JavaFX 2.0 основы.

В данной рубрике мы будем говорить об очень интересной платформе для создания клиентских приложений Rich Internet Applications (RIAs), т.е. приложения, доступные через интернет (на подобии Flash), и обладающие богатой функциональностью, как настольные приложения.

Главное преимущество таких приложений в том, что основные вычисления происходят на стороне клиента, снижая тем самым нагрузку на сервер. Такие приложения более интерактивны и не требуют постоянной связи с сервером.

Hello, World! JavaFX

Лучший способ изучить и понять какой-либо язык — рассмотреть пример (хотя JavaFX — это уже не отдельный язык, а набор библиотек для Java). Поэтому создадим приложение JavaFX. Нам понадобятся, сама JavaFX 2.0 и Netbeans.

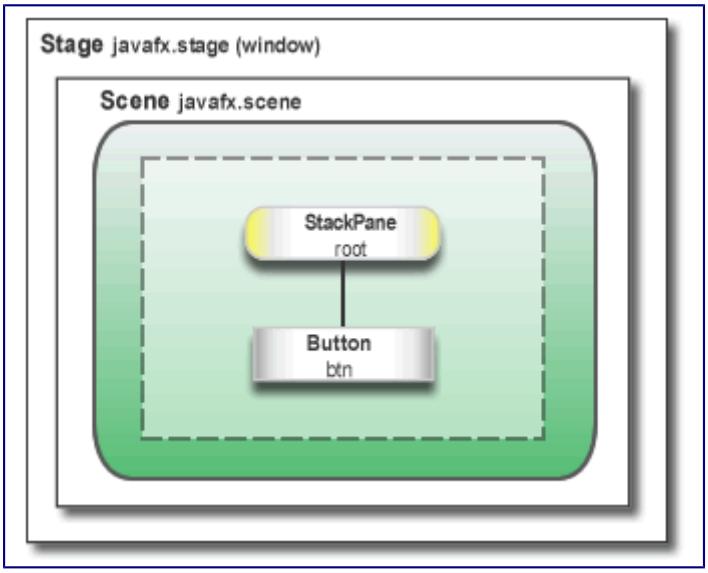
- В меню **File** выберите **New Project**.
- В категории **JavaFX** выберите **JavaFX Application**, далее
- Назовите проект HelloWorld, Готово.

Netbeans создаст проект и файл HelloWorld.java, который будет содержать простую JavaFX программу.

```
package helloworld;

import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

public class HelloWorld extends Application {
    public static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Hello World!");
        Button btn = new Button();
        btn.setText("Say 'Hello World'");
        btn.setOnAction(new EventHandler<ActionEvent>() {

            @Override
            public void handle(ActionEvent event) {
                System.out.println("Hello World!");
            }
        });

        StackPane root = new StackPane();
        root.getChildren().add(btn);
        primaryStage.setScene(new Scene(root, 300, 250));
        primaryStage.show();
    }
}
```

Рассмотрим основные компоненты структуры JavaFX приложения.

- Главный класс JavaFX приложения унаследован от javafx.application.Application. Метод start() — главный метод приложения, в отличие от обычного Java приложения, где главный — main().
- В JavaFX приложении присутствуют два главных компонента stage и scene. В примере мы создаем сцену с определенными размерами и делаем ее видимой.
- Программа JavaFX имеет иерархическую структуру в виде дерева, где узлы — элементы программы (кнопка, текст и т.д.). В нашем случае корневой узел — объект StackPane — слой с изменяемым размером, это означает, что размер окна программы можно будет изменять.
- Наш корневой узел содержит одного потомка — кнопку с обработчиком нажатия (для вывода сообщения в консоль).



Запуск приложения

Запустите программу и нажмите на кнопку, в окне вывода Netbeans программа выведет «Hello World!».

В папке dist нашего проекта должны появиться файлы HelloWorld.html — страница со встроенным апплетом, HelloWorld.jnpl — нужен для запуска приложения в браузере через WebStart, HelloWorld.jar — Java архив и папка web-files содержит картинки, использованные в программе и JavaScript файл для загрузки апплета на страницу.

