

LabW03 – Handling Interactions

Objectives:

1. Use click event listener for ListView
2. Use UI element dialog
3. Use Activity navigation
4. Experience Cross-platform app development [Optional]

Tasks:

1. Handle long click event for an item in Listview
2. Pop up a dialog
3. Handle item clicking in Listview with Intent and another Activity
4. Develop Cross-platform app with Xamarin [Optional]

You must finish LabW02 before proceed to the following tasks.

Task 1: Handling item long clicking in Listview

Currently nothing happens when an item is clicked in the Listview. We will first implement “deleting an item” by long clicking it.

1. Add the following method into the MainActivity.

It sets two listeners for LongClick and Click events.

```
private void setupListViewListener() {
    listView.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {
        public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long rowId)
        {
            Log.i("MainActivity", "Long Clicked item " + position);
            return true;
        }
    });
    listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView <? > parent, View view,
                                int position, long id) {
            String updateItem = (String) itemsAdapter.getItem(position);
            Log.i("MainActivity", "Clicked item " + position + ": " + updateItem);
        }
    });
}
```

2. Call this method at the end of the onCreate() method

```
@Override
protected void onCreate(Bundle savedInstanceState) {

    XXXXXX
    XXXXXX

    setupListViewListener();
}
```

3. Run it. Click and long click an TodoItem in the list view.
4. Now update the listener for onItemLongClickListener.

It adds two lines of code to remove an item from the ArrayList, and notify the Listview adapter to update the list. Run it.

```
Log.i("MainActivity", "Long Clicked item " + position);
items.remove(position);
itemsAdapter.notifyDataSetChanged();
return true;
```

Run your code, and long click a TodoItem, you should be able to see the following lines are printed on the Android Studio logcat:

```
comp5216.sydney.edu.au.todolist I/MainActivity: Long Clicked item 1
comp5216.sydney.edu.au.todolist I/MainActivity: Long Clicked item 2
```

Task 2: Popping up a dialog

1. Add a dialog to let user confirm the delete operation. Replace the existing OnItemLongClick method.

```
listview.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    public boolean onItemClick(AdapterView<?> parent, View view, final int position, long rowId) {
        Log.i("MainActivity", "Long Clicked item " + position);
        AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
        builder.setTitle(R.string.dialog_delete_title)
            .setMessage(R.string.dialog_delete_msg)
            .setPositiveButton(R.string.delete, new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
                    //delete the item
                    items.remove(position);
                    itemsAdapter.notifyDataSetChanged();
                }
            })
            .setNegativeButton(R.string.cancel, new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
                    //User cancelled the dialog
                    //nothing happens
                }
            });

        builder.create().show();
        return true;
    }
});
```

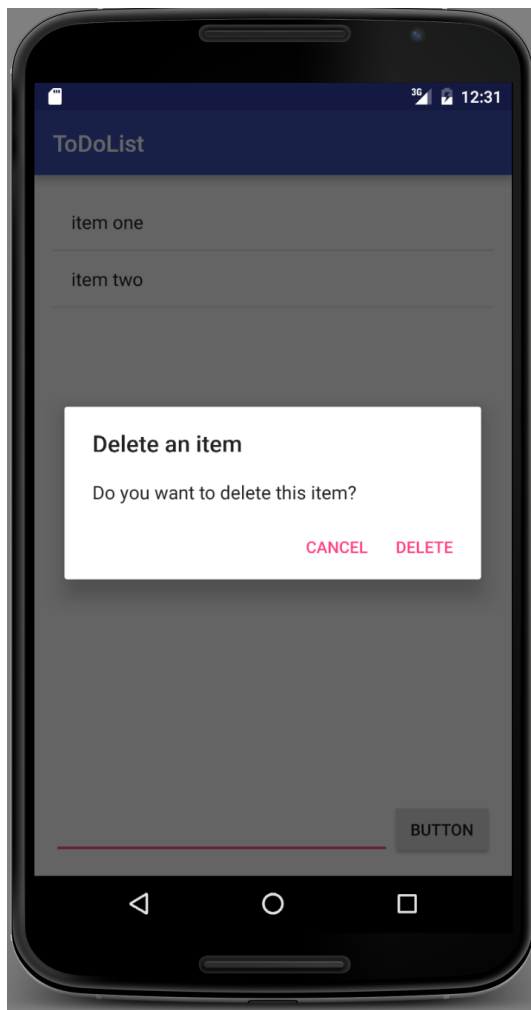
You will also need to add the following String resources into “res/values/strings.xml”

```
<string name="dialog_delete_title">Delete an item</string>
<string name="dialog_delete_msg">Do you want to delete this item?</string>
<string name="delete">Delete</string>
<string name="cancel">Cancel</string>
<string name="edit">Edit</string>
```

Run it and long click an item.

2. In order to further customise a dialog, such as adding a third button, or using your own layout, please refer to the following tutorial:
<http://developer.android.com/guide/topics/ui/dialogs.html>

If you do the above steps right, you should be able to see the following screenshot once you long click a TodoItem.



Task 3: Handling item clicking in Listview with Intent and another Activity

When clicking an item (NOT long click), we should open another Activity to edit the item.

1. Copy the `EditToDoItemActivity.java` to your “**src**” folder next to the `MainActivity`.

This Activity receives the data from the `MainActivity` when an item is clicked, and send back the updated content to the `MainActivity`. Read the inline comments for details.

