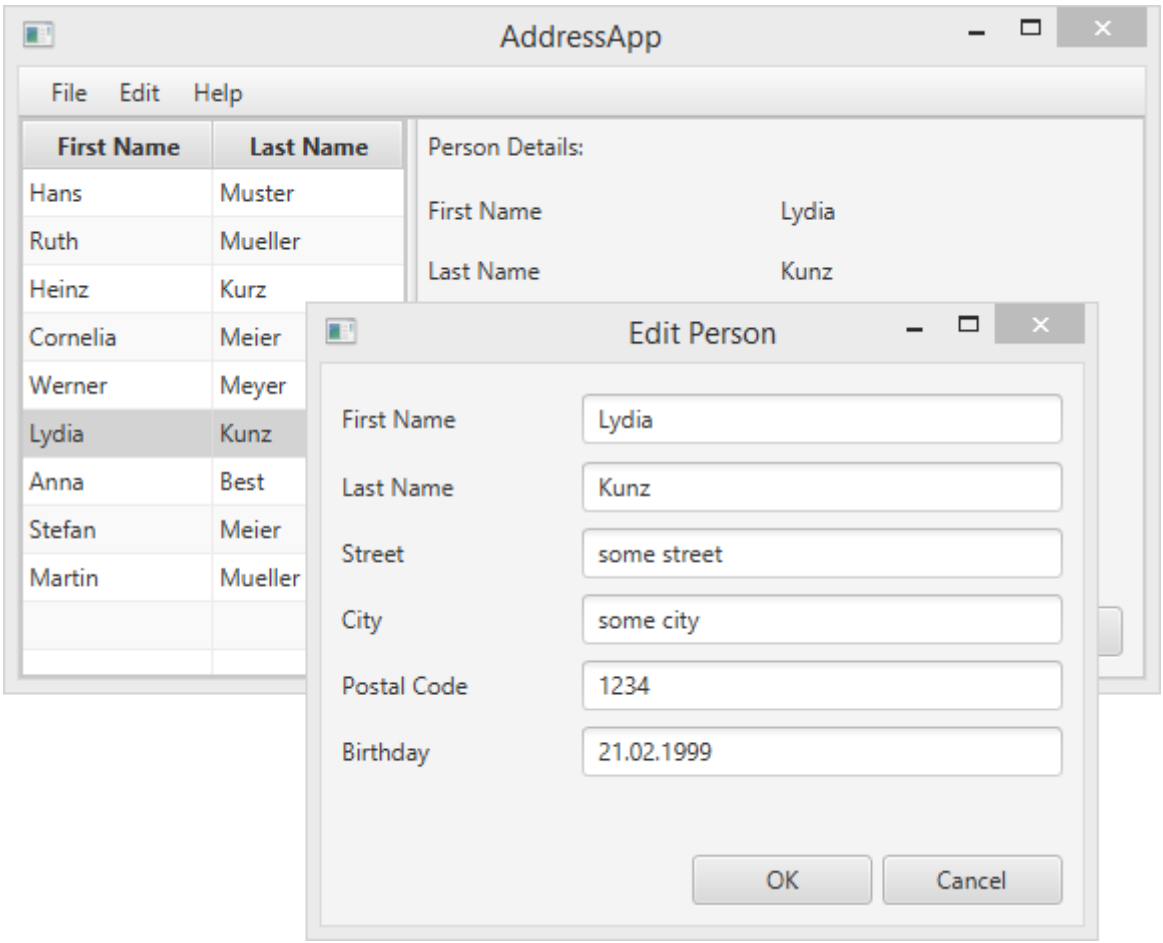
JavaFX 8 - Часть 3: Взаимодействие с пользователем



Часть 3: Содержание

- Реакция на выбор адресатов в таблице.
- Добавление функциональности кнопкам **add**, **edit** и **remove**.
- Создание диалогового окна для изменения информации об адресатах.
- Проверка пользовательского ввода.

Реакция на выбор адресатов в таблице

Пока мы ещё не использовали правую часть нашего приложения. Идея заключается в том, чтобы при выборе адресата в таблице, в правой части приложения отображать дополнительную информацию об этом адресате.

Сначала давайте добавим новый метод в класс `FXMLDocumentController`. Он поможет нам заполнять текстовые метки данными указанного адресата (`Person`).

