Exercise 3: Intents

Due: Mar. 5, 23:59 PDT

Overview

This exercise demonstrates how to:

- Communicate between different activities of the same app using an explicit Intent.
- Set up 2 apps in one Android Studio project.
- Communicate between different apps using an implicit Intent.
- Set an app chooser when sending the implicit Intent.

There will be 2 different apps (app & mybrowser), where the first app is your "main" app that contains 2 activities, MainActivity and TextActivity. The MainActivity has two buttons:

- "Get Text" button. Click it will send an explicit Intent to TextActivity for getting a text. On TextActivity, if the user clicks "Confirm", it will return to the MainActivity with a text set to your name (Fig. 1); if the user clicks "Cancel", the text will be set to "cancelled".

Fig. 1 Click Get Text to go to the TextActivity and Confirm to set the name

- "Web browser" button. Click it will send an implicit Intent with a view action/request. An app chooser will show all available apps that can handle this request and mybrowser is one of them. When the user chooses mybrowser, mybrowser will start and show the website URL (amazon.com) that the first app passed in. Click "Return" on mybrowser will return to the MainActivity of the first app with a text set to "Website viewed" (Fig. 2).

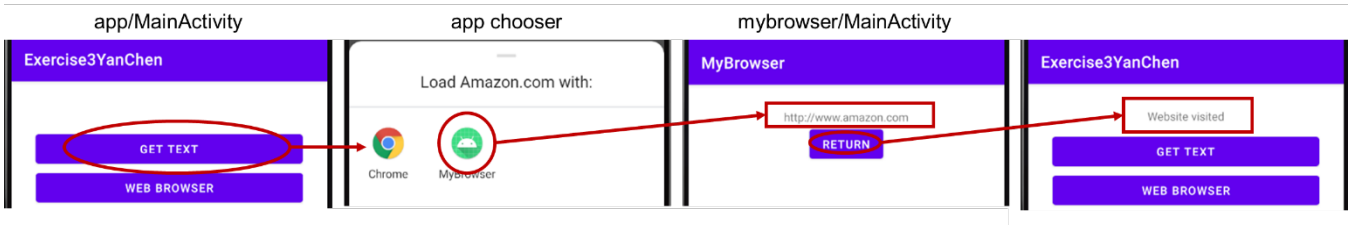

Fig. 2 Click Web Browser for a view request

Prerequisites:

- Know the process of submitting your work (exercise 0)
- Be familiar with previous topics related to project structure and set up UI (exercise 1 - 2, lesson 2 - 4)
- Has set up an emulator (API 29)

Procedure related to the above topics will not be provided in the instruction. Refer to corresponding exercise/lecture notes if needed.

If you set any different view id's, filenames, etc., remember to modify the corresponding part of code.

Set the project name as "Exercise3YourName"

Step 0. Add a new module

To add another app in the same project, simply click File -> New -> New Module (Fig. 3).

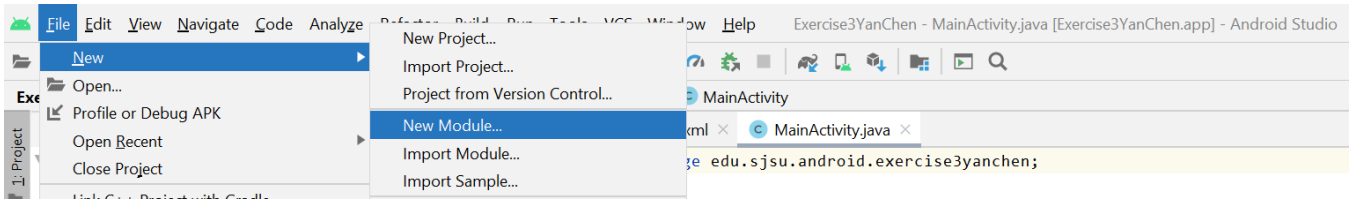

Fig. 3 Add a new module

Choose "Phone & Table Module". Set the Application/Library name as "My Browser". Keep other settings. Finally, choose "Empty Activity", next, finish (you don't need to change the Activity Name).

Now you have 2 apps in one project. You can choose which app to run as shown in Fig. 4.

Note that you have 2 module build.gradle files now (one for each app). We won't deal with many widgets, so the instructions don't use view binding. If you plan to use view binding, you need to set it in both files.

