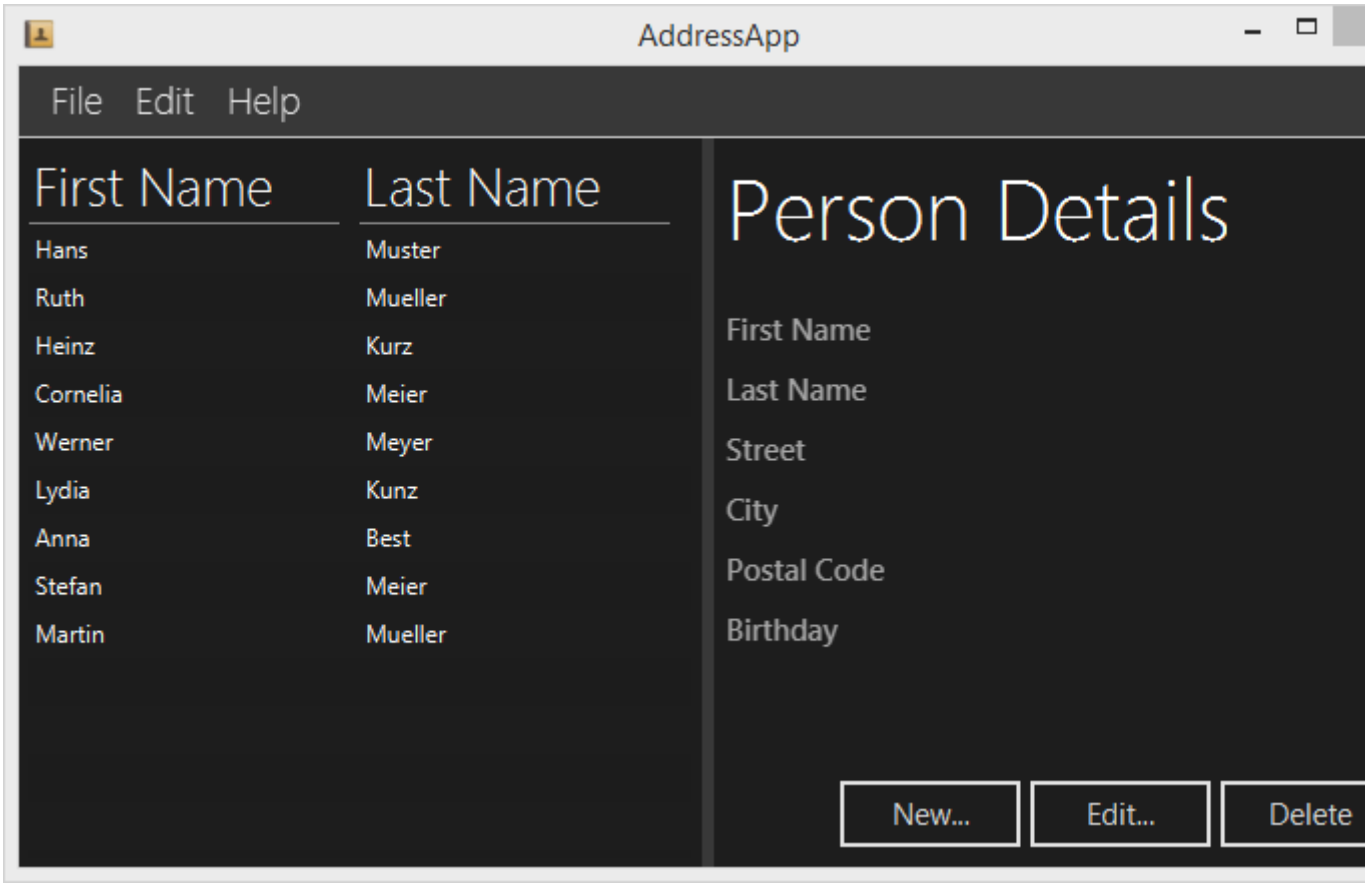
# JavaFX 8 - Часть 4: Стилизация с помощью CSS




## Часть 4: Содержание

- Стилизация с помощью каскадных таблиц стилей (CSS)
- Добавление иконки приложения

---

## Стилизация с помощью CSS

В JavaFX с помощью каскадных таблиц стилей (CSS) можно стилизировать интерфейс пользователя. Это просто здорово! Ещё никогда не было так легко настроить внешний вид приложения Java.

В этом учебнике мы создадим тему *DarkTheme*, вдохновленную Метро-дизайном из Windows 8. Принятые стили для кнопок заимствованы из статьи JMetro - Windows 8 Metro controls on Java, написанной Pedro Duque Vieira.

