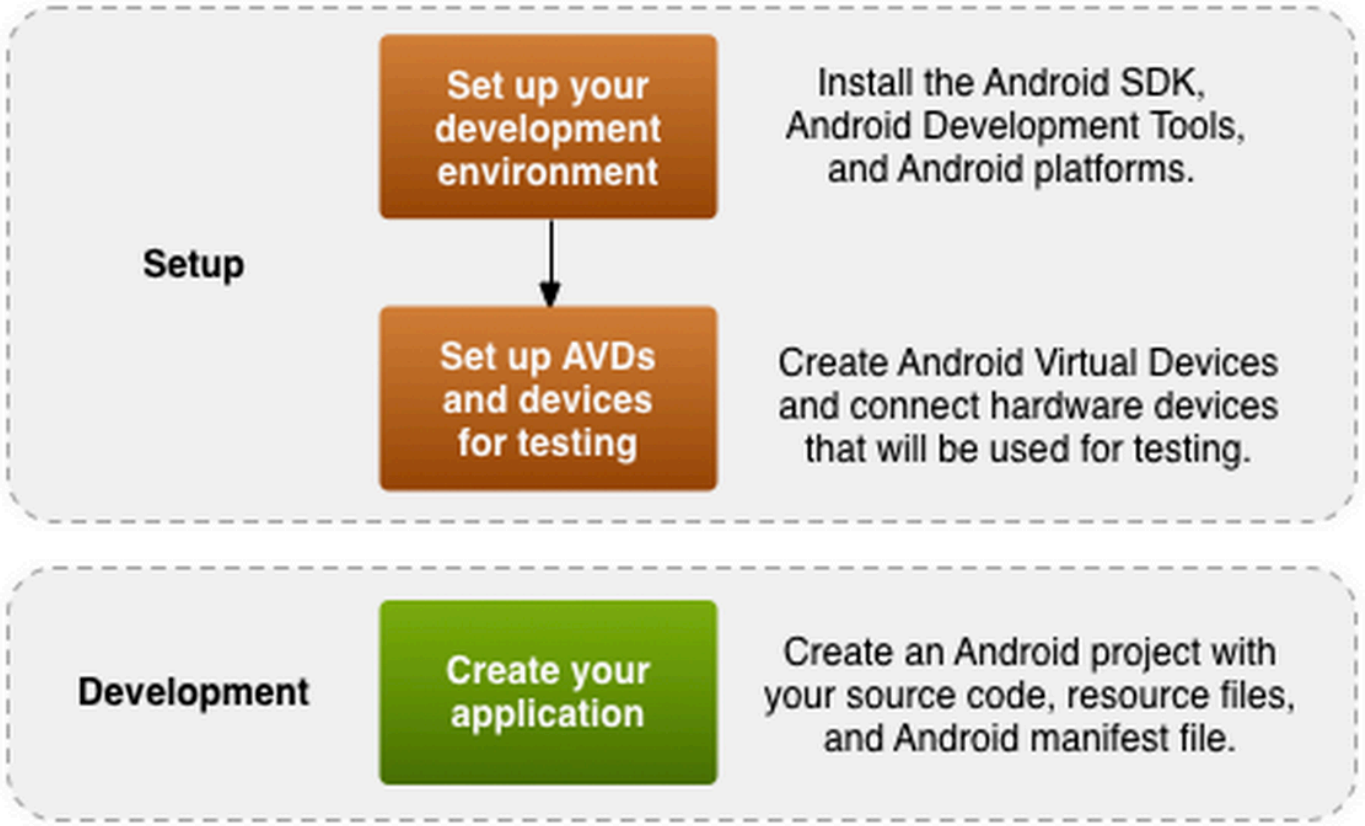
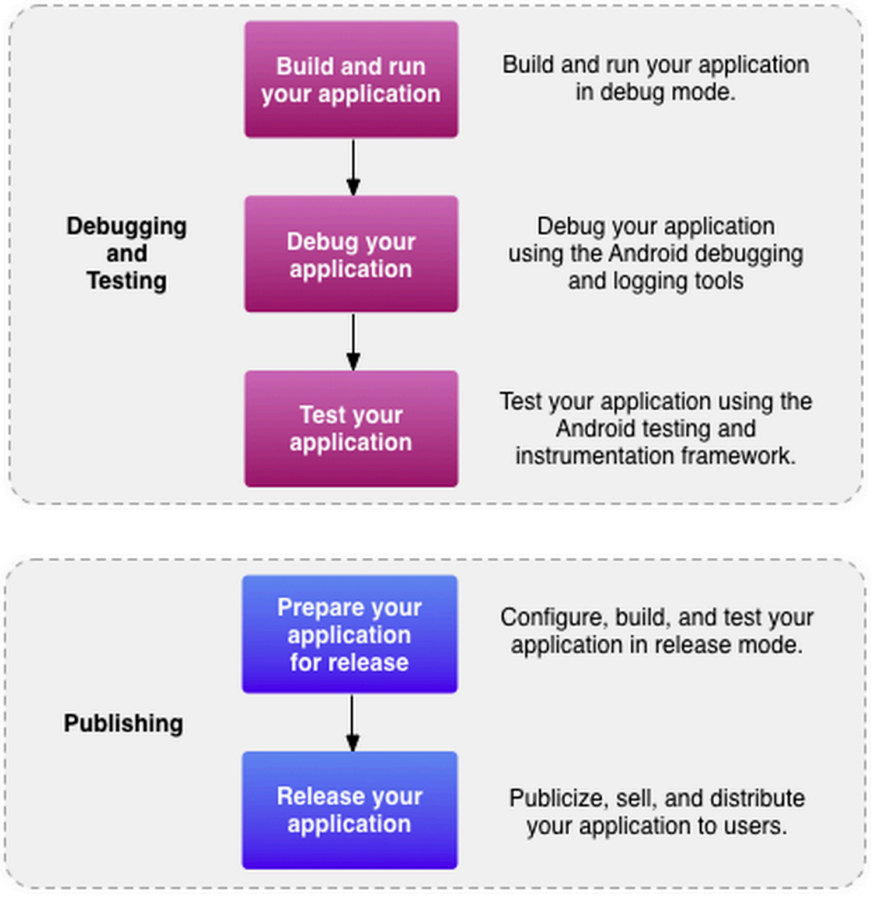
Introduction to Android

Building a Todo Application with Android Studio

Android App Workflow



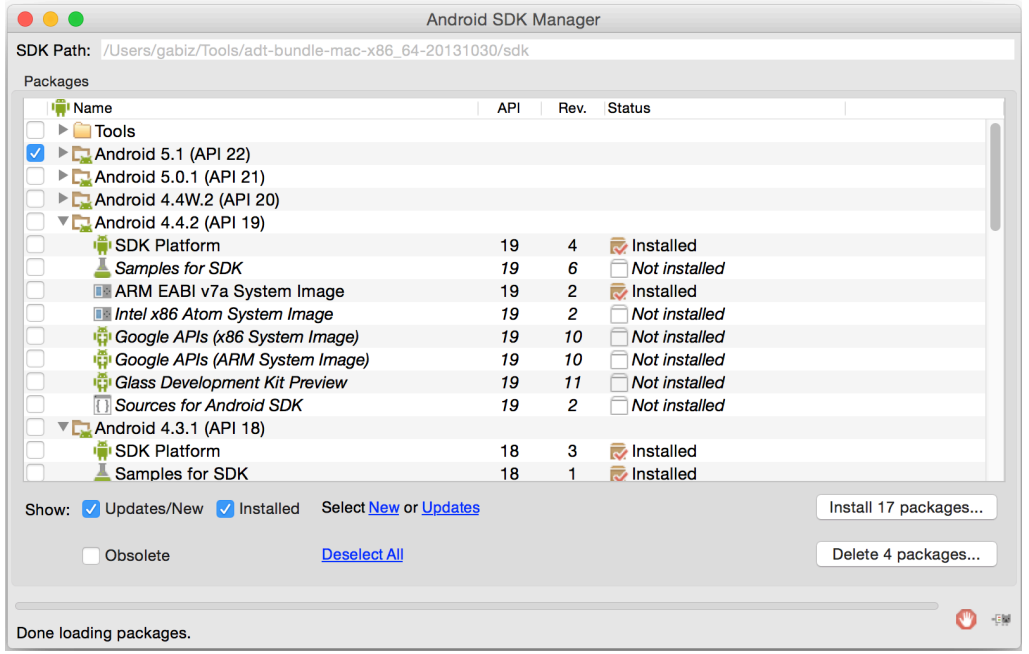
Android App Workflow (cont)



Android SDK Manager

Enables downloading individual tools, platforms and other components.

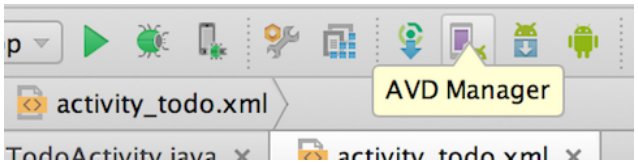
Manages tools for different API Levels.

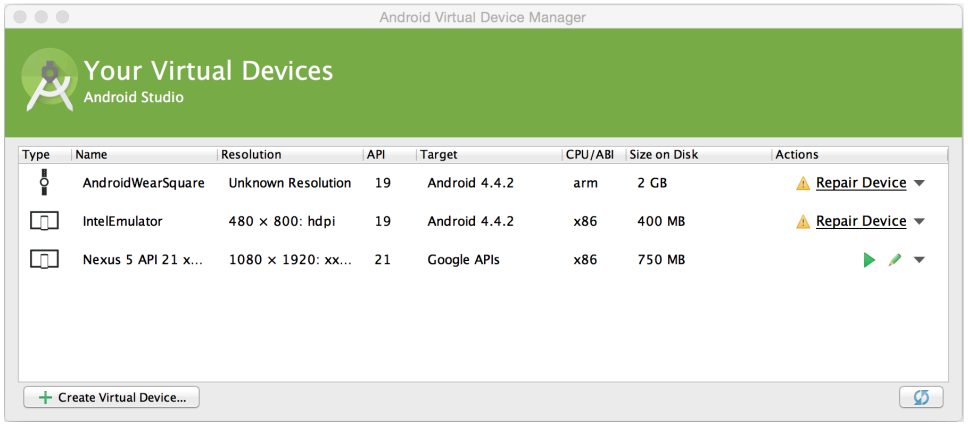


AVD Manager

Manages Virtual Devices (Emulators)