2. Copy **activity_edit_item.xml** to the “**res/layout**” folder. This is the layout for `EditToDoItemActivity`

3. Update the onItemClick method in MainActivity to open the EditToDoItemActivity

```
@Override

public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
    String updateItem = (String) itemsAdapter.getItem(position);
    Log.i("MainActivity", "Clicked item " + position + ": " + updateItem);

    Intent intent = new Intent(MainActivity.this, EditToDoItemActivity.class);
    if (intent != null) {
        // put "extras" into the bundle for access in the edit activity
        intent.putExtra("item", updateItem);
        intent.putExtra("position", position);
        // brings up the second activity
        startActivityForResult(intent, EDIT_ITEM_REQUEST_CODE);
        itemsAdapter.notifyDataSetChanged();
    }
}
```

When an item is clicked, it creates an Intent to start another activity and wait for its result. You need to define a variable to remember the request code in the MainActivity

```
public final int EDIT_ITEM_REQUEST_CODE = 647;
```

4. When the EditToDoItemActivity finish, MainActivity will receive the result by checking the request code first, and then uses its result.

Add the following method into MainActivity

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == EDIT_ITEM_REQUEST_CODE) {
        if (resultCode == RESULT_OK) {
            // Extract name value from result extras
            String editedItem = data.getExtras().getString("item");
            int position = data.getIntExtra("position", -1);
            items.set(position, editedItem);
            Log.i("Updated Item in list:", editedItem + ",position:" + position);

            Toast.makeText(this, "updated:" + editedItem, Toast.LENGTH_SHORT).show();
            itemsAdapter.notifyDataSetChanged();
        }
    }
}
```

To learn more about using intent, please read:

https://github.com/thecodepath/android_guides/wiki/Using-Intents-to-Create-Flows

You may also notice the use of Toast. It is a simple popup notification method in Android. Learn more at:

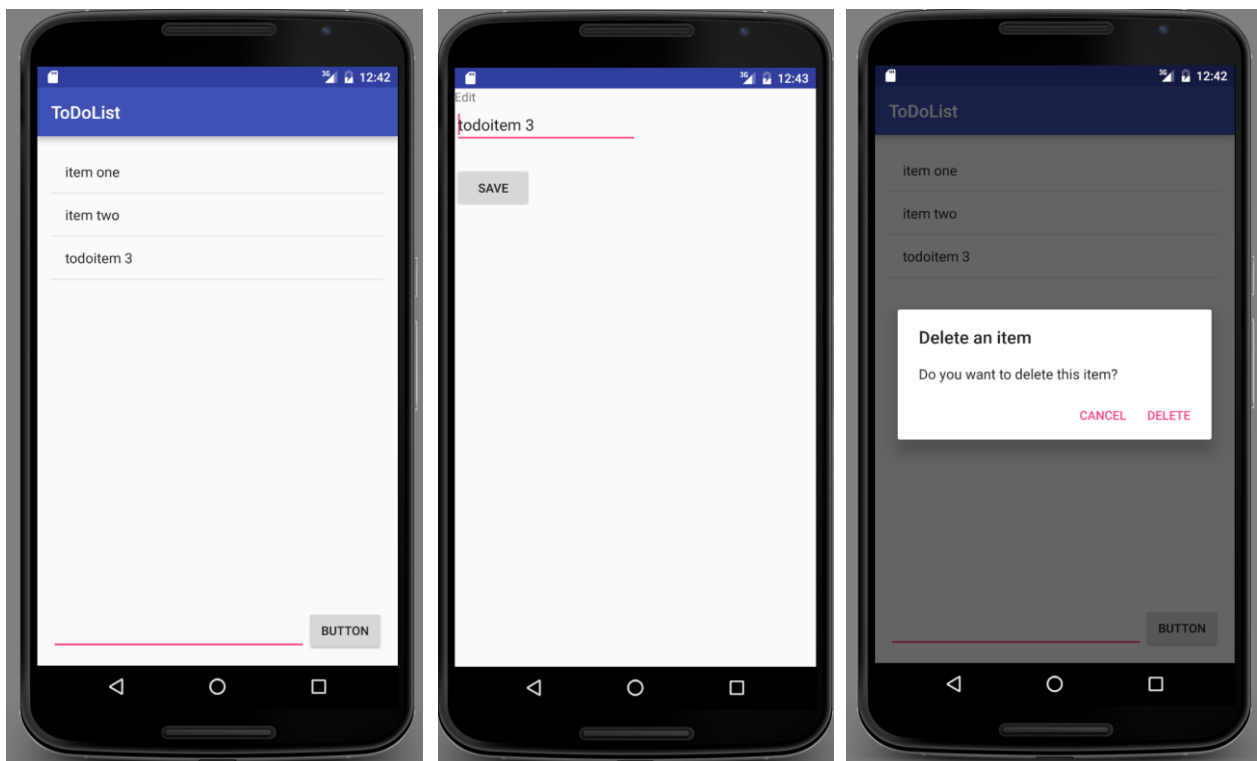
<http://developer.android.com/guide/topics/ui/notifiers/toasts.html>

5. Lastly, the application manifest must include the new `EditToDoItemActivity`, otherwise this Activity cannot be used.

Open **AndroidManifest.xml**, and add the following code inside the `<Application>` tag.

```
<activity
    android:name=".EditToDoItemActivity"
    android:label="@string/app_name" >
</activity>
```

6. Run it. Test the above steps. If all the steps are correct, your app now are able to handle the click and long click events:



Task 4 [Optional]: Using Xamarin Studio

Xamarin is a tool for cross-platform app development. It is free for students at: <https://xamarin.com/student>

In Windows platform, Xamarin has been part of the latest Visual Studio. In Mac platform, Xamarin Studio is the development environment.

- 1) Windows visual studio for Xamarin:
https://developer.xamarin.com/guides/android/getting_started/hello_android/

2) Xamarin Studio in Mac:

https://developer.xamarin.com/guides/mac/getting_started/hello_mac/

Please refer to the following URL for more details on Xamarin:

<http://developer.xamarin.com/>