Maydm

# The Quizzler: Part 2

Maydm

## The Widget Attribute "ID"

**Check out the TextView and first Button in activity_main.xml**

```
<TextView
    android:id="@+id/textView"

<Button
    android:id="@+id/button"
```

Android has automatically added id attributes to these widgets. Notice that the second Button widget does not have an ID. That is because IDs must be uniquely named.

Let's edit the Button widgets such that each button has a unique ID.

Maydm

## Adding IDs to Widgets

**IDs help developers interact with widgets by offering a unique name to call it by.**

Let's name our buttons with unique id attributes. (Pro tip, show android:id attributes on the first line of the widget attributes so that they're always easy to spot. Edit the existing Button ID and add a new ID to the second button.)

```
<Button
   android:id="@+id/true_button"
… />
```

```
<Button
   android:id="@+id/false_button"
… />
```

Maydm

## One Detail about ID Attributes

**Did you notice that there is a "+" plus sign in the `android:id` attribute values but not in other values such as those for `android:text`?**

There is a + sign in the android:id attributes because we are creating the IDs. This syntax doesn't exist for the android:text value because we are referencing the string.

```
<Button
    android:id="@+id/true_button"
…
android:text="@string/false_button"
… />
```

Creating the ID in activity_main.xml

Referencing string in strings.xml

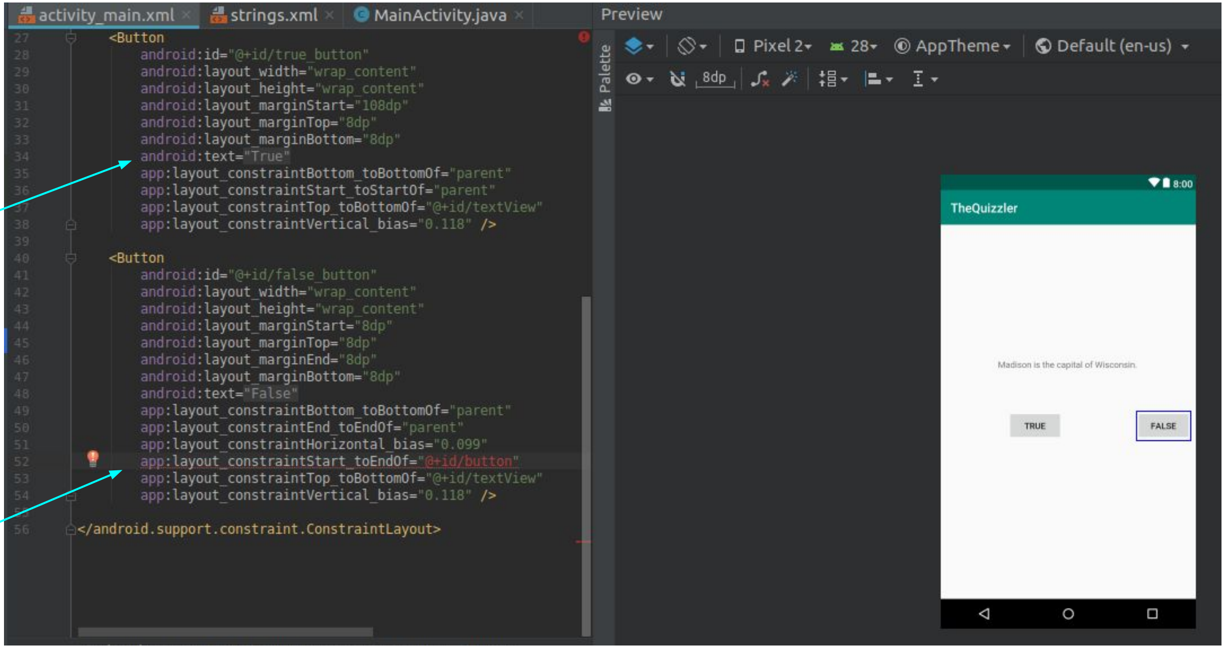
Maydm

# Potential Bug to Fix

You may have noticed that the FALSE button has moved since updating the Button ID names.

