Module: Mobile Application development (Android)

Session 30 : Android Services Assignments (Practice)

|  |  |
| --- | --- |
| |  | | --- | | IntroFor this lab you will be developing a new Application named AppRater that suggests other Applications for users to download and try. The purpose of the application is to share fun and interesting applications with other users. The users can then rate the applications. Its a simple application that will be extended and used to run the App Development Contest at the end of the quarter.[https://sites.google.com/site/androidcoursearchive/_/rsrc/1267216937731/labs/lab-6/AppRaterUseCase.png](https://sites.google.com/site/androidcoursearchive/labs/lab-6/AppRaterUseCase.png?attredirects=0)ObjectivesAt the end of this lab you will be expected to know:How to create and use ContentProviders.How to create and use Services.How to broadcast Intents.How to post notifications in the Notification Bar.How to respond to Intent broadcasts by creating and using BroadcastRecievers.How to perform work in a Background thread.ActivitiesFor this lab you will be working with a brand new application, completely independent of the previous labs. Once you have a general understanding of the components of the application you will begin implementing it.*IMPORTANT:**You will be given a Skeleton Project to work with. This project contains all of the java and resource files you will need to complete the lab. Some method stubs, member variables, and resource values and ids have been added as well. It is important that you not change the names of these methods, variables, and resource values and ids. These are given to you because there are unit tests included in this project as well that depend on these items being declared exactly as they are. These units test will be used to evaluate the correctness of your lab. You have complete access to these test cases during development, which gives you the ability to run these tests yourself. In fact, you are encouraged to run these tests to ensure that your application is functioning properly.*1. Setting Up...1.1 Creating the ProjectTo begin, you will need to download and extract the [skeleton project for the AppRater](https://sites.google.com/site/androidcoursearchive/resources/lab-skeletons/Lab6_v1_partial_stub.zip?attredirects=0) application.Click Here to download the skeleton project.Extract the project, making sure to preserve the directory structure.Take note of the path to the root folder of the skeleton project.Next you will need to setup an Android project for this app.  Use the settings listed below for the remaining fields:Project Name:  lab6<userid>Build Target:  Android 1.6Application Name:  AppRater1.2 Familiarize Yourself with the ProjectThe project contains seven java files and two XML layout files. AppRater.java will contain the definition for the main AppRater Activity class. This is the class that will display the list of applications the user is supposed to test and rate. The AppRater class makes use of a very simple XML layout file called app\_list.xml which has been completed for you. It is composed of a ListView that displays a list of applications that a user is supposed to try out.Applications that users are supposed to try out and rate are represented by the App model class, defined in App.java. This is a simple class that encapsulates the name of the application, a rating for it, the Market-URI from which the app can be downloaded (*we'll discuss this later but think of it as a link to install it from the Market application*), a boolean used to flag the application as installed, and a unique integer ID.The AppView class is a custom class that is used for visualizing the state of an App object. The AppView class has an App member variable on which it bases the state of its display. It's layout is defined in the app\_view.xml XML layout file, which has already been completed for you. This is a very simple layout file, composed of a LinearLayout containing a single TextView. The TextView merely displays the name of the App object that the AppView is visualizing.The AppView class has three different states:If the AppView's App member has not been installed yet, then it's background color is red.If the AppView's App member has been installed but hasn't been rated yet, then it's background color is yellow.If the AppView's App member has been rated, then it's background color is green.The AppRater Activity gets the Apps it will display from the AppContentProvider class as a Cursor of Apps. This is quite similar to how the JokeList Activity got a Cursor of Jokes from the JokeDBAdapter. The AppRater Activity's ListView then uses the AppCursorAdapterclass to bind AppViews to the cursor of Apps.The AppRater Activity starts the AppDownloadService in order to add new Apps to the underlying database. The new Apps are retrieved from a web site.  