



JavaFX and FXML

How to use FXML to define the components in a user interface.

FXML

FXML is an XML format text file that describes an interface for a JavaFX application.

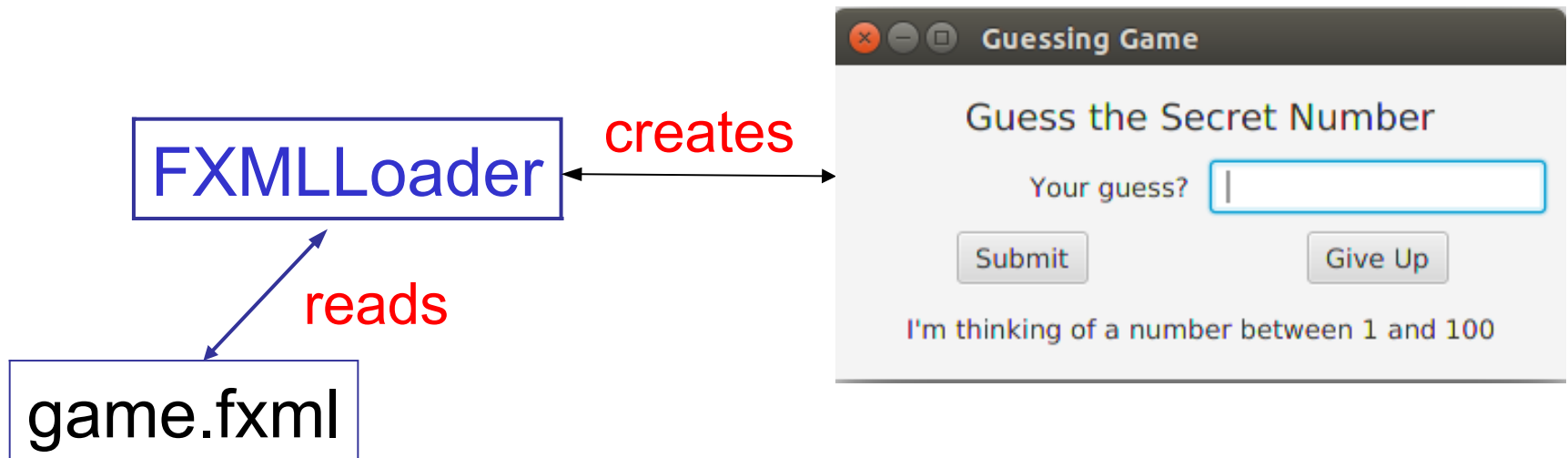
You can define **components**, **layouts**, **styles**, and **properties** in FXML instead of writing code.

```
<GridPane fx:id="root" hgap="10.0" vgap="5.0" xmlns="...">
  <children>
    <Label fx:id="topMessage"
            GridPane.halignment="CENTER"/>
    <TextField fx:id="inputField" width="80.0" />
    <Button fx:id="submitButton" onAction="#handleGuess" />
    <!-- more components -->
  </children>
</GridPane>
```

Creating a UI from FXML

The FXMLLoader class reads an FXML file and creates a **scene graph** for the UI (not the window or Stage).

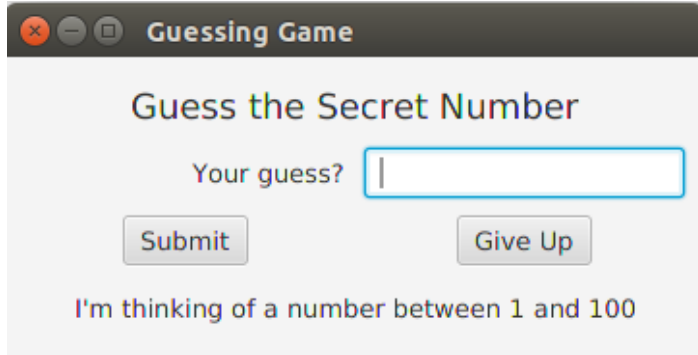
It creates objects for Buttons, Labels, Panes, etc. and performs layout according to the fxml file.



Code to Provide Behavior

The FXML scene define components, layouts, and property values, but **no behavior** or **event handlers**.

