Institut für Betriebswirtschaftslehre und Wirtschaftsinformatik

JavaFX for ooRexx

(Creating Powerful Portable GUIs)

Edited: 2017-11-09

Business Programming 2

Rony G. Flatscher

Wirtschaftsuniversität Wien ■ Welthandelsplatz 1 ■ A-1020 Wien

Agenda

- Brief historic overview
  - Java GUI packages for creating portable GUIs
- Overview of JavaFX
  - Concepts
  - ooRexx nutshell examples
- Roundup

Brief Historic Overview, 1

- Java package "java.awt"
  - "awt": "**a**bstract **w**indow **t**oolkit"
    - Java GUI classes for the most important GUI controls
      - Implement what
    - Uses JNI (Java Native Interface) to interact with the platform's GUI
      - "heavy" interaction with peer native GUI controls
    - Insulates the Java programmer from the platform
    - Event management and handling carried out in an "awt"-thread
  - Released with Java 1.0 (1996)

Brief Historic Overview, 2

- Java package "javax.swing"
  - "javax": Java extension
  - Java GUI classes for the most important GUI controls
    - "light-weight"
      - Uses Java2d to draw the controls
        - Text fields can be formatted with HTML style-attributes of that time
      - Contained in awt container
    - Swing class names may start with "J", if an awt class exists
      - e.g. javax.swingJButton vs. java.awt.Button
    - Adds PLAF
      - **P**luggable **L**ook **a**nd **F**eel
  - Released with Java 1.2 (1998)

Brief Historic Overview, 3

- Java package "javafx."
  - 2008 a standalone Java package
    - Also included a proper script engine named "JavaFX Script"
      - Removed with JavaFX 2.0 (2011)
  - Replaces java.awt and javax.swing
    - Introduces "Properties"
    - Totally new class hierarchy
    - Many new multiplatform classes for
      - e.g. charts, sound, video
  - Released with Java 1.8/8 (2014) as part of the JRE/JDK as "JavaFX8"
    - Already included in Java 1.7/7 updates as part of the JRE/JDK (7u15)

Overview of JavaFX, 1
Concepts

- "Property"
  - Contains a value, has setter and getter methods
  - Can be bound to other properties
    - Auto-update values!
  - GUI classes use properties to display and interact with

Overview of JavaFX, 2
Example "property_binding.rex"

```
--- import the Java class, allow it to be used like an ooRexx class thereafter
sipClz=bsf.import("javafx.beans.property.SimpleIntegerProperty")
num1 = sipClz~new(1)
say "num1:" num1 "|" num1~toString "|" num1~getValue
num2 = sipClz~new(2)
say "num2:" num2 "|" num2~toString "|" num2~getValue
say
sum=num1~add(num2)
say "sum: " sum
say "sum: " sum~toString "|" sum~getValue "|" sum~toString
say "---"
say "num1:" num1~getValue "num2:" num2~getValue "-> sum:" sum~getValue
say "setting 'num1=2' ..."
num1~set(2)
say "num1:" num1~get      "num2:" num2~get      "-> sum:" sum~get
say "setting 'num2=3' ..."
num2~set(3)
say "num1:" num1~getValue "num2:" num2~getValue "-> sum:" sum~getValue

::requires "BSF.CLS"    -- get Java support
```

Output:

```
num1: javafx.beans.property.SimpleIntegerProperty@67c3bb | IntegerProperty [value: 1] | 1
num2: javafx.beans.property.SimpleIntegerProperty@19bb37 | IntegerProperty [value: 2] | 2

sum:  javafx.beans.binding.Bindings$15@1d10166
sum:  IntegerBinding [invalid] | 3 | IntegerBinding [value: 3]
---
num1: 1 num2: 2 -> sum: 3
setting 'num1=2' ...
num1: 2 num2: 2 -> sum: 4
setting 'num2=3' ...
num1: 2 num2: 3 -> sum: 5
```

Overview of JavaFX, 3
Concepts

- FXML
  - **FX M**arkup **L**anguage
  - Allows to define the GUI as an XML file
    - Tool SceneBuilder to create GUIs interactively!
      - Cf. http://gluonhq.com/labs/scene-builder/
  - Allows to set up an available javax.script engine
    - Run script code, e.g. for events!
  - A Java loader class will read the FXML and create the GUI
    - GUI controls with 'fx:id' attribute directly addressable!

Overview of JavaFX, 4
Concepts

- FXML (continued)
  - Invoking script code occurs with the help of javax.script
    - Creates a separate Engine for each FXML document
    - Each invocation gets its own ScriptContext with a GLOBAL_SCOPE and ENGINE_SCOPE Binding
      - GLOBAL_SCOPE Binding contains
        - The created JavaFX GUI controls that have the attribute 'fx:id' set!
        - A Rexx script can access all of these GUI controls

Overview of JavaFX, 5
Concepts

- MVC
  - **M**odel-**V**iew-**C**ontroller (introduced with Smalltalk-76)
    - **M**odel – the data to maintain
      - Our program
    - **V**iew – the program to display the data
      - Our program, JavaFX or a combination of both
      - View and model can be bound with Properties!
    - **C**ontroller – to control interaction with the model and view
      - Our program serving as the bridge between the model and the view(s)

Overview of JavaFX, 6
Example 1

Output:

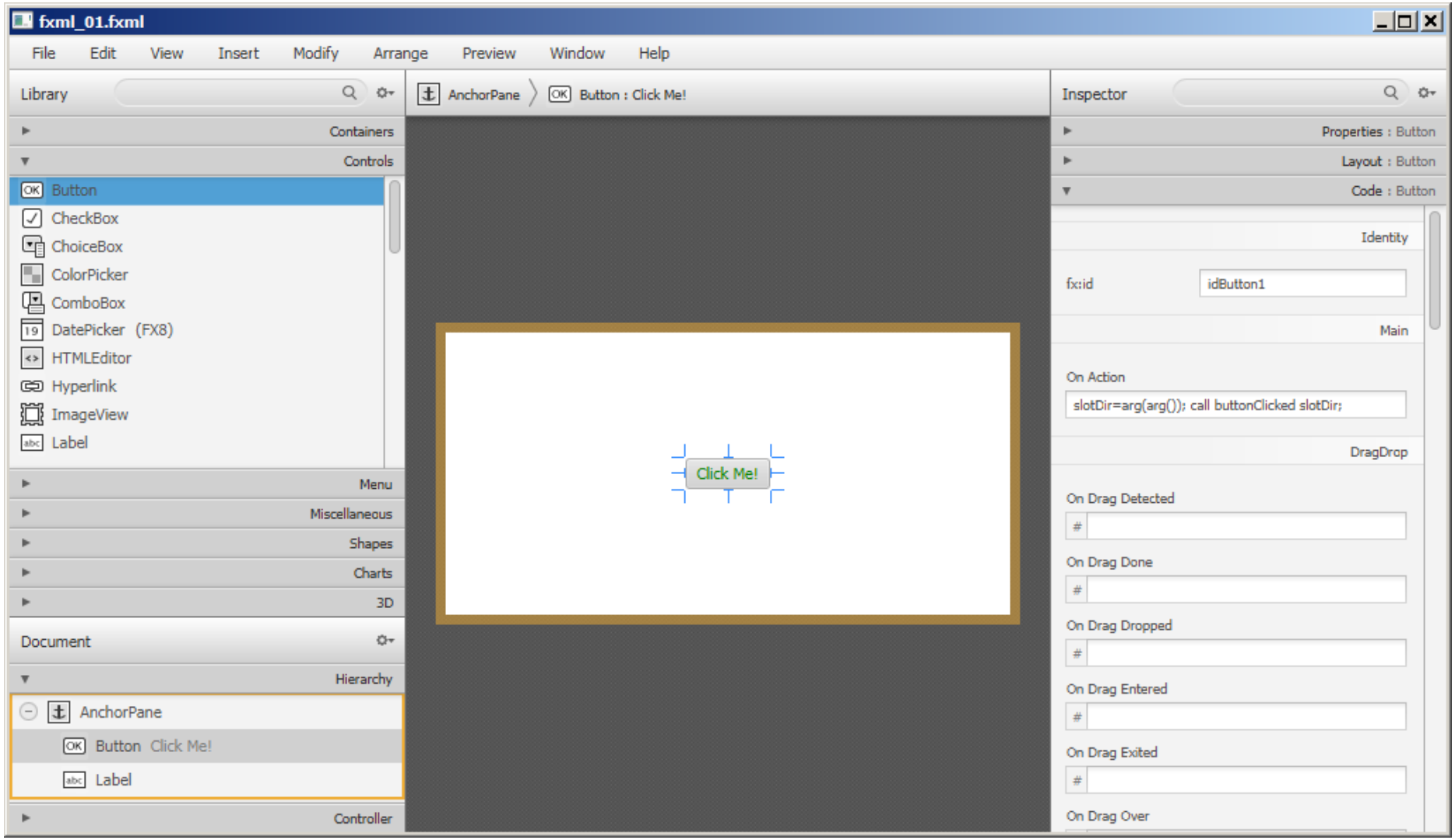
```
E:\fxml_01>fxml_01.rex

REXXout>2017-10-31T19:02:11.047000: arrived in routine 'buttonClicked' ...
REXXout>... current value of label=[]
REXXout>...     new value of label=[Clicked at: 2017-10-31T19:02:11.047000]
REXXout>
REXXout>2017-10-31T19:02:29.911000: arrived in routine 'buttonClicked' ...
REXXout>... current value of label=[Clicked at: 2017-10-31T19:02:11.047000]
REXXout>...     new value of label=[Clicked at: 2017-10-31T19:02:29.911000]
REXXout>
```

Overview of JavaFX, 7
Example 1, Three Files

- FXML_01_Document.fxml
  - The FXML file defining our GUI
    - Defines "rexx" to be used as the script language
    - Defines an AnchorPane containing a
      - Button with fx:id="button" (with Rexx code) and a
      - Label with fx:id="label"
      - Text (textFill) of both controls is GREEN
- fxml_01_controller.rex
  - Defines a public Rexx routine "klickButtonAction"
- fxml_01.rex
  - Runs the program using the javafx package

Overview of JavaFX, 8
Example 1, Using "SceneBuilder" for the Dialog



Overview of JavaFX, 9
Example 1, "FXML_01_Document.fxml"

```
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.layout.AnchorPane?>
<!-- comment: the following process instruction (PI) defines the Java script engine named 'rexx'
              to be used for the code in event attributes like 'onAction' -->
<?language rexx?>

<AnchorPane id="AnchorPane" prefHeight="200" prefWidth="400" xmlns="http://javafx.com/javafx/8.0.65"
            xmlns:fx="http://javafx.com/fxml/1">

    <!-- comment: defines the attribute in GLOBAL_SCOPE named 'rexxStarted' to be used for labelStart -->
    <fx:script source="fxml_01_controller.rex" />

    <!-- comment: define the JavaFx controls that make up the GUI, all controls that possess a fx:id
                  attribute are stored by their id in the ScriptContext's GLOBAL_SCOPE -->

    <children>
        <!-- comment: the Rexx code in the 'onAction' attribute will be invoked by JavaFX via a
                      Rexx call; note that JavaFX will remove any newline characters between the
                      double-quotes ("), hence each Rexx statement is explicitly ended with the
                      semi-colon character -->
        <Button fx:id="idButton1" layoutX="170.0" layoutY="89.0"
                      onAction="slotDir=arg(arg()); call buttonClicked slotDir;"
                      text="Click Me!" textFill="GREEN"
        />

        <Label fx:id="idLabel1" alignment="CENTER" contentDisplay="CENTER"
                                layoutX="76.0" layoutY="138.0"
                                minHeight="16" minWidth="49"
                                prefHeight="16.0" prefWidth="248.0"
                                textFill="GREEN" />
    </children>
</AnchorPane>
```

Overview of JavaFX, 10
Concept of "RexxScript Annotation"

- A "boon" implemented into the ooRexx javax.script RexxEngine
  - A Rexx block comment, which may be one of
    - /* @get(idx1 idx2 ...) */
      - Fetches entries named "idx1", "idx2" from the ScriptContext's Bindings and makes them available as Rexx variables by the same name ("IDX1", "IDX2")
    - /* @set(idx1 idx2 ...) */
      - Sets the entries named "idx1", "idx2" in the ScriptContext Bindings, using the values of the Rexx variables "IDX1", "IDX2"
    - /* @showsource */
      - Displays the Rexx code that gets executed by the RexxEngine

Overview of JavaFX, 11
Example 1, "fxml_01_controller.rex"

- Defines the *public* Rexx routine "klickButtonAction"
  - Usually there is one controller for each FXML file
  - Fetches the supplied slotDir argument
    - Can be used to access the ScriptContext and its Bindings
    - This example uses "RexxScript annotations"
  - Fetches and updates the Label with fx:id="label"
    - Taking advantage of "RexxScript annotations"
      - `/* @get(label) */` instead of coding:

        ```
        label=slotDir~scriptContext~getAttribute("label")
        ```
    - Outputs information to stdout

Overview of JavaFX, 12
Example 1, "fxml_01_controller.rex"

```
/* This routine will be called from the Rexx code defined in the Button element in
   with the fx:id="button" the "onAction" attribute in the FXML Button definition */
::routine buttonClicked public
  slotDir=arg(arg())  -- note: last argument is the slotDir argument from BSF4ooRexx
  now=.dateTime~new -- time of invocation
  say now": arrived in routine 'buttonClicked' ..."
  /* RexxScript annotation fetches "label" from ScriptContext
     and makes it available as the Rexx variable "LABEL":  */
  /* @get(idLabel1) */
  say '... current value of label='pp(idLabel1~getText)
  idLabel1~text="Clicked at:" now         -- set text property
  say '...     new value of label='pp(idLabel1~getText)
  say
```

Responsible for updating the Label object using the (fx:)id value (case-sensitive!) "label" and for the following output to stdout:

```
E:\fxml_01>fxml_01.rex

REXXout>2017-10-31T19:02:11.047000: arrived in routine 'buttonClicked' ...
REXXout>... current value of label=[]
REXXout>...     new value of label=[Clicked at: 2017-10-31T19:02:11.047000]
REXXout>
REXXout>2017-10-31T19:02:29.911000: arrived in routine 'buttonClicked' ...
REXXout>... current value of label=[Clicked at: 2017-10-31T19:02:11.047000]
REXXout>...     new value of label=[Clicked at: 2017-10-31T19:02:29.911000]
REXXout>
```