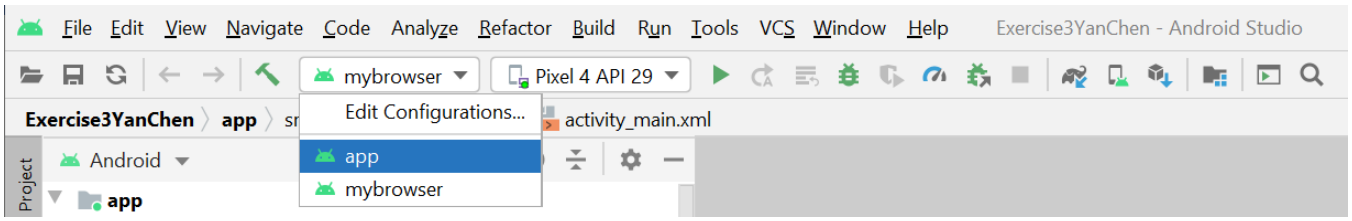

Fig. 4 Choose which app to run

Step 1. Set up UI files

1.1 Create Another Activity for the First App

The first app includes 2 activities. Besides MainActivity.java, create a new Activity by right clicking the Java package, choose New -> Activity -> Empty Activity and name the new Activity as "TextActivity". In this way, a layout file will be generated for you, also the new Activity will be registered in manifest.

1.2 Set up UI

The positions of the widgets do not matter but you should include the elements listed on next page.

Note that when you set the onClick attribute, the IDE will complain because we haven't implemented the methods yet. After finishing the remaining steps, the error will be gone.

App/Activity	Widget	id	Text	onClick
app/MainActivity	TextView	result	Will set later	n/a
	Button	explicit	“Get Text”	choose
	Button	implicit	“Web Browser”	choose
app/TextActivity	Button	confirm	“Confirm”	setText
	Button	cancel	“Cancel”	setText
mybrowser/MainActivity	TextView	website	Will set later	n/a
	Button	back	“Return”	getBack

Since after sending out the Intents, we also want to get some data back, it may be easier to implement those components that receive the Intents first, so you don’t need to switch back and forth between different .java files.

Step 2. Implement mybrowser app to respond to the implicit Intent

Let’s first implement mybrowser app. All files you need to modify are under mybrowser module.

4.1 Set Intent Filter

You need to register the app for certain implicit Intents by setting intent filter in AndroidManifest.xml file. According to the official guide, “... If you want your activity to receive implicit intents, it must include a category for "android.intent.category.DEFAULT" in its intent filters...”. So, we need to add that.

Also, we want the browser app to respond to a “view” action, specifically, to view a webpage. So, we also need to register for a view action, a browsable category, and two data schemes (http/https).

```
<application
    ...>
    <activity android:name=".MainActivity">
        <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <action android:name="android.intent.action.VIEW" />

        <category android:name="android.intent.category.LAUNCHER" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />


        <data android:scheme="http" />
        <data android:scheme="https" />
        </intent-filter>
    </activity>
</application>
```

4.2 Get Intent and Data

The app gets the Intent and the data once it’s created, so the code for this sub-step is in the onCreate method. Since the data passed in (URL for a website) should be a string, so we can use getDataString method. And then set the TextView to display the website URL.

```
String website = getIntent().getDataString();
TextView text = findViewById(R.id.website);
text.setText(website);
```

4.3 Return to the First App

When user clicks the Return button, return back to the first app. So, let's implement the "getBack" method we've set that is executed for an onClick event. We don't need any Intent since we don't need to send data back; simply set result to OK and close the activity.

```
public void getBack(View view){
    // Set the activity's result to RESULT_OK
    setResult(RESULT_OK);
    // Finish and close the current activity
    finish();
}
```

Step 3. Implement TextActivity in the first app to respond to the explicit Intent

In the first app, MainActivity will send an explicit Intent to open TextActivity (in the same app). To receive explicit Intents, you don't need to register anything, because the Activity was already specified as the target Activity for the particular Intent.

Also, TextActivity does not to get any data from the Intent. If you need to get the data from an explicit Intent, the procedure is similar as you did in step 2.2, depends what type of data you sent.

