



Object Oriented Architectures and Secure Development

Resource Bundles

Matthias Blomme

Mattias De Wael

Frédéric Vlummens

A Classic FXML file

```
<VBox xmlns="http://javafx.com/javafx"
      xmlns:fx="http://javafx.com/fxml"
      fx:controller="fx.ScreenController"
      prefHeight="400.0" prefWidth="600.0">

  <Label text="The Absolutely Useless Screen"/>
  <Separator/>
  <Label text="Overview"/>
  <ListView fx:id="items"/>
  <Separator/>
  <Label text="Add:"/>
  <GridPane vgap="5" hgap="10">
    <Label text="Name" GridPane.rowIndex="0" GridPane.columnIndex="0"/>
    <TextField fx:id="name" GridPane.rowIndex="0" GridPane.columnIndex="1"/>
    <Label text="Age" GridPane.rowIndex="1" GridPane.columnIndex="0"/>
    <TextField fx:id="age" GridPane.rowIndex="1" GridPane.columnIndex="1"/>
    <Label text="Sex" GridPane.rowIndex="2" GridPane.columnIndex="0" />
    <ComboBox fx:id="sex" promptText="SELECT" GridPane.rowIndex="2" GridPane.columnIndex="1">
    </ComboBox>
  </GridPane>
  <Label fx:id="error" />
  <Button text="ADD" onAction="#add"/>
  <Separator/>

</VBox>
```

A Classic Controller

```
public class ScreenController {  
  
    @FXML  
    private TextField name;  
  
    @FXML  
    private TextField age;  
  
    @FXML  
    private ComboBox<Sex> sex;  
  
    @FXML  
    private ListView<Item> items;  
  
    @FXML  
    private Label error;  
  
    ...  
}
```

A Classic FX Application

```
public class App extends Application {  
  
    public static void main(String[] args) {  
        launch(args);  
    }  
  
    @Override  
    public void start(Stage primaryStage) throws Exception {  
        Parent root = FXMLLoader.load(App.class.getResource("/fxml/Screen.fxml"));  
  
        Scene scene = new Scene(root);  
  
        primaryStage.setScene(scene);  
        primaryStage.show();  
    }  
}
```

A Classic FX Application

The Absolutely Useless Screen

Overview

Alice (1, F)

Bob (2, M)

C3PO (3, X)

Add:

Name

David

Age

Sex

M

'elf' is not a valid age.

ADD

What are Resource Bundles and why use them?

- A resource bundle is special kind of configuration (.properties) file used for:
 - localization (i10n) and
 - internationalization (i18n).
 - *The adaptation of an application to meet the language and cultural requirements*
- Just a properties file, optionally suffixed with language and region (e.g., “nl_BE”)
 - Dutch language
 - BELGIUM region
- Benefits:
 - Write/prepare once, can be translated by non-IT specialist.
 - Easy to manage all text style related issues because all UI text is stored in one place.

An FX Application with Resource Bundle

```
public class App extends Application {  
  
    public static void main(String[] args) {  
        launch(args);  
    }  
  
    @Override  
    public void start(Stage primaryStage) throws Exception {  
        ResourceBundle bundle = ResourceBundle.getBundle("Screen");  
        Parent root = FXMLLoader.load(App.class.getResource("/fxml/Screen.fxml"), bundle);  
  
        Scene scene = new Scene(root);  
  
        primaryStage.setScene(scene);  
        primaryStage.show();  
    }  
}
```

A FXML File that Relies on a Resource Bundle

```
<VBox xmlns="http://javafx.com/javafx"
      xmlns:fx="http://javafx.com/fxml"
      fx:controller="fx.ScreenController"
      prefHeight="400.0" prefWidth="600.0">
```

```
<Label text="%screen.label.title"/>
```

```
<Separator/>
```

```
<Label text="%screen.label.overview"/>
```

```
<ListView fx:id="items"/>
```

```
<Separator/>
```

```
<Label text="%screen.label.add"/>
```

```
<GridPane vgap="5" hgap="10">
```

```
<Label text="%screen.label.name" GridPane.rowIndex="0" GridPane.columnIndex="0"/>
```

```
<TextField fx:id="name" GridPane.rowIndex="0" GridPane.columnIndex="1"/>
```

```
<Label text="%screen.label.age" GridPane.rowIndex="1" GridPane.columnIndex="0"/>
```

```
<TextField fx:id="age" GridPane.rowIndex="1" GridPane.columnIndex="1"/>
```

```
<Label text="%screen.label.sex" GridPane.rowIndex="2" GridPane.columnIndex="0"/>
```

```
<ComboBox fx:id="sex" promptText="%screen.combo.sex.prompt" GridPane.rowIndex="2" GridPane.columnIndex="1">
```

```
</ComboBox>
```

```
</GridPane>
```

```
<Label fx:id="error" />
```

```
<Button text="%screen.button.add" onAction="#add"/>
```

```
<Separator/>
```

```
</VBox>
```

<Label text="%screen.label.overview"/>

Anywhere you can write text for the user, you can use a **key name prefixed with %**.