You may have also noticed that the `android:text` attribute has changed to "`true`" and "`false`". The string reference is still in place. Android displays the text that a user will see as a convenience for the developer.

Fix the undefined error by editing the ID name to "`true_button`". This will fix the placement of the FALSE button


activity_main.xml | strings.xml | MainActivity.java

```xml
    <Button
        android:id="@+id/true_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="108dp"
        android:layout_marginTop="8dp"
        android:layout_marginBottom="8dp"
        android:text="True"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView"
        app:layout_constraintVertical_bias="0.118" />

    <Button
        android:id="@+id/false_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:text="False"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.099"
        app:layout_constraintStart_toEndOf="@+id/button"
        app:layout_constraintTop_toBottomOf="@+id/textView"
        app:layout_constraintVertical_bias="0.118" />

</android.support.constraint.ConstraintLayout>
```

## Why do we create IDs?

We create IDs so that we can access the elements when required.

Next, we will edit MainActivity.java by adding two variables that will refer to the TRUE and FALSE buttons.

Type the new code by yourself without autocomplete assistance to gain more complete practice.

Maydm

# Edit MainActivity.java

Add two private Button variables to MainActivity.java after “`public class MainActivity extends AppCompatActivity {`”

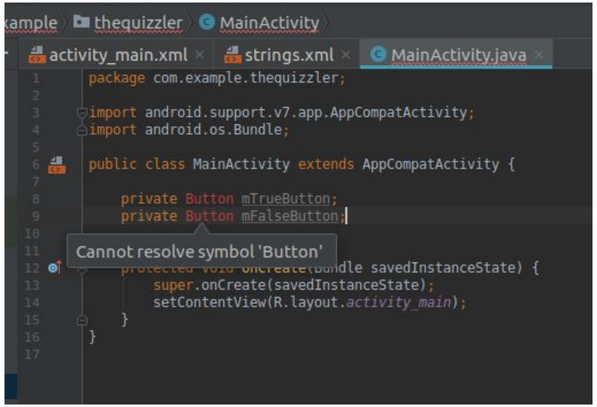
```
public class MainActivity extends AppCompatActivity {

    private Button mTrueButton;
    private Button mFalseButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Maydm

# Errors to Resolve

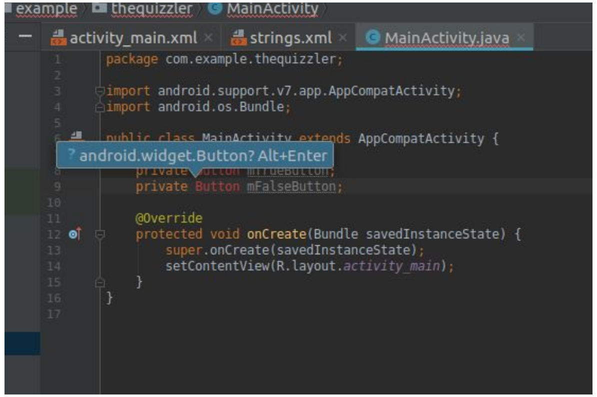
You may notice a prompt asking if “Button” is referring to a class called android.widget.Button. Also, if you hover over “Button” then you’ll likely see a popup error message that says “Cannot resolve symbol ‘Button’. This is an error message that’s directing you to import the Button class for widgets.

Let’s fix the error messages by adding a new import statement.



