

Unit Testing

Programming the Android Platform

Android Unit Testing

- Android-specific extensions to Junit
- MoreAsserts
 - Additional result checking classes
- ViewAsserts
 - Asserts about view layout
- TouchUtils
 - Classes for generating touch events
- Instrumentation
 - For monitoring application interaction with system

Android Unit Testing (cont.)

- Steps for testing Activity
 - Create an Android test project
 - Create one or more Android test classes
 - Add code to each Test class
- Test code usually includes test methods for
 - App initialization
 - UI behavior & class functionality
 - Application lifecycle state management

Android Unit Testing (cont.)

- **AndroidTestCase**
 - Test case with access to Activity Context
 - **ActivityUnitTestCase**
 - isolated testing of a single activity
- **InstrumentationTestCase**
 - Test case with access to system instrumentation & events
 - **ActivityInstrumentationTestCase2**
 - Functional testing of a single activity
- **Many other test case types**
 - developer.android.com/guide/topics/testing/testing_android.html

Android Unit Testing (cont.)

- Junit Test flow
- `setup()`
 - Put application in known state
- `run()`
 - Create additional data
 - Execute method under test
 - Check result with Assert
- `tearDown()`
 - Return to known state

HelloAndroidTest

```
import android.test.ActivityInstrumentationTestCase2;
import course.examples.HelloWorld.HelloWorld.HelloAndroid;
...

public class HelloAndroidTest extends
    ActivityInstrumentationTestCase2<HelloAndroid> {

    private HelloAndroid mActivity;
    private TextView mView;
    private String resourceString;

    public HelloAndroidTest() {
        super("course.examples.HelloWorld.HelloWorld",
            HelloAndroid.class);
    }
}
```

HelloAndroidTest (cont.)

```
protected void setUp() throws Exception {  
    super.setUp();  
    mActivity = this.getActivity();  
    mView = (TextView) mActivity.findViewById  
        (course.examples.HelloWorld.HelloWorld.R.id.textview);  
    resourceString = mActivity.getString  
        (course.examples.HelloWorld.HelloWorld.R.string.hello);  
}
```

HelloAndroidTest (cont.)

```
public void testPreconditions() {  
    assertNotNull(mView);  
}
```

```
public void testText() {  
    assertEquals(resourceString,(String)mView.getText());  
}
```


Android Unit Testing (cont.)

- developer.android.com/resources/tutorials/testing/activity_test.html

Layout inspector

The screenshot displays the Hierarchy Viewer in Android Studio. The window title is "Hierarchy Viewer". At the top, there are several buttons: "Save as PNG", "Capture Layers", "Load View Hierarchy", "Display View", "Invalidate Layout", and "Request Layout".

The main area shows a tree view of the UI hierarchy:

- PhoneWindow\$DecorView (@4051e7a0) - 0 children
- LinearLayout (@4051eba8) - 0 children
 - FrameLayout (@4051ede8) - 0 children
 - TextView (@4051f100) - id/title - 0 children
 - FrameLayout (@4051f698) - id/content - 1 child
 - TextView (@4051d630) - 0 children

Each node in the hierarchy includes a small icon with colored dots representing its children. The selected node, TextView (@4051f100), is highlighted with a yellow border. A tooltip for this node shows: "1 view", "Measure: 0.421 ms", "Layout: 0.048 ms", and "Draw: 3.096 ms".

On the right side, there is a "Property" table with columns "Property" and "Value". The table is currently empty. Below the table are a "Show Extras" checkbox (checked) and a "Load All Views" button.

At the bottom, there is a search bar with the text "Filter by class or id: textview", a zoom slider set to 20%, and a "200%" label.

At the bottom right, there is a preview window showing a visual representation of the selected TextView with a red border.