How to Work with Threads, Files, adapters and intents

L9 – Threads, Files, Adapters & Intents

Ref: Chapter 10 Murach's Android Programming 2Ed

Learning Outcomes

Applied

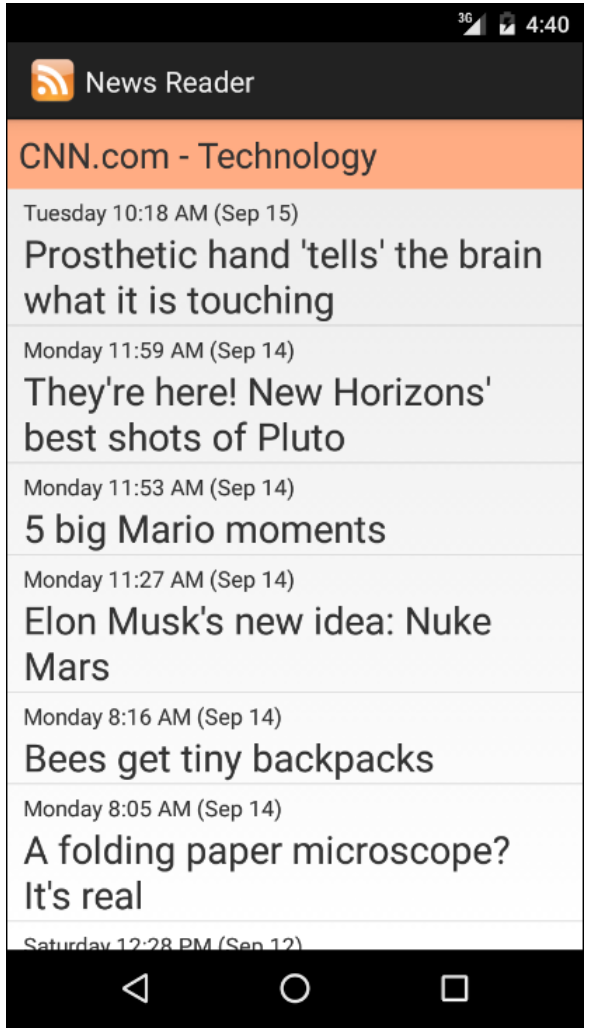
- Given a task that takes more than a few seconds to execute, create a new thread to begin executing that task right away.
- Given a task that takes more than a few seconds to execute, create a new thread that executes that that task after a specified delay or at a specified interval.
- After a thread finishes executing, update the UI thread.
- Given the URL for a file that's available from the Internet, write the code that downloads the file from the Internet and saves it to the file system of the Android device.
- Given data that's stored in an array, use an adapter to display the data in a ListView widget and to handle the events that occur on this data.
- Use an intent to pass data from one activity to another.
- Use an intent to start a web browser or to dial or call a phone number.

Learning Outcomes

Knowledge

- Explain why you should use a separate thread for any task that might slow or stop the responsiveness of the UI thread for an Android app.
- In general terms, describe how an asynchronous task works.
- Describe the difference between an explicit intent and an implicit intent.

