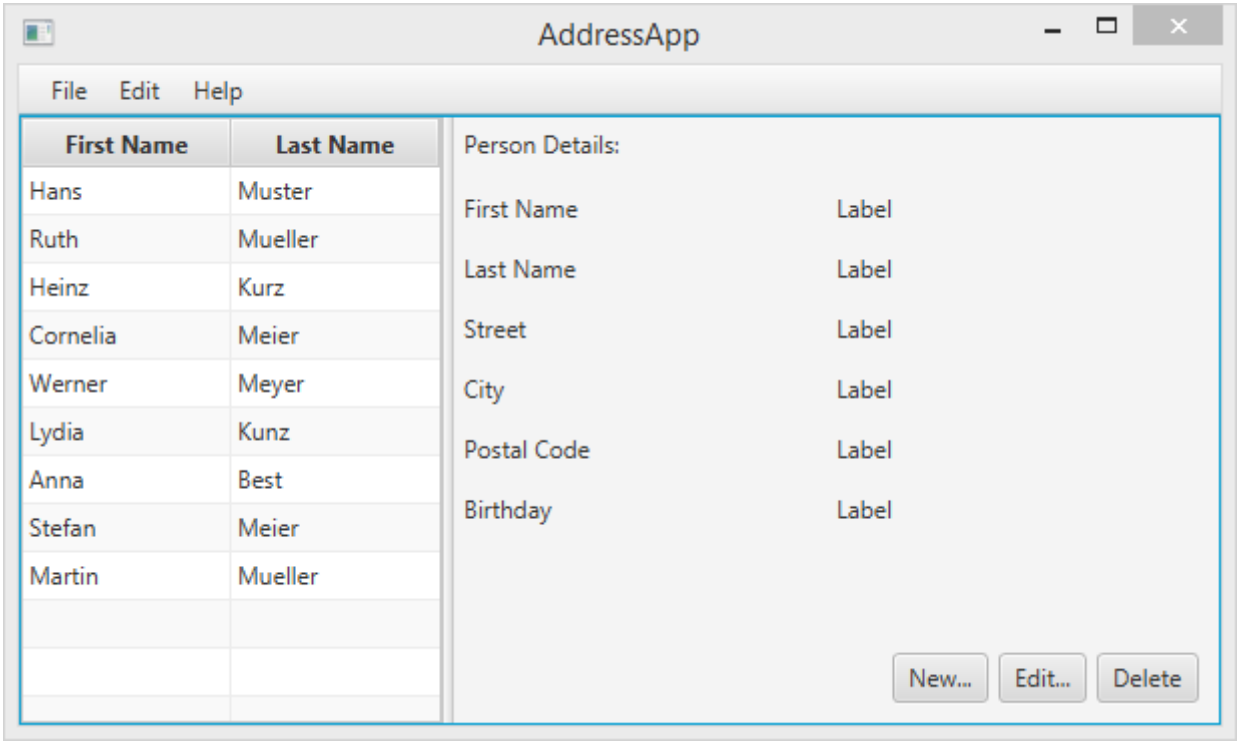
JavaFX 8 - Часть 2: Модель и компонент



Часть 2: Содержание

- Создание класса-**модели**;
- Использование класса-модели в коллекции **ObservableList**;
- Отображение данных в компоненте **TableView** с помощью **Контроллеров**.

Создание класса-модели

Класс-модель необходим для хранения в нашей будущей адресной книге информации об адресатах. Добавьте класс `Person.java` в пакет `addressapp`. В нём будет несколько переменных для хранения информации об имени, адресе и дне рождения.

Мы так же добавим в код конструктор, который будет создавать некоторые демонстрационный данные и методы-геттер и сеттер с публичным модификатором доступа:

Добавьте в этот класс следующий код.

