ANDROID RSS READER TUTORIAL

RSS stands for Really Simple Syndication. RSS is an easy way to share your website updates and content with your users so that users might not have to visit your site daily for any kind of updates.

RSS Example

RSS is a doucment that is created by the website with .xml extension. You can easily parse this document and show it to the user in your application. An RSS document looks like this.

```
<rss version="2.0">
<channel>
   <title>Sample RSS</title>
   <link>http://www.google.com</link>
   <description>World's best search engine</description>
</channel>
</rss>
```

RSS Elements

An RSS document such as above has the following elements.

Sr.No	Component & description
1	**channel** This element is used to describe the RSS feed
2	**title** Defines the title of the channel
3	**link** Defines the hyperlink to the channel
4	**description** Describes the channel

Parsing RSS

Parsing an RSS dcument is more like parsing XML. So now lets see how to parse an XML document.

For this, We will create XMLPullParser object , but in order to create that we will first create XmlPullParserFactory object and then call its newPullParser() method to create XMLPullParser. Its syntax is given below:

```
private XmlPullParserFactory xmlFactoryObject = XmlPullParserFactory.newInstance();
private XmlPullParser myparser = xmlFactoryObject.newPullParser();
```

The next step involves specifying the file for XmlPullParser that contains XML. It could be a file or could be a Stream. In our case it is a stream.Its syntax is given below:

```
myparser.setInput(stream, null);
```

The last step is to parse the XML. An XML file consist of events , Name , Text , AttributesValue e.t.c. So XMLPullParser has a seperate function for parsing each of the component of XML file. Its syntax is given below:

```
int event = myParser.getEventType();
while (event != XmlPullParser.END_DOCUMENT)
{
   String name=myParser.getName();
   switch (event){
      case XmlPullParser.START_TAG:
      break;
      case XmlPullParser.END_TAG:
      if(name.equals("temperature")){
         temperature = myParser.getAttributeValue(null,"value");
       }
       break;
    }
    event = myParser.next();
}
```

The method **getEventType** returns the type of event that happens. e.g: Document start , tag start e.t.c. The method **getName** returns the name of the tag and since we are only interested in temperature , so we just check in conditional statement that if we got a temperature tag , we call the method **getAttributeValue** to return us the value of temperature tag.

Apart from the these methods , there are other methods provided by this class for better parsing XML files. These methods are listed below:

Sr.No	Method & description
1	**getAttributeCount()** This method just Returns the number of attributes of the current start tag.
2	**getAttributeName(int index)** This method returns the name of the attribute specified by the index value.
3	**getColumnNumber()** This method returns the Returns the current column number, starting from 0.
4	**getDepth()** This method returns Returns the current depth of the element.
5	**getLineNumber()** Returns the current line number, starting from 1.
6	**getNamespace()** This method rReturns the namespace URI of the current element.
7	**getPrefix()** This method returns the prefix of the current element.

8	**getName()** This method returns the name of the tag.
9	**getText()** This method returns the text for that particular element.
10	**isWhitespace()** This method checks whether the current TEXT event contains only whitespace characters.

Example

Here is an example demonstrating the use of XMLPullParser class. It creates a basic Parsing application that allows you to parse an RSS document present here at http://tutorialspoint.com/android/sampleXML.xml and then show the result.

To experiment with this example , you can run this on an actual device or in an emulator.

Steps	Description
1	You will use Eclipse IDE to create an Android application and name it as RSSReader under a package com.example.rssreader. While creating this project, make sure you Target SDK and Compile With at the latest version of Android SDK to use higher levels of APIs.
2	Modify src/MainActivity.java file to add necessary code.
3	Modify the res/layout/activity_main to add respective XML components.
4	Modify the res/values/string.xml to add necessary string components.
5	Create a new java file under src/HandleXML.java to fetch and parse XML data.
6	Modify AndroidManifest.xml to add necessary internet permission.
7	Run the application and choose a running android device and install the application on it and verify the results.