You write a Java class called a **Controller** to provide behavior, including event handlers:



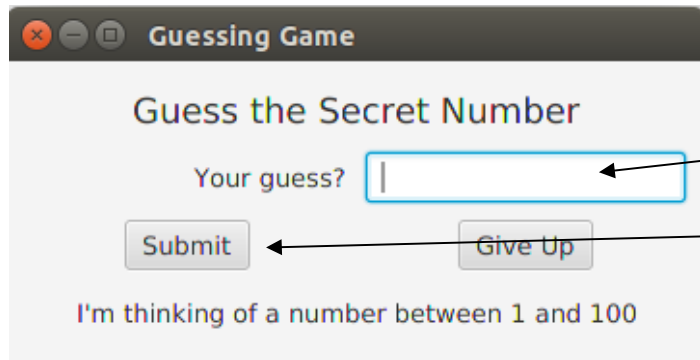
```
class GameController {  
    private TextField inputField;  
    private Button submitButton;  
  
    /** event handler */  
    void handleGuess(ActionEvent e)...
```

Connecting References to Objects

The FXML scene contains **objects** for Button, TextField, ...

The **Controller** contains **references** to the objects, and **methods** to supply behavior.

How to Connect **Objects** to **References**?



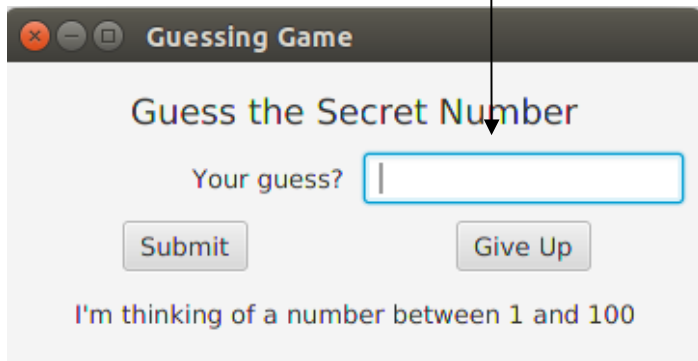
```
class GameController {  
    private TextField inputField;  
    private Button submitButton;  
  
    /** event handler */  
    void handleGuess(ActionEvent e)...
```

fx:id and @FXML

In the FXML file, you assign objects an "fx:id".

The fx:id is the **name** of a variable in the Controller class annotated with **@FXML**. You can annotate methods, too.

fx:id="inputField"



```
class GameController {  
    @FXML  
    private TextField inputField;  
    @FXML  
    private Button submitButton;  
  
    /** event handler */  
    @FXML  
    void handleGuess(ActionEvent e)
```

The fxml "code"

You can use SceneBuilder to create the fxml file. This example is just to illustrate the connection.

Name of controller class (can also set this in code)

```
<GridPane fx:id="root" hgap="10.0" vgap="5.0" xmlns="..."  
    fx:controller="game.GameController" >  
    <children>  
        <Label fx:id="topMessage" GridPane.halignment="CENTER"/>  
        <TextField fx:id="inputField" width="80.0" />  
        <Button fx:id="submitButton" onAction="#handleGuess" />  
        <!-- more components -->  
    </children>  
</GridPane>
```

Reference to variables and methods in Controller.

FXMLLoader creates scene from FXML

Tell **FXMLLoader** to load FXML and create a scene graph.
Instead of initComponents() use **FXMLLoader.load()**

```
public void start(Stage stage) throws IOException {  
    // Find the fxml file as part of our application  
    URL fxmlfile = getClass().getResource("game/GameUI.fxml");  
    // Create the scene graph  
    Parent root = FXMLLoader.load( fxmlfile );  
    Scene scene = new Scene(root);  
    stage.setScene( scene );  
    stage.setTitle("Guessing Game");  
    stage.show();  
}
```


What is getClass().getResource()?

`getClass().getResource(filename)` finds files that are included as **part of your application**.

Parameter is the relative path to the file.

You must use `getResource(filename)` because you do not know where on the user's computer your app is installed.

[YOUR APPLICATION] /

~~GameApp.class~~

game/

GameUI.fxml

GuessingGame.class

GameController.class

path is relative to your application

Example: image as resource

Suppose your app has some images in a subdirectory named `images/` relative to your source code root.

These images will be copied to the compiler output (`bin/`) and included in a JAR file.

`URL background =`

```
getClass().getResource("images/Backgnd.jpg");
```

```
src/  
  GameApp.java  
  images/  
    Backgnd.jpg  
    Player.png
```

Example: getting an InputStream

You can also create an InputStream for the resource.

```
InputStream in = getClass()  
    .getResourceAsStream("images/Player.png");  
ImageView image = new ImageView( in );
```

```
src/  
  GameApp.java  
  images/  
    Backgnd.jpg  
    Player.png
```

getResource() is a "short cut"

getClass().getResource() is a *convenience method*.