Overview of JavaFX, 13
Concepts, Running a JavaFX Application

- A JavaFX application uses
  - Stages to display Scenes
    - A Stage is usually some kind of a window
    - A Scene is a GUI container placed on a Stage for interaction
  - There may be multiple Stages and Scenes
- Abstract class javafx.application.Application
  - Initializes JavaFX, creates a ("primary") Stage and invokes the abstract method start(Stage primaryStage) in its launch method
  - Defines a Rexx class implementing the method start
  - Uses BsfCreateRexxProxy() to create a proxied Application, send it the launch message

Overview of JavaFX, 14
Example 1, "fxml_01.rex"

- Defines the Rexx class RexxApplication
  - Implements the abstract method start
  - A Rexx instance will be used in BsfCreateRexxProxy()
    - The resulting Java object (of type javafx.application.Application) gets the launch message sent to it, which eventually will invoke the method start, causing a Rexx message of that name to be sent to the embedded Rexx instance

Overview of JavaFX, 15
Example 1, "fxml_01.rex"

```
rxApp=.RexxApplication~new -- create Rexx object that will control the FXML set up
jrxApp=BSFCreateRexxProxy(rxApp, ,"javafx.application.Application")
jrxApp~launch(jrxApp~getClass, .nil)    -- launch the application, invokes "start"

::requires "BSF.CLS"    -- get Java support

-- Rexx class defines "javafx.application.Application" abstract method "start"
::class RexxApplication -- implements the abstract class "javafx.application.Application"

::method start          -- Rexx method "start" implements the abstract method
  use arg primaryStage  -- fetch the primary stage (window)
  primaryStage~setTitle("Hello JavaFX from ooRexx! (Green Version)")

   -- create an URL for the FMXLDocument.fxml file (hence the protocol "file:")
  fxmlUrl=.bsf~new("java.net.URL", "file:fxml_01.fxml")
   -- use FXMLLoader to load the FXML and create the GUI graph from its definitions:
  rootNode=bsf.loadClass("javafx.fxml.FXMLLoader")~load(fxmlUrl)

  scene=.bsf~new("javafx.scene.Scene", rootNode)    -- create a scene for our document
  primaryStage~setScene(scene)  -- set the stage to our scene
  primaryStage~show             -- show the stage (and thereby our scene)
```


Overview of JavaFX, 15
Concept, JavaFX without Employing FXML

- FXML contains all GUI declarations
  - Which javafx controls
  - Position of javafx controls
  - Attributes of javafx controls, e.g.
    - Color information
    - Position and size information
    - Unique and case-sensitive fx:id values for javafx controls
- Without taking advantage of FXML
  - The code needs to do all this setting up
  - Needs to take over event handling

Overview of JavaFX, 16
Example 1, "javafx_01.rex"

```
rxApp=.RexxApplication~new  -- create Rexx object that will control the FXML set up
   -- rxApp will be used for "javafx.application.Application"
jrxApp=BSFCreateRexxProxy(rxApp, ,"javafx.application.Application")
jrxApp~launch(jrxApp~getClass, .nil)    -- launch the application, invokes "start"

::requires "BSF.CLS"    -- get Java support

-- Rexx class defines "javafx.application.Application" abstract method "start"
::class RexxApplication -- implements the abstract class "javafx.application.Application"
::method start          -- Rexx method "start" implements the abstract method
  use arg primaryStage  -- fetch the primary stage (window)
  primaryStage~setTitle("Hello JavaFX from ooRexx! (Blue Version)")

  -- get Java class objects to ease access to their constants (static fields)
  colorClz=bsf.loadClass("javafx.scene.paint.Color") -- JavaFX colors
  cdClz=bsf.loadClass("javafx.scene.control.ContentDisplay") -- ContentDisplay constants
  alClz=bsf.loadClass("javafx.geometry.Pos")      -- alignment constants (an Enum class)

  root=.bsf~new("javafx.scene.layout.AnchorPane") -- create the root node
  root~prefHeight=200           -- or: root~setPrefHeight(200)
  root~prefWidth=400            -- or: root~setPrefWidth(400)
      -- define the Label
  lbl=.bsf~new("javafx.scene.control.Label")
  lbl~textFill=colorClz~BLUE    -- or: lbl~setTextFill(colorClz~BLUE)
  lbl~setLayoutX(76)            -- or: lbl~layoutX=76
  lbl~setLayoutY(138)           -- or: lbl~layoutY=138
  lbl~prefHeight="16.0"         -- or: lbl~setPrefHeight("16.0")
  lbl~prefWidth="248.0"         -- or: lbl~setPrefWidth("248.0")
  lbl~contentDisplay=cdClz~CENTER       -- or: lbl~setContentDisplay (cdClz~CENTER)
  lbl~alignment=alClz~valueOf("CENTER") -- or: lbl~setAlignment(alClz~valueOf("CENTER"))

... continued on next slide ...
```

Overview of JavaFX, 17
Example 1, "javafx_01.rex"

```
... continued from previous slide ...

    -- define and add the Button, assign values as if we deal with Rexx attributes
  btn=.bsf~new("javafx.scene.control.Button")
  btn~textFill=colorClz~BLUE     -- or: btn~setTextFill(colorClz~BLUE)
  btn~layoutX=170                -- or: btn~setLayoutX(170)
  btn~layoutY=89                 -- or: btn~setLayoutY(89)
  btn~text="Click Me!"           -- or: btn~setText("Click Me!")
      -- create a Rexx ButtonHandler, wrap it up as a Java RexxProxy
  rh=.RexxButtonHandler~new(lbl)-- create Rexx object, supply it the label "lbl"
  jrh=BSFCreateRexxProxy(rh, ,"javafx.event.EventHandler")
  btn~setOnAction(jrh)           -- forwards "handle" message to Rexx object
      -- add the button and label to the AnchorPane object
  root~getChildren~~add(btn)~~add(lbl)
      -- put the scene on the stage
  primaryStage~setScene(.bsf~new("javafx.scene.Scene", root))
  primaryStage~show -- show the stage (window) with the scene

   -- Rexx class which handles the button presses
::class RexxButtonHandler   -- implements "javafx.event.EventHandler" interface
::method init       -- Rexx constructor method
  expose label      -- allow direct access to ooRexx attribute
  use arg label     -- save reference to javafx.scene.control.Label

::method handle     -- will be invoked by the Java side
  expose label      -- allow direct access to ooRexx attribute, not used in this example
  -- use arg event, slotDir    -- expected arguments
  now=.dateTime~new -- time of invocation
  say now": arrived in method 'handle' ..."
  say '... current value of label='pp(label~getText)
  label~text="Clicked at:" now         -- set text property
  say '...     new value of label='pp(label~getText)
  say
```

Overview of JavaFX, 18
Concepts

- DOM and CSS
  - All javafx controls are organized in a DOM tree
    - DOM: Document Object Model
    - W3C standard
  - All javafx controls can be formatted using CSS
    - CSS: Cascading Style Sheets
    - Defining styles for all nodes of the DOM tree
  - JavaFX employs webkit for rendering
    - Open source rendering engine
    - e.g. Apple uses it for Safari, Google forked it for Chrome

Overview of JavaFX, 19
Example 2, Six Files

- Image files  
  - bsf4oorexx_032.png (application icon), oorexx_032.png (background)
- Dialog files
  - fxml_02.css, FXML_02_Document.fxml, fxml_02_controller.rex
  - Automatic substitution of values (problems with SceneBuilder 2.0!)
    - %year, %clickMe: FXML_02_de.properties, FXML_02_en.properties
    - $-prefix: fetch value from ScriptContext at startup
    - ${name}: fetch value continuously from ScriptContext
- Starting the application
  - fxml_02.rex

