Software Engineering Large Practical: XML Parsing

Stephen Gilmore

School of Informatics

October 15th, 2013

An example of downloading and parsing XML

Many services provide information in XML format, allowing apps to download this information and display it in a convenient form for the user. This involves copying the XML file from the server onto the device and then parsing it to extract the content. We follow an example of this.

Parsing XML Data

PREVIOUS | NEXT

Extensible Markup Language (XML) is a set of rules for encoding documents in machine-readable form. XML is a popular format for sharing data on the internet. Websites that frequently update their content, such as news sites or blogs, often provide an XML feed so that external programs can keep abreast of content changes. Uploading and parsing XML data is a common task for network-connected apps. This lesson explains how to parse XML documents and use their data.

Choose a Parser

We recommend `XmlPullParser`, which is an efficient and maintainable way to parse XML on Android. Historically Android has had two implementations of this interface:

- `KXmlParser` via `XmlPullParserFactory.newPullParser()`.
- `ExpatPullParser`, via `Xml.newPullParser()`.

Either choice is fine. The example in this section uses `ExpatPullParser`, via `Xml.newPullParser()`.

Analyze the Feed

TRY IT OUT

Download the sample

NetworkUsage.zip

Parsing XML Data

PREVIOUS | NEXT

THIS LESSON TEACHES YOU TO

1. Choose a Parser
2. Analyze the Feed
3. Instantiate the Parser
4. Read the Feed
5. Parse XML
6. Skip Tags You Don't Care About
7. Consume XML Data

YOU SHOULD ALSO READ

- Web App[illegible]erview

Download the sample

NetworkUsage.zip

Extensible Markup Language (XML) is a set of rules for encoding documents in machine-readable form. XML is a popular format for sharing data on the internet. Websites that frequently update their content, such as news sites or blogs, often provide an XML feed so that external programs can keep abreast of content changes. Uploading and parsing XML data is a common task for network-connected apps. This lesson explains how to parse XML documents and use their data.

Choose a Parser

We recommend `XmlPullParser`, which is an efficient and maintainable way to parse XML on Android. Historically Android has had two implementations of this interface:

- `KXmlParser` via `XmlPullParserFactory.newPullParser()`.
- `ExpatPullParser`, via `Xml.newPullParser()`.

Either choice is fine. The example in this section uses `ExpatPullParser`, via `Xml.newPullParser()`.

Analyze the Feed

The first step in parsing a feed is to decide which fields you're interested in. The parser extracts data
fields and ignores the rest.

Here is an excerpt from the feed that's being parsed in the sample app. Each post to StackOverflow.
in the feed as an `entry` tag that contains several nested tags:

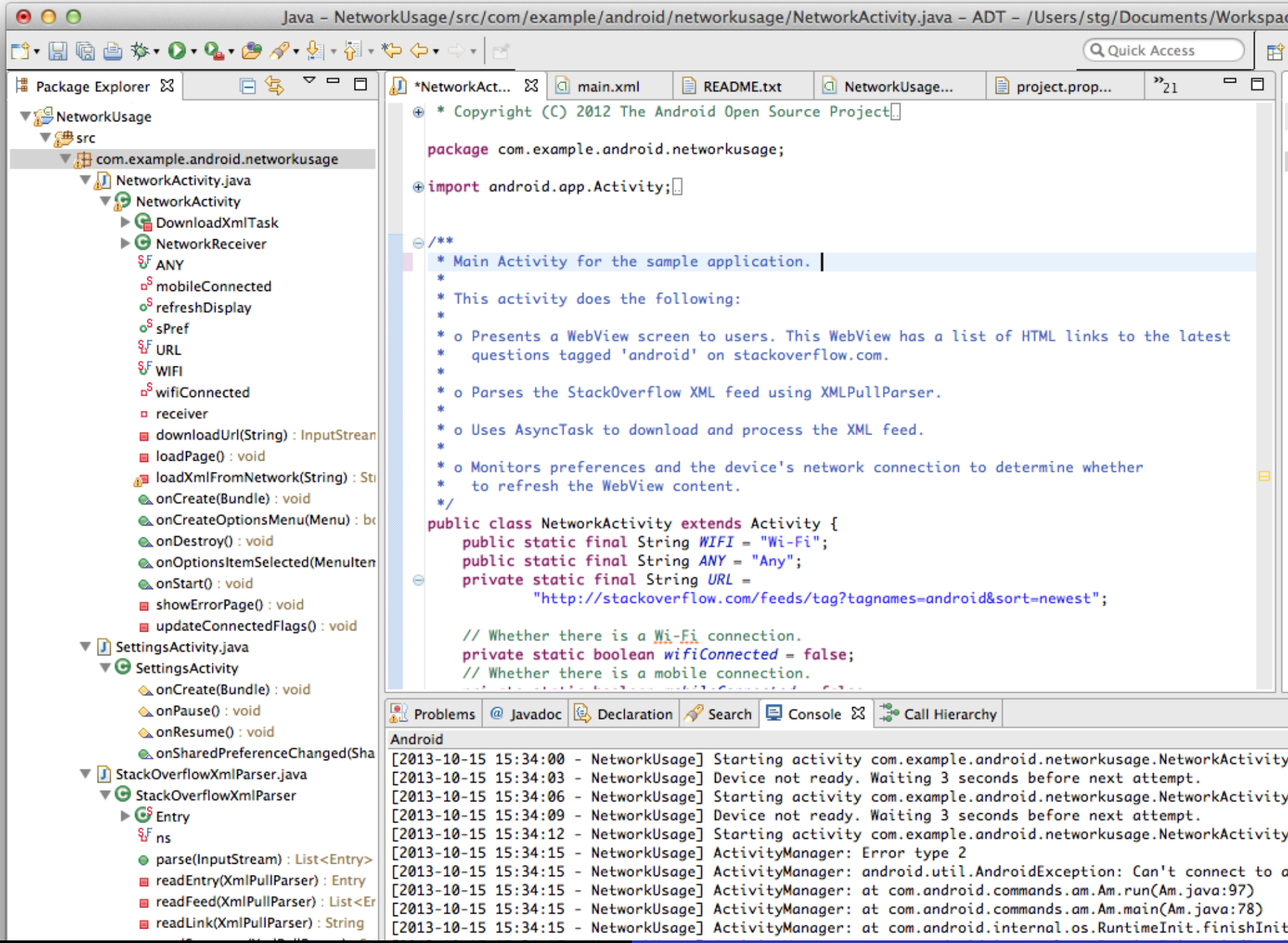
```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:creativeCommons="http://back
<title type="text">newest questions tagged android - Stack Overflow</title>
...
    <entry>
    ...
    </entry>
    <entry>
        <id>http://stackoverflow.com/q/9439999</id>
        <re:rank scheme="http://stackoverflow.com">0</re:rank>
        <title type="text">Where is my data file?</title>
        <category scheme="http://stackoverflow.com/feeds/tag?tagnames=androi
        <category scheme="http://stackoverflow.com/feeds/tag?tagnames=androi
        <author>
            <name>cliff2310</name>
            <uri>http://stackoverflow.com/users/1128925</uri>
        </author>
        <link rel="alternate" href="http://stackoverflow.com/questions/94399
        <published>2012-02-25T00:30:54Z</published>
        <updated>2012-02-25T00:30:54Z</updated>
        <summary type="html">
            <p>I have an Application that requires a data file...</p>

        </summary>
    </entry>
    <entry>
    ...
    </entry>
...
```