Создайте метод `showPersonDetails(Person person)`. Его код проходится по меткам, и, используя метод `setText(...)`, присваивает им соответствующие значения, взятые из

переданного в параметре объекта Person. Если в качестве параметра передаётся `null`, то весь текст в метках будет очищен.

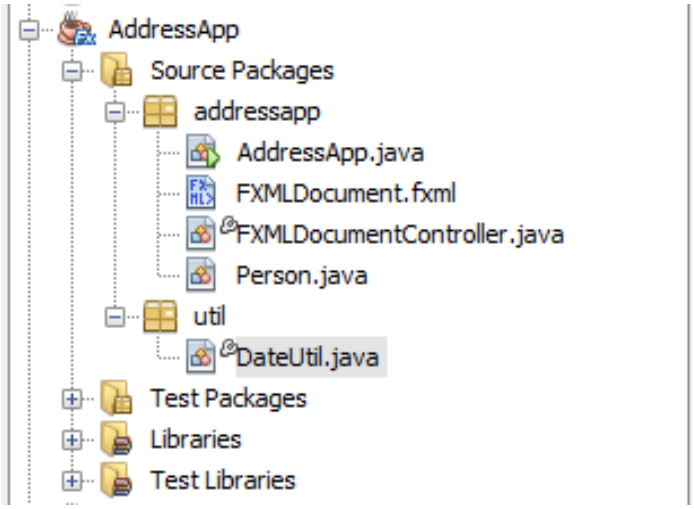
FXMLDocumentController.java

```
/**
     * Заполняет все текстовые поля, отображая подробности об адресате. Если
     * указанный адресат = null, то все текстовые поля очищаются.
     *
     * @param person — адресат типа Person или null
     */
    private void showPersonDetails(Person person) {
        if (person != null) {
            // Заполняем метки информацией из объекта person.
            firstNameLabel.setText(person.getFirstName());
            lastNameLabel.setText(person.getLastName());
            streetLabel.setText(person.getStreet());
            postalCodeLabel.setText(Integer.toString(person.getPostalCode()));
            cityLabel.setText(person.getCity());

            birthdayLabel.setText(DateUtil.format(person.getBirthday()));
            // birthdayLabel.setText(...);
        } else {
            // Если Person = null, то убираем весь текст.
            firstNameLabel.setText("");
            lastNameLabel.setText("");
            streetLabel.setText("");
            postalCodeLabel.setText("");
            cityLabel.setText("");
            birthdayLabel.setText("");
        }

    }
```

Преобразование дня рождения в строку

Обратите внимание, что мы просто так не можем присвоить текстовой метке значение поля `birthday`, так как тип его значения `LocalDate` а не `String`. Для того чтобы это сделать, нам надо сначала отформатировать дату.

Мы будем преобразовывать тип `LocalDate` в тип `String` и обратно в нескольких местах программы, поэтому, хорошей практикой считается создание для этой цели вспомогательного класса, содержащего статические методы. Этот вспомогательный класс мы назовем `DateUtil` и разместим его в новом пакете `ch.makery.address.util`:



AddressApp\src\util\DateUtil.java

```
package util;

import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.format.DateTimeParseException;

/**
 * Вспомогательные функции для работы с датами.
 *
 * @author Computer School
 */
public class DateUtil {

    /**
     * Шаблон даты, используемый для преобразования. Можно поменять на свой.
     */
    private static final String DATE_PATTERN = "dd.MM.yyyy";

    /**
     * Форматировщик даты.
     */
    private static final DateTimeFormatter DATE_FORMATTER
            = DateTimeFormatter.ofPattern(DATE_PATTERN);

    /**
     * Возвращает полученную дату в виде хорошо отформатированной строки.
     * Используется определённый выше {@link DateUtil#DATE_PATTERN}.
     *
     * @param date - дата, которая будет возвращена в виде строки
     * @return отформатированную строку
     */
    public static String format(LocalDate date) {
        if (date == null) {
            return null;
        }
        return DATE_FORMATTER.format(date);
    }

    /**
     * Преобразует строку, которая отформатирована по правилам шаблона
     * {@link DateUtil#DATE_PATTERN} в объект {@link LocalDate}.
     *
     * Возвращает null, если строка не может быть преобразована.
     *
```

```
    * @param dateString - дата в виде String
    * @return объект даты или null, если строка не может быть преобразована
    */
   public static LocalDate parse(String dateString) {
      try {
         return DATE_FORMATTER.parse(dateString, LocalDate::from);
      } catch (DateTimeParseException e) {
         return null;
      }
   }

   /**
    * Проверяет, является ли строка корректной датой.
    *
    * @param dateString
    * @return true, если строка является корректной датой
    */
   public static boolean validDate(String dateString) {
      // Пытаемся разобрать строку.
      return DateUtil.parse(dateString) != null;
   }
}
```

