ABOUT ANDREAS POMAROLLI

Andreas has graduated from Computer Science and Bioinformatics at the University of Linz. During his studies he has been involved with a large number of research projects ranging from software engineering to data engineering and at least web engineering. His scientific focus includes the areas of software engineering, data engineering, web engineering and project management. He currently works as a software engineer in the IT sector where she is mainly involved with projects based on Java, Databases and Web Technologies.

JavaFX FXML Controller Example

Posted by: Andreas Pomarolli in FXML April 7th, 2016

This is a JavaFX FXML Controller Example. FXML is an XML-based language designed to build the user interface for JavaFX applications. You can use FXML to build an entire Scene or part of a

```
Scene
```

. FXML allows application developers to separate the logic for building the UI from the business logic. If the UI part of the application changes, you do not need to recompile the JavaFX code. Instead you can change the FXML using a text editor and rerun the application. You still use JavaFX to write business logic using the Java language. An FXML document is an XML document.

A JavaFX scene graph is a hierarchical structure of Java objects. XML format is well suited for storing information representing some kind of hierarchy. Therefore, using FXML to store the scene-graph is very intuitive. It is common to use FXML to build a scene graph in a JavaFX application.

The following table shows an overview of the whole article:

Table Of Contents

The following examples uses Java SE 7 and JavaFX 2.2.

1. Introduction to FXML

1.1 The FXML Code

FxFXMLExample1.fxml

NEWSLETTER

162682 insiders are alrea
weekly updates and complime
whitepapers!
Join them now to gai
access to the latest news in
as well as insights about Andr
Groovy and other related tech

Email address:

Your email address

Sign up

JOIN US

With **1,**
unique v
500 au
placed a
related s
Constant
lookout f
encourag
So If you
unique and interesting content th
check out our **JCG** partners prog
be a **guest writer** for Java Code
your writing skills!

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

<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<VBox fx:id="vbox" layoutX="10.0" layoutY="10.0" prefHeight="250.0" prefWidth="300.0" spacing="10"
xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/2.2">
  <style>
    -fx-padding: 10;
    -fx-border-style: solid inside;
    -fx-border-width: 2;
    -fx-border-insets: 5;
    -fx-border-radius: 5;
    -fx-border-color: blue;
  </style>
  <children>
    <Label fx:id="inputLbl" alignment="CENTER_LEFT" cache="true" cacheHint="SCALE" prefHeight="30.0"
prefWidth="200.0" text="Please insert Your Input here:" textAlignment="LEFT" />
    <TextField fx:id="inputText" prefWidth="100.0" />
    <Button fx:id="okBtn" alignment="CENTER_RIGHT" contentDisplay="CENTER" mnemonicParsing="false"
text="OK" textAlignment="CENTER" />
    <Label fx:id="outputLbl" alignment="CENTER_LEFT" cache="true" cacheHint="SCALE" prefHeight="30.0"
prefWidth="200.0" text="Your Input:" textAlignment="LEFT" />
    <TextArea fx:id="outputText" prefHeight="100.0" prefWidth="200.0" wrapText="true" />
  </children>
</VBox>
```

1.2 Adding UI Elements

The root element of the FXML document is the top-level object in the object-graph. The top-level object of the above example is a VBox. Therefore, the root element of your FXML would be:

```
<VBox>
</VBox>
```

How do you know that to represent a

```
VBox
```

in the object-graph, you need to use a tag in FXML? It is both difficult and easy. It is difficult because there is no documentation for FXML tags. It is easy because FXML has a few rules explaining what constitutes a tag name. For example, if a tag name is the simple or fullqualified name of a class, the tag will create an object of that class. The above element will create an object of the

```
VBox
```

class. The above FXML can be rewritten using the fully qualified class name:

```
<javafx.scene.layout.VBox>
</javafx.scene.layout.VBox>
```

In JavaFX, layout panes have children. In FXML, layout panes have children as their child elements. You can add a Label and a Button and other elements to the

```
VBox
```

as follows:

```
<children>
    <Label/>
    <TextField/>
    <Button/>
    <Label/>
    <TextArea/>
</children>
```

This defines the basic structure of the object-graph for our application. It will create a

```
VBox
```

with two labels, a TextField, a TextArea and a

```
Button
```

.

1.3 Importing Java Types in FXML

To use the simple names of Java classes in FXML, you must import the classes as you do in Java programs. There is one exception. In Java programs, you do not need to import classes from the

```
java.lang package
```

. However, in FXML, you need to import classes from all packages, including the

