CSE 437

Mobile Computing

App Development Basics

Agenda

Application Fundamentals

Sandboxing Concept

- Android OS is a multi-user Linux system in which each app is a different user
- System assigns each app a unique Linux user (by default)
- System sets permissions for all files in an app. Only user ID assigned can access app
- Each process has its own virtual machine (VM). App's code runs in isolation from other apps
- Each app runs in its own Linux process (by default)

Application Components

Types of Components

Each serves a distinct purpose and has a distinct lifecycle that defines how it is created and destroyed.

Four types of app components:

1. Activities
2. Services
3. Content providers
4. Broadcast receivers

Reference: https://developer.android.com/guide/components/fundamentals.html#Components

Application Components

Activities

- Represents a single screen with a user interface.
    - E.g. calendar app: activity for list of events, activity to create a new event, activity for reading event details.

  (For example, a calendar app might have one activity that shows a list of events, another activity to create a new event, and another activity for reading event details.)

- Work together cohesively, but each activity is independent of the others.

- Different app can start any one of the available activities.

Reference: https://developer.android.com/guide/components/fundamentals.html#Components

Application Components

Services

- Runs in the background to perform long-running operations or work for remote processes.
- Does not provide a user interface.
- Another component can start the service and let it run or bind to it in order to interact with it.

Reference: https://developer.android.com/guide/components/fundamentals.html#Components

Application Components

Content providers

- Manages a shared set of app data
- Allows apps to query or even modify the data of other apps
- Useful for reading and writing data that is private to your app and not shared
- Must implement a standard set of APIs that enable other apps to perform transactions

Reference: https://developer.android.com/guide/components/fundamentals.html#Components

Application Components

Broadcast receivers

- Responds to system-wide broadcast announcements.
- Many broadcasts originate from the system
  - for example, a broadcast announcing that the screen has turned off, the battery is low, or a picture was captured.
- Apps also initiate broadcasts—
  - for example, to let other apps know that some data has been downloaded to the device and is available for them to use.
- Don't display a user interface, but may create a status bar notification to alert the user when a broadcast event occurs.

Reference: https://developer.android.com/guide/components/fundamentals.html#Components

Project Structure in Android Studio

Projects overview

- Contains everything that defines your workspace (source code, assets, test code and build configurations)
- Android Studio creates the necessary structure for all files and makes them visible in the **Project** window.

Reference: https://developer.android.com/studio/projects/index.html

Project Structure in Android Studio

Inside the Android project - Modules

- Collection of source files and build settings that allow you to divide your project into discrete units of functionality.
- Your project can have one or many modules and one module may use another module as a dependency.
- Each module can be independently built, tested, and debugged.
- Module Types:
  1. Android app module
  2. Library module
  3. Google Cloud module

Reference: https://developer.android.com/studio/projects/index.html

Project Structure in Android Studio

Android Project - Structure

Contains build outputs.

Contains private libraries.

Contains all code and resource files for the module.

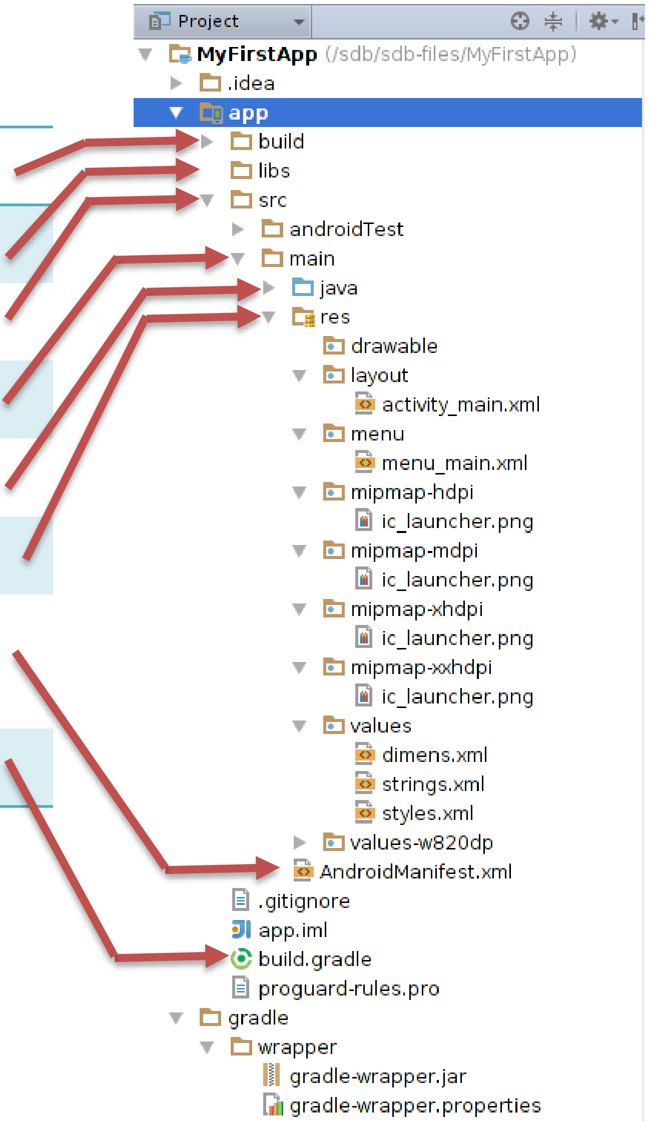
Contains the "main" source set files.

Contains Java code sources.

Contains application resources.

Describes the nature of the application and each of its components.

This defines the module-specific build configurations.



