About This Article

This article provides information about how to use Loaders in Android applications. Android application developers can use this article as reference to implement Loaders in their applications. Loaders were introduced from Android 3.0(Honeycomb).

Scope:

This article is intended for Android developers wishing to develop mobile applications. It assumes basic knowledge of Android and Java programming languages.

To find out more about Android, please refer to the Knowledge Base under Samsung Developers. https://developer.samsung.com/android

Introduction

Loading data or working on lengthy operations in a UI thread is not recommended. Loading data in a UI thread may lead to Application Not Responding (ANR). To avoid this ANR, develop the applications to perform long running tasks in a separate thread. For example, long running operations may retrieve the data from the content provider or from a network instead.

From Android 3.0 (Honeycomb) onwards, Loaders were introduced to achieve the same results. The Loaders also useAsyncTask to perform the background work. So you cannot expect a performance gain in loaders over AsyncTask, but loaders make it easier to implement data loading and manage lifecycle events.

Characteristics of Loaders

Loaders have the following properties:

- Provide asynchronous data loading
- Monitor the source of the data and updates new results when content changes
- Are available for every Activity and Fragment
- Automatically reconnect to the last loader's cursor when being recreated after a configuration change such as soft keyboard popup or orientation change. So Loaders do not need to re-query the same data.

This article provides information on how to use loaders in an application to achieve better performance. Take a look at few API’s for implementing loaders in your application.

API Summary

LoaderManager

This abstract class is used in Activity or Fragment to manage the different loader instances. LoaderManager also helps the application to manage the background operations as per the Activity/Fragment life cycle.

Loader

This abstract class is responsible for loading the data asynchronously. While loaders are active they should monitor the source of their data and update the results when contents change.

AsyncTaskLoader

An abstract loader that provides an AsyncTask to do the long running operations or data loading.

Cursor Loader

The CursorLoader is the best way to load data asynchronously from the content providers. It is a subclass ofAsyncTaskLoader.

LoaderManager.LoaderCallbacks

The client requires a callback interface to interact with the LoaderManager. The client implements the following callbacks:

- onCreateLoader(): Creates and returns a new Loader for a given ID.
- onLoadFinished(): Callback for previously created loader that has finished the assigned work.
- onLoaderReset(): Callback for previously created loader if reset.

To know more about the Loader API referhttp://developer.android.com/guide/topics/fundamentals/loaders.html

Using Loaders in an Application

An application that uses loaders consists the following:

- An Activity or Fragment
- An instance of the LoaderManager
- A CursorLoader to load data backed by a ContentProvider
- An implementation for LoaderManager.LoaderCallbacks
- A way of displaying the loader's data as listview.

Example

This example shows all the SMS messages available in the handset. To accomplish this, the database loads the SMS using loaders. Here’s step-by-step explanation. To show SMS, you need list view. You can use a list view in Activity or you can take advantage from the ListActivity. The code snippet shows the class file:

```
public class LoadersSampleActivity extends ListActivity
        {
                @Override
                public void onCreate(Bundle savedInstanceState) {
                        super.onCreate(savedInstanceState);

                        //setContentView(R.layout.main);


                }
        }
```

Now, populate the list view with the existing SMS messages. In this example for populating the list view with SMS you need a CursorAdapter to set to the setListAdapter() method. The following code snippet shows the implementation. You need to place this in an OnCreate() method.

```
// Create an empty adapter we will use to display the loaded data.
        mAdapter = new SimpleCursorAdapter(this,
                                           android.R.layout.simple_list_item_2, null,
                                           new String[] {"address","body"},
                                           new int[] { android.R.id.text1,
```

```
android.R.id.text2 }, 0);
        setListAdapter(mAdapter);
```

Set the empty adapter to the list as of now. Populate the adapter with the list of SMS messages available in the handset. Here you are going to use the Loaders concept apart from the traditional AsyncTask or Threads/Handlers mechanism. The following code snippet shows how to initiate the loaders:

```
getLoaderManager().initLoader(0, null, this);
```

The first argument is a unique ID that identifies the loader, second one is optional argument to supply to the loader at construction, and the third one is the LoaderManager.LoaderCallbacks implementation, which the LoaderManagercalls to report the loader events.

The local class implements the LoaderManager.LoaderCallbacks interface, so it passes a reference to itself, this. The below code snippet shows the implementation of LoaderCallbacks interface.

```
public class LoadersSampleActivity extends ListActivity implements
LoaderCallbacks<cursor></cursor>
        {
                @Override
                public void onCreate(Bundle savedInstanceState) {
                        super.onCreate(savedInstanceState);

                //setContentView(R.layout.main);

                // Create an empty adapter we will use to display the loaded data.
                mAdapter = new SimpleCursorAdapter(this,
                                                android.R.layout.simple_list_item_2,
null,
                                                new String[] {"address","body"},
                                                new int[] { android.R.id.text1,
android.R.id.text2 }, 0);
                setListAdapter(mAdapter);


                //Initiating the loader
                getLoaderManager().initLoader(0, null, this);
                }
                @Override
                public Loader<cursor></cursor> onCreateLoader(int id, Bundle args) {
                        // TODO Auto-generated method stub

                }

                @Override
                public void onLoadFinished(Loader<cursor></cursor> arg0, Cursor arg1) {
                        // TODO Auto-generated method stub
```

```
                }

                @Override
                public void onLoaderReset(Loader<cursor<>/cursor> arg0) {
                          // TODO Auto-generated method stub

                }
        }
```