This will fetch the correct words from the resource bundle.

A Resource Bundle

screen.label.title=**The Absolutely Useless Screen**

screen.label.overview=**Overview:**

screen.label.add=**Add:**

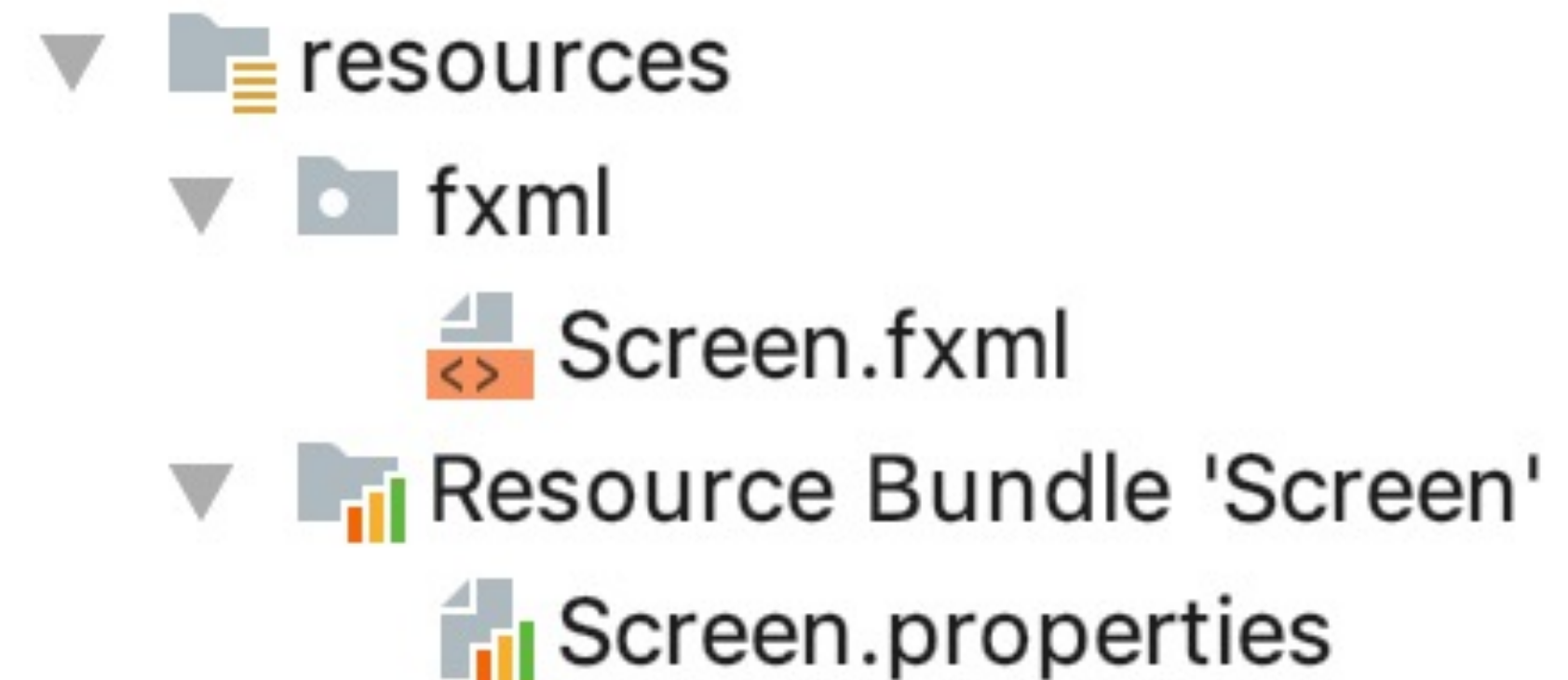
screen.button.add=**ADD**

screen.combo.sex.prompt=**SELECT**

screen.label.name=**Name:**

screen.label.age=**Age:**

screen.label.sex=**Sex:**



A resource bundle looks just like a regular properties file.