Reference: https://developer.android.com/studio/projects/index.html

Using XML and Android Manifest File

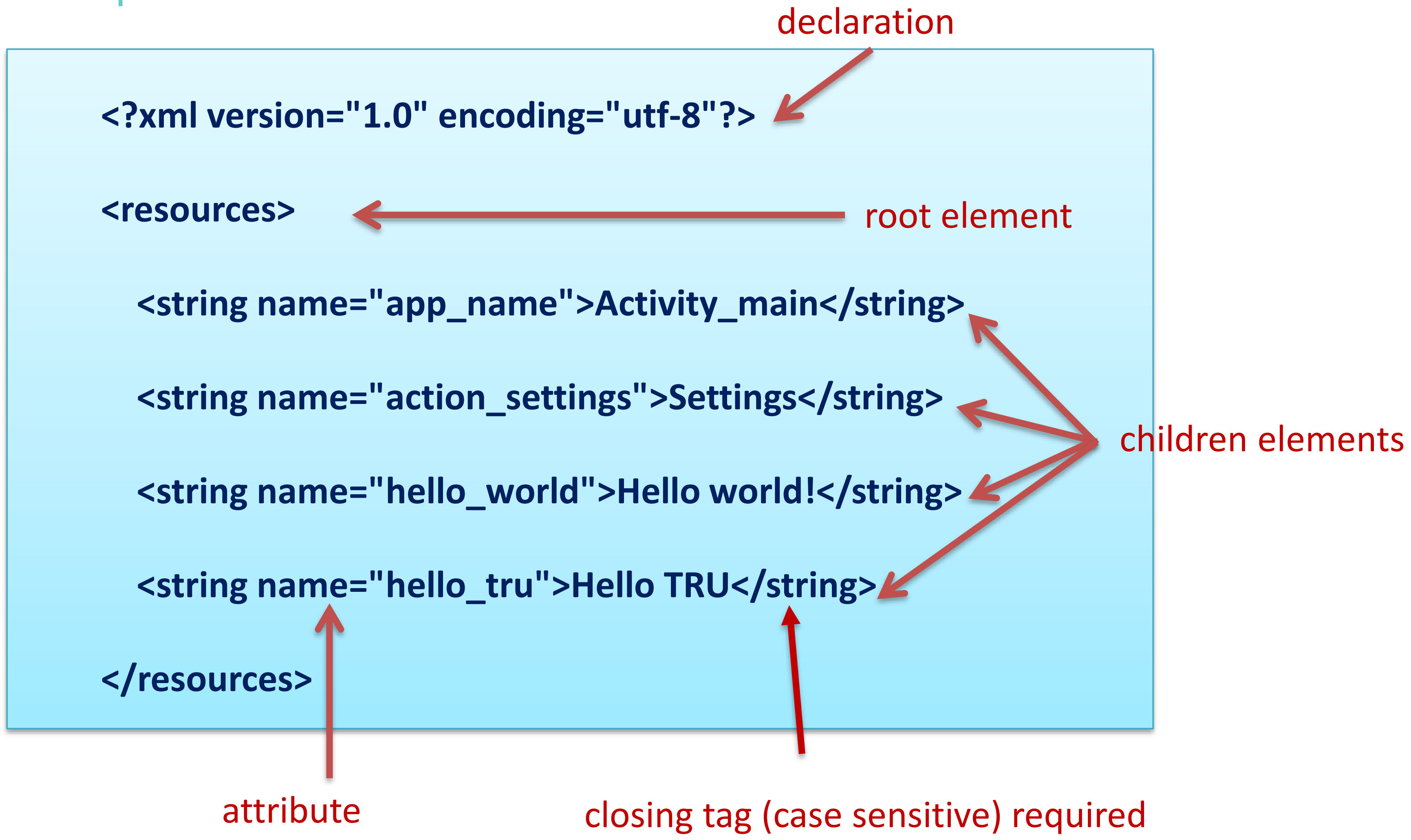
Introduction to XML (EXtensible Markup Language)

- A markup language much like HTML.
- Designed to store and transport data not to present data.
- Self-descriptive and a W3C Recommendation
- Differences Between XML and HTML
  - XML was designed to carry data - with focus on what data is
  - HTML was designed to display data - with focus on how data looks
  - XML tags are not predefined like HTML tags are

Reference: http://www.w3schools.com/xml/xml_whatis.asp

Using XML and Android Manifest File

Example of XML



```
<?xml version="1.0" encoding="utf-8"?>

<resources>

   <string name="app_name">Activity_main</string>

   <string name="action_settings">Settings</string>

   <string name="hello_world">Hello world!</string>

   <string name="hello_tru">Hello TRU</string>

</resources>
```

Reference: http://www.w3schools.com/xml/xml_whatis.asp

Using XML and Android Manifest File

XML Elements

- An XML element is everything from (including) the element's start tag to (including) the element's end tag.
  - <rate>47.32</rate>

- XML elements must follow these naming rules:
  - Element names are case-sensitive
  - Element names must start with a letter or underscore
  - Element names cannot start with the letters xml (or XML, or Xml, etc)
  - Element names can contain letters, digits, hyphens, underscores, and periods
  - Element names cannot contain spaces

Reference: http://www.w3schools.com/xml/xml_whatis.asp

Using XML and Android Manifest File

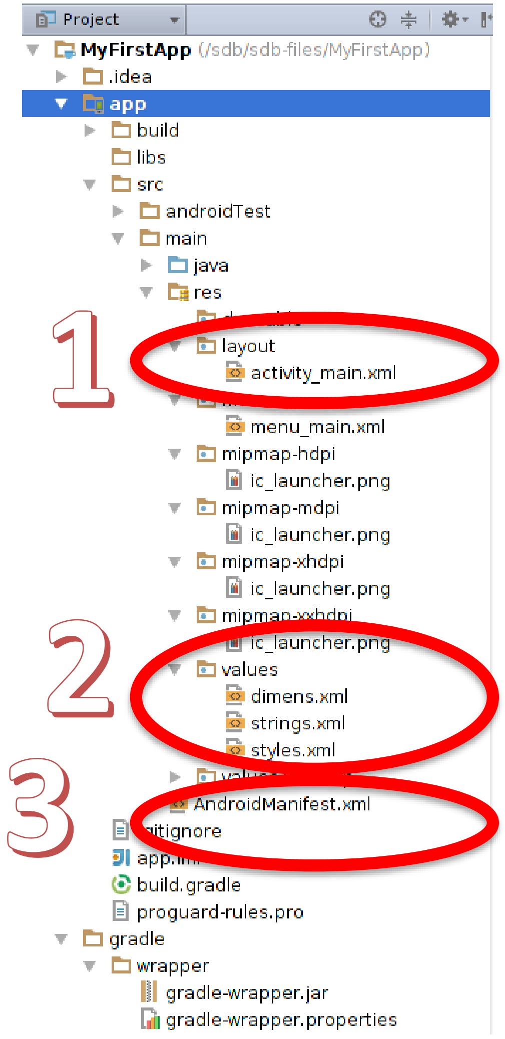
XML Attributes

- XML elements can have attributes, just like HTML.
- Attributes are designed to contain data related to a specific element.
    - <note **date="2008-01-10"**>
- Attribute values must always be quoted. Either single or double quotes can be used.
- Attributes cannot contain multiple values (elements can)
- Attributes cannot contain tree structures (elements can)
- Attributes are not easily expandable (for future changes)

Reference: http://www.w3schools.com/xml/xml_whatis.asp

Using XML and Android Manifest File

XML in Android App Development



1. XML Layout Files

2. XML Value files

3. XML AndroidManifest file

Using XML and Android Manifest File

AndroidManifest.xml

- Names the Java package for the application.
- Describes the components of the application.
- Determines which processes will host application components.
- Declares which permissions the application must have in order to access protected parts of the API and interact with other applications.
- Lists the instrumentation classes that provide profiling and other information as the application is running.