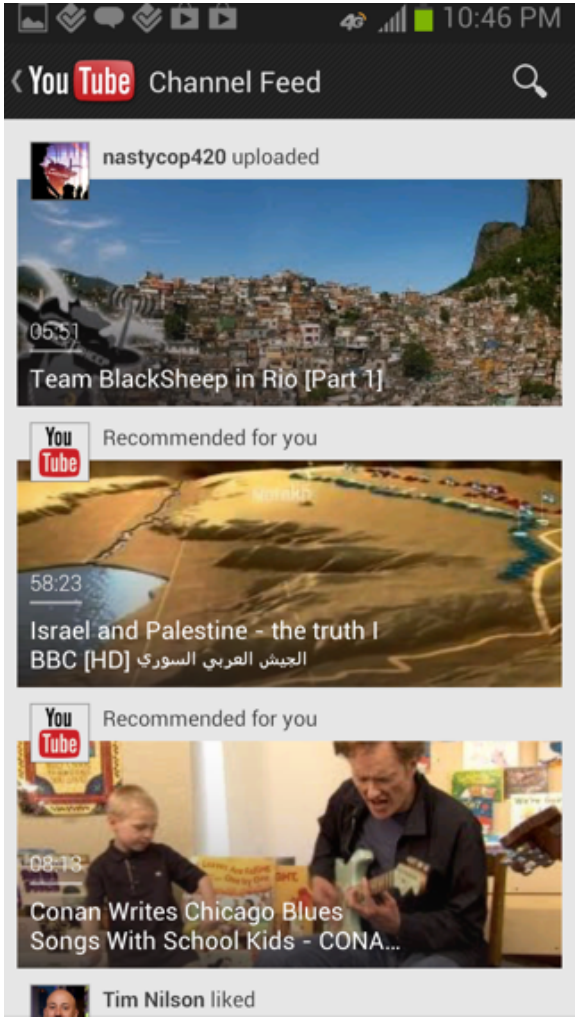
Fast Emulators: HAXM (Intel), GenyMotion



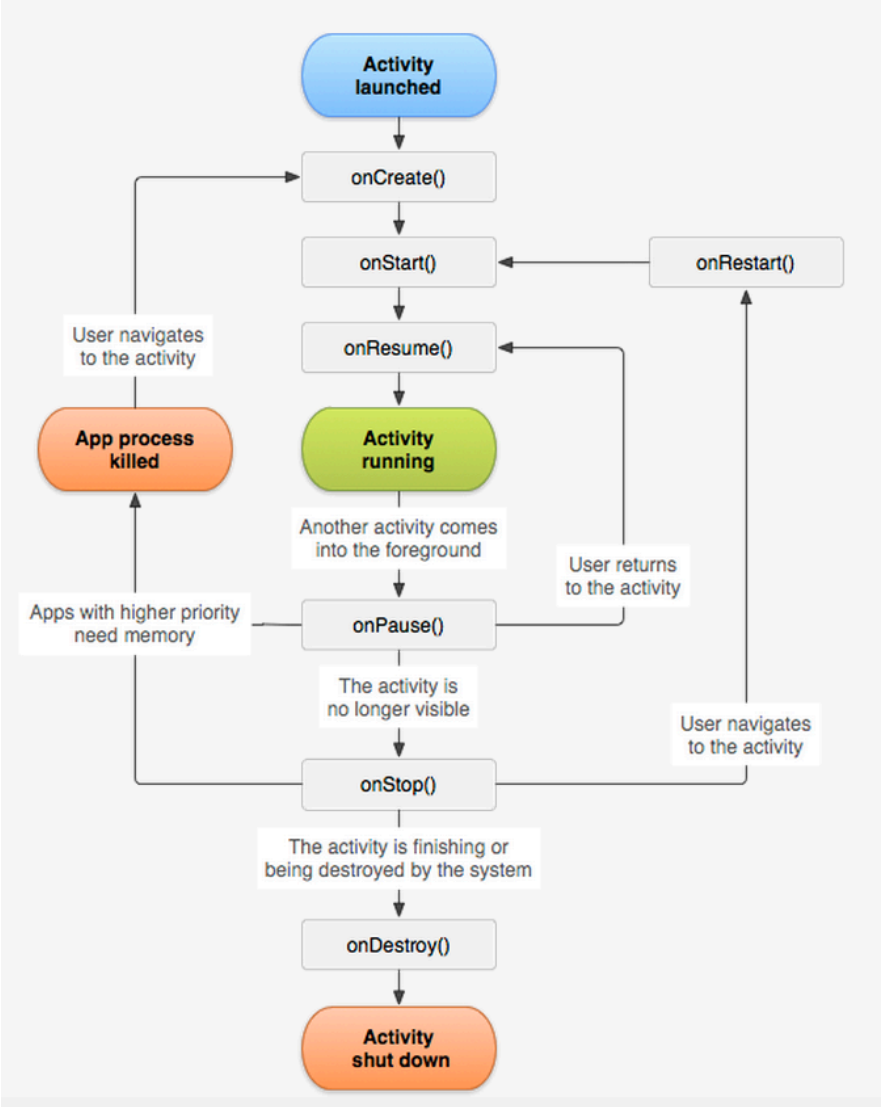


Activities

- Provide one screen in which to draw the user interface.
- An app usually consists of multiple activities.
- The main activity is the first screen presented when launching the application for the first time.



Activity LifeCycle

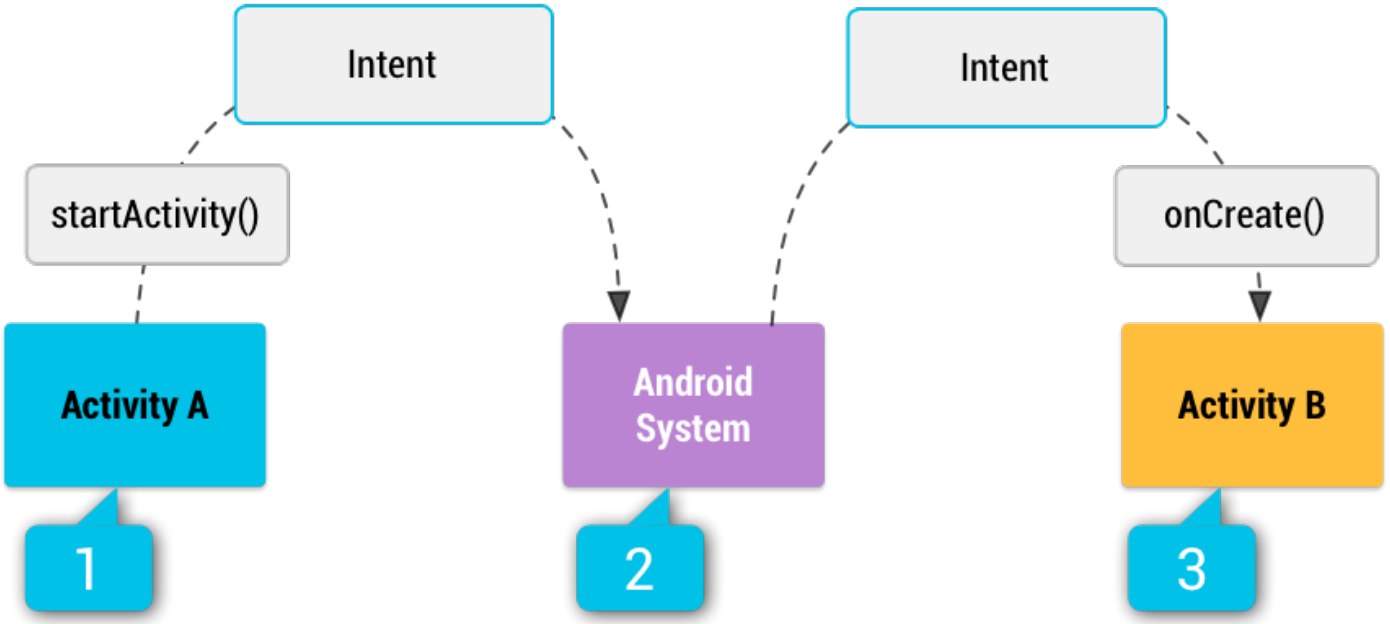


- OnCreate() – The activity has been created.
- OnStart() – The activity is running but not visible.
- OnResume() – The activity is visible.
- OnPause() – The activity is no longer visible
- OnStop() – The activity is being destroyed.

Intents

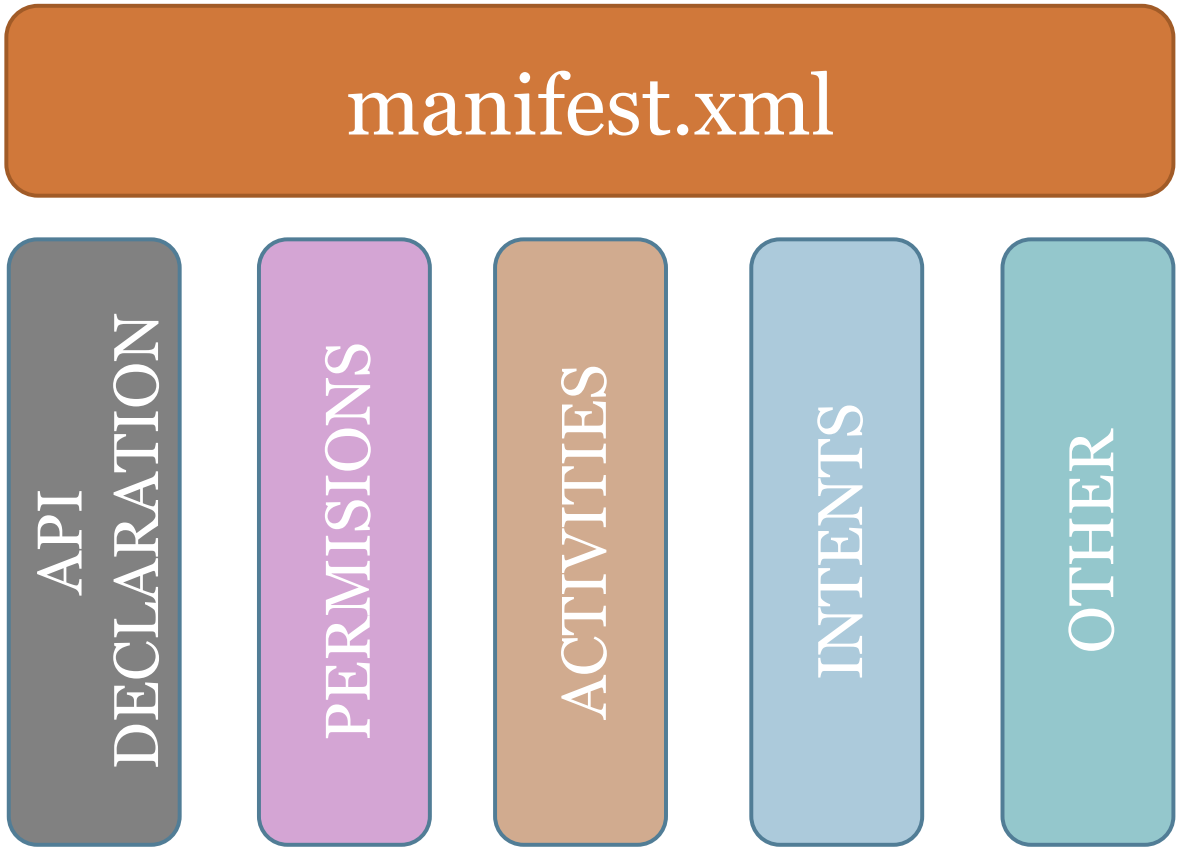
Intent is a messaging object that can be used to request an action from another component.

