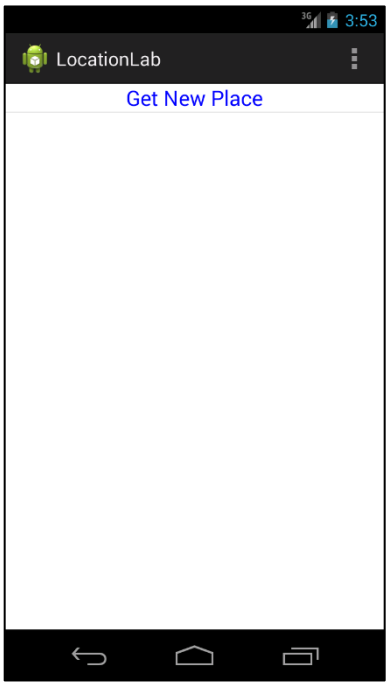
Content Provider Lab

Use Content Providers to allow your application to save, access and share data.

Objectives:

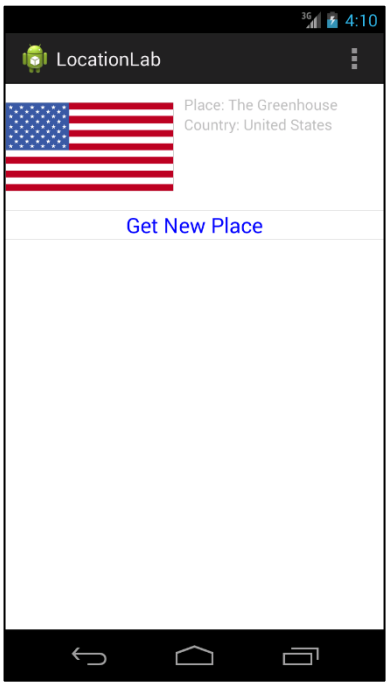
In this week's lab, you'll learn more about Content Providers. Content Providers are used to store, manage and share data across applications. Often, but not always, this data is stored in an SQLite database.

This week's lab will build off last week's Location lab. As shown below, this application will have the same UI elements as last week's, displaying a ListView containing a set of Place Badges. In this application, however, the Place Badges that a user collects will be stored in an SQLite database. You will be using URI's to access your Content Provider.

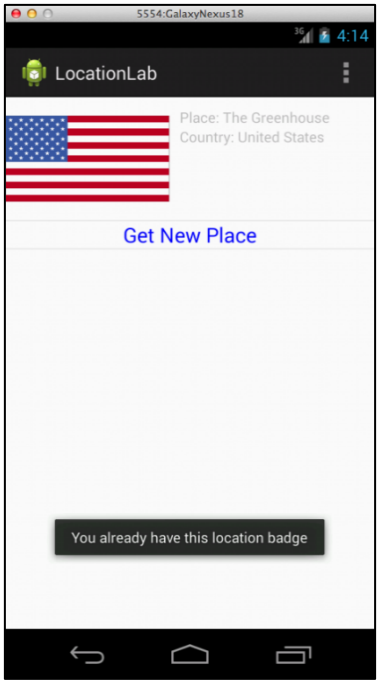


If the user clicks on the Footer and the application does not already have a Place Badge for a location within 1000 meters of the user's current location, then the application acquires the data needed to create the Place Badge. While this is happening, PlaceViewAdapter will insert this newly collected information into the Content Provider.

Once these processes are complete, the new Place Badge will appear in the ListView.



If the user clicks on the Footer, but the application already has a Place Badge for a location within 1000 meters of the user's current location, then the application should display a Toast message with the text, "You already have this location badge."



As before, the application should gracefully handle or better yet prevent the user clicking on the Footer when the application has not acquired a valid user location.

To implement this application, you will need to acquire location readings from Android. How you implement this is up to you, however, your app will need to listen for location updates from the NETWORK_PROVIDER (which we will control for testing purposes). Be aware that listening for location updates from other providers is likely to cause problems during testing.

Implementation Notes:

1. Download the application skeleton files and import them into your IDE.
2. **IMPORTANT NOTE:** For this Lab we needed to change several of last week's project files. As a result, we strongly encourage you to copy your code changes from last week's into the corresponding location in this week's skeleton files, rather than trying to simply add new changes to last week's Lab files.
3. Complete the TODO items, which are in the PlaceViewActivity.java and PlaceViewAdapter.java files. Your geonames.org username will also need to be updated in PlaceDownloaderTask.java.

Testing:

The test cases for this Lab are in the ContentProviderLabTest project, which is located in the ContentProviderLab/SourceFiles/TestCases/ContentProviderLabTest.zip file. 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., TestOneValidLocation.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 to drive your app through a set of steps, passing the test case is not a guarantee that your submission will be accepted. The TestOneValidLocation test case should output exactly 5 Log messages. The TestSameLocation test case should output exactly 7 Log messages. The TestTwoValidLocations test case should output exactly 9 Log messages.

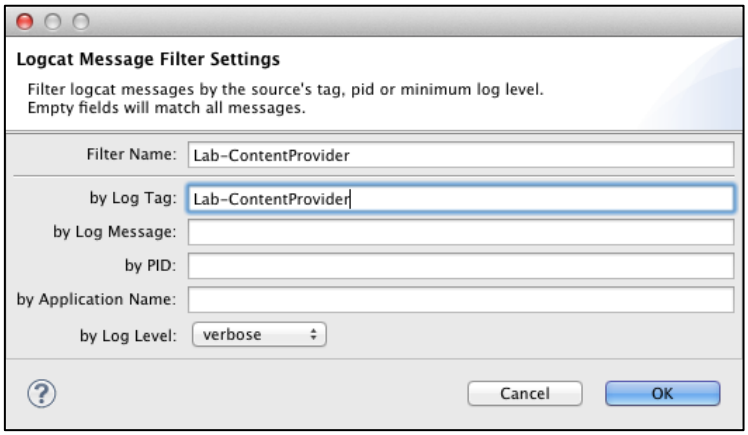
Warnings:

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.
2. Our MockLocationProvider relies on a method that was included in API level 17. Therefore, the TestCases will fail to compile on earlier platforms.
3. During testing you should not provide your own location information.
4. The application works best with an actual network connection. You will also need to create an account at http://www.geonames.org/login. Your username will need to be updated in PlaceDownloaderTask.java.

Once you've run all the test cases, submit your log information 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, "Lab-ContentProvider" 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 " Lab-ContentProvider " 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.