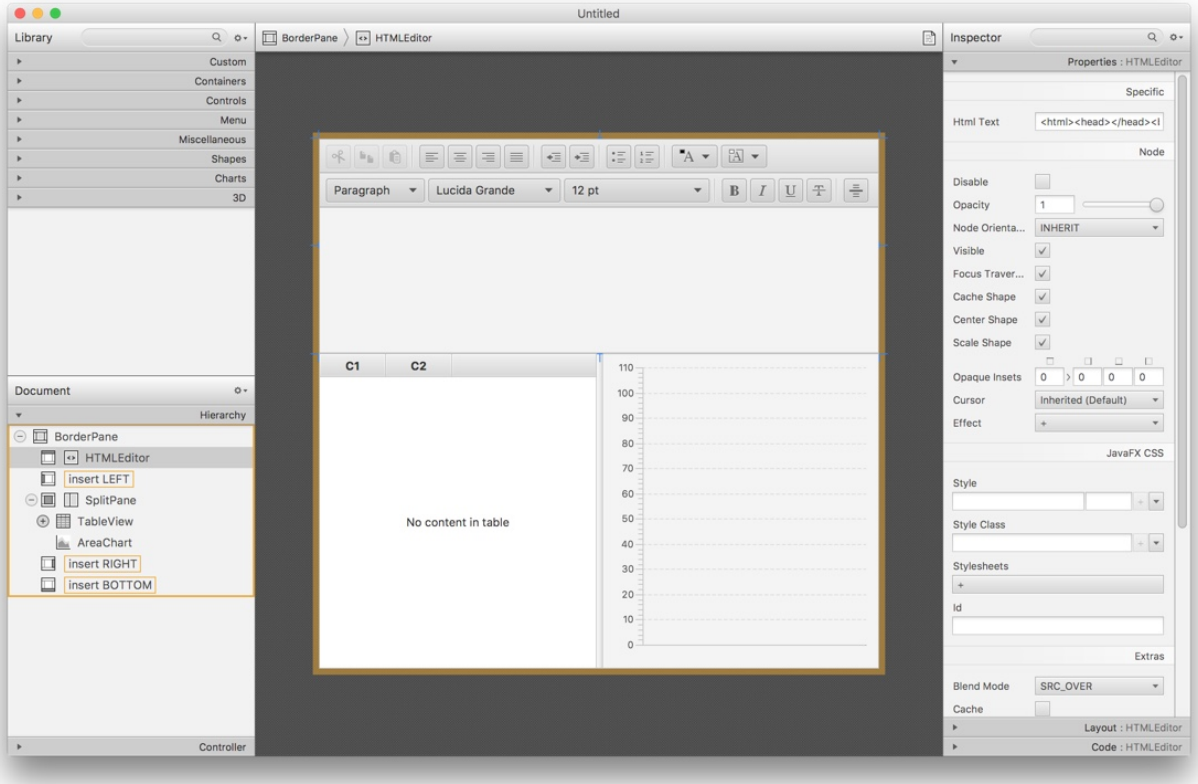
```
1  import javafx.application.Application;
2
3  public class Main extends Application {
4      @Override
5      public void start(Stage primaryStage) {
6          try {
7              Parent root = FXMLLoader.load(
                   getClass().getResource("Main.fxml
                   "));
8              primaryStage.setTitle("Hello World")
                   ;
9              primaryStage.setScene(new Scene(root
                   ));
10             primaryStage.show();
11         } catch(Exception e) {
12             e.printStackTrace();
13         }
14     }
15     public static void main(String[] args) {
16         launch(args);
```

JavaFx Hello World



Jak to wygląda w praktyce?