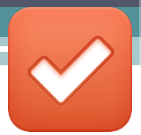
Usually an intent is used to launch a new activity (screen) within an app.



Android Manifest

The manifest file presents essential information about your app to the Android system





Let's Now Build a Todo App

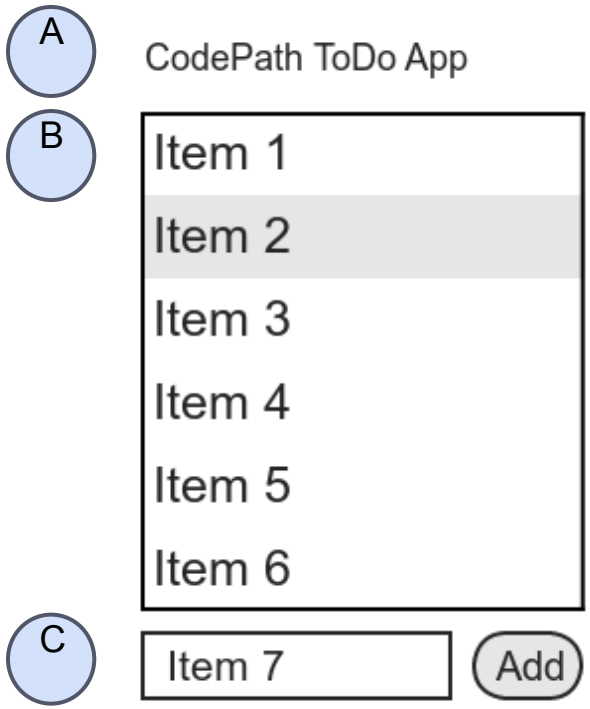
Main app features (use cases):

- **View** a list of existing todo items
- **Add** a new item to the todo list
- **Remove** an item from the todo list

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

Wireframe

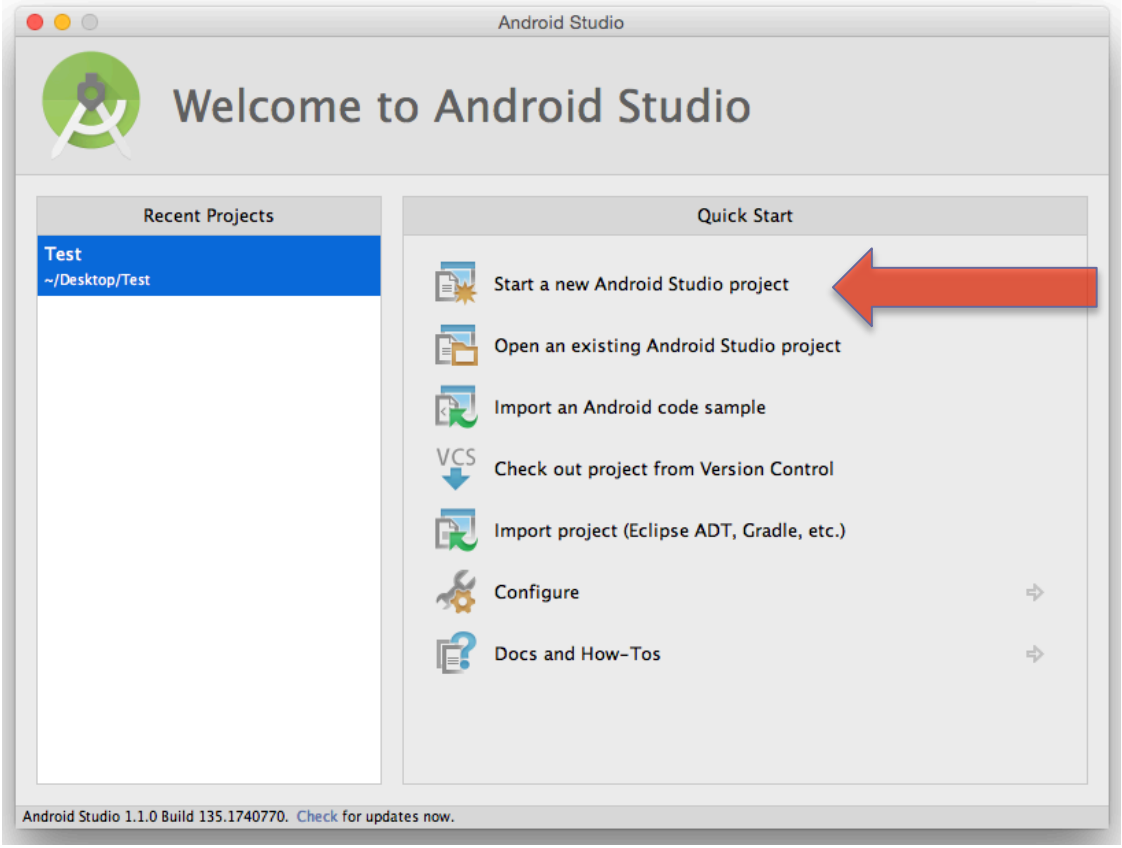
Let's wireframe our Todo App, to sketch the basic interface:



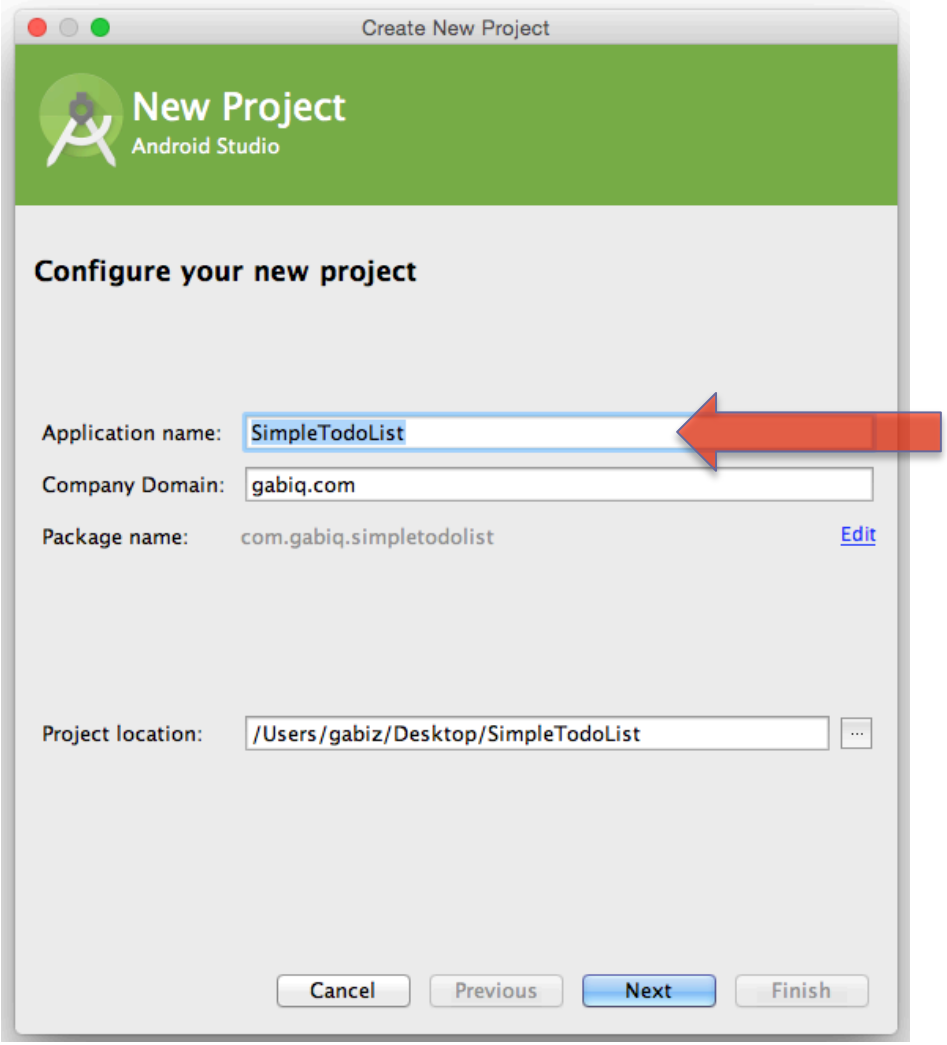
- (A) Basic Label with App Name
- (B) Basic List of Items
  - Vertically Scrollable
  - Hold Down to Remove Item
- (C) Adding with Textbox and Button