Overview of JavaFX, 20
Example 2, "xml_02.css"

```
/* Some Java-FX CSS definitions, cf. <http://docs.oracle.com/javafx/2/get_started/css.htm>,
   especially: <https://docs.oracle.com/javafx/2/api/javafx/scene/doc-files/cssref.html>.
   The following style definitions only have the purpose to demonstrate the power available.
   These definitions are only meant for fun and starting point for experiments, not for
   professional use! :)    2016-11-22, rgf  */

/* define the background of the scene, will be applied to AnchorPane: */
.root {
     -fx-background-image: url("bsf4oorexx_032.png");
     -fx-background-color: LightGoldenRodYellow;
}
/* this is the basic formatting for all Label:s */
.label {
    -fx-font-size: 11px;
    -fx-font-weight: bold;
    -fx-text-fill: #333333;
    -fx-effect: dropshadow( gaussian , rgba(255,255,255,0.5) , 0,0,0,1 );
    -fx-border-color: red;
    -fx-border-radius: 3px;
    -fx-border-style: dashed;
    -fx-border-width: 1px;
}
/* this will change the appearence of Button a little bit: */
.button {
    -fx-text-fill: royalblue;
    -fx-font-weight: 900;
}

/* this will apply alpha (fourth value) to get the background to shine thru the
   label with the class "rexxInfo"; to be able to apply the alpha, one
   needs to turn the hexadecimal values into their decimal representations like:
   hence: oldlace = #fdf5e6 -> fd~x2d f5~x2d e5~x2d -> rgb(253, 245, 230)
*/
.rexxStarted {
     -fx-background-color: rgb(253, 245, 230, 0.75) ;
     -fx-text-fill: royalblue;
}
```

Overview of JavaFX, 21
Example 2, "FXML_02_Document.fxml"

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

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.layout.AnchorPane?>

<!-- processing instruction (PI) defines the Java script engine named 'rexx'
     to be used to execute programs (fx:script or in event attributes) -->
<?language rexx?>

<AnchorPane id="AnchorPane" fx:id="idRoot" prefHeight="240.0" prefWidth="480.0"
            styleClass="root" stylesheets="@fxml_02.css"
            xmlns:fx="http://javafx.com/fxml/1">

    <!-- defines entries for ScriptContext bindings, public routine 'klickButton' -->
    <fx:script source="fxml_02_controller.rex" />

    <children>
        <Label fx:id="idLabelRexxStarted" alignment="CENTER" layoutX="50.0"
               layoutY="26.0"                minHeight="16" minWidth="69"
               prefHeight="16.0" prefWidth="380.0" styleClass="rexxStarted"
               stylesheets="@fxml_02.css" text="$rexxStarted" />

        <Button fx:id="idButton" layoutX="210.0" layoutY="137.0" onAction=
                  "slotDir=arg(arg()) /* last argument added by BSF4ooRexx */;
                   say ' /// onAction eventHandler calling routine ''klickButton'' \\\';
                   call klickButton slotDir /* now process the event */; "
           text="%clickMe" />

        <Label fx:id="idLabelYear" layoutX="50.0" layoutY="175.0" minHeight="16"
               minWidth="20" style="-fx-background-color:palegoldenrod;" text="%year" />

... continued on next slide ...
```

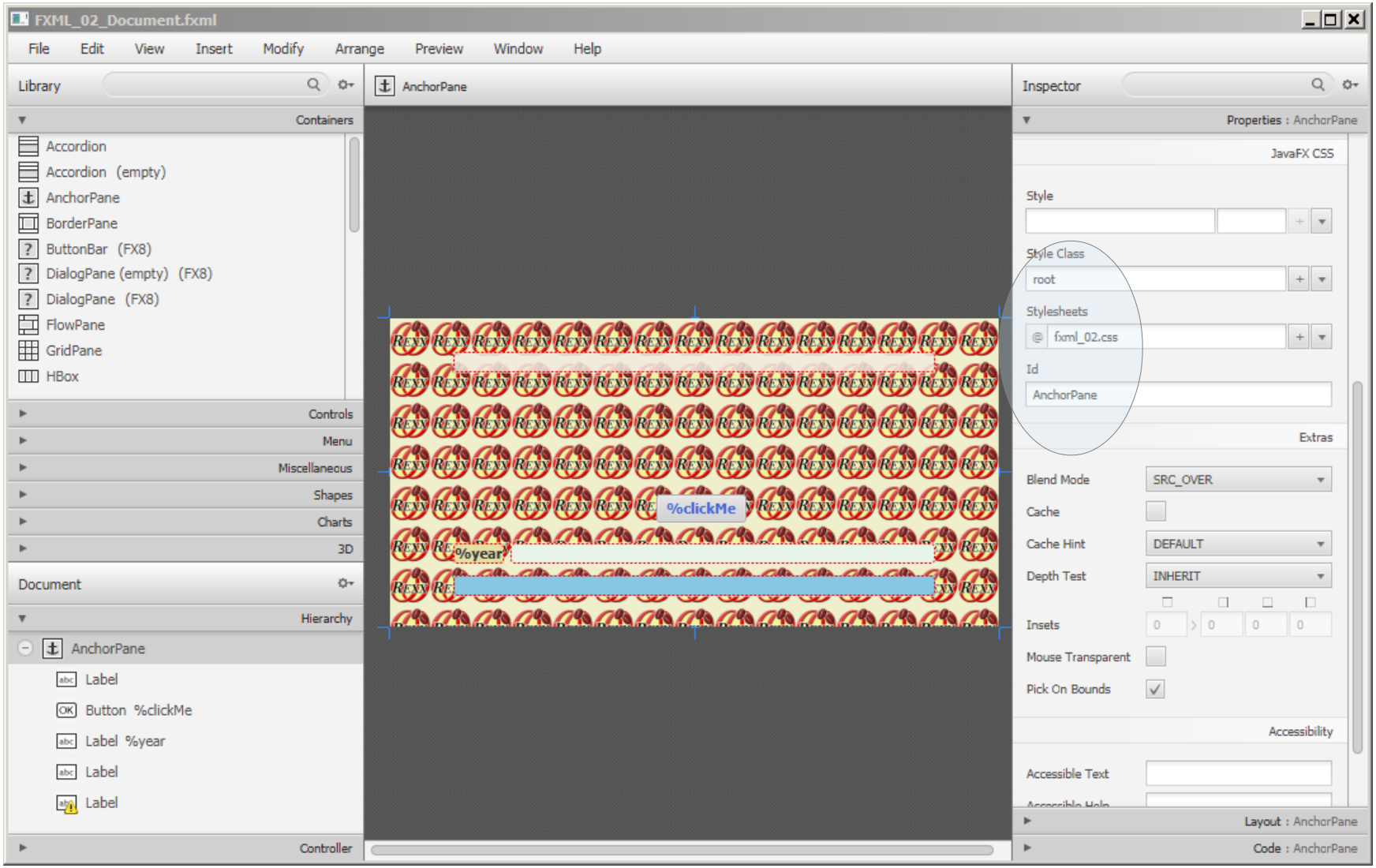
Overview of JavaFX, 22
Example 2, "FXML_02_Document.fxml"

```
... continued from previous slide ...