Reference: https://developer.android.com/guide/topics/manifest/manifest-intro.html

Types of Layouts

Layout Declaration

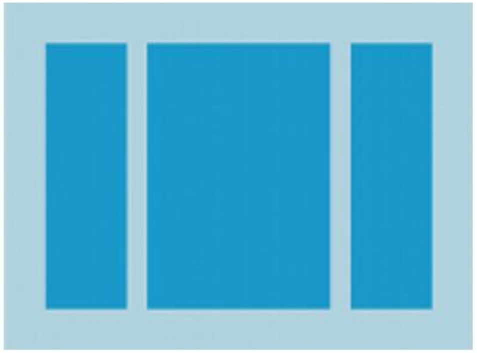
- A layout defines the visual structure for a user interface, such as the UI for an activity or app widget.
- A developer can declare a layout in two ways:
  - Declare UI elements in XML.
  - Instantiate layout elements at runtime.
- The advantage to declaring your UI in XML is that it enables you to better **separate** the *presentation* of your application from the *code* that controls its behavior.

Reference: https://developer.android.com/guide/topics/ui/declaring-layout.html

Types of Layouts

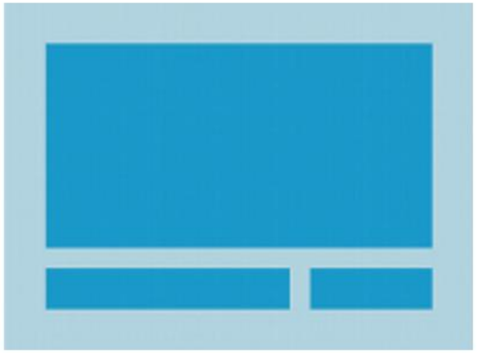
Common Layouts

Linear Layout



A layout that organizes its children into a single horizontal or vertical row. It creates a scrollbar if the length of the window exceeds the length of the screen.

Relative Layout



Enables you to specify the location of child objects relative to each other (child A to the left of child B) or to the parent (aligned to the top of the parent).

Web View

Displays web pages.

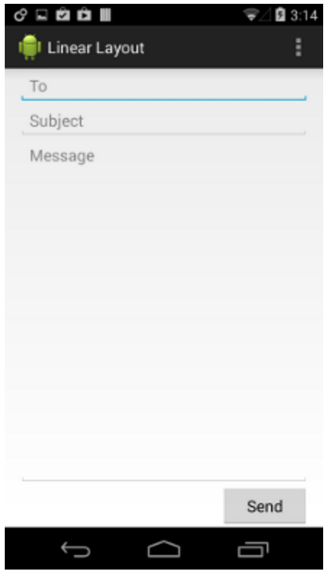
Reference: https://developer.android.com/guide/topics/ui/declaring-layout.html#CommonLayouts

Types of Layouts

Linear Layout

- *LinearLayout* is a view group that aligns all children in a single direction, vertically or horizontally.
- You can specify the layout direction with the **android:orientation** attribute.
- All children of a LinearLayout are stacked one after the other.

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/to" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/subject" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/send" />
</LinearLayout>
```



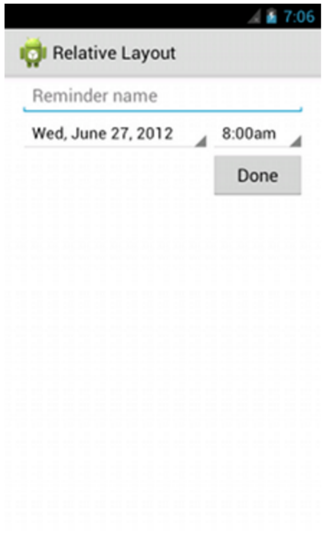
Reference: https://developer.android.com/guide/topics/ui/layout/linear.html

Types of Layouts

Relative Layout

- *RelativeLayout* is a view group that displays child views in relative positions.
- The position of each view can be specified as relative to sibling elements or in positions relative to the parent RelativeLayout area.

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <EditText
        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/reminder" />
    <Spinner
        android:id="@+id/dates"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/times" />
    <Spinner
        android:id="@id/times"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true" />
    <Button
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/times"
        android:layout_alignParentRight="true"
        android:text="@string/done" />
</RelativeLayout>
```

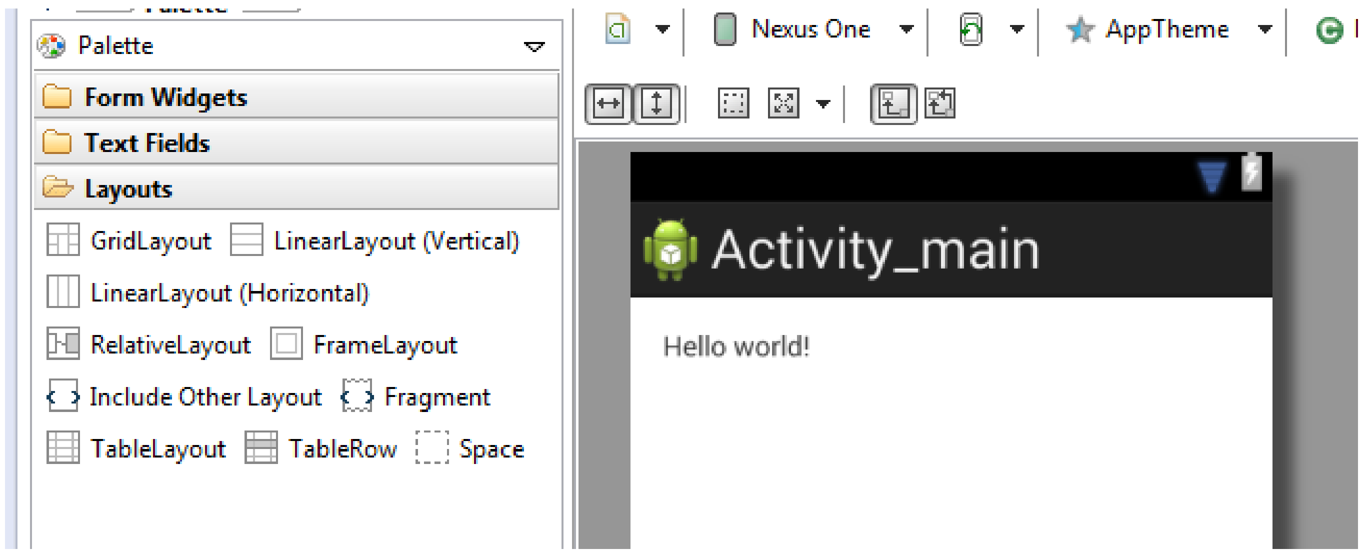


Reference: https://developer.android.com/guide/topics/ui/layout/relative.html

Types of Layouts

Layouts in Palette

- Android Studio Palette provides different types of layouts.
- You can specify width and height with exact measurements, though you probably won't want to do this often. More often, you will use one of these constants to set the width or height:
  - **wrap_content** tells your view to size itself to the dimensions required by its content.
  - **match_parent** tells your view to become as big as its parent view group will allow.