Creating a New Project

Let's create a new Android project in Android Studio

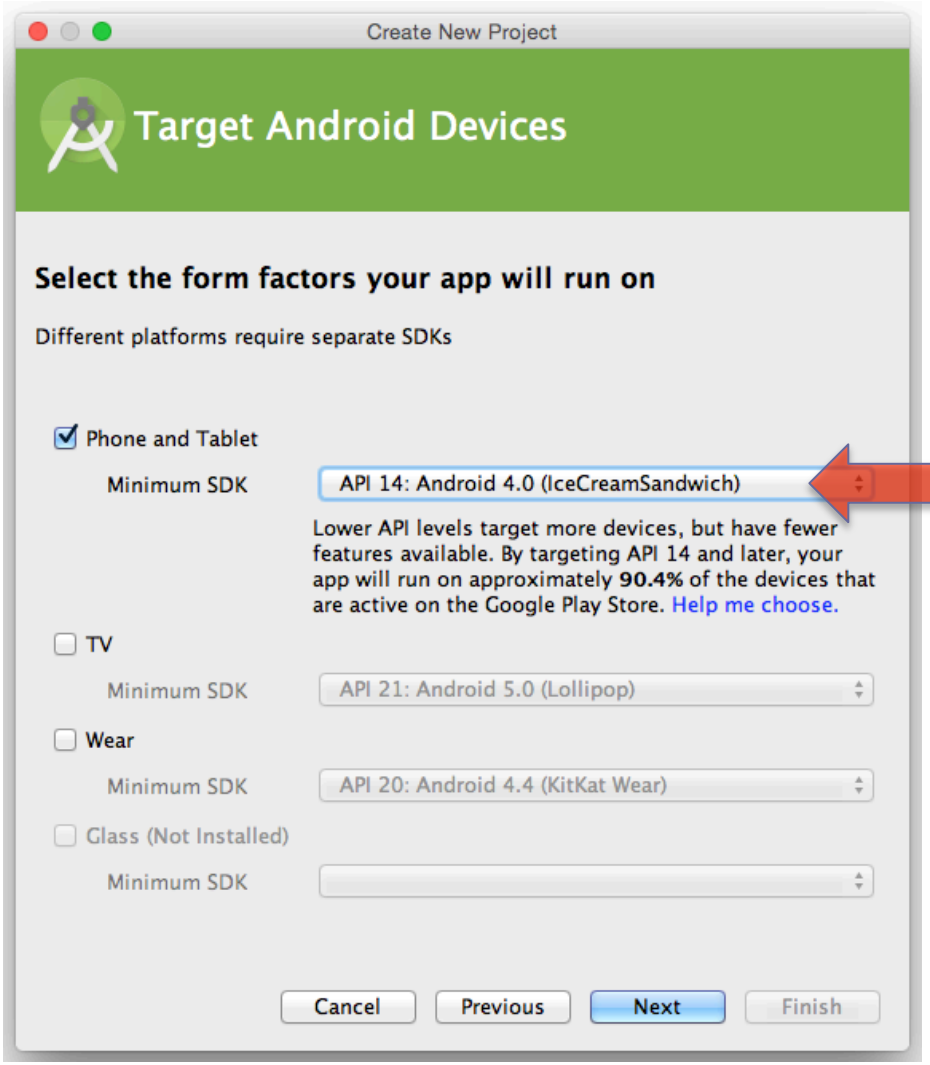


Creating a New Project (cont)



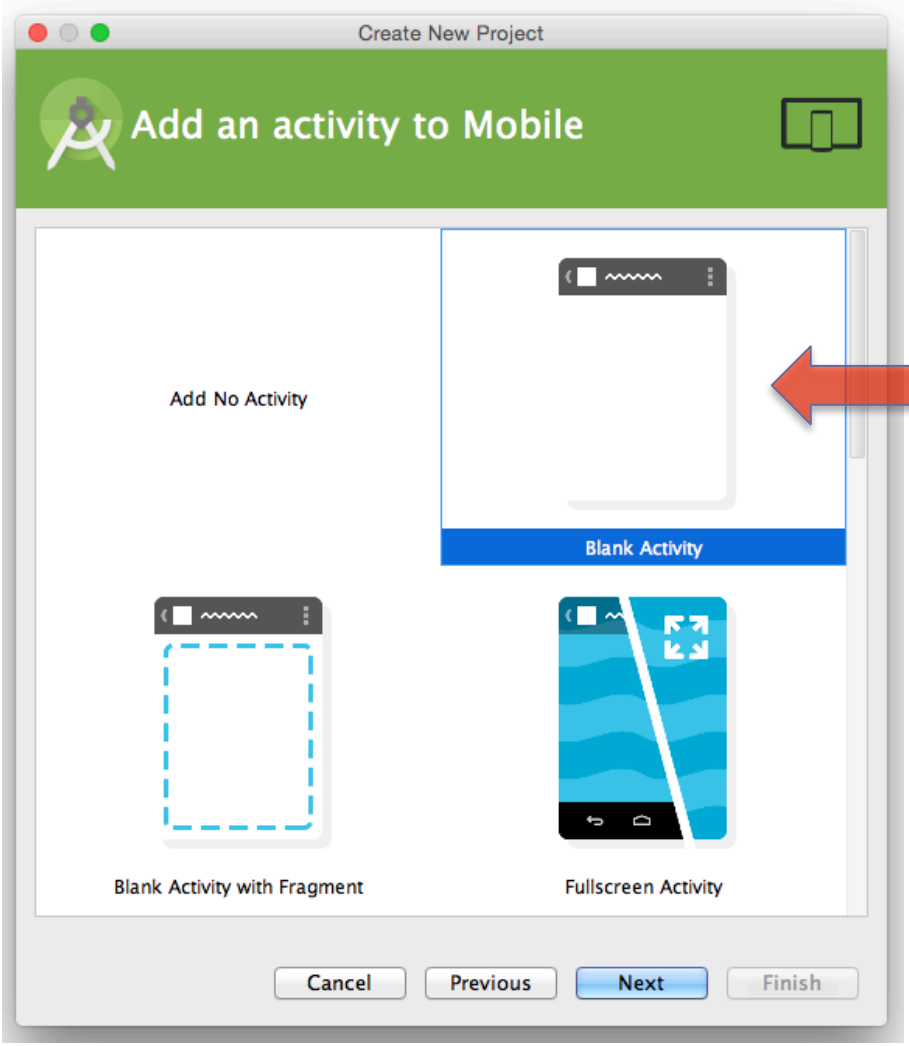
Choose an application name

Creating a New Project (cont)



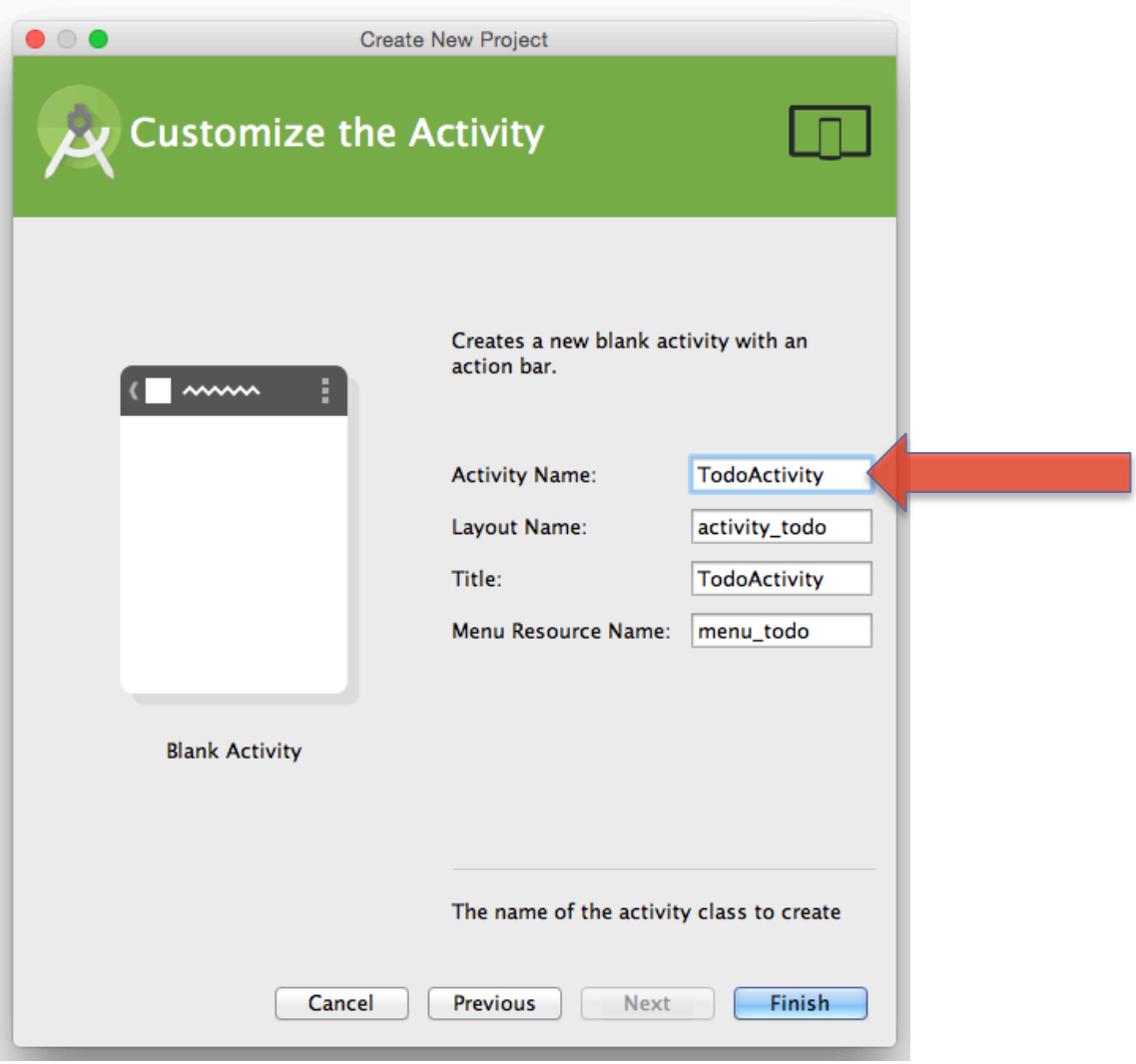
Choose API Level

Creating a New Project (cont)



