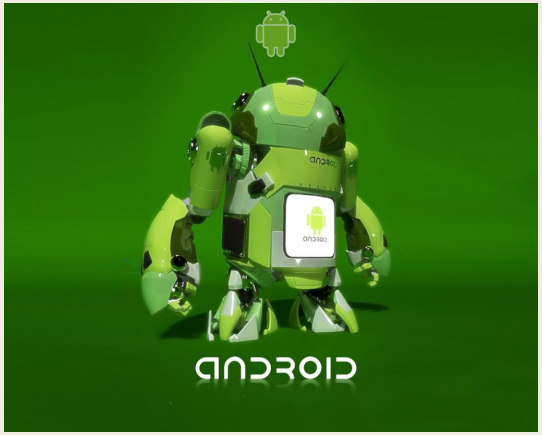
# Android Quickstart

Building our First Application

# Selecting an App

Let's begin by building a **simple** and **functional** Android application.

- Best way to **learn** Android is to **build** (simple) apps
- Let's take a look at the **essentials** for apps
- We are going to **build** a simple **Todo** List




# Scoping our Todo App

## Let's scope our Todo app before we get started

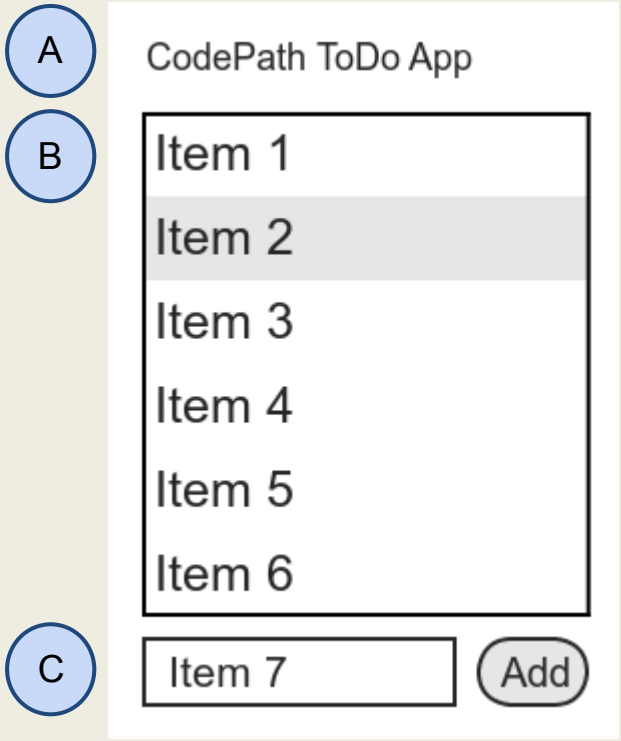
- User can **view** a list of existing todo items
- User can **add** a new item to the todo list
- User can **remove** an item from the todo list

This means that for this application, we just need **one screen** which allows us to view, add and remove simple list items.

This first version will **not support** "marking" completion or setting priority, but we could add this in a **future** version.

# Wireframe the Todo App

Let's wireframe our Todo App next, to sketch the basic interface:

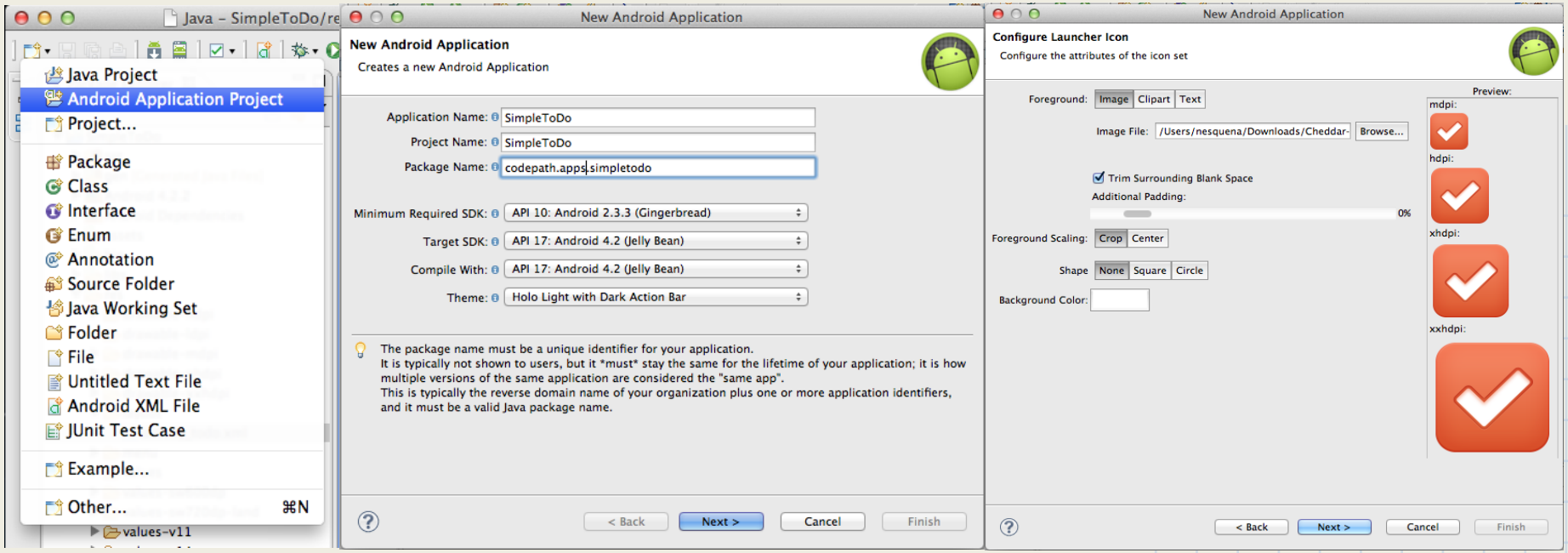



- (A) Basic Label with App Name
- (B) Basic List of Items
  - Vertically Scrollable
  - Hold Down to Remove Item
- (C) Adding with Textbox and Button

# Building the Todo App

Now that we have scoped and wireframed our basic app, let's get started coding.

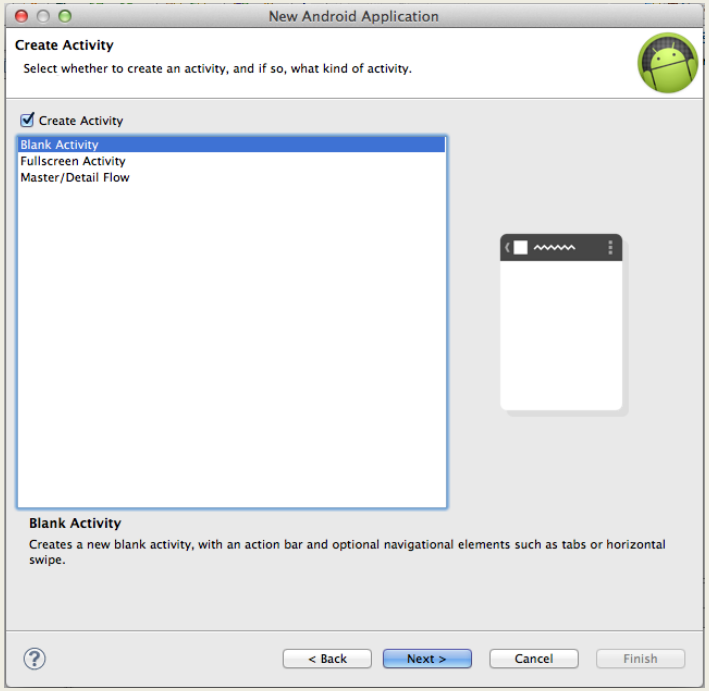
- Let's create a new Android project in Eclipse

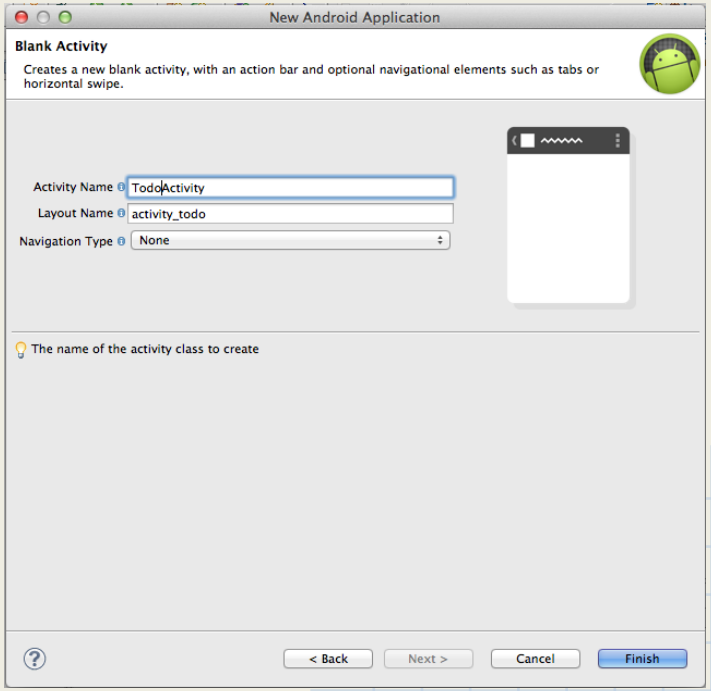


# Building the Todo App

Now that we have scoped and wireframed our basic app, let's get started coding.