The Items activity



The Item activity

The URL for the RSS feed

```
http://rss.cnn.com/rss/cnn_tech.rss
```

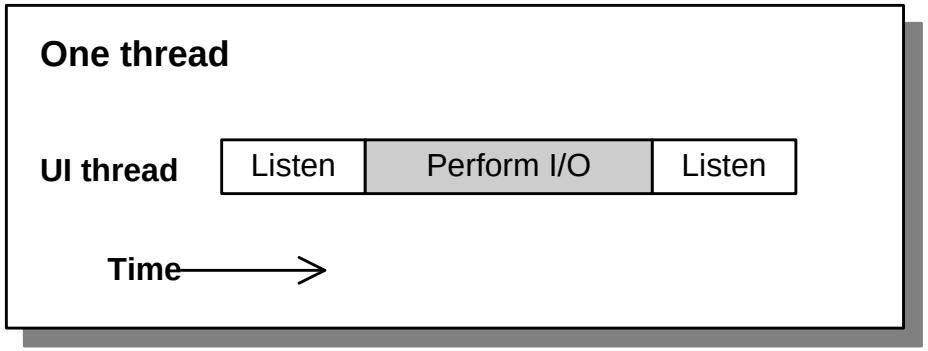
Simplified XML for the RSS feed

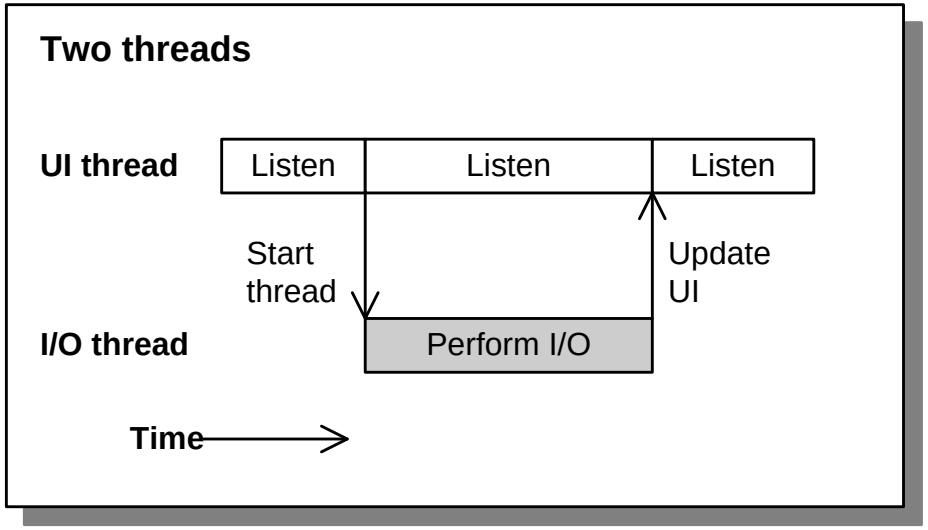
```
<rss xmlns:media="http://search.yahoo.com/mrss/"
     xmlns:feedburner=
         "http://rssnamespace.org/feedburner/ext/1.0"
     version="2.0">
<channel>
  <title>CNN.com - Technology</title>
  <pubDate>Tue, 15 Sep 2015 17:18:54 GMT</pubDate>
  <item>
    <title>Prosthetic hand 'tells' the brain what it is
      touching</title>
    <link>http://rss.cnn.com/c/35492/f/676960/s/story01.htm
       </link>
    <description>Research on prosthetic hands has come a long
       way, but most of it has focused on improving the way
       the body controls the device.</description>
    <pubDate>Tue, 15 Sep 2015 17:18:08 GMT</pubDate>
  </item>
```

Simplified XML for the RSS feed (continued)

```
  <item>
    <title>They're here! New Horizons' best shots
      of Pluto</title>
    <link>http://rss.cnn.com/c/35492/f/676960/story01.htm</link>
    <description>&lt;br clear='all'/&gt;</description>
    <pubDate>Mon, 14 Sep 2015 18:59:52 GMT</pubDate>
  </item>
  <item>
    <title>5 big Mario moments</title>
    <link>http://rss.cnn.com/c/35492/story01.htm</link>
    <description>&lt;br clear='all'/&gt;</description>
    <pubDate>Mon, 14 Sep 2015 18:53:03 GMT</pubDate>
  </item>
  ...
  ...
</channel>
</rss>
```

How using threads improves user interface responsiveness





An activity with a nested AsyncTask class

```
package com.murach.newsreader;

import android.os.AsyncTask;
import android.os.Bundle;
import android.app.Activity;
import android.content.Context;
import android.widget.Toast;

public class ItemsActivity extends Activity {

    private static String URL_STRING =
            "http://rss.cnn.com/rss/cnn_tech.rss";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_items);

        new DownloadFeed().execute(URL_STRING);
    }
```