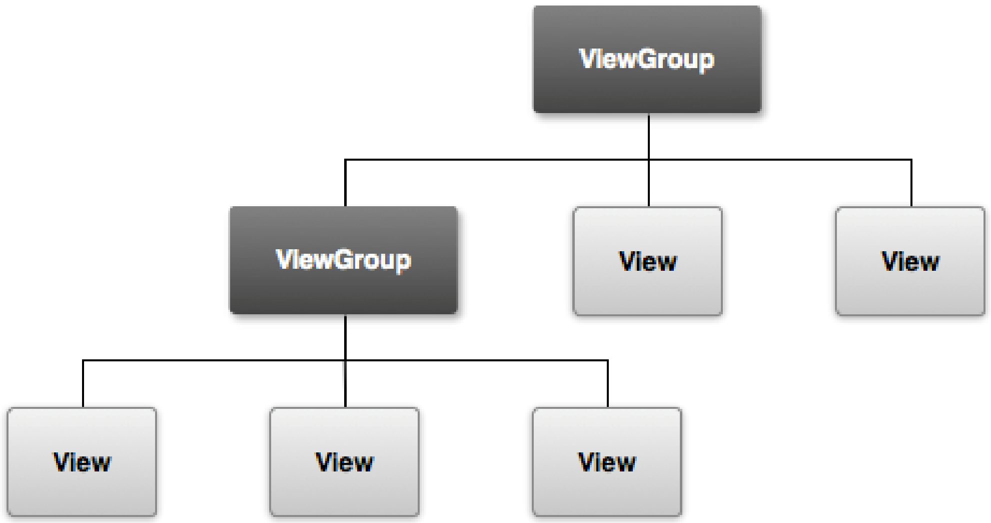


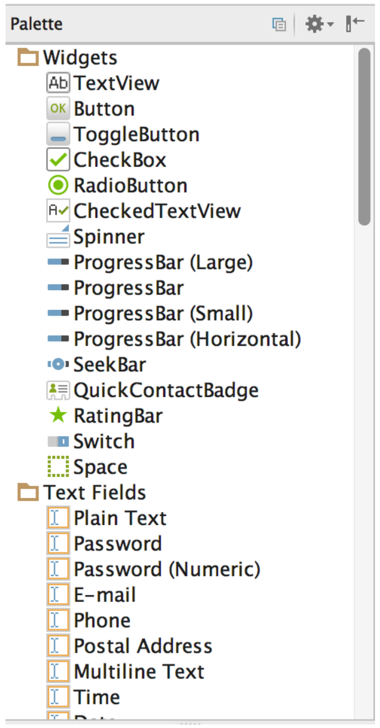
Reference: https://developer.android.com/guide/topics/ui/declaring-layout.html

Types of Layouts

User Interface Layout

- The graphical user interface for an Android app is built using a hierarchy of View and ViewGroup objects.
- View objects are usually UI widgets such as buttons or text fields and ViewGroup objects are invisible view containers that define how the child views are laid out, such as in a grid or a vertical list.



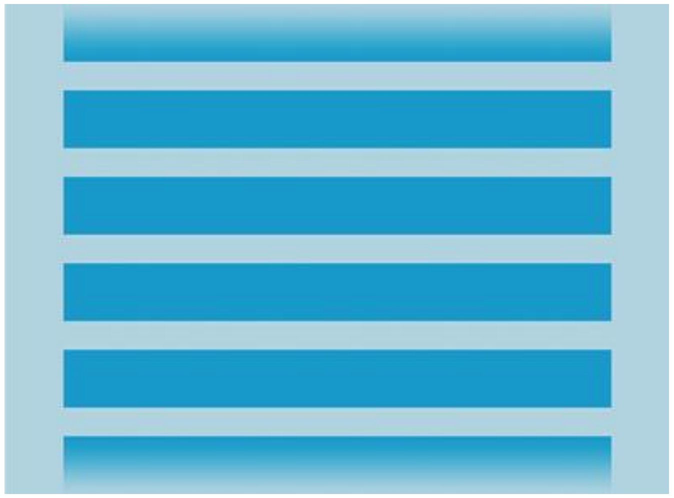


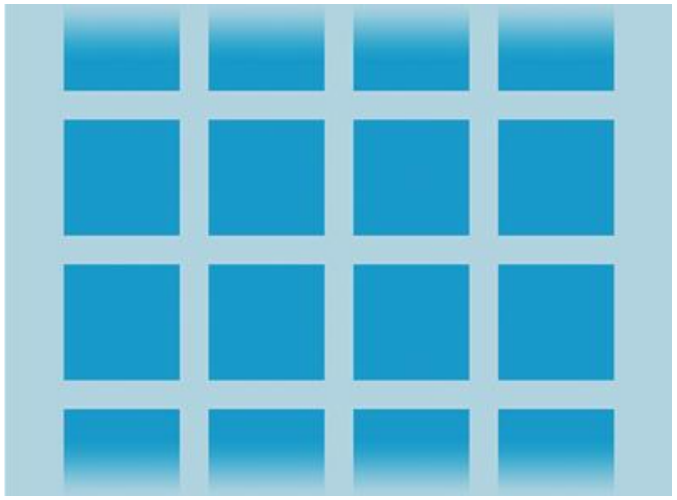
Reference: http://developer.android.com/training/basics/firstapp/building-ui.html#Weight0

Types of Layouts

List and Grid Views

- ListView is a view group that displays a list of scrollable items.
- GridView is a ViewGroup that displays items in a two-dimensional, scrollable grid.