As stated previously, the onCreateLoader() method activates once the called initLoader() method. Return a Loader for the given ID from this callback method. In this example, you need to return a SMS Loader. The following code snippet shows retrieving the SMS messages and returning the CursorLoader.

```
@Override
        public Loader<cursor></cursor> onCreateLoader(int id, Bundle args) {
        // TODO Auto-generated method stub
                Uri baseUri = Uri.parse("content://sms");

                   // Now create and return a CursorLoader that will take care of
                   // creating a Cursor for the data being displayed.

                   return new CursorLoader(this, baseUri,
                                    null, null, null,null);
        }
```

Another callback method onLoadFinished() activates when the onCreateLoader() returns the loader. In this method you need to populate the empty adapter with the cursor. The following code snippet shows populating the adapter:

```
@Override
                public void onLoadFinished(Loader<cursor></cursor> arg0, Cursor arg1) {
                         // TODO Auto-generated method stub
                          mAdapter.swapCursor(arg1);
                }
```

The other callback method onLoaderReset() gets activated when a previously created loader is being reset by calling the restartLoader() method in ListActivity. In this example, you do not call the restartLoader() method but it’s better to set the adapter to null. The following code snippet shows how to set the adapter to null:

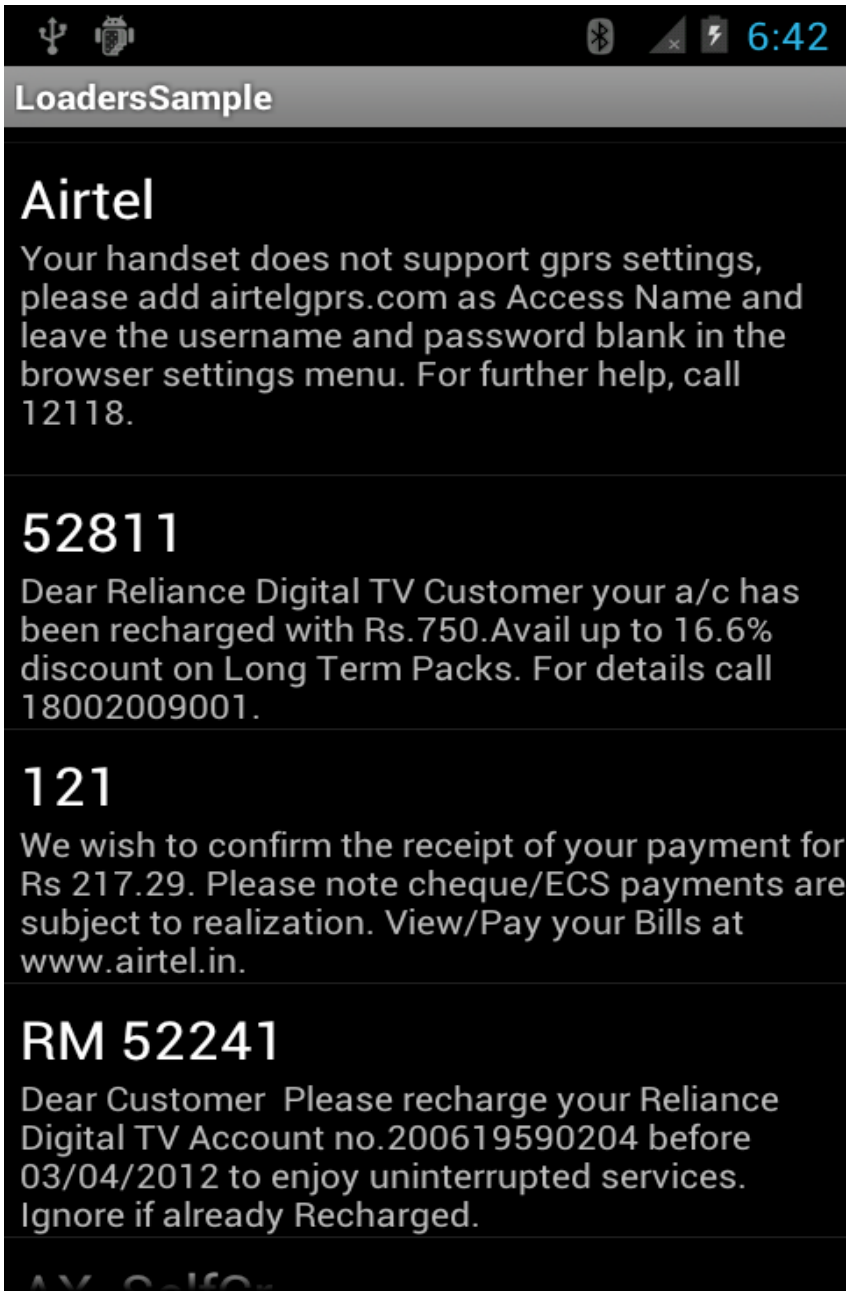
```
@Override
        public void onLoaderReset(Loader<cursor></cursor> arg0) {
                // TODO Auto-generated method stub
                 mAdapter.swapCursor(null);
        }
```

Using the loaders you have implemented everything to show the contacts in list view. But before running the application you need to add the permission READ_SMS in Android manifest file for reading SMS messages.

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

Now run the application and the following sample screen displays the SMS messages depending on

your handset.



ref:http://developer.samsung.com/android/technical-docs/Loaders-in-Android