

# MODULE 3

# VIEW

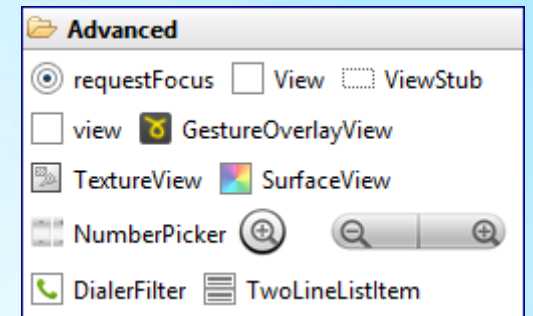
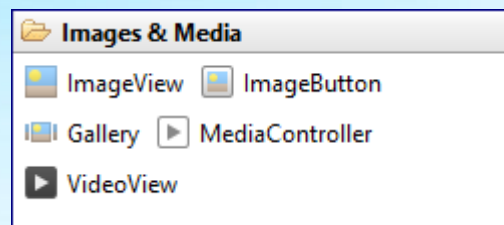
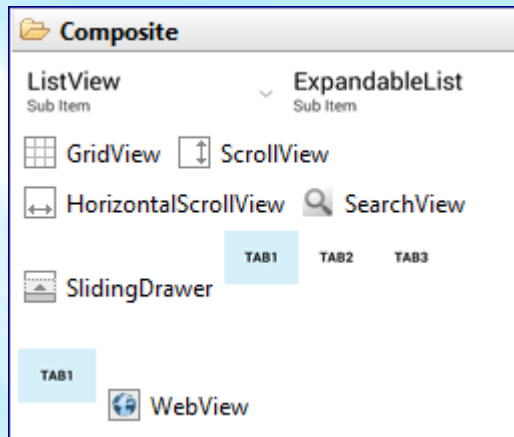
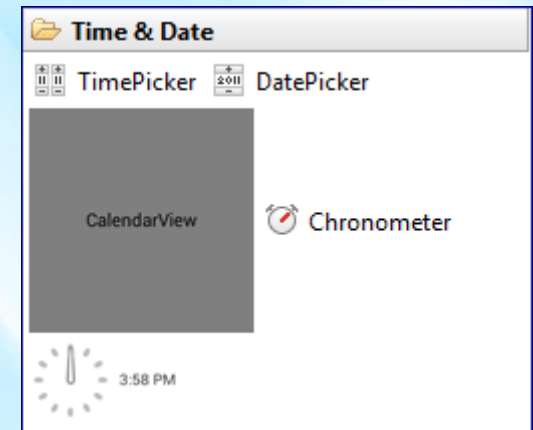
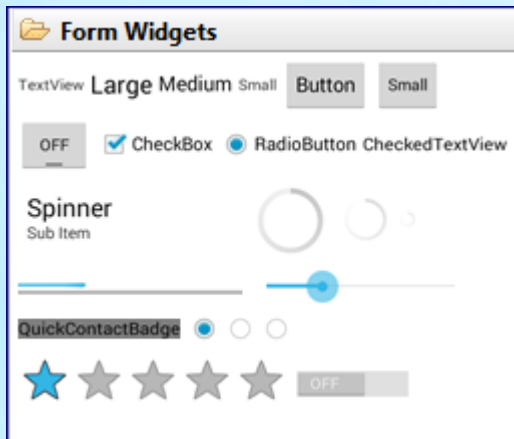
# TABLE OF CONTENT

- Android Views
- Assigning an ID to a View
- Using String Resource
- Event Handling
- Webography

# Android Views

- ▶ GUI components
  - Components on the screen
  - To interact with the user
- ▶ Widgets
  - Cfr C# control

# Android Views



# Assigning an ID to a View

- ▶ To access views from XML files into Java code
- ▶ Add an ID to a view

- Will be stored in **R.id**

- Through

```
android:id="@+id/identifierName"
```

- Where the (+) sign means that this is a new resource name that must be created and added to resources (in the R.java file)