An activity with a nested AsyncTask class (cont.)

```
    class DownloadFeed extends AsyncTask<String, Void, String> {
        @Override
        protected String doInBackground(String... params) {
            // get the parameter
            String urlString = params[0];

            // download the feed and write it to a file

            // return a message
            return "Feed downloaded";
        }

        @Override
        protected void onPostExecute(String result) {
            Context context = ItemsActivity.this;
            Toast.makeText(context,
                           result, Toast.LENGTH_LONG).show();
        }
    }
}
```

The generic types for the AsyncTask class

```
AsyncTask<Params, Progress, Result>
```

Possible class declarations

```
class DownloadFeed extends AsyncTask<String, Integer, String> {}
class DownloadFeed extends AsyncTask<URL, Void, String> {}
class DownloadFeed extends AsyncTask<Void, Void, Void> {}
class ReadFeed extends AsyncTask<String, Integer, RSSFeed> {}
```

The AsyncTask class

Method	Is executed…
`onPreExecute()`	On the UI thread.
`doInBackground(Params...)`	On the background thread.
`onProgressUpdate(Progress...)`	On the UI thread.
`onPostExecute(Result)`	On the UI thread.

The classes used to work with timed tasks

```
java.util.Timer
java.util.TimerTask
```

A method that starts a timed task

```
private void startTimer() {
    final long startMillis = System.currentTimeMillis();
    TimerTask task = new TimerTask() {

        @Override
        public void run() {
            long elapsedMillis =
                System.currentTimeMillis() - startMillis;
            updateView(elapsedMillis);
        }
    };
    Timer timer = new Timer(true);
    timer.schedule(task, 0, 1000);  // execute every second
}
```

A constructor and method of the TimerTask class

```
TimerTask()
run()
```

A constructor and some methods of the Timer class

```
Timer(isDaemon)
schedule(task, delay)
schedule(task, delay, interval)
cancel()
```

A method that updates the UI thread

```
private void updateView(final long elapsedMillis) {
    // UI changes need to be run on the UI thread
    messageTextView.post(new Runnable() {

        int elapsedSeconds = (int) elapsedMillis/1000;

        @Override
        public void run() {
            messageTextView.setText(
                "Seconds: " + elapsedSeconds);
        }
    });
}
```

A method of the View class

```
post(runnable)
```

A method of the Runnable interface

```
run()
```

The classes for downloading from the Internet

`java.net.URL`

`android.content.Context`

How to download a file from the Internet

```
final String FILENAME = "news_feed.xml";
try{
    // get the input stream
    URL url = new URL("http://rss.cnn.com/rss/cnn_tech.rss");
    InputStream in = url.openStream();

    // get the output stream
    FileOutputStream out =
            openFileOutput(FILENAME, Context.MODE_PRIVATE);

    // read input and write output
    byte[] buffer = new byte[1024];
    int bytesRead = in.read(buffer);
    while (bytesRead != -1)
    {
        out.write(buffer, 0, bytesRead);
        bytesRead = in.read(buffer);
    }
    out.close();
    in.close();
}
catch (IOException e) {
    Log.e("News reader", e.toString());
}
```

A method of the URL class

```
openStream()
```

The INTERNET permission in the AndroidManifest.xml file

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

A method of the Context class

```
openFileOutput(filename, mode)
```

The classes used to work with SAX

```
javax.xml.parsers.SAXParser
javax.xml.parsers.SAXParserFactory

org.xml.sax.InputSource
org.xml.sax.XMLReader
```