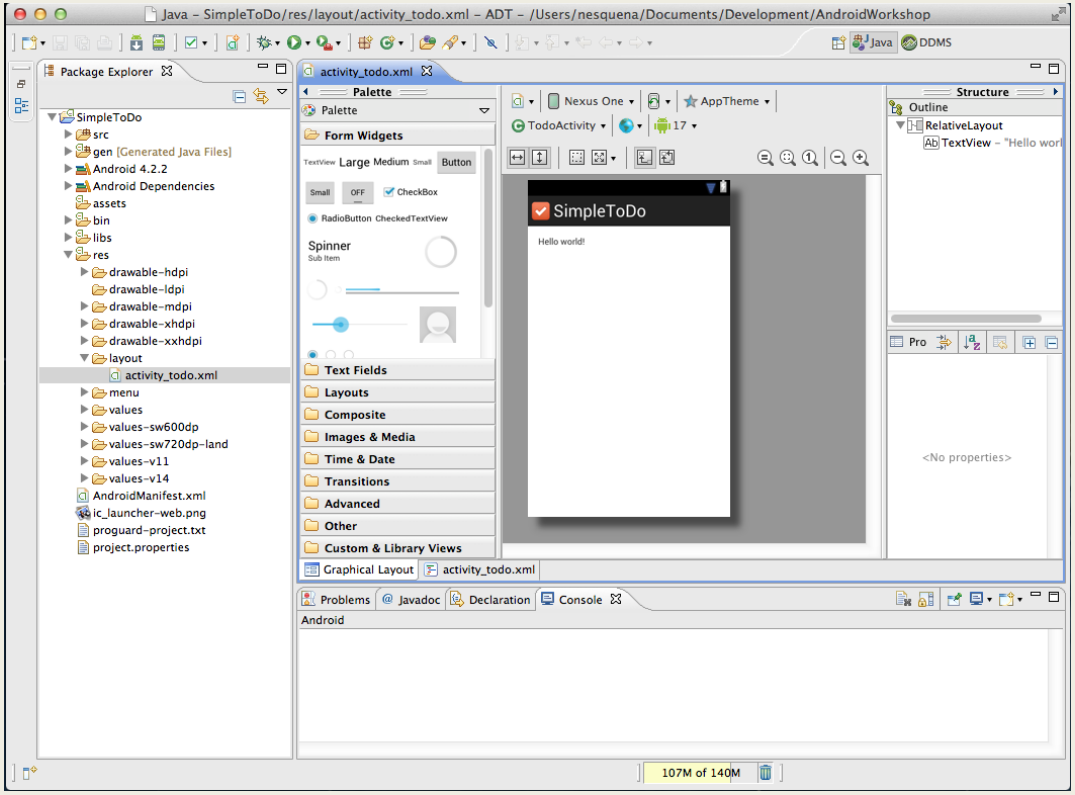
- Let's generate a new Android project in Eclipse



# Building the Todo App

Now that we have scoped and wireframed our basic app, let's get started coding.




- Notice the first Activity is open for us by default
- We can start visually building our ToDo application right away
- You may notice this screen is actually an XML file, click on the second tab to reveal the file