```
java.lang package
```

. An import processing instruction is used to import a class or all classes from a package. The following processing instructions import the

```
VBox
```

,

```
Label
```

, and

```
Button
```

classes:

```
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.Button?>
```

The following import processing instructions import all classes from the

```
javafx.scene.control
```

and

```
java.lang
```

packages:

```
<?import javafx.scene.control.*?>
<?import java.lang.*?>
```

1.4 Setting Properties in FXML

You can set properties for Java objects in FXML. A property for an object can be set in FXML if the property declaration follows the JavaBean conventions. The attribute name or the property element name is the same as the name of the property being set. The following FXML creates a

```
TextField
```

and sets its

```
prefWidth
```

property using an attribute:

```
<TextField fx:id="inputText" prefWidth="100.0" />
```

1.5 Specifying FXML Namespace

FXML does not have an XML schema. It uses a namespace that needs to be specified using the namespace prefix "fx". For the most part, the FXML parser will figure out the tag names such as tag names that are classes, properties of the classes, and so on. FXML uses special elements and attribute names, which must be qualified with the "fx" namespace prefix. Optionally, you can append the version of the FXML in the namespace URI. The FXML parser will verify that it can parse the specified.

The following FXML declares the "fx" namespace prefix.

```
<VBox xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/2.2">...</VBox>
```

1.6 Assigning an Identifier to an Object

An object created in FXML can be referred to somewhere else in the same document. It is common to get the reference of UI objects created in FXML inside the JavaFX code. You can achieve this by first identifying the objects in FXML with an

```
fx:id
```

attribute. The value of the

```
fx:id
```

attribute is the identifier for the object. If the object type has an id property, the value will be also set for the property. Note that each Node in JavaFX has an id property that can be used to refer to them in CSS. The following is an example of specifying the

```
fx:id
```

attribute for a

```
Label
```

.

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

1.7 The Corresponding Java Class

FxFXMLExample1.java

```
import java.io.FileInputStream;
import java.io.IOException;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class FxFXMLExample1 extends Application
{
```

```
    public static void main(String[] args)
    {
        Application.launch(args);
    }

    @Override
    public void start(Stage stage) throws IOException
    {
        // Create the FXMLLoader
        FXMLLoader loader = new FXMLLoader();
        // Path to the FXML File
        String fxmlDocPath = "Path-To-Your-FXML-Files/FxFXMLExample1.fxml";
        FileInputStream fxmlStream = new FileInputStream(fxmlDocPath);

        // Create the Pane and all Details
        VBox root = (VBox) loader.load(fxmlStream);

        // Create the Scene
        Scene scene = new Scene(root);
        // Set the Scene to the Stage
        stage.setScene(scene);
        // Set the Title to the Stage
        stage.setTitle("A simple FXML Example");
        // Display the Stage
        stage.show();
    }
}
```

An FXML document defines the view part (the GUI) of a JavaFX application. You need to load the FXML document to get the object-graph it represents. Loading an FXML is performed by an instance of the FXMLLoader class. The

```
FXMLLoader
```

class provides several constructors that let you specify the location, charset, resource bundle, and other elements to be used for loading the document.

```
FXMLLoader
```

supports loading a FXML document using an InputStream. The following snippet of code loads the same FXML document using an

```
InputStream
```

.

```
// Create the FXMLLoader
FXMLLoader loader = new FXMLLoader();
// Path to the FXML File
String fxmlDocPath = "Path-To-Your-FXML-Files/FxFXMLExample1.fxml";
FileInputStream fxmlStream = new FileInputStream(fxmlDocPath);
```

Internally, the

```
FXMLLoader
```

reads the document using streams, which may throw an IOException. All versions of the

```
load()
```

method in

```
FXMLLoader
```

class throw

```
IOException
```

. In your application, you will need to handle the exception. The

```
FXMLLoader
```

class contains several versions of the

```
load()
```

method. Some of them are instance methods and some static methods. You need to create an

```
FXMLLoader
```

instance and use the instance

```
load()
```

method, if you want to retrieve more information from the loader, such as the controller reference, resource bundle, the location, charset, and root object.

