**Sources:**

**MS Bing Chat Copilot**

**+Contributors:**

Trieu Huu Phuong

**+Scenario 1:**

I have a file called TopNavigationBarController.java. The AnchorPane syntax inside it looks like this:

public class TopNavigationBarController{

@FXML

AnchorPane topNavigationBar\_anchorPane1;

}

Notice that I’ve already created this AnchorPane tool in FXML file and I thought that I can modify it in TopNavigationBarController.java file by writing some simple syntax like:

**topNavigationBar\_anchorPane1.getChildren().add(label1);**

In normal JavaFX program, the statement above may work well, but it doesn’t work in Eclipse+JavaFX+FXML combination program

So, How to modify the content of topNavigationBar\_anchorPane1?

A screenshot of a chat

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

import javafx.fxml.FXML;

import javafx.scene.control.Label;

import javafx.scene.layout.AnchorPane;

public class TopNavigationBarController {

@FXML

private AnchorPane topNavigationBar\_anchorPane1; // Automatically linked to your FXML file

public void initialize() {

// Add a Label to the AnchorPane

Label label = new Label("Welcome to the Top Navigation Bar!");

topNavigationBar\_anchorPane1.getChildren().add(label);

// Optionally, set its position within the AnchorPane

AnchorPane.setTopAnchor(label, 10.0);

AnchorPane.setLeftAnchor(label, 20.0);

}

}

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

public class TopNavigationBarController {

@FXML

private AnchorPane topNavigationBar\_anchorPane1;

public void initialize() {

// Set a style for the AnchorPane

topNavigationBar\_anchorPane1.setStyle("-fx-background-color: lightgray;");

// Add a dynamic label to the AnchorPane

Label label = new Label("Hello, Welcome!");

AnchorPane.setTopAnchor(label, 10.0); // Positioning within the AnchorPane

AnchorPane.setLeftAnchor(label, 10.0);

topNavigationBar\_anchorPane1.getChildren().add(label);

}

}

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Ex:

I have a small app like this:

A screenshot of a computer

AI-generated content may be incorrect.

Syntax in **Primary.fxml**:

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

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.image.Image?>

<?import javafx.scene.image.ImageView?>

<?import javafx.scene.layout.BorderPane?>

<BorderPane maxHeight=*"-Infinity"* maxWidth=*"-Infinity"* minHeight=*"-Infinity"* minWidth=*"-Infinity"* prefHeight=*"400.0"* prefWidth=*"400.0"* xmlns=*"http://javafx.com/javafx/23.0.1"* xmlns:fx=*"http://javafx.com/fxml/1"*

fx:controller=*"application.PrimaryController"*>

<left>

<Button fx:id=*"button1"* mnemonicParsing=*"false"* onAction=*"#button1Activate"* text=*"Button1"* BorderPane.alignment=*"CENTER"* />

</left>

<center>

<ImageView fx:id=*"imageView1"* fitHeight=*"150.0"* fitWidth=*"200.0"* pickOnBounds=*"true"* preserveRatio=*"true"* BorderPane.alignment=*"CENTER"*>

<image>

<Image url=*"@../../wallpaper2.jpg"* />

</image>

</ImageView>

</center>

<right>

<Button fx:id=*"button2"* mnemonicParsing=*"false"* onAction=*"#button2Activate"* text=*"Button2"* BorderPane.alignment=*"CENTER"* />

</right>

<top>

<Label fx:id=*"label1"* text=*"Label"* BorderPane.alignment=*"CENTER"* />

</top>

<bottom>

<Label fx:id=*"label2"* text=*"Label"* BorderPane.alignment=*"CENTER"* />

</bottom>

</BorderPane>

Syntax in **Main.java:**

package application;

import java.io.\*;

import javafx.application.Application;

import javafx.stage.Stage;

import javafx.scene.Scene;

import javafx.scene.control.Label;

import javafx.scene.control.Button;

import javafx.scene.layout.HBox;

import javafx.scene.layout.VBox;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.StackPane;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

public class Main extends Application {

private static Scene scene;

@Override

public void start(Stage stage) throws IOException {

scene = new Scene(loadFXML("primary"), 640, 480);

stage.setScene(scene);

stage.show();

}

static void setRoot(String fxml) throws IOException {

scene.setRoot(loadFXML(fxml));

