MOBILE DEVELOPMENT

CONSUMING RSS FEEDS

CONTENTS

WHAT IS AN RSS FEED?

RSS Feeds define a structured world-wide distribution system in which users subscribe to a source in order to pull in XML formatted online content.

Typical RSS sources include:

- news organizations,
- weather,
- financial services,
- public services,
- customer services,
- marketing & advertisement,
- blogs and
- video providers.

Why ?
RSS feeds keep users informed about subjects of interest to them.

Ref: Channel Definition Format (CDF) http://www.w3.org/TR/NOTE-CDFsubmit.html

WHAT IS AN RSS FEED?

First version of RSS was created by Netscape around 1999.

Often called “Really Simple Syndication”

A typical news feed (or channel) contains entries which may be:

- headlines,
- full-text articles excerpts,
- summaries,
- Thumbnails, and/or
- links to content on a website along with various metadata

Atom Syndication Format and RSS are common XML standards used to organize, create and update web feeds (these formats have been adopted by Google, Yahoo!, Apple/iTunes, CNN, NY Times…)

Validity of ATOM/RSS documents can be tested at http://validator.w3.org/appc/ (many other tools are available)

STRUCTURE OF RSS FEEDS

Figure 1: An RSS feed is an XML document that consists of a <channel> and zero or more <item> elements.

```
<rss>
 <channel>
  Channel_Elements
  <item> Item1 <\item>
  <item> Item2 <\item>
 </channel>
</rss>
```

STRUCTURE OF RSS FEEDS (<channel>)

Elements	Description	Type	#allowed
LastMod	Last modified date for this web page	ISO 8601:1988 Date	0 or 1
Title	Title	String	0 or 1
Abstract	Short description summarizing the article (200 characters or less recommended)	String	0 or 1
Author	Author	String	Any
Publisher	Publisher	String	Any
Copyright	Copyright	String	0 or 1
PublicationDate	Publication Date	String	0 or 1
Logo	Visual Logo for channel	Logo element	Any
Keywords	Comma delimited keywords that match this channel	String	Any
Category	A category to which this web page belongs in (as an URI).	Category element	Any
Ratings	Rating of the channel by one or more ratings services.	String	Any
Schedule	Schedule for keeping channel up to date	Schedule element	0 or 1
UserSchedule	Reference to a client/user specified schedule	UserSchedule element	0 or 1

Reference: http://www.w3.org/TR/NOTE-CDFsubmit.html

STRUCTURE OF RSS FEEDS (<item>)

A channel may contain any number of <item>s. An item may represent a “story” - similar to a story in a newspaper or magazine.

Elements	Description
title	The title of the item.
link	The URL of the item.
description	The item synopsis.
author	Email address of the author of the item.
category	Includes the item in one or more categories.
comments	URL of a page for comments relating to the item.
enclosure	Describes a media object that is attached to the item.
guid	A string that uniquely identifies the item.
pubDate	Indicates when the item was published.
source	The RSS channel that the item came from.

STRUCTURE OF RSS FEEDS (Example)

```
<?xml version="1.0" encoding="utf-8" ?>
<rss version="2.0" xmlns:atom="http://www.w3.org/2005/atom" >
 <channel>
  <title>rss title goes here...</title>
  <description>a description goes here...</description>
  <link>http://www.publisherSite.com/index.html</link>
  <lastbuilddate>mon, 05 jul 2014 10:15:00 -0200</lastbuilddate>
  <pubdate>tue, 06 jul 2014 12:00:00 -0200</pubdate>
  <item>
   <title>Item's title goes here...</title>
   <description>item's synopsis goes here...</description>
   <link>http://www.moreAboutItemLink.org/</link>
   <guid>http://www.publisherSite.com/archives/id000123.html</guid>
   <pubdate>wed, 07 jul 2014 12:00:15 -0200</pubdate>
  </item>
 </channel>
</rss>
```

STRUCTURE OF RSS FEEDS (using the <![CDATA[. . .]]> tag)

You may simplify the <description> portion of an <item> by entering non-escaped HTML text inside a CDATA tag.

For example, if your item's text is literally: This is bold then the escaped <description> would be:

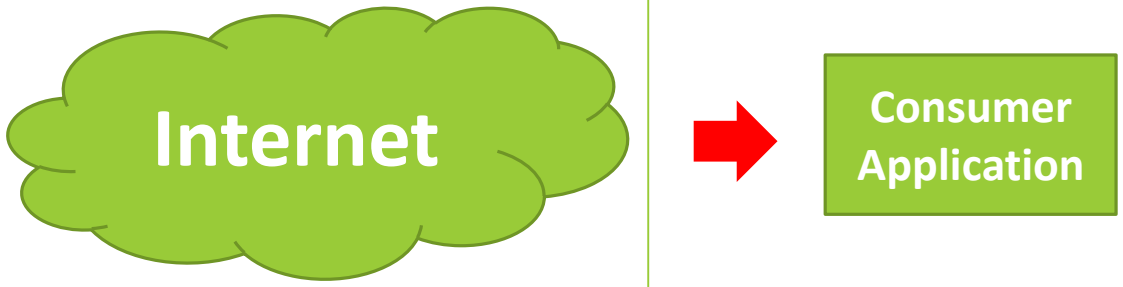
```
<description>This is &lt;b&gt;bold&lt;/b&gt;</description>
```

In the example "<" becomes "<" and ">" turns into ">". The equivalent version using the XML CDATA tag would be:

```
<description><![CDATA[This is <b>bold</b>]]></description>
```