A Resource Bundle in Dutch

screen.label.title=**Het Meest Interessante Scherm Ooit**

screen.label.overview=**Elementen:**

screen.label.add=**Voeg item toe:**

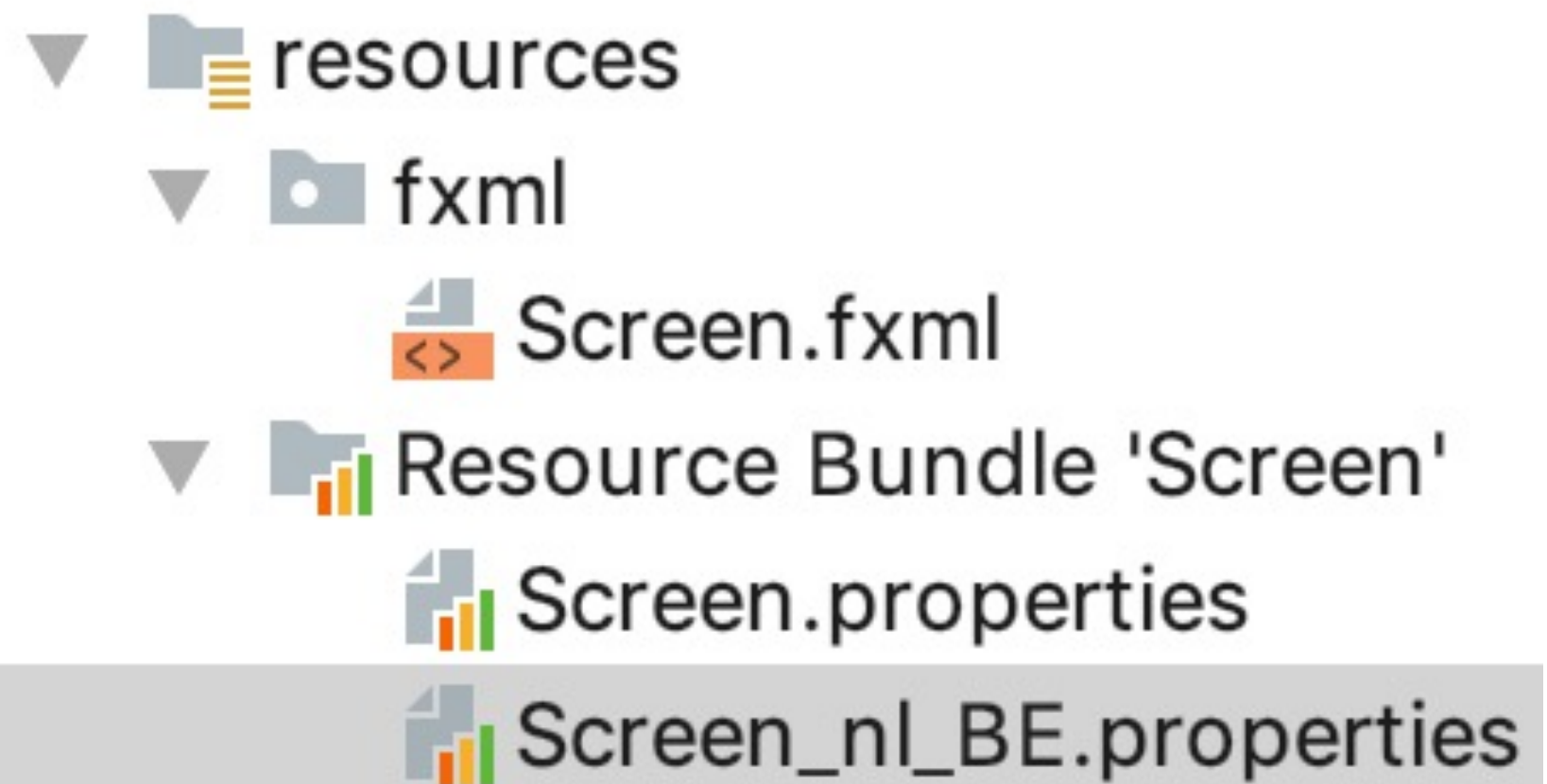
screen.button.add=**NIEUW**

screen.combo.sex.prompt=**Selecteer geslacht**

screen.label.name=**Naam:**

screen.label.age=**Leeftijd:**