}

private static Parent loadFXML(String fxml) throws IOException {

FXMLLoader fxmlLoader = new FXMLLoader(Main.class.getResource(fxml + ".fxml"));

return fxmlLoader.load();

}

public static void main(String[] args) {

launch();

}

}

Syntax in **PrimaryController.java:**

package application;

import java.io.IOException;

import javafx.fxml.FXML;

import javafx.scene.control.Label;

import javafx.scene.control.Button;

import javafx.scene.image.Image;

import javafx.scene.image.ImageView;

public class PrimaryController {

@FXML

Label label1;

@FXML

ImageView imageView1;

@FXML

Button button1;

@FXML

Button button2;

@FXML

Label label2;

@FXML

public void button1Activate() throws IOException {

label1.setText("Welcome");

}

@FXML

public void button2Activate() throws IOException {

label1.setText("Hello");

}

}

\*Now let’s add **initialize method** into **PrimaryController** class in **PrimaryController.java** file

**public** **class** **PrimaryController** {

**@FXML**

**Label** label1;

**@FXML**

**ImageView** imageView1;

**@FXML**

**Button** button1;

**@FXML**

**Button** button2;

**@FXML**

**Label** label2;

**public** **void** **initialize**() {

label1.**setText**(**"Welcome to this app"**);

}

**@FXML**

**public** **void** **button1Activate**() **throws** **IOException** {

label1.**setText**(**"Welcome"**);

}

**@FXML**

**public** **void** **button2Activate**() **throws** **IOException** {

label1.**setText**(**"Hello"**);

}

}

\*Let’s save and run the program

A screenshot of a computer

AI-generated content may be incorrect.

**+Scenario 2: Modify-add-remove event action methods**

According to **PrimaryController.java** file above, we can see 2 methods like

**@FXML**

**public** **void** **button1Activate**() **throws** **IOException** {

label1.**setText**(**"Welcome"**);

}

**@FXML**

**public** **void** **button2Activate**() **throws** **IOException** {

label1.**setText**(**"Hello"**);

}

The syntaxes above are pretty straightforward so I won’t explain them in this document, but the point to remember is that these methods exist because they attached to some tools in your program that was created by **FXML file,** and **throws IOException** lines are required to prevent any exception from occurring

**Let’s examine syntaxes in Primary.fxml file**

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

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.image.Image?>

<?import javafx.scene.image.ImageView?>

<?import javafx.scene.layout.BorderPane?>

<BorderPane maxHeight=*"-Infinity"* maxWidth=*"-Infinity"* minHeight=*"-Infinity"* minWidth=*"-Infinity"* prefHeight=*"400.0"* prefWidth=*"400.0"* xmlns=*"http://javafx.com/javafx/23.0.1"* xmlns:fx=*"http://javafx.com/fxml/1"*

fx:controller=*"application.PrimaryController"*>

<left>

<Button fx:id=*"button1"* mnemonicParsing=*"false"* onAction=*"#button1Activate"* text=*"Button1"* BorderPane.alignment=*"CENTER"* />

</left>

<center>

<ImageView fx:id=*"imageView1"* fitHeight=*"150.0"* fitWidth=*"200.0"* pickOnBounds=*"true"* preserveRatio=*"true"* BorderPane.alignment=*"CENTER"*>

<image>

<Image url=*"@../../wallpaper2.jpg"* />

</image>

</ImageView>

</center>

<right>

<Button fx:id=*"button2"* mnemonicParsing=*"false"* onAction=*"#button2Activate"* text=*"Button2"* BorderPane.alignment=*"CENTER"* />

</right>

<top>

<Label fx:id=*"label1"* text=*"Label"* BorderPane.alignment=*"CENTER"* />

</top>

<bottom>

<Label fx:id=*"label2"* text=*"Label"* BorderPane.alignment=*"CENTER"* />

</bottom>

</BorderPane>

Look at the two buttons’ syntaxes

<Button fx:id=*"button1"* mnemonicParsing=*"false"* onAction=*"#button1Activate"* text=*"Button1"* BorderPane.alignment=*"CENTER"* />

<Button fx:id=*"button2"* mnemonicParsing=*"false"* onAction=*"#button2Activate"* text=*"Button2"* BorderPane.alignment=*"CENTER"* />

Here, you can see that button1 attached the onAction event with method button1Activate based on onAction=*"#button1Activate"*, and button2 attached the onAction event with method button2Activate based on onAction=*"#button2Activate"*