Following is the content of the modified main activity file **src/com.example.rssreader/MainActivity.java**.

```
package com.example.rssreader;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.widget.EditText;

public class MainActivity extends Activity {

   private String finalUrl="http://tutorialspoint.com/android/sampleXML.xml";
   private HandleXML obj;
   private EditText title,link,description;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      title = (EditText)findViewById(R.id.editText1);
      link = (EditText)findViewById(R.id.editText2);
      description = (EditText)findViewById(R.id.editText3);
   }

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      // Inflate the menu; this adds items to the action bar if it is present.
      getMenuInflater().inflate(R.menu.main, menu);
      return true;
   }
   public void fetch(View view){
      obj = new HandleXML(finalUrl);
      obj.fetchXML();
      while(obj.parsingComplete);
         title.setText(obj.getTitle());
         link.setText(obj.getLink());
         description.setText(obj.getDescription());
   }
}
```

Following is the content of the java file **src/com.example.rssreader/HandleXML.java**.

```
package com.example.rssreader;

import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URL;

import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserFactory;

import android.util.Log;

public class HandleXML {

   private String title = "title";
   private String link = "link";
   private String description = "description";

   private String urlString = null;
   private XmlPullParserFactory xmlFactoryObject;
   public volatile boolean parsingComplete = true;
   public HandleXML(String url){
      this.urlString = url;
   }
   public String getTitle(){
      return title;
   }
   public String getLink(){
      return link;
   }
   public String getDescription(){
      return description;
   }
   public void parseXMLAndStoreIt(XmlPullParser myParser) {
      int event;
      String text=null;
      try {
         event = myParser.getEventType();
         while (event != XmlPullParser.END_DOCUMENT) {
         String name=myParser.getName();
         switch (event){
            case XmlPullParser.START_TAG:
                break;
            case XmlPullParser.TEXT:
                       text = myParser.getText();
            break;
            case XmlPullParser.END_TAG:
                   if(name.equals("title")){
                  title = text;
               }
               else if(name.equals("link")){
                  link = text;
               }
               else if(name.equals("description")){
                  description = text;
               }
               else{
               }
               break;
         }
         event = myParser.next();
       }
       parsingComplete = false;
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
   public void fetchXML(){
   Thread thread = new Thread(new Runnable(){
   @Override
   public void run() {
      try {
         URL url = new URL(urlString);
         HttpURLConnection conn = (HttpURLConnection) url.openConnection();
         conn.setReadTimeout(10000 /* milliseconds */);
         conn.setConnectTimeout(15000 /* milliseconds */);
         conn.setRequestMethod("GET");
         conn.setDoInput(true);
         // Starts the query
         conn.connect();
         InputStream stream = conn.getInputStream();
         xmlFactoryObject = XmlPullParserFactory.newInstance();
         XmlPullParser myparser = xmlFactoryObject.newPullParser();
         myparser.setFeature(XmlPullParser.FEATURE_PROCESS_NAMESPACES, false);
         myparser.setInput(stream, null);
         parseXMLAndStoreIt(myparser);
         stream.close();
      } catch (Exception e) {
      }
      }
      });
      thread.start();
   }
}
```

Modify the content of **res/layout/activity_main.xml** to the following:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context=".MainActivity" >

   <TextView
      android:id="@+id/textView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="26dp"
      android:text="@string/hello_world"
      android:textAppearance="?android:attr/textAppearanceLarge" />

   <TextView
      android:id="@+id/textView2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_below="@+id/textView1"
      android:layout_marginTop="48dp"
      android:layout_toLeftOf="@+id/textView1"
      android:text="@string/title" />

   <TextView
      android:id="@+id/textView3"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignLeft="@+id/textView2"
      android:layout_below="@+id/textView2"
      android:layout_marginTop="27dp"
      android:text="@string/link" />
   <EditText
      android:id="@+id/editText1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignBaseline="@+id/textView2"
      android:layout_alignBottom="@+id/textView2"
      android:layout_alignParentRight="true"
      android:ems="10" >
   <requestFocus />
   </EditText>

   <EditText
      android:id="@+id/editText2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignBaseline="@+id/textView3"
      android:layout_alignBottom="@+id/textView3"
      android:layout_alignLeft="@+id/editText1"
      android:ems="10" >
   </EditText>

   <EditText
```

```
      android:id="@+id/editText3"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignBaseline="@+id/textView4"
      android:layout_alignBottom="@+id/textView4"
      android:layout_alignLeft="@+id/editText2"
      android:ems="10" />