ListView

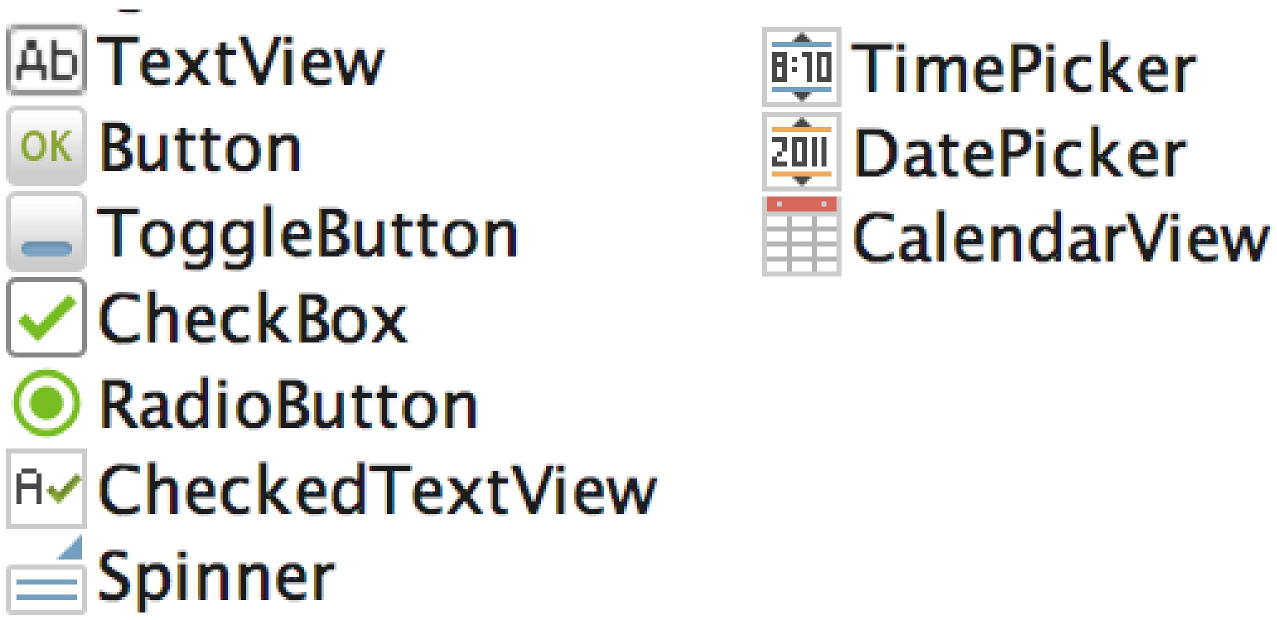

GridView

Reference: http://developer.android.com/guide/topics/ui/declaring-layout.html

Using Input Controls

Common Controls

- Input controls are the interactive components in your app's user interface. Android provides a wide variety of controls you can use in your UI, such as buttons, text fields, seek bars, checkboxes, zoom buttons, toggle buttons, and many more.



Reference: https://developer.android.com/guide/topics/ui/controls.html#CommonControls

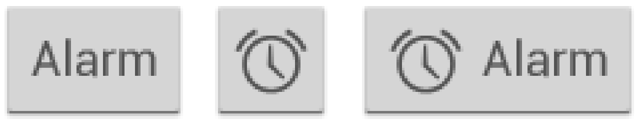
Using Input Controls

Buttons

- Depending on whether you want a button with text, an icon, or both, you can create the button in your layout in three ways

With text, using the Button class:

```
<Button
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/button_text"
   ... />
```



With an icon, using the ImageButton class:

```
<ImageButton
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:src="@drawable/button_icon"
   ... />
```

With text and an icon, using the Button class with the android:drawableLeft attribute:

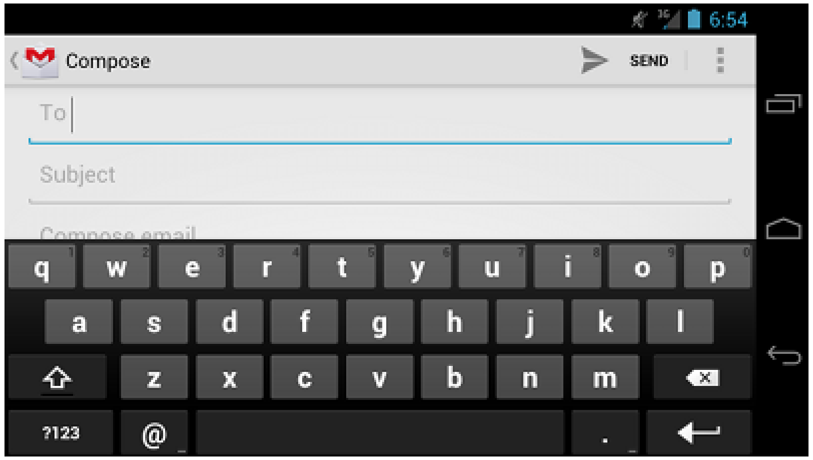
```
<Button
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/button_text"
   android:drawableLeft="@drawable/button_icon"
   ... />
```

Using Input Controls

Text Fields

- You can add a text field to you layout with the EditText object. You should usually do so in your XML layout with a <EditText> element.
- You can specify the type of keyboard you want for your EditText object with the android:inputType attribute.

```
<EditText
    android:id="@+id/email_address"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/email_hint"
    android:inputType="textEmailAddress" />
```



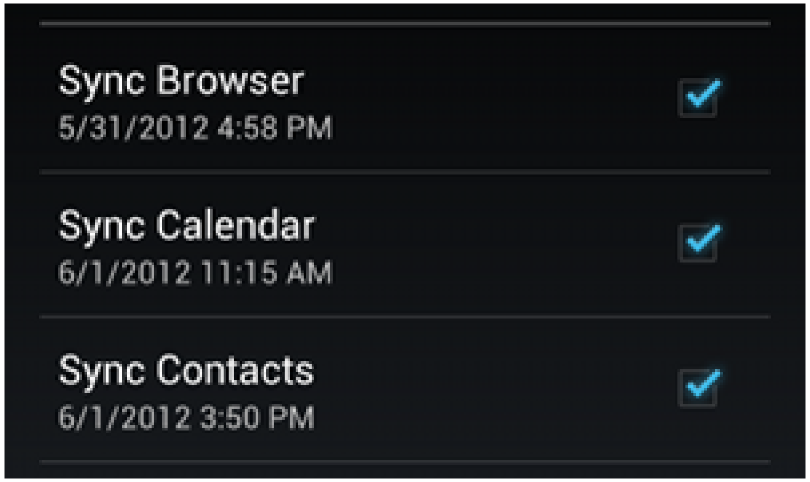
Reference: https://developer.android.com/guide/topics/ui/controls/text.html

Using Input Controls

Checkboxes

- Checkboxes allow the user to select one or more options from a set. Typically, you should present each checkbox option in a vertical list.

- To create each checkbox option, create a CheckBox in your layout. Because a set of checkbox options allows the user to select multiple items, each checkbox is managed separately and you must register a click listener for each one.