Select Blank Activity

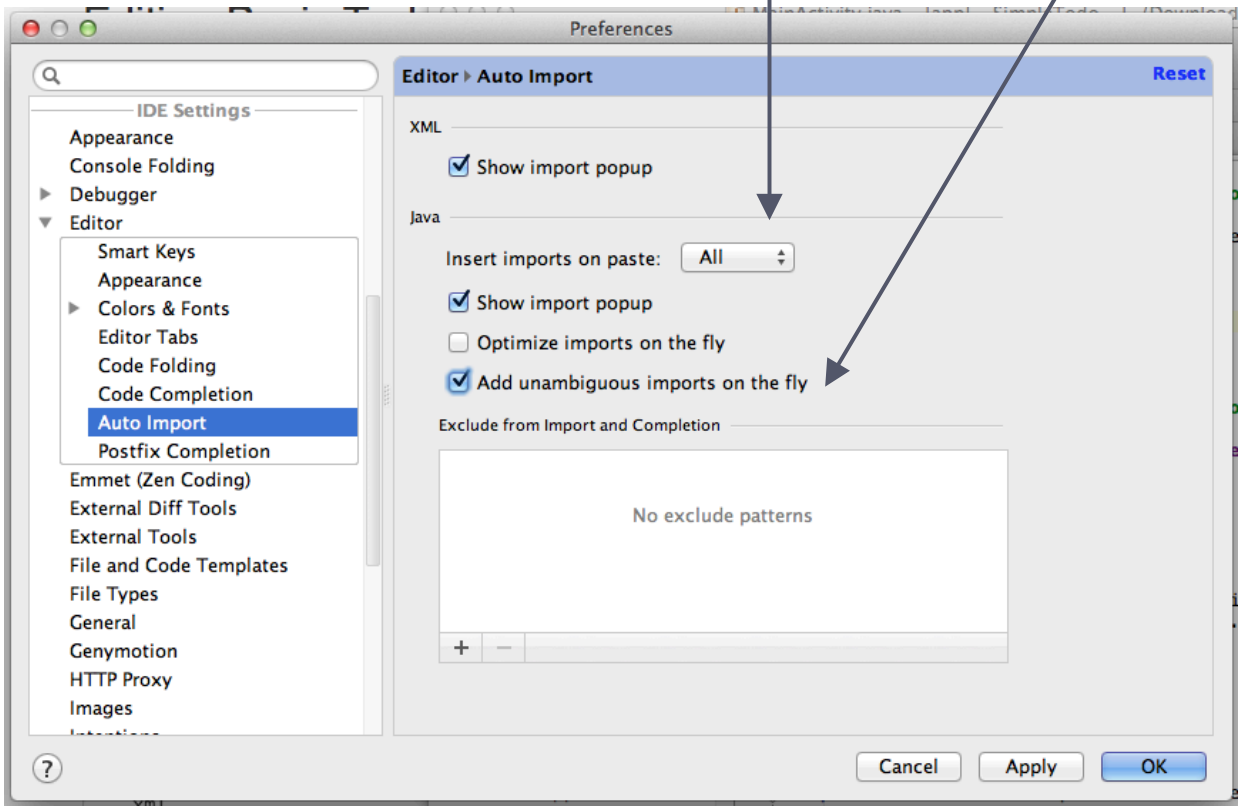
Creating a New Project (cont)



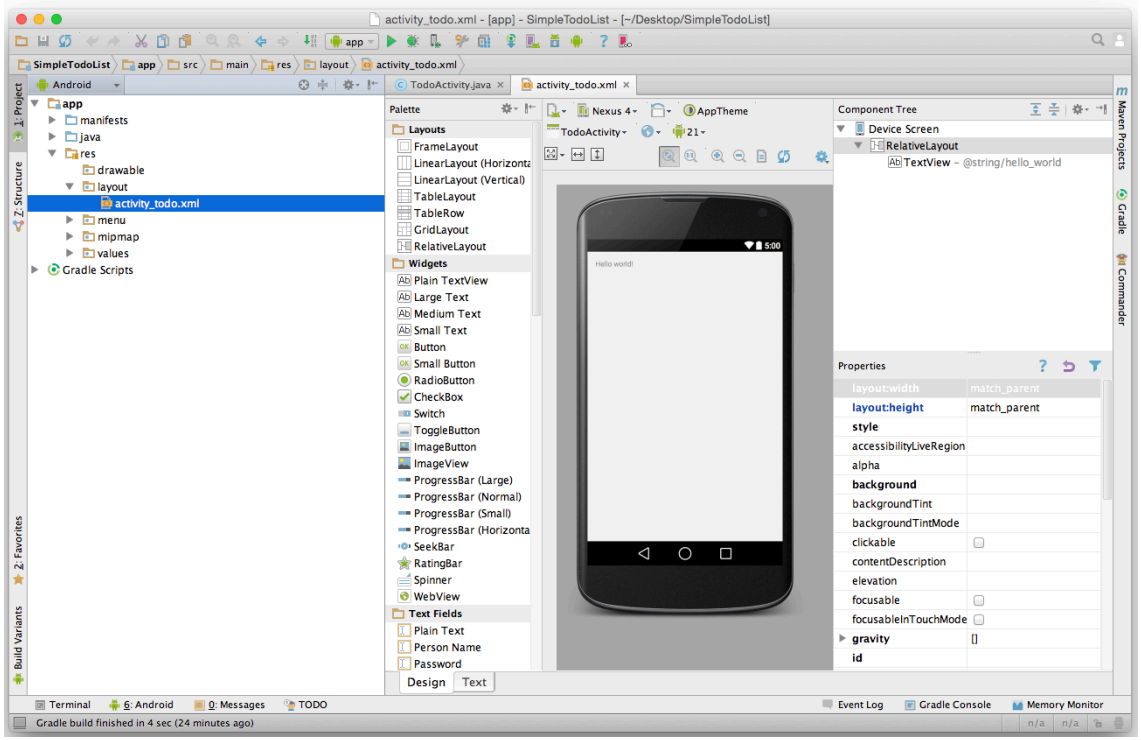
Choose Activity Name

Configure Auto Imports

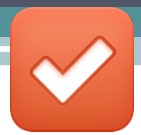
Set "**Imports on Paste**" to "**All**"
Check "**Add unambiguous imports on the fly**"



Let's Start Coding



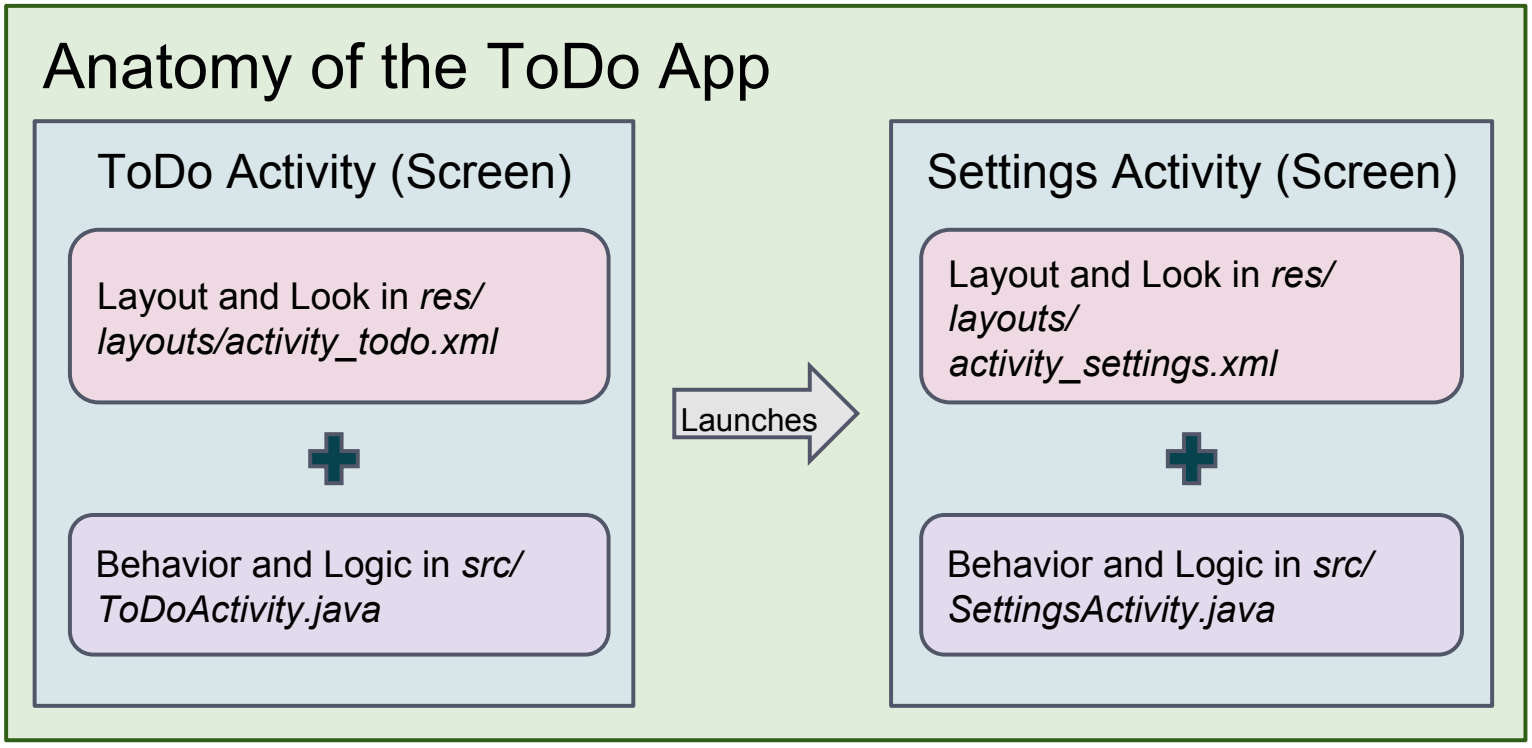
- Notice the first Activity is open for us by default
- We can start visually building our ToDo application right away
- Click on the second tab to reveal the XML file.



Android Pattern

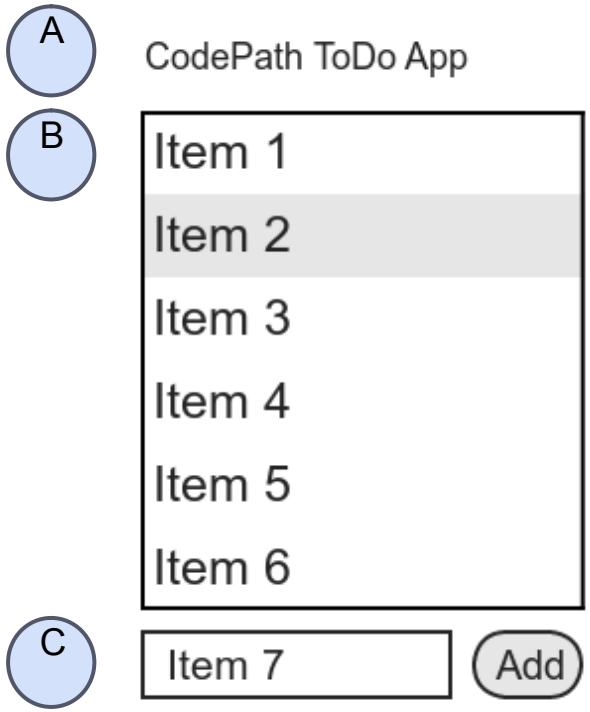
Activity == 1 UI Screen	**res/ layouts** == Look	**src/ *.java** == Behavior