screen.label.sex=**Geslacht:**



Locale

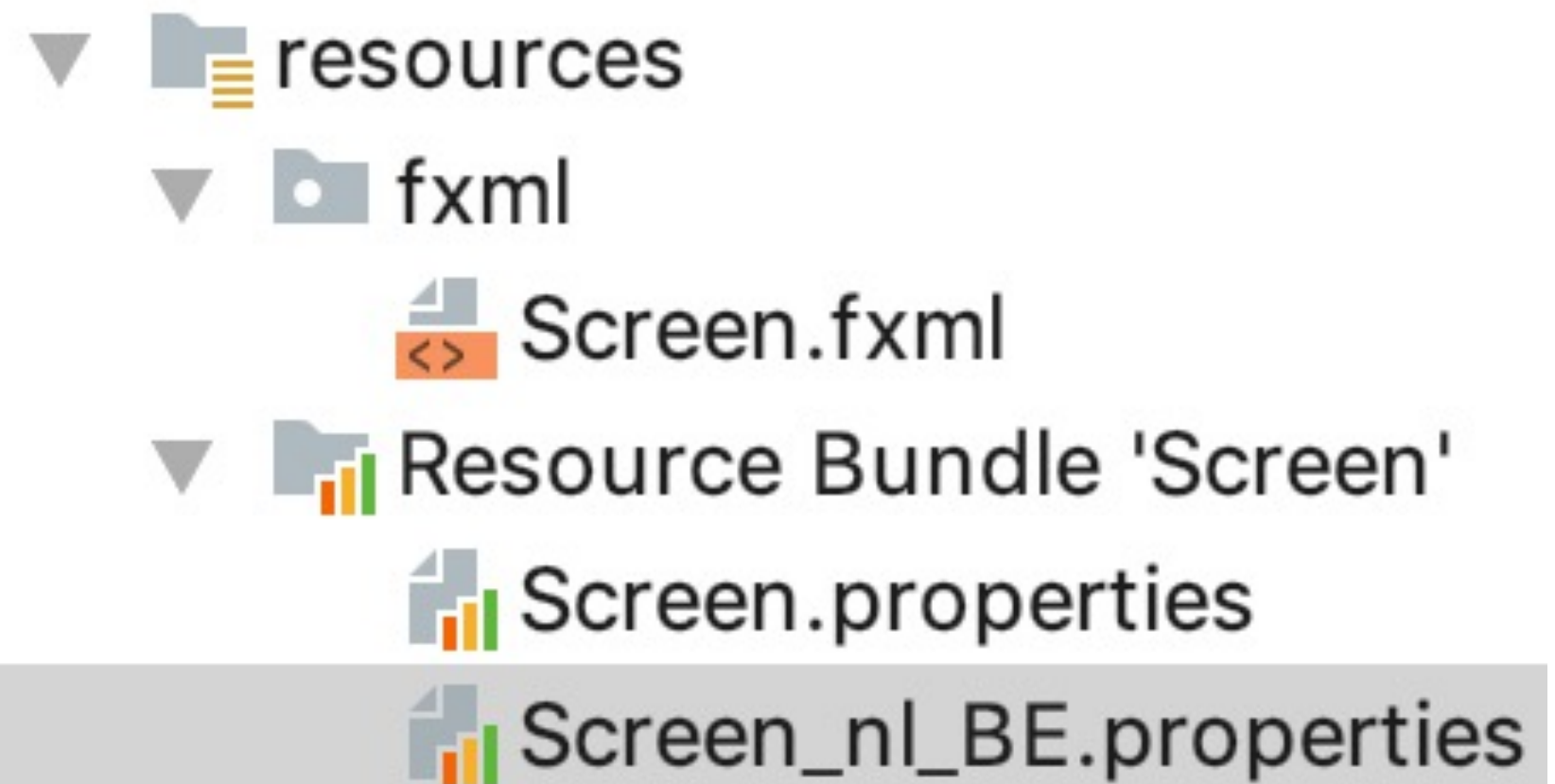
nl_BE

en_UK

Locale = language + region

You should store your resource bundles in the resources directory (preferably in a dedicated sub-directory).

You can have multiple resource bundles with the same name but with a different suffix (locale).



