# Lab – Intents

Objectives:

Familiarize yourself with Android Intents. You will create Intents and then use them to activate Activities.

The Intent class

In this lab, you will use both explicit and implicit Intents to activate Activities. In some cases, you may wish to receive results back from an Activity that you would like to start by calling startActivityForResult(). See http://developer.android.com/training/basics/intents/ for more documentation.

The main Activity for this application is called ActivityLoaderActivity. It should display two Buttons, labeled "Explicit Activation" and "Implicit Activation", and one TextView, initially displaying the words, "No Text Entered".
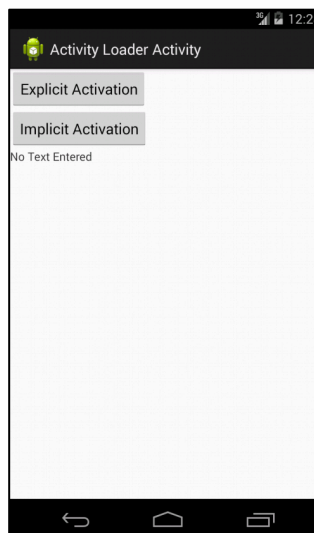
The application should behave as follows. When the user clicks the "Explicit Activation" Button, the ActivityLoaderActivity should launch a new Activity called ExplicitlyLoadedActivity. This activity should display a user interface containing an EditText and a Button. An EditText object allows a user to enter text.

When the user selects the "Enter" Button, this Activity should finish, returning the contents of the EditText object (seen above as 'Hello World!) back to the ActivityLoaderActivity. Once the
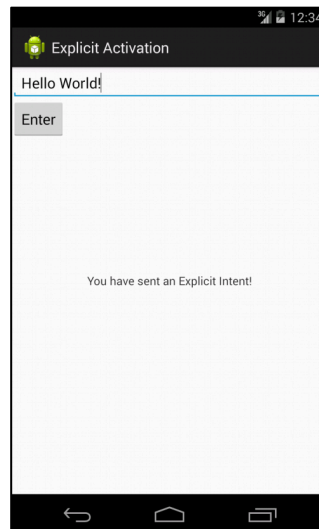

ActivityLoaderActivity is again visible, any text that was returned from the ExplicitlyLoadedActivity should now appear in the ActivityLoaderActivity's TextView object, as shown below.
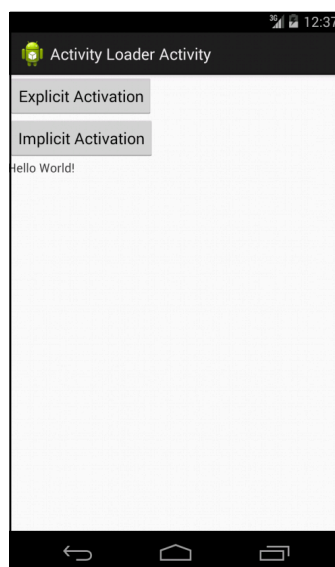
When the user clicks on the "Implicit Activation" Button, the ActivityLoaderActivity will create an implicit Intent, and then use it to implicitly activate a separate application to view the URL, "http://www.google.com". Because multiple applications may be able to handle a given Intent, ActivityLoaderActivity will create and display an App Chooser (it's shown below, but yours may have a slightly different layout), allowing the user to select the specific application to handle the Intent. For this assignment, the Chooser should display at least two choices: Android's built-in Browser and a separate application you've created called MyBrowser. To create an App Chooser, start by creating an initial Intent to view a webpage, (as part of this process you'll need to use the Uri class' parse() method). Then, create a second Intent, based on the first one, by calling the Intent class' createChooser() method. Finally, start a new Activity using this second Intent. See "Sending the User to Another App" for more information about creating chooser Intents,
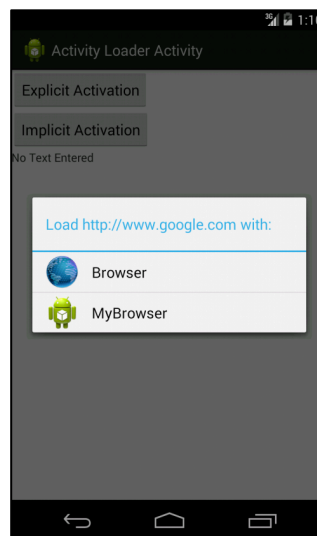
If the user selects the Android Browser from this chooser dialog, then that application will open and display the webpage at the given URL. If the user instead selects the MyBrowser application, then that application will open and simply display the text of the URL in a TextView.