Anatomy of the ToDo App



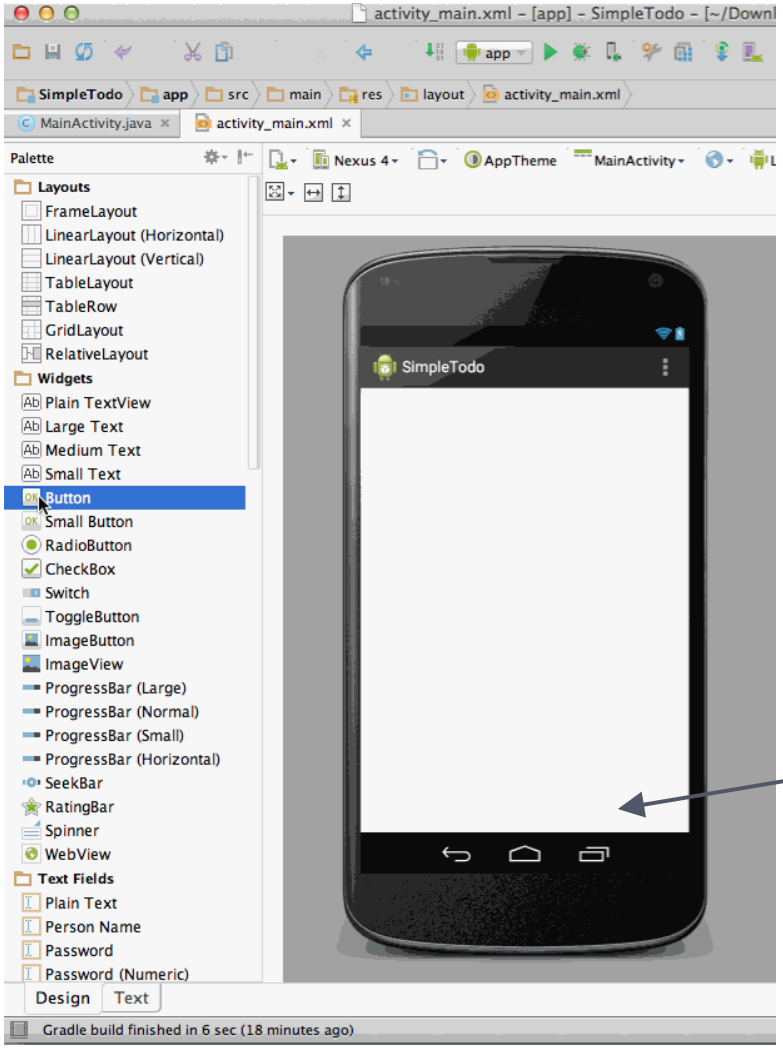
Building the Interface

Let's layout the interface for our application:



- (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 Interface (cont)

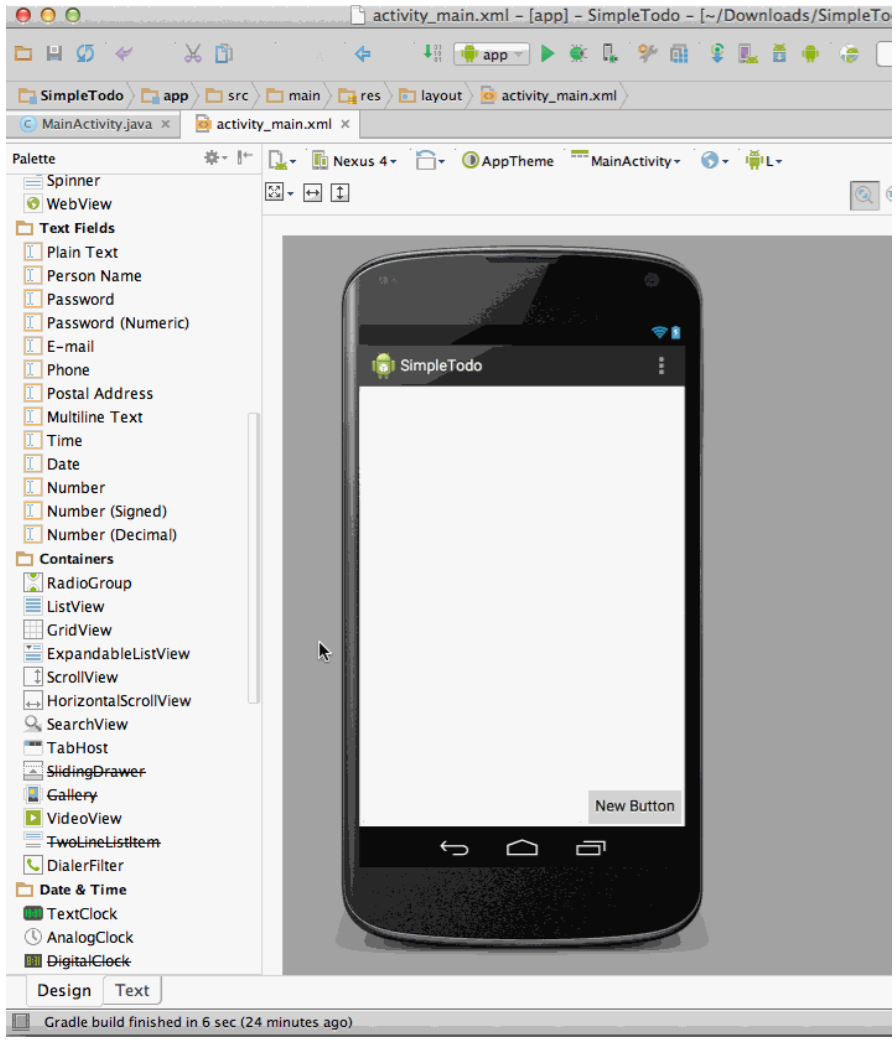


1. Click on “Hello World” label and press the backspace key to remove.

2. Drag a Button from left-hand side (Widgets) to Bottom-Right of Layout

Note: When dragging views onto layout, ensure **green lines** confirm location before releasing.

Building the Interface (cont)

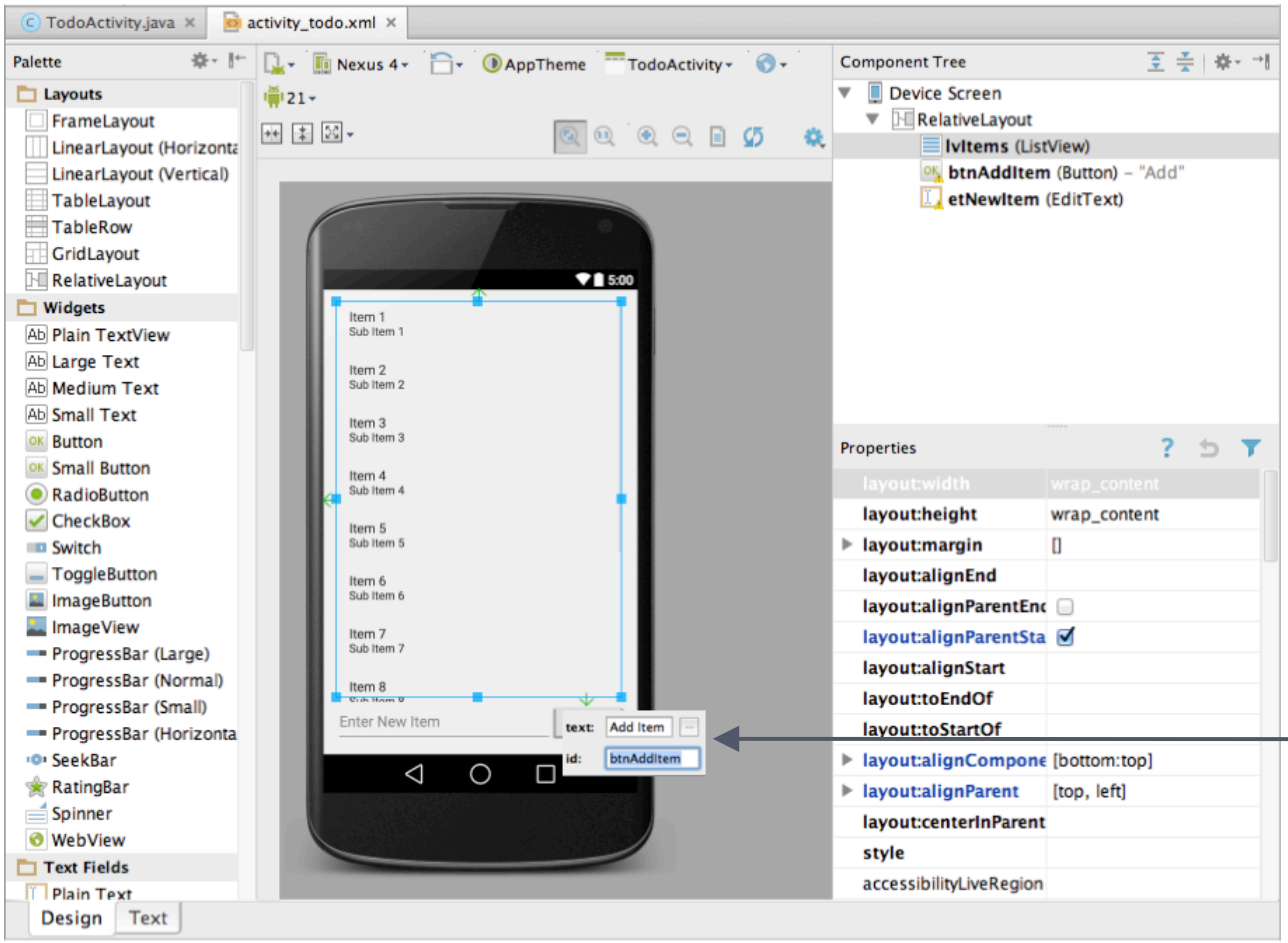


Drag More Views onto Layout

- 1. Text Fields → **Plain Text**
  - Drag to Bottom-Left
  - Resize Right Side to Button
- 2. Containers → **ListView**
  - Drag to Top Left Corner
  - Resize Bottom Above Button

Note: When dragging new views to layout, ensure **green lines** confirm location before releasing.