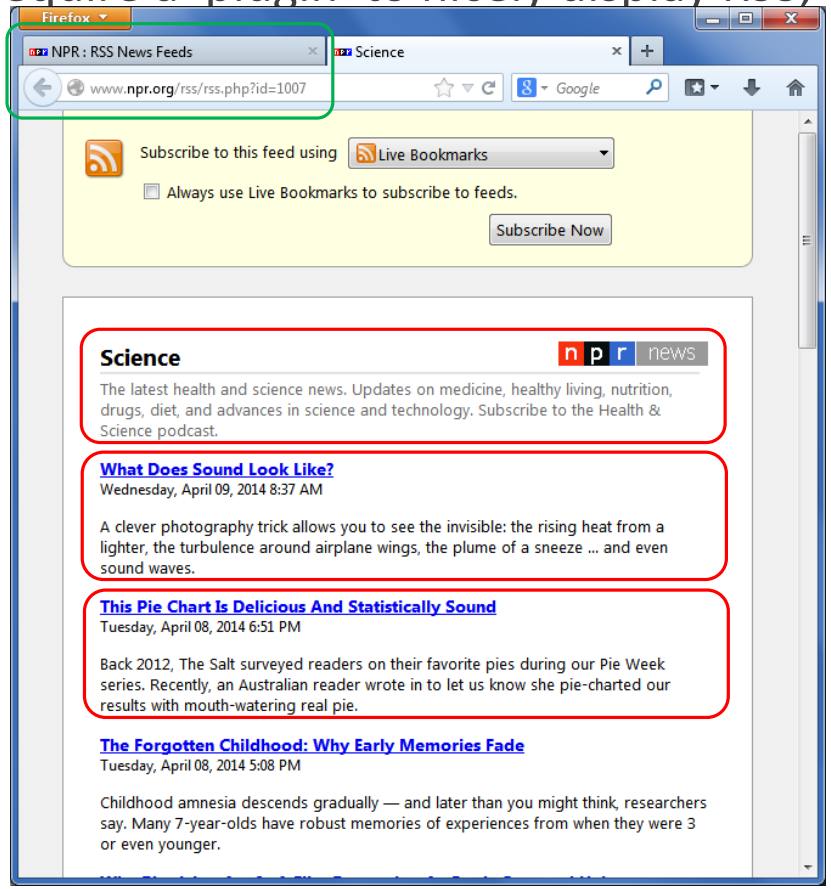
STRUCTURE OF RSS FEEDS (Sample of RSS aggregators)

World weather: http://www.rssweather.com/dir
US weather: http://www.weather.gov/view/national.php?map=on
The Weather Channel: http://rss.weather.com/weather/rss/local/44114
News:
http://www.npr.org/rss/
http://www.cnn.com/services/rss/
http://news.bbc.co.uk/2/hi/help/3223484.stm
http://www.nytimes.com/services/xml/rss
Money Exchange: http://themoneyconverter.com/RSSFeeds.aspx
Entertainment:
http://www.nbclosangeles.com/rss/
http://www.movies.com/rss/
RSS Aggregator:
http://www.rss-network.com/
http://www.nytimes.com/services/xml/rss
Corporate:
http://www.toyota.co.jp/en/rss/rss-responsibility.html
http://home3.americanexpress.com/corp/rss/
http://www.aa.com/i18n/urls/rss.jsp
http://www.amazon.com/gp/tagging/rss-help.html



STRUCTURE OF RSS FEEDS
(How do RSS feeds look like when using a browser?)

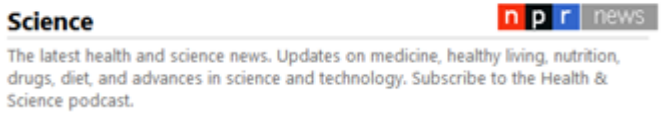
Note: Your browser may require a ‘plugin’ to nicely display RSS, otherwise it may show plain XML text.



NPR National Public Radio (9-Apr-2014)

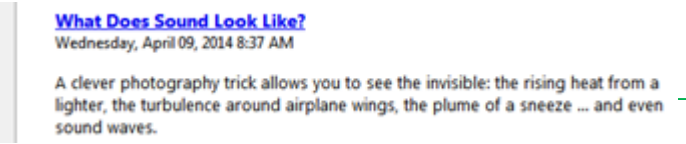
STRUCTURE OF RSS FEEDS (XML version of NPR RSS feed - fragment 1/3)


```
<?xml version="1.0" encoding="UTF-8"?>
<rss xmlns:npr="http://www.npr.org/rss/" xmlns:nprml="http://api.npr.org/nprml"
        xmlns:itunes="http://www.itunes.com/dtds/podcast-1.0.dtd"
        xmlns:content="http://purl.org/rss/1.0/modules/content/" version="2.0">
 <channel>
  <title>Science</title>
  <link>http://www.npr.org/templates/story/story.php?storyId=1007&amp;</link>
  <description>The latest health and science news. Updates on medicine, healthy living, nutrition, drugs, diet, and
advances in science and technology. Subscribe to the Health &amp; Science podcast.</description>
  <language>en</language>
  <copyright>Copyright 2014 NPR - For Personal Use Only</copyright>
  <generator>NPR API RSS Generator 0.94</generator>
  <lastBuildDate>Tue, 09 Apr 2014 12:28:00 -0400</lastBuildDate>
  <image>
   <url>http://media.npr.org/images/npr_news_123x20.gif</url>
   <title>Science</title>
   <link>http://www.npr.org/templates/story/story.php?storyId=1007&amp;ft=1&amp;f=1007</link>
  </image>
```

STRUCTURE OF RSS FEEDS (XML version of NPR RSS feed - fragment 2/3)

```
<item>
  <title>What Does Sound Look Like?</title>
  <description>
    A clever photography trick allows you to see the invisible: the rising heat from a lighter, the turbulence around airplane
    wings, the plume of a sneeze ... and even sound waves.
  </description>
  <pubDate>Wed, 09 Apr 2014 08:37:19 -0400</pubDate>
  <link>http://www.npr.org/2014/04/09/300563606/what-does-sound-look-like?ft=1&amp;f=1007</link>
  <guid>http://www.npr.org/2014/04/09/300563606/what-does-sound-look-like?ft=1&amp;f=1007</guid>
  <content:encoded>
    <![CDATA[
      <p>A clever photography trick allows you to see the invisible: the rising heat from a lighter, the turbulence around airplane
wings, the plume of a sneeze ... and even sound waves.
      </p>
      <p><a href="http://www.npr.org/templates/email/emailAFriend.php?storyId=300563606">&raquo; E-Mail This</a></p>]]>
  </content:encoded>
</item>
```



STRUCTURE OF RSS FEEDS (XML version of NPR RSS feed - fragment 2/3)