        <Label fx:id="idLabel" layoutX="95.0" layoutY="175.0" minHeight="16"
               minWidth="49" prefHeight="16.0" prefWidth="335.0"
               style="-fx-background-color: honeydew;" />

        <Label fx:id="idLabelRexxInfo" alignment="CENTER" layoutX="50.0" layoutY="200.0"
               minHeight="16.0" minWidth="49.0" prefHeight="16.0" prefWidth="380.0"
               style="-fx-background-color: skyblue; -fx-cursor: wait;
                      -fx-font-family: serif; -fx-font-weight: lighter;"
               text="${rexxInfo}" />
    </children>
</AnchorPane>
```

Overview of JavaFX, 23
Example 2, Using "SceneBuilder" for the Dialog



Overview of JavaFX, 24
Example 2

Overview of JavaFX, 25
Example 2, "fxml_02_controller.rex"

```
    slotDir=arg(arg())       -- last argument is the slotDir argument, added by BSF4ooRexx
    started=.dateTime~new    -- get current date and time
    parse source s           -- get the source information and show it
    say "just arrived at" pp(started)": parse source ->" pp(s)

    sc=slotDir~scriptContext -- get the ScriptContext entry from slotDir
       -- add the attribute "rexxStarted" to the ScriptContext's GLOBAL_SCOPE Bindings
    sc~setAttribute("rexxStarted", "Rexx started at:" started~string, sc~global_scope)
    parse version v          -- get Rexx version, display it in the "rexxInfo" label
    sc~setAttribute("rexxInfo", "Rexx version:" v, sc~global_scope)
       -- set attribute at ENGINE_SCOPE (visible for this script engine only):
    sc~setAttribute("title", "--> -> >", sc~engine_scope)
       -- set attribute at global scope (visible for all script engines):
    sc~setAttribute("count", "1", sc~global_scope)

    /* ------------------------------------------------------------------------------ */
    /* This routine will be called from the Rexx code defined with the "onAction" event
       attribute; cf.    the JavaFX control with the id "idButton" in the fxml_02.fxml    */
    ::routine klickButton public
        use arg slotDir                       -- fetch the slotDir argument
        scriptContext=slotDir~scriptContext -- get the slotDir entry
       /* @get( idLabel count title ) */

        rexxInfo="Updated from public Rexx routine 'klickButton'."
        if count//2=0 then rexxInfo=rexxInfo~reverse   -- if even, reverse the current text
       /* @set( rexxInfo ) */   -- update the "rexxInfo" attribute, will auto update label
```

... continued on next page ...

Overview of JavaFX, 26
Example 2, "fxml_02_controller.rex"

... continued from previous page …

```
        /* show the currently defined attributes in the default ScriptContext's scopes   */
        say "getting all attributes from all ScriptContext's scopes..."
        do sc over .array~of(100, 200)
            say "ScriptContext scope:" pp(sc) pp(iif(sc=100,'ENGINE','GLOBAL')"_SCOPE")":"
            bin=scriptContext~getBindings(sc)
            if bin=.nil then iterate -- inexistent scope
            keys=bin~keySet          -- get kay values
            it=keys~makearray        -- get the keys as a Rexx array
            do key over it~sortWith(.CaselessComparator~new) -- sort keys caselessly
               val=bin~get(key)      -- fetch the key's value
               str=" " pp(key)~left(31,".") pp(val)
               if key="location" then str=str "~toString="pp(val~toString)
               say str
            end
            if sc=100 then say "-"~copies(86); else say "="~copies(86)
        end
          -- change the text of idLabel
        idLabel~setText(title .dateTime~new~string "(count #" count")")
        count+=1                 -- increase counter
        /* @set(count) */        -- save it in the ScriptContext bindings
        say
```

Overview of JavaFX, 27
Example 2, "fxml_02_controller.rex", Some Output

```
E:\fxml_02>rexx fxml_02.rex
REXXout>just arrived at [2017-11-02T19:47:35.611000]: parse source -> [WindowsNT SUBROUTINE
rexx_invoked_via_[fxml_02.fxml]_at_2017_11_02T18_47_35_584Z.rex]
REXXout> /// onAction eventHandler calling routine 'klickButton' \\\
REXXout>getting all attributes from all ScriptContext's scopes...
REXXout>ScriptContext scope: [100] [ENGINE_SCOPE]:
REXXout>  [event]..... [javafx.event.ActionEvent@10c0221]
REXXout>  [javax.script.engine]..... [Open Object Rexx (ooRexx)]
REXXout>  [javax.script.engine_version].. [100.20170923]
REXXout>  [javax.script.language]..... [ooRexx]
REXXout>  [javax.script.language_version] [REXX-ooRexx_5.0.0(MT)_32-bit 6.05 19 Oct 2017]
REXXout>  [javax.script.name]..... [Rexx]
REXXout>  [title]..... [--> -> >]
REXXout>-------------------------------------------------------------------------------
REXXout>ScriptContext scope: [200] [GLOBAL_SCOPE]:
REXXout>  [count]..... [1]
REXXout>  [idButton]..... [javafx.scene.control.Button@1c62fae]
REXXout>  [idLabel]..... [javafx.scene.control.Label@12e9675]
REXXout>  [idLabelRexxInfo]..... [javafx.scene.control.Label@15a1ca1]
REXXout>  [idLabelRexxStarted]..... [javafx.scene.control.Label@7683c9]
REXXout>  [idLabelYear]..... [javafx.scene.control.Label@137e560]
REXXout>  [idRoot]..... [javafx.scene.layout.AnchorPane@100fa1b]
REXXout>  [location]..... [java.net.URL@1a9d3d7] ~toString=[file:fxml_02.fxml]
REXXout>  [resources]..... [java.util.PropertyResourceBundle@14e2f70]
REXXout>  [rexxInfo]..... [Updated from public Rexx routine 'klickButton'.]
REXXout>  [rexxStarted]..... [Rexx started at: 2017-11-02T19:47:35.611000]
REXXout>===============================================================================
REXXout>
```

... continued on next page ...

Overview of JavaFX, 28
Example 2, "fxml_02_controller.rex", Some Output

```
... continued from previous page ...

REXXout> /// onAction eventHandler calling routine 'klickButton' \\\
REXXout>getting all attributes from all ScriptContext's scopes...
REXXout>ScriptContext scope: [100] [ENGINE_SCOPE]:
REXXout>  [event]..... [javafx.event.ActionEvent@117e598]
REXXout>  [javax.script.engine]..... [Open Object Rexx (ooRexx)]
REXXout>  [javax.script.engine_version].. [100.20170923]
REXXout>  [javax.script.language]..... [ooRexx]
REXXout>  [javax.script.language_version] [REXX-ooRexx_5.0.0(MT)_32-bit 6.05 19 Oct 2017]
REXXout>  [javax.script.name]..... [Rexx]
REXXout>  [title]..... [--> -> >]
REXXout>-------------------------------------------------------------------------------
REXXout>ScriptContext scope: [200] [GLOBAL_SCOPE]:
REXXout>  [count]..... [2]
REXXout>  [idButton]..... [javafx.scene.control.Button@1c62fae]
REXXout>  [idLabel]..... [javafx.scene.control.Label@12e9675]
REXXout>  [idLabelRexxInfo]..... [javafx.scene.control.Label@15a1ca1]
REXXout>  [idLabelRexxStarted]..... [javafx.scene.control.Label@7683c9]
REXXout>  [idLabelYear]..... [javafx.scene.control.Label@137e560]
REXXout>  [idRoot]..... [javafx.scene.layout.AnchorPane@100fa1b]
REXXout>  [location]..... [java.net.URL@1a9d3d7] ~toString=[file:fxml_02.fxml]
REXXout>  [resources]..... [java.util.PropertyResourceBundle@14e2f70]
REXXout>  [rexxInfo]..... [.'nottuBkcilk' enituor xxeR cilbup morf detadpU]
REXXout>  [rexxStarted]..... [Rexx started at: 2017-11-02T19:47:35.611000]
REXXout>===============================================================================
REXXout>
---------------------- after jrxApp~launch ------------------------
```

Overview of JavaFX, 29
Example 2, "fxml_02.rex"

```
   /* usage: fxml_02.rex [de] ... "de" will cause fxml_02_de.properties to be used */
parse arg locale .