Подсказка: формат даты можно поменять, просто изменив константу `DATE_PATTERN`. Все возможные форматы описаны в документации к классу DateTimeFormatter

Использование класса DateUtil

Теперь мы можем использовать наш новый класс `DateUtil` в методе `showPersonDetails` класса `FXMLDocumentController`. Замените комментарий *TODO* следующей строкой:

```
birthdayLabel.setText(DateUtil.format(person.getBirthday()));
```

Наблюдение за выбором адресатов в таблице

Чтобы знать о том, что пользователь выбрал определённую запись в таблице, нам необходимо *прослушивать изменения*.

Для этого в JavaFX существует интерфейс `ChangeListener` с единственным методом `changed(...)`. Этот метод имеет три параметра: `observable`, `oldValue` и `newValue`.

Мы будем реализовывать интерфейс `ChangeListener` с помощью *лямбда-выражений* из Java 8. Давайте добавим несколько строчек кода к методу `initialize()` класса `FXMLDocumentController`. Теперь этот метод выглядит так:

FXMLDocumentController.java

```
@FXML
private void initialize() {
……
      // Очистка дополнительной информации об адресате.
      showPersonDetails(null);

      // Слушаем изменения выбора, и при изменении отображаем
      // дополнительную информацию об адресате.
      personTable.getSelectionModel().selectedItemProperty().addListener(
            (observable, oldValue, newValue) -> showPersonDetails(newValue));
}
```

Если мы передаём в параметр метода `showPersonDetails(...)` значение `null`, то все значения меток будут стёрты.

В строке `personTable.getSelectionModel...` мы получаем *selectedItemProperty* таблицы и добавляем к нему слушателя. Когда пользователь выбирает запись в таблице, выполняется наше лямбда-выражение. Мы берём только что выбранную запись и передаём её в метод `showPersonDetails(...)`.

Запустите свое приложение и проверьте, отображаются ли детали адресата в правой части, когда в таблице выбирается определённый адресат.

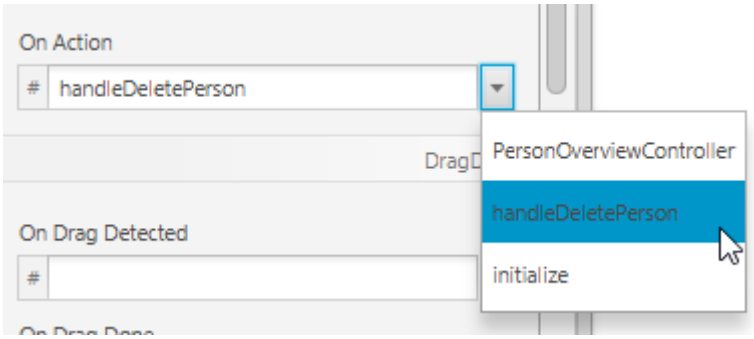
Кнопка Delete

В нашем пользовательском интерфейсе есть кнопка **Delete**, но пока она не работает. В приложении Scene Builder мы можем задать действие, которое будет выполняться при нажатии на эту кнопку. Любой метод внутри класса-контроллера, помеченный аннотацией @FXML (или публичный), доступен Scene Builder. Поэтому, давайте сперва добавим в конец класса `FXMLDocumentController` метод удаления адресата, а уже потом назначим его обработчиком кнопки **Delete**.

FXMLDocumentController.java

```
    /**
     * Вызывается, когда пользователь кликает по кнопке удаления.
     */
    @FXML
    private void handleDeletePerson() {
        int selectedIndex = personTable.getSelectionModel().getSelectedIndex();
        personTable.getItems().remove(selectedIndex);
    }
```

Теперь в приложении Scene Builder откройте файл `FXMLDocument.fxml`. Выберите кнопку **Delete**, откройте вкладку *Code* и укажите метод `handleDeletePerson` в значение пункта *On Action*.



Помните: После изменения FXML-файла в Scene Builder, для того, чтобы изменения вступили в силу выполните сохранение.

Обработка ошибок

Если вы сейчас запустите приложение, то уже сможете удалять выбранных адресатов из таблицы. Но что произойдёт, если не будет выбран ни один адресат, а вы нажмёте кнопку **Delete**,?