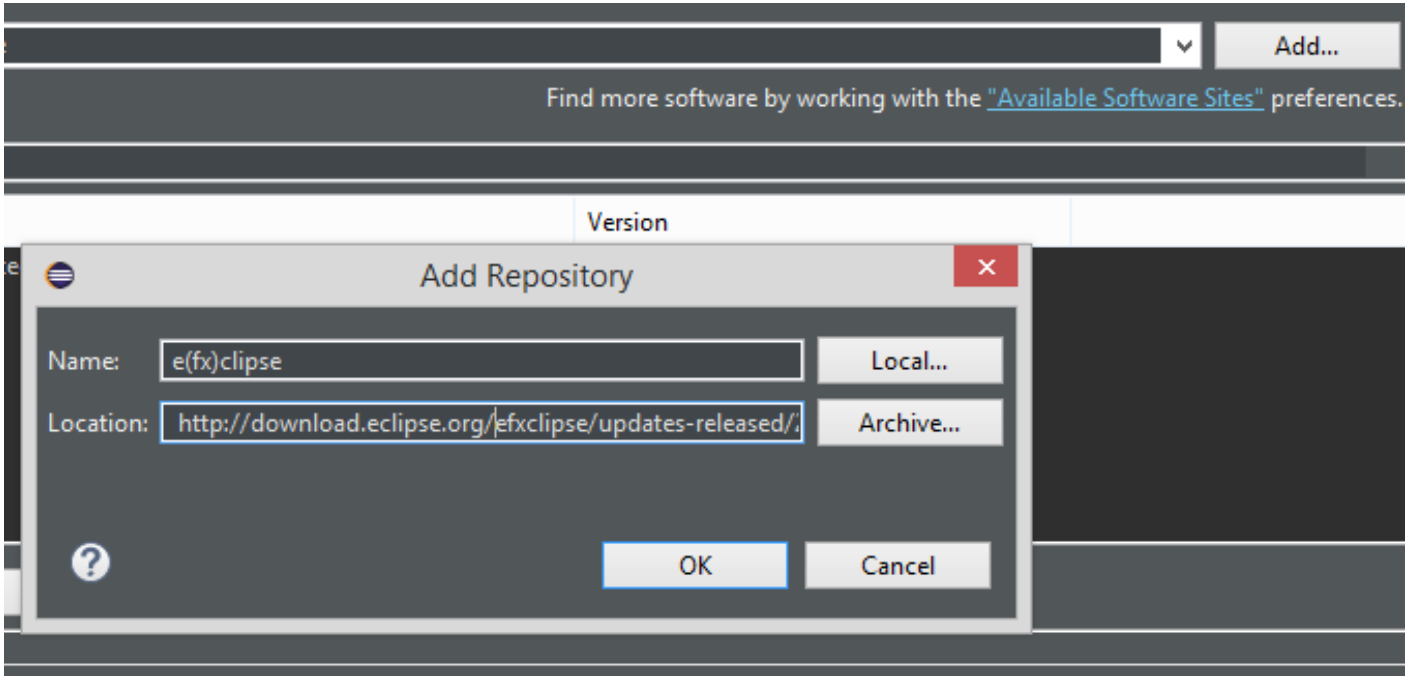
Będziemy potrzebowali Scene Builder'a



http://gluonhq.com/products/scene-builder/

Dodawanie JavaFX do Eclipse

Aby dodać JavaFX do Eclipse przechodzimy do
Help → Install New Software → Add po czym uzupełniamy



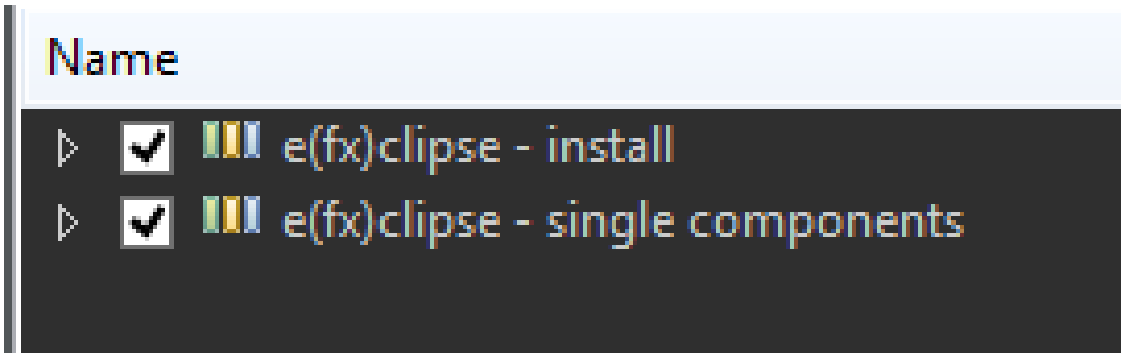
Name: e(fx)clipse

Location:

http://download.eclipse.org/efxclipse/updates-released/2.4.0/site

Dodawanie JavaFX do Eclipse

Zaznaczamy komponenty

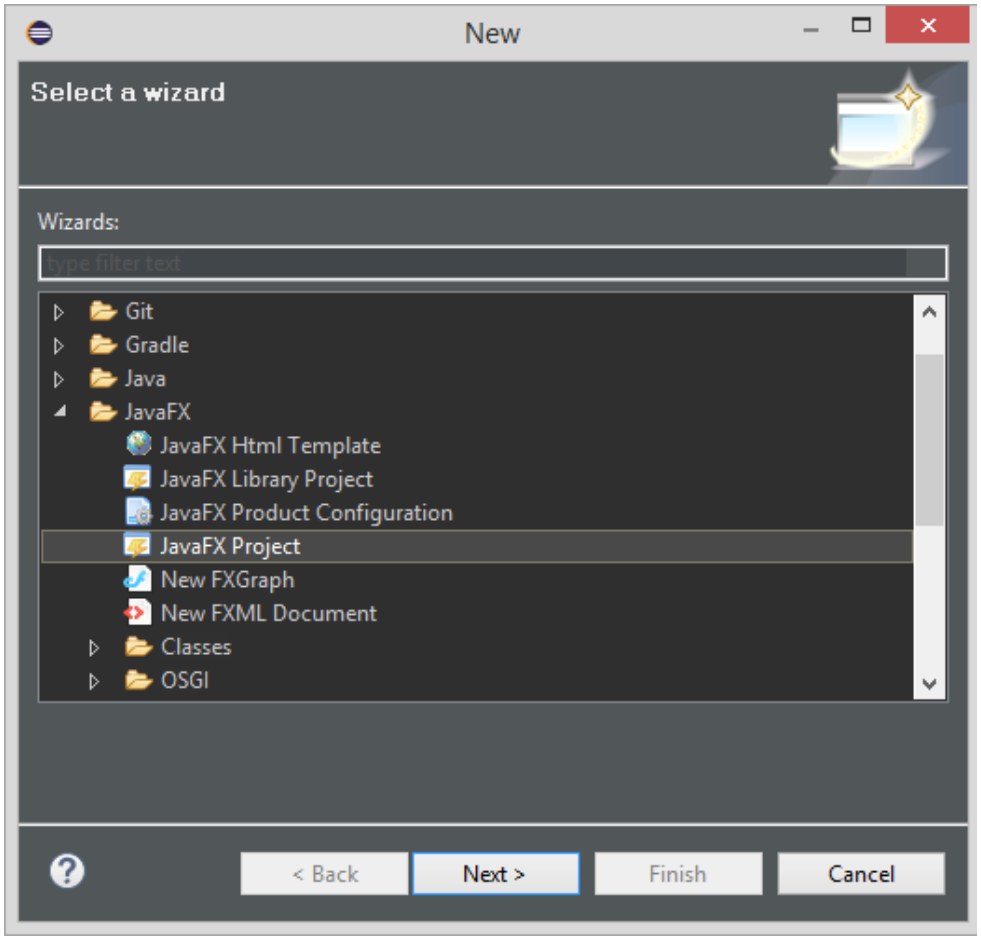


... i instalujemy

Tworzenie nowego projektu JavaFX Eclipse

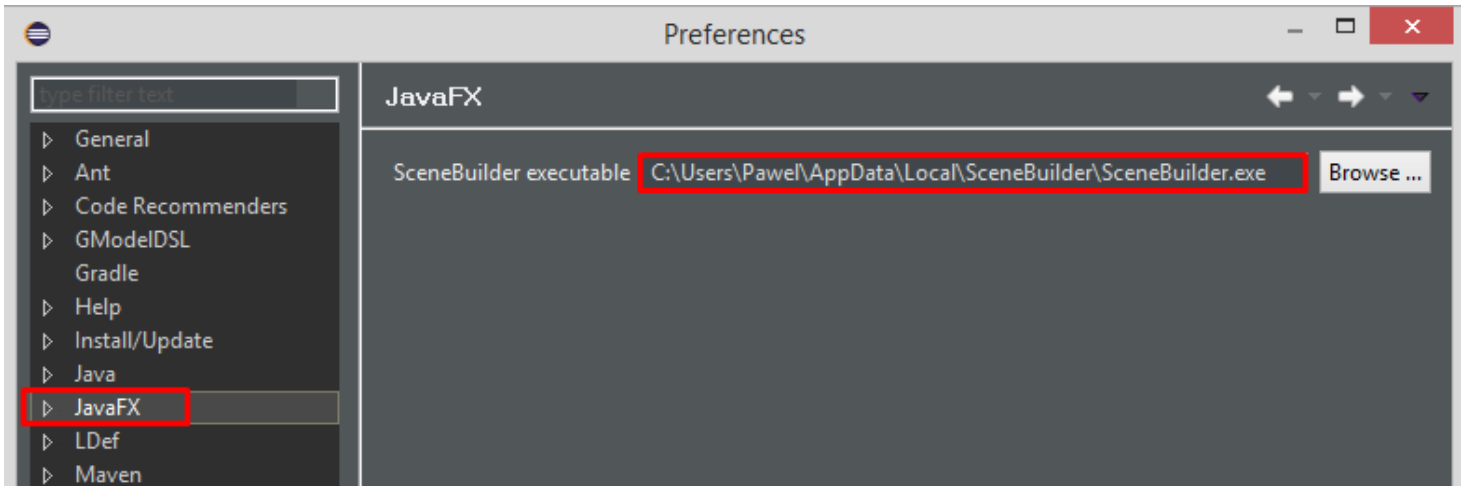
Po zainstalowaniu tworzymy nowy projekt JavaFX

W tym celu wybieramy File → New → Other...



Dodawanie Scene Builder'a

Pozostało jeszcze skonfigurować Scene Builder'a
W tym celu wybieramy Windows → Preferences →



Dodajemy ścieżkę do SceneBuilder.exe domyślnie w

```
1 C:\Users\Pawel\AppData\Local\SceneBuilder\
      SceneBuilder.exe
```

Treść zadania

1. baza danych ma liczyć co najmniej 5 encji,
2. każda encja powinna liczyć co najmniej 5 atrybutów (należy co najmniej raz użyć (w dowolnej encji): daty, liczby całkowitej, liczby rzeczywistej, łańcucha znaków, kwoty waluty),
3. powinien wystąpić co najmniej jeden związek N:M (uwaga: powstanie tabela pośrednicząca!) oraz
4. powinien wystąpić co najmniej jeden związek 1:N,
5. użytkownik nie może znać wartości kluczy głównych (można wyświelić LP - liczbę porządkową),
6. użytkownik nie może znać wartości identyfikatorów kluczy obcych (za zmuszanie użytkownika do pamiętania wartości kluczy obcych będę zabierał 1/3 punktów za to zadanie),
7. dla każdej z tabel powinny być zdefiniowane operacje: dodawania, modyfikacji, usuwania, wyświetlania pojedynczego i wszystkich rekordów,
8. zrealizowany CRUD powinien być "idiotoodporny" (walidacja poprawności wartości atrybutów, zabezpieczenie przed przypadkowym kliknięciem (czytaj potwierdzenie przy usuwaniu), itp.),
9. dla wygody można tworzyć bazy danych w SQLite,
10. jeżeli baza danych nie istnieje (lub została skasowana) powinna zostać utworzona automatycznie,
11. Zalecana architektura rozwiązania: MVC (model-widok-kontroler) inaczej -3 pkt,

MVC

Model

View

Controller

Model-Widok-Kontroler to w tej chwili najczęściej używany wzorzec projektowy. Niemal każda aplikacja, a zwłaszcza aplikacje webowe, wykorzystują go (często pod przykrywką jakiegoś frameworka).

Jak działa MVC?

