Best Practices For Efficient Development Of JavaFX Applications

Grzegorz Kruk, CERN, Beams Department, Controls Group
What is JavaFX?
What is JavaFX?
What is JavaFX?
What is JavaFX?
What is JavaFX?
Separation Of Concerns
Separation Of Concerns

```xml
<BorderPane fx:id="rootPane"
    xmlns="http://javafx.com/javafx/8.0.60"
    xmlns:fx="http://javafx.com/fxml/1"
    fx:controller="cern.acsoft.app.optimizer.OptimizerController"
    --> ...
    <bottom>
      <HBox>
        <children>
          <Button fx:id="btnStart" text="START" onAction="#startOptimization"/>
          <Button fx:id="btnStop" text="STOP" onAction="#stopOptimization"/>
        </children>
      </HBox>
    --> ...
```
Separation Of Concerns

```java
public class OptimizerController {
    @Inject OptimizerModel model;
    @FXML Button btnStart;
    @FXML Button btnStop;
    // ...

    @FXML private void initialize() {
        btnStart.disableProperty().bind(model.optimizingProperty());
        btnStop.disableProperty().bind(model.optimizingProperty().not());
        // ...
    }

    @FXML void startOptimization() {
        model.setOptimizing(true);
        // ...
    }
}
```
Separation Of Concerns
Separation Of Concerns

Scene Builder

FXML

CSS

Controller

12/10/2017
Separation Of Concerns

Courtesy of E. Piselli and A. Akroh: TUPHA120
Architectural Guidelines Missing

- Every team/developer works out its own style
- Every application has its own structure

- Sharing development
- Sharing components
- Applications take over
- Maintenance
How to deal with this?

Structure

+

Tools
Project Structure
Project Structure
Project Structure
Project Structure
FxmlView

- Convention over Configuration
  - Inspired by Afterburner.fx library

- Small and simple to use
- Simplifies instantiation of views
- Built-in @Inject Dependency Injection (DI)
  - Models, Services, Properties
  - Plug-in external DI framework e.g. Spring or Google Guice
Project Generator

Java Project Details
Set the details of your new java project

- Project Name: accsoft-app-optimizer
- Main Package Name: cern.accsoft.app.optimizer
- Launcher name: OptimizerApp
- SVN Location: accsoft/app/accsoft-app-optimizer
- Template: Sample FX
- Template Description: This is a sample project it contains a basic application, rbac integration and some tests

Errors:
- Use default location

Location: C:\Workspace\gui\accsoft-app-optimizer

Finish | Cancel

TestFX Test
JUnit Test

12/10/2017
View Generator

[Image of JavaFX View Wizard]

JavaFX View Details
Create a new JavaFX view for the project: accsoft-app-optimizer

Source Folder Name: src/java
Parent Package Name: cern.accsoft.app.optimizer
View Name: Detector
Create view sub-package: [ ]
Template: Empty View
Template Description: Creates an empty FXML and corresponding controller e.g. for View Name: Header, it will create Header.fxml and HeaderController.java

[Image of file structure]

12/10/2017
Twofold Advantage

FxmlView

Simplifies and reduces code

Enforces structure and naming

Project Generator

Speeds up creation of projects and views

Promotes best practices
Keep FXML and Controller in Sync

```java
public class OptimizerController {
    @Inject OptimizerModel model;

    @FXML Button btnStart;
    @FXML Button btnStop;

    @FXML void startOptimization() {
        model.setOptimizing(true);
    }

    @FXML void stopOptimization() {
        model.setOptimizing(false);
    }
}

// ...
```
Keep FXML and Controller in Sync

```xml
<Button fx:id="btnStart" text="START" onAction="#startOptimization"/>
<Button fx:id="btnStop" text="STOP" onAction="#stopOptimization"/>
<Button fx:id="btnRefresh" text="Refresh" onAction="#refreshView"/>
```

```java
public class OptimizerController {
    @Inject OptimizerModel model;

    @FXML Button btnStart;
    @FXML Button btnStop;

    @FXML void startOptimization() {
        model.setOptimizing(true);
        //...
    }

    @FXML void stopOptimization() {
        model.setOptimizing(false);
        //...
    }
}
```
Keep FXML and Controller in Sync

```
public class OptimizerController {
    @Inject OptimizerModel model;

    @FXML Button btnStart;
    @FXML Button btnStop;

    @FXML void startOptimization() {
        model.setOptimizing(true);
        //...
    }

    @FXML void stopOptimization() {
        model.setOptimizing(false);
        //...
    }
}
```

<Button fx:id="btnStart" text="START" onAction="#startOptimization"/>
<Button fx:id="btnStop" text="STOP" onAction="#stopOptimization"/>
<Button fx:id="btnRefresh" text="Refresh" onAction="#refreshView"/>

Generate Missing FXML Fields and Handlers
Keep FXML and Controller in Sync

```java
public class OptimizerController {
    @Inject OptimizerModel model;

    @FXML Button btnStart;
    @FXML Button btnStop;
    @FXML Button btnRefresh;

    @FXML void startOptimization() {
        model.setOptimizing(true);
    }

    @FXML void stopOptimization() {
        model.setOptimizing(false);
    }

    @FXML void refreshView() {
    }
}
```

Generate Missing FXML Fields and Handlers
Summary

JavaFX: rich, mature and productive
…but doesn’t provide structural guidelines

- Don’t waste time on repeatable things
- Follow examples
- Focus on specific logic