This is kind of like launching a new Activity except that there is no User Interface for the Service. The AppDownloadService then downloads and inserts new Apps into the database. In order to insert new Apps into the database the AppDownloadService must use the AppContentProvider's insert method. This is quite similar to how the JokeList Activity (in previous labs) downloads and inserts new Jokes by using JokeDBAdapter.Once the AppDownloadService has successfully added a new App, it broadcasts a special Intent. The AppRater Activity will use its internal NewAppReceiver class to listen for that special Intent. When it hears the Intent it knows that it has to update its cursor of Apps.Lastly, once a user has downloaded, installed, and tested an App they can give it a rating. The user can apply a rating to an App via a ContextMenu in the AppRater Activity. The AppRater Activity then saves the changes to the App through the AppContentProvider.A general class diagram for the project is depicted below. Classes you will be implementing are colored yellow, the new Android classes that you will be working with are colored in blue, and the standard Android classes you should be familiar with are colored in white.2. AppRater ActivityIn this section of the lab you will be learning how to use ContentProviders to retrieve and update persistant data, how to use Services to perform work in the background, and how to respond to system-wide notifications by using BroadcastReceivers. All of the work done in the following subsections should be performed in the AppRater.java file. The file has been setup to import the already completed versions of the AppContentProvider and AppDownloadService classes.2.1 Querying a ContentProviderYou can interact with a ContentProvider in much the same way that you interacted with the JokeDBAdapter. You can execute a query which will return a Cursor, you can delete data, you can update data, and you can insert data. The main difference is that ContentProviders offer a way to share data between applications. Where your previous Joke database was only visible to the JokeList application, the AppContentProvider that you will implement later on will make your database usable by any Application.ContentProviders also abstract away how the underlying data is persisted. While in this application you will be using an SQLiteDatabase, you could just as easily store the data in files, on a remote server, or whatever else you can come up with. For a complete background on ContentProviders see the Android Developer Guide on [ContentProviders](http://developer.android.com/guide/topics/providers/content-providers.html).For this section you must Query the AppContentProvider for a cursor of Apps that the AppRater Activity will display. When a user clicks on one of the listed apps, the Market application will be launched to display the App's info page. The AppContentProvider you are using is currently pre-populated with a single App so that you can test your implementation.[https://sites.google.com/site/androidcoursearchive/_/rsrc/1267226153443/labs/lab-6/LaunchMarketActivity.png](https://sites.google.com/site/androidcoursearchive/labs/lab-6/LaunchMarketActivity.png?attredirects=0)2.1.1 Fill in onCreate(...)Start by initializing the AppRater Activity. You should query the AppContentProvider to retrieve a Cursor containing all of the Apps it contains. Use this cursor to initialize your AppCursorAdapter and ListView members.The Android Developer Guide offers instructions on how to [Query a ContentProvider](http://developer.android.com/guide/topics/providers/content-providers.html#querying).The URI for the AppContentProvider is defined in the AppContentProvider.CONTENT\_URI constant String.You should be able to run your application and view a list of Apps.2.1.2 Launch the Market ActivityWhen a user clicks on an AppView, you should start the Market Activity implicitly, meaning that you should not use an Intent that explicitly launches the Market Application. Instead, you should use an Intent object that should cause the Market application to be launched. You do this by specifying an Intent action-string and the data to be acted upon. When the Market application starts, it immediately displays the info page for the App that was clicked.If you are confused on how to implicitly start an Activity or for more details on Intents, see either the [Android Documentation](http://developer.android.com/reference/android/content/Intent.html) or the [Android Developer Guide](http://developer.android.com/guide/topics/intents/intents-filters.html#iobjs) on Intents:You should use the App.m\_strInstallURI as the data which should be acted upon.You should use the Intent.ACTION\_VIEW as the action to be performed.You should be able to run your application, click on an App, and have the Market application display the selected App's info page.2.2 Starting & Stopping ServicesServices are application components that run in the background and do not have User Interfaces. Any time that there is some type of code that needs to be run regularly but does not need a user interface you could probably implement it as a Service. Services can be started from your