The work is actually done by the ClassLoader object:

```
this.getClass().getResource("dat/foo.csv");
```

is short cut for:

```
this.getClass().getClassLoader()  
    .getResource("dat/foo.csv");
```

```
src/  
    dat/  
        foo.csv
```

Better code for start()

Two problems may occur when creating a scene graph from an fxml file:

```
public void start(Stage stage) throws IOException {
```

(1) Returns null if the file is not found

```
URL fxmlfile = getClass().getResource("game/GameUI.fxml");
```

(2) May throw IOException

```
Parent root = FXMLLoader.load( fxmlfile );
```

FXMLLoader as Instance

Sometimes you want a [reference](#) to the FXMLLoader.

Using the loader, you can get/set the Controller class or set properties. To get an instance of FXMLLoader:

```
// Find the fxml file as part of our application
```

```
URL fxmlfile = getClass().getResource("game/GameUI.fxml");
```

```
// Create the scene graph
```

```
FXMLLoader loader = new FXMLLoader( fxmlfile );
```

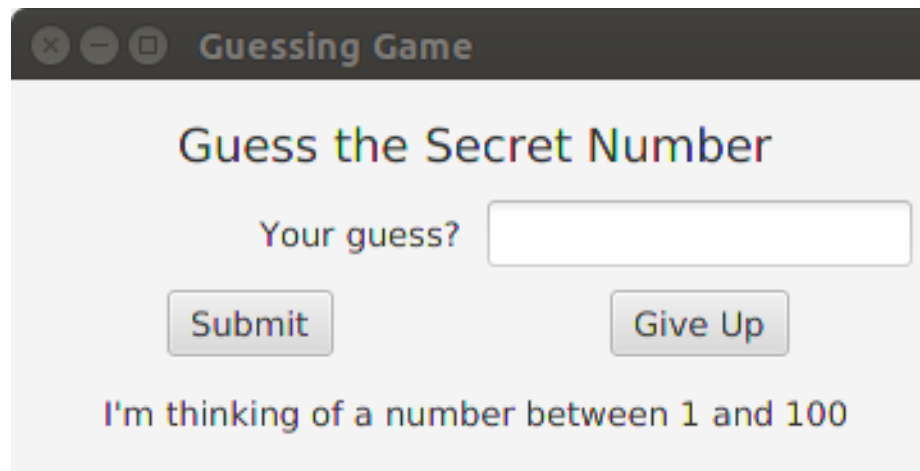
```
Parent root = loader.load();
```

Example Using SceneBuilder

Create a GUI for the GuessingGame using FXML.

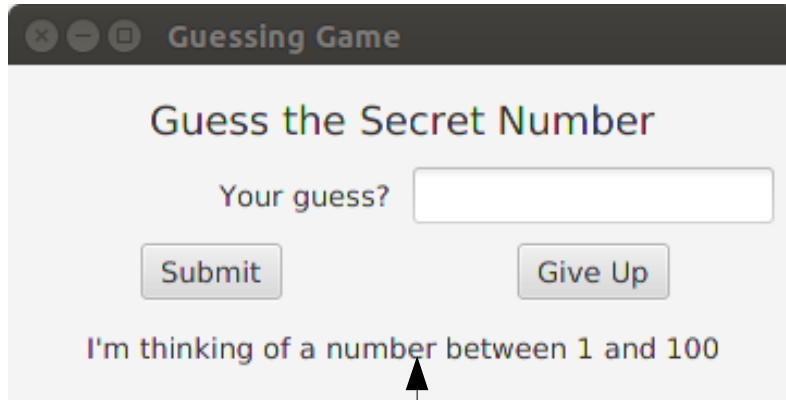
Using SceneBuilder for FXML and VS Code for Code.

Eclipse & IntelliJ can invoke SceneBuilder inside the IDE, but in VS Code you use SceneBuilder separately.