3
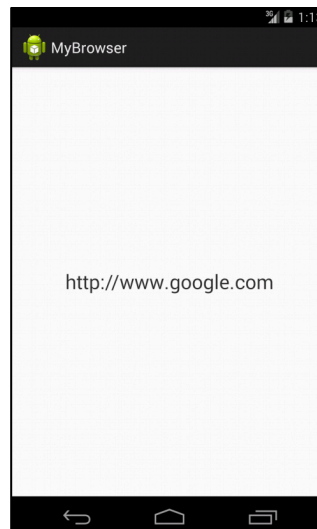
Implementation Notes:

From the download package, import the two project skeletons in the Intents/SourceFiles/Skeletons directory, IntentsLab.zip and MyBrowser.zip. For the IntentsLab project, do the following steps.

1) In ActivityLoaderActivity.java, implement launching the ExplicitlyLoadedActivity and returning the text result to the ActivityLoaderActivity.

   a) When the user clicks the "Explicit Activation" Button create an explicit Intent for launching the ExplicitlyLoadedActivity.

   b) In the ActivityLoaderActivity onActivityResult() will eventually be called. In that method you should use Intent.getStringExtra() to read it back out. See "Getting a Result from an Activity" for more information.

2) In ExplicitlyLoadedActivity.java, return user-entered text. when the user presses the Enter Button.

   a) To return the text, you can use the Intent.putExtra() method to store the text in an Intent to be returned to the ActivityLoaderActivity. What key will you use when storing the text? You will pass this Intent to the setResult() method, add an appropriate result code, and then call finish().

3) In ActivityLoaderActivity, implement launching an unknown Activity to view a URL.

   a) When the "Implicit Activation" Button is clicked you should create an implicit Intent indicating that you want to a view a URL. The application should use an app chooser to display Activities that are capable of viewing URLs. See the "Sending the User to Another App" section on the Android developer site.

For the MyBrowser project, do the following.

4) Implement and deploy the MyBrowser application.

   a) In MyBrowser's AndroidManifest.xml file, add the appropriate Intent Filter to MyBrowserActivity so that Android will know that this Activity can view web pages. To indicate that the Activity can handle this type of Intent, you will need to add an <intent-filter> to the <activity> in MyBrowser's AndroidManifest.xml file. Be sure to add the correct <action>, <category>, and <data> elements to an <intent-filter> element.

As explained above, you can find the skeleton code in the Lab's download package. Below are the list of classes and methods you need to implement. You can find them in the code by looking for comments with the String TODO:

1. In ActivityLoaderActivity.java implement the following methods.

private void startExplicitActivation(): create an intent and start the ExplicitlyLoadedActivity.

private void startImplicitActivation(): create an Implicit Intent and use the Intent.createChooser() method to select an application for viewing a web page. What action string should you use? Activate the user-selected Activity using this Implicit Intent.

protected void onActivityResult(**int** requestCode, **int** resultCode, Intent data): get the data that is returned from the ExplicitlyLoadedActivity Activity and display it in the mUserTextView. How do you find and access that TextView?

2. In ExplicitlyLoadedActivity.java implement the following method.

private void enterClicked(): get user-entered text from the EditText, store it in an Intent, and return it to the ActivityLoaderActivity.