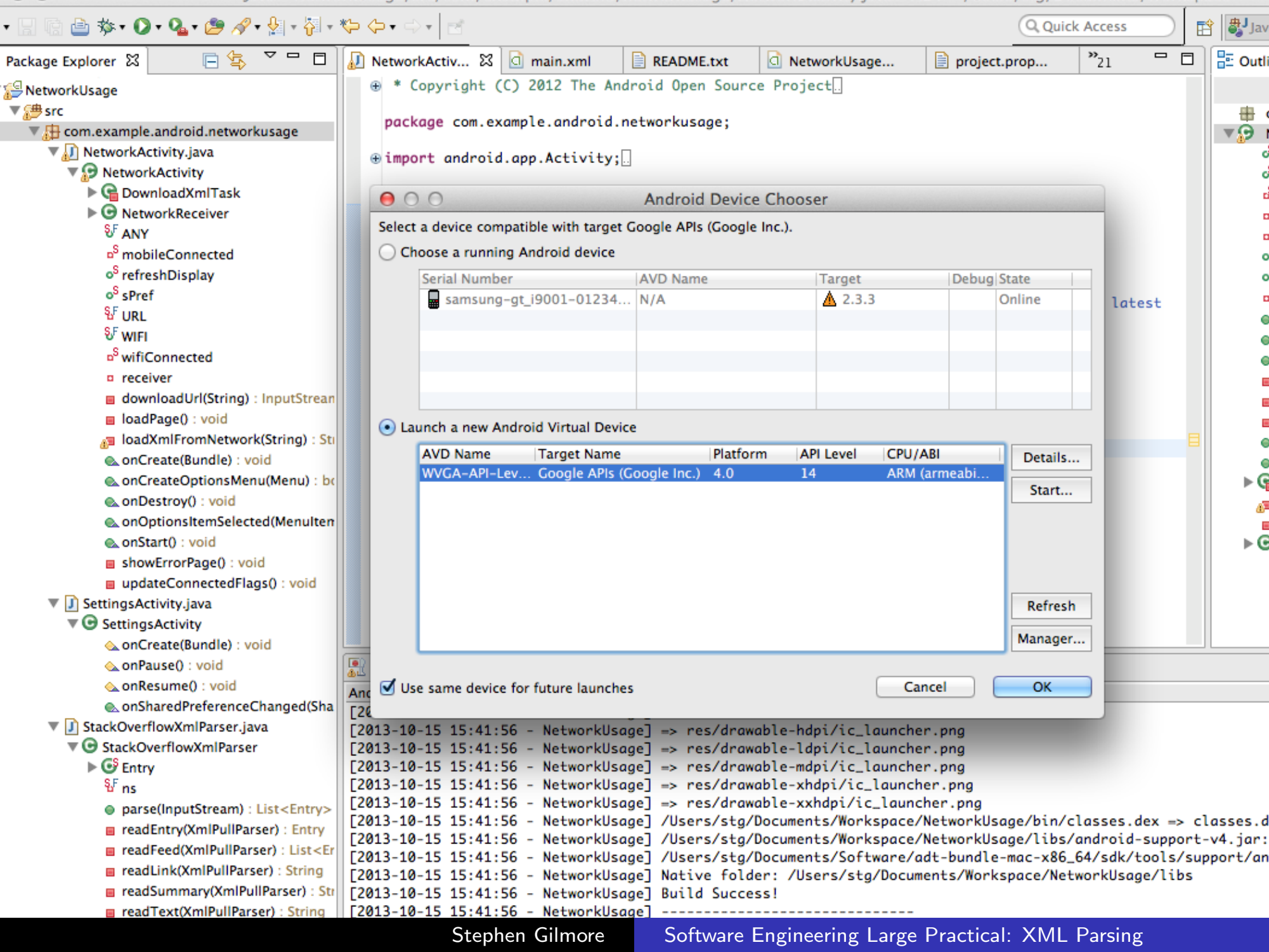
Structure of the NetworkUsage app

The app has three Java classes,

- NetworkActivity,
- SettingsActivity, and
- StackOverflowXmlParser.

We will concentrate on NetworkActivity class and the StackOverflowXmlParser class here.

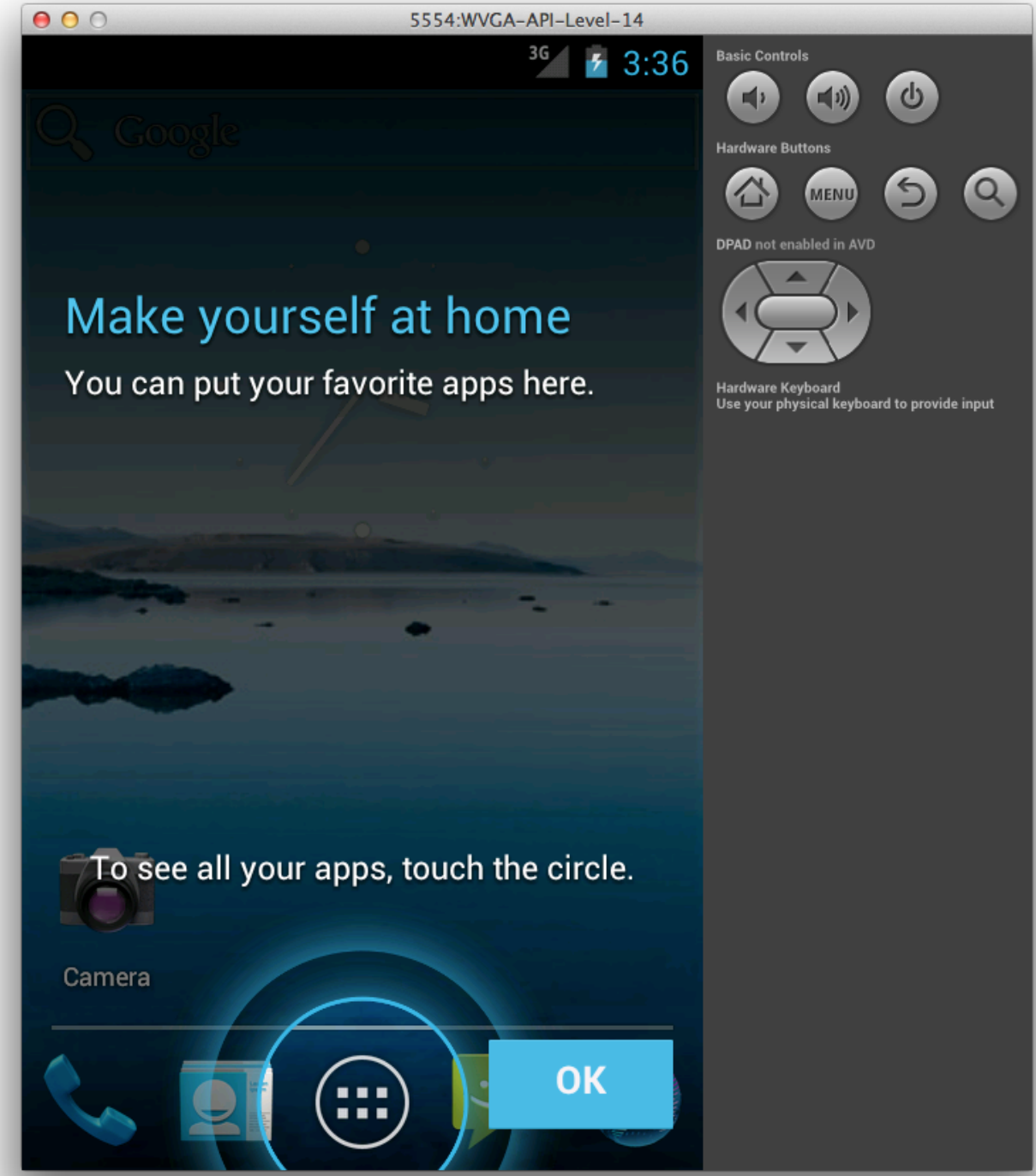

Java – NetworkUsage/src/com/example/android/networkusage/NetworkActivity.java – ADT – /Users/stg/Documents/Workspac
Quick Access
Package Explorer
NetworkUsage
src
com.example.android.networkusage
NetworkActivity.java
NetworkActivity
DownloadXmlTask
NetworkReceiver
ANY
mobileConnected
refreshDisplay
sPref
URL
WIFI
wifiConnected
receiver
downloadUrl(String) : InputStrean
loadPage() : void
loadXmlFromNetwork(String) : Str
onCreate(Bundle) : void
onCreateOptionsMenu(Menu) : bo
onDestroy() : void
onOptionsItemSelected(MenuIten
onStart() : void
showErrorPage() : void
updateConnectedFlags() : void
SettingsActivity.java
SettingsActivity
onCreate(Bundle) : void
onPause() : void
onResume() : void
onSharedPreferenceChanged(Sha
StackOverflowXmlParser.java
StackOverflowXmlParser
Entry
ns
parse(InputStream) : List<Entry>
readEntry(XmlPullParser) : Entry
readFeed(XmlPullParser) : List<Er
readLink(XmlPullParser) : String
*NetworkAct...
main.xml
README.txt
NetworkUsage...
project.prop...
21
* Copyright (C) 2012 The Android Open Source Project
package com.example.android.networkusage;
import android.app.Activity;
/**
* Main Activity for the sample application.
*
* This activity does the following:
*
* o Presents a WebView screen to users. This WebView has a list of HTML links to the latest
* questions tagged 'android' on stackoverflow.com.
*
* o Parses the StackOverflow XML feed using XMLPullParser.
*
* o Uses AsyncTask to download and process the XML feed.
*
* o Monitors preferences and the device's network connection to determine whether
* to refresh the WebView content.
*/
public class NetworkActivity extends Activity {
public static final String WIFI = "Wi-Fi";
public static final String ANY = "Any";
private static final String URL =
"http://stackoverflow.com/feeds/tag?tagnames=android&sort=newest";
// Whether there is a Wi-Fi connection.
private static boolean wifiConnected = false;
// Whether there is a mobile connection.
Problems
Javadoc
Declaration
Search
Console
Call Hierarchy
Android
[2013-10-15 15:34:00 - NetworkUsage] Starting activity com.example.android.networkusage.NetworkActivity
[2013-10-15 15:34:03 - NetworkUsage] Device not ready. Waiting 3 seconds before next attempt.
[2013-10-15 15:34:06 - NetworkUsage] Starting activity com.example.android.networkusage.NetworkActivity
[2013-10-15 15:34:09 - NetworkUsage] Device not ready. Waiting 3 seconds before next attempt.
[2013-10-15 15:34:12 - NetworkUsage] Starting activity com.example.android.networkusage.NetworkActivity
[2013-10-15 15:34:15 - NetworkUsage] ActivityManager: Error type 2
[2013-10-15 15:34:15 - NetworkUsage] ActivityManager: android.util.AndroidException: Can't connect to a
[2013-10-15 15:34:15 - NetworkUsage] ActivityManager: at com.android.commands.am.Am.run(Am.java:97)
[2013-10-15 15:34:15 - NetworkUsage] ActivityManager: at com.android.commands.am.Am.main(Am.java:78)
[2013-10-15 15:34:15 - NetworkUsage] ActivityManager: at com.android.internal.os.RuntimeInit.finishInit