```
// Create the Pane and all Details
VBox root = (VBox) loader.load(fxmlStream);
```

What do you do next after loading an FXML document? The loader returns a

```
VBox
```

, which is set as the root for the

```
Scene
```

. The rest of the code is the same as you have been using except for one difference in the declaration of the

```
start()
```

method. The method declares that it may throw an

```
IOException
```

, which you had to add because you have called the

```
load()
```

method of the

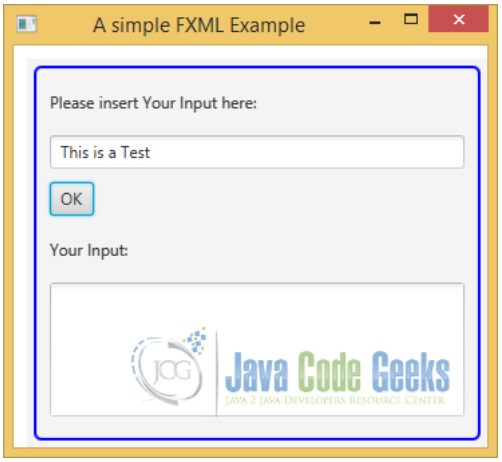
```
FXMLLoader
```

inside the method.

```
// Create the Scene
Scene scene = new Scene(root);
// Set the Scene to the Stage
stage.setScene(scene);
// Set the Title to the Stage
stage.setTitle("A simple FXML Example");
// Display the Stage
stage.show();
```

1.8 The GUI

The following mage shows the application after starting. But at this time, a click on the OK-Button has no effect. The reason for this behaviour is the fact, that we have not defined an EventHandler at this time.



A simple JavaFX FXML Example

2. Using Script Event Handlers

2.1 The FXML Code

FxFXMLExample2.fxml

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

<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<VBox fx:id="vbox" layoutX="10.0" layoutY="10.0" prefHeight="250.0" prefWidth="300.0" spacing="10" xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/2.2">
  <style>
    -fx-padding: 10;
    -fx-border-style: solid inside;
    -fx-border-width: 2;
    -fx-border-insets: 5;
    -fx-border-radius: 5;
    -fx-border-color: blue;
  </style>
  <children>
    <Label fx:id="inputLbl" alignment="CENTER_LEFT" cache="true" cacheHint="SCALE" prefHeight="30.0" prefWidth="200.0" text="Please insert Your Input here:" textAlignment="LEFT" />
    <TextField fx:id="inputText" prefWidth="100.0" />
    <Button fx:id="okBtn" alignment="CENTER_RIGHT" contentDisplay="CENTER" mnemonicParsing="false" onAction="printOutput();" text="OK" textAlignment="CENTER" />
    <Label fx:id="outputLbl" alignment="CENTER_LEFT" cache="true" cacheHint="SCALE" prefHeight="30.0" prefWidth="200.0" text="Your Input:" textAlignment="LEFT" />
    <TextArea fx:id="outputText" prefHeight="100.0" prefWidth="200.0" wrapText="true" />
    <fx:script>
        function printOutput()
        {
            outputText.setText(inputText.getText());
        }
    </fx:script>
  </children>
</VBox>
```

You can set event handlers for nodes in FXML. Setting an event handler is similar to setting any other properties. In FXML, you can specify two types of event handlers:

- Script Event Handlers
- Controller Event Handlers

In this chapter we will discuss Script Event Handlers. The Controller Event Handlers will be discussed in the following chapter.

The script event handler is used when the event handler is defined in a scripting language. The value of the attribute is the script itself, such as a function call or one or more statements. The following snippet of FXML sets the ActionEvent handler for a

```
Button
```

that calls the

```
printOutput()
```

function defined using JavaScript.

```
<?language JavaScript?>

<fx:script>
    function printOutput()
    {
        outputText.setText(inputText.getText());
    }
</fx:script>
```

If you want to execute the function

```
printOutput()
```

when the

```
Button
```

is clicked, you can set the event handler as:

```
<Button fx:id="okBtn" onAction="printOutput();" text="OK" textAlignment="CENTER" />
```

2.2 The Corresponding Java Class

FxFXMLExample2.java

```
import java.io.FileInputStream;
import java.io.IOException;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class FxFXMLExample2 extends Application
{
    public static void main(String[] args)
    {
        Application.launch(args);
    }

    @Override
    public void start(Stage stage) throws IOException
    {
        // Create the FXMLLoader
        FXMLLoader loader = new FXMLLoader();
        // Path to the FXML File
        String fxmlDocPath = "Path-To-Your-FXML-Files/FxFXMLExample2.fxml";
        FileInputStream fxmlStream = new FileInputStream(fxmlDocPath);

        // Create the Pane and all Details
        VBox root = (VBox) loader.load(fxmlStream);

        // Create the Scene
        Scene scene = new Scene(root);
        // Set the Scene to the Stage
        stage.setScene(scene);
        // Set the Title to the Stage
        stage.setTitle("A FXML Example with a Script Event Handler");
        // Display the Stage
        stage.show();

    }
}
```