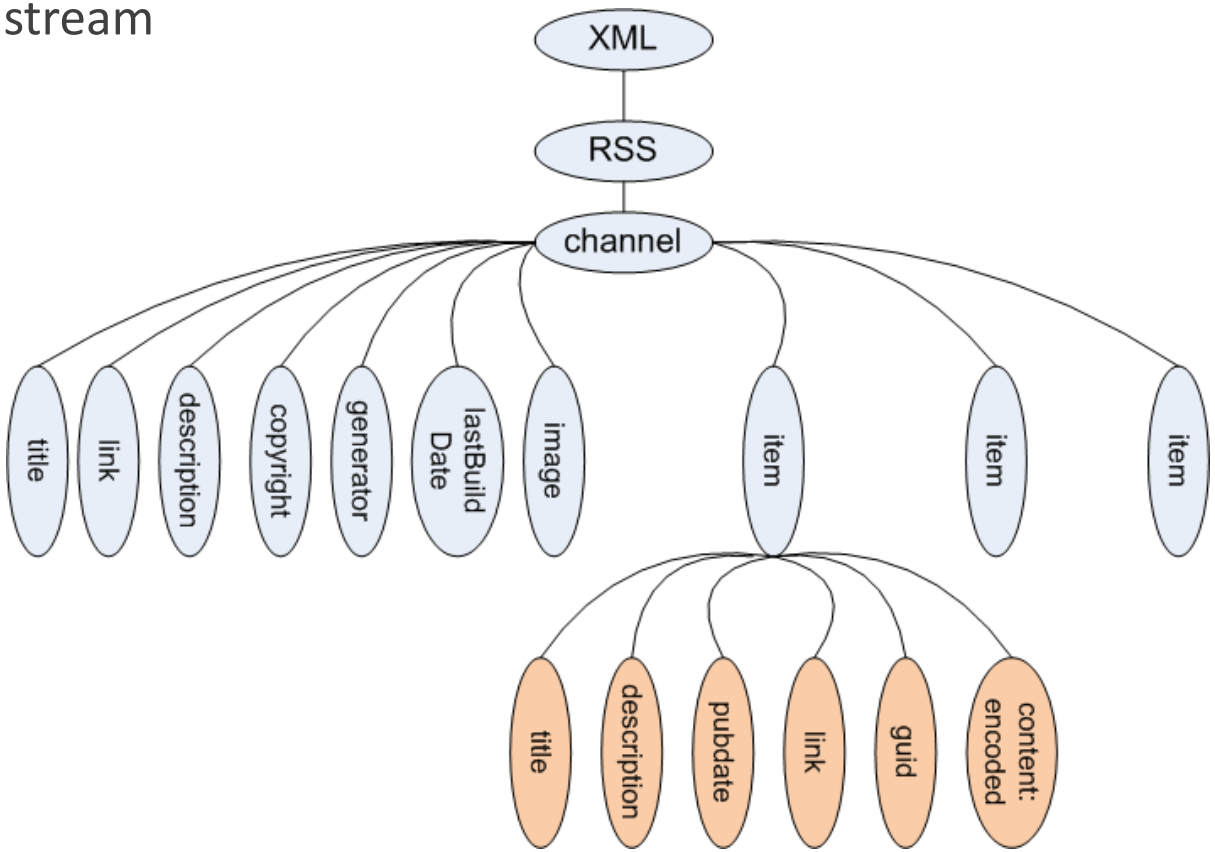

```
  <item>
   <title>This Pie Chart Is Delicious And Statistically Sound</title>
   <description>
    Back 2012, The Salt surveyed readers on their favorite pies during our Pie Week series. Recently, an Australian reader wrote in
    to let us know she pie-charted our results with mouth-watering real pie.
   </description>
   <pubDate>Tue, 08 Apr 2014 18:51:00 -0400</pubDate>
   <link>http://www.npr.org/blogs/thesalt/2014/04/08/300620654/this-pie-chart-is-delicious-and-statistically-sound?ft=1&amp;f=1007</link>
   <guid>http://www.npr.org/blogs/thesalt/2014/04/08/300620654/this-pie-chart-is-delicious-and-statistically-sound?ft=1&amp;f=1007</guid>
   <content:encoded><![CDATA[
    <p>Back 2012, The Salt surveyed readers on their favorite pies during our Pie Week series. Recently, an Australian reader wrote in to let us know she pie-
charted our results with mouth-watering real pie. </p>
    <p><a href="http://www.npr.org/templates/email/emailAFriend.php?storyId=300620654">&raquo; E-Mail This</a></p>]]></content:encoded>
  </item>

 </channel>
</rss>
```


Many <item> s were intentionally removed to fit page size

DOM – DOCUMENT OBJECT MODEL

The Android API includes a DocumentBuilderFactory class to create DOM object trees from an XML input stream



DOM – DOCUMENT OBJECT MODEL

The Document Object Model (DOM) is a language-independent API that allows applications to make parsers to produce a tree-based representation of valid HTML and well-formed XML documents. DOM-trees are exposed as a collection of data Nodes

With the Document Object Model, programmers can build documents, navigate their structure, and add, modify, or delete elements and content.

```
DocumentBuilder db = DocumentBuilderFactory
                                    .newInstance()
                                    .newDocumentBuilder();
Document dom = db.parse(someHttpInputStream);
```

http://docs.oracle.com/javase/7/docs/api/javax/xml/parsers/DocumentBuilderFactory.html

DOM – DOCUMENT OBJECT MODEL

Example: the tree in previous figure contains a set of item nodes.

Assume dom is the DOM-tree made by parsing the input stream returned by an RSS aggregator.

Accessing item data could be done as follows

```
// define access to all nodes in the parse tree
Element treeElements = dom.getDocumentElement();
// look for individual news (“items” in this case)
// put items in a NodeList collection
NodeList itemNodes = treeElements.getElementsByTagName(“item”);
```

DOM – DOCUMENT OBJECT MODEL

Android's handling of HTTP network resources is typically done using either of the client-side included APIs

- 1. Standard Java network java.net package, and/or
- 2. Apache HttpClient library.