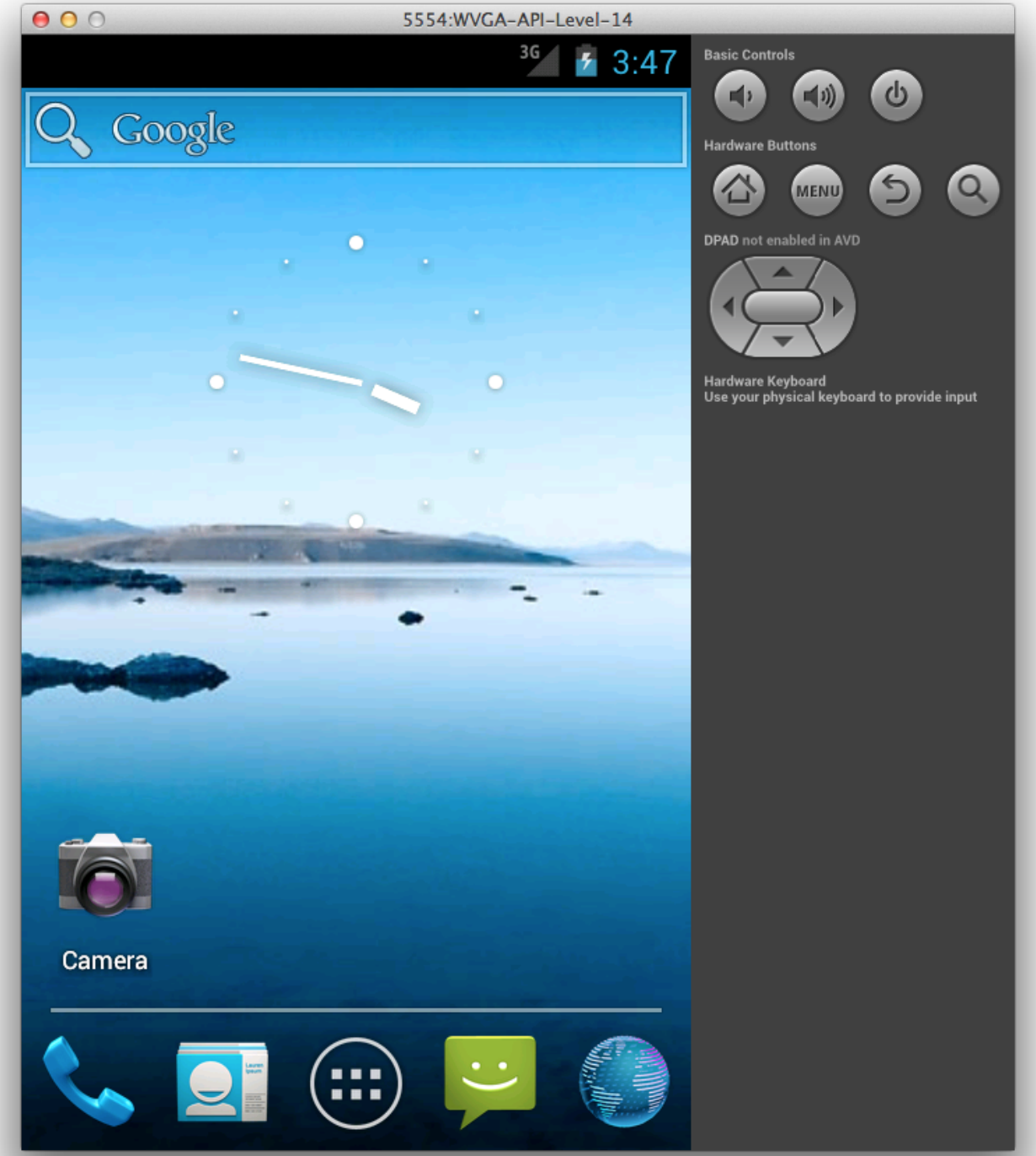

Package Explorer
NetworkUsage
src
com.example.android.networkusage
NetworkActivity.java
NetworkActivity
DownloadXmlTask
NetworkReceiver
ANY
mobileConnected
refreshDisplay
sPref
URL
WIFI
wifiConnected
receiver
downloadUrl(String) : InputStrean
loadPage() : void
loadXmlFromNetwork(String) : Str
onCreate(Bundle) : void
onCreateOptionsMenu(Menu) : bo
onDestroy() : void
onOptionsItemSelected(MenuIten
onStart() : void
showErrorPage() : void
updateConnectedFlags() : void
SettingsActivity.java
SettingsActivity
onCreate(Bundle) : void
onPause() : void
onResume() : void
onSharedPreferenceChanged(Sha
StackOverflowXmlParser.java
StackOverflowXmlParser
Entry
ns
parse(InputStream) : List<Entry>
readEntry(XmlPullParser) : Entry
readFeed(XmlPullParser) : List<Er
readLink(XmlPullParser) : String
readSummary(XmlPullParser) : Str
readText(XmlPullParser) : String
Quick Access
NetworkActiv...
main.xml
README.txt
NetworkUsage...
project.prop...
* Copyright (C) 2012 The Android Open Source Project
package com.example.android.networkusage;
import android.app.Activity;
Android Device Chooser
Select a device compatible with target Google APIs (Google Inc.).
Choose a running Android device
Serial Number
AVD Name
Target
Debug
State
samsung-gt_i9001-01234...
N/A
2.3.3
Online
Launch a new Android Virtual Device
AVD Name
Target Name
Platform
API Level
CPU/ABI
WVGA-API-Lev...
Google APIs (Google Inc.)
4.0
14
ARM (armeabi...
Details...
Start...
Refresh
Manager...
Use same device for future launches
Cancel
OK
latest
[2013-10-15 15:41:56 - NetworkUsage] => res/drawable-hdpi/ic_launcher.png
[2013-10-15 15:41:56 - NetworkUsage] => res/drawable-ldpi/ic_launcher.png
[2013-10-15 15:41:56 - NetworkUsage] => res/drawable-mdpi/ic_launcher.png
[2013-10-15 15:41:56 - NetworkUsage] => res/drawable-xhdpi/ic_launcher.png
[2013-10-15 15:41:56 - NetworkUsage] => res/drawable-xxhdpi/ic_launcher.png
[2013-10-15 15:41:56 - NetworkUsage] /Users/stg/Documents/Workspace/NetworkUsage/bin/classes.dex => classes.d
[2013-10-15 15:41:56 - NetworkUsage] /Users/stg/Documents/Workspace/NetworkUsage/libs/android-support-v4.jar:
[2013-10-15 15:41:56 - NetworkUsage] /Users/stg/Documents/Software/adt-bundle-mac-x86_64/sdk/tools/support/an
[2013-10-15 15:41:56 - NetworkUsage] Native folder: /Users/stg/Documents/Workspace/NetworkUsage/libs
[2013-10-15 15:41:56 - NetworkUsage] Build Success!
[2013-10-15 15:41:56 - NetworkUsage] ----------

