Formulářové aplikace v JavaFX a FXML

8

V minulém semináři jsme si ukázali tvorbu jednoduchých formulářových aplikací na platformě JavaFX. Je potřeba přiznat, že tento kód nebyl z pohledu dobrých programátorských zvyklostí úplně na úrovni, protože na jednom místě kombinoval prezentační logiku (vzhled aplikace) a aplikační (business) logiku (chování aplikace). JavaFX umožňuje tento problém elegantně vyřešit pomocí jazyka FXML, což je na XML postavený jazyk, kterým se definuje strom grafických uživatelských prvků (komponent), a jazyka CSS, kterým lze upravovat vzhled.

V tomto semináři se budeme věnovat tvorbě úplně stejných aplikací jako minule, jen k tomu využijeme jazyk FXML.

1 Hello World v FXML

Začneme tím, že ukázkový program HelloWorld rozdělíme do tří částí, kterými jsou

(i) formulář definovaný v jazyce FXML popisující strukturu okna;

(ii) *Controller* definující chování formuláře;

(iii) aplikace, která obstarává inicializaci a načtení formuláře.

1.1 Formulář v jazyce FXML

Jazyk FXML slouží k tomu, abychom pomocí elementů jazyka XML popsali strukturu jednotlivých komponent. Ukázkový formulář z úvodu minulého semináře by po přepsání do FXML mohl vypadat následovně.

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.layout.FlowPane?>
<?import javafx.scene.text.Text?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.control.Button?>

<FlowPane xmlns:fx="http://javafx.com/fxml/1"
    fx:controller="cz.upol.zp4jv.lecture08.HelloWorldController"
    alignment="CENTER" hgap="10" vgap="5">
```

```
    <Text text="Name: " />
    <TextField text="" fx:id="txtInput"/>
    <Button text="Ok" onAction="#submitAction"/>
</FlowPane>
```

Význam některých elementů, jako jsou `FlowPane`, `Text`, `Button` a jejich atributů, by měl být zřejmý, zastavme se však u těch méně obvyklých vlastností dokumentu.

V úvodu je uvedena sekvence instrukcí pro zpracování XML dokumentu `<?import ?>`, která udává, jaké elementy může FXML dokument obsahovat. V zásadě to odpovídá importu tříd, jak jej známe přímo z jazyka Java.

U kořenového elementu jsou vedeny dva atributy. Ten první, `xmlns:fx="http://javafx.com/fxml/1"`, definuje vlastní jmenný prostor označený `fx`, který slouží pro potřeby JavaFX. Druhý atribut, `fx:controller`, patří do tohoto jmenného prostoru a určuje kvalifikovaný[1] název *controller*u, tj. třídy, která bude obsluhovat chování tohoto formuláře.

S atributem z tohoto jmenného prostoru se setkáváme ještě u elementu `TextField`, kde atribut `fx:id` udává, pod jakým názvem bude tento element k nalezení v controlleru.

U elementu `Button` je reakce na stisk definována pomocí atributu `onAction` a názvem metody uvozeným znakem `#`.

1.2 Controller

V předchozí části jsme několikrát zmínili controller, třídu řešící chování (logiku) formuláře. Na tuto třídu nejsou kladeny žádné výrazné nároky. Musí se jednat o třídu, která má veřejný konstruktor bez parametru, aby by bylo možné controller jednoduše vytvořit.

Dále metody a atributy, které jsou odkazovány z FXML souboru, musí být označeny anotací `@FXML`,[2] jak ukazuje následující kód.

```
import javafx.fxml.FXML;
import javafx.scene.control.TextField;

public class HelloWorldController {
    @FXML private TextField txtInput;

    @FXML public void submitAction() {
        System.out.println("Hello: " + txtInput.getText());
    }
}
```

[1] balíček + cesta

[2] Anotacím se budeme podrobněji věnovat v některém z následujících seminářů. Nyní nám bude stačit vědět, že anotace umožňují vkládat do kódu dodatečné informace, které mohou různé nástroje číst a využívat pro svou činnost.

1.3 Aplikace

Třída, která obstará spuštění, je stejná jako to, co jsme již několikrát viděli. Hlavní rozdíl je v tom, jakým způsobem je načten kořen stromu jednotlivých komponent. To ukazuje následující kód.

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

    @Override
    public void start(Stage stage) throws Exception {
        Parent root = FXMLLoader.load(
                HelloWorldFX.class.getResource("/lecture08/HelloWorld.fxml"));
        stage.setTitle("Hello World");
        stage.setScene(new Scene(root, 300, 200));
        stage.show();
    }
}
```

Načtení je realizováno pomocí třídy `FXMLLoader` a její metody `load(URL)`, kde objekt URL udává, kde se FXML soubor nalézá. Na tomto místě bychom mohli použít i cestu k souboru na disku, to by ale přinášelo problémy s distribucí a spouštěním aplikace. Proto jsme použili zdroje (*resources*), které je možné mít jako součást projektu a následně přibalit do jar-souboru.