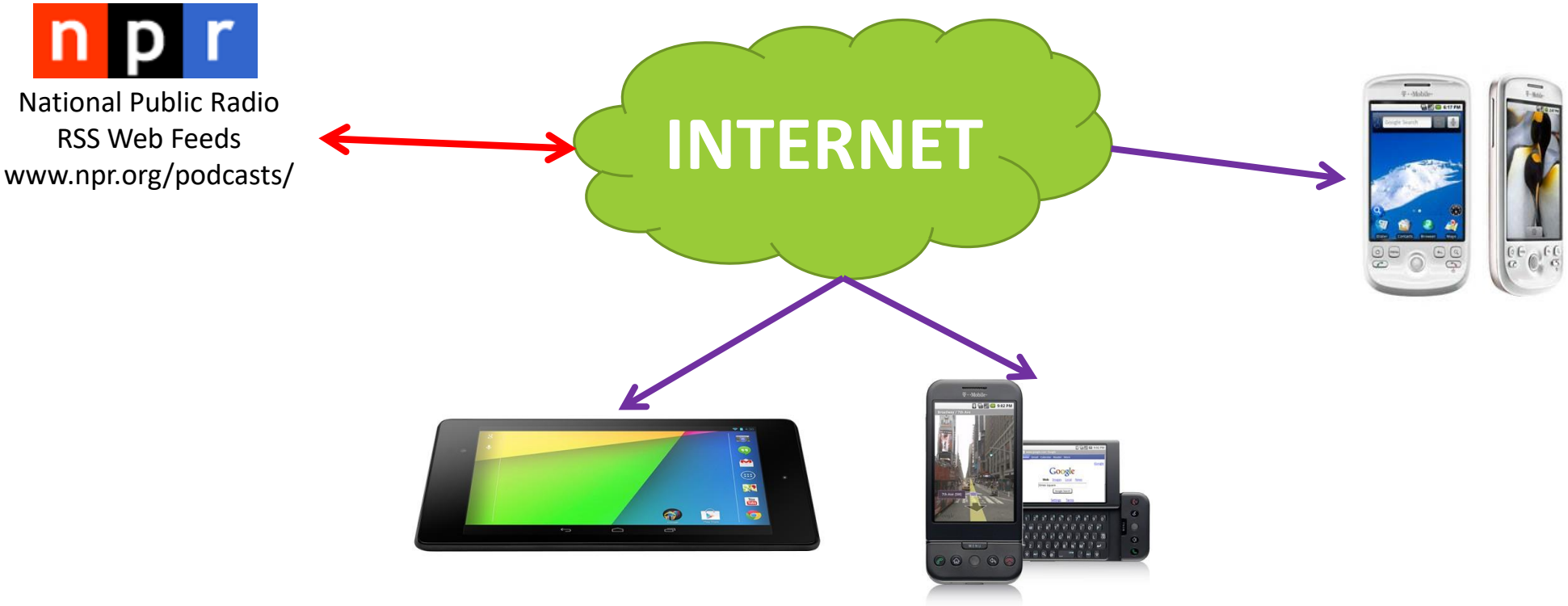
In particular, the often used java.net class HttpUrlConnection follows the next steps:

- 1. Obtain a new HttpURLConnection
- 2. Prepare the request (URI including header, credentials, content, cookies…)
- 3. Read the response (non-buffered stream returned by getInputStream())
- 4. Disconnect as soon as response is read.

http://docs.oracle.com/javase/6/docs/api/java/net/package-summary.html
http://hc.apache.org/httpcomponents-client-ga/

EXAMPLE (NPR PROJECT – ACTION PLAN)

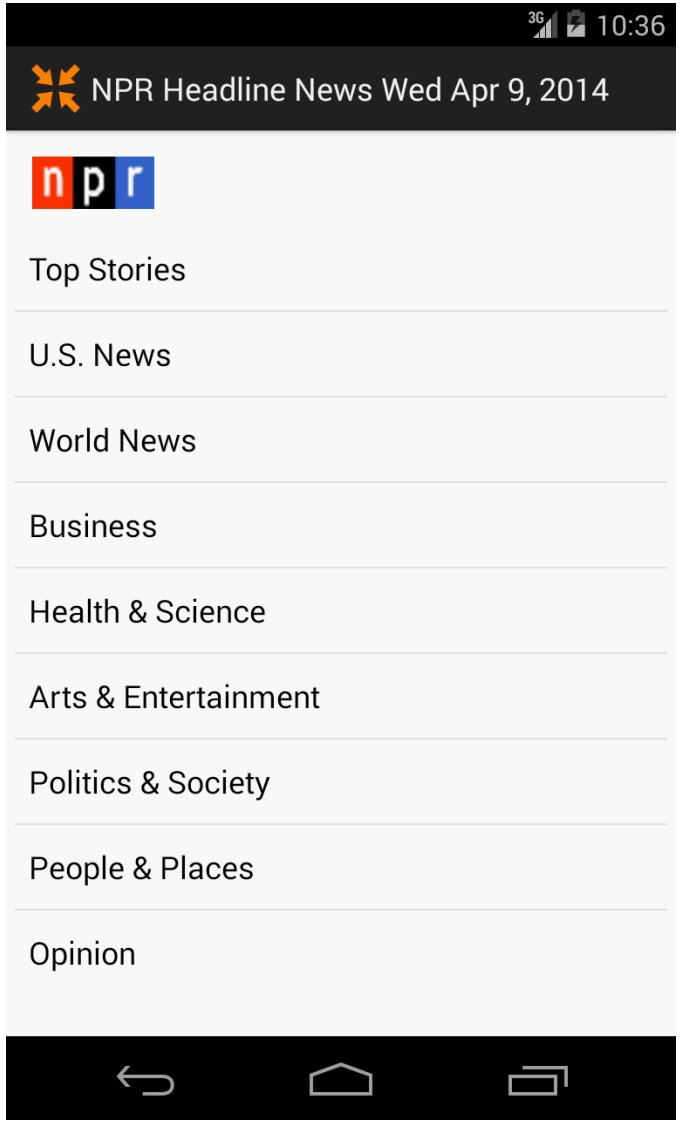
In this project we will develop an application to expose on Android devices the public-access RSS material aggregated by National Public Radio (NPR).



EXAMPLE (NPR PROJECT – ACTION PLAN)

Step1: a little research shows NPR supports a number of web feeds, among them the following:

Topic	URL
Business	https://feeds.npr.org/510289/podcast.xml
Comedy	https://feeds.npr.org/344098539/podcast.xml
Science	https://feeds.npr.org/510308/podcast.xml
Technology	https://feeds.npr.org/510298/podcast.xml
Music	https://feeds.npr.org/510306/podcast.xml
Kid & Family	https://feeds.npr.org/510354/podcast.xml
Society & culture	https://feeds.npr.org/510309/podcast.xml