```
package com.example.thequizzler;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    private Button mTrueButton;
    private Button mFalseButton;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Cannot resolve symbol 'Button'



```
package com.example.thequizzler;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {
    private Button mTrueButton;
    private Button mFalseButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

? android.widget.Button? Alt+Enter

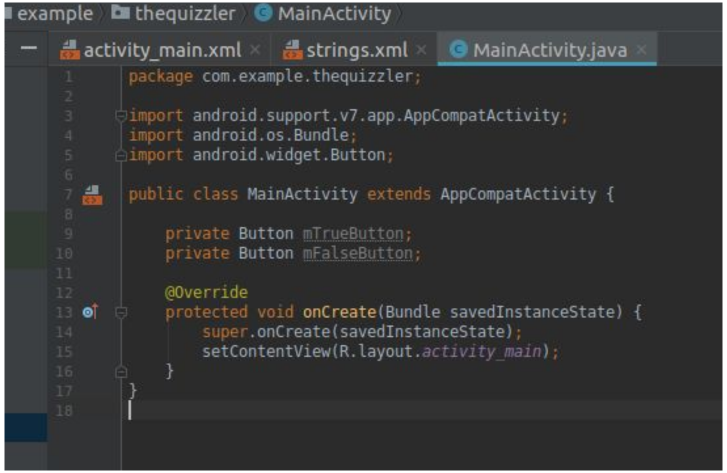
Maydm

# Errors to Resolve

Adding “`import android.widget.Button`” after the existing import statements will resolve the errors we discussed on the previous slide.

```
import android.os.Bundle;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {

    private Button mTrueButton;
    private Button mFalseButton;
```

If you see errors after adding the import statement then check for spelling errors and missing semicolons.



```
package com.example.thequizzler;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {

    private Button mTrueButton;
    private Button mFalseButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Maydm

## Wire Widgets

Now that we have imported the Button class for our widgets we will complete a two-part process known as “wiring widgets.”

1) Reference inflated View objects
   a) Inflating a View object basically means that the XML is rendered as a viewable object.
2) Establish event listeners on the View objects

## Reference the Button Widgets

We'll add two more methods to MainActivity.java.

```java
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mTrueButton = (Button) findViewById(R.id.true_button);
        mFalseButton = (Button) findViewById(R.id.false_button);
    }
}
```

findViewById() is an Activity method that allows us to reference an inflated widget. (A widget that has been rendered onto the UI from the XML.)

findViewById() accepts a widget's resource ID and returns a View object.

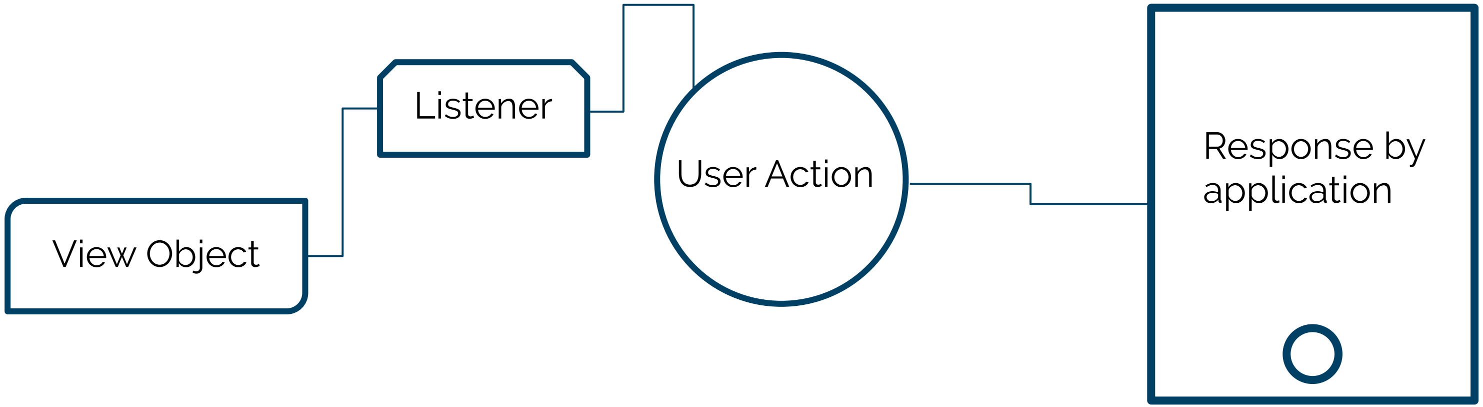
Maydm

## Establish Event Listeners

Android Applications are most often event driven.

That means an application loads View Objects on the UI and then waits, or listens, for the user to make an action.

Events include touching a button and swiping across the screen.