Reference: https://developer.android.com/guide/topics/ui/controls/checkbox.html

Using Input Controls

Radio Buttons

- Radio buttons allow the user to select one option from a set.
- You should use radio buttons for optional sets that are mutually exclusive if you think that the user needs to see all available options side-by-side.
- If it's not necessary to show all options side-by-side, use a spinner instead.

```
<?xml version="1.0" encoding="utf-8"?>
<RadioGroup xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:orientation="vertical">
   <RadioButton android:id="@+id/radio_pirates"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/pirates"
      android:onClick="onRadioButtonClicked"/>
   <RadioButton android:id="@+id/radio_ninjas"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/ninjas"
      android:onClick="onRadioButtonClicked"/>
</RadioGroup>
```

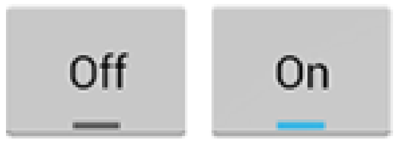
Reference: https://developer.android.com/guide/topics/ui/controls/radiobutton.html

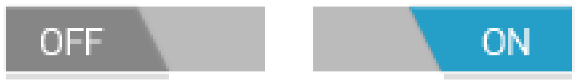
Using Input Controls

Toggle Buttons

- You can add a basic toggle button to your layout with the ToggleButton object. Android 4.0 (API level 14) introduces another kind of toggle button called a switch that provides a slider control, which you can add with a Switch object.

- If you need to change a button's state yourself, you can use the CompoundButton.setChecked() orCompoundButton.toggle() methods.



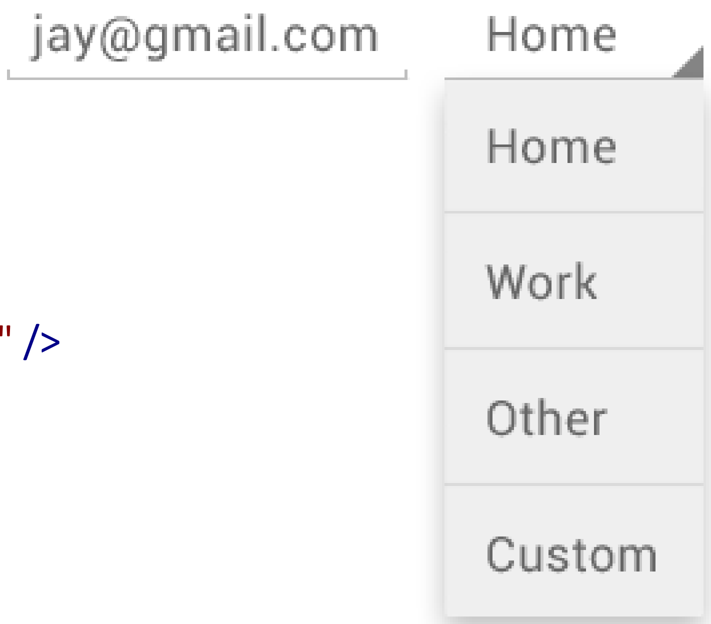


Reference: https://developer.android.com/guide/topics/ui/controls/togglebutton.html

Using Input Controls

Spinners

- Spinners provide a quick way to select one value from a set.



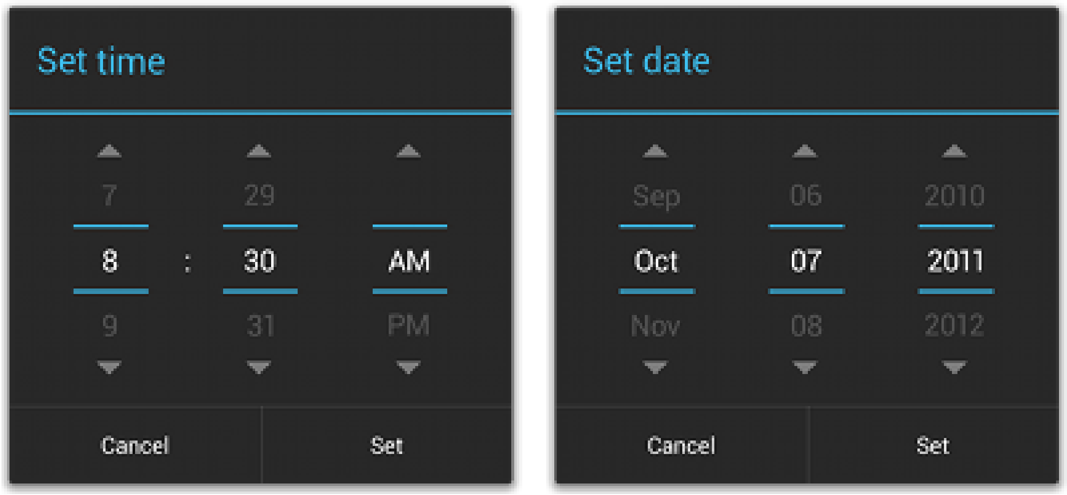
```
<Spinner
   android:id="@+id/planets_spinner"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content" />
```