How to parse an XML file

```
final String FILENAME = "news_feed.xml";
RSSFeed feed;
try {
    // get the XML reader
    SAXParserFactory factory = SAXParserFactory.newInstance();
    SAXParser parser = factory.newSAXParser();
    XMLReader xmlreader = parser.getXMLReader();
    // set content handler
    RSSFeedHandler theRssHandler = new RSSFeedHandler();
    xmlreader.setContentHandler(theRssHandler);

    // get the input stream
    FileInputStream in = openFileInput(FILENAME);

    // parse the data
    InputSource is = new InputSource(in);
    xmlreader.parse(is);

    // get the content handler and return it
    feed = theRssHandler.getFeed();
}
catch (Exception e) {
    Log.e("News reader", e.toString());
}
```

A method of the Context class

```
openFileInput(filename)
```

The RSSFeedHandler class

```
package com.murach.newsreader;

import org.xml.sax.helpers.DefaultHandler;
import org.xml.sax.*;

public class RSSFeedHandler extends DefaultHandler {
    private RSSFeed feed;
    private RSSItem item;

    private boolean feedTitleHasBeenRead = false;
    private boolean feedPubDateHasBeenRead = false;

    private boolean isTitle = false;
    private boolean isDescription = false;
    private boolean isLink = false;
    private boolean isPubDate = false;

    public RSSFeed getFeed() {
        return feed;
    }
```

The RSSFeedHandler class (continued)

```
public void startDocument() throws SAXException {
    feed = new RSSFeed();
    item = new RSSItem();
}

public void endDocument() throws SAXException { }

public void startElement(String namespaceURI, String localName,
        String qName, Attributes atts) throws SAXException {

    if (qName.equals("item")) {
        item = new RSSItem();
        return;
    }
    else if (qName.equals("title")) {
        isTitle = true;
        return;
    }
    else if (qName.equals("description")) {
        isDescription = true;
        return;
    }
```

The RSSFeedHandler class (continued)

```
        else if (qName.equals("link")) {
            isLink = true;
            return;
        }
        else if (qName.equals("pubDate")) {
            isPubDate = true;
            return;
        }
    }

    public void endElement(String namespaceURI, String localName,
            String qName) throws SAXException {
        if (qName.equals("item")) {
            feed.addItem(item);
            return;
        }
    }
```

The RSSFeedHandler class (continued)

```
public void characters(char ch[], int start, int length) {
    String s = new String(ch, start, length);
    if (isTitle) {
        if (feedTitleHasBeenRead == false) {
            feed.setTitle(s);
            feedTitleHasBeenRead = true;
        }
        else {
            item.setTitle(s);
        }
        isTitle = false;
    }
    else if (isLink) {
        item.setLink(s);
        isLink = false;
    }
    else if (isDescription) {
        if (s.startsWith("<")) {
            item.setDescription("No description available.");
        } else{
            item.setDescription(s);
        }
        isDescription = false;
    }
```

The RSSFeedHandler class (continued)

```
        else if (isPubDate) {
            if (feedPubDateHasBeenRead == false) {
                feed.setPubDate(s);
                feedPubDateHasBeenRead = true;
            }
            else {
                item.setPubDate(s);
            }
            isPubDate = false;
        }
    }
}
```

The RSSFeed class

```
package com.murach.newsreader;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;

public class RSSFeed {
    private String title = null;
    private String pubDate = null;
    private ArrayList<RSSItem> items;

    private SimpleDateFormat dateInFormat =
            new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss Z");

    public RSSFeed() {
        items = new ArrayList<RSSItem>();
    }
```

The RSSFeed class (continued)

```
public void setTitle(String title) {
    this.title = title;
}

public String getTitle() {
    return title;
}

public void setPubDate(String pubDate) {
    this.pubDate = pubDate;
}

public long getPubDateMillis() {
    try {
        Date date = dateInFormat.parse(pubDate.trim());
        return date.getTime();
    }
    catch (ParseException e) {
        throw new RuntimeException(e);
    }
}
```

