

Hallo Welt mit JavaFX unter Linux

- Jens Kapitza
 - M. Sc. Angewandte Informatik (Ingenieur)
 - Fachinformatiker-AE
 - Java Entwickler seit 2009 (V. 1.4)
 - Linux Anwender seit 2007
- kapitza@bluepair.de
- <http://www.bluepair.de>



Hallo Welt

- Erstellen einer Anwendung mit einem Textfeld und einem Button, der den Anwender begrüßen soll.
 - Benötigte Ressourcen: Betriebssystem mit grafischer Oberfläche, JDK, IDE
 - Vorgehensweisen: mit FXML oder Code-First-Ansatz.

Betriebssystem

- Linux + Xorg
- Linux + Monocle
- Windows
- FreeBSD

JVMs

- Oracle JDK
- OpenJDK
- ...
- Für Debian-basierende Systeme:
 - `sudo apt-get install openjdk-8-jdk openjdk-8-source openjdk-8-demo openjfx-source openjfx`

IDE

- `sudo apt-get install vim`
- Eclipse
- Netbeans
- ...

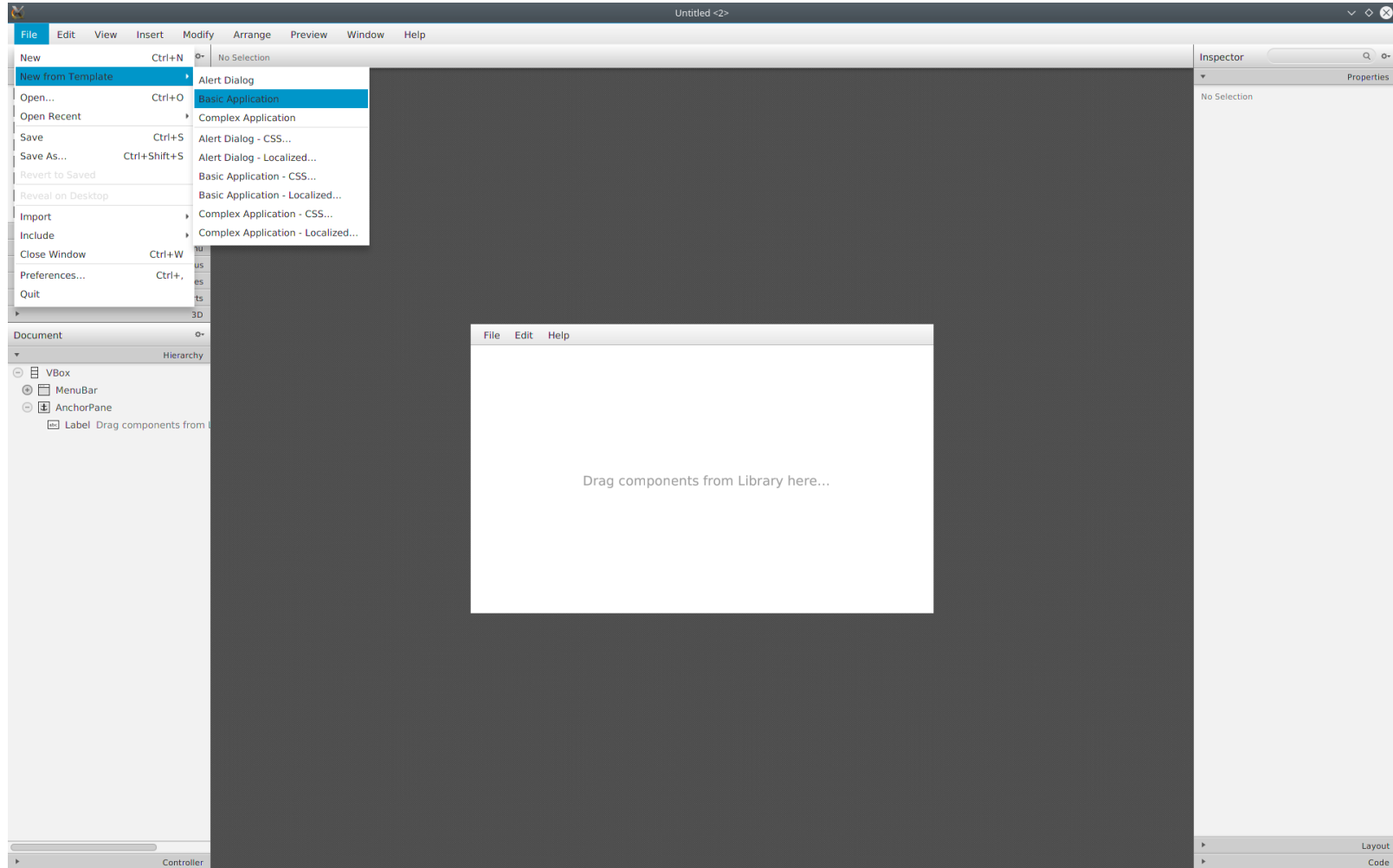
Scene Builder

- Scene Builder
- Design der Anwendung
 - Erzeugung von FXML Dateien
- Architekturmuster: **MVC** (Model-View-Controller)
 - Hier: nur die View
 - Controller kann aber generiert werden