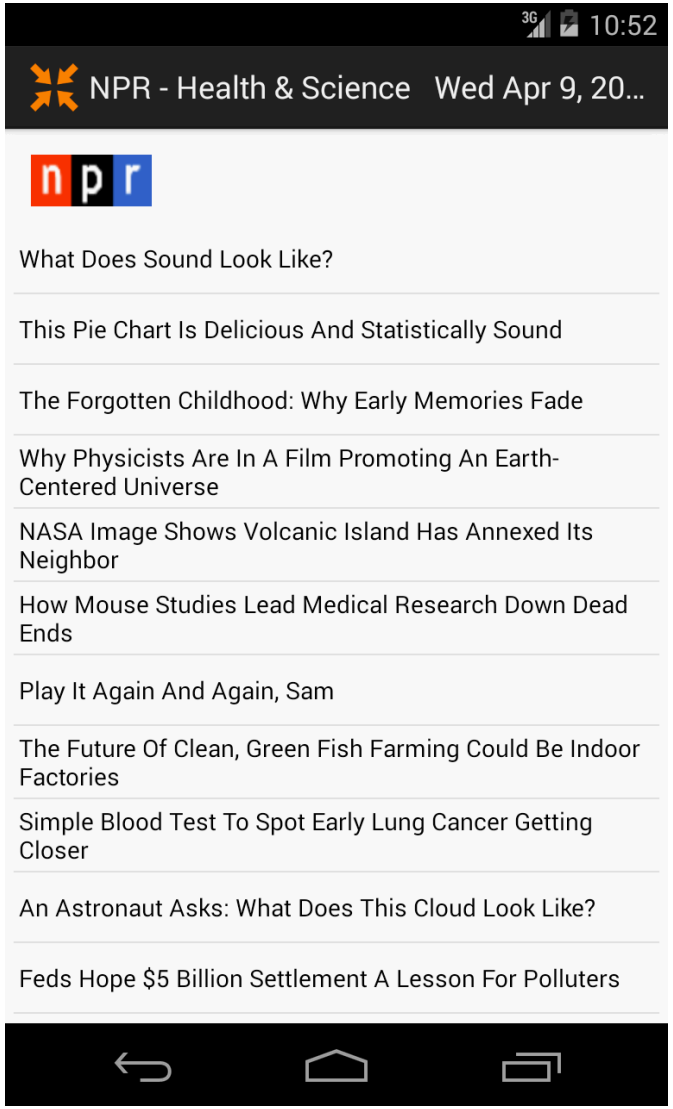


EXAMPLE (NPR PROJECT – ACTION PLAN)

Step2:

- we will display on a ListView widget, a basic menu consisting of a fixed set of topics (for instance: Top Stories, US News, World News, Business, etc)
- We wait for the user to make a selection. Once a category is chosen its corresponding headlines will be downloaded.

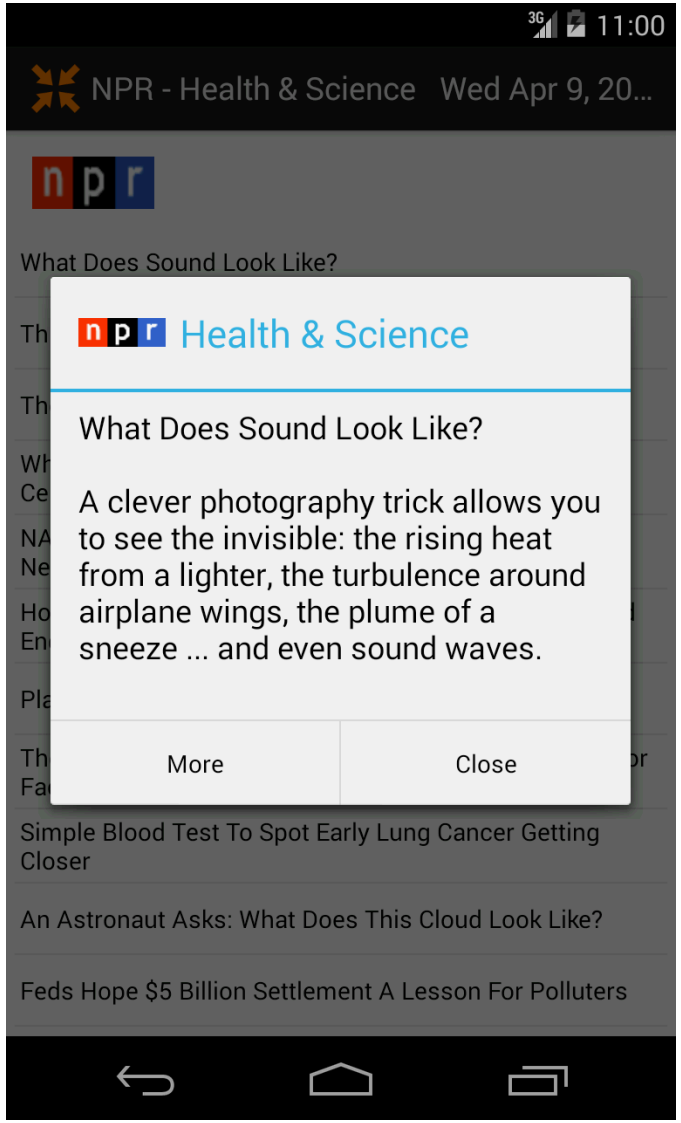
Choose "Health & Science"



EXAMPLE (NPR PROJECT – ACTION PLAN)

Step3:

- again, a simple ListView box is used to show the most current headlines from the selected category (notice the TextSize is now slightly smaller). The user can scroll the list and click on a particular story.
- Observe that individual lines in the ListView correspond to the feed's XML <item> entries discussed earlier.
- We have already expressed our interest in the "Health & Science" subject. Assume we want to follow the first article dealing with the 'shape of sounds'.