The data return to the MainActivity depends on what button being clicked. As set in step 1.2, both buttons use the same method (setText) to respond to an onClick event. Therefore, you only need to implement the setText method; you don't need to modify the onCreate method.

```
// Set unique key for extra
public final static String EXTRA_TEXT ="edu.sjsu.android.text";

public void setText(View view){
    Intent replyIntent = new Intent();
    if(view.getId() == R.id.confirm){
        // If confirm button is clicked, set the data as your name
        replyIntent.putExtra(EXTRA_TEXT, "Your Name");
        // And set the activity's result to RESULT_OK
        setResult(RESULT_OK, replyIntent);
    } else if (view.getId() == R.id.cancel){
        // If cancel button is clicked, set result to RESULT_CANCEL
        setResult(RESULT_CANCELED, replyIntent);
    }
    // Finish and close the current activity
    finish();
}
```

Step 4. Implement MainActivity in the first app to sent Intents and get data back

Finally let's implement the MainActivity of the first app. Note that you don' need to change onCreate.

4.1 Set Request Codes

Since we are dealing with 2 different Intents, we need to set unique integers as the request codes to differentiate those Intents.

```
public static final int TEXT_REQUEST = 1;
public static final int VIEW_REQUEST = 2;
```

4.2 Send Intents

The Intent to send based on the buttons the user clicked. Again, we've set a method called "choose" for both buttons' onClick attribute. So, let's implement this method. Explanations in comment. Pay attention to the difference between setting an explicit Intent and setting an implicit Intent.

```
public void choose(View view) {
    // If the explicit button ("Set Text") is clicked
    if (view.getId() == R.id.explicit) {
        // Create an explicit Intent with a specific activity (TextActivity)
        Intent explicit = new Intent(this, TextActivity.class);
        // Start an explicit intent for result
        // with request code for text request
        startActivityForResult(explicit, TEXT_REQUEST);
    }
    // If the implicit button ("Web Browser") is clicked
    else if (view.getId() == R.id.implicit) {
        // Let the data to be amazon's homepage
        Uri website = Uri.parse("http://www.amazon.com");
        // Create an implicit Intent with a specific action (view) and the data
        Intent implicit = new Intent(Intent.ACTION_VIEW, website);
        // Create a chooser for the implicit Intent
        // so the user can choose the app to perform the action
        Intent choose = Intent.createChooser(implicit,
                        "Load Amazon.com with:");
        // Start the chooser for result with request code for view request
        startActivityForResult(choose, VIEW_REQUEST);
    }
}
```

Note that, the app chooser part is only for demonstrating how to explicitly set an app chooser. As stated in the official guide, "...if the action to be performed could be handled by multiple apps and the user might prefer a different app each time—such as a "share" action, for which users might have several apps through which they might share an item—you should explicitly show a chooser dialog".

Without explicitly setting an app chooser, the system will still list all apps that are available for that implicit Intent you sent. When testing your app, you can try to start the implicit Intent without the chooser part (startActivityForResult(implicit, VIEW_REQUEST)) and see the difference.

4.3 Handle the Data Returned

Implement onActivityResult method (inherited from Activity class) to handle the data returned.

See the comments for explanations.

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    String result = "";
    // You can also use if-else-if statements here
    switch (requestCode) {
        case TEXT_REQUEST: // For the text request
            if (resultCode == RESULT_OK) // If user confirmed
                // Get data from the intent got from TextActivity
                // EXTRA_TEXT is The key of the extra,
                // a public static final string defined in TextActivity
                result = data.getStringExtra(TextActivity.EXTRA_TEXT);
            else if (resultCode == RESULT_CANCELED) // If user cancelled
                result = "Cancelled";
            break;
        case VIEW_REQUEST: // For the view request
            if (resultCode == RESULT_OK)
                result = "Website visited";
            break;
    }

    // Set the text to the corresponding result
    TextView text = findViewById(R.id.result);
    text.setText(result);
}
```

Step 5. Test your app

When testing your app, you need to run mybrowser first to get it installed in your emulator. Be sure to use an emulator with API level 29 or lower, otherwise, you may need some extra settings. As mentioned in class, API level 30 updated on the visibility of the user-defined app, so you may need to add <queries> in manifest to make your app visible to other apps. Check here for more information.

Submission

- Push your project to a Bitbucket repository (name it “exercise3”) by the due date.
- Invite and share your Bitbucket repository the grader (edmond.lin@sjsu.edu) and the instructor (yan.chen01@sjsu.edu).
- Submit repository links, etc. by answering all the questions in the “Exercise 3 - Intents” quiz on Canvas.
- Only your last submission before deadline will be graded based on the following criteria:
  2 pts if meets all requirements (the header of each row should be your name initially);
  1 pt if app failed/missing any requirement.