GuessingGame Components

GameUI.fxml



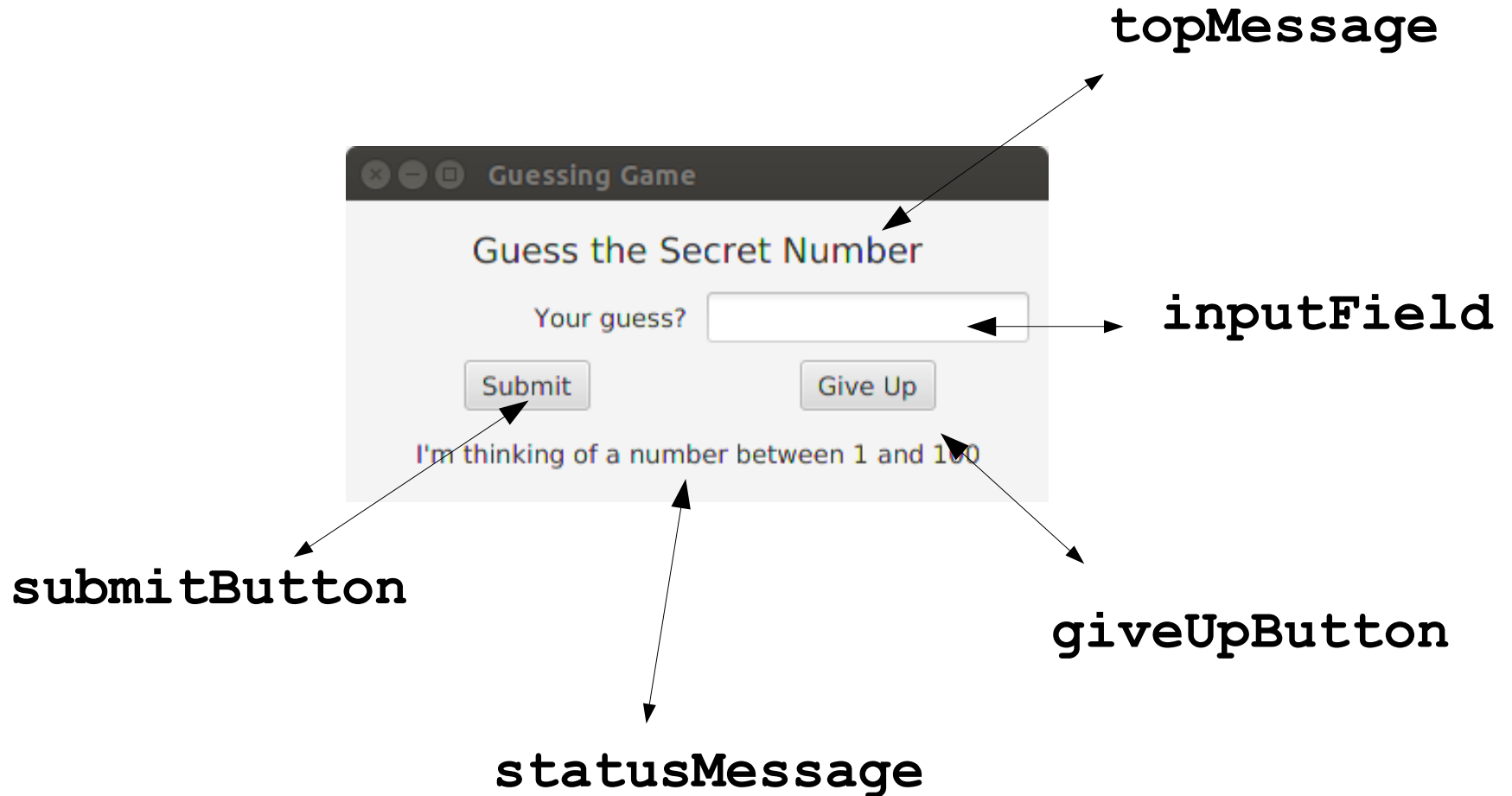
FXMLLoader.load()

GameApp
the "main" class
main() start()

GameController
inputField: TextField submitButton: Button giveUpButton: Button
handleGuess() initialize()