2.3 The GUI

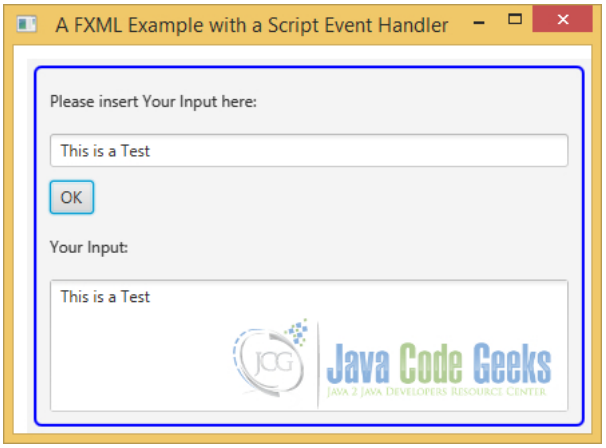
The following image shows the result of our program after inserting a Text in the

```
TextField
```

and pressing the

```
Button
```

"OK":



A JavaFX FXML Example with a JavaScript Event Handler

3. Using Controller Event Handlers

3.1 The FXML Code

FxFXMLExample3.fxml

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

<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<VBox fx:id="vbox" layoutX="10.0" layoutY="10.0" prefHeight="250.0" prefWidth="300.0" spacing="10" xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/2.2" fx:controller="FxFXMLController">
  <style>
    -fx-padding: 10;
    -fx-border-style: solid inside;
    -fx-border-width: 2;
    -fx-border-insets: 5;
    -fx-border-radius: 5;
    -fx-border-color: blue;
  </style>
  <children>
    <Label fx:id="inputLbl" alignment="CENTER_LEFT" cache="true" cacheHint="SCALE" prefHeight="30.0" prefWidth="200.0" text="Please insert Your Input here:" textAlignment="LEFT" />
    <TextField fx:id="inputText" prefWidth="100.0" />
    <Button fx:id="okBtn" alignment="CENTER_RIGHT" contentDisplay="CENTER" mnemonicParsing="false" onAction="#printOutput" text="OK" textAlignment="CENTER" />
    <Label fx:id="outputLbl" alignment="CENTER_LEFT" cache="true" cacheHint="SCALE" prefHeight="30.0" prefWidth="200.0" text="Your Input:" textAlignment="LEFT" />
    <TextArea fx:id="outputText" prefHeight="100.0" prefWidth="200.0" wrapText="true" />
  </children>
</VBox>
```

A controller is simply a class name whose object is created by FXML and used to initialize the UI elements. FXML lets you specify a controller on the root element using the

```
fx:controller
```

attribute. Note that only one controller is allowed per FXML document, and if specified, it must be specified on the root element. The following FXML specifies a controller for the

```
VBox
```

element.

```
<VBox fx:id="vbox" xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/2.2" fx:controller="FxFXMLController">
```

A controller needs to conform to some rules and it can be used for different reasons:

- The controller is instantiated by the FXML loader.
- The controller must have a public no-args constructor. If it does not exist, the FXML loader will not be able to instantiate it, which will throw an exception at the load time.
- The controller can have accessible methods, which can be specified as event handlers in FXML.
- The FXML loader will automatically look for accessible instance variables of the controller. If the name of an accessible instance variable matches the fx:id attribute of an element, the object reference from FXML is automatically copied into the controller instance variable. This feature makes the references of UI elements in FXML available to the controller. The controller can use them later, such as binding them to model.
- The controller can have an accessible initialize() method, which should take no arguments and have a return type of void. The FXML loader will call the initialize() method after the loading of the FXML document is complete.

3.2 The Controller Class

FxFXMLController.java

```
import java.net.URL;
import java.util.ResourceBundle;

import javafx.fxml.FXML;
```

```
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;

public class FxFXMLController
{
    @FXML
    // The reference of inputText will be injected by the FXML loader
    private TextField inputText;

    // The reference of outputText will be injected by the FXML loader
    @FXML
    private TextArea outputText;

    // location and resources will be automatically injected by the FXML loader
    @FXML
    private URL location;

    @FXML
    private ResourceBundle resources;