Założenia wzorca Model-Widok-Kontroler są generalnie bardzo proste. Zacznijmy od analizy trzech tytułowych składowych:

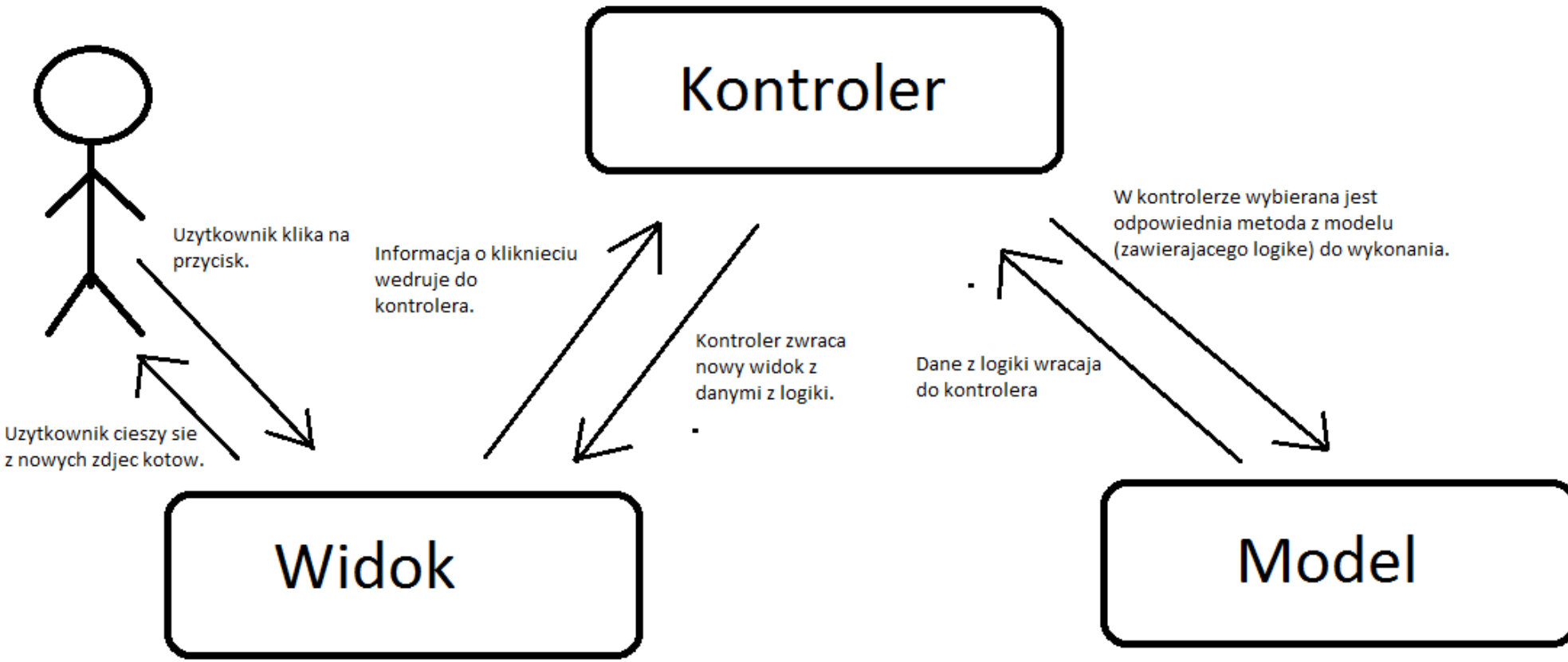
Model reprezentuje naszą logikę biznesową. Tutaj znajdują się wszelkie obiekty, które służą do wykonywania wszelkich operacji związanych z implementacją funkcjonalności naszej aplikacji.

Widok warstwa prezentacji. Widok odpowiedzialny jest za prezentację użytkownikowi wyników działania logiki biznesowej (Modelu).

Kontroler obsługuje żądania użytkownika. Wszelkie żądania deleguje do odpowiednich metod Modelu.

MVC

Ogólny schemat klasycznego wzorca MVC wygląda tak:



Tworzenie połączenia z bazą danych SQLite

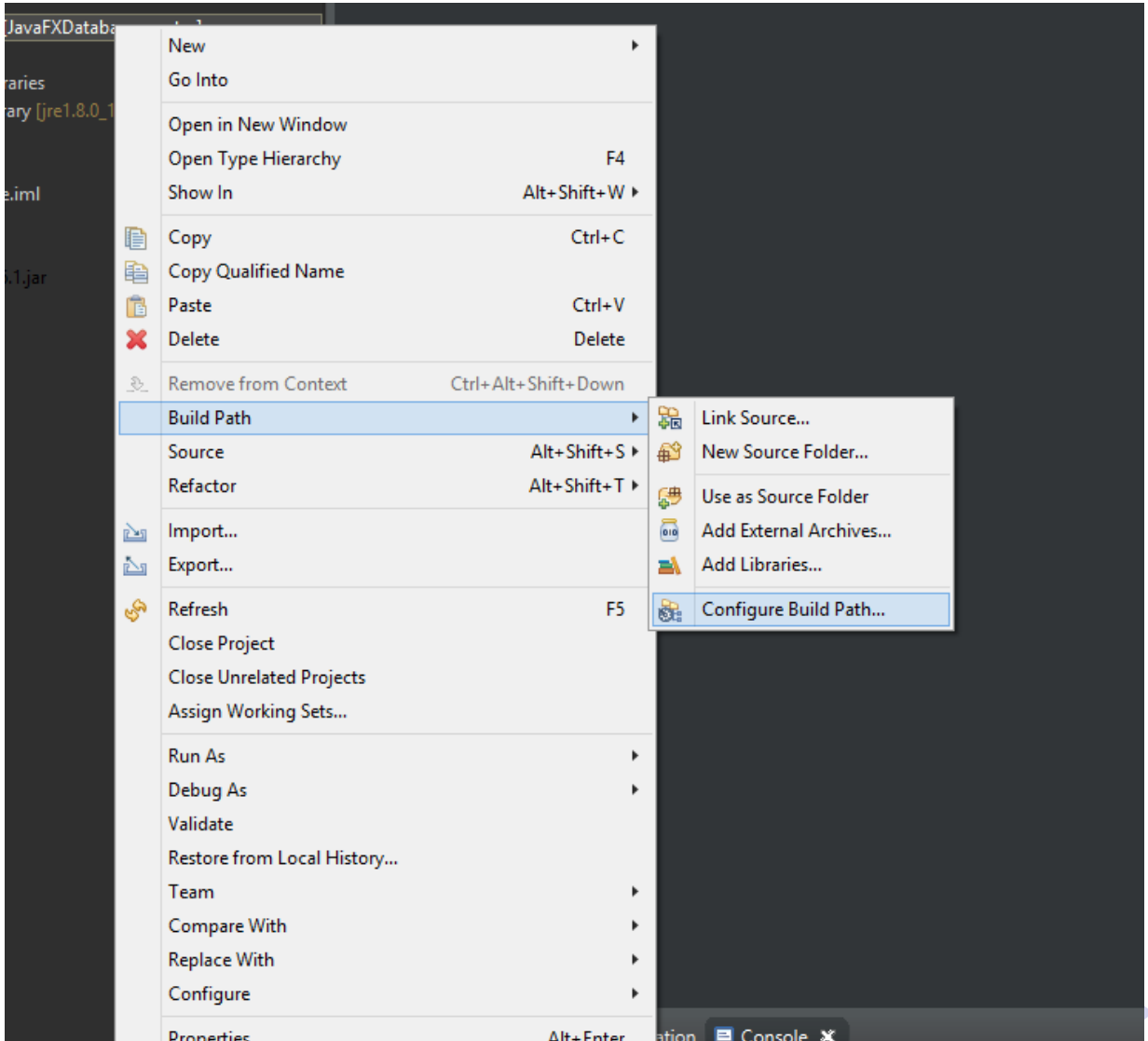
Aby stworzyć połączenie z bazą danych będziemy potrzebowali biblioteki ze sterownikiem konkretnej bazy danych.

W tym przypadku stworzymy połączenie z bazą danych SQLite.