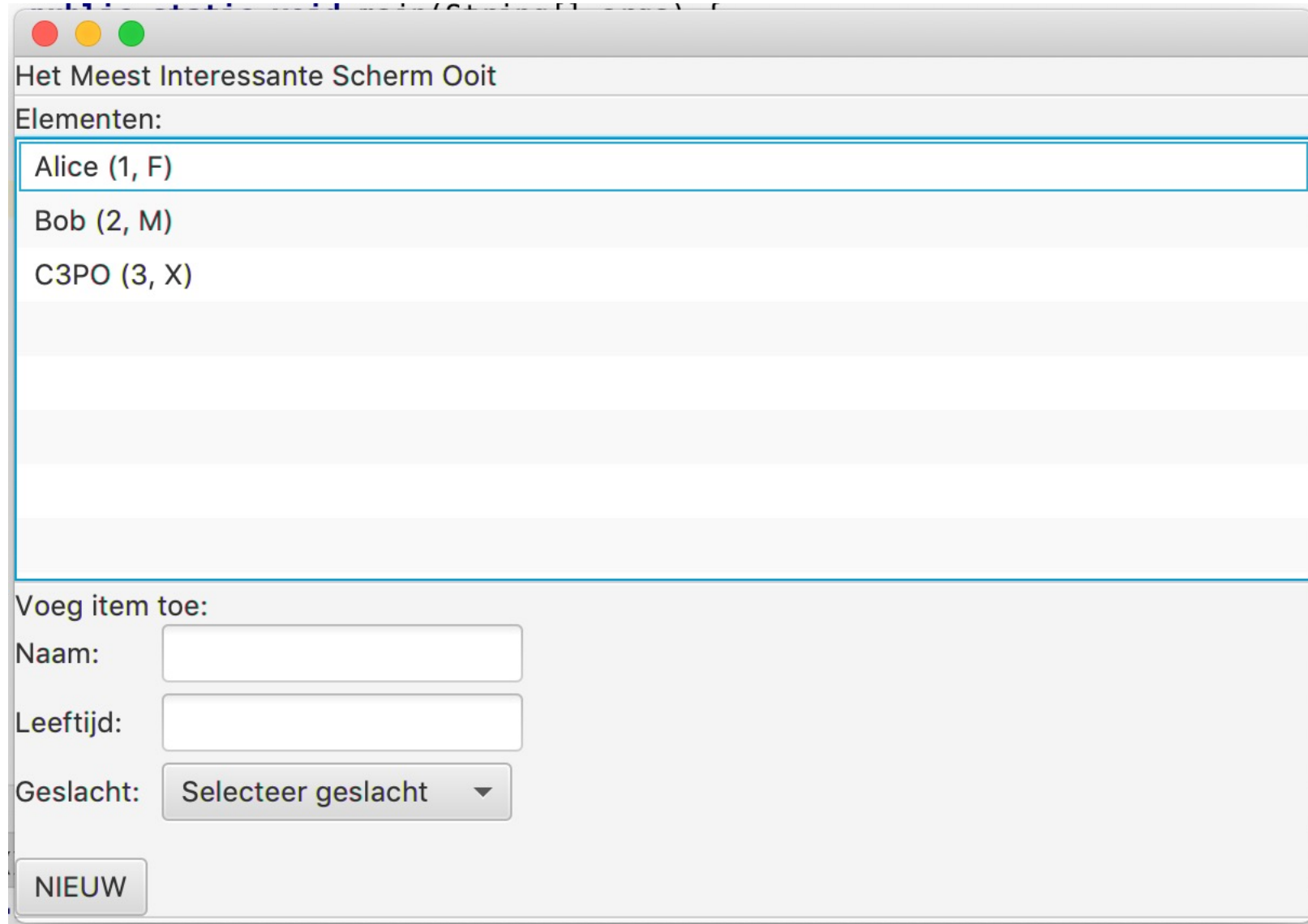
An FX Application with Specific Resource Bundle

```
public class App extends Application {  
  
    public static void main(String[] args) {  
        launch(args);  
    }  
  
    @Override  
    public void start(Stage primaryStage) throws Exception {  
        ResourceBundle bundle = ...;  
        Parent root = FXMLLoader.load(App.class.getResource("/fxml/Screen.fxml"), bundle);  
  
        Scene scene = new Scene(root);  
  
        primaryStage.setScene(scene);  
        primaryStage.show();  
    }  
}
```

You can load a resource bundle for a specific LOCALE:

```
ResourceBundle bundle = ResourceBundle.getBundle("Screen",  
    new Locale.Builder()  
        .setLanguage("nl")  
        .setRegion("BE")  
        .build()  
);
```

An FX Application with Specific Resource Bundle



Het Meest Interessante Scherm Ooit

Elementen:

Alice (1, F)
Bob (2, M)
C3PO (3, X)

Voeg item toe:

Naam:

Leeftijd:

Geslacht:

A Controller with a Resource Bundle

```
public class ScreenController {  
  
    @FXML  
    private TextField name;  
  
    @FXML  
    private Label error;  
  
    ....  
  
    @FXML  
    private ResourceBundle resources;  
  
    ...  
}
```

If you add a ResourceBundle field to your controller and annotate it with @FXML, it will be automatically injected by the FXML loader.

You can just use it like any other config file/ resource bundle in your code:

```
error.setText(resources.getString("screen.error.no.name"));
```