Вылетит исключение `ArrayIndexOutOfBoundsException`, потому что не получится удалить адресата с индексом `-1`. Значение `-1` возвращается методом `getSelectedIndex()`, когда в таблице ничего не выделено.

Естественно, игнорировать эту ошибку будет некрасиво. Мы должны сообщать пользователю о том, что он, перед тем как нажимать кнопку **Delete**, должен выбрать запись в таблице. (Ещё лучше совсем деактивировать кнопку, чтобы у пользователя не было соблазна сделать что-то не так).

Добавим в метод `handleDeletePerson` некоторые изменения. Теперь, когда пользователь нажимает на кнопку **Delete**, а в таблице ничего не выбрано, он увидит простое диалоговое окно:

FXMLDocumentController.java

```
    /**
     * Вызывается, когда пользователь кликает по кнопке удаления.
     */
    @FXML
    private void handleDeletePerson() {
        int selectedIndex = personTable.getSelectionModel().getSelectedIndex();
        if (selectedIndex >= 0) {
            personTable.getItems().remove(selectedIndex);
        } else {
            // Ничего не выбрано.
            Alert alert = new Alert(AlertType.WARNING);
            alert.initOwner(null);
            //Не выделено
            alert.setTitle("No Selection");
            //Не выбран Person
            alert.setHeaderText("No Person Selected");
            //Пожалуйста выберите человека в таблице
            alert.setContentText("Please select a person in the table.");
            alert.showAndWait();
        }
    }
```

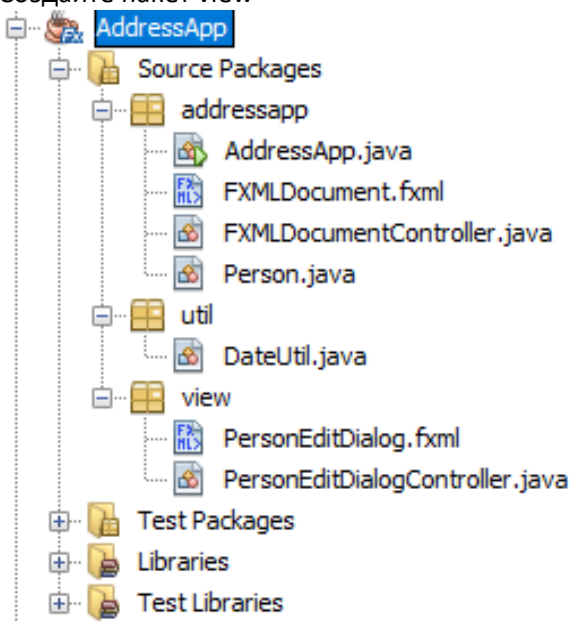
Чтобы посмотреть другие примеры использования диалогов, откройте статью Диалоги JavaFX.