    // Add a public no-args constructor
    public FxFXMLController()
    {
    }

    @FXML
    private void initialize()
    {
    }

    @FXML
    private void printOutput()
    {
        outputText.setText(inputText.getText());
    }
}
```

The controller class uses a

`@FXML`

annotation on some members. The

`@FXML`

annotation can be used on fields and methods. It cannot be used on classes and constructors. By using a

`@FXML`

annotation on a member, you are declaring that the FXML loader can access the member even if it is private. A public member used by the FXML loader does not need to be annotated with

`@FXML`

. However, annotating a public member with

`@FXML`

is not an error. It is better to annotate all members, public and private, used by the FXML loader with the

`@FXML`

annotation. This tells the reader of your code how the members are being used.

The following FXML sets the

`printOutput()`

method of the controller class as the event handler for the

`Button`

:

```
<VBox fx:id="vbox" layoutX="10.0" layoutY="10.0" prefHeight="250.0" prefWidth="300.0" spacing="10"
xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/2.2"
fx:controller="FXFXML.FxFXMLController">

<Button fx:id="okBtn" alignment="CENTER_RIGHT" contentDisplay="CENTER" mnemonicParsing="false"
onAction="#printOutput" text="OK" textAlignment="CENTER" />
```

There are two special instance variables that can be declared in the controller and they are automatically injected by the FXML loader:

- @FXML private URL location;
- @FXML private ResourceBundle resources;

The location is the location of the FXML document. The resources is the reference of the ResourceBundle. When the event handler attribute value starts with a hash symbol (#), it indicates to the FXML loader that

`printOutput()`

is the method in the controller, not in a script.

The event handler method in the controller should conform to some rules:

- The method may take no arguments or a single argument. If it takes an argument, the argument type must be a type assignment compatible with the event it is supposed to handle.
- Conventionally, the method return type should be void, because there is no taker of the returned value.

- The method must be accessible to the FXML loader: make it public or annotate it with @FXML.
- When the FXML loader is done loading the FXML document, it calls the initialize() method of the controller. The method should not take any argument. It should be accessible to the FXML loader. In the controller, you used the @FXML annotation to make it accessible to the FXML loader.

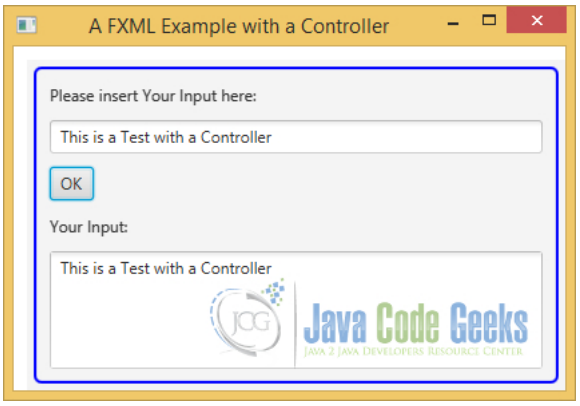
3.3 The Corresponding Java Class

FxFXMLExample3.java

```
01 import java.io.FileInputStream;
02 import java.io.IOException;
03
04 import javafx.application.Application;
05 import javafx.fxml.FXMLLoader;
06 import javafx.scene.Scene;
07 import javafx.scene.layout.VBox;
08 import javafx.stage.Stage;
09
10 public class FxFXMLExample3 extends Application
11 {
12     public static void main(String[] args)
13     {
14         Application.launch(args);
15     }
16
17     @Override
18     public void start(Stage stage) throws IOException
19     {
20         // Create the FXMLLoader
21         FXMLLoader loader = new FXMLLoader();
22         // Path to the FXML File
23         String fxmlDocPath = "Path-To-Your-FXML-Files/FxFXMLExample3.fxml";
24         FileInputStream fxmlStream = new FileInputStream(fxmlDocPath);
25
26         // Create the Pane and all Details
27         VBox root = (VBox) loader.load(fxmlStream);
28
29         // Create the Scene
30         Scene scene = new Scene(root);
31         // Set the Scene to the Stage
32         stage.setScene(scene);
33         // Set the Title to the Stage
34         stage.setTitle("A FXML Example with a Controller");
35         // Display the Stage
36         stage.show();
37
38     }
39 }
```

3.4 The GUI

The following image shows the result of our program:



A JavaFX FXML Controller Example

4. Download Java Source Code

This was an example JavaFX FXML Controller Example.

Download
You can download the full source code of this example here: **JavaFxFXMLExample.zip**

Tagged with: CONTROLLER JAVAFX