### Знакомство с CSS

Если вы хотите приукрасить внешний вид своего приложение JavaFX, то надо иметь хотя бы начальное представление о том, что такое CSS. Хорошее место для старта - этот учебник по CSS.

# Стиль, используемый в JavaFX по умолчанию

Стиль, который используется в JavaFX по умолчанию хранится в файле `modena.css`. Его можно найти в файле `jfxrt.jar`, который, в свою очередь, располагается в директории Java `/jdk1.8.x/jre/lib/ext/jfxrt.jar`.

Разархивируйте его и вы найдете `modena.css` в папке `com/sun/javafx/scene/control/skin/modena`.

Этот стиль всегда применяется по умолчанию для всех приложений JavaFX. Добавляя пользовательские стили мы переопределяем стили из файла `modena.css`.

**Подсказка:** Для того, чтобы знать какие стили следует переопределить, просмотрите этот файл.

# Подключение пользовательских CSS-стилей

Добавьте файл `DarkTheme.css` в пакет `view`. (New- Other-Cascading Style Sheet)

DarkTheme.css

```
.background {
    -fx-background-color: #1d1d1d;
}

.label {
    -fx-font-size: 11pt;
    -fx-font-family: "Segoe UI Semibold";
    -fx-text-fill: white;
    -fx-opacity: 0.6;
}

.label-bright {
    -fx-font-size: 11pt;
    -fx-font-family: "Segoe UI Semibold";
    -fx-text-fill: white;
```

```
    -fx-opacity: 1;
}

.label-header {
    -fx-font-size: 32pt;
    -fx-font-family: "Segoe UI Light";
    -fx-text-fill: white;
    -fx-opacity: 1;
}

.table-view {
    -fx-base: #1d1d1d;
    -fx-control-inner-background: #1d1d1d;
    -fx-background-color: #1d1d1d;
    -fx-table-cell-border-color: transparent;
    -fx-table-header-border-color: transparent;
    -fx-padding: 5;
}

.table-view .column-header-background {
    -fx-background-color: transparent;
}

.table-view .column-header, .table-view .filler {
    -fx-size: 35;
    -fx-border-width: 0 0 1 0;
    -fx-background-color: transparent;
    -fx-border-color:
        transparent
        transparent
        derive(-fx-base, 80%)
        transparent;
    -fx-border-insets: 0 10 1 0;
```

```
}

.table-view .column-header .label {
    -fx-font-size: 20pt;
    -fx-font-family: "Segoe UI Light";
    -fx-text-fill: white;
    -fx-alignment: center-left;
    -fx-opacity: 1;
}

.table-view:focused .table-row-cell:filled:focused:selected {
    -fx-background-color: -fx-focus-color;
}

.split-pane:horizontal > .split-pane-divider {
    -fx-border-color: transparent #1d1d1d transparent #1d1d1d;
    -fx-background-color: transparent, derive(#1d1d1d,20%);
}

.split-pane {
    -fx-padding: 1 0 0 0;
}

.menu-bar {
    -fx-background-color: derive(#1d1d1d,20%);
}

.context-menu {
    -fx-background-color: derive(#1d1d1d,50%);
}

.menu-bar .label {
    -fx-font-size: 14pt;
```