- E.g,

```
<TextView  
    android:id="@+id/helloText"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/hello_world" />
```

# Assigning an ID to a View

- ▶ To use views into Java code
  - Through **findViewById** method
    - Argument: **R.id.identifierName**
    - Casting to do !
  - E.g,

```
public class MainActivity extends Activity {  
  
    private TextView mess;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        mess = (TextView) findViewById(R.id.helloText);  
        mess.setText("Welcome");  
    }  
}
```

# Using String Resource

- ▶ Do not hardcode String values!
- ▶ Better to define String resources in external XML file
  - Easier to maintain
  - E.g, for internationalization

# Using String Resource

## ► Add new resource entry in strings.xml file

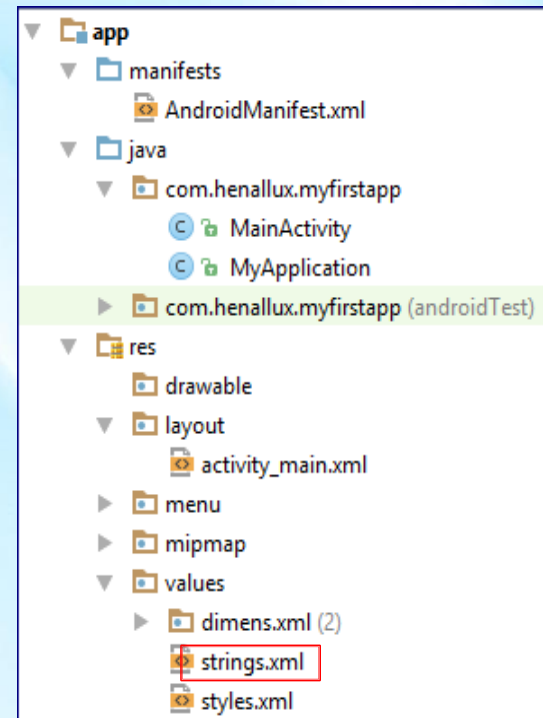
- Through **<string>** tag
  - Attribute **name** : identifies resource

## ► E.g,

```
<resources>
  <string name="app_name">MyFirstApp</string>
  <string name="action_settings">Settings</string>
  <string name="hello_world">Hello world</string>
</resources>
```

Resource identifier

String value





# String Resource

## ► Use String resource

### ◦ In XML files

- Through `@string/resourceIdentifier`

- E.g,

```
<TextView  
    android:id="@+id/helloText"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/hello_world" />
```

### ◦ In Java

- Through **R.string**

- E.g,

```
String message = getString(R.string.hello_world);
```

```
Toast.makeText(MainActivity.this, R.string.hello_world, Toast.LENGTH_LONG).show();
```

# Event Handling

- ▶ In the Activity Java file
- ▶ First, find the view through its Id
- ▶ Then, add listener to the view
  - And implement methods from the interface with the reaction to event

# Event Handling

► E.g,

```
import android.app.Activity;  
import android.os.Bundle;  
import android.view.Menu;  
import android.view.View;  
import android.view.View.OnClickListener;  
import android.widget.Button;
```

# Event Handling

► E.g,

```
public class MainActivity extends Activity {  
  
    private Button clickMeButton;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        clickMeButton = (Button) findViewById(R.id.clickMeButton);  
  
        clickMeButton.setOnClickListener(new OnClickListener() {  
            @Override  
            public void onClick(View arg0) {  
                Toast.makeText(MainActivity.this, "Welcome!", Toast.LENGTH_SHORT).show();  
            }  
        });  
    }  
}
```

→ Find the control

→ Add listener

↓  
Implement reaction to event

# Webography

- ▶ <http://developer.android.com/guide/topics/ui/controls.html>
- ▶ <http://developer.android.com/guide/topics/ui/ui-events.html>