1.4 Poznámky

(i) **Při spuštění je potřeba uvést modul** `javafx.fxml`, tj. `--add-modules=javafx.controls,javafx.fxml`

(ii) Všimněme si, že FXMLLoader se postará nejen o vytvoření formuláře, ale i controlleru a jejich vzájemné provázání. Principy, kterými je toho dosaženo (včetně anotací), ukáže seminář č. 11 věnovaný *reflexi*.

(iii) Soubory s doplňujícími daty je zvykem ukládat do samostatného adresáře typicky pojmenovaného `res` nebo `resources`. Tento adresář by měl být označený, že obsahuje zdrojové soubory. V Eclipse se toho dosáhne pomocí *New* → *Source Folder*, v NetBeans je potřeba adresář nejdřív vytvořit a následně v *Project* → *Properties* → *Sources* jej doplnit do seznam *Source Package Folders*.

2 Složitější aplikace v FXML

Převod aplikace s telefonními kontakty z minulého semináře by měl být přímočarý, avšak najdou se tam i určitá úskalí, se kterými je nutné se vypořádat.

Samotné převedení formulářů do FXML by nemělo obsahovat nic zásadního, co zde ještě nebylo probráno, a proto teď FXML souborům nebudeme věnovat pozornost a necháme na p.t. čtenáři, aby sám prostudoval

obsah souborů `/res/lecture08/phonefx/*.fxml`.

2.1 Inicializace kompoment

Jeden z problémů, na který lze narazit, je inicializace hodnot komponent. Jako příklad uvažujme komponentu se seznamem telefonních čísel. Té je potřeba nastavit seznam položek. V tradičním programu by pro to bylo ideální místo v konstruktoru formuláře. To zde neplatí, protože formulář je vytvářen čistě programově na základě definice v FXML souboru. Konstruktor controlleru taky nepřichází v úvahu, protože nejdříve `FXMLLoader` vytvoří instanci controlleru, a až následně provede provázání controlleru a vytvořených komponent. Není tedy možné v konstruktoru controlleru jakkoliv přistupovat ke komponentám vytvořeným pomocí FXML.

Pro tyto situace ale může controller implementovat rozhraní `javafx.fxml.Initializable`, které obsahuje metodu `void initialize(URL location, ResourceBundle resources)`, která je zavolána po inicializaci stromu komponent a controlleru.

2.2 Inicializace controlleru

Controller je vytvářen automaticky třídou `FXMLLoader` a nelze mu proto předat do konstruktoru hodnoty, se kterými by měl být inicializován a se kterými by měl pracovat. Jednou z takových hodnot, kterou je nutné controlleru předat je *stage*, se kterou bude pracovat. Tu je potřeba znát pro vytvoření dialogu, zavření stage, atp. Případně můžeme chtít do controlleru předat další informace.

S touto vlastností se dá vypořádat pomocí objektu `FXMLLoader`, kdy si nejdříve vytvoříme instanci této třídy a zavoláme metodu `FXMLLoader.load()`, např. následovně.

```
FXMLLoader loader = new FXMLLoader(PhoneFX.class
   .getResource("/lecture08/phonefx/MainWindow.fxml"));
Parent root = loader.load();
```

V tento okamžik jsme dosáhli stejného efektu, jako bychom volali metodu `FXMLLoader.load(URL)`, avšak máme navíc objekt `loader`, který nám dává k dispozici přístup k vytvořenému controlleru. Tomu následně můžeme pomocí tradičních setterů nastavit potřebné vlastnosti, jak ilustruje následující kód.

```
PhoneFXController controller = loader.getController();
controller.setPrimaryStage(primaryStage);
```

3 Úpravy vzhledu s CSS

JavaFX má svým původem velice blízko k webovým technologiím a zároveň se snaží o oddělení prezentační a aplikační logiky, proto pro úpravu vzhledu adoptovala kaskádové styly, kterými je možné měnit vzhled aplikace.

Jednak lze používat styly přímo v FXML souborech.

```
<Text text="New Contact" style="-fx-font-size: 20pt" />
```

Nebo lze definovat vzhled v oddělených CSS souborech, ať už pro jednotlivé objekty, tak pro celé třídy.

```
#lbNotification {
        -fx-fill: red;
}

.button {
        -fx-font-size: 12pt;
}
```

Nastavení stylů se děje na úrovni jednotlivých scén.

```
scene.getStylesheets().add(
    PhoneFXController.class.getResource("/lecture08/phonefx/NewContact.css").toString());
```

V každém případě je dobré s touto funkcionalitou nakládat opatrně, protože radikální odchylky vzhledu od běžných standardů mohou komplikovat používání aplikace, případně mít nečekané důsledky.