Reference: https://developer.android.com/guide/topics/ui/controls/spinner.html

Using Input Controls

Pickers

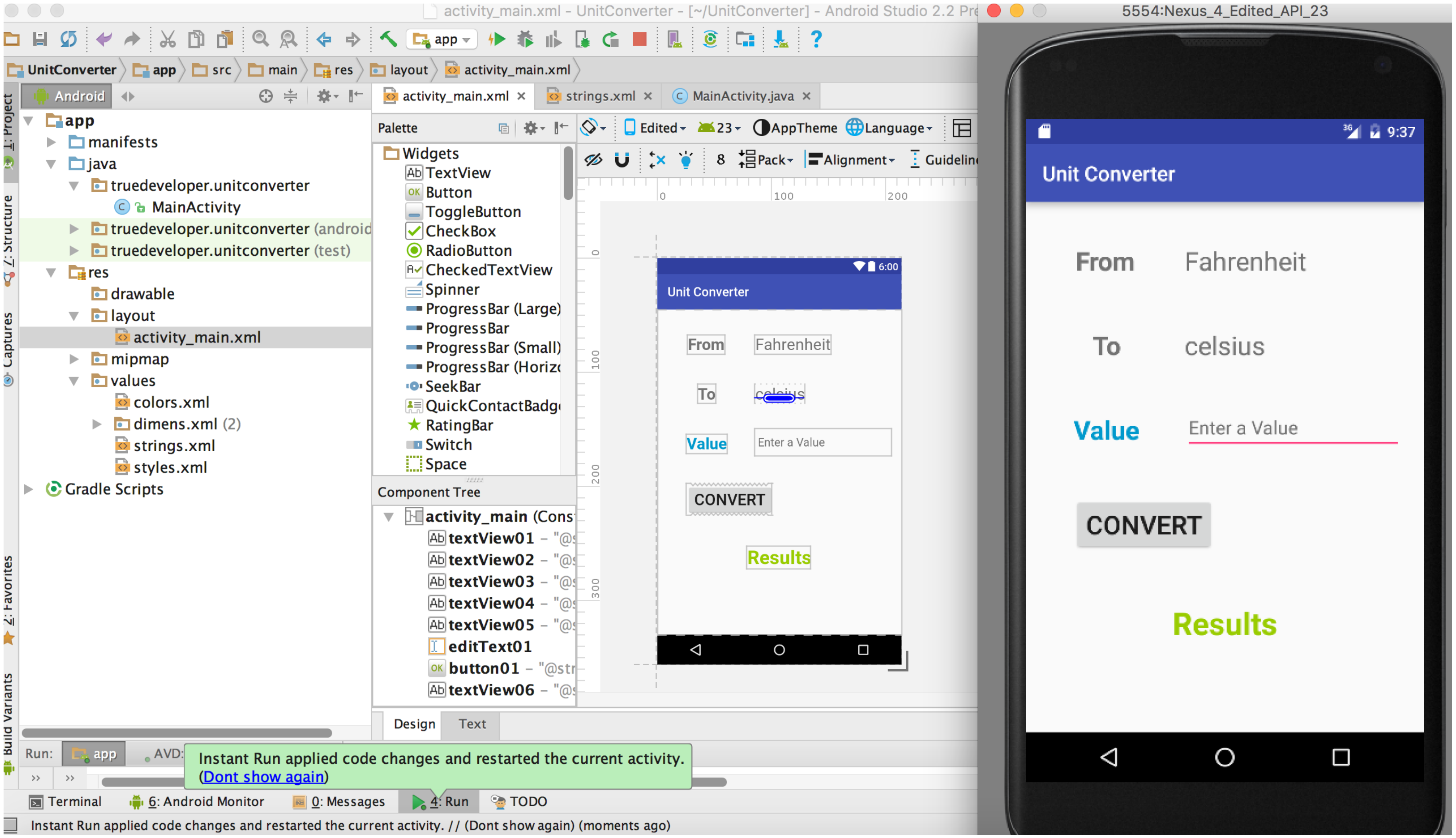
- Android provides controls for the user to pick a time or pick a date as ready-to-use dialogs. Each picker provides controls for selecting each part of the time (hour, minute, AM/PM) or date (month, day, year).



Reference: https://developer.android.com/guide/topics/ui/controls/pickers.html

Types of Layouts

Layout Design of a Unit Converter App



Instant Run applied code changes and restarted the current activity. (Dont show again)

Instant Run applied code changes and restarted the current activity. // (Dont show again) (moments ago)

Responding to Events

Buttons

1. Using ***android:onClick*** attribute

a layout with a button using android:onClick:

```
<?xml version="1.0" encoding="utf-8"?>
<Button xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/button_send"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send"
    android:onClick="sendMessage" />
```

Within the Activity that hosts this layout, the following method handles the click event:

```
/** Called when the user touches the button */
public void sendMessage(View view) {
    // Do something in response to button click
}
```

Responding to Events

Buttons

2. Using an ***OnClickListener***

```
Button button = (Button) findViewById(R.id.button_send);

button.setOnClickListener(new View.OnClickListener() {

    public void onClick(View v) {

        // Do something in response to button click

    }

});
```

Responding to Events

Checkboxes

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
  <CheckBox android:id="@+id/checkbox_meat"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/meat"
    android:onClick="onCheckboxClicked"/>
  <CheckBox android:id="@+id/checkbox_cheese"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/cheese"
    android:onClick="onCheckboxClicked"/>
</LinearLayout>
```

Within the Activity that hosts this layout, the following method handles the click event for both checkboxes:

```
public void onCheckboxClicked(View view) {
  // Is the view now checked?
  boolean checked = ((CheckBox) view).isChecked();

  // Check which checkbox was clicked
  switch(view.getId()) {
    case R.id.checkbox_meat:
      if (checked)
        // Put some meat on the sandwich
      else
        // Remove the meat
      break;
    case R.id.checkbox_cheese:
      if (checked)
        // Cheese me
      else
        // I'm lactose intolerant
      break;
    // TODO: Veggie sandwich
  }
}
```

Responding to Events

Spinners

- Populate the Spinner with User Choices

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string-array name="planets_array">
      <item>Mercury</item>
      <item>Venus</item>
      <item>Earth</item>
      <item>Mars</item>
      <item>Jupiter</item>
      <item>Saturn</item>
      <item>Uranus</item>
      <item>Neptune</item>
   </string-array>
</resources>
```