   -- create Rexx object that will control the FXML set up with or without local
if locale<>"" then rxApp=.rexxApplication~new(locale)
              else rxApp=.rexxApplication~new

   -- instantiate the abstract JavaFX class, abstract "start" method implemented in Rexx
jrxApp=BsfCreateRexxProxy(rxApp,,"javafx.application.Application")
   -- launch the application, which will invoke the methdos "init" followed by "start"
jrxApp~launch(jrxApp~getClass, .nil)    -- need to use this version of launch in order to work
say center(" after jrxApp~launch ", 70, "-")

::requires "BSF.CLS"    -- get Java support

/* implements the abstract method "start" of javafx.application.Application */
::class RexxApplication

::method init   -- constructor to fetch the locale ("de": "fxml_01_de.properties")
  expose locale -- get direct access to attribute
  use strict arg locale="en"    -- if omitted use "fxml_01_en.properties"

   /* loads the FXML file (doing translations), sets up a scene for it and shows it */
::method start  -- implementation in Rexx
  expose locale -- get direct access to attribute
  use arg stage -- we get the stage to use for our UI

   -- create a file URL for fxml_02.fxml file (hence the protocol "file:")
  fxmlUrl=.bsf~new("java.net.URL", "file:fxml_02.fxml")
  jLocale=.bsf~new("java.util.Locale", locale)  -- get the desired Locale
  jRB=bsf.importClass("java.util.ResourceBundle")~getBundle("fxml_02", jLocale)
  rootNode=bsf.loadClass("javafx.fxml.FXMLLoader")~load(fxmlUrl, jRB)

  scene=.bsf~new("javafx.scene.Scene", rootNode)    -- create a scene from the tree
  stage~setScene(scene)      -- set our scene on stage
  stage~title="A Crazy FXML Rexx Application"   -- set the title for the stage
  img=.bsf~new("javafx.scene.image.Image", "oorexx_032.png")    -- create Image
  stage~getIcons~add(img)    -- use image as the application icon
  stage~show                 -- show the stage with the scene
```

Overview of JavaFX, 30
Example 2, "FXML_02_{de|en}.properties"

FXML_02_en.properties

```
! "fxml_02_en.properties"
! This is the English (en) translation for two terms.
!
! the following key is used in the idLabelYear: text="%year"
year = Year->

! the following key is used in the idButton: text="%clickMe"
clickMe = Click Me!
```

FXML_02_de.properties

```
! "fxml_02_de.properties"
! This is the German (de) translation for two terms.
!
! the following key is used in the idLabelYear: text="%year"
year = Jahr->

! the following key is used in the idButton: text="%clickMe"
clickMe = Drück mich!
```

Overview of JavaFX, 31
Example 2