Создание FX-Формы

Научимся создавать простые формы в JavaFx. При создании веб-приложения не обойтись без форм, например, для логина или для обратной связи, регистрации и т.д. В этом

уроке мы создадим простую форму для логина и научимся базовой разметке формы, добавим элементы формы и обработаем события, происходящие при работе с ней. В этом уроке мы будем использовать среду NetBeans, перед тем как начать, убедитесь, что ваша версия поддерживает работу с JavaFX 2.

Создание проекта

Для начала нам надо создать JavaFX проект в NetBeans, назовем его Login:

1. В меню файл выберите «Создать проект».
2. В категории JavaFX выберите «Приложение JavaFX», нажмите «Далее»
3. Назовите проект Login. Когда NetBeans создаст JavaFX проект, вы увидите Hello World приложение, как в предыдущем уроке.
4. Удалите метод start(), который создал NetBeans, и замените его следующим:

```
@Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("JavaFX Welcome");

        primaryStage.show();
    }
```

Подсказка: после добавления кода в проект NetBeans, нажмите Ctrl (или Cmd) + Shift + I для импорта необходимых пакетов. После появления окна со списком, выберите тот, который начинается с javafx.

Создание сетки GridPane

Для формы логина удобно использовать GridPane, потому что этот объект позволяет создать гибкую сетку из столбцов и строк, в которые удобно поместить поля для ввода и кнопки. Вы можете поместить элементы управления в любую ячейку в сетке.

Код для создания GridPane, добавьте его перед строкой primaryStage.show():

```
GridPane grid = new GridPane();
grid.setAlignment(Pos.CENTER);
grid.setHgap(10);
grid.setVgap(10);
grid.setPadding(new Insets(25, 25, 25, 25));

Scene scene = new Scene(grid, 300, 275);
primaryStage.setScene(scene);
```

Данный код создает объект GridPane, хранящийся в переменной grid. Свойство alignment меняет стандартное положение сетки с «вверху-слева» на центральное положение в окне. Свойства gap отвечают за пробелы между строками и столбцами, а свойство padding отвечает за пространство вокруг сетки. Порядок перечисления в insets: вверху, справа, снизу и слева. В нашем примере отступ вокруг сетки — 25 пикселей с каждой стороны.

Сцена создана с GridPane в качестве корневого узла, это обычная практика при верстке контейнеров. Таким образом при изменении размеров окна, элементы формы будут соответственно реагировать. В нашем примере форма будет оставаться по-середине при увеличении или уменьшении окна. Свойство padding гарантирует, что вокруг формы останется отступ при уменьшении размеров окна.

Данный код создает сцену шириной 300 и высотой 275. Если вы не зададите эти параметры, то сцена будет минимального размера, который необходим для отображения её содержимого.

Добавляем текст, метки и поля формы

Как видно на картинке выше, наша форма имеет заголовок «Welcome», текстовое поле и поле для ввода пароля пользователя. Рассмотрим код, создающий эти поля. Добавьте данный код после задания свойства padding GridPane:

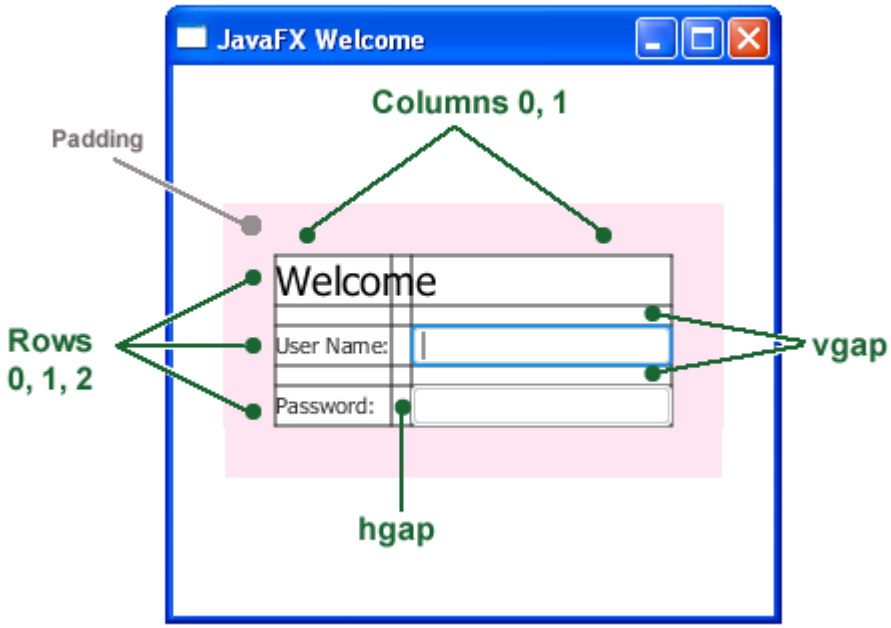
```
 Text scenetitle = new Text("Welcome");
 scenetitle.setFont(Font.font("Tahoma", FontWeight.NORMAL, 20));
 grid.add(scenetitle, 0, 0, 2, 1);

 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);
```

Первая строка создает объект типа Text со значением Welcome, который нельзя изменить. Следующая строка использует метод setFont() для задания шрифта, размера и стиля переменной scenetitle. В данном примере мы используем встроенный стиль, но лучше использовать таблицы CSS, работу с ними мы рассмотрим в следующем уроке.

Метод grid.add() добавляет объект scenetitle к слою нашей формы. Нумерация столбцов и строк начинается с 0, т.о. заголовок формы будет добавлен в столбец 0 и строку 0. Последние два аргумента метода grid.add() устанавливают промежутки между столбцами — 2, между строками — 1.

Следующая строка создает объект Label с текстом User Name в столбце 0, строке 1 и Текстовое поле, которое можно редактировать. Текстовое поле добавлено в столбец 1, строку 1. Поле для ввода пароля и метка для него создаются аналогично.

При работе с GridPane можно отобразить границы, это бывает полезно при отладке. Чтобы отобразить границы разметки необходимо установить свойство gridLinesVisible равным true. Запустив после этого программу, вы увидите линии — границы столбцов и строк сетки:



Добавление кнопки и текста

Для полноценной работы форме не хватает еще два элемента: Button — кнопка, для отправки формы и Text — объект, который будет отображать информацию при нажатии на кнопку.

Сначала создадим кнопку и поместим её снизу справа — стандартное расположение кнопки в формах. Добавьте этот код перед кодом создания сцены:

```
1Button btn = new Button("Sign in");
2HBox hbBtn = new HBox(10);
3hbBtn.setAlignment(Pos.BOTTOM_RIGHT);
4hbBtn.getChildren().add(btn);
5grid.add(hbBtn, 1, 4);
```

Данный код создает кнопку с текстом Sign in и помещает её в вспомогательный объект HBox с отступами 10 пикселей, который служит выравнивания кнопки. Свойство alignment

объекта hbBtn установлено равным Pos.BOTTOM_RIGHT, это означает, что кнопка будет выровнена по правому нижнему краю. Сам объект HBox поместим в нашу сетку в ячейку 1, 4.

Теперь добавим элемент Text, для отображения информации. Добавьте этот код до кода создания сцены:

```
final Text actiontarget = new Text();
        grid.add(actiontarget, 1, 6);
```

Сейчас мы не увидим сообщения, которое должно появляться при нажатии кнопки, чтобы отобразить его, нам надо обработать событие нажатия на кнопку.

Обработка событий

Заставим кнопку показывать сообщение, когда на нее нажмут. Добавим следующий код в программу:

```
btn.setOnAction(new EventHandler<ActionEvent>() {

    @Override
    public void handle(ActionEvent e) {
        actiontarget.setFill(Color.FIREBRICK);
        actiontarget.setText("Sign in button pressed");
    }
});
```

Метод setOnAction() используется для регистрации обработчика события нажатия на кнопку. Когда происходит нажатие на кнопку мы меняем цвет текста сообщения на красный:

Теперь запустите приложение, нажмите на кнопку и увидите результат.