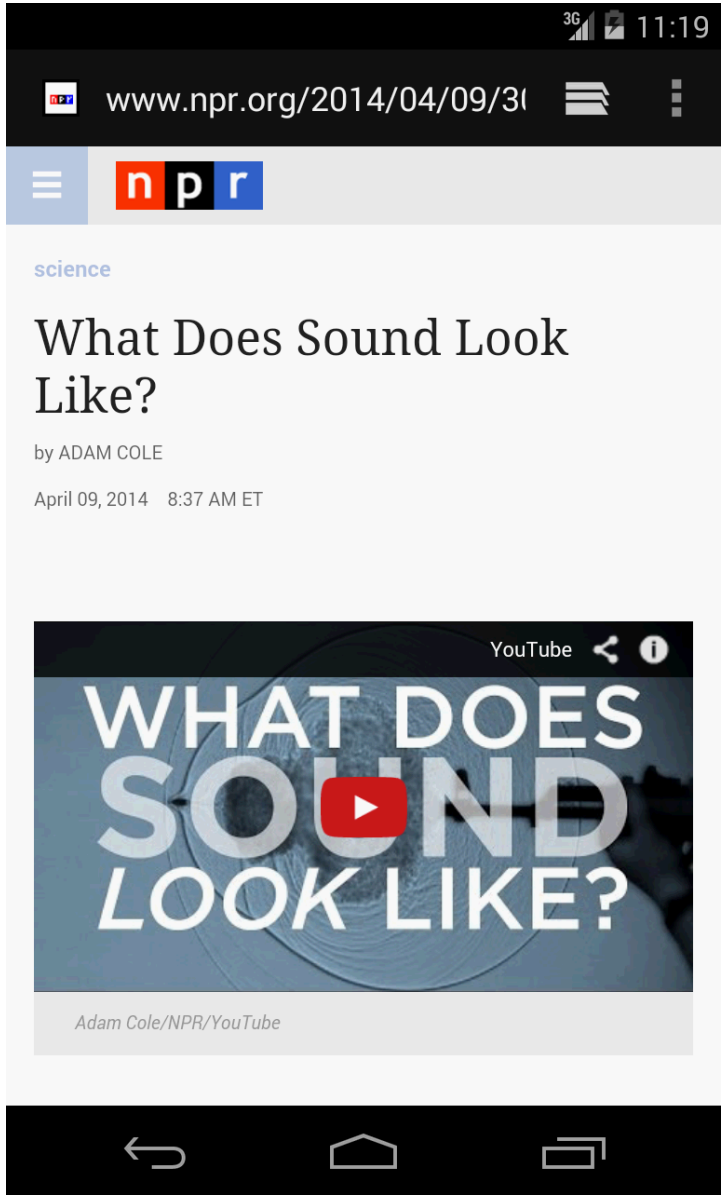


EXAMPLE (NPR PROJECT – ACTION PLAN)

Step 4:

- A brief summary of the chosen story is displayed inside a DialogBox (this material corresponds to a <content:encoded> tag held in the source web-feed).
- The user is given the option of closing the window or obtaining more information.

Assume we want additional information, so we click the “More” button



EXAMPLE (NPR PROJECT – ACTION PLAN)

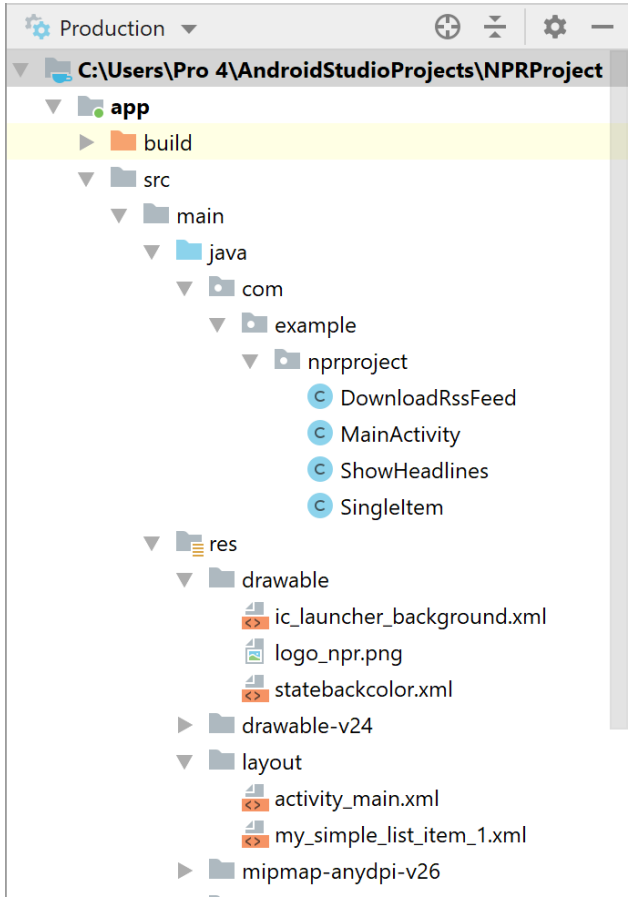
Step5.

- The <link> associated to the <item> that is currently displayed is given to a browser so the full document that is stored at the NPS site could be read.
- An internal browser on the given URL is started using a basic ACTION_VIEW Intent.
- To return to the app, the users taps on the BACK key

In addition to text, NPR stories often include images, videos, and sound clips; which are all available to the Android app.

EXAMPLE (NPR PROJECT – MANIFEST)

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.example.nprproject">
  <uses-permission android:name="android.permission.INTERNET"/>
  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
    <activity android:name=".ShowHeadlines"/>
  </application>
</manifest>
```



EXAMPLE (NPR PROJECT – LAYOUTS)

App's Main GUI (activity_main.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
                  android:layout_width="match_parent"
                  android:layout_height="match_parent"
                  android:padding="5dp"
                  android:orientation="vertical">
 <ImageView  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:layout_margin="10dp"
                  android:background="@drawable/logo_npr"/>
 <ListView       android:id="@+id/myListView"
                  android:layout_width="match_parent"
                  android:layout_height="wrap_content"/>
</LinearLayout>
```

Custom version of ListView's row (my_simple_list_item_1.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@android:id/text1"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:gravity="center_vertical"
   android:minHeight="40sp"
   android:padding="3dip"
   android:textColor="#ff000000"
   android:background="@drawable/statebackcolor"
   android:textAppearance="@android:style/TextAppearance.DeviceDefault.Small">
</TextView>
```

Text size is smaller than default, the drawable 'statebackcolor' uses different color (light blue) to signal 'state_pressed' (see Appendix A)

EXAMPLE (NPR PROJECT–MAINACTIVITY.JAVA)

```
public class MainActivity extends Activity {
  // Main GUI - A NEWS application based on National Public Radio RSS material
  ArrayAdapter<String> adapterMainSubjects;
  ListView myMainListView;
  Context context;
  SingleItem selectedNewsItem;
  // hard-coding main NEWS categories (TODO: use a resource file)
  String [][] myUrlCaptionMenu = {
        {"https://feeds.npr.org/510289/podcast.xml", "Business"},
        {"https://feeds.npr.org/344098539/podcast.xml", "Comedy"},
        {"https://feeds.npr.org/510308/podcast.xml", "Science"},
        {"https://feeds.npr.org/510298/podcast.xml", "Technology"},
        {"https://feeds.npr.org/510306/podcast.xml", "Music"},
        {"https://feeds.npr.org/510354/podcast.xml", "Kid & family"},
        {"https://feeds.npr.org/510309/podcast.xml", "Society & culture"}
  };
  //define convenient URL and CAPTIONs arrays
  String[] myUrlCaption = new String[myUrlCaptionMenu.length];
  String[] myUrlAddress = new String[myUrlCaptionMenu.length];
  public static String niceDate() {
    SimpleDateFormat sdf = new SimpleDateFormat("EE MMM d, yyyy",
                                                                Locale.US);
    return sdf.format(new Date()); //Monday Apr 7, 2014
  }
```

1