```
rexx fxml_02.rex
rexx fxml_02.rex en
```


```
rexx fxml_02.rex de
```

An Address Book Application with JavaFX, 1 Example 3, Overview

- Cf. http://code.makery.ch/library/javafx-8-tutorial/
- Simple address book example
  - Data loaded from JSON file, if available
  - Data stored in JSON file
  - List persons
  - Allow for
    - Adding, deleting, changing persons
    - Create and show statistics about the months of birth
    - Print persons according to the current list order

An Address Book Application with JavaFX, 2 Example 3, Files

- Rendering, graphics: address_book_128.png, DarkTheme.css, DarkThemePrint.css
- Rexx-Utilities: json-rgf.cls, put_FXID_objects_into.my.app.rex
- Controlling the application
  - MainApp.rex
    - For each FXML file a Rexx class is defined to control it
- FXML-files defined with SceneBuilder
  - RootLayout.fxml, PersonOverview.fxml, BirthdayStatistics.fxml, PersonEditDialog.fxml, PersonPrinterDialog.fxml

An Address Book Application with JavaFX, 3 Example 3, Overview

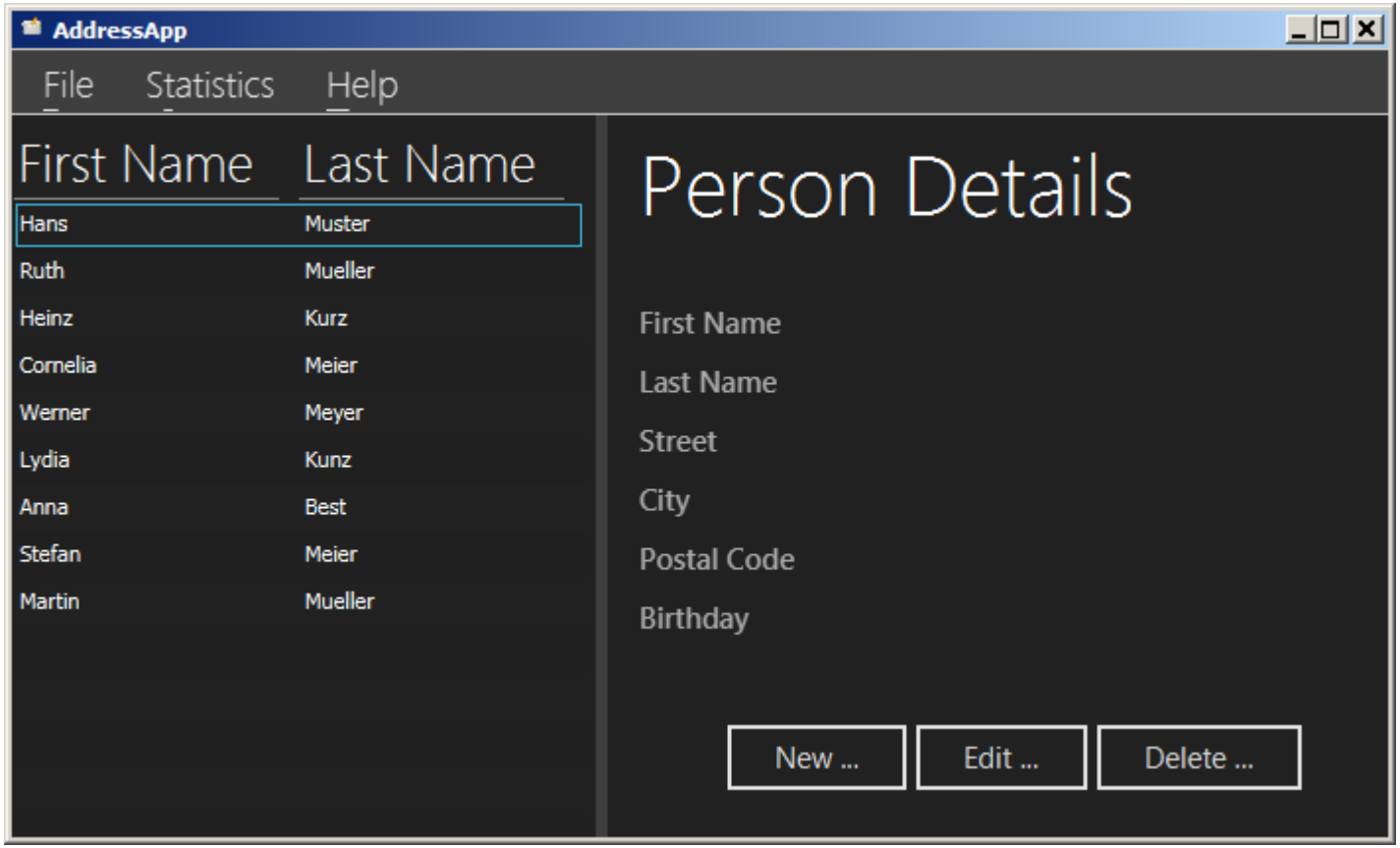
- Needs ooRexx 5.0 or higher (beta as of 2017-01-19)
- MainApp.rex
  - In addition creates an entry "MY.APP" in global .environment
  - The controller classes will be able to fetch the JavaFX objects to interact with from .MY.APP stored in a directory named after the FXML file
- put_FXID_objects_into.my.app.rex
  - Will be called at the end of each FXML file, after all JavaFX objects got defined
  - If there is no entry named MY.APP in the global Rexx .environment, then one will get created by that name referring to a newly created Rexx directory, such that it can be referred to by its environment symbol .MY.APP
  - Will store all JavaFX objects with an fx:id attribute in .MY.APP under the name of the FXML file name (location entry in global ScriptContext) for later retrieval

An Address Book Application with JavaFX, 4
Example 3, Sample JSON Content

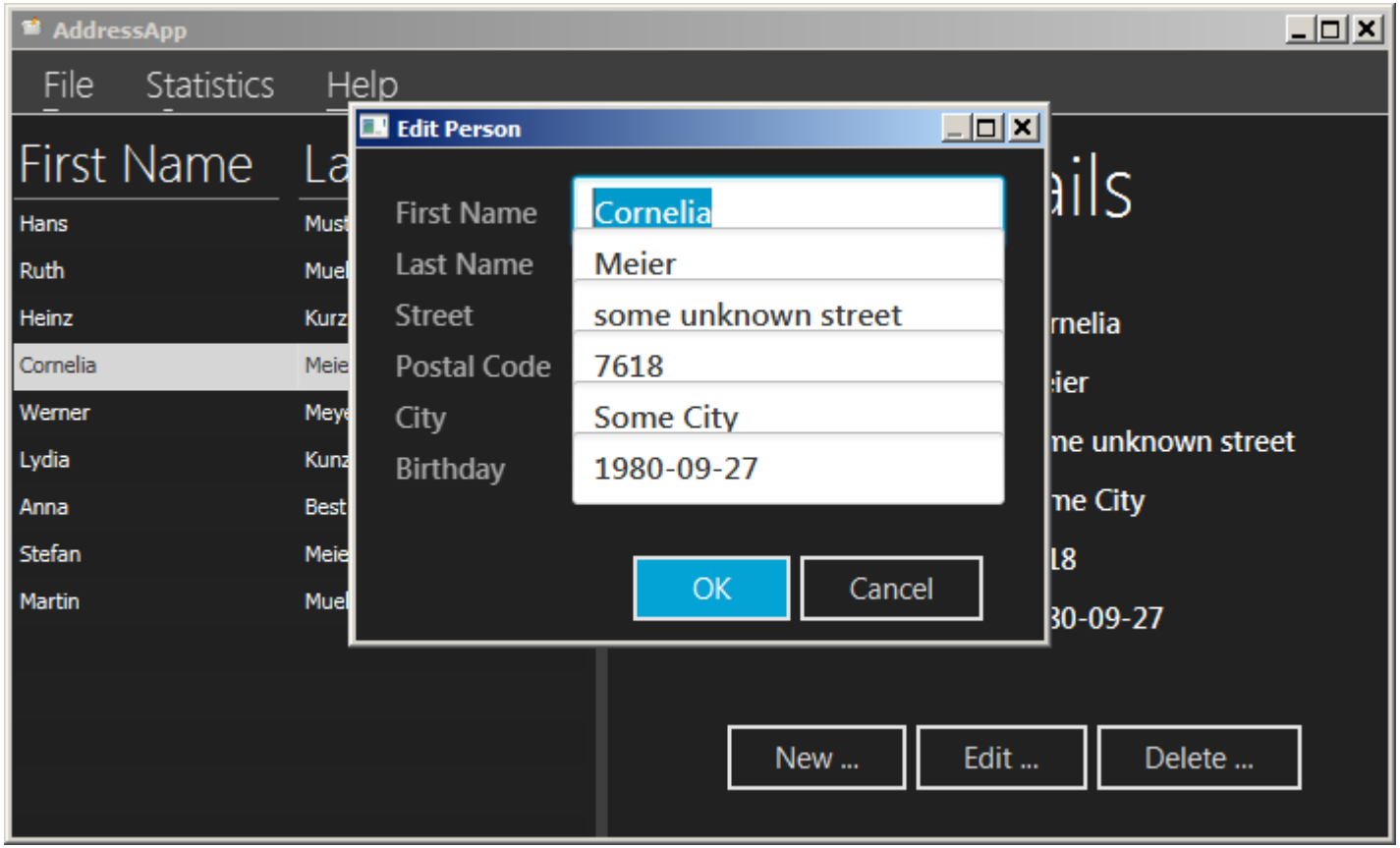
```
[
  {
    "birthday": "1979-03-11",
    "city": "Some City",
    "firstName": "Hans",
    "lastName": "Muster",
    "postalCode": 8985,
    "street": "some unknown street"
  }
  ,
  {
    "birthday": "2014-04-08",
    "city": "Some City",
    "firstName": "Ruth",
    "lastName": "Mueller",
    "postalCode": 9940,
    "street": "some unknown street"
  }
  ,

