Exercise 8: Service & Broadcast Receiver

Due: May 9, 23:59 PDT

Overview

In this exercise, you will build an app (Fig. 1) that download a file using IntentService and notify the download is done via a broadcast receiver. You will get 1 point extra credit if you use JobIntentService instead of IntentService.

Note that this exercise will be different from the previous one. You need to write some of the code.

Name the project as “Exercise8YourName”.

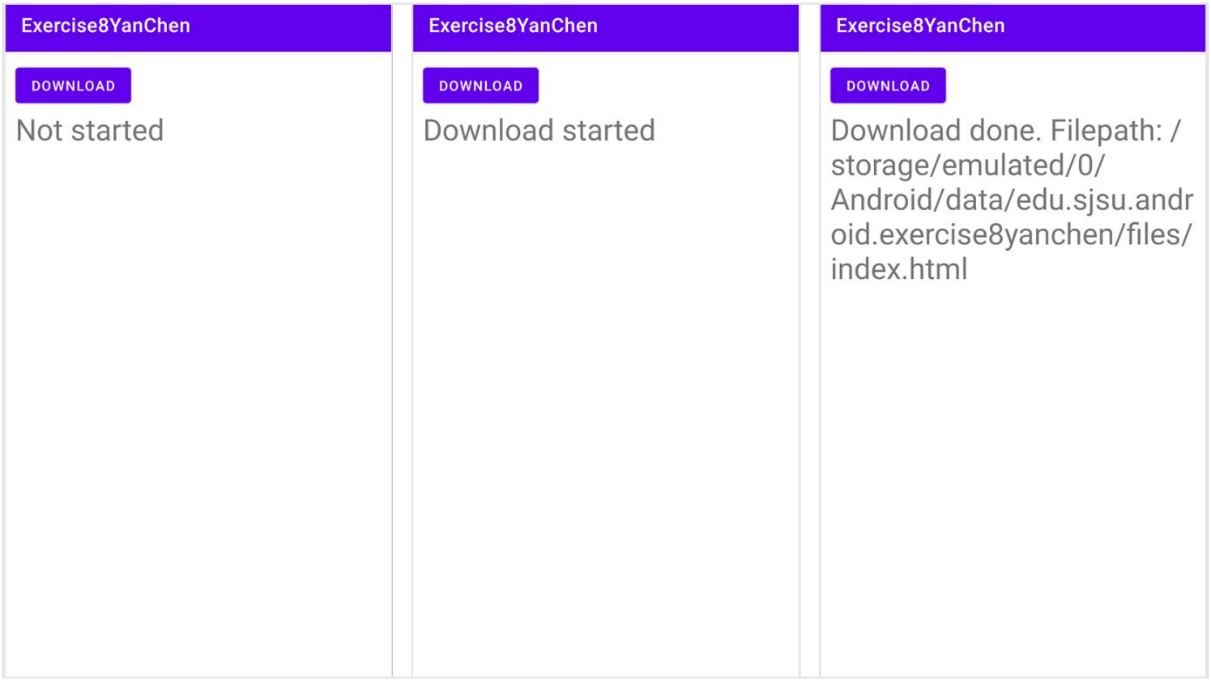

Fig. 1 Sample run of the app: launch page (left); service started after user clicks “DOWNLOAD” (middle); After the download service is done (right).

Prerequisites:

- Know the process of submitting your work (exercise 0)
- Be familiar with previous topics related to...
  - Project structure and UI (exercise 1 - 2, lesson 2 - 4)
  - Intent (exercise 3, lesson 8)
  - Write to android private external storage (exercise 5, lesson 15)
- 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.

Step 0. Request internet permission

Since we try to download file from the internet, we need to request the permission in manifest. It's not a dangerous permission so you don't need to request it at runtime.

```
<uses-permission android:name="android.permission.INTERNET" />
```

Step 1. Setup UI

Here are all the elements you need to add. As usual, the position does not matter.

Widget	id	Text	onClick
Button	n/a	"Download"	startDownload
TextView	status	"Not started"	n/a

Step 2. Implement DownloadService

Right-click the package, choose New -> Service -> Service (IntentService). Set the Class Name as "DownloadService", and uncheck "include helper start method". In this way, a service will be automatically registered in manifest. You can also add a service by adding a regular Java class that extending Service or IntentService (or any other subclass of Service), and then manually register that service in manifest.

Note that there will be some example code generated even if you unchecked "include helper start method". Except for the constructor, and the onHandleIntent method, you can delete everything else (including the code in the onHandleIntent method).

2.1 Set Attributes

Recall that services and broadcast receivers are all related to Intent, and when you send data via Intent, you need to provide a unique key (String) to identify that piece of data. Therefore, you need Strings for those keys. Make them as public static final objects so other classes in your package can also access them.

```
// Keys for extras used for the service
public static final String URLPATH = "edu.sjsu.android.exercise8.urlpath";
public static final String FILENAME = "edu.sjsu.android.exercise8.filename";
// Keys for extras used for the broadcast receiver
public static final String FILEPATH = "edu.sjsu.android.exercise8.filepath";
public static final String RESULT = "edu.sjsu.android.exercise8.result";
```

Also, define a String for the name of the broadcast action.

```
public static final String NOTIFICATION = " edu.sjsu.android.exercise8";
```

And define an int for the result, set it as private though.

```
private int result = Activity.RESULT_CANCELED;
```

2.2 Check if External Storage is Available

Similar to what you did in exercise 5, set a private helper method to check if the external storage is available. Note that it should be private, not public or static.