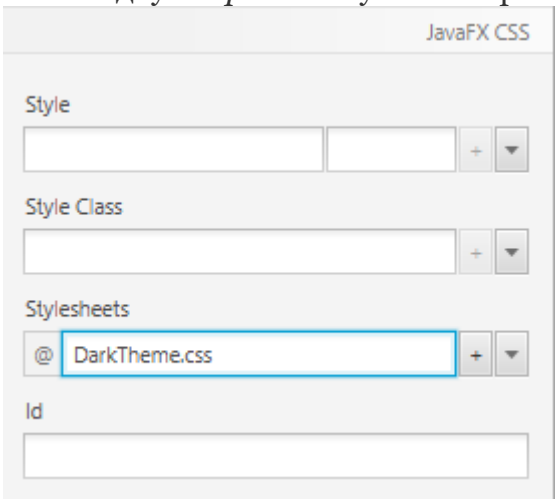
```
    -fx-font-family: "Segoe UI Light";
    -fx-text-fill: white;
    -fx-opacity: 0.9;
}

.menu .left-container {
    -fx-background-color: black;
}

.text-field {
    -fx-font-size: 12pt;
    -fx-font-family: "Segoe UI Semibold";
}

/*
 * Push Button в стиле Metro
 * Автор: Pedro Duque Vieira
 * http://pixelduke.wordpress.com/2012/10/23/jmetro-windows-8-controls-on-java/
 */
.button {
    -fx-padding: 5 22 5 22;
    -fx-border-color: #e2e2e2;
    -fx-border-width: 2;
    -fx-background-radius: 0;
    -fx-background-color: #1d1d1d;
    -fx-font-family: "Segoe UI", Helvetica, Arial, sans-serif;
    -fx-font-size: 11pt;
    -fx-text-fill: #d8d8d8;
    -fx-background-insets: 0 0 0 0, 0, 1, 2;
}

.button:hover {
    -fx-background-color: #3a3a3a;
```

```
}

.button:pressed, .button:default:hover:pressed {
  -fx-background-color: white;
  -fx-text-fill: #1d1d1d;
}

.button:focused {
    -fx-border-color: white, white;
    -fx-border-width: 1, 1;
    -fx-border-style: solid, segments(1, 1);
    -fx-border-radius: 0, 0;
    -fx-border-insets: 1 1 1 1, 0;
}

.button:disabled, .button:default:disabled {
    -fx-opacity: 0.4;
    -fx-background-color: #1d1d1d;
    -fx-text-fill: white;
}

.button:default {
    -fx-background-color: -fx-focus-color;
    -fx-text-fill: #ffffff;
}

.button:default:hover {
    -fx-background-color: derive(-fx-focus-color,30%);
}
```

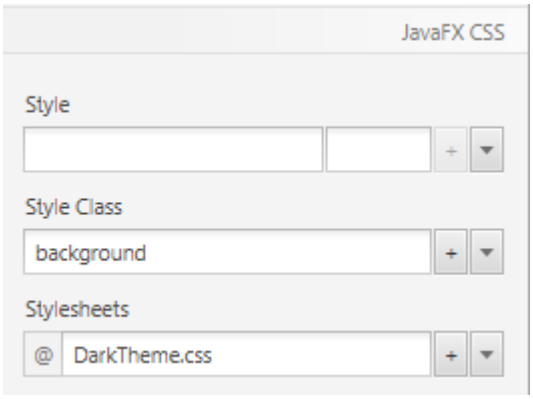
Теперь надо подключить эти стили к нашей сцене. Мы можем сделать это программно, в коде Java, но в этом уроке, чтобы добавить стили в наши fxml-файлы, мы будем использовать Scene Builder:

## *Подключаем таблицы стилей к файлу FXMLDocument.fxml*

1. В приложении Scene Builder откройте файл `FXMLDocument.fxml`.
2. Во вкладке *Hierarchy* выберите корневой контейнер `BorderPane`, перейдите на вкладку *Properties* и укажите файл `DarkTheme.css` в роли таблиц стилей.

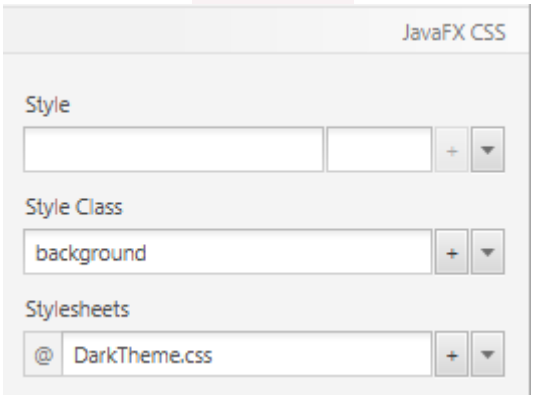



3. Фон справа всё ещё белый, поэтому укажите для правого компонента `AnchorPane` в классе стиля значение `background`.




## *Подключаем таблицы стилей к файлу PersonEditDialog.fxml*

1. В приложении Scene Builder откройте файл `PersonEditDialog.fxml`. Во вкладке *Hierarchy* выберите корневой контейнер `AnchorPane`, перейдите на вкладку *Properties* и укажите файл `DarkTheme.css` в роли таблиц стилей.
2. Фон всё ещё белый, поэтому укажите для корневого компонента `AnchorPane` в классе стиля значение `background`.