... cut ...
  ,
  {
    "birthday": "1978-05-20",
    "city": "Some City",
    "firstName": "Martin",
    "lastName": "Mueller",
    "postalCode": 4979,
    "street": "some unknown street"
  }
]
```

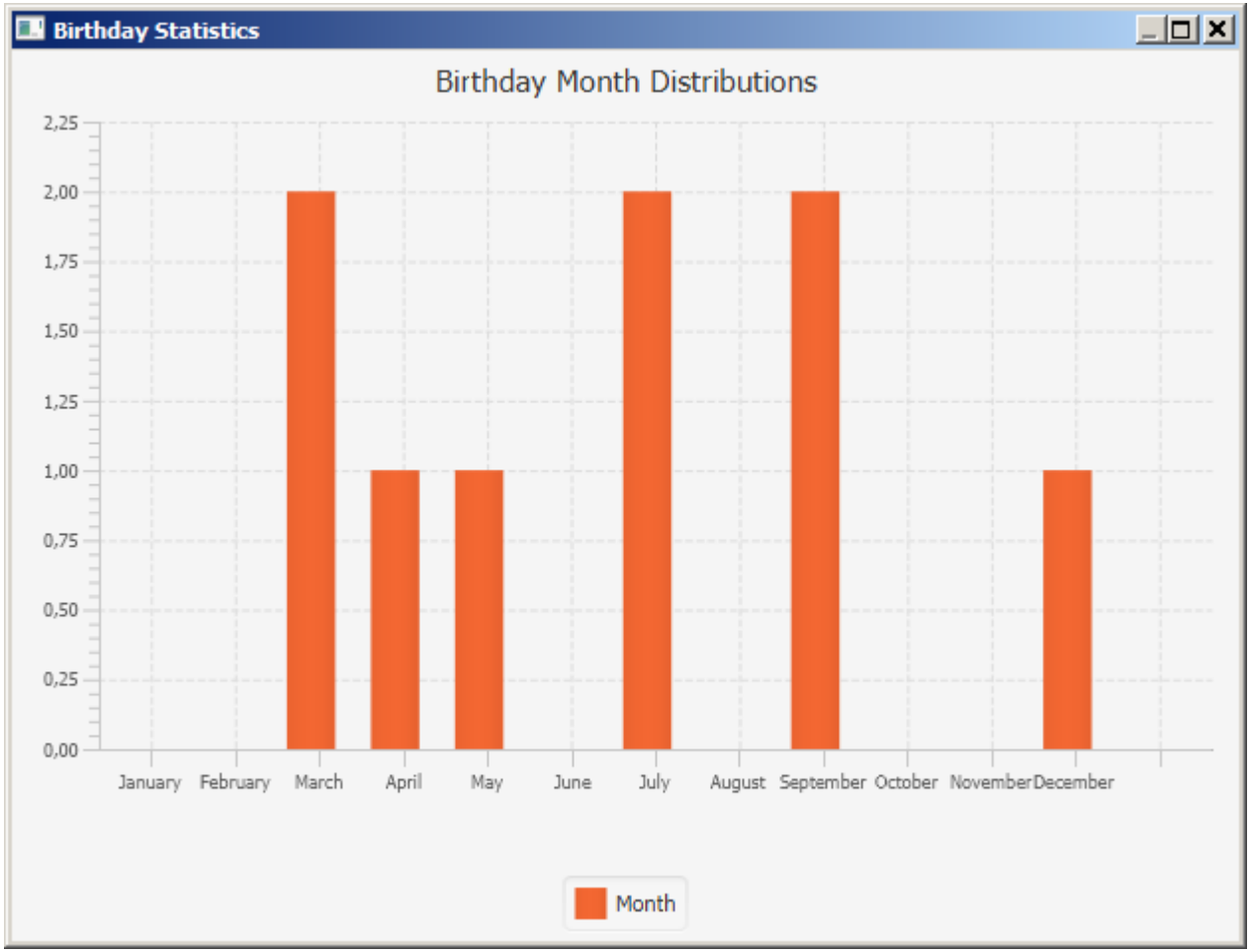
An Address Book Application with JavaFX, 5 Example 3, Screenshots 1/4



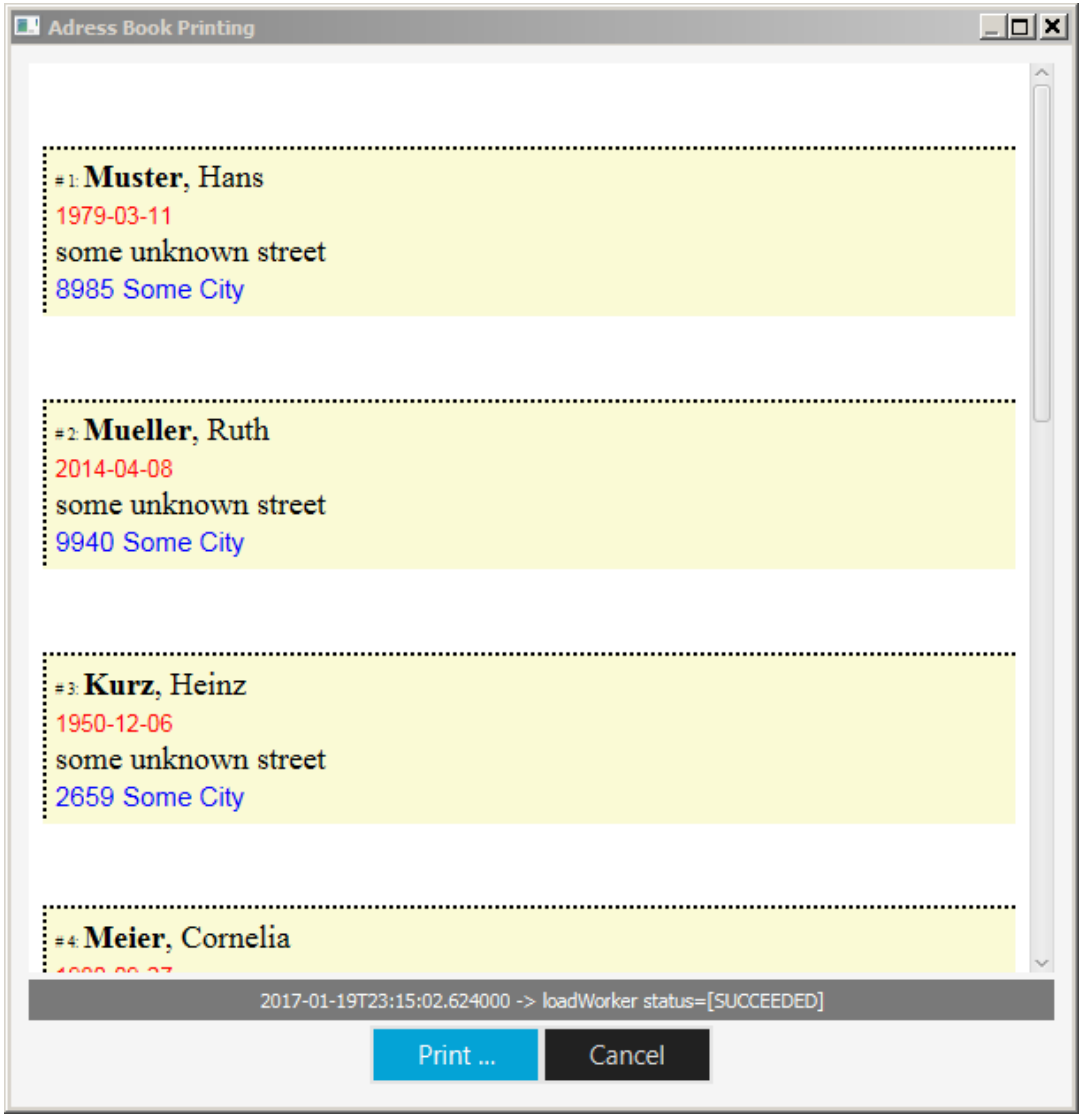
An Address Book Application with JavaFX, 5 Example 3, Screenshots 2/4



An Address Book Application with JavaFX, 5 Example 3, Screenshots 3/4



An Address Book Application with JavaFX, 5 Example 3, Screenshots 4/4



Roundup and Outlook

- Roundup
  - JavaFX
    - A great and extremely powerful GUI programming infrastructure
    - Allows meeting the most challenging GUI demands
    - SceneBuilder makes it easy to take full advantage of JavaFX
    - DOM and CSS (webkit)
  - BSF4ooRexx' javax.script support makes it very easy to use JavaFX from ooRexx!
    - Allows for powerful and portable (!) ooRexx applications
    - No excuse not to create great GUIs with ooRexx! :)