application's Activities that are currently visible, or they can be awoken by System Notifications when your Activities are all closed. The AppRater application will use a Service to update its ContentProvider with new data. For a more detailed description, read the Android Developer Guide on [Service Application Components](http://developer.android.com/guide/topics/fundamentals.html#appcomp).The AppContentProvider is currently only storing a single App. In this next section you will have the AppRater Activity start the AppDownloadService class. The AppDownloadService will download more Apps from a server and add them to the AppContentProvider. Most importantly, the AppDownloadService will execute in a background thread to keep the main AppRater Activity responsive. The AppDownloadService will be controlled through the AppRater Activity's Options Menu.[https://sites.google.com/site/androidcoursearchive/_/rsrc/1267226294267/labs/lab-6/AppRaterOptionsMenu2.png](https://sites.google.com/site/androidcoursearchive/labs/lab-6/AppRaterOptionsMenu2.png?attredirects=0)2.2.1 Create the Options Menu The AppDownloadService needs to be started and stopped from the AppRater's Options Menu. Additionally, you may want to remove all the Apps from the ContentProvider in order to repeatedly test the AppDownloadService, so you will need to create a MenuItem for this as well. Implement the AppRater Activity's Options Menu as follows: Consider overriding the [Activity.onOptionsItemSelected(...)](http://developer.android.com/reference/android/app/Activity.html#onOptionsItemSelected(android.view.MenuItem)) method instead of implementing and setting OnMenuItemClickListeners.The "Start Downloading" MenuItem should use AppRater.MENU\_ITEM\_STARTDOWNLOAD as its ID and should start the AppDownloadService:Use the [Context.startService(...)](http://developer.android.com/reference/android/content/Context.html#startService(android.content.Intent)) method with an *explicit*Intent.The "Stop Downloading" MenuItem should use AppRater.MENU\_ITEM\_STOPDOWNLOAD as its ID and should stop the AppDownloadService:Use the [Context.stopService(...)](http://developer.android.com/reference/android/content/Context.html#stopService(android.content.Intent)) method.The "Remove All Apps" MenuItem should use AppRater.MENU\_ITEM\_REMOVEALL as its ID and should delete all Apps from the ContentProvider:See Android Developer Guide on [Modifying ContentProviders](http://developer.android.com/guide/topics/providers/content-providers.html#modifying) for information on how to delete rows from a ContentProvider if you're confused.*Passing in null for the where and selectionArgs parameters will cause all rows in the ContentProvider to be deleted.*Run you application and you should see the single pre-populated App. Click the "Start Downloading" MenuItem, and wait a few seconds for the Apps to download and appear in your ListView. You should see a notification in the Notification Bar. Click the "Remove All" MenuItem and all the Apps should disappear.It is important to note that your AppDownloadService will continue to run. If you wait a few more seconds the ListView should repopulate itself with the Apps. The AppDownloadService will run even after you have closed the AppRater Activity. You must explicitly stop it by using the "Stop Downloading" MenuItem in order to shut it down.2.3 Responding to BroadcastsIntents are used as a System-level message passing system. They can be used to start application components, or they can be used to send messages between components. In order to listen for a message, you need to use a Broadcast Receiver.In the AppRater Application, the AppDownloadService will broadcast an Intent containing the AppDownloadService.NEW\_APP\_TO\_REVIEW action string constant after it successfully downloads and saves a new App to the database. Since the AppDownloadService runs in the background as a Service, this could happen whether the AppRater Activity is running or not. If the AppRater Activity is running, it should display a Toast notification telling the user that a new App was downloaded.In this section, you will implement the AppReceiver class, which extends BroadcastReceiver. AppReceiver will listen for AppDownloadService.NEW\_APP\_TO\_REVIEW Intents and display a Toast notification when it hears them.[https://sites.google.com/site/androidcoursearchive/_/rsrc/1267226326733/labs/lab-6/BroadcastToastNotification2.png](https://sites.google.com/site/androidcoursearchive/labs/lab-6/BroadcastToastNotification2.png?attredirects=0)2.3.1 Implement the AppRater.AppReceiver ClassFill in the AppReceiver.onReceive(...) method so that the AppRater Activity displays a Toast notification telling the user that a new app was downloaded:You should test the Intent argument to ensure it has AppDownloadService.NEW\_APP\_TO\_REVIEW as its Action-String.Your Toast notification should use the R.string.newAppToast String resource.2.3.1 Register & Un-Register for BroadcastsIn order for your AppReceiver to be notified when AppDownloadService.NEW\_APP\_TO\_REVIEW action Intents are broadcasted, you must register an instance of it with your Activity. Likewise, when the Activity moves out of the foreground, you need to unregister the AppReciever.Override and fill in Activity.onResume(...).Initialize AppRater.m\_receiver.Construct a new IntentFilter that only listens for the AppDownloadService.NEW\_APP\_TO\_REVIEW action string constant.