   <Button
      android:id="@+id/button1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_below="@+id/editText3"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="37dp"
      android:onClick="fetch"
      android:text="@string/fetch" />

   <TextView
      android:id="@+id/textView4"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentLeft="true"
      android:layout_centerVertical="true"
      android:text="@string/description" />
</RelativeLayout>
```

Modify the **res/values/string.xml** to the following

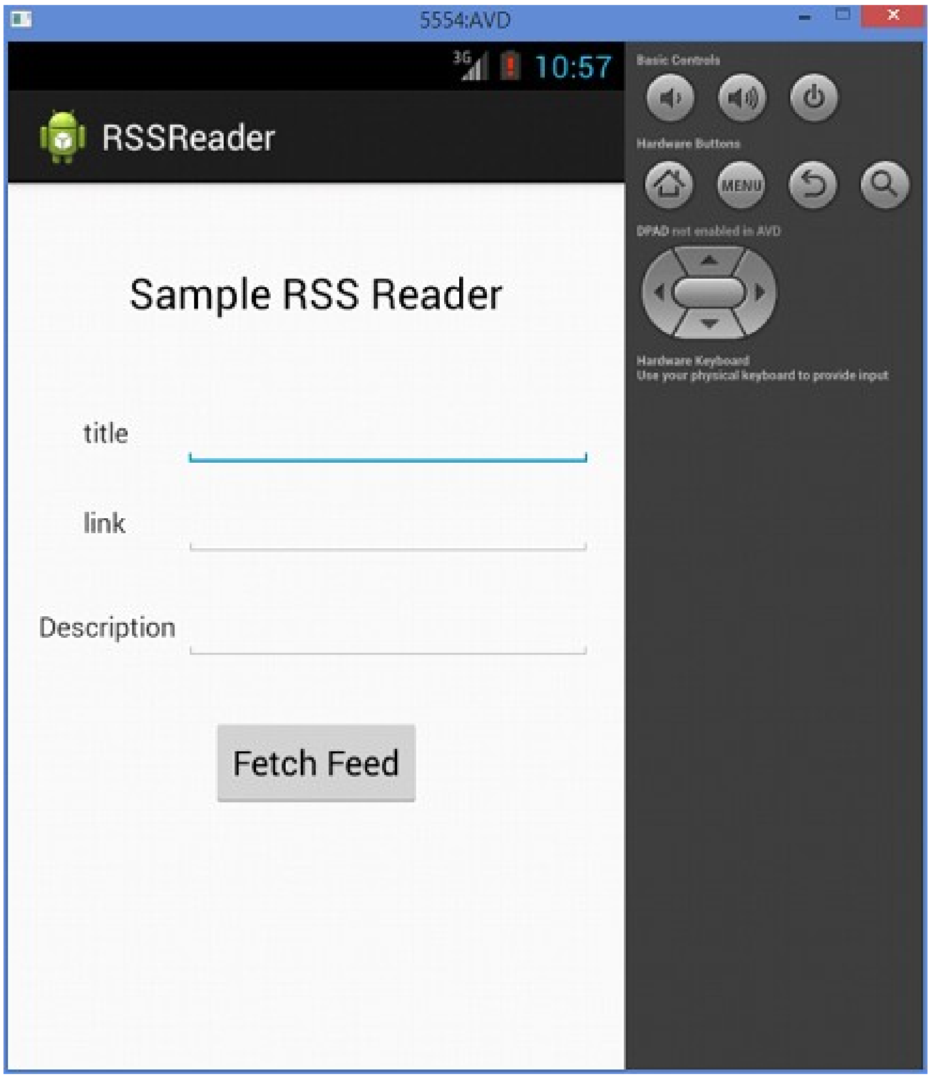
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="app_name">RSSReader</string>
   <string name="action_settings">Settings</string>
   <string name="hello_world">Sample RSS Reader</string>
   <string name="title">title</string>
   <string name="link">link</string>
   <string name="description">Description</string>
   <string name="fetch">Fetch Feed</string>
</resources>
```

This is the default **AndroidManifest.xml**.