Building the Interface (cont)



- Assign Hint to EditText
- Double-click on view to assign ID
  - ListView = lvItems
  - EditText = etNewItem
  - Button = btnAddItem

XML is Generated Automatically

MainActivity.java × activity_main.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"
    tools:context=".MainActivity">

    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/lvItems"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_above="@+id/btnAddItem" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/etNewItem"
        android:layout_alignTop="@+id/btnAddItem"
        android:hint="Enter a new item"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_toLeftOf="@+id/btnAddItem"
        android:layout_toStartOf="@+id/btnAddItem"
        android:layout_alignParentBottom="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Add Item"
        android:id="@+id/btnAddItem"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />

</RelativeLayout>
```

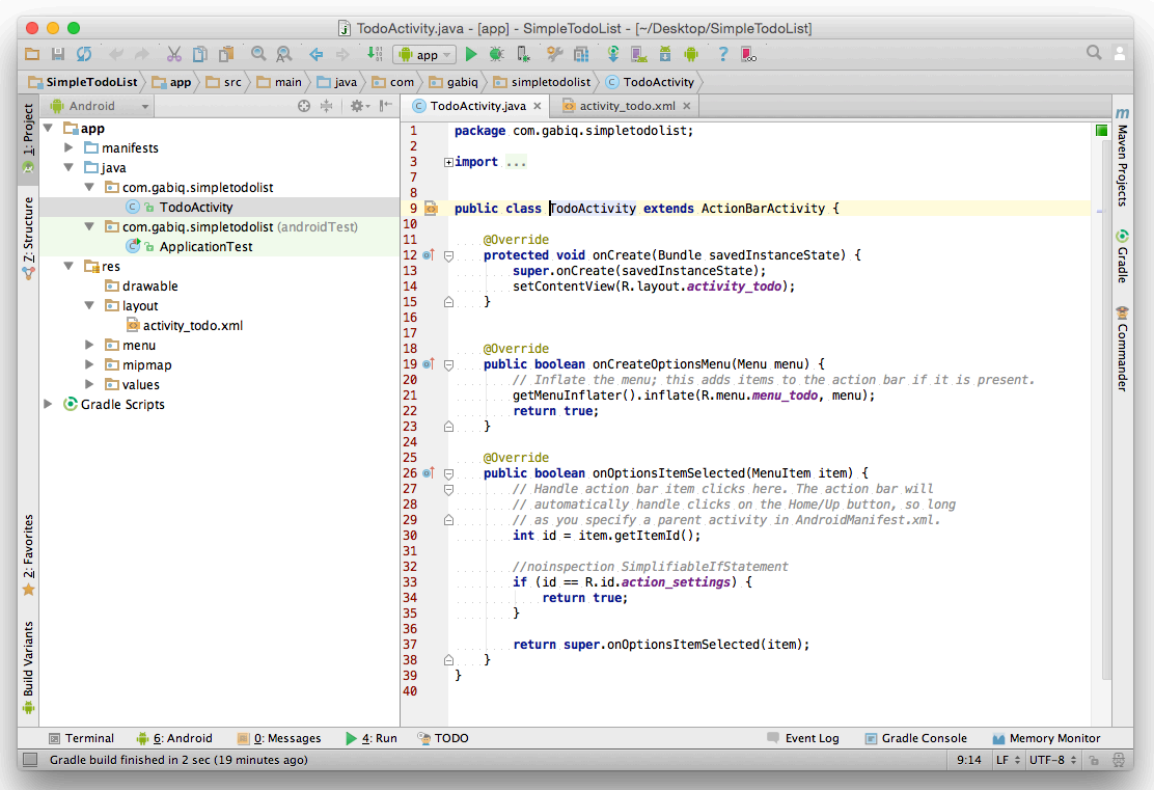
Design Text

- 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, etc)

Coding the Basic Behavior

Let's code the basic ToDo List behavior.

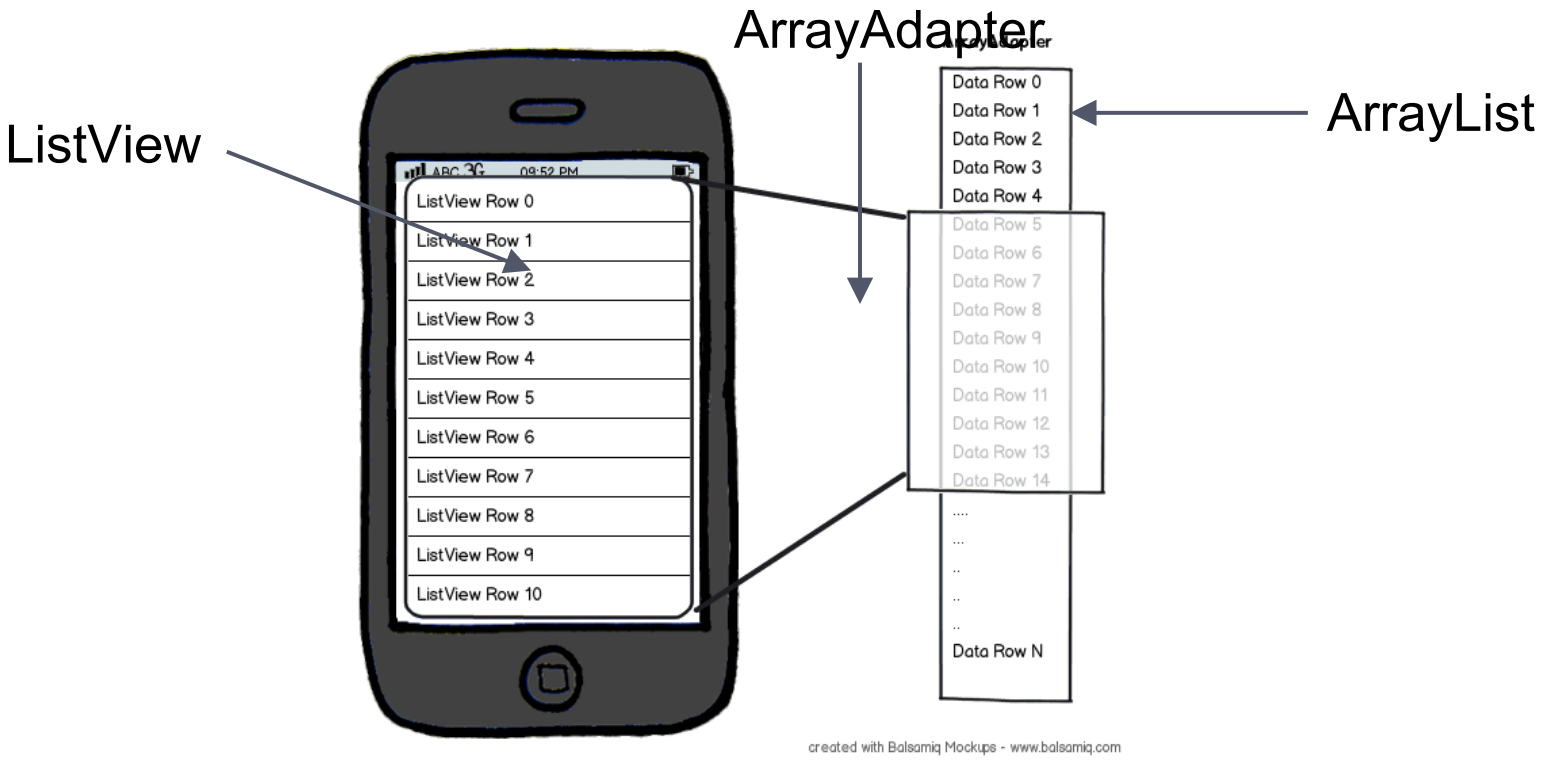
Add code in the Java source file for this activity:

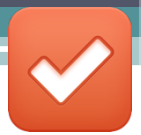


- `onCreate` is where the XML layout for this activity is applied
- Notice every Activity extends from the same base class
- This file is where we add our application logic

Adapters

Adapters connect a data source like an array or database to a UI Component and manage the display of a subset of the data.





Coding the Behavior (cont)

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

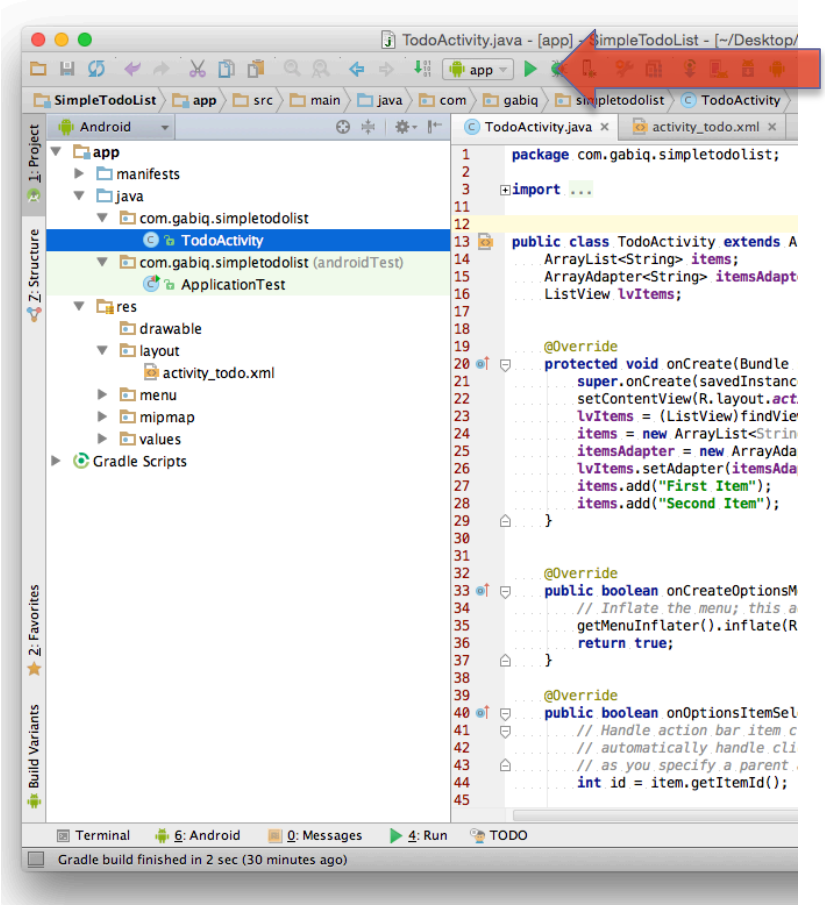
```
public class TodoActivity extends Activity {
    ArrayList<String> items;
    ArrayAdapter<String> itemsAdapter;
    ListView lvItems;