# Create an Event Listener for True Button Widget

Android provides prebuilt methods to assist in the creation and use of event listeners.

In this case, we want to list for a button press. The method that we'll use for this job is called `View.OnClickListener`.

Let's add the Click Listener to the True Button. This snippet will be added right after the mTrueButton variable.

```
mTrueButton = (Button) findViewById(R.id.true_button);
       mTrueButton.setOnClickListener(new View.OnClickListener() {
           @Override
           public void onClick(View v) {
               // Do something here soon...
           }
       });
```

```
       mFalseButton = (Button) findViewById(R.id.false_button);
   }
}
```

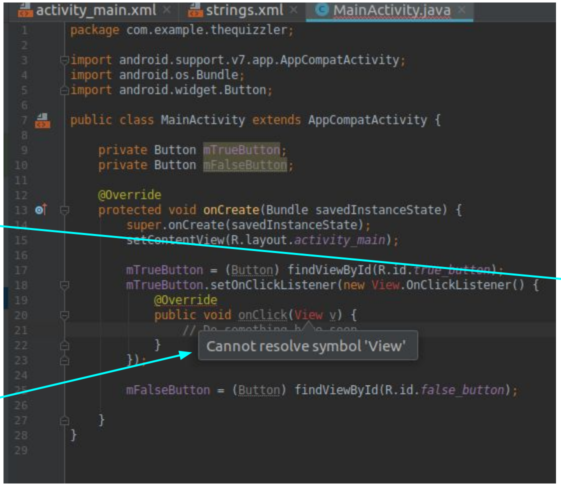
Maydm

# Import the View class

Did you notice that we called a new class called “View”?

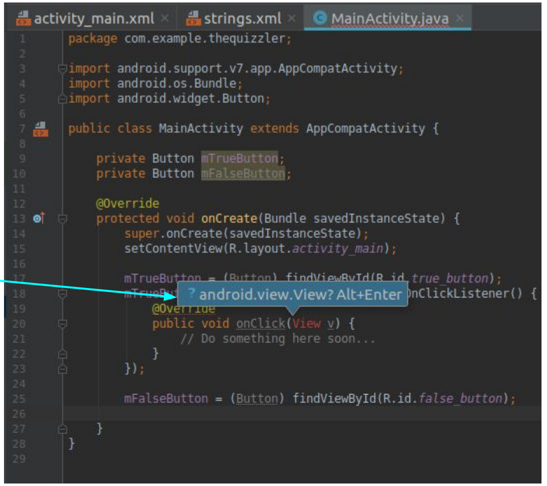
This is actually a class that we will need Android to import. Let’s add “import android.view.View”
Pay attention to the capitalization. Only the last “View” is capitalized.

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
```



```java
package com.example.thequizzler;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {

    private Button mTrueButton;
    private Button mFalseButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mTrueButton = (Button) findViewById(R.id.true_button);
        mTrueButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Do something h[illegible]e soon
            }
        });

        mFalseButton = (Button) findViewById(R.id.false_button);

    }
}
```

Cannot resolve symbol 'View'



```java
package com.example.thequizzler;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {

    private Button mTrueButton;
    private Button mFalseButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mTrueButton = (Button) findViewById(R.id.true_button);
        mTrueBu[illegible]nClickListener() {
            @Override
            public void onClick(View v) {
                // Do something here soon...
            }
        });

        mFalseButton = (Button) findViewById(R.id.false_button);

    }
}
```

? android.view.View? Alt+Enter

Helpful tip suggesting we need to import the View class

Error message informing that Android doesn’t understand View without an import

Maydm

# Create an Event Listener for False Button Widget

Did you guess that the next thing we'd do is add an event listener for the false button widget? If you did you're correct!

We'll add the False Button Event Listener immediately after the mFalseButton variable.