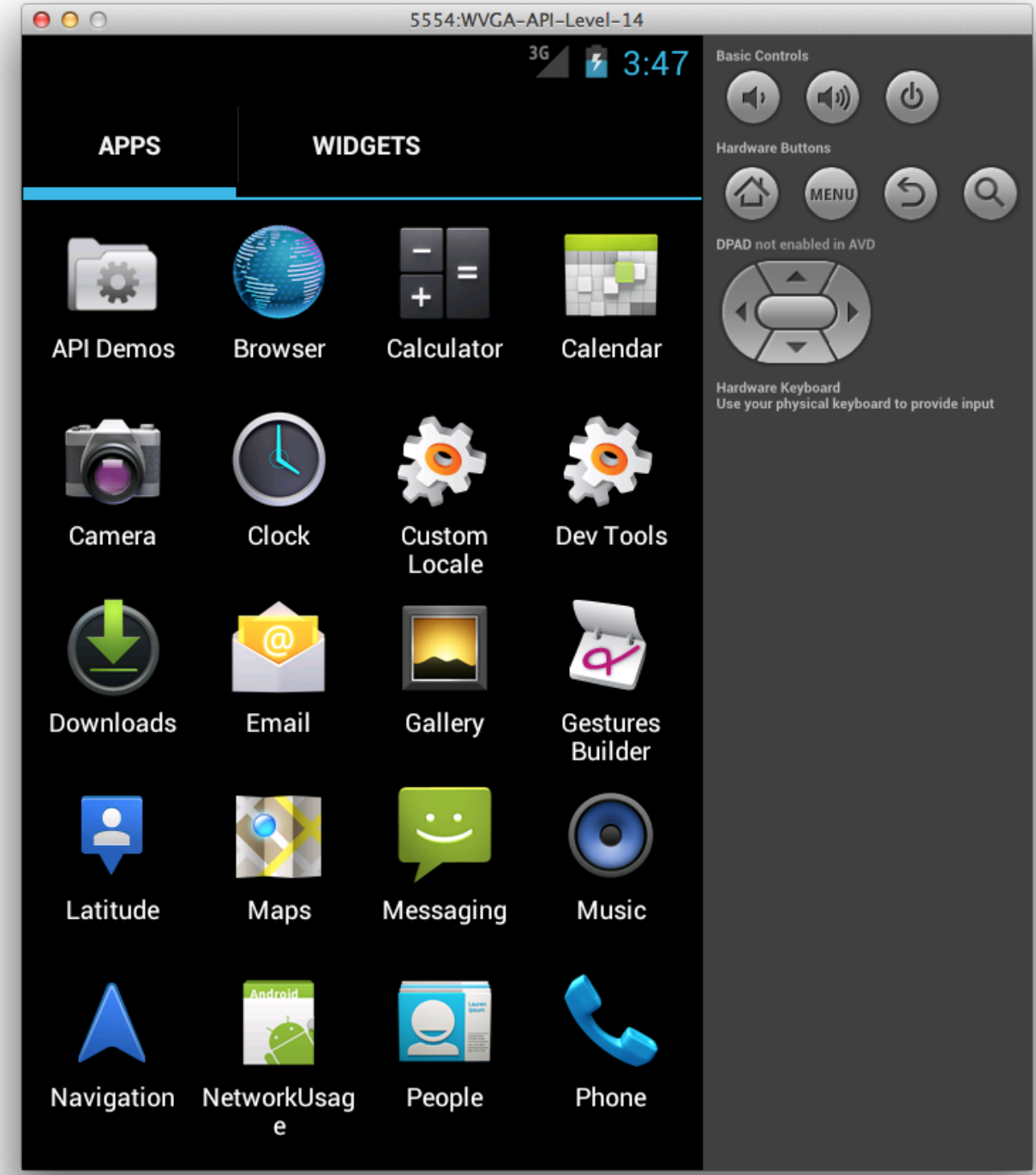
Functionality of the NetworkUsage app

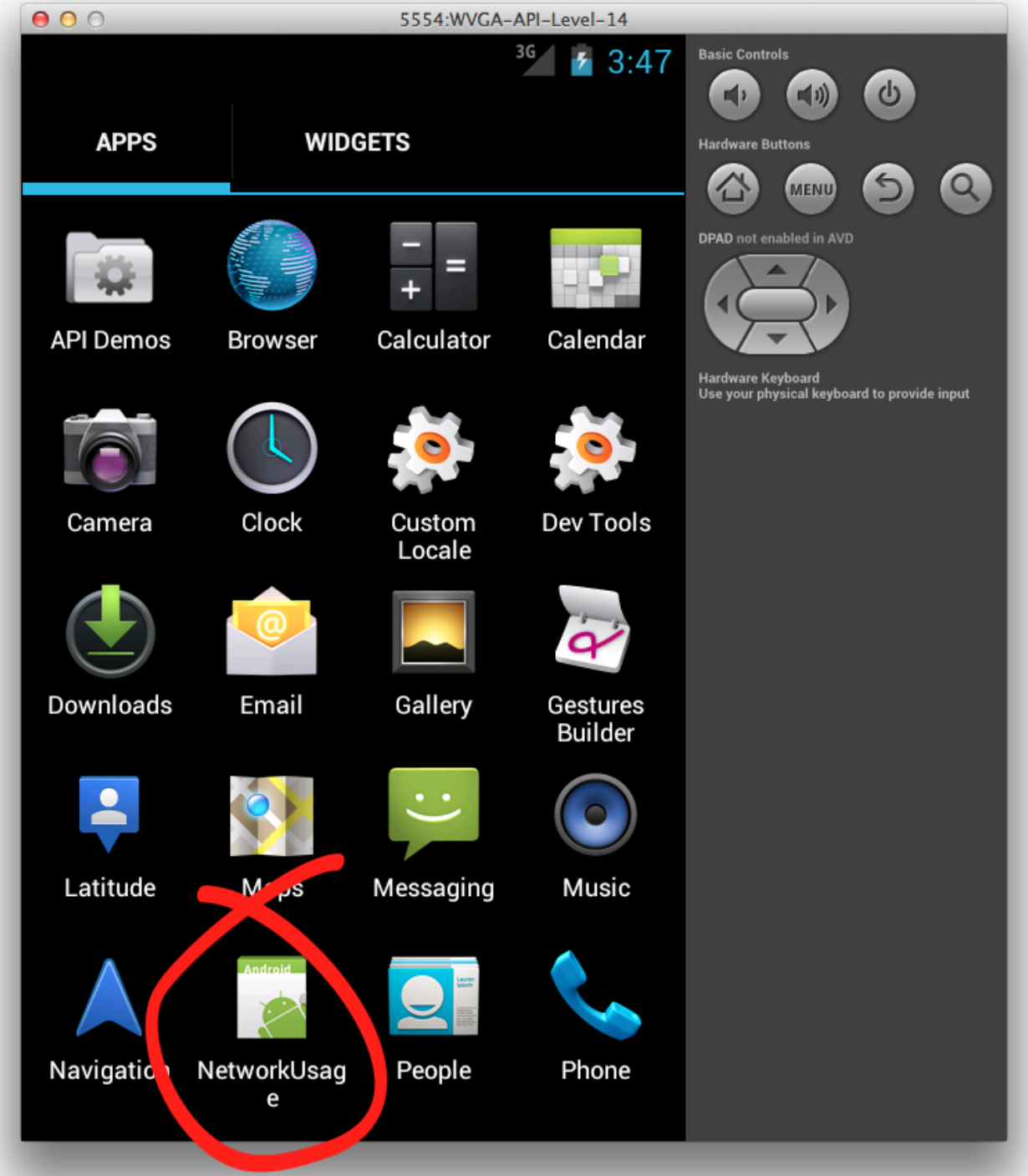
The app performs a relatively simple task: loading pages from the StackOverflow website; finding the newest questions tagged "android" and displaying these to the user.

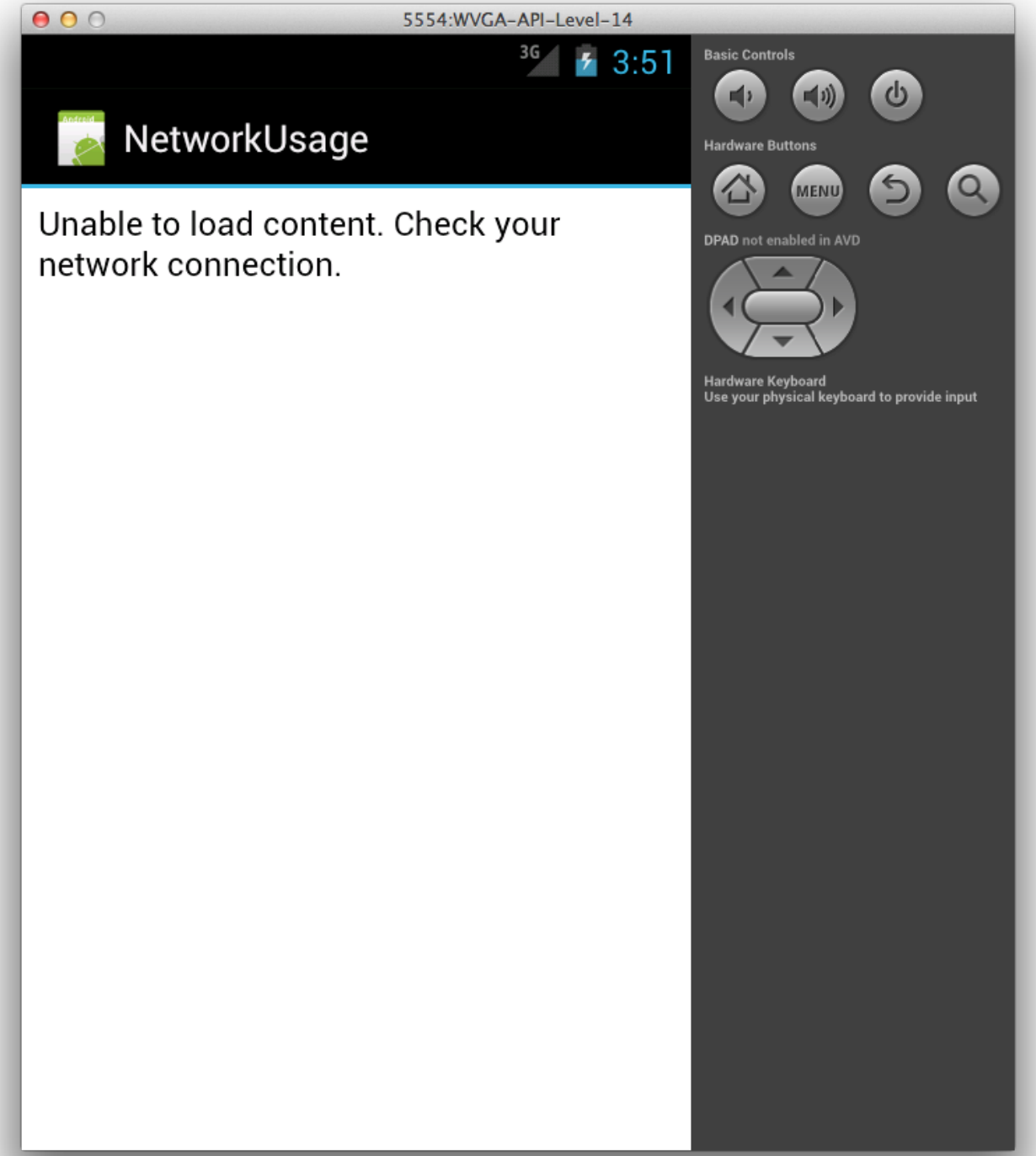
The user can choose to download only over wi-fi, or when connected to any network.