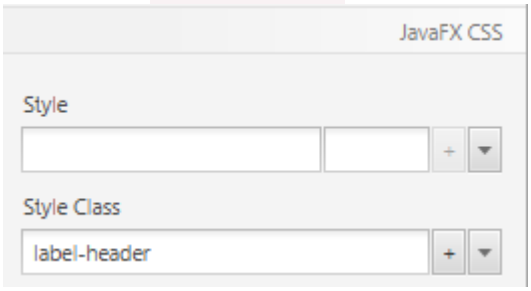


3. Выберите кнопку *OK* и отметьте свойство *Default Button* в вкладке *Properties*. В результате изменится цвет кнопки и она будет использоваться по умолчанию когда пользователь, находясь в окне, будет нажимать клавишу Enter.

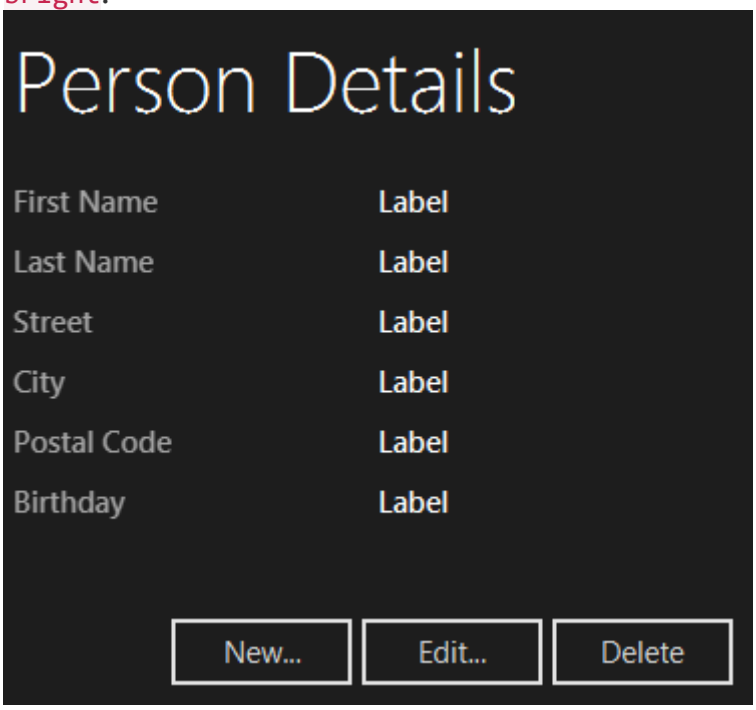
### *Текстовые метки с другими стилями*

Откройте файл `FXMLDocument.fxml`. Сейчас все текстовые метки с правой стороны имеют одинаковый размер. Для дальнейшей стилизации текстовых меток мы будем использовать уже определённые стили `.label-header` и `label-bright`.

1. Выберите метку *Person Details* и добавьте в качестве класса стиля значение `label-header`.




2. Для каждой метки в правой колонке (где отображаются фактические данные об адресатах) добавьте в качестве класса стиля значение `label-bright`.




---

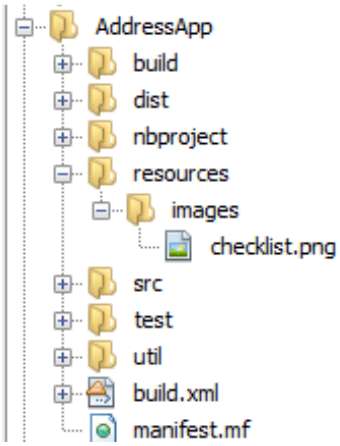
# Добавляем иконку приложения

На данный момент в нашем приложении в панели названия и панели задач используется иконка по умолчанию.

# Файл иконки

Одно из возможных мест, где можно свободно скачать иконки — это сайт Icon Finder. Советую скачать маленькую иконку адресной книги.

Создайте внутри вашего проекта AddressApp обычную папку **resources**, а в ней папку **images**. Поместите выбранную вами иконку в папку изображений. Структура папок должна иметь такой вид:




# Установка иконки для сцены

Для того, чтобы для нашей сцены установить новую иконку, в классе `AdressApp.java` добавьте следующий код в метод `start(...)`

AdressApp.java

```
stage.getIcons().add(new Image("file:resources/images/checklist.png"));
```

Весь метод `start(...)` теперь будет выглядеть так:

```
public void start(Stage stage) {

Parent root = FXMLLoader.load(getClass().getResource("FXMLDocument.fxml"));

        Scene scene = new Scene(root);

        stage.setScene(scene);

        stage.setTitle("AddressApp");


    // Устанавливаем иконку приложения.

    stage.getIcons().add(new Image("file:resources/images/checklist.png"));

    stage.show();

}
```

Вы также можете добавить иконку в окно редактирования адресатов.