Делаем JavaFX приложение красивым с помощью CSS

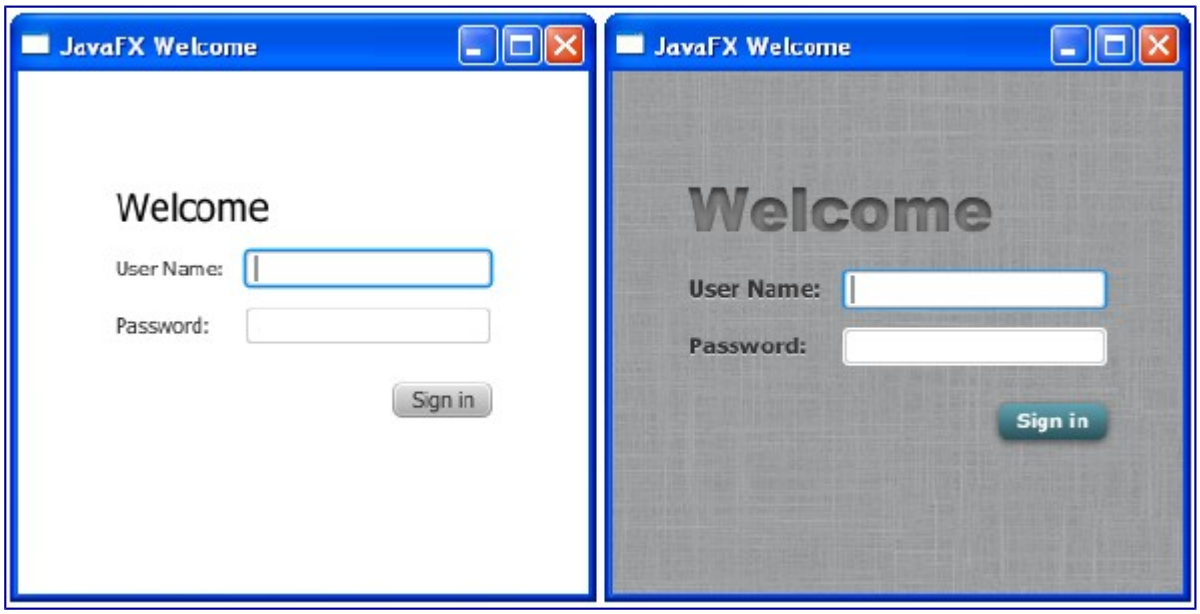
Этот урок расскажет о том, как сделать ваше JavaFX приложение привлекательным с помощью CSS.

В пердыдущем уроке мы создали форму для авторизации с помощью JavaFx, теперь придадим ей интересный вид с помощью CSS.

Мы создадим отдельный .css файл, разместим в нем необходимые стили и применим их в нашем приложении, созданном в пердыдущем уроке (Создание форм в JavaFx).

Напишем стили для кнопок, фона и шрифта.

В итоге форма будет выглядеть примерно так:



Для создания приложения вам понадобится IDE NetBeans. Также убедитесь, что версия NetBeans поддерживает JavaFX2. Создайте проект, как в предыдущем уроке или скачайте исходники.

Создание CSS файла

Сперва создадим CSS файл и сохраним его в той же папке, что и главный класс программы. После нам надо добавить файл в программу.

1. Откройте проект в NetBeans, найдите пакет исходных фалов login и добавьте CSS файл в проект. Назовите файл login.css и убедитесь, что он в папке `src\login`.
2. В файл `Login.java` вставьте следующий код для добавления CSS файла в сцену.

```
1Scene scene = new Scene(grid, 300, 275);
2primaryStage.setScene(scene);
3scene.getStylesheets().add
4 (Login.class.getResource("Login.css").toExternalForm());
5primaryStage.show();
```

Этот код будет искать файл в папке `src\login` проекта.