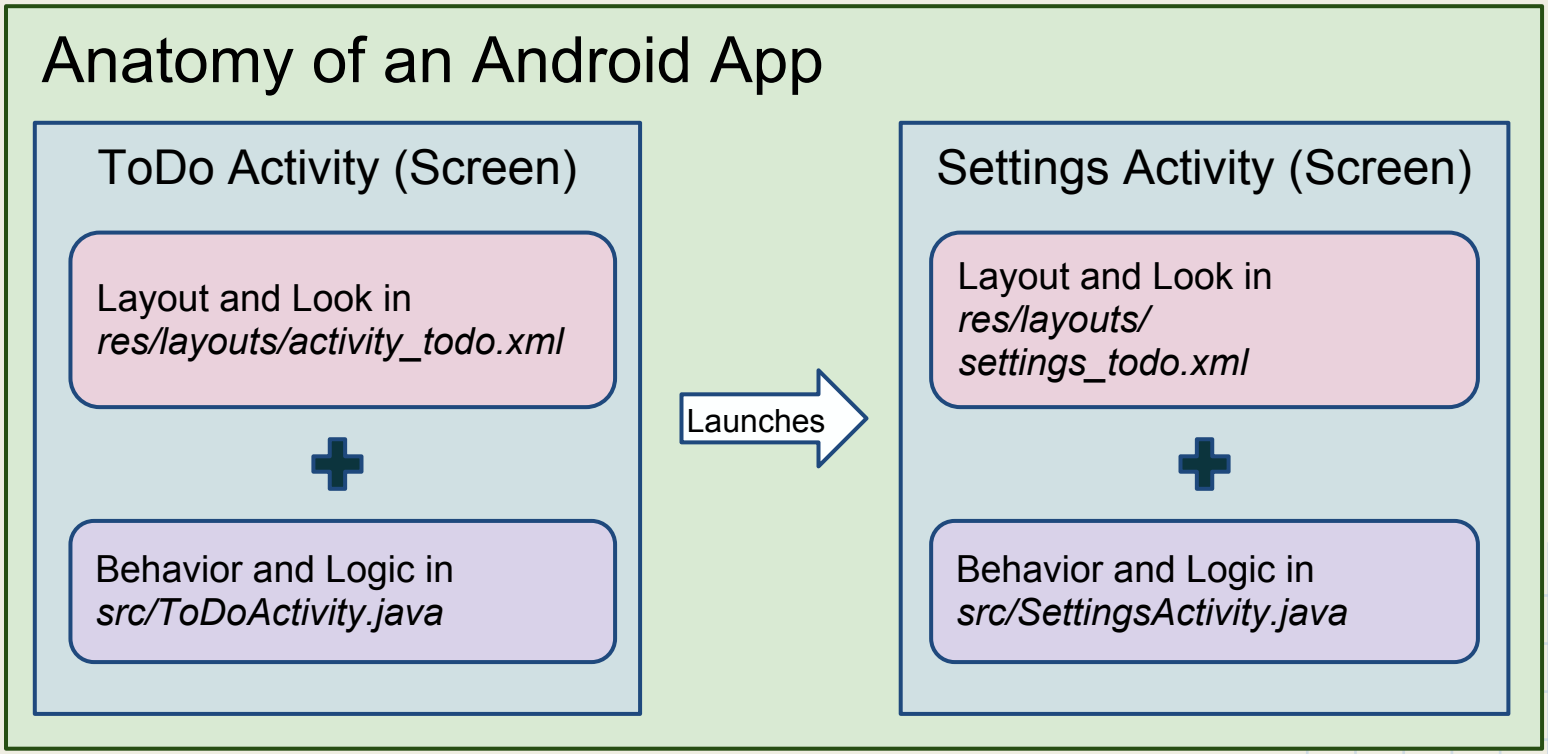
# Short Orientation

Let's quickly orient ourselves with the Android framework.

| **Activity** | **res/ layouts** | **src/ *.java** |
|---|---|---|
| **==** | **==** | **==** |
| 1 UI Screen | Look | Behavior |

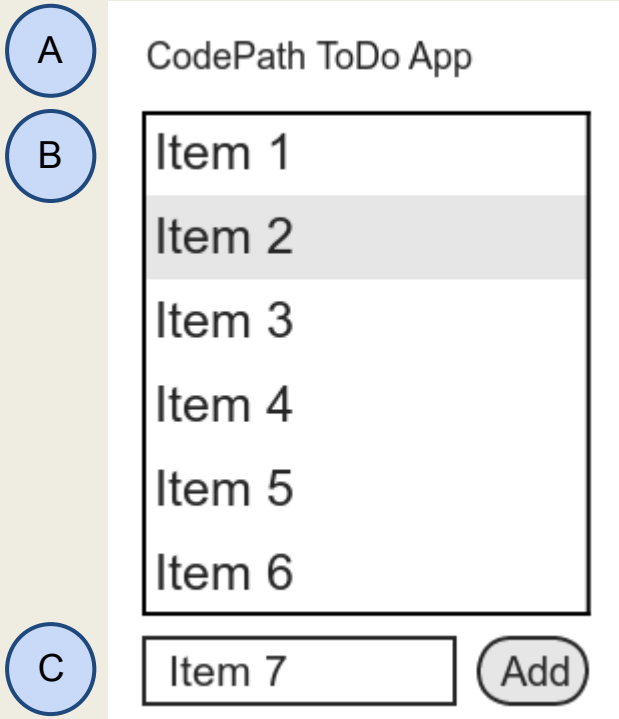
# Short Orientation

Let's quickly orient ourselves with the Android framework.