The RSSFeed class (continued)

```
    public int addItem(RSSItem item) {
        items.add(item);
        return items.size();
    }

    public RSSItem getItem(int index) {
        return items.get(index);
    }

    public ArrayList<RSSItem> getAllItems() {
        return items;
    }
}
```

The RSSItem class

```
package com.murach.newsreader;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class RSSItem {

    private String title = null;
    private String description = null;
    private String link = null;
    private String pubDate = null;

    private SimpleDateFormat dateOutFormat =
        new SimpleDateFormat("EEEE h:mm a (MMM d)");

    private SimpleDateFormat dateInFormat =
        new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss Z");
```

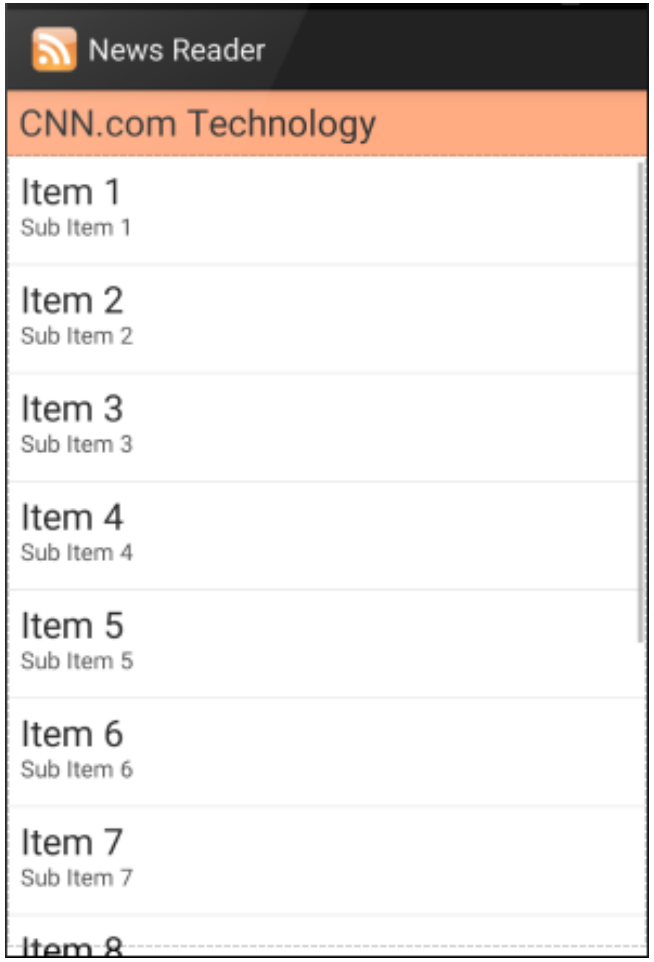
The RSSItem class (continued)

```
public void setTitle(String title) {
    this.title = title;
}

public String getTitle() {
    return title;
}

public void setDescription(String description) {
    this.description = description;
}

public String getDescription() {
    return description;
}

public void setLink(String link) {
    this.link = link;
}

public String getLink() {
    return link;
}
```