Диалоги создания и изменения адресатов

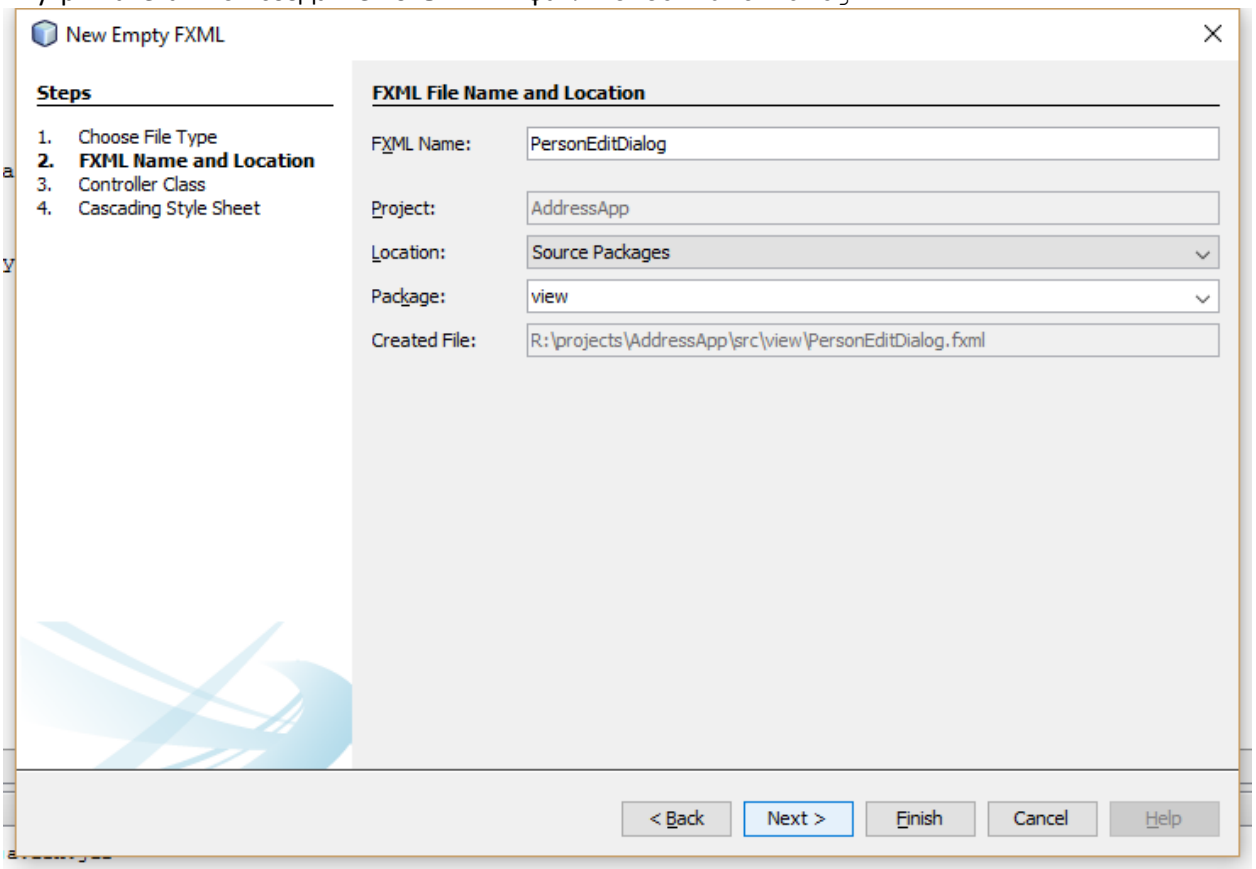
Для реализации методов-обработчиков создания и изменения адресатов нам придётся потрудится несколько больше. Необходимо создать новое пользовательское окно (то есть сцену) с формой, содержащей поля для заполнения всей необходимой информации об адресате.