AddressApp\src\addressapp\Person.java

```
package addressapp;

import java.time.LocalDate;
```

```java
public class Person {

    private  String firstName;
    private  String lastName;
    private  String street;
    private  int postalCode;
    private  String city;
    private  LocalDate birthday;


    public Person(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
        // Какие-то фиктивные начальные данные для удобства тестирования.
        this.street = "какая-то улица";
        this.postalCode = 1234;
        this.city = "какой-то город";
        this.birthday = LocalDate.of(1999, 2, 21);
    }

    public Person() {

    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getStreet() {
        return street;
    }

    public void setStreet(String street) {
        this.street = street;
    }

    public Integer getPostalCode() {
        return postalCode;
    }

    public void setPostalCode(Integer postalCode) {
        this.postalCode = postalCode;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
```

```
    }

    public LocalDate getBirthday() {
        return birthday;
    }

    public void setBirthday(LocalDate birthday) {
        this.birthday = birthday;
    }
}
```

Объяснение

- Класс `LocalDate`, тип которого мы выбрали для нашей переменной `birthday`, это часть нового Date and Time API для JDK 8.

Список людей

Основные данные, которыми оперирует наше приложение - это группа экземпляров класса `Person`. Давайте создадим в классе `FXMLDocumentController.java` список объектов класса `Person`. Все остальные классы-контроллеры позже получат доступ к этому центральному списку внутри этого класса.

Список ObservableList

Мы работаем с классами-представлениями JavaFX, которые необходимо информировать при любых изменениях в списке адресатов. Это важно, потому что, не будь этого, мы бы не смогли синхронизировать представление данных с самими данными. Для этой цели в JavaFX были введены некоторые новые классы коллекций.

- Из этих классов нам понадобится класс `ObservableList`.
- Для того, чтобы получить доступ к таблице и меткам представления, мы определим некоторые переменные. Эти переменные и некоторые методы имеют специальную аннотацию `@FXML`. Она необходима для того, чтобы fxml-файл имел доступ к приватным полям и методам. После этого мы настроим наш fxml-файл так, что при его загрузке приложение автоматически заполняло эти переменные данными.
- . Итак, давайте добавим следующий код в наш класс:

Примечание: При импорте пакетов всегда используйте пакет ***javafx***, а НЕ *awt* или *swing*!

FXMLDocumentController.java

```
package addressapp;


import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.control.Label;
```

```
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;


public class FXMLDocumentController {
    /**
     * Данные, в виде наблюдаемого списка адресатов.
     */
    private final ObservableList<Person> personData = FXCollections.observableArrayList();

    @FXML
    private TableView<Person> personTable;
    @FXML
    private TableColumn<Person, String> firstNameColumn;
    @FXML
    private TableColumn<Person, String> lastNameColumn;

    @FXML
    private Label firstNameLabel;
    @FXML
    private Label lastNameLabel;
    @FXML
    private Label streetLabel;
    @FXML
    private Label postalCodeLabel;
    @FXML
    private Label cityLabel;
    @FXML
    private Label birthdayLabel;

    @FXML
    public void initialize() {
        // В качестве образца добавляем некоторые данные
        personData.add(new Person("Hans", "Muster"));
        personData.add(new Person("Ruth", "Mueller"));
        personData.add(new Person("Heinz", "Kurz"));
        personData.add(new Person("Cornelia", "Meier"));
        personData.add(new Person("Werner", "Meyer"));
        personData.add(new Person("Lydia", "Kunz"));
        personData.add(new Person("Anna", "Best"));
        personData.add(new Person("Stefan", "Meier"));
        personData.add(new Person("Martin", "Mueller"));
        // Инициализация таблицы адресатов с двумя столбцами.
        firstNameColumn.setCellValueFactory(new PropertyValueFactory<>("firstName"));
        lastNameColumn.setCellValueFactory(new PropertyValueFactory<>("lastName"));
        // Добавление в таблицу данных из наблюдаемого списка
        personTable.setItems(personData);
    }

}
```

Этот код требует некоторых разъяснений:

- Все поля и методы, к которым fxml-файлу потребуется доступ, должны быть отмечены аннотацией `@FXML`. Несмотря на то, что это требование предъявляется только для полей и методов с модификатором private, лучше оставить их закрытыми и помечать аннотацией, чем делать публичными!
- После загрузки fxml-файла автоматически вызывается метод `initialize()`. На этот момент все FXML-поля должны быть инициализированы;

