springboot-javafx-support Documentation

Release latest

May 16, 2018

Contents

In an ideal world, a UX designer creates nice and cool scenes and elements spiced with CSS, while the developer writes the logic for the application.

This small library links Spring Boot with JavaFx 8. Let all your view and controller classes be Spring Beans and make use of all features in the Spring Universe.

You'll find a set of example applications at github and check my homepage. In the example I use concrete classes everywhere. In a real world application you can (and should) of course use interfaces for views and controllers and let Spring do the magic to instantiate the right bean.

CHAPTER 1

0. Prerequisites

You will at least need JDK1.8 patch level 40.

You find the latest springboot-javafx-support library at maven-central.

CHAPTER 2

1. Generate your GUI with FXML using SceneBuilder

You find SceneBuilder here: http://gluonhq.com/open-source/scene-builder/ Put the files in a folder called fxml in your classpath, so that Spring's resource loader can find them. Hint: Create a dedicated jar with all FXML, css and resource files and add it as a dependency using your preferred tooling (Maven, Gradle…).

CHAPTER 3

2. Create your view classes

Extend your view class from AbstractFxmlView and annotate it with @FXMLView. Name your class <FXML-File>View. E.g. given your FXML-file is named somelist.fxml the corresponding view class is SomeListView. When you want to name your class different, you need to add the fxml file name as value to your @FXMLView annotation: @FXMLView("/fxml/myviewfile.fxml")

CHAPTER 4

3. Create a Controller, Presenter

Create your controller class for your view as you defined it in the fxml file: fx:controller=”de.example.MyCoolPresenter” and annotate MyCoolPresenter with @FXMLController.

CHAPTER 5

4. Create a Starter class

Create a starter class extending AbstractJavaFxApplicationSupport. Annotate this one with @SpringBootApplication and call launchApp() from the main method.

CHAPTER 6

5. Style your views

You have multiple ways to style your view: First on is adding your JavaFX-css with SceneBuilder (the common JavaFX-way). Second is to add one or more css-files to the @FXMLView annotation: @FXMLView(css={"/css/company.css", "/css/project.css"}). Or third possibility: Add a property javafx.css=/global.css to your application.properties (or application.yaml).

CHAPTER 7

6. Adding resource bundles to the view

To i18n your application you can either add your properties files inside the package of your view class or add a bundle parameter to the @FXMLView annotation. Example: Your View is named foo.myapp.main.CoolView.class then your properties should be in the package foo.myapp.main as: cool.properties (default and fallback) and cool_de.properties (german), cool_fr.properties (french) etc. Or if you want to have your files reside in a different location (e.g. /i18n/messages_*.properties) then add your bundle by adding @FXMLView(bundle = "i18n.messages") Be aware of the dot because the FXMLLoader assumes that this is a classpath.

CHAPTER 8

Clone and improve me!

Please clone the sources from https://github.com/roskenet/springboot-javafx-support.git Pull requests welcome!

Felix Roske <felix.roske@zalando.de>