    @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");
    }
```

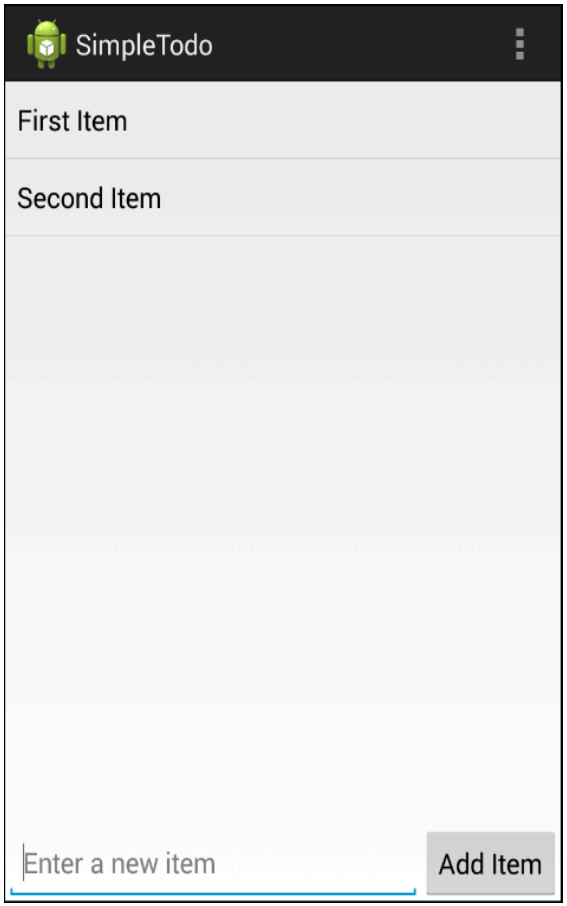
- Create an ArrayList
- Create an ArrayAdapter
- Get a handle to ListView
- Attach adapter to ListView

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

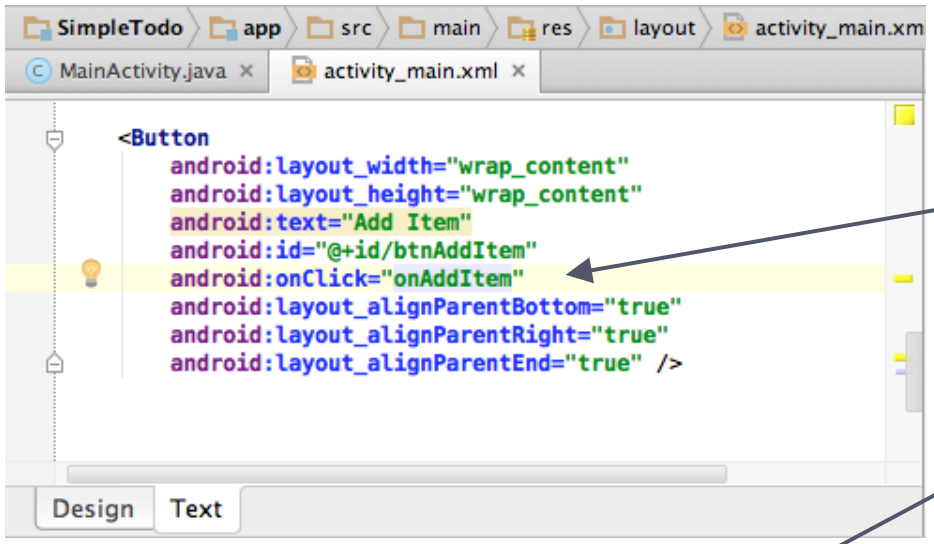
Testing the Behavior

Let's run the app in our emulator:

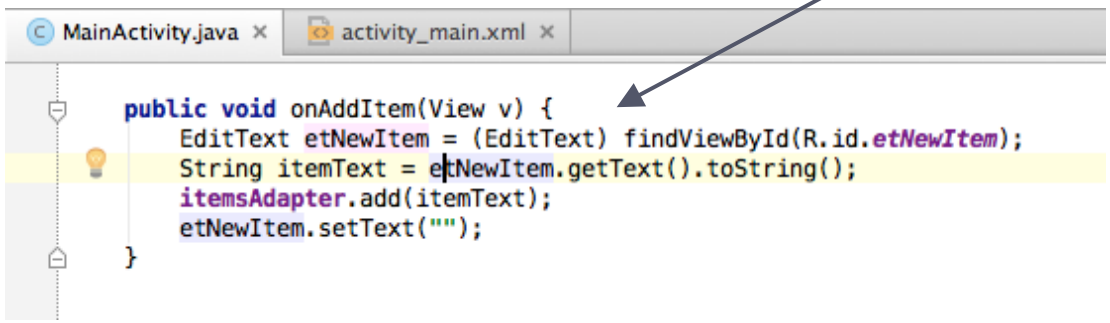




Adding Items to the List



Add "onClick" property to Button in XML



Define ”onAddItem" method to activity which adds input item to the list.

Remove Items from the List

```
@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<?> parent,
                View view, int position, long rowId) {
            items.remove(position);
            itemsAdapter.notifyDataSetChanged();
            return true;
        }
    });
}
```

Create a new method for setting up the listener.

Invoke the listener from onCreate().

Attach a "LongClickListener" to each Item for ListView:

- Removes that item
- Refreshes the adapter.

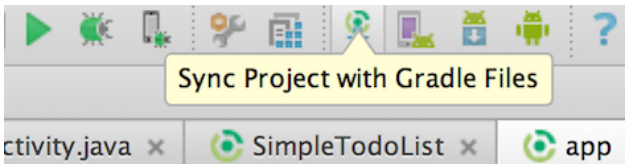
Loading and Saving Files

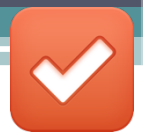
We need the Commons.io Library in order to read and write the items array from/to a file.

Open file build.gradle and add a compile directive for commons.io to the dependencies section.

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:21.0.3'
    compile 'commons-io:commons-io:2.4'
}
```

Then select "Sync Project with Gradle Files" to sync your project.





Loading and Saving Files (cont)

Add support for 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();
    }
}
```

Create method to open a file and read a newline-delimited list of items

Create method to save a file and write a newline-delimited list of items

Note: FileUtils is in a library called **Apache Commons IO** that we previously added.

Loading and Saving Files (cont)

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    // ...super, setContentView, define lvItems
    readItems();
    itemsAdapter = new ArrayAdapter<String>(getBaseContext(),
            android.R.layout.simple_list_item_1, items);
    // ...rest of the code
}
```

Load the items onCreate

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

Save items when a new list item is added

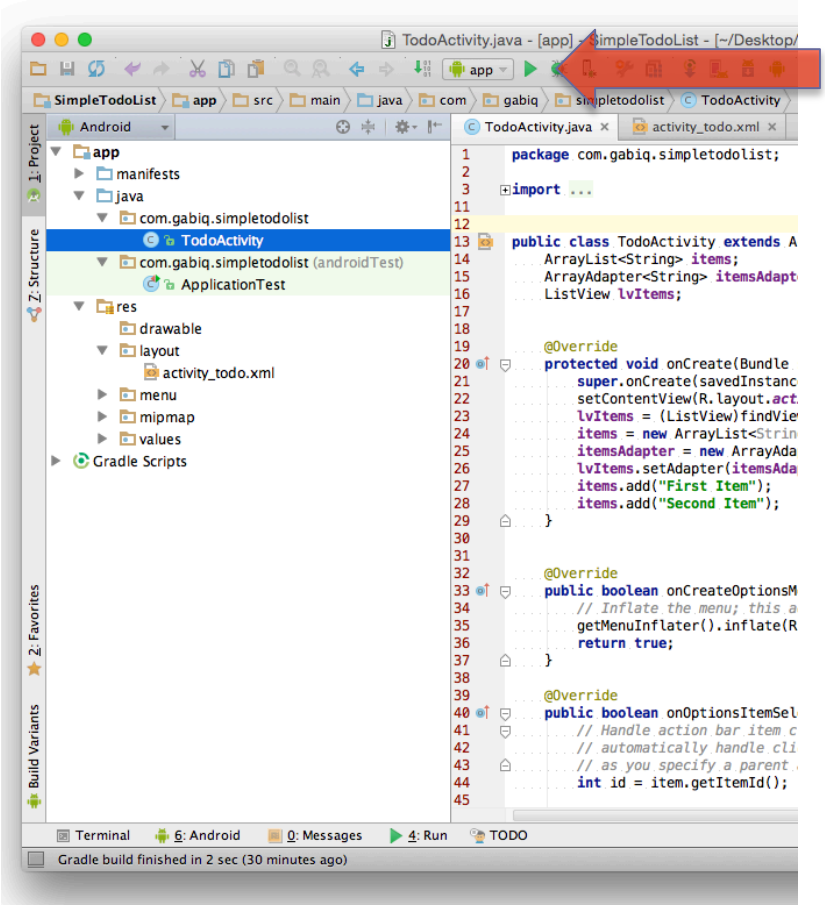
```
lvItems.setOnItemLongClickListener(new OnItemLongClickListener() {
    @Override
    public boolean onItemLongClick(AdapterView<?> parent,
            View view, int position, long rowId) {
        items.remove(position);
        itemsAdapter.notifyDataSetChanged();
        saveItems();
        return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

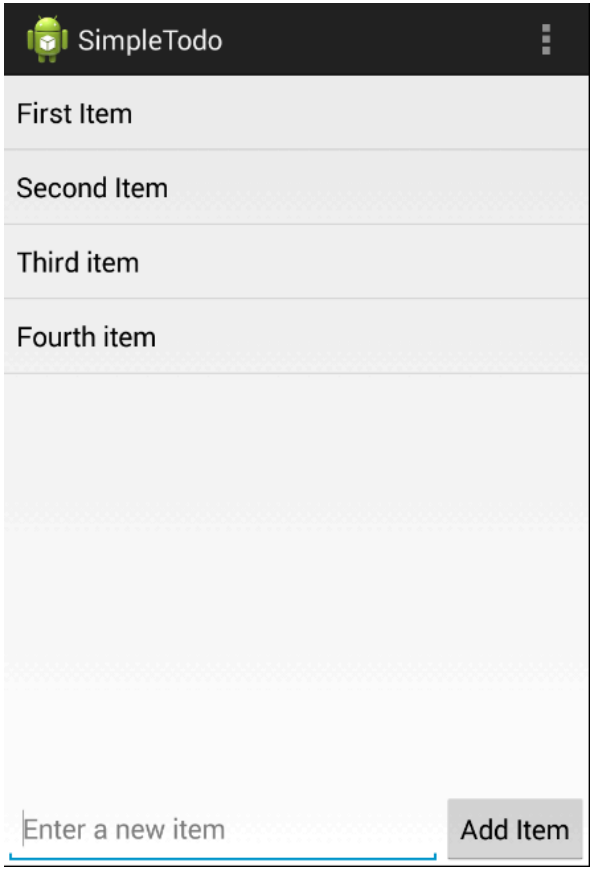
Save the items when a list item is removed

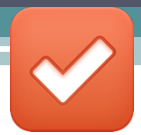
Remove “new Array<String>” from onCreate and the test items.

Testing the Behavior

Let's run the app again in our emulator. If we add 4 items and rerun the app then it should show the 4 items we previously added.



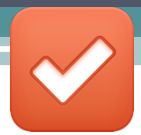




ToDo App Summary

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 Now?

Andriod Developer Portal
http://developer.android.com

Play with Android Samples
http://developer.android.com/samples/index.html

Vogella Android Tutorials:
http://www.vogella.com/tutorials/android.html

CodePath CliffNotes
http://guides.codepath.com/android

Apply to a CodePath bootcamp.