# Building the Interface

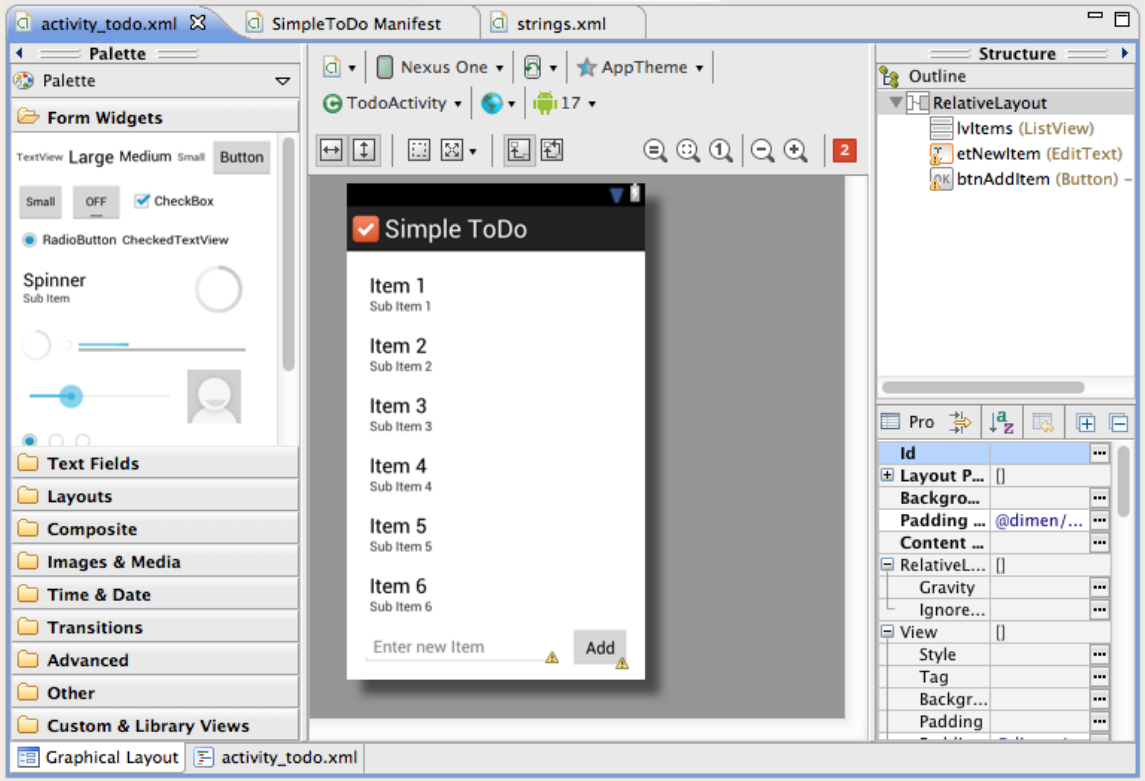
## Let's layout the interface for our application:




- (A) Basic Label with App Name
- (B) Basic List of Items
  - Horizontally Scrollable
  - Hold Down to Remove Item
- (C) Adding with Textbox and Button

# Building the Interface

## Let's layout the interface for our application:




- Drag Views onto Layout
  - Composite → ListView
  - Text Fields → PlainText
  - Form Widgets → Button
- Assign Layout **Height**
  - ListView → 380dp
- Assign Hint to EditText
- Assign ID to Views
  - ListView = lvItems
  - EditText = etNewItem
  - Button = btnAddItem

# Building the Interface

## Using the Visual UI Builder automatically generates the relevant XML:

activity_todo.xml | SimpleToDo Manifest | strings.xml

```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".TodoActivity" >

    <ListView
        android:id="@+id/lvItems"
        android:layout_width="match_parent"
        android:layout_height="380dp"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true" >
    </ListView>

    <EditText
        android:id="@+id/etNewItem"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/lvItems"
        android:layout_below="@+id/lvItems"
        android:layout_marginTop="14dp"
        android:ems="10"
        android:hint="Enter new Item" />

    <Button
        android:id="@+id/btnAddItem"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/etNewItem"
        android:layout_alignBottom="@+id/etNewItem"
        android:layout_alignRight="@+id/lvItems"
        android:text="Add" />

</RelativeLayout>
```

Graphical Layout | activity_todo.xml

- Every action we took was translated into this XML
- Notice all three views are wrapped in a "Layout"
- All three views have their listed properties (id, height, width, et al)