Добавляем фоновое изображение

Загрузите картинку для фона и положите её в папку `src\login` вашего проекта.

Теперь добавим код для создания фона в CSS файл. Учтите, что путь к файлам задается относительно CSS файла. В коде ниже файл `background.jpg` находится в той же папке, что и CSS файл.

```
.root {
    -fx-background-image: url("background.jpg");
}
```

Фоновое изображение назначено классу `.root` это означает, что стиль будет применен к корневому узлу объекта `Scene`. Определение стиля состоит из свойства (`-fx-background-image`) и значения этого свойства (`url("background.jpg")`). Что должно получиться:

Стилизуем метки

Следующий шаг — написать стили для меток. Мы будем использовать CSS класс `.label`, а значит этот стиль будет применен на все метки в форме.

```
.label {
    -fx-font-size: 12px;
    -fx-font-weight: bold;
    -fx-text-fill: #333333;
    -fx-effect: dropshadow( gaussian , rgba(255,255,255,0.5) , 0,0,0,1 );
}
```

Этот код увеличит размер шрифта и с делает его жирным, а также создаст тень серого цвета (#333333). Тень нужна для создания контраста между темно-серым текстом и светло-серым фоном:

Стилизуем текст

Напишем стиль для двух текстовых объектов формы: `scenetitle`, который содержит текст `Welcome`, и `actiontarget`, который содержит текст возвращаемый после нажатия на кнопку.

1. В файле `Login.java` удалите строки, задающие стиль текста этих объектов: `scenetitle.setFont(Font.font(“Tahoma”, FontWeight.NORMAL, 20)); actiontarget.setFill(Color.FIREBRICK);` Таким образом мы отделим дизайн приложения от исходного кода, что позволит изменять стиль приложения без изменения кода.
2. Создайте ID для каждого объекта, используя метод `setID()`: `scenetitle.setId("welcome-text"); actiontarget.setId("actiontarget");`
3. В файле `Login.css` определите стили для созданных ID: `welcome-text` и `actiontarget:`

```
#welcome-text {
    -fx-font-size: 32px;
    -fx-font-family: "Arial Black";
    -fx-fill: #818181;
    -fx-effect: innershadow( three-pass-box , rgba(0,0,0,0.7) , 6, 0.0 , 0 , 2 );
}
#actiontarget {
   -fx-fill: FIREBRICK;
   -fx-font-weight: bold;
   -fx-effect: dropshadow( gaussian , rgba(255,255,255,0.5) , 0,0,0,1 );
}
```

Размер текста Welcome увеличен до 32 пикселей и шрифт изменен на Arial Black. Цвет

изменили на темно-серый (#818181), а также добавили внутреннюю тень.

Стиль объекта `actiontarget` такой же как у меток.

Стилизуем кнопку

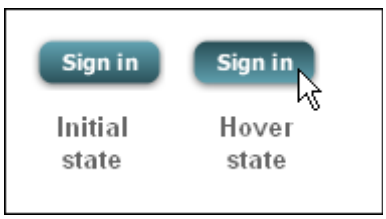
И, наконец, напишем стиль для кнопки. Сделаем так, чтобы она менялась при наведении.

Сначала напишем начальный стиль кнопки. Этот код использует CSS селектор класса `.button` и этот стиль будет применен ко всем кнопкам формы.

```
.button {
    -fx-text-fill: white;
    -fx-font-family: "Arial Narrow";
    -fx-font-weight: bold;
    -fx-background-color: linear-gradient(#61a2b1, #2A5058);
    -fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0.6) , 5, 0.0 , 0 , 1 );
}
```

Теперь напишем стиль кнопки при наведении на неё курсора. Делается это с помощью псевдо-класса:

```
.button:hover {
 -fx-background-color: linear-gradient(#2A5058, #61a2b1);
 }
```



Готово!