Дизайн окна редактирования

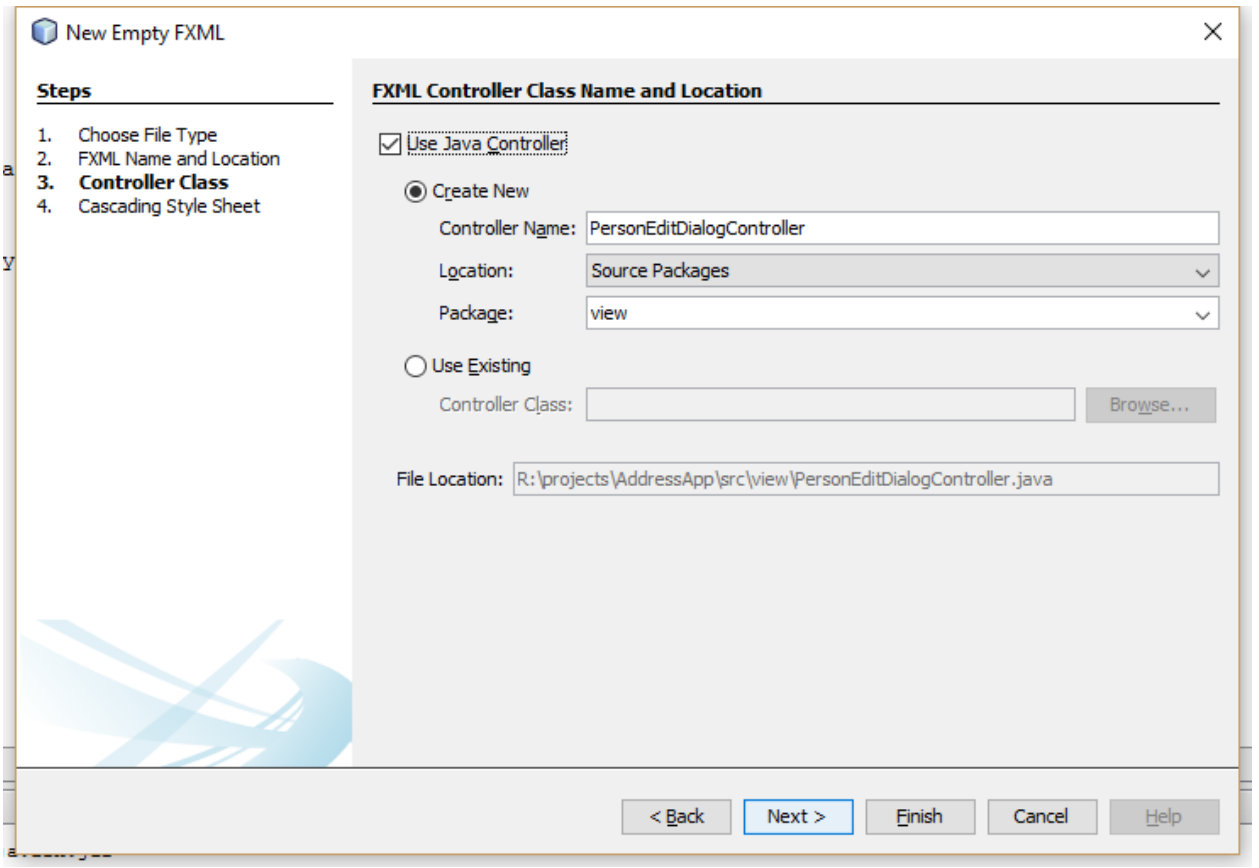
1. Создайте пакет view



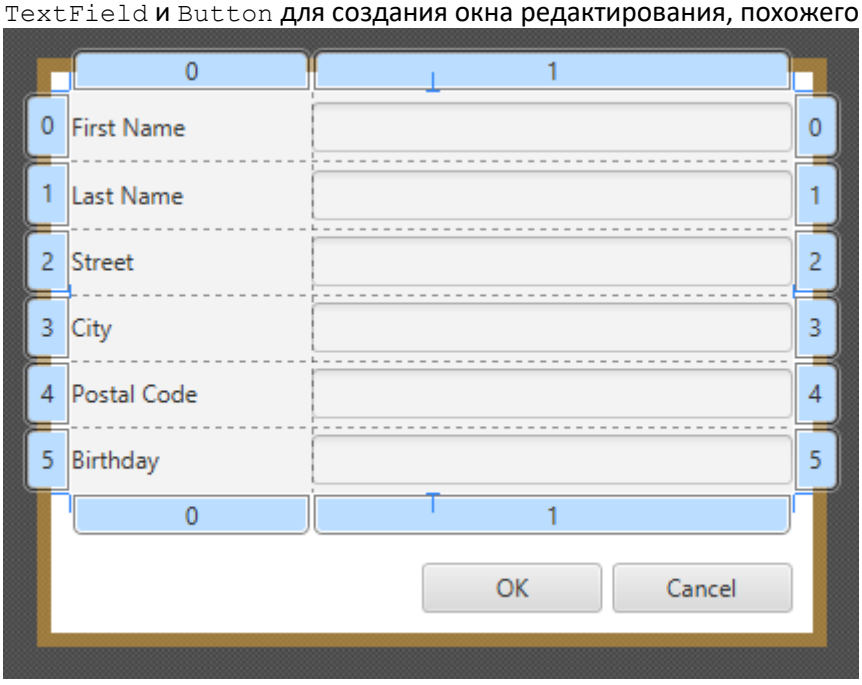
2. Внутри пакета `view` создайте новый fxml-файл `PersonEditDialog`.



3. Нажмите Next и отметьте пункт Use Java Controller



4. Finish
5. Отредактируйте `PersonEditDialog.fxml`. Используйте компоненты `GridPane`, `Label`, `TextField` и `Button` для создания окна редактирования, похожего на это:



Если что-то не работает, вы можете скачать [PersonEditDialog.fxml](#).