# Coding the Behavior

Let's code the basic ToDo List behavior. We code in the Java source file for this activity:

Package Explorer

- SimpleToDo
  - src
    - codepath.apps.simpletodo
      - TodoActivity.java
  - gen [Generated Java Files]
  - Android 4.2.2
  - Android Dependencies
  - assets
  - bin
  - libs
  - res
    - drawable-hdpi
    - drawable-ldpi
    - drawable-mdpi
    - drawable-xhdpi
    - drawable-xxhdpi
    - layout
      - activity_todo.xml
    - menu
    - values
    - values-sw600dp
    - values-sw720dp-land
    - values-v11
    - values-v14
  - AndroidManifest.xml
  - ic_launcher-web.png
  - proguard-project.txt
  - project.properties

activity_todo.xml | SimpleToDo Manifest | strings.xml | TodoActivity.java

```java
package codepath.apps.simpletodo;

import android.os.Bundle;

public class TodoActivity extends Activity {

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


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.todo, menu);
        return true;
    }

}
```

- onCreate is where the XML layout for this activity is applied
- Notice every Activity extends from the same base class
- This file is where we add our application logic

# Coding the Behavior

Let's create the basic list of items and display them in the ListView.

```
public class TodoActivity extends Activity {
    ArrayList<String> items;
    ArrayAdapter<String> itemsAdapter;
    ListView lvItems;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_todo);
        lvItems = (ListView) findViewById(R.id.lvItems);
        items = new ArrayList<String>();
        itemsAdapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_1, items);
        lvItems.setAdapter(itemsAdapter);
        items.add("First Item");
        items.add("Second Item");
    }
```

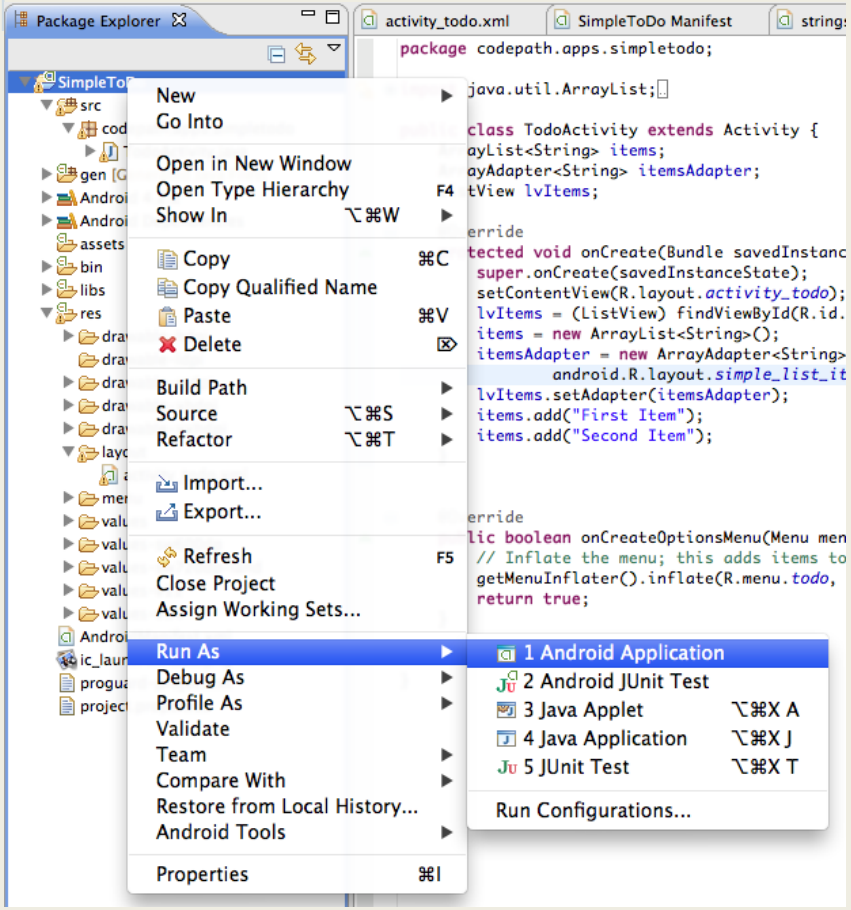
Here's what we are doing:

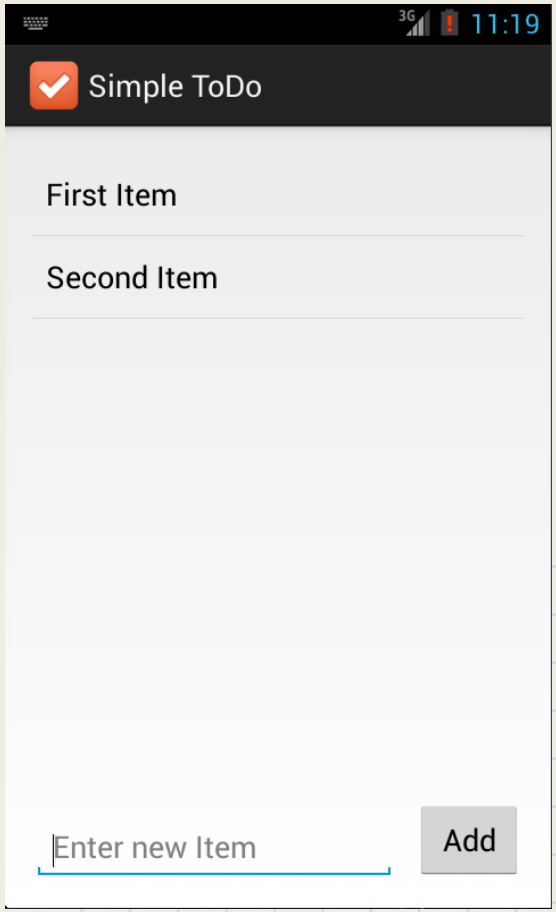
- Creating an ArrayList
- Creating an ArrayAdapter
- Get a handle to ListView
- Attach adapter to ListView

An adapter allows us to easily display the contents of an ArrayList within a ListView.

# Testing the Behavior

## Let's run the app in our emulator:



# Coding the Behavior

## Now let's support adding items to our list:

activity_todo.xml | SimpleToDo Manifest | strings.xml

```
<Button
    android:id="@+id/btnAddItem"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBottom="@+id/etNewItem"
    android:layout_alignRight="@+id/lvItems"
    android:onClick="addTodoItem"
    android:text="Add" />
```

Graphical Layout | activity_todo.xml

TodoActivity.java | activity_todo.xml | SimpleToDo Manifest

```
public void addTodoItem(View v) {
    EditText etNewItem = (EditText)
            findViewById(R.id.etNewItem);
    itemsAdapter.add(etNewItem.getText().toString());
    etNewItem.setText("");
}
```

Here's what we are doing:

- Add "onClick" property to Button in XML
- Define "addTodoItem" method to activity which adds input item to the list.

# Coding the Behavior

## Now let's support removing items from the list:

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_todo);
    lvItems = (ListView) findViewById(R.id.lvItems);
    items = new ArrayList<String>();
    itemsAdapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_1, items);
    lvItems.setAdapter(itemsAdapter);
    items.add("First Item");
    items.add("Second Item");
    setupListViewListener();
}

private void setupListViewListener() {
    lvItems.setOnItemLongClickListener(new OnItemLongClickListener() {
        @Override
        public boolean onItemLongClick(AdapterView<?> aView,
                View item, int pos, long id) {
            items.remove(pos);
            itemsAdapter.notifyDataSetInvalidated();
            return true;
        }
   });
}
```

Here's what we are doing:

- Defines a new method for setting up the listener and invokes from onCreate
- Attach a "LongClickListener" to each Item for ListView:
  - Removes that item
  - Refreshes the adapter.

# Coding the Behavior

## Now let's support loading/saving items from a file:

```
private void readItems() {
    File filesDir = getFilesDir();
    File todoFile = new File(filesDir, "todo.txt");
    try {
        items = new ArrayList<String>(FileUtils.readLines(todoFile));
    } catch (IOException e) {
        items = new ArrayList<String>();
        e.printStackTrace();
    }
}

private void saveItems() {
    File filesDir = getFilesDir();
    File todoFile = new File(filesDir, "todo.txt");
    try {
        FileUtils.writeLines(todoFile, items);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Here's what we are doing:

- Opening a file and reading a newline-delimited list of items.
- Opening a file and writing a newline-delimited list of items.