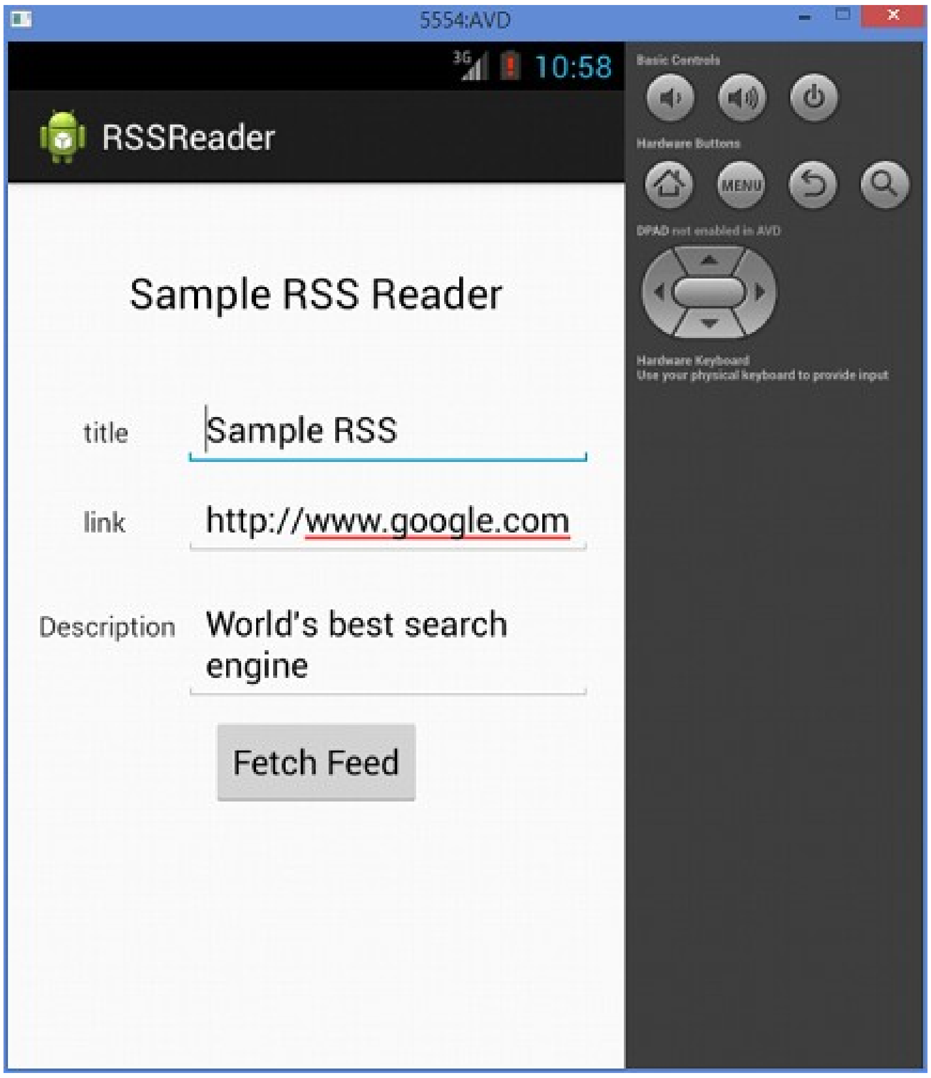
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.rssreader"
   android:versionCode="1"
   android:versionName="1.0" >

   <uses-sdk
      android:minSdkVersion="8"
      android:targetSdkVersion="17" />
      <uses-permission android:name="android.permission.INTERNET"/>
   <application
      android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >
      <activity
         android:name="com.example.rssreader.MainActivity"
         android:label="@string/app_name" >
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
   </application>

</manifest>
```

Let's try to run your RSSReader application.I assume you had created your **AVD** while doing environment setup. To run the app from Eclipse, open one of your project's activity files and click Run

icon from the toolbar. Eclipse installs the app on your AVD and starts it and if everything is fine with your setup and application, it will display following Emulator window:



Just press the Fetch Feed button to fetch RSS feed. After pressing , following screen would appear which would show the RSS data.



Sample RSS Reader

title Sample RSS

link http://www.google.com

Description World's best search engine

Fetch Feed