3. In MyBrowser's AndroidManifest.xml file, add the necessary intent filter tags. You'll need to define the right <action>, <category> and <data> elements. What values should be used for these? Remember that how you express certain values in Java can be different than how you express them in xml.

Testing

The test cases for this Lab are in the IntentsLabTest project. You can run the test cases either all at once, by right clicking the project folder and then selecting Run As>Android JUnit Test, or one at a time, by right clicking on an individual test case class (e.g., ExplicitTest.java) and then continuing as before. The test classes are Robotium test cases. You will eventually have to run each test case, one at a time, capture log output, and submit the output to Coursera. These test cases are designed so that if you pass the test case, your submission should pass as well. The Explicit test case should output exactly 3 Log messages. The TestDangerousApp test case should output exactly 2 Log messages.

As you implement various steps of the Lab, run the test cases every so often to see if you are making progress toward completion of the Lab.

Warnings:

1. These test cases have been tested on a Galaxy Nexus AVD emulator with API level 18. To limit configuration problems, you should test you app against a similar AVD. Also, when testing, make sure that your device is in Portrait mode when the test cases start running.

2. The ImplicitTest test case requires that you've installed both the IntentsLab and the MyBrowser applications. Remember - If you change MyBrowser, you'll need to reinstall it.
3. The ImplicitTest test case causes the MyBrowser application to start. Due to Android security policies, Robotium test cases cannot test across multiple applications. So when a test case starts up MyBrowser, it can't interact with IntentsLab after that. What this means for you, is that after running ImplicitTest, you must manually exit MyBrowser (for example, by hitting the back button). If you don't and then try to run another test case, Eclipse gets completely stuck.
4. Spelling counts. Pay attention to how to specify data values in your xml files. Leave out or misspell a single letter and Android won't understand what you really meant.

As you implement various steps of the lab, run the test cases every so often to see if you are making progress toward completion of the lab.

Once you've passed all the test cases, follow the instructions below to capture log information to submit to the Coursera system for grading.

Tips: Saving a LogCat filter.

1. In the LogCat View, press the green "**+**" sign to "add a new LogCat filter."
2. A dialog box will pop up. Type in the information shown in the screenshot below.
3. A saved filter called, "IntentsLab " will appear in the LogCat View.
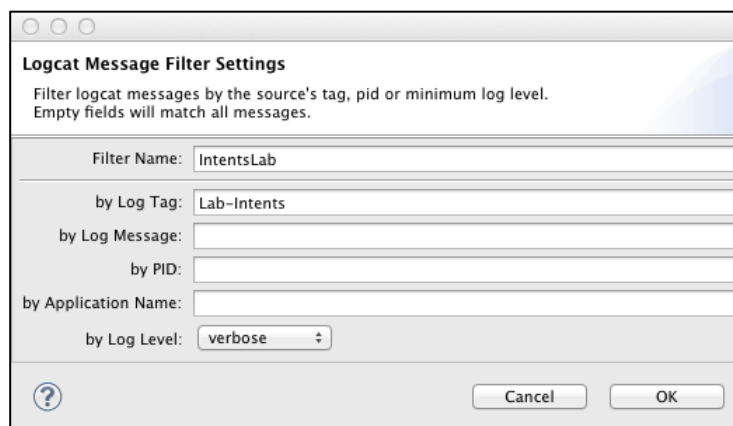
Tips: Running your test cases and capturing your LogCat output for submission.

1. For each test case, clear the LogCat console by clicking on the "Clear Log" Button (the Button with the red x in the upper right of the LogCat View).
2. Then right click on an individual test case file in the Project Explorer. Run As -> Android JUnit Test.
3. When the test case finishes, if it's not already selected, click on the "PermissionsLab" filter in the left hand pane of the LogCat View.
4. Select everything within the LogCat View (Ctrl-A or Cmd-A) and press the "Export Selected Items To Text File" button (floppy disk icon) at the top right of the LogCat View.
5. Submit this file to the Coursera system.