# Coding the Behavior

## Adding the readItems and writeItems methods:

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

    readItems();
```

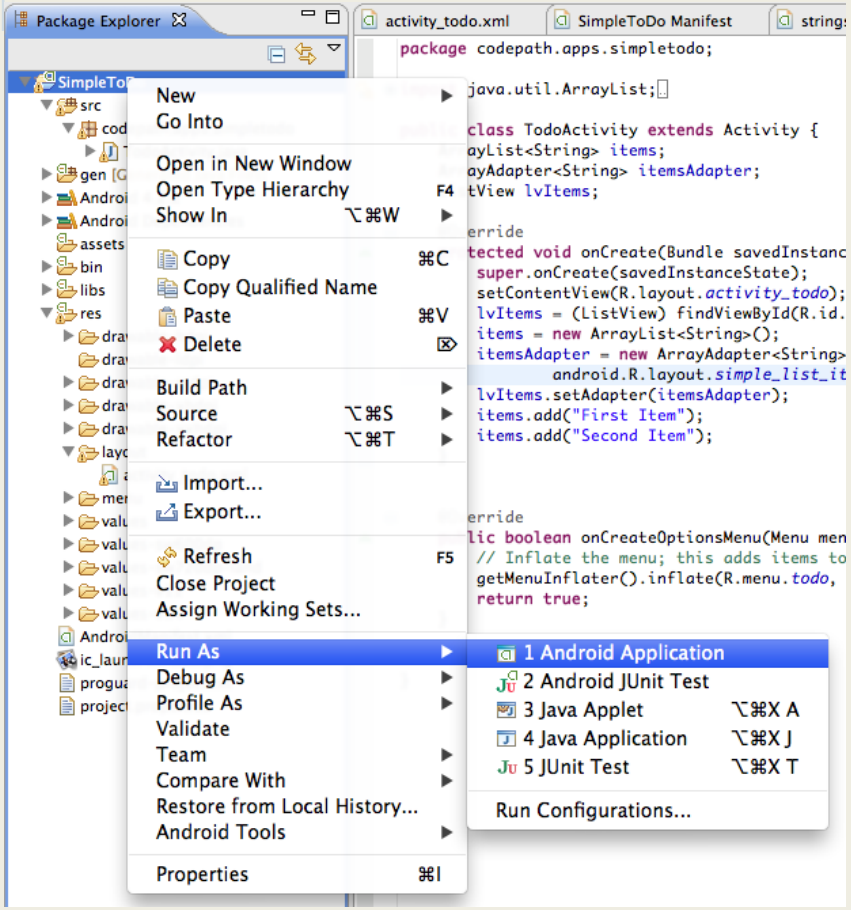
```
lvItems.setOnItemLongClickListener(new OnItemLongClickListener() {

    @Override
    public boolean onItemLongClick(AdapterView<?> parent, View view,
            int position, long rowId) {
        items.remove(position);
        listAdapter.notifyDataSetChanged();
        saveItems();
        return true;
    }
});
```

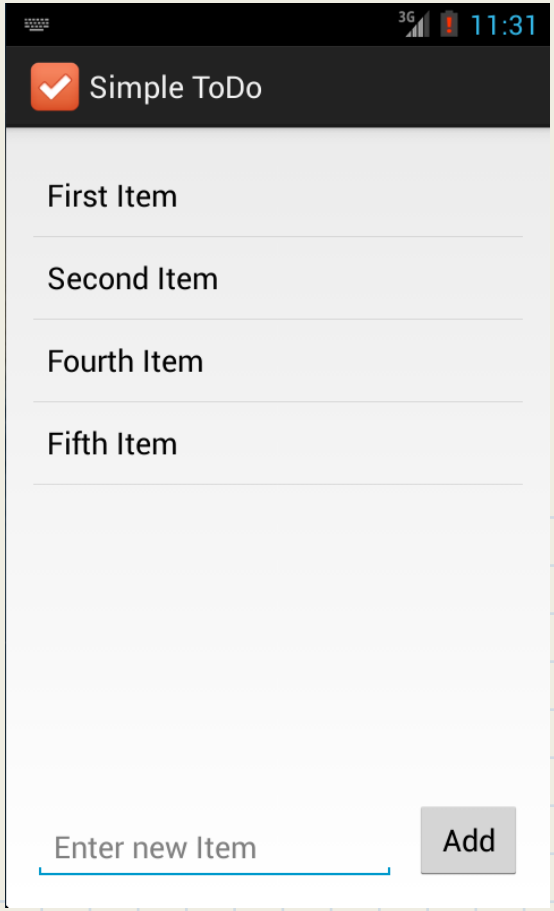
Here's what we are doing:

- Loading the items onCreate
- Saving the items when a new one is added

# Testing the Behavior

## Let's run the app again in our emulator:



# ToDo App Summary

We have now built our very first functioning application using many essential concepts:

- Activity XML (Layouts and Views)
- Activity Source (Java Code for App Logic)
- View IDs and Properties
- ListViews, EditText and Button View Types
- List Adapters for Displaying List Items
- Click Handling for Buttons and List Items
- Testing Applications with the Emulator

What you would add to the ToDo List next?