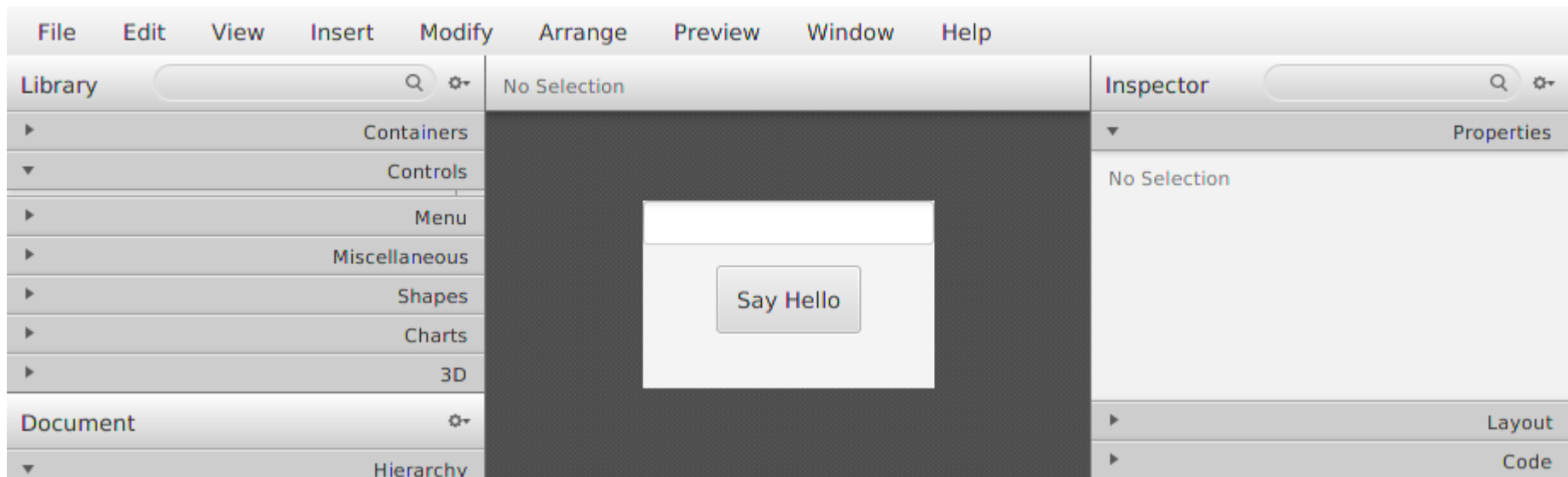
Tooling

- Maven
- Grandle
- Ant, ...
- Shellscript (unpraktisch)
- IDE

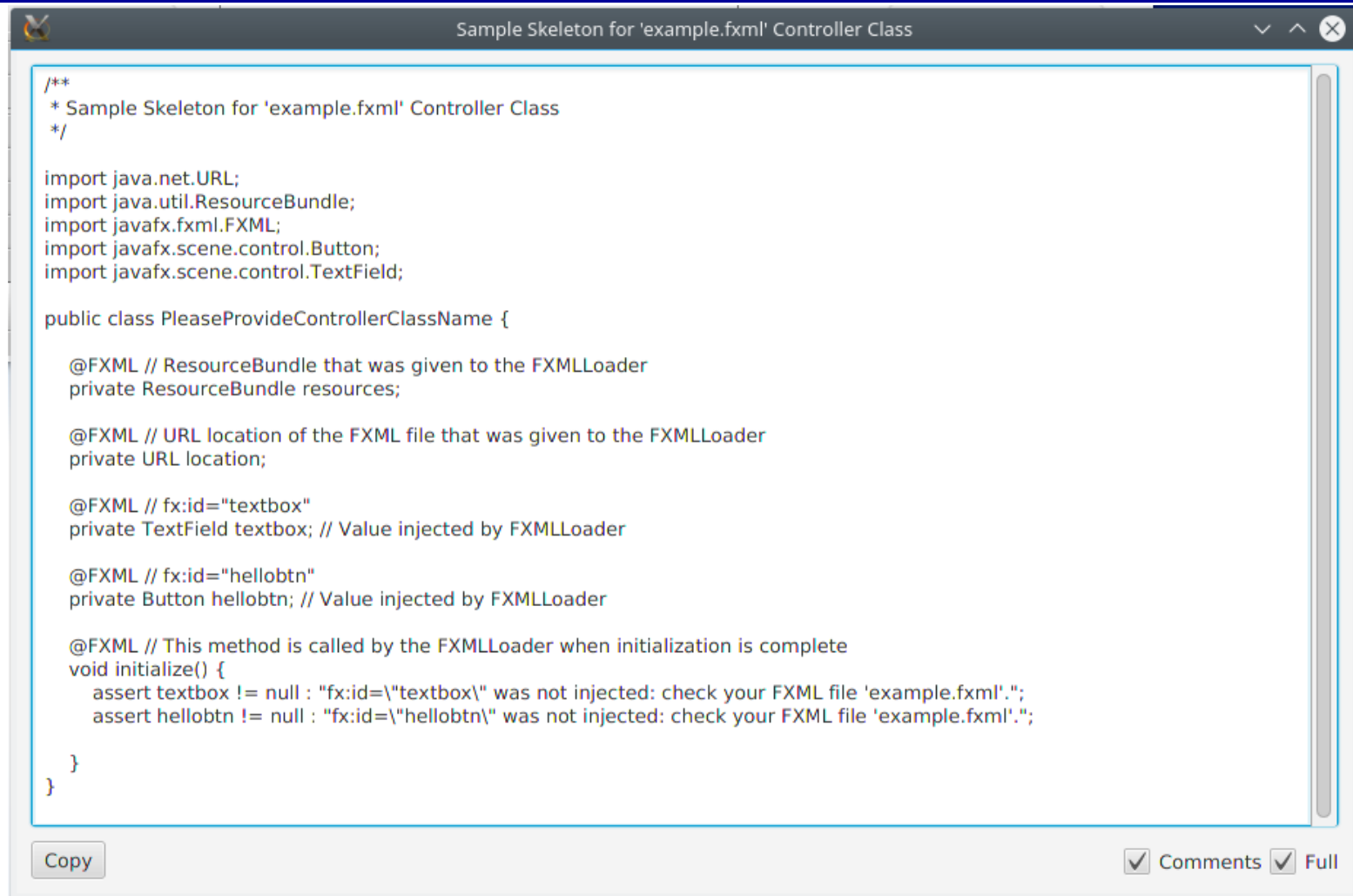
Basic Application



HelloWorld



HelloWorld Controller



The screenshot shows a code editor window titled "Sample Skeleton for 'example.fxml' Controller Class". The code is as follows:

```
/**
 * Sample Skeleton for 'example.fxml' Controller Class
 */

import java.net.URL;
import java.util.ResourceBundle;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;

public class PleaseProvideControllerClassName {

    @FXML // ResourceBundle that was given to the FXMLLoader
    private ResourceBundle resources;

    @FXML // URL location of the FXML file that was given to the FXMLLoader
    private URL location;

    @FXML // fx:id="textbox"
    private TextField textbox; // Value injected by FXMLLoader

    @FXML // fx:id="hellobtn"
    private Button hellobtn; // Value injected by FXMLLoader

    @FXML // This method is called by the FXMLLoader when initialization is complete
    void initialize() {
        assert textbox != null : "fx:id=\"textbox\" was not injected: check your FXML file 'example.fxml'.";
        assert hellobtn != null : "fx:id=\"hellobtn\" was not injected: check your FXML file 'example.fxml'.";
    }
}
```