Создание контроллера

Создайте класс-контроллер для нового окна - `PersonEditDialogController.java`:

AddressApp\src\view\PersonEditDialogController.java

```
package view;

import addressapp.Person;
import javafx.fxml.FXML;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.TextField;
import javafx.stage.Stage;
import util.DateUtil;

/**
 * Окно для изменения информации об адресате.
 *
 * @author Computer School
 */
public class PersonEditDialogController {

    @FXML
    private TextField firstNameField;
    @FXML
    private TextField lastNameField;
    @FXML
    private TextField streetField;
    @FXML
    private TextField postalCodeField;
    @FXML
    private TextField cityField;
    @FXML
    private TextField birthdayField;

    private Stage dialogStage;
    private Person person;
    private boolean okClicked = false;

    /**
     * Инициализирует класс-контроллер. Этот метод вызывается автоматически
     * после того, как fxml-файл будет загружен.
     */
    @FXML
    private void initialize() {
    }

    /**
     * Устанавливает сцену для этого окна.
     *
     * @param dialogStage
     */
    public void setDialogStage(Stage dialogStage) {
        this.dialogStage = dialogStage;
    }

    /**
     * Задаёт адресата, информацию о котором будем менять.
     *
     * @param person
     */
    public void setPerson(Person person) {
        this.person = person;

        firstNameField.setText(person.getFirstName());
```

```
        lastNameField.setText(person.getLastName());
        streetField.setText(person.getStreet());
        postalCodeField.setText(Integer.toString(person.getPostalCode()));
        cityField.setText(person.getCity());
        birthdayField.setText(DateUtil.format(person.getBirthday()));
        birthdayField.setPromptText("dd.mm.yyyy");
    }

    /**
     * Returns true, если пользователь кликнул OK, в другом случае false.
     *
     * @return
     */
    public boolean isOkClicked() {
        return okClicked;
    }

    /**
     * Вызывается, когда пользователь кликнул по кнопке OK.
     */
    @FXML
    private void handleOk() {

        if (isInputValid()) {
            person.setFirstName(firstNameField.getText());
            person.setLastName(lastNameField.getText());
            person.setStreet(streetField.getText());
            person.setPostalCode(Integer.parseInt(postalCodeField.getText()));
            person.setCity(cityField.getText());
            person.setBirthday(DateUtil.parse(birthdayField.getText()));

            okClicked = true;
            dialogStage.close();
        }
    }

    /**
     * Вызывается, когда пользователь кликнул по кнопке Cancel.
     */
    @FXML
    private void handleCancel() {
        dialogStage.close();
    }

    /**
     * Проверяет пользовательский ввод в текстовых полях.
     *
     * @return true, если пользовательский ввод корректен
     */
    private boolean isInputValid() {
        String errorMessage = "";

        if (firstNameField.getText() == null || firstNameField.getText().length() ==
0) {
            errorMessage += "No valid first name!\n";
        }
        if (lastNameField.getText() == null || lastNameField.getText().length() ==
0) {
            errorMessage += "No valid last name!\n";
        }
        if (streetField.getText() == null || streetField.getText().length() == 0) {
```

```
            errorMessage += "No valid street!\n";
        }

        if (postalCodeField.getText() == null || postalCodeField.getText().length()
== 0) {
            errorMessage += "No valid postal code!\n";
        } else {
            // пытаемся преобразовать почтовый код в int.
            try {
                Integer.parseInt(postalCodeField.getText());
            } catch (NumberFormatException e) {
                errorMessage += "No valid postal code (must be an
integer)!\n";
            }
        }

        if (cityField.getText() == null || cityField.getText().length() == 0) {
            errorMessage += "No valid city!\n";
        }

        if (birthdayField.getText() == null || birthdayField.getText().length() == 0)
{
            errorMessage += "No valid birthday!\n";
        } else if (!DateUtil.validDate(birthdayField.getText())) {
            errorMessage += "No valid birthday. Use the format dd.mm.yyyy!\n";
        }

        if (errorMessage.length() == 0) {
            return true;
        } else {
            // Показываем сообщение об ошибке.
            Alert alert = new Alert(AlertType.ERROR);
            alert.initOwner(dialogStage);
            alert.setTitle("Invalid Fields");
            alert.setHeaderText("Please correct invalid fields");
            alert.setContentText(errorMessage);
            alert.showAndWait();

            return false;
        }
    }

}
```

Кое-какие заметки по поводу этого контроллера:

- Для указания адресата, данные которого должны быть изменены, метод `setPerson(...)` может быть вызван из другого класса;
- Когда пользователь нажимает на кнопку **OK**, то вызывается метод `handleOK()`. Первым делом данные, введённые пользователем, проверяются в методе `isInputValid()`. Если проверка прошла успешно, то объект адресата заполняется данными, которые ввёл пользователь. Эти изменения будут напрямую применяться к объекту адресата, который был передан в качестве аргумента метода `setPerson(...)`!
- Логическая переменная `okClicked` служит для определения того, какую из двух кнопок, `OK` или `Cancel` нажал пользователь.