```
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState); setContentView(R.layout.activity_main);
    for (int i=0; i<myUrlAddress.length; i++) {
      myUrlAddress[i] = myUrlCaptionMenu[i][0]; myUrlCaption[i] = myUrlCaptionMenu[i][1];
    }
    context = getApplicationContext();
    this.setTitle("NPR Headline News\n" + niceDate() );
    // user will tap on a ListView's row to request category's headlines
    myMainListView = (ListView)this.findViewById(R.id.myListView);
    myMainListView.setOnItemClickListener(new OnItemClickListener() {
      public void onItemClick(AdapterView<?> _av, View _v, int _index, long _id) {
        String urlAddress = myUrlAddress[_index], urlCaption = myUrlCaption[_index];
        //create an Intent to talk to activity: ShowHeadlines
        Intent callShowHeadlines = new Intent(MainActivity.this, ShowHeadlines.class);
        //prepare a Bundle and add the input arguments: url & caption
        Bundle myData = new Bundle();
        myData.putString("urlAddress", urlAddress); myData.putString("urlCaption", urlCaption);
        callShowHeadlines.putExtras(myData); startActivity(callShowHeadlines);
      }
    });
    // fill up the Main-GUI's ListView with main news categories
    adapterMainSubjects = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, myUrlCaption);
    myMainListView.setAdapter(adapterMainSubjects);
  }//onCreate
} // MainActivity
```

2

3

EXAMPLE (NPR PROJECT–MAINACTIVITY.JAVA)

Comments

1. This is the main thread. It shows a menu (as a ListView) on which the main categories are listed. We have hard-coded the URL and CAPTION for each menu entry, a better practice is to supply a resource file with this set of values. The main NPR categories are subjects such as: ‘Top Stories’, ‘US. News’, ‘World News’, ‘Business’, etc.

2. A listener waiting for the onItemClick event is set on the main GUI’s ListView. When the user selects a row, its index is used to get from the menu array the corresponding URL and CAPTION. Those values are stored in a Bundle and sent to the ShowHeadlines activity; which is started using a non-result returning Intent.

3. The main level ListView is shown to the user. This ListView is displayed using the standard android.R.layout.simple_list_item_1 row layout (medium text size, etc.) Later, in the ShowHeadlines activity we use a custom layout (smaller font, light blue background color on selected state)

EXAMPLE (NPR project–showheadlines.java)

```
public class ShowHeadlines extends Activity {
 // Main category has already been selected by user: 'World News', Business', ...
 // ["urlCaption", "urlAddress"] comes in a bundle sent by main thread
 // here we access RSS-feed and show corresponding headlines
 ArrayList<SingleItem> newsList = new ArrayList<SingleItem>();
 ListView myListView; String urlAddress = "", urlCaption = ""; SingleItem selectedNewsItem;
 @Override
 public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState); setContentView(R.layout.activity_main);
  myListView = (ListView)this.findViewById(R.id.myListView);
  // find out which intent is calling us & grab data bundle holding selected url & caption sent to us
  Intent callingIntent = getIntent();
  Bundle myBundle = callingIntent.getExtras();   ← 1
  urlAddress = myBundle.getString("urlAddress"); urlCaption = myBundle.getString("urlCaption");
  // update app's top 'TitleBar' (eg. 'NPR - Business Wed April 09, 2014')
  this.setTitle("NPR – " + urlCaption + " \t" + MainActivity.niceDate());
  myListView = (ListView)this.findViewById(R.id.myListView);
  myListView.setOnItemClickListener(new OnItemClickListener() {   ← 2
   public void onItemClick(AdapterView<?> av, View v, int index, long id) {
    selectedNewsItem = newsList.get(index);
    showNiceDialogBox(selectedNewsItem, getApplicationContext());
   }});
  // get stories for the selected news option
  DownloadRssFeed downloader = new DownloadRssFeed(ShowHeadlines.this);   ← 3
  downloader.execute(urlAddress, urlCaption);
 }//onCreate
```

```
 public void showNiceDialogBox(SingleItem selectedStoryItem, Context context){
  // make a nice-looking dialog box (story summary, btnClose, btnMore)
  // CAUTION: (check)on occasions title and description are the same!
  String title = selectedStoryItem.getTitle(),
  String description = selectedStoryItem.getDescription();
  if (title.toLowerCase().equals(description.toLowerCase())){ description = ""; }
  try {
   //CAUTION: sometimes TITLE and DESCRIPTION include HTML markers   ← 4
   final Uri storyLink = Uri.parse(selectedStoryItem.getLink());
   AlertDialog.Builder myBuilder = new AlertDialog.Builder(this);
   myBuilder.setIcon(R.drawable.logo_npr)
              .setTitle(Html.fromHtml(urlCaption) )
              .setMessage(title + "\n\n" + Html.fromHtml(description) + "\n")
              .setPositiveButton("Close", null)
              .setNegativeButton("More", new OnClickListener() {
               public void onClick(DialogInterface dialog, int whichOne) {
                Intent browser = new Intent(Intent.ACTION_VIEW, storyLink);
                startActivity(browser);
              }}) //setNegativeButton
    .show();
  }
  catch (Exception e) { Log.e("Error DialogBox", e.getMessage() ); }
 }//showNiceDialogBox
}//ShowHeadlines
```

EXAMPLE (NPR project–showheadlines.java)

Comments

1. The activity begins by extracting the urlAddress and urlCaption data supplied in the incoming Bundle.

2. A listener (bound to the local ListView displaying selected stories) watches for the onItemClick event to show a DialogBox offering an expanded description of the clicked-on item.

3. The incoming arguments are passed to an asynctask responsible for contacting NPR RSS computer and download the selected channel. Before it finishes, the asynctask updates the current activity's ListView with all the stories retrieved from the RSS feed.

4. A 'nice' DialogBox holding: title, description, and two buttons (cancel & more) is displayed when the user requests a summary of a story. Observe the method checks whether or not title and description are the same (not to repeat the same message). Also the HTML.fromHtlm(…) method is used to properly display non-escaped text (commonly used in the <description> items)

EXAMPLE (NPR project–DownloadRssFeed.java)