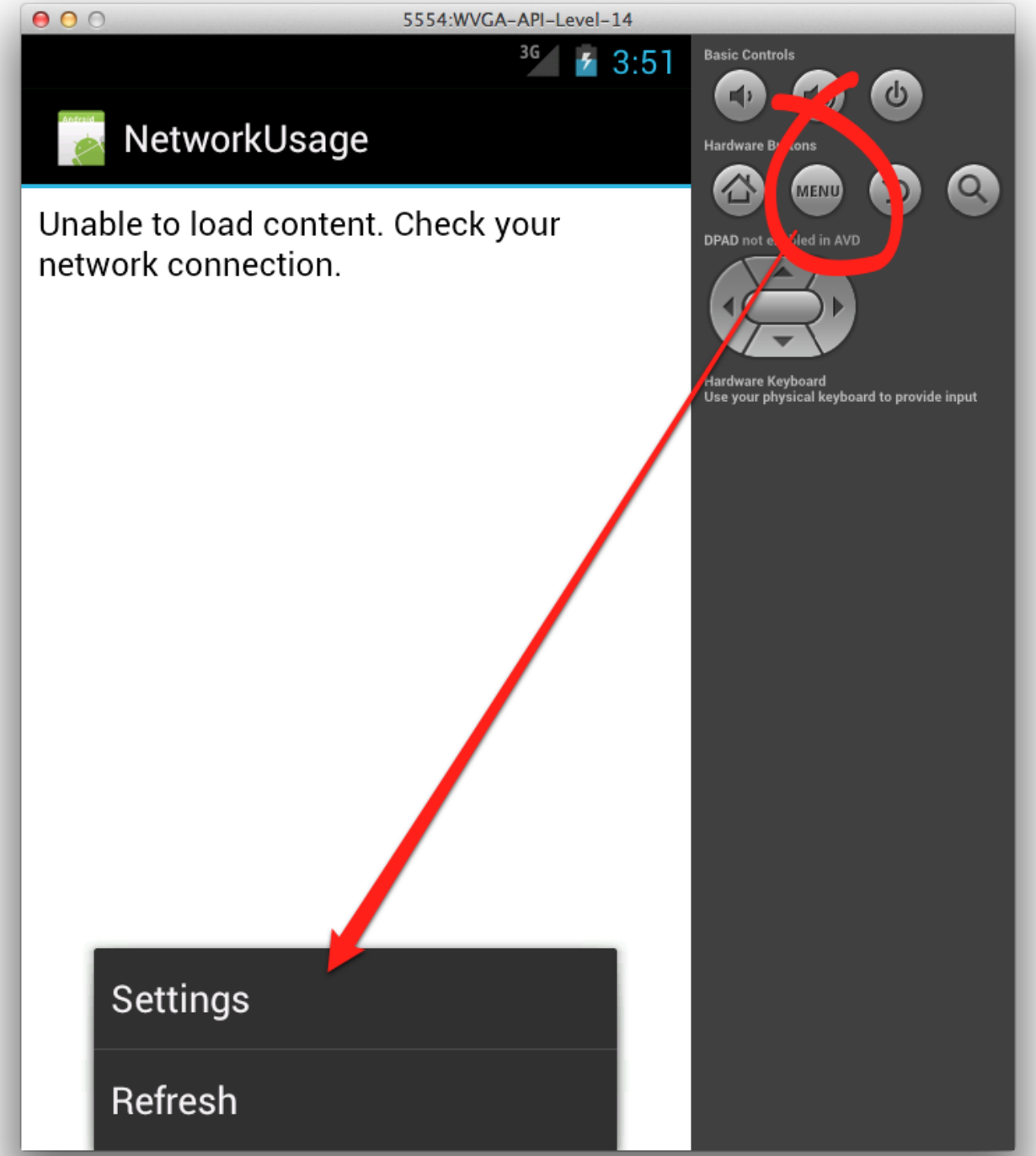

5554:WVGA-API-Level-14
3G
3:36
Google
Make yourself at home
You can put your favorite apps here.
To see all your apps, touch the circle.
Camera
OK
Basic Controls
Hardware Buttons
MENU
DPAD not enabled in AVD
Hardware Keyboard
Use your physical keyboard to provide input

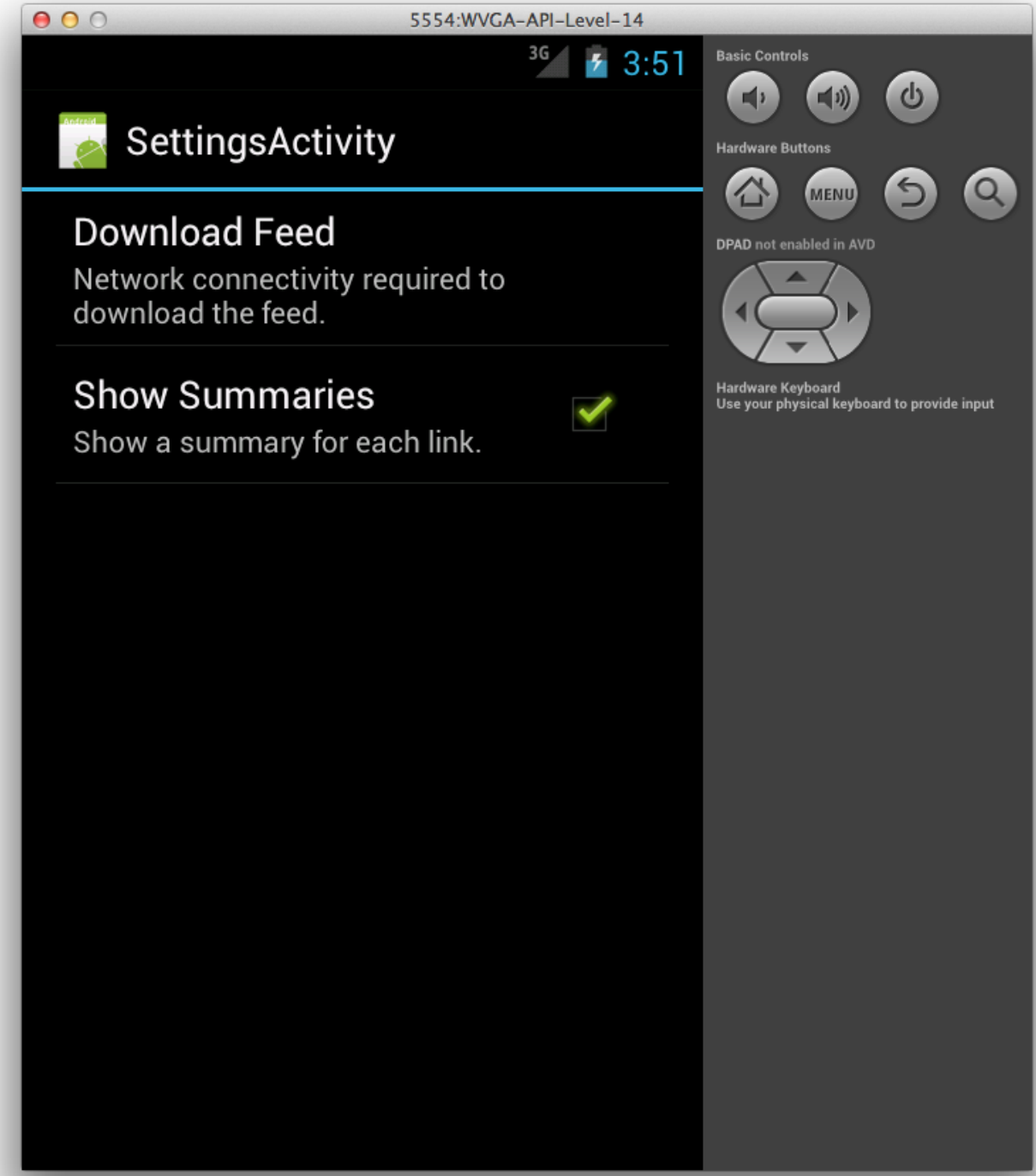

5554:WVGA-API-Level-14
3G
3:47
Google
Camera
Basic Controls
Hardware Buttons
MENU
DPAD not enabled in AVD
Hardware Keyboard
Use your physical keyboard to provide input

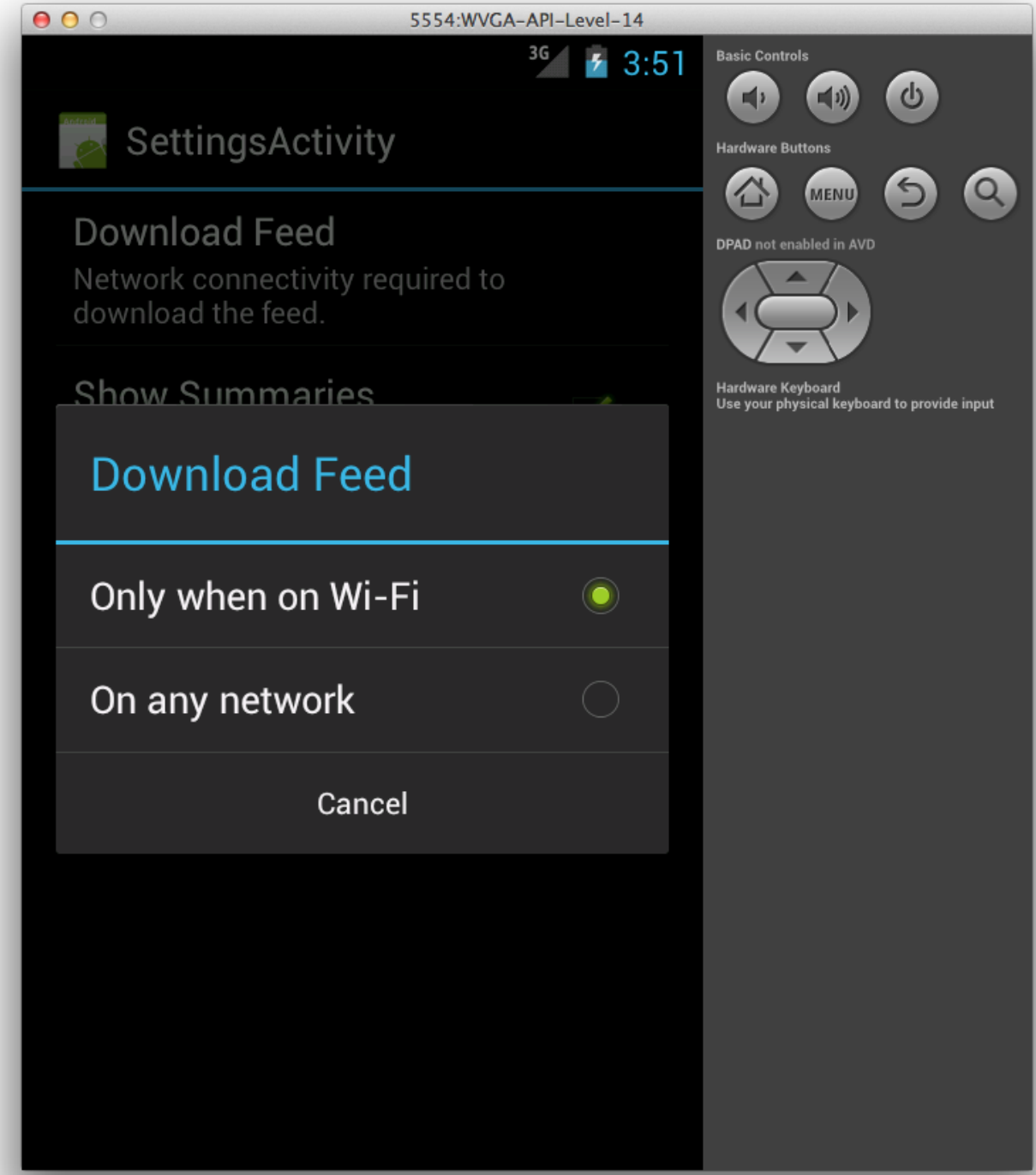

5554:WVGA-API-Level-14
3G
3:47
APPS
WIDGETS
API Demos
Browser
Calculator
Calendar
Camera
Clock
Custom Locale
Dev Tools
Downloads
Email
Gallery
Gestures Builder
Latitude
Maps
Messaging
Music
Navigation
NetworkUsage
People
Phone
Basic Controls
Hardware Buttons
MENU
DPAD not enabled in AVD
Hardware Keyboard
Use your physical keyboard to provide input