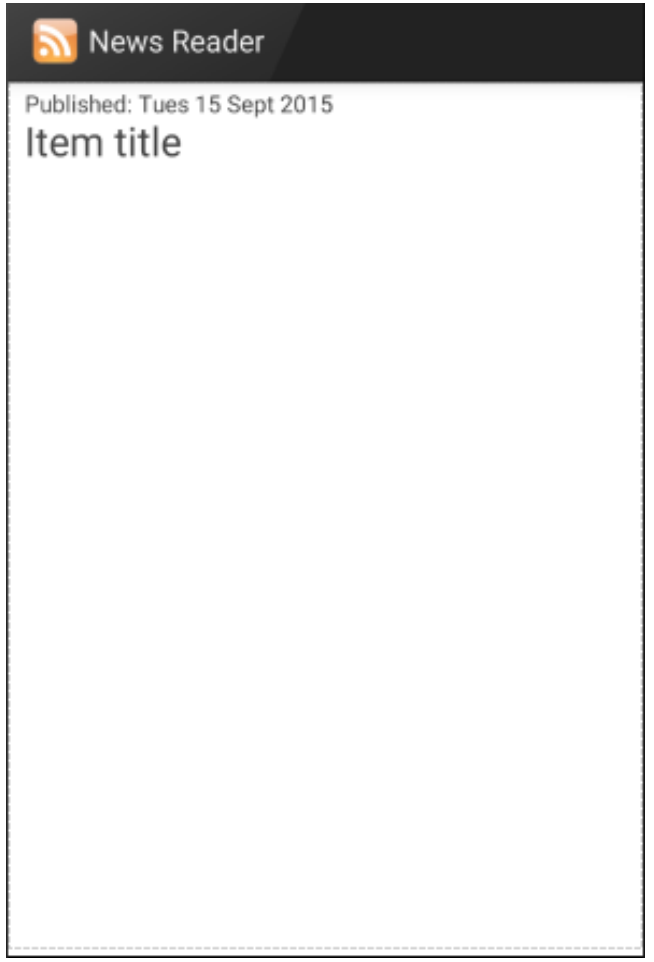
The RSSItem class (continued)

```
    public void setPubDate(String pubDate) {
        this.pubDate = pubDate;
    }

    public String getPubDate() {
        return pubDate;
    }

    public String getPubDateFormatted() {
        try {
            Date date = dateInFormat.parse(pubDate.trim());
            String pubDateFormatted = dateOutFormat.format(date);
            return pubDateFormatted;
        }
        catch (ParseException e) {
            throw new RuntimeException(e);
        }
    }
}
```

activity_items layout



listview_item layout



The ListView widget in the activity_items layout

```
<ListView
    android:id="@+id/itemsListView"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

The listview_item layout

```
<?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:orientation="vertical" >

    <TextView
        android:id="@+id/pubDateTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="5dp"
        android:text="@string/item_pub_date" />
    <TextView
        android:id="@+id/titleTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp"
        android:text="@string/item_title"
        android:textSize="24sp" />

</LinearLayout>
```

Code that creates and sets the adapter

```
// get the items for the feed
ArrayList<RSSItem> items = feed.getAllItems();

// create a List of Map<String, ?> objects
ArrayList<HashMap<String, String>> data =
        new ArrayList<HashMap<String, String>>();
for (RSSItem item : items) {
    HashMap<String, String> map = new HashMap<String, String>();
    map.put("date", item.getPubDateFormatted());
    map.put("title", item.getTitle());
    data.add(map);
}

// create the resource, from, and to variables
int resource = R.layout.listview_item;
String[] from = {"date", "title"};
int[] to = {R.id.pubDateTextView, R.id.titleTextView};

// create and set the adapter
SimpleAdapter adapter =
    new SimpleAdapter(this, data, resource, from, to);
itemsListView.setAdapter(adapter);
```

The constructor for the SimpleAdapter class

Parameter	Description
`context`	The context of the View associated with this SimpleAdapter object.
`data`	A List object that contains Map objects that contain the data for the items in the list.
`resource`	The ID of a layout for each item in the list.
`from`	An array of column names that are in the Map objects.
`to`	An array of IDs for the widgets that should display the data.

How to handle events for an adapter

Step 1: Import the interface for the listener

```
import android.widget.AdapterView.OnItemClickListener;
```

Step 2a: Implement the interface for the listener

```
public class ItemsActivity extends Activity
implements OnItemClickListener {
```

Step 2b: Implement the interface for the listener

```
@Override
public void onItemClick(AdapterView<?> parent, View v,
        int position, long id) {

    // get item at position
    RSSItem item = feed.getItem(position);
}
```

Step 3: Set the listeners