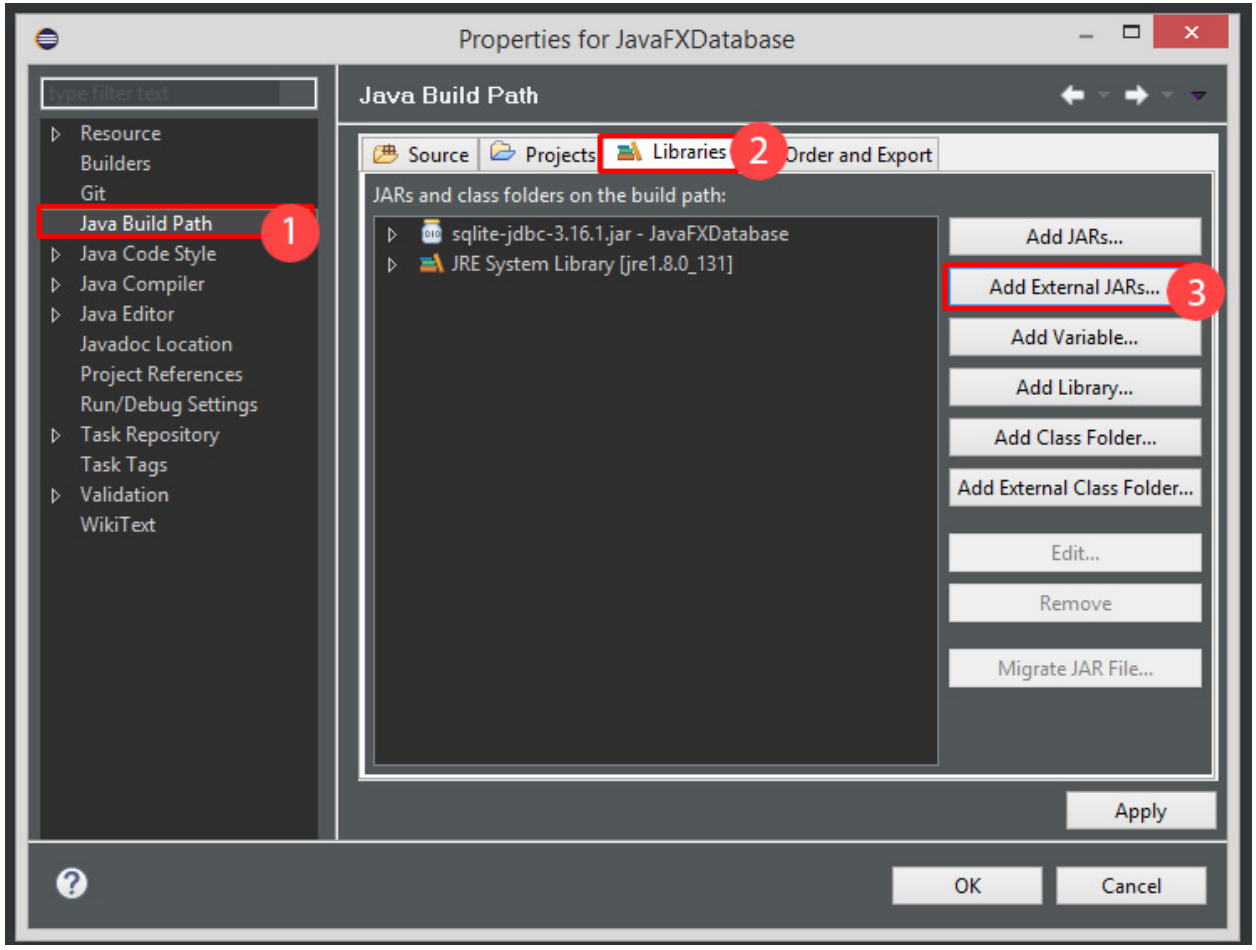
Aby pobrać najnowszy sterownik JDBC SQLite przechodzimy do `https://bitbucket.org/xerial/sqlite-jdbc/downloads/`

Podpięcie JDBC do projektu

Po stworzeniu projektu w musimy podpiąć pobraną bibliotekę *.jar



Podpięcie JDBC do projektu



Szkielet aplikacji JavaFX

W projekcie tworzymy klasę np. dbConnection.java

```
1  public class dbConnection {
2
3      private static final String SQLITECONN = "
           jdbc:sqlite:nazwaBazy.sqlite";
4
5      public static Connection getConnection()
           throws SQLException{
6
7          try {
8              Class.forName("org.sqlite.JDBC");
9              return DriverManager.getConnection(
                   SQLITECONN);
10         } catch (ClassNotFoundException e) {
11             e.printStackTrace();
12         }
13         return null;
14     }
15 }
```

Kod zadania

Kod projektu jest dostępny na Github'ie

`https://github.com/FiliusBonacci/SchoolManagementProject`