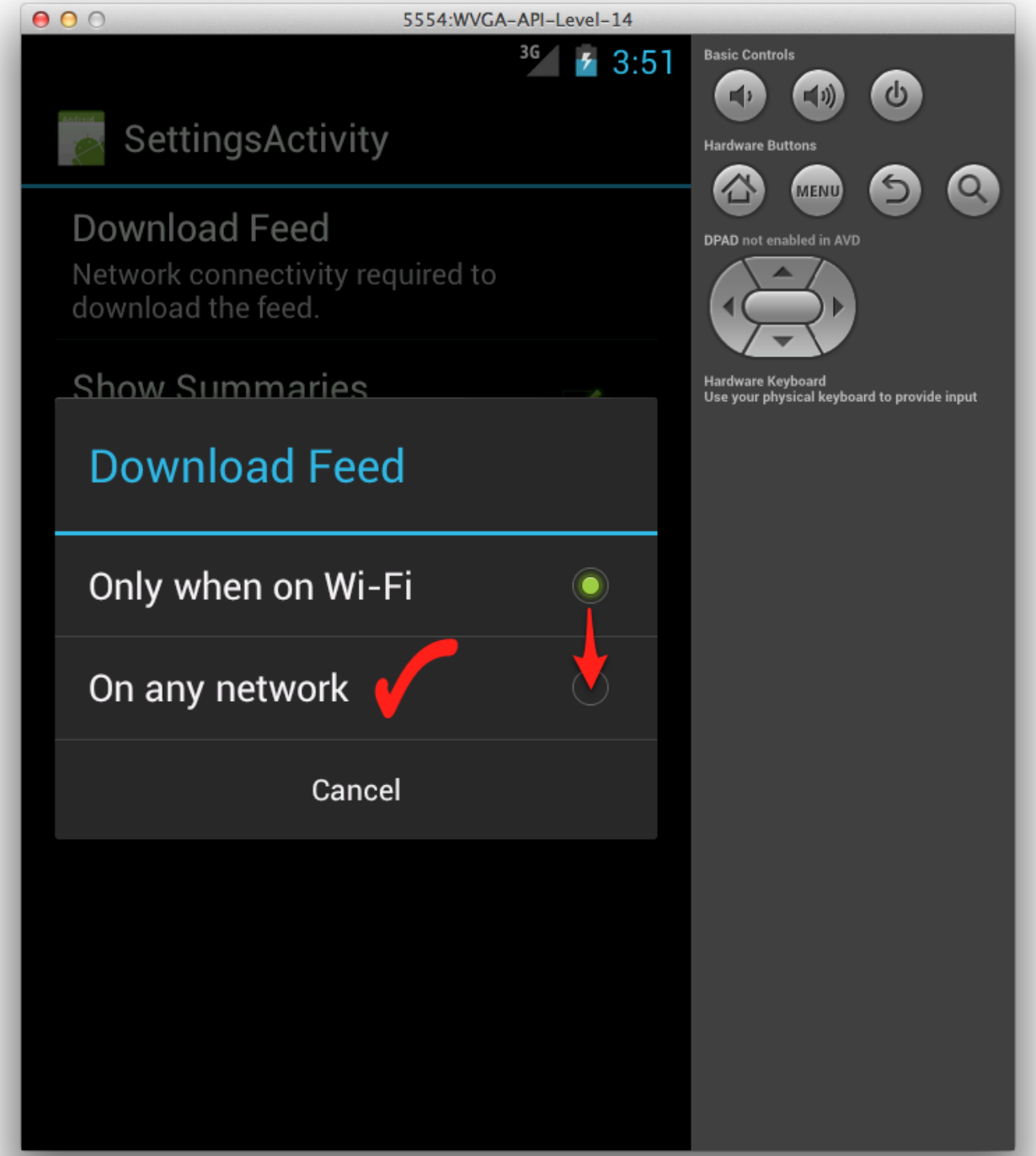

5554:WVGA-API-Level-14
3G
3:47
APPS
WIDGETS
API Demos
Browser
Calculator
Calendar
Camera
Clock
Custom Locale
Dev Tools
Downloads
Email
Gallery
Gestures Builder
Latitude
Messaging
Music
Navigation
NetworkUsage
People
Phone
Basic Controls
Hardware Buttons
MENU
DPAD not enabled in AVD
Hardware Keyboard
Use your physical keyboard to provide input

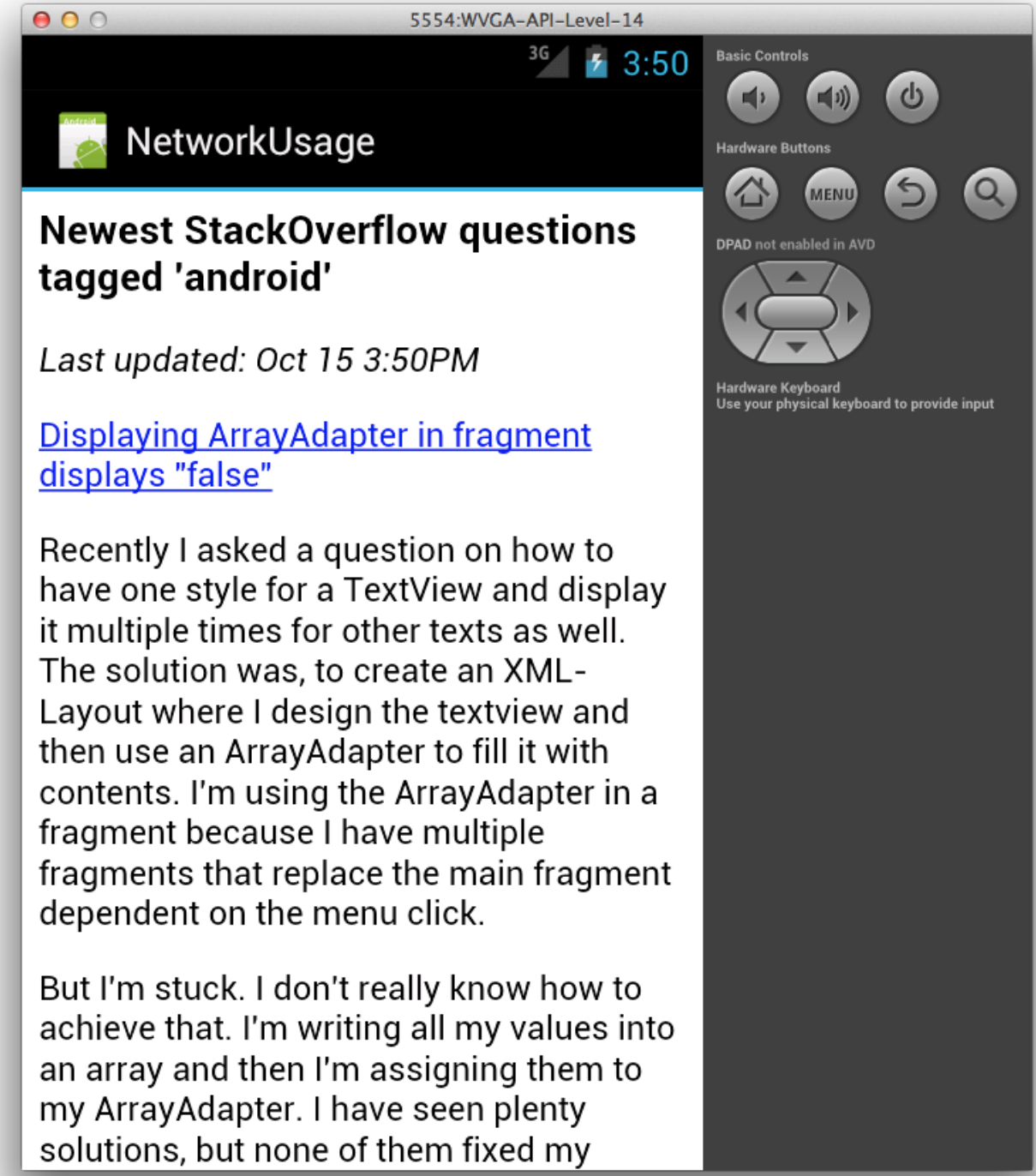

5554:WVGA-API-Level-14
3G
3:51
NetworkUsage
Unable to load content. Check your network connection.
Basic Controls
Hardware Buttons
MENU
DPAD not enabled in AVD
Hardware Keyboard
Use your physical keyboard to provide input


5554:WVGA-API-Level-14
3G
3:51
NetworkUsage
Unable to load content. Check your network connection.
Settings
Refresh
Basic Controls
MENU
Hardware Keyboard
Use your physical keyboard to provide input


5554:WVGA-API-Level-14
3G
3:51
SettingsActivity
Download Feed
Network connectivity required to download the feed.
Show Summaries
Show a summary for each link.
Basic Controls
Hardware Buttons
MENU
DPAD not enabled in AVD
Hardware Keyboard
Use your physical keyboard to provide input


5554:WVGA-API-Level-14
3G
3:51
SettingsActivity
Download Feed
Network connectivity required to download the feed.
Show Summaries
Download Feed
Only when on Wi-Fi
On any network
Cancel
Basic Controls
Hardware Buttons
MENU
DPAD not enabled in AVD
Hardware Keyboard
Use your physical keyboard to provide input


5554:WVGA-API-Level-14
3G
3:51
SettingsActivity
Download Feed
Network connectivity required to download the feed.
Show Summaries
Download Feed
Only when on Wi-Fi
On any network
Cancel
Basic Controls
Hardware Buttons
MENU
DPAD not enabled in AVD
Hardware Keyboard
Use your physical keyboard to provide input