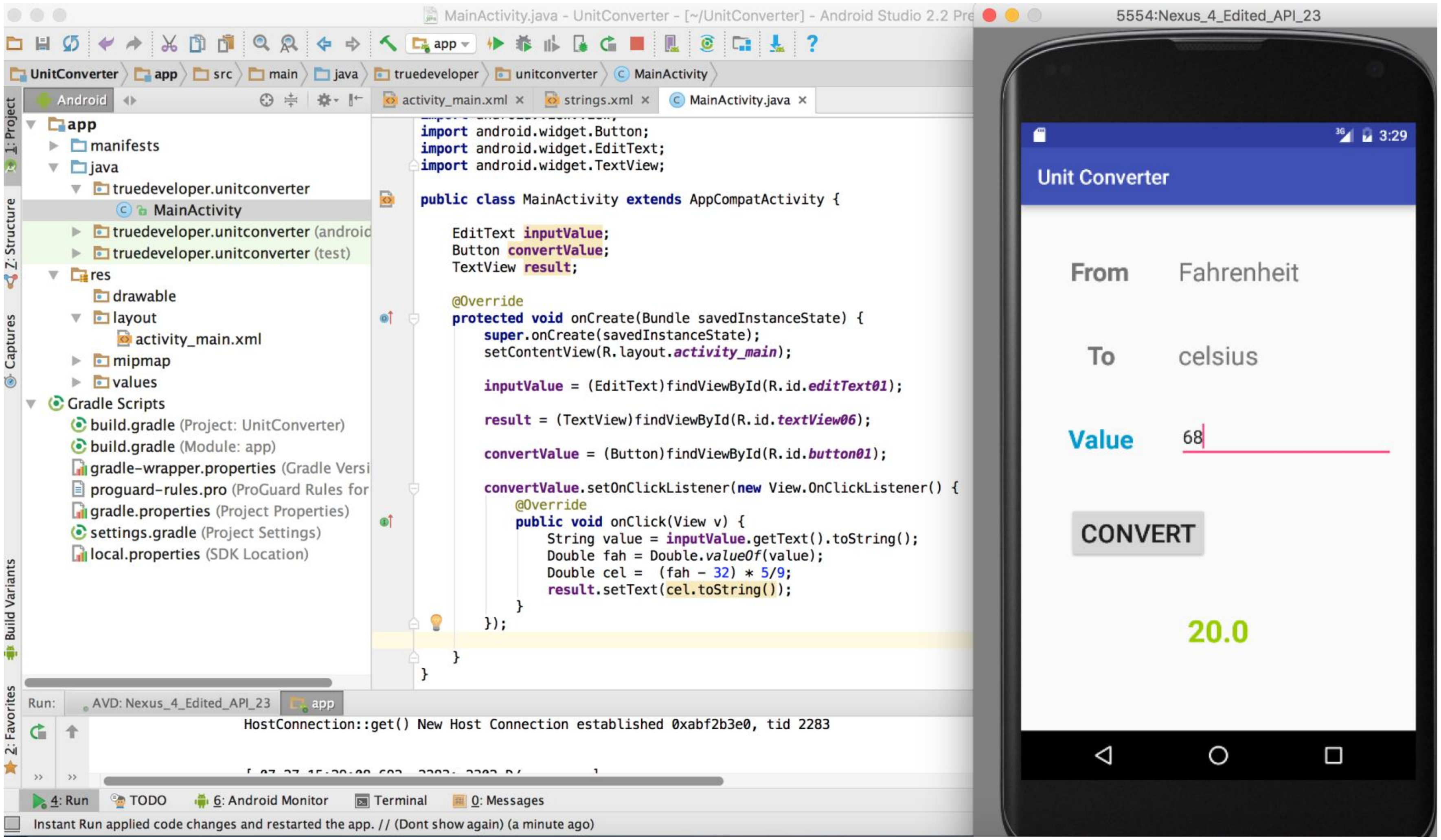
```
Spinner spinner = (Spinner)
findViewById(R.id.spinner);
// Create an ArrayAdapter using the string array
and a default spinner layout
ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(this,
        R.array.planets_array,
android.R.layout.simple_spinner_item);

adapter.setDropDownViewResource(android.R.la
yout.simple_spinner_dropdown_item);
// Apply the adapter to the spinner
spinner.setAdapter(adapter);
```

Reference: https://developer.android.com/guide/topics/ui/controls/spinner.html

Responding to Events

Reading and Writing onClick


MainActivity.java - UnitConverter - [~/UnitConverter] - Android Studio 2.2 Pre
UnitConverter
app
src
main
java
truedeveloper
unitconverter
MainActivity
Android
activity_main.xml
strings.xml
MainActivity.java
1: Project
Z: Structure
Captures
Build Variants
2: Favorites
app
manifests
java
truedeveloper.unitconverter
MainActivity
truedeveloper.unitconverter (androic
truedeveloper.unitconverter (test)
res
drawable
layout
activity_main.xml
mipmap
values
Gradle Scripts
build.gradle (Project: UnitConverter)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Versi
proguard-rules.pro (ProGuard Rules for
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
Run:
AVD: Nexus_4_Edited_API_23
app
4: Run
TODO
6: Android Monitor
Terminal
0: Messages
5554:Nexus_4_Edited_API_23
3:29
Unit Converter
From
Fahrenheit
To
celsius
Value
68
CONVERT
20.0

```java
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    EditText inputValue;
    Button convertValue;
    TextView result;

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

        inputValue = (EditText)findViewById(R.id.editText01);

        result = (TextView)findViewById(R.id.textView06);

        convertValue = (Button)findViewById(R.id.button01);

        convertValue.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String value = inputValue.getText().toString();
                Double fah = Double.valueOf(value);
                Double cel =  (fah - 32) * 5/9;
                result.setText(cel.toString());
            }
        });

    }
}
```

```
HostConnection::get() New Host Connection established 0xabf2b3e0, tid 2283
```

Instant Run applied code changes and restarted the app. // (Dont show again) (a minute ago)

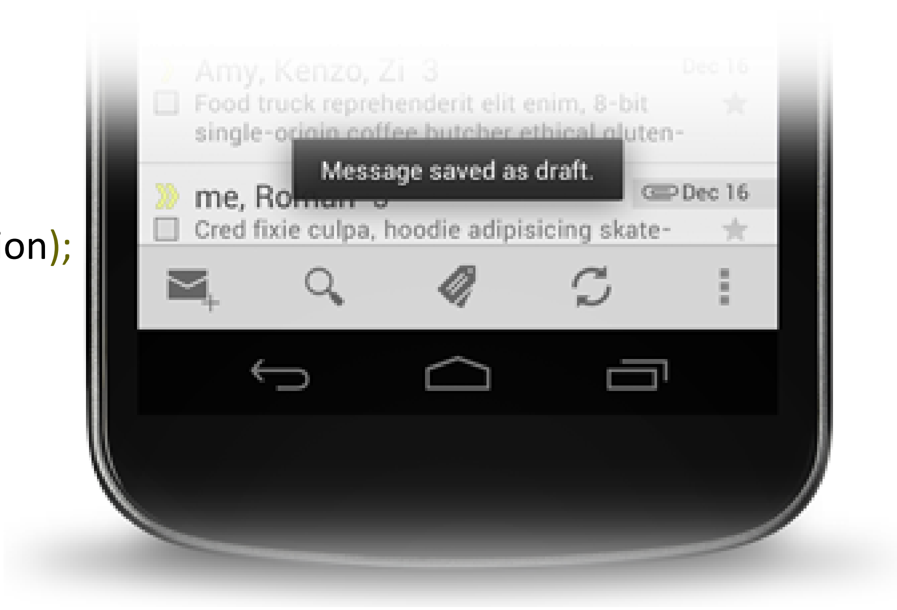
Responding to Events

Adding Toast Messages

A toast provides simple feedback about an operation in a small popup.

```
Context context = getApplicationContext();
CharSequence text = ”Message saved as draft";
int duration = Toast.LENGTH_SHORT;

Toast toast = Toast.makeText(context, text, duration);
toast.show();
```


Amy, Kenzo, Zi 3
Dec 16
Food truck reprehenderit elit enim, 8-bit
single-origin coffee butcher ethical gluten-
Message saved as draft.
me, Roman 3
Dec 16
Cred fixie culpa, hoodie adipisicing skate-

Reference: https://developer.android.com/guide/topics/ui/notifiers/toasts.html

Thank You