*Intents can be broadcasted from any android component to any android component. Declaring an IntentFilter allows you to specify which Intents you want to listen for by specifying certain Intent characteristics. In this case we're only interested in Intents that have an action string of AppDownloadService.NEW\_APP\_TO\_REVIEW.*Use the [Activity.registerReceiver(BroadcastReceiver receiver, IntentFilter filter)](http://developer.android.com/reference/android/content/Context.html#registerReceiver(android.content.BroadcastReceiver,%20android.content.IntentFilter)) method to register m\_receiver for Intents containing the AppDownloadService.NEW\_APP\_TO\_REVIEW action string.Override and fill in Activity.onPause(...).Use the [Activity.unregisterReceiver(...)](http://developer.android.com/reference/android/content/Context.html#unregisterReceiver(android.content.BroadcastReceiver)) method to unregister m\_receiver.Run your application and click the "Remove All Apps" MenuItem to clear the AppContentProvider. Then click the "Start Downloading" MenuItem. After a few seconds the AppRaterActivity should display the new app Toast notification as new Apps are added to the ListView.2.4 Updating Data in ContentProvidersIt is sometimes necessary to modify the data contained in a ContentProvider. In this next section you will add functionality to persist changes to App objects in the ContentProvider. In particular, you will enable the user to edit whether they have installed an App, as well as apply a rating to it. Both of these actions should be performed through a ContextMenu that gets generated by Long-Clicking on an AppView. The main use case is pictured below.[https://sites.google.com/site/androidcoursearchive/_/rsrc/1267226356859/labs/lab-6/AppViewContextMenu.png](https://sites.google.com/site/androidcoursearchive/labs/lab-6/AppViewContextMenu.png?attredirects=0)2.4.1 Create the ContextMenu The Context menu should use a CheckBox MenuItem for selecting whether the App has been installed, and a group of four RadioButtons for selecting the rating to apply to an App.   Impelmentation Requirements:   * The "Installed" CheckBox should use AppRater.MENU\_ITEM\_INSTALLED as its MenuItem ID. * The "X Stars" Rating RadioButtons should use AppRater.MENU\_RATING\_GROUP as their Group ID, and should use the AppRater MENU\_ITEM\_RATING1, MENU\_ITEM\_RATING2, MENU\_ITEM\_RATING3, MENU\_ITEM\_RATING4 constants as their individual MenItemID's.   Functional Requirements:   * If the App has not been installed you should not be able to give it a rating and all the RadioButtons should be disabled (but still visible). * Once an App has installed, the RadioButtons should be enabled to allow the user to edit the rating. * If the user has installed and rated an app, then the ContextMenu should have the proper CheckBox and proper RadioButton selected respectively. * If the App has not been rated, then no RadioButton should be selected.  2.4.2 Persist App ChangesThe tricky part here is figuring out which App to make the changes to. You need to identify the AppView that was clicked when the ContextMenu was generated. From the AppView you can retrieve the App. Follow these instructions to retrieve the AppView and its App:When the onCreateContextMenu(...) was called, your ListView that generated the ContextMenu provided the method with an [AdapterView.AdapterContextMenuInfo](http://developer.android.com/reference/android/widget/AdapterView.AdapterContextMenuInfo.html) parameter.When you add a MenuItem to a ContextMenu the ContextMenuInfo gets added to the MenuItem.The AdapterContextMenuInfo has a public member variable that references the AppView that was clicked.Retrieve the ContextMenuInfo from the MenuItem.You'll need to cast it to an AdapterView.AdapterContextMenuInfo.Retrieve the targetView member from the AdapterContextMenuInfo object.You'll need to Cast it to an AppView.Retrieve the App from the AppView.*This is the App that was clicked. So simple right :)*Here are the Functional/Implementation Requirements you should statisfy:When the user clicks the "Installed" MenuItem, you should toggle the App's m\_bInstalled member variable and reset its m\_nRating to App.UNRATED.When the user clicks a "X Stars" Radio Button, you should set the App's m\_nRating to the ID of the clicked MenuItem.When the user clicks any MenuItem you should update the ContentProvider with the new data for the App.When the user clicks any MenuItem this should cause the background color of the AppView to change.Should change to red if the app has been uninstalled.Should change to yellow if the app has been installed.Should change to green if that app has been rated.An additional use case for uninstalling a previously installed and rated App is pictured in the screenshots below. Notice that the rating should be reset to App.UNRATED if it is re-installed.