5554:WVGA-API-Level-14
3G
3:50
NetworkUsage
Newest StackOverflow questions tagged 'android'
Last updated: Oct 15 3:50PM
Displaying ArrayAdapter in fragment displays "false"
Recently I asked a question on how to have one style for a TextView and display it multiple times for other texts as well. The solution was, to create an XML-Layout where I design the textview and then use an ArrayAdapter to fill it with contents. I'm using the ArrayAdapter in a fragment because I have multiple fragments that replace the main fragment dependent on the menu click.
But I'm stuck. I don't really know how to achieve that. I'm writing all my values into an array and then I'm assigning them to my ArrayAdapter. I have seen plenty solutions, but none of them fixed my
Basic Controls
Hardware Buttons
MENU
DPAD not enabled in AVD
Hardware Keyboard
Use your physical keyboard to provide input

The NetworkActivity class

The NetworkActivity class is responsible for downloading the XML content over the network. It must handle connection issues. It uses an asynchronous task to download and process the XML feed.

```
 * Copyright (C) 2012 The Android Open Source Project

package com.example.android.networkusage;

import android.app.Activity;


/**
 * Main Activity for the sample application.
 *
 * This activity does the following:
 *
 * o Presents a WebView screen to users. This WebView has a list of HTML links to the latest
 *   questions tagged 'android' on stackoverflow.com.
 *
 * o Parses the StackOverflow XML feed using XMLPullParser.
 *
 * o Uses AsyncTask to download and process the XML feed.
 *
 * o Monitors preferences and the device's network connection to determine whether
 *   to refresh the WebView content.
 */
public class NetworkActivity extends Activity {
    public static final String WIFI = "Wi-Fi";
    public static final String ANY = "Any";
    private static final String URL =
            "http://stackoverflow.com/feeds/tag?tagnames=android&sort=newest";

    // Whether there is a Wi-Fi connection.
    private static boolean wifiConnected = false;
    // Whether there is a mobile connection.
    private static boolean mobileConnected = false;
    // Whether the display should be refreshed.
    public static boolean refreshDisplay = true;

    // The user's current network preference setting.
    public static String sPref = null;

    // The BroadcastReceiver that tracks network connectivity changes.
    private NetworkReceiver receiver = new NetworkReceiver();

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Register BroadcastReceiver to track connection changes.
        IntentFilter filter = new IntentFilter(ConnectivityManager.CONNECTIVITY_ACTION);
        receiver = new NetworkReceiver();
        this.registerReceiver(receiver, filter);
    }
```

```
    // Refreshes the display if the network connection and the
    // pref settings allow it.
    @Override
    public void onStart() {
        super.onStart();

        // Gets the user's network preference settings
        SharedPreferences sharedPrefs = PreferenceManager.getDefaultSharedPreferences(this);

        // Retrieves a string value for the preferences. The second parameter
        // is the default value to use if a preference value is not found.
        sPref = sharedPrefs.getString("listPref", "Wi-Fi");

        updateConnectedFlags();

        // Only loads the page if refreshDisplay is true. Otherwise, keeps previous
        // display. For example, if the user has set "Wi-Fi only" in prefs and the
        // device loses its Wi-Fi connection midway through the user using the app,
        // you don't want to refresh the display--this would force the display of
        // an error page instead of stackoverflow.com content.
        if (refreshDisplay) {
            loadPage();
        }
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        if (receiver != null) {
            this.unregisterReceiver(receiver);
        }
    }

    // Checks the network connection and sets the wifiConnected and mobileConnected
    // variables accordingly.
    private void updateConnectedFlags() {
        ConnectivityManager connMgr =
                (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);

        NetworkInfo activeInfo = connMgr.getActiveNetworkInfo();
        if (activeInfo != null && activeInfo.isConnected()) {
            wifiConnected = activeInfo.getType() == ConnectivityManager.TYPE_WIFI;
            mobileConnected = activeInfo.getType() == ConnectivityManager.TYPE_MOBILE;
        } else {
            wifiConnected = false;
            mobileConnected = false;
        }
    }

    // Uses AsyncTask subclass to download the XML feed from stackoverflow.com
```

```
    // Uses AsyncTask subclass to download the XML feed from stackoverflow.com.
    // This avoids UI lock up. To prevent network operations from
    // causing a delay that results in a poor user experience, always perform
    // network operations on a separate thread from the UI.
    private void loadPage() {
        if (((sPref.equals(ANY)) && (wifiConnected || mobileConnected))
                || ((sPref.equals(WIFI)) && (wifiConnected))) {
            // AsyncTask subclass
            new DownloadXmlTask().execute(URL);
        } else {
            showErrorPage();
        }
    }

    // Displays an error if the app is unable to load content.
    private void showErrorPage() {
        setContentView(R.layout.main);

        // The specified network connection is not available. Displays error message.
        WebView myWebView = (WebView) findViewById(R.id.webview);
        myWebView.loadData(getResources().getString(R.string.connection_error),
                "text/html", null);
    }

    // Populates the activity's options menu.
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.mainmenu, menu);
        return true;
    }

    // Handles the user's menu selection.
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
        case R.id.settings:
                Intent settingsActivity = new Intent(getBaseContext(), SettingsActivity.class);
                startActivity(settingsActivity);
                return true;
        case R.id.refresh:
                loadPage();
                return true;
        default:
                return super.onOptionsItemSelected(item);
        }
    }

    // Implementation of AsyncTask used to download XML feed from stackoverflow.com.
    private class DownloadXmlTask extends AsyncTask<String, Void, String> {
```

```
// Implementation of AsyncTask used to download XML feed from stackoverflow.com.
private class DownloadXmlTask extends AsyncTask<String, Void, String> {

    @Override
    protected String doInBackground(String... urls) {
        try {
            return loadXmlFromNetwork(urls[0]);
        } catch (IOException e) {
            return getResources().getString(R.string.connection_error);
        } catch (XmlPullParserException e) {
            return getResources().getString(R.string.xml_error);
        }
    }

    @Override
    protected void onPostExecute(String result) {
        setContentView(R.layout.main);
        // Displays the HTML string in the UI via a WebView
        WebView myWebView = (WebView) findViewById(R.id.webview);
        myWebView.loadData(result, "text/html", null);
    }
}

// Uploads XML from stackoverflow.com, parses it, and combines it with
// HTML markup. Returns HTML string.
private String loadXmlFromNetwork(String urlString) throws XmlPullParserException, IOException {
    InputStream stream = null;
    StackOverflowXmlParser stackOverflowXmlParser = new StackOverflowXmlParser();
    List<Entry> entries = null;
    String title = null;
    String url = null;
    String summary = null;
    Calendar rightNow = Calendar.getInstance();
    DateFormat formatter = new SimpleDateFormat("MMM dd h:mmaa");

    // Checks whether the user set the preference to include summary text
    SharedPreferences sharedPrefs = PreferenceManager.getDefaultSharedPreferences(this);
    boolean pref = sharedPrefs.getBoolean("summaryPref", false);

    StringBuilder htmlString = new StringBuilder();
    htmlString.append("<h3>" + getResources().getString(R.string.page_title) + "</h3>");
    htmlString.append("<em>" + getResources().getString(R.string.updated) + " " +
            formatter.format(rightNow.getTime()) + "</em>");

    try {
        stream = downloadUrl(urlString);
        entries = stackOverflowXmlParser.parse(stream);
    // Makes sure that the InputStream is closed after the app is
    // finished using it.
    } finally {
```

```
    // Uploads XML from stackoverflow.com, parses it, and combines it with
    // HTML markup. Returns HTML string.
    private String loadXmlFromNetwork(String urlString) throws XmlPullParserException, IOException {
        InputStream stream = null;
        StackOverflowXmlParser stackOverflowXmlParser = new StackOverflowXmlParser();
        List<Entry> entries = null;
        String title = null;
        String url = null;
        String summary = null;
        Calendar rightNow = Calendar.getInstance();
        DateFormat formatter = new SimpleDateFormat("MMM dd h:mmaa");

        // Checks whether the user set the preference to include summary text
        SharedPreferences sharedPrefs = PreferenceManager.getDefaultSharedPreferences(this);
        boolean pref = sharedPrefs.getBoolean("summaryPref", false);

        StringBuilder htmlString = new StringBuilder();
        htmlString.append("<h3>" + getResources().getString(R.string.page_title) + "</h3>");
        htmlString.append("<em>" + getResources().getString(R.string.updated) + " " +
                formatter.format(rightNow.getTime()) + "</em>");

        try {
            stream = downloadUrl(urlString);
            entries = stackOverflowXmlParser.parse(stream);
        // Makes sure that the InputStream is closed after the app is
        // finished using it.
        } finally {
            if (stream != null) {
                stream.close();
            }
        }

        // StackOverflowXmlParser returns a List (called "entries") of Entry objects.
        // Each Entry object represents a single post in the XML feed.
        // This section processes the entries list to combine each entry with HTML markup.
        // Each entry is displayed in the UI as a link that optionally includes
        // a text summary.
        for (Entry entry : entries) {
            htmlString.append("<p><a href='");
            htmlString.append(entry.link);
            htmlString.append("'>" + entry.title + "</a></p>");
            // If the user set the preference to include summary text,
            // adds it to the display.
            if (pref) {
                htmlString.append(entry.summary);
            }
        }
        return htmlString.toString();
    }

    // Given a string representation of a URL, sets up a connection and gets
```

```
        // StackOverflowXmlParser returns a List (called "entries") of Entry objects.
        // Each Entry object represents a single post in the XML feed.
        // This section processes the entries list to combine each entry with HTML markup.
        // Each entry is displayed in the UI as a link that optionally includes
        // a text summary.
        for (Entry entry : entries) {
            htmlString.append("<p><a href='");
            htmlString.append(entry.link);
            htmlString.append("'>" + entry.title + "</a></p>");
            // If the user set the preference to include summary text,
            // adds it to the display.
            if (pref) {
                htmlString.append(entry.summary);
            }
        }
        return htmlString.toString();
    }

    // Given a string representation of a URL, sets up a connection and gets
    // an input stream.
    private InputStream downloadUrl(String urlString) throws IOException {
        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();
        return stream;
    }

    /**
     *
     * This BroadcastReceiver intercepts the android.net.ConnectivityManager.CONNECTIVITY_ACTION,
     * which indicates a connection change. It checks whether the type is TYPE_WIFI.
     * If it is, it checks whether Wi-Fi is connected and sets the wifiConnected flag in the
     * main activity accordingly.
     *
     */
    public class NetworkReceiver extends BroadcastReceiver {

        @Override
        public void onReceive(Context context, Intent intent) {
            ConnectivityManager connMgr =
                    (ConnectivityManager) context.getSystemService(Context.CONNECTIVITY_SERVICE);
            NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();

            // Checks the user prefs and the network connection. Based on the result, decides
            // whether
```

```
        conn.setReadTimeout(10000 /* milliseconds */);
        conn.setConnectTimeout(15000 /* milliseconds */);
        conn.setRequestMethod("GET");
        conn.setDoInput(true);
        // Starts the query
        conn.connect();
        InputStream stream = conn.getInputStream();
        return stream;
    }

    /**
     *
     * This BroadcastReceiver intercepts the android.net.ConnectivityManager.CONNECTIVITY_ACTION,
     * which indicates a connection change. It checks whether the type is TYPE_WIFI.
     * If it is, it checks whether Wi-Fi is connected and sets the wifiConnected flag in the
     * main activity accordingly.
     *
     */
    public class NetworkReceiver extends BroadcastReceiver {

        @Override
        public void onReceive(Context context, Intent intent) {
            ConnectivityManager connMgr =
                    (ConnectivityManager) context.getSystemService(Context.CONNECTIVITY_SERVICE);
            NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();

            // Checks the user prefs and the network connection. Based on the result, decides
            // whether
            // to refresh the display or keep the current display.
            // If the userpref is Wi-Fi only, checks to see if the device has a Wi-Fi connection.
            if (WIFI.equals(sPref) && networkInfo != null
                    && networkInfo.getType() == ConnectivityManager.TYPE_WIFI) {
                // If device has its Wi-Fi connection, sets refreshDisplay
                // to true. This causes the display to be refreshed when the user
                // returns to the app.
                refreshDisplay = true;
                Toast.makeText(context, R.string.wifi_connected, Toast.LENGTH_SHORT).show();

                // If the setting is ANY network and there is a network connection
                // (which by process of elimination would be mobile), sets refreshDisplay to true.
            } else if (ANY.equals(sPref) && networkInfo != null) {
                refreshDisplay = true;

                // Otherwise, the app can't download content--either because there is no network
                // connection (mobile or Wi-Fi), or because the pref setting is WIFI, and there
                // is no Wi-Fi connection.
                // Sets refreshDisplay to false.
            } else {
                refreshDisplay = false;
                Toast.makeText(context, R.string.lost_connection, Toast.LENGTH_SHORT).show();
            }
```