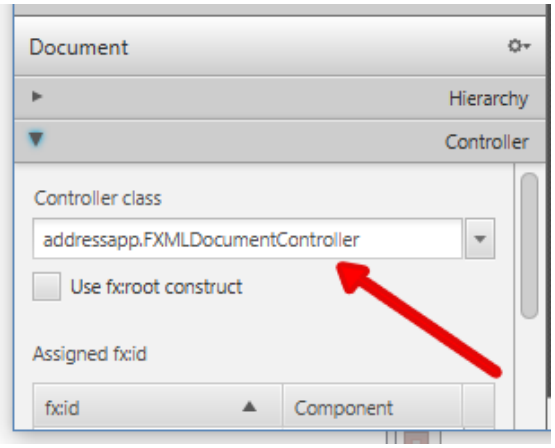
- Метод `setCellValueFactory(...)` определяет, какое поле внутри класса `Person` будет использоваться для конкретного столбца в таблице (PropertyValueFactory,).

Класс FXMLDocumentController

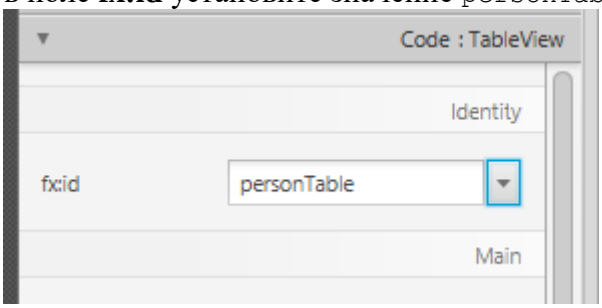
Привязка класса-контроллера к fxml-файлу

Данная часть учебника близится к своему завершению, однако мы пропустили одну маленькую деталь! Мы не сказали файлу `FXMLDocument.fxml`, какой контроллер он должен использовать, а так же не указали соответствие между элементами представления и полями внутри класса-контроллера. Для этого:

1. Откройте файл `FXMLDocument.fxml` в приложении *Scene Builder*.
2. Откройте вкладку *Controller* слева на панели *Document* и выберите класс `FXMLDocumentController` в качестве класса-контроллера.

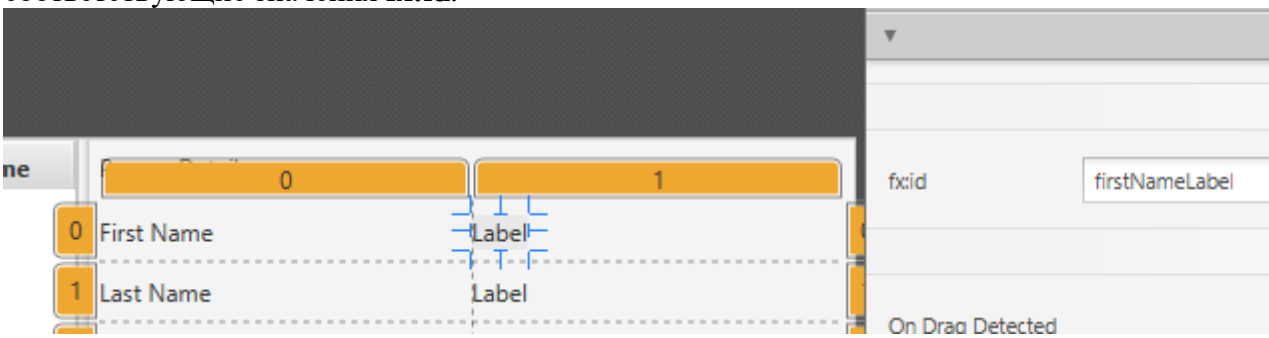


3. Выберите компонент `TableView` на вкладке *Hierarchy*, перейдите на вкладку *Code* и в поле **fx:id** установите значение `personTable`.



4. Сделайте то же самое для колонок таблицы и установите значения свойства **fx:id** `firstNameColumn` и `lastNameColumn` соответственно.

5. Для каждой метки во второй колонке компонента GridPane также установите соответствующие значения **fx:id**.



6. Важно: сохраните файл `FXMLDocument.fxml`, вернитесь в среду разработки NetBeans

Запуск приложения

После запуска приложения мы должны увидеть что-то похожее на то, что изображено на картинке в начале данной статьи.

Поздравляю!

Примечание: пока ещё при выборе конкретного адресата у нас не обновляются метки. Взаимодействие с пользователем мы будем программировать в следующей части учебника.