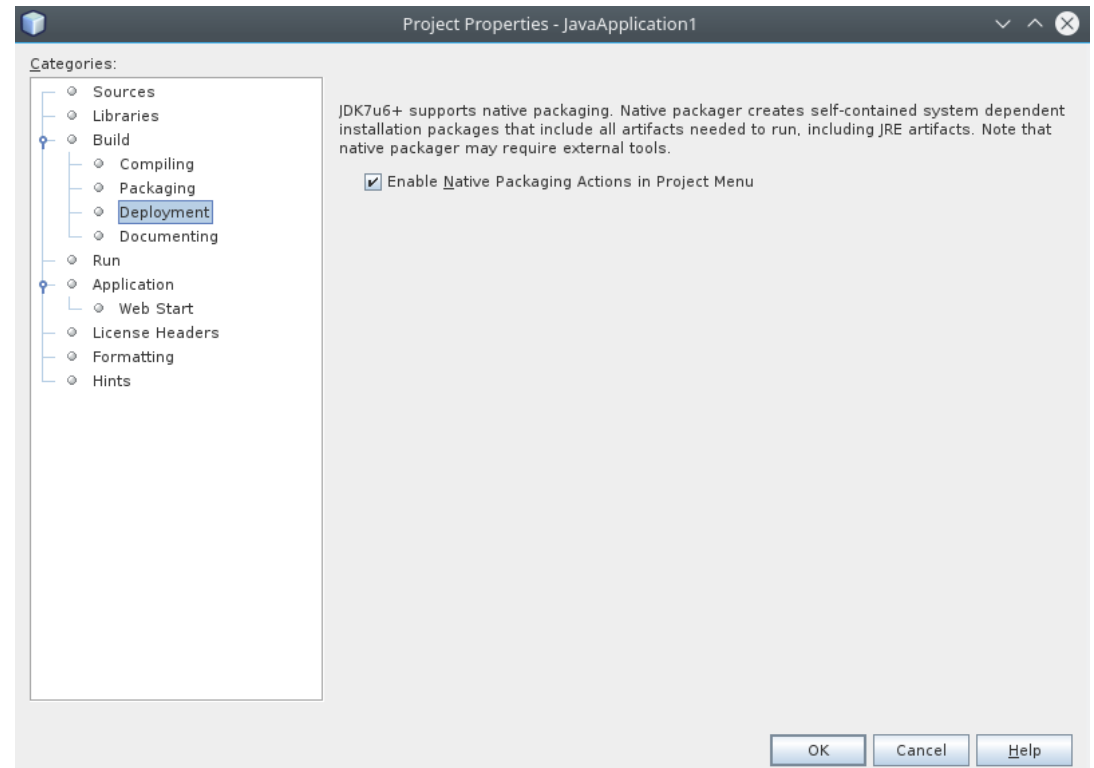
At the bottom left of the editor is a "Copy" button. At the bottom right are two checked checkboxes: "Comments" and "Full".

Demo

- Netbeans und zuvor gennantes Zeigen
- Daten entsprechend Kopieren

Self-contained Application Packaging

- Unterstützung via Ant/Maven/Grandle
- Dokumentation Oracle [EN](#)



Packaging Tools

- JavaFX native packager requires Debian Packager tools to create DEB package, but dpkg could not be found.
- Debian packages should specify a license. The absence of a license will cause some linux distributions to complain about the quality of the application.
- `sudo apt-get install dpkg-dev fakeroot depmake`

Erzeugen via Netbeans

The screenshot displays the NetBeans IDE environment for a JavaFX project named "JavaApplication1". The main window shows the source code for the class `JavaApplication1`, which extends `Application`. The code includes a `main` method and an overridden `start` method that initializes an `FXMLLoader`, sets a controller, and displays a scene with the title "HelloWorld".

```
1  /**
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5
6  */
7
8  package javaapplication1;
9
10 import javaafx.application.Application;
11 import javaafx.fxml.FXMLLoader;
12 import javaafx.scene.Parent;
13 import javaafx.scene.Scene;
14 import javaafx.stage.Stage;
15
16 /**
17 *
18 * @author kapitza
19 */
20 public class JavaApplication1 extends Application {
21
22     /**
23     * @param args the command line arguments
24     */
25     public static void main(String[] args) {
26         Application.launch(args);
27     }
28
29     @Override
30     public void start(Stage stage) throws Exception {
31         FXMLLoader loader = new FXMLLoader(getClass().getResource("example.fxml"));
32         loader.setController(new PleaseProvideControllerClassName());
33
34         Parent root = loader.load();
35
36         stage.setTitle("HelloWorld");
37         stage.setScene(new Scene(root));
38         stage.show();
39     }
40 }
```

The console window at the bottom shows the output of the build process, including the creation of the `build` directory, compilation of source files, and the launch of the application in native packaging mode. The output indicates that the application is being launched in native packaging mode and that no base JDK is specified.

Danke

Fragen?