The StackOverflowXmlParser class

The StackOverflowXmlParser class uses an XmlPullParser to process the input stream. It recognises

- a feed;
- each entry within the feed; and
- the title, summary and link within each entry.

```
* Copyright (C) 2012 The Android Open Source Project

package com.example.android.networkusage;

import android.util.Xml;

/**
 * This class parses XML feeds from stackoverflow.com.
 * Given an InputStream representation of a feed, it returns a List of entries,
 * where each list element represents a single entry (post) in the XML feed.
 */
public class StackOverflowXmlParser {
    private static final String ns = null;

    // We don't use namespaces

    public List<Entry> parse(InputStream in) throws XmlPullParserException, IOException {
        try {
            XmlPullParser parser = Xml.newPullParser();
            parser.setFeature(XmlPullParser.FEATURE_PROCESS_NAMESPACES, false);
            parser.setInput(in, null);
            parser.nextTag();
            return readFeed(parser);
        } finally {
            in.close();
        }
    }

    private List<Entry> readFeed(XmlPullParser parser) throws XmlPullParserException, IOException {
        List<Entry> entries = new ArrayList<Entry>();

        parser.require(XmlPullParser.START_TAG, ns, "feed");
        while (parser.next() != XmlPullParser.END_TAG) {
            if (parser.getEventType() != XmlPullParser.START_TAG) {
                continue;
            }
            String name = parser.getName();
            // Starts by looking for the entry tag
            if (name.equals("entry")) {
                entries.add(readEntry(parser));
            } else {
                skip(parser);
            }
        }
        return entries;
    }

    // This class represents a single entry (post) in the XML feed.
    // It includes the data members "title," "link," and "summary."
    public static class Entry {
        public final String title;
```

```
// This class represents a single entry (post) in the XML feed.
// It includes the data members "title," "link," and "summary."
public static class Entry {
    public final String title;
    public final String link;
    public final String summary;

    private Entry(String title, String summary, String link) {
        this.title = title;
        this.summary = summary;
        this.link = link;
    }
}

// Parses the contents of an entry. If it encounters a title, summary, or link tag, hands them
// off
// to their respective &amp;quot;read&amp;quot; methods for processing. Otherwise, skips the tag.
private Entry readEntry(XmlPullParser parser) throws XmlPullParserException, IOException {
    parser.require(XmlPullParser.START_TAG, ns, "entry");
    String title = null;
    String summary = null;
    String link = null;
    while (parser.next() != XmlPullParser.END_TAG) {
        if (parser.getEventType() != XmlPullParser.START_TAG) {
            continue;
        }
        String name = parser.getName();
        if (name.equals("title")) {
            title = readTitle(parser);
        } else if (name.equals("summary")) {
            summary = readSummary(parser);
        } else if (name.equals("link")) {
            link = readLink(parser);
        } else {
            skip(parser);
        }
    }
    return new Entry(title, summary, link);
}

// Processes title tags in the feed.
private String readTitle(XmlPullParser parser) throws IOException, XmlPullParserException {
    parser.require(XmlPullParser.START_TAG, ns, "title");
    String title = readText(parser);
    parser.require(XmlPullParser.END_TAG, ns, "title");
    return title;
}

// Processes link tags in the feed.
private String readLink(XmlPullParser parser) throws IOException, XmlPullParserException {
```

```
// Processes title tags in the feed.
private String readTitle(XmlPullParser parser) throws IOException, XmlPullParserException {
    parser.require(XmlPullParser.START_TAG, ns, "title");
    String title = readText(parser);
    parser.require(XmlPullParser.END_TAG, ns, "title");
    return title;
}

// Processes link tags in the feed.
private String readLink(XmlPullParser parser) throws IOException, XmlPullParserException {
    String link = "";
    parser.require(XmlPullParser.START_TAG, ns, "link");
    String tag = parser.getName();
    String relType = parser.getAttributeValue(null, "rel");
    if (tag.equals("link")) {
        if (relType.equals("alternate")) {
            link = parser.getAttributeValue(null, "href");
            parser.nextTag();
        }
    }
    parser.require(XmlPullParser.END_TAG, ns, "link");
    return link;
}

// Processes summary tags in the feed.
private String readSummary(XmlPullParser parser) throws IOException, XmlPullParserException {
    parser.require(XmlPullParser.START_TAG, ns, "summary");
    String summary = readText(parser);
    parser.require(XmlPullParser.END_TAG, ns, "summary");
    return summary;
}

// For the tags title and summary, extracts their text values.
private String readText(XmlPullParser parser) throws IOException, XmlPullParserException {
    String result = "";
    if (parser.next() == XmlPullParser.TEXT) {
        result = parser.getText();
        parser.nextTag();
    }
    return result;
}

// Skips tags the parser isn't interested in. Uses depth to handle nested tags. i.e.,
// if the next tag after a START_TAG isn't a matching END_TAG, it keeps going until it
// finds the matching END_TAG (as indicated by the value of "depth" being 0).
private void skip(XmlPullParser parser) throws XmlPullParserException, IOException {
    if (parser.getEventType() != XmlPullParser.START_TAG) {
        throw new IllegalStateException();
    }
    int depth = 1;
```

```
    // Processes link tags in the feed.
    private String readLink(XmlPullParser parser) throws IOException, XmlPullParserException {
        String link = "";
        parser.require(XmlPullParser.START_TAG, ns, "link");
        String tag = parser.getName();
        String relType = parser.getAttributeValue(null, "rel");
        if (tag.equals("link")) {
            if (relType.equals("alternate")) {
                link = parser.getAttributeValue(null, "href");
                parser.nextTag();
            }
        }
        parser.require(XmlPullParser.END_TAG, ns, "link");
        return link;
    }

    // Processes summary tags in the feed.
    private String readSummary(XmlPullParser parser) throws IOException, XmlPullParserException {
        parser.require(XmlPullParser.START_TAG, ns, "summary");
        String summary = readText(parser);
        parser.require(XmlPullParser.END_TAG, ns, "summary");
        return summary;
    }

    // For the tags title and summary, extracts their text values.
    private String readText(XmlPullParser parser) throws IOException, XmlPullParserException {
        String result = "";
        if (parser.next() == XmlPullParser.TEXT) {
            result = parser.getText();
            parser.nextTag();
        }
        return result;
    }

    // Skips tags the parser isn't interested in. Uses depth to handle nested tags. i.e.,
    // if the next tag after a START_TAG isn't a matching END_TAG, it keeps going until it
    // finds the matching END_TAG (as indicated by the value of "depth" being 0).
    private void skip(XmlPullParser parser) throws XmlPullParserException, IOException {
        if (parser.getEventType() != XmlPullParser.START_TAG) {
            throw new IllegalStateException();
        }
        int depth = 1;
        while (depth != 0) {
            switch (parser.next()) {
            case XmlPullParser.END_TAG:
                    depth--;
                    break;
            case XmlPullParser.START_TAG:
                    depth++;
                    break;
            }
```