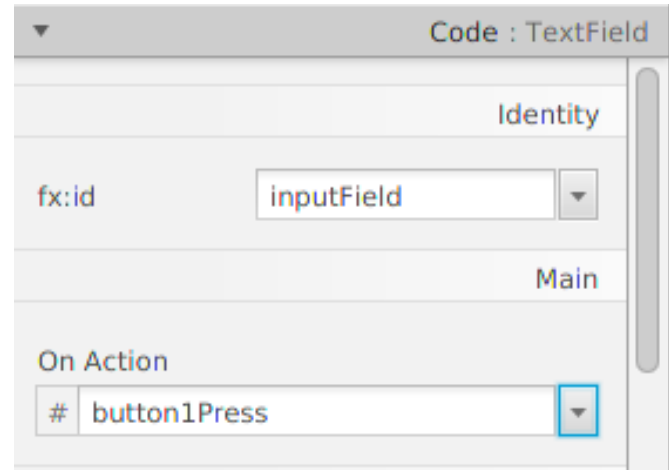
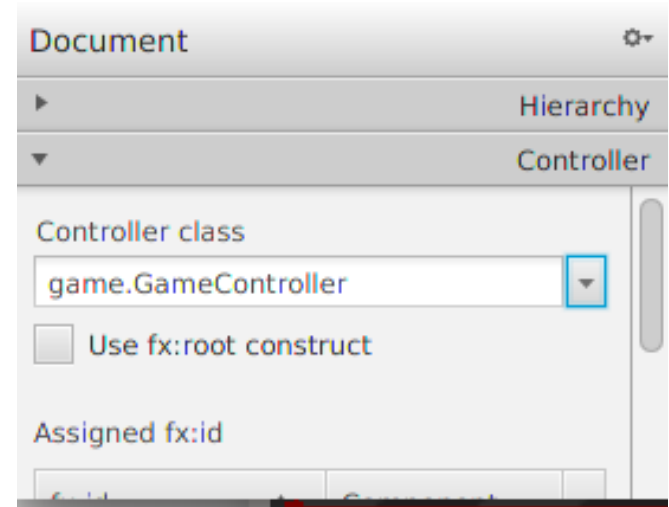
GuessingGame
secret count
guess(number) getMessage()

fx:id for components



SceneBuilder Important Parts

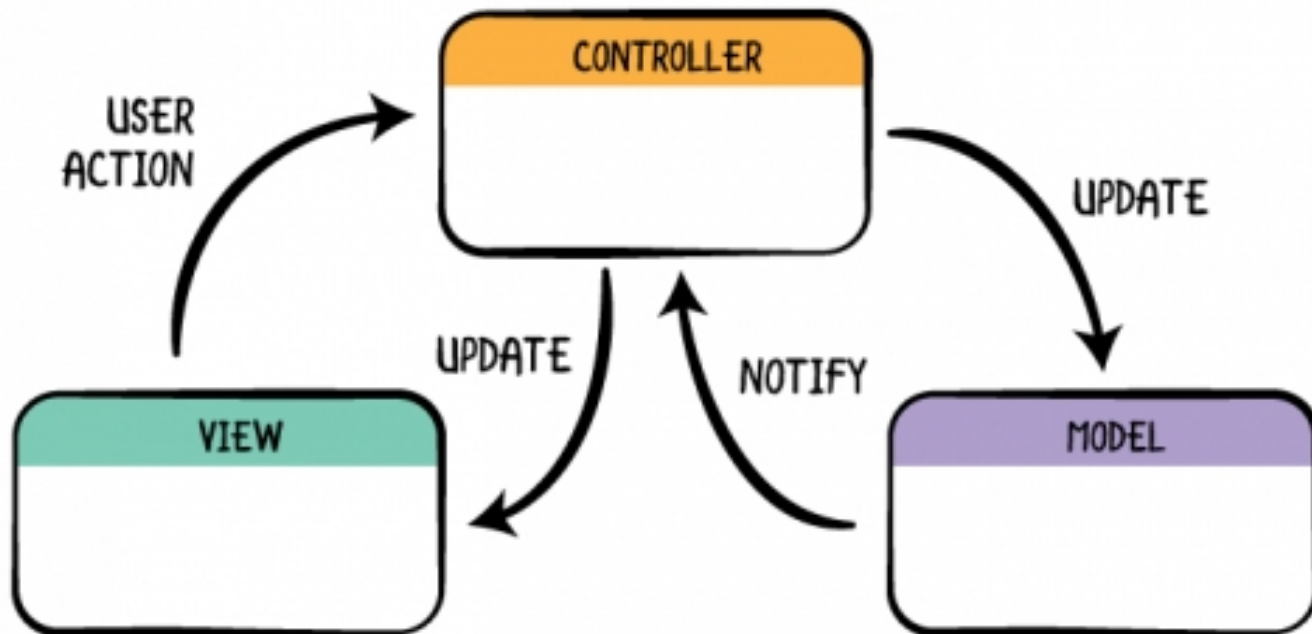
1. Specify the Controller class (left side, at bottom).
2. Assign fx:id to components:
 - select the component & open "Code" pane
 - select the fx:id from a list
3. Assign Event Handlers (or do it in code).



Model - View - Controller Design

Most Web and GUI applications use the **Model-View-Controller** design.

The Controller translates UI requests into requests the model understands and conveys results to the UI.



Source Code for Example

<https://github.com/jbrucker/guessing-game.git>

Some details are slightly different.