Привязка класса-контроллера к fxml-файлу

Теперь мы должны связать наш класс-контроллер с fxml-файлом. Для этого выполните следующие действия:

1. Откройте файл `PersonEditDialog.fxml` в Scene Builder;
2. С левой стороны во вкладке *Controller* установите наш класс `PersonEditDialogController` в качестве значения параметра *Controller Class*;
3. Установите соответствующие значения **fx:id** для всех компонентов `TextField`;
4. В значениях параметров **onAction** для наших кнопок укажите соответствующие методы-обработчики.

Вызов диалога редактирования

Добавьте в класс `FXMLDocumentController` метод для загрузки и отображения диалога редактирования записей:

AddressApp\src\addressapp\FXMLDocumentController.java

```
 /**
 * Открывает диалоговое окно для изменения деталей указанного адресата.
 * Если пользователь кликнул OK, то изменения сохраняются в предоставленном
 * объекте адресата и возвращается значение true.
 *
 * @param person - объект адресата, который надо изменить
 * @return true, если пользователь кликнул OK, в противном случае false.
 */
public boolean showPersonEditDialog(Person person) {

    try {
        // Загружаем fxml-файл и создаём новую сцену
        // для всплывающего диалогового окна.
        FXMLLoader loader = new FXMLLoader();

loader.setLocation(PersonEditDialogController.class.getResource("PersonEditDi
alog.fxml"));
        AnchorPane page = (AnchorPane) loader.load();

        // Создаём диалоговое окно Stage.
        Stage dialogStage = new Stage();
        dialogStage.setTitle("Edit Person");
        dialogStage.initModality(Modality.WINDOW_MODAL);
        dialogStage.initOwner(null);
        Scene scene = new Scene(page);
        dialogStage.setScene(scene);

        // Передаём адресата в контроллер.
        PersonEditDialogController controller = loader.getController();
        controller.setDialogStage(dialogStage);
        controller.setPerson(person);

        // Отображаем диалоговое окно и ждём, пока пользователь его не
закроет
        dialogStage.showAndWait();
        return controller.isOkClicked();
    } catch (IOException e) {
        e.printStackTrace();
        return false;
    }
}
```

Добавьте еще методы в класс `FXMLDocumentController`. Когда пользователь будет нажимать на кнопки `New...` или `Edit...`, эти методы будут обращаться к методу `showPersonEditDialog(...)` в классе.

AddressApp\src\addressapp\FXMLDocumentController.java

```
/**
 * Вызывается, когда пользователь кликает по кнопке New...
 * Открывает диалоговое окно с дополнительной информацией нового адресата.
 */
@FXML
private void handleNewPerson() {
    Person tempPerson = new Person();
    boolean okClicked = showPersonEditDialog(tempPerson);
    if (okClicked) {
        personData.add(tempPerson);
    }
}

/**
 * Вызывается, когда пользователь кликает по кнопке Edit...
 * Открывает диалоговое окно для изменения выбранного адресата.
 */
@FXML
private void handleEditPerson() {
    Person selectedPerson = personTable.getSelectionModel().getSelectedItem();
    if (selectedPerson != null) {
        boolean okClicked =showPersonEditDialog(selectedPerson);
        if (okClicked) {
            showPersonDetails(selectedPerson);
            int selectedIndex =
personTable.getSelectionModel().getSelectedIndex();
            personData.set(selectedIndex, selectedPerson);
        }

    } else {
        // Ничего не выбрано.
        Alert alert = new Alert(AlertType.WARNING);
        alert.initOwner(null);
        alert.setTitle("No Selection");
        alert.setHeaderText("No Person Selected");
        alert.setContentText("Please select a person in the table.");
        alert.showAndWait();
    }
}
```

В приложении Scene Builder откройте представление `FXMLDocument.fxml` и для кнопок `New...` и `Edit...` задайте соответствующие методы-обработчики в параметре *On Action* .

Готово!

Сейчас у вас должно быть работающее приложение адресной книги. Оно позволяет добавлять, изменять и удалять адресатов. Это приложение так же осуществляет проверку всего, что вводит пользователь.

Если у вас что-то не работает, то вы можете сравнить свой класс `FXMLDocumentController` с FXMLDocumentController.java.

Я надеюсь, что идея и структура данного приложения поможет вам начать писать свои собственные приложения JavaFX! Удачи.