[https://sites.google.com/site/androidcoursearchive/_/rsrc/1267226389582/labs/lab-6/UninstallReinstallApp.png](https://sites.google.com/site/androidcoursearchive/labs/lab-6/UninstallReinstallApp.png?attredirects=0)Hints:You've changed your underlying data, don't forget to update your Cursor & Adapter.The Android Developer Guide offers instructions on how to [Update a ContentProvider](http://developer.android.com/guide/topics/providers/content-providers.html#modifying).The URI for the AppContentProvider is defined in the AppContentProvider.CONTENT\_URI constant String.IMPORTANT: You must restrict your update to a single App row in the ContentProvider or you will end up persisting your change to all the Apps. The section on [Querying a ContentProvider](http://developer.android.com/guide/topics/providers/content-providers.html#querying) offers advice on how to restrict your URI to affect only a single record.*If you create your URI correctly you can pass in null for the where and selectionArgs parameters.*3. Implementing Backgound Services3.1 AppDownloadServiceFor this section of the Lab you will implement your own version of the AppDownloadService that you used in the AppRater Activity. For a complete Background on the Service class you should probably read the the Android Developer Guide on the [Service Lifecycle](http://developer.android.com/guide/topics/fundamentals.html#servlife) and the Android Documentation on the [Service Class](http://developer.android.com/reference/android/app/Service.html).3.1.1 Handle Initialization and Destruction of the ServiceWhen the Service is first started, the onCreate() method is called. You should initialize your Timer and TimerTask.You should read the Documentation on the [java.util.TimerTask Class](http://developer.android.com/reference/java/util/TimerTask.html). A TimerTask represents a [java.util.Runnable](http://developer.android.com/reference/java/lang/Runnable.html) object that your Timer will execute.Your TimerTask should only call the AppDownloadService.getAppsFromServer() method.You should read the Documentation on the [java.util.Timer Class](http://developer.android.com/reference/java/util/Timer.html). Timer's are essentially threads which can be scheduled to execute TimerTasks at specified rates.Your Timer should schedule your TimerTask for *"fixed-delay"* execution.Your TimerTask should be scheduled at a period of AppDownloadService.UPDATE\_FREQUENCY milliseconds.*Feel free to adjust this number for testing purposes.*You shouldn't create a daemon Timer.When the Service is stopped, its onDestroy() method is called. You should free up any resources that you have allocated in this method, including threads.If you don't explicitly cancel your Timer and TimerTask, they will continue to run indefinitely.3.1.2 Fill In the getAppsFromServer() MethodThis method should download a list of Application Name and InstallURI pairs from a server. For each Application Name and InstallURI pair, the method should construct an App object and hand it over to the AppDownloadService.addNewApp(...) method.You should use the two argument App constructor.The method should download from the URL that is stored in the AppDownloadService.GET\_APPS\_URL String constant.If you visit the URL, you will see the format the data comes in:The individual Application Name and InstallURI's data pieces, which correspond to the same App, are separated by a comma ',' character .Application Name and InstallURI pairs, which represent separate Apps, are separated by a semicolon ';' character.Example: *(Three separate Apps - Green; Red; Yellow)*app-name-1,market://details?id=com.package1.app1;app-name-2,market://details?id=com.package2.app2;app-name-3,market://details?id=com.package3.app3;3.1.3 Fill in the addNewApp(...) MethodThis method should first test to see if the App already exists in the ContentProvider. You should query the AppContentProvider. If the App does not exist, then you should insert it into the ContentProvider and call the AppDownloadService.announceNewApp() method.Two Apps are considered equal if they have the same name.The Android Developer Guide offers instructions on how to [Insert](http://developer.android.com/guide/topics/providers/content-providers.html#modifying)and [Query](http://developer.android.com/guide/topics/providers/content-providers.html#querying)a ContentProvider.*Note: Do not add the ID to the ContentValues for the new App you are inserting, which will be -1 if you used the correct constructor. The Database will automatically generate a unique ID for you. Trying to insert a new App with an ID of -1 will cause an SQLException.*The Android Developer Guide offers instructions on how to [Broadcast Intents](http://developer.android.com/guide/appendix/faq/commontasks.html#broadcastreceivers). **3.1.4 Fill in the announceNewApp() Method**  This method should broadcast an Intent containing the AppDownloadService.NEW\_APP\_TO\_REVIEW action string, and launch a Notification.   * For the notification, use the following resources:   + Content Title: R.string.app\_name   + Content Text: R.string.newAppNotificationText   + Ticker Text: R.string.newAppNotificationTicker   + Notification Icon: R.drawable.icon. * When you click on the notification, it should launch the AppRater Activity. * The Android Developer Guide offers instructions on how to [Create Notifications](http://developer.android.com/guide/topics/ui/notifiers/notifications.html).  3.1.5 Update the AppRater ActivityThe AppRater Activity class is currently using the AppDownloadService implementation that was provided to you. You need to remove the import statement at the top of the AppRater.java file. Delete the following line:import edu.calpoly.android.appraterkey.AppDownloadService; You should now test your application to ensure that it functions as it did before.3.1.6 Update Your ManifestYou will need to add your Service to the AndroidManifest.xml file. You can look in the file to see how it was done for the appraterkey.AppDownloadService.4. Implementing a ContentProvider4.1In this final section of the lab, you will implement your own version of the AppContentProvider you have been working with. The AppContentProvider uses an SQLiteDatabase to persist the App data. Much of the work you will do to implement this class will be very similar to the work that you performed for the JokeDBAdapter class. The main difference being that there are only four basic methods, as you have seen each of them so far. One method each for querying, inserting, updating, and deleting data from the SQLiteDatabse.4.1.1 Implement AppDBHelperYou should begin by implementing the AppDBHelper class so that you can use it for opening the SQLiteDatabase. The Database creation and deletion strings are stored as constants of the AppDBHelper class.4.1.2 Initialize the ContentProviderWhen ContentProvider is being started, the onCreate(...) method gets called. You should initialize your SQLiteDatabase member variable here. Read the Android Documentation on the [ContentProvider.onCreate(...)](http://developer.android.com/reference/android/content/ContentProvider.html#onCreate()) to make sure that you are returning the proper value.4.1.3 Implement the query(...) MethodYou should start by reading the Android Developer Guide on [Summary URI Structure](http://developer.android.com/guide/topics/providers/content-providers.html#urisum) before continuing on. This method should execute one of two different queries on its SQLiteDatabase member variable depending on the Uri argument that was passed in:If the Uri parameter is of the form content://edu.calpoly.android.provider.apprater/apps then it should query for and return a cursor of all records that match the search criteria.This is similar to the JokeDBAdapter.getAllJokes(String ratingFilter) method.If the Uri parameter is of the form content://edu.calpoly.android.provider.apprater/apps/<ID>, where <ID> is a number, then it should query for and return a cursor of only the record whose ID matches the <ID> value.This is similar to the JokeDBAdapter.getJoke(long id) method.Tips:Initialize and make use of the static AppContentProvider.s\_uriMatcher member to help with identifying which Uri you were given. You can read the Android Documentation on the [UriMatcher Class](http://developer.android.com/reference/android/content/UriMatcher.html) to learn how to initialize and use it.Make sure to properly check for incorrect Uri's.Read the Android Documentation on ContentProvider.query(...) if you are still unclear on what this method should do.4.1.4 Implement the Remaining ContentProvider MethodsYou should continue on to implement the insert(...), update(...), and delete(...) methods. Each of these methods should check the Uri parameter just like the query(...) method did and either insert/update/delete all matching records or a single record. Use the tips you were given in the previous section.   If you're at all confused about what the method should do, what the method should return, or what the parameters are for, read the Android Documentation on the [ContentProvider Class](http://developer.android.com/reference/android/content/ContentProvider.html).4.1.7 Use Your AppContentProviderThe AppRater class and AppDownloadService class are currently using the AppContentProvider implementation that was provided to you. You need to remove the import statements at the top of the AppRater.java and AppDownloadService.java files in order to use your implementation. Delete the following line from each of these files:import edu.calpoly.android.appraterkey.AppContentProvider;You should now test your application to ensure that it functions as it did before.4.1.8 Update Your ManifestYou will need to add your ContentProvider to the AndroidManifest.xml file. You can look in the file to see how it was done for the appraterkey.AppContentProvider. | |