Android JumpStart Workshop

By IEEE SJEC SB, Computer Society Chapter

Manual

Widgets

The widget package contains (mostly visual) UI elements to use on your Application screen.

- TextView: It is used to display text to the user.

```
<TextView
    android:id="@+id/helloText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World "
    android:textSize="18sp" />
```

- EditText: A user interface element for entering and modifying text.

```
<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Enter Name" />
```

- Button:

```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/button"
    android:text="Click Here" />
```

- ImageView: It is used to display an image file.

```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/imagename" />
```

- ProgressBar: A user interface element that indicates the progress of an operation.

```
<ProgressBar
    android:id="@+id/progressBar"
    style="?android:attr/progressBarStyle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="57dp" />
```

Layouts

Relative Layout

RelativeLayout is a view group that displays child views in relative positions

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp">
    <TextView
        android:id="@+id/helloText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        android:text="Hello World" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/helloText"
        android:layout_centerHorizontal="true"
        android:text="Button" />
</RelativeLayout>
```

XML Attributes	Description
layout_alignParentTop	Accepts boolean value. If specified, the widget will be docked to the top of RelativeLayout.
layout_alignParentBottom	Accepts boolean value. If specified, the widget will be docked to the bottom of RelativeLayout.
layout_alignParentLeft	Accepts boolean value. If specified, the widget will be docked to the left edge of RelativeLayout.
layout_alignParentRight	Accepts boolean value. If specified, the widget will be docked to the right edge of RelativeLayout.
layout_centerInParent	Accepts boolean value. If specified, the widget will be aligned to center of RelativeLayout.
layout_centerHorizontal	Accepts boolean value. If specified, the widget will be horizontally center aligned

layout_centerVertical	Accepts boolean value. If specified, the widget will be vertically center aligned
layout_below	Accepts sibling widget id. Places the widget below the view as specified widget id.
layout_above	Accepts sibling widget id. Places the widget above the specified widget id.
layout_toRightOf	Accepts sibling widget id. Places the widget to right of the view as specified widget id.
layout_toLeftOf	Accepts sibling widget id. Places the widget to left of the view as specified widget id.
layout_toEndOf	Accepts sibling widget id. Places the widget to end of the view as specified widget id.
layout_toStartOf	Accepts sibling widget id. Places the widget to the beginning of the view as specified widget id.

Linear Layout

A layout that arranges other views either horizontally in a single column or vertically in a single row.

(horizontal orientation)

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >
    <Button
        android:id="@+id/Apple"
        android:text="Button 1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <Button
        android:id="@+id/Mango"
        android:text=" Button 2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <Button
        android:id="@+id/Banana"
        android:text=" Button 3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

(vertical orientation)

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <Button
        android:id="@+id/Apple"
        android:text="Button 1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <Button
        android:id="@+id/Mango"
        android:text=" Button 2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <Button
        android:id="@+id/Banana"
        android:text=" Button 3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

Xml attributes	description
layout_weight	Weight of each child proportionally.
weightSum	Sum up of child weight
gravity	This specifies how an object should position its content, on both the X and Y axes. Possible values are top, bottom, left, right, center, center vertical, center horizontal etc.
orientation	This specifies the direction of arrangement and you will use "horizontal" for a row, "vertical" for a column. The default is horizontal.

Creating References

activity_main.xml

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/relative_layout">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/text"
        android:text="Hello"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button"
        android:text="click"
        android:layout_below="@+id/Text"/>
</RelativeLayout>
```

MainActivity.java

```
RelativeLayout relativeLayout = (RelativeLayout)
findViewById(R.id.relative_layout);


Button button = (Button)findViewById(R.id.button);
TextView textView = (TextView)findViewById(R.id.text);
```

Basic View Operations

```
Button button = (Button)findViewById(R.id.button);
TextView textView = (TextView)findViewById(R.id.text);


textView.setText("Text is being set");
textView.setTextColor(Color.RED);


button.setText("Click Me");
button.setTextColor(Color.BLACK);
```

OnClickListeners

OnClickListener is used when you want your components to react when users click on them.

activity_main.xml

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:text="Click Me" />

</RelativeLayout>
```

MainActivity.java

```
Button button = (Button) findViewById(R.id.button);
button.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View view) {
        Toast.makeText(MainActivity.this, "Button Clicked",
     Toast.LENGTH_SHORT).show();
    }
});
```

Intent

An Android Intent is an abstract description of an operation to be performed. It can be used with startActivity to launch an Activity,

```
Intent i = new Intent(this, ActivityTwo.class);
startActivity(i);
```

To send parameter to newly created activity `putExtra()` method will be used.

```
Intent i = new Intent(this, ActivityTwo.class);
i.putExtra("extra1",125);
i.putExtra("extra2","This is being passed");
startActivity(i);
```

To receive parameters on newly created activity **`getExtras()`** method will be used.

```
String data = getIntent().getExtras().getString("extra2");
int value=getIntent().getExtras().getInt("extra1");
```

Log

- **Log.e**: This is for when bad stuff happens. Use this tag in places like inside a catch statement. You *know* that an *error* has occurred and therefore you're logging an error.
- **Log.w**: Use this when you suspect something shady is going on. You may not be completely in full on error mode, but maybe you recovered from some unexpected behavior. Basically, use this to log stuff you didn't expect to happen but isn't necessarily an error. Kind of like a "hey, this happened, and it's *weird*, we should look into it."
- **Log.i**: Use this to post useful *information* to the log. For example: that you have successfully connected to a server. Basically use it to report successes.
- **Log.d**: Use this for *debugging* purposes. If you want to print out a bunch of messages so you can log the exact flow of your program, use this. If you want to keep a log of variable values, use this.
- **Log.v**: Use this when you want to go absolutely nuts with your logging. If for some reason you've decided to log every little thing in a particular part of your app, use the Log.v tag.

```
Log.v("MyActivity", "this logged");
```

ListView

Android ListView is a view which groups several items and display them in vertical scrollable list. The list items are automatically inserted to the list using an Adapter that pulls content from a source such as an array or database.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ListView
        android:id="@+id/list_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

</RelativeLayout>
```

row.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/itemText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="20dp"
        android:textColor="#222"
        android:textSize="18sp"
        tools:text="item" />

</LinearLayout>
```

MainActivity.java

```
//To be placed as instance variables
ListView listView;
ArrayList<String> list = new ArrayList<>();


//To be placed in onCreate() method of the class

list.add("Element 1");
list.add("Element 2");
list.add("Element 3");
list.add("Element 4");
list.add("Element 5");
list.add("Element 6");

listView = findViewById(R.id.list_view);

SimpleAdapter adapter = new SimpleAdapter(this, list);
listView.setAdapter(adapter);

listView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view,
int position, long id) {
        Toast.makeText(MainActivity.this, list.get(position),
Toast.LENGTH_SHORT).show();
    }
});
```

SimpleAdapter.java

```
class SimpleAdapter extends BaseAdapter {

    private Context ctx;
    private List<String> list;
    private LayoutInflater inflater;

    SimpleAdapter(Context ctx, List<String> list) {
        this.ctx = ctx;
        this.list = list;
        inflater = LayoutInflater.from(ctx);
    }

    @Override
    public int getCount() {
        return list.size();
    }

    @Override
    public Object getItem(int position) {
        return list.get(position);
    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
    public View getView(int position, View convertView,
ViewGroup parent) {
        View view = inflater.inflate(R.layout.row, parent,
false);
        TextView itemText = view.findViewById(R.id.itemText);

        itemText.setText(list.get(position));

        return view;
    }

}
```