```
          @Override
           public void onClick(View v) {
               /* Do something here soon... */
           }
       });
```

```
       mFalseButton = (Button) findViewById(R.id.false_button);
       mFalseButton.setOnClickListener(new View.OnClickListener() {
           @Override
           public void onClick(View v) {
               // Do something here soon...
           }
       });
   }
}
```

## Let’s make a Toast!

Toast: A short message that informs the user of something but does not require user input or action.

## Toasts Provide Feedback

Toasts offer a level of interactivity for Android Applications.

Our toasts are going to tell the user if they chose a correct or incorrect response.

Creating a toast is a two step process. First, we will add the reference strings in strings.xml and then we will create the functionality in MainActivity.java

Maydm

## Toasts: Add String References in strings.xml

Add a correct and incorrect toast string reference after the false_button string reference we created earlier.

```
<resources>
   <string name="app_name">TheQuizzler</string>
   <string name="question_text">Madison is the capital of Wisconsin.</string>
   <string name="true_button">True</string>
   <string name="false_button">False</string>
   <string name="correct_toast">Correct!</string>
   <string name="incorrect_toast">Incorrect</string>
</resources>
```

Maydm

## Add new method with Toast class in MainActivity.java

Remember the “Do something here soon…” comments that we created in the Event Listeners? It’s time to do something now. The Event Listener will listen for a click and when the user makes an action it will respond with the correct or incorrect toast string.

Here’s the code for the Correct Button

```
        mTrueButton = (Button) findViewById(R.id.true_button);
        mTrueButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(MainActivity.this,
                                R.string.correct_toast,
                                Toast.LENGTH_SHORT).show();
            }
        });
```

# New method for False Button

```
        mFalseButton = (Button) findViewById(R.id.false_button);
        mFalseButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(MainActivity.this,
                                R.string.incorrect_toast,
                                Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

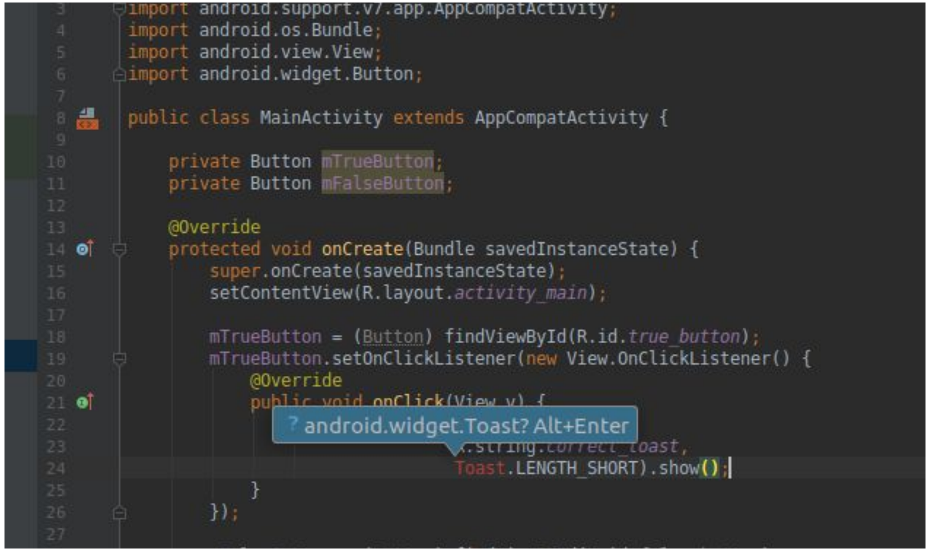
## Add an import statement for the Toast class

You may have noticed another helpful tip suggesting that we need a new import statement.

It will look like this:

```
import android.widget.Toast;
```

Either add the import statement manually or use the Alt + Enter shortcut for Android to create it automatically.



```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {

    private Button mTrueButton;
    private Button mFalseButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mTrueButton = (Button) findViewById(R.id.true_button);
        mTrueButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                          .string.correct_toast,
                          Toast.LENGTH_SHORT).show();
            }
        });
```

## Review

Your app should be error free and ready to emulate.
Before we run the emulator, however, let's review

- Widget attribute ID
- Wire Widgets
- Import android.widget.Button
- Event Listeners
- Import android.view.View
- Make Toasts with string references
- And methods in the event listeners
- Import android.widget.Toast

Maydm