```
public class DownloadRssFeed extends AsyncTask<String, Void, ArrayList<SingleItem>> {
  // Use supplied URL to download web-feed. This process is inherently
  // slow and MUST be performed inside a thread or asynctask (as in here)
  ShowHeadlines callerContext; //caller class
  String urlAddress, urlCaption;
  ProgressDialog dialog = null;
  public DownloadRssFeed(Context callerContext){
    this.callerContext = (ShowHeadlines) callerContext;
    dialog = new ProgressDialog(callerContext);
  }
  protected void onPreExecute() {
    this.dialog.setMessage("Please wait\nReading RSS feed ...");
    this.dialog.setCancelable(false); //outside touching doesn't dismiss you
    this.dialog.show();
  }
  @Override
  protected void onPostExecute(ArrayList<SingleItem> result) {  ← 4
    super.onPostExecute(result); callerContext.newsList = result;
    // the 'result' list contains headlines for selected news category
    // use custom row layout (small font, blue background on state-pressed)
    int layoutID = R.layout.my_simple_list_item_1;
    ArrayAdapter<SingleItem> adapterNews = new ArrayAdapter<SingleItem>(callerContext, layoutID, result);
    callerContext.myListView.setAdapter(adapterNews);
    dialog.dismiss();
  }
```

EXAMPLE (NPR project–DownloadRssFeed.java)

```
public SingleItem dissectItemNode(NodeList nodeList, int i){
 // disassemble i-th entry in NodeList collection get the first child of elements: extract fields:
 // title, description, pubData, and link. Put those pieces
 // together into a POJO 'SingleItem' object, and return it
 try {
  Element entry = (Element) nodeList.item(i);  ← 5
  Element title = (Element) entry.getElementsByTagName("title").item(0);
  Element description = (Element) entry.getElementsByTagName("description").item(0);
  Element pubDate = (Element) entry.getElementsByTagName("pubDate").item(0);
  Element link = (Element) entry.getElementsByTagName("link").item(0);
  String titleValue = title.getFirstChild().getNodeValue();
  String descriptionValue =description.getFirstChild().getNodeValue();
  String dateValue = pubDate.getFirstChild().getNodeValue();
  String linkValue = link.getFirstChild().getNodeValue();
  SingleItem singleItem = new SingleItem(dateValue, titleValue, descriptionValue, linkValue );
  return singleItem;
 }
 catch (DOMException e) { return new SingleItem("", "Error", e.getMessage(), null); }
}//dissectNode
```

EXAMPLE (NPR project–DownloadRssFeed.java)

```
@Override
 protected ArrayList<SingleItem> doInBackground(String... params) {
  ArrayList<SingleItem> newsList = new ArrayList<SingleItem>();
  urlAddress = params[0]; // eg. "http://www.npr.org/rss/rss.php?id=1004"
  urlCaption = params[1]; // eg. "World News"
  this.dialog.setMessage("Please wait\nReading RSS feed " + urlCaption + "...");
  try { // try to get connected to RSS source
   URL url = new URL(urlAddress);
   URLConnection connection;
   connection = url.openConnection();
   HttpURLConnection httpConnection = (HttpURLConnection) connection;
   int responseCode = httpConnection.getResponseCode();
   if (responseCode == HttpURLConnection.HTTP_OK) {
    InputStream in = httpConnection.getInputStream();
    // define a document builder to work on incoming stream
    DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
    DocumentBuilder db = dbf.newDocumentBuilder();
    // make DOM-tree for incoming XML stream
    Document dom = db.parse(in);
    // make available all access nodes in the parse tree
    Element treeElements = dom.getDocumentElement();
    // look for individual 'stories' (<items> in this case)
    // add each found item to a NodeList collection (newsList)
    newsList.clear();
```

1

2

```
    NodeList itemNodes = treeElements.getElementsByTagName("item");
    if ((itemNodes != null) && (itemNodes.getLength() > 0)) {
     for (int i = 0; i < itemNodes.getLength(); i++) {
      newsList.add( dissectItemNode(itemNodes, i) );
     }// for
    }// if
   }// if
   // time to close. we don't need the connection anymore
   httpConnection.disconnect();
  }
  catch (Exception e) { Log.e("Error>> ", e.getMessage() ); }
  return newsList; //to be consumed by onPostExecute
 }//doInBackground
}//AsyncTask
```

3

EXAMPLE (NPR project–DownloadRssFeed.java)

Comments

1. The activity begins by extracting the urlAddress and urlCaption parameters. Anticipating slow Internet traffic, the method displays a rotating DialogBox telling the user to wait for results to be fetched.

2. The asyntask uses common java.net HTTP methods to set a connection to the NPR RSS site. If successful, the InputStream arriving from the RSS source is converted into a DOM-tree. The method .getDocumentElement() allows direct access to all the tree nodes inside the document.

3. Each item-type node stored in the tree is fetched (remember that each <item> represents a story). The publication-date, title, description, and link are extracted from the item-node and stored in a custom SingleItem object (see bullet 5). SingleItem objects are added to a result list.

4. As soon as the HTTP transfer is over, the asynctask activity closes the connection, dismisses the circular progress bar, and updates the caller's ListView with the headlines held in the result list.

EXAMPLE (NPR PROJECT–SINGLEITEM.JAVA)

```
public class SingleItem {
  private String pubDate;
  private String title;
  private String description;
  private String link;
  public String getPubDate() { return pubDate; }
  public String getTitle() { return title;}
  public String getDescription() { return description; }
  public String getLink() { return link; }
  public SingleItem(String _pubDate, String _title, String _description, String _link) {
    pubDate = _pubDate;
    description = _description;
    title = _title;
    link = _link;
  }
  @Override
  public String toString() { return title; }
}
```

APPENDIX

Instead of using the default layout specs in android.R.layout.simple_list_item_1 you may tell your ArrayAdapter to use a custom row layout.

For instance, the file my_simple_list_item_1.xml contains our own specs for how a ListView's row should look like. In that file we made textSize smaller. We also set its background to a specification provided by /res/drawable/statebackcolor.

We did this so, when the row is selected, we apply a background color of our choosing (light-blue in this example). The state specification is given below.

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
  <!-- pressed -->
  <item android:drawable="@android:color/holo_blue_light" android:state_pressed="true"/>
  <!-- default -->
  <item android:drawable="@android:color/transparent"/>
</selector>
```