2.3 Implement publishResult

Implement a private helper method to publish the result via a broadcast receiver. Finish the TODOs.

```
private void publishResults(String filepath) {
    Intent intent = new Intent(NOTIFICATION);
    // TODO: add the filepath and the result to the above intent
    // using the corresponding keys
    sendBroadcast(intent);
}
```

2.4 Implement saveFile

Implement a private helper method to fetch the file from the url passed in, and then save (write) it to the file passed in. Returns true if successfully fetched and saved the file. Otherwise, return false.

```
private boolean saveFile(String urlpath, File output){
    try {
        // Fetch the file from the urlpath
        URL url = new URL(urlpath);
        InputStream stream = url.openConnection().getInputStream();
        InputStreamReader reader = new InputStreamReader(stream);
        // Save (Write) the file to output
        FileOutputStream fos = new FileOutputStream(output.getPath());
        int next;
        while ((next = reader.read()) != -1) {
            fos.write(next);
        }
        // Close streams and return true after done
        stream.close();
        fos.close();
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}
```

2.5 Implement onHandleIntent

This method starts the service, which will be stopped after this method returns.

In the method, get the url and the filename through the intent object passed in. And call the saveFile method you implemented in 2.4 to fetch and save the file. Only set result to OK if saveFile returns true. Finally, call publishResult.

Finish the TODOs.

```
@Override
protected void onHandleIntent(@NonNull Intent intent) {
    // TODO: return if external storage is not available
    // Hint: use the private helper method you implemented in 2.2

    // TODO: get the urlPath and filename from the intent
    //  by the corresponding keys
    String urlpath = "See todo";
    String filename = "See todo";
    if (filename == null) return;

    // TODO: set path to the path to private external storage (the root)
    File path = null;
    File output = new File(path, filename);

    // TODO: call saveFile, pass in urlpath and output
    // Set result to Activity.RESULT_OK only if saveFile returns true

    // TODO: call publishResults, pass in output.getAbsolutePath()
}
```

Step 3. Implement MyReceiver

Since we will register the broadcast receiver dynamically in code instead of in manifest, simply create a regular Java class that extends BroadcastReceiver.

3.1 Set Attribute and Constructor

Since we want to update the textview, so set a TextView object named “textview” as a class attribute and initialize it with the argument passed in the constructor. Remember to call super() at first.

3.2 Implement onReceive

This method is called when receiving the intent after the download service is done (Recall that you called sendBroadcast with an intent in step 2.3).

In the method, get the filepath and the resultCode through the intent object passed in. Set the textview based on the resultCode. Finish the TODOs.

```
@Override
public void onReceive(Context context, Intent intent) {
    // TODO: get the filepath (String) and resultCode (int) from the intent
    // by the corresponding keys (static Strings in DownloadService)
    // Hint: when getting int, you need to set a default value
    // Set it to Activity.RESULT_CANCELED
    // TODO: If resultCode is Activity.RESULT_OK,
    // set textview as "Download done. Filepath: " + filepath
    // else, set textview as "Download failed".
}
```

Step 4. Implement MainActivity

Since there is only one textview, using findViewById seems easier

4.1 Set Attributes and Implement onCreate

We need one TextView and one MyReceiver. Initialize them in onCreate.

```
private TextView textView;
private MyReceiver receiver;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    textView = findViewById(R.id.status);
    receiver = new MyReceiver(textView);
}
```

4.2 Register and Unregister the Broadcast Receiver

Register the MyReceiver object you set in 4.1 either in onCreate or in onResume. The intent filter parameter should be new IntentFilter(DownloadService.NOTIFICATION).

Unregister the MyReceiver object you set in 4.1 either in onPause or in onDestory.

Note that if you override any lifecycle callbacks in Activity, you need to first call the superclass implementations (super.onXXX).

4.3 Implement startDownload

This is the method to respond to a click on the button. When user clicks the button, the download service starts. Finish all TODOs.

```
public void startDownload(View view) {
    // TODO: sent an explicit Intent to start DownloadService
    // Hint: similar as starting an activity class
    // TODO: add following data to the above Intent:
    // filename: "index.html"
    // urlpath: "https://www.sjsu.edu/cs/index.html"
    // by the corresponding keys (static Strings in DownloadService)
    // TODO: start the service using the above Intent
    textView.setText("Download started");
}
```

Step 5. Test your app

Run the app, click the button, the textview should include your name if your package name is correct (recall the package name should be edu.sjsu.android.exercise8yourname).

If you open Device File Explorer, you should see the downloaded file (index.html) is under sdcard/Android/data/edu.sjsu.android.exercise8yourname/files/

Extra Credit

Migrate IntentService to JobIntentService as explained in lesson 23 page 14 & 15.

Submission

- Push your project to a Bitbucket repository (name it “exercise8”) 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 8 - Service & Broadcast Receiver” quiz on Canvas.
- Only your last submission before deadline will be graded based on the following criteria:
  3 pts if you use JobIntentService instead of IntentService;
  2 pts if meets all requirements;
  1 pt if app failed/missing any requirement.