```
itemsListView.setOnItemClickListener(this);
```

Code in the ItemsActivity class

```
// create the intent
Intent intent = new Intent(this, ItemActivity.class);

// put data in the intent
intent.putExtra("title", item.getTitle());
intent.putExtra("position", position);

// start the intent
this.startActivity(intent);
```

Code in the ItemActivity class

```
// get the intent
Intent intent = getIntent();

// get data from the intent
String pubDate = intent.getStringExtra("pubDate");
int position = intent.getIntExtra("position", 0);
```

A constructor and some methods of the Intent class

```
Intent(context, class)
putExtra(name, value)
getStringExtra(name)
getIntExtra(name, default)
```

How to view a URL in a web browser

```
// create a Uri object for the link
String link = "http://rss.cnn.com/~r/rss/cnn_tech/~3/N9m_DSAe5rY/";
Uri uri = Uri.parse(link);

// create the intent and start it
Intent viewIntent = new Intent(Intent.ACTION_VIEW, uri);
startActivity(viewIntent);
```

How to call a phone number

```
// get the Uri for the phone number
String number = "tel:800-111-1111";
Uri callUri = Uri.parse(number);

// create the intent and start it
Intent callIntent = new Intent(Intent.ACTION_DIAL, callUri);
startActivity(callIntent);
```

Another constructor of the Intent class

```
Intent(action, uri)
```

Some action constants of the Intent class

```
ACTION_VIEW
ACTION_DIAL
ACTION_CALL
```

The CALL_PHONE permission in the AndroidManifest.xml file

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

Explicit vs. implicit intents

- An *explicit intent* specifies a component such as an activity.
- You can use an explicit intent to pass data from one activity to another.
- An *implicit intent* specifies the action you want to perform. Then, Android determines the best app to perform that action.
- You can use an implicit intent to view a URL in a web browser or to call a phone number.

The activity_items layout

```
<?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:orientation="vertical" >

    <TextView
        android:id="@+id/titleTextView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#FFAC83"
        android:padding="7dp"
        android:text="@string/items_title"
        android:textSize="22sp" />

    <ListView
        android:id="@+id/itemsListView"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

The listview_item layout

```
<?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:orientation="vertical" >

    <TextView
        android:id="@+id/pubDateTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="5dp"
        android:text="@string/item_pub_date" />

    <TextView
        android:id="@+id/titleTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp"
        android:text="@string/item_title"
        android:textSize="24sp" />

</LinearLayout>
```

The ItemsActivity class

```
package com.murach.newsreader;

import java.util.ArrayList;
import java.util.HashMap;

import android.os.AsyncTask;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.TextView;
import android.widget.AdapterView.OnItemClickListener;
```

The ItemsActivity class (continued)

```
public class ItemsActivity extends Activity
implements OnItemClickListener {

    private RSSFeed feed;
    private FileIO io;

    private TextView titleTextView;
    private ListView itemsListView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_items);

        io = new FileIO(getApplicationContext());

        titleTextView = (TextView) findViewById(R.id.titleTextView);
        itemsListView = (ListView) findViewById(R.id.itemsListView);

        itemsListView.setOnItemClickListener(this);

        new DownloadFeed().execute();
    }
```

The ItemsActivity class (continued)

```
class DownloadFeed extends AsyncTask<Void, Void, Void> {
    @Override
    protected Void doInBackground(Void... params) {
        io.downloadFile();
        return null;
    }

    @Override
    protected void onPostExecute(Void result) {
        Log.d("News reader", "Feed downloaded");
        new ReadFeed().execute();
    }
}

class ReadFeed extends AsyncTask<Void, Void, Void> {
    @Override
    protected Void doInBackground(Void... params) {
        feed = io.readFile();
        return null;
    }
```

The ItemsActivity class (continued)

```
        @Override
        protected void onPostExecute(Void result) {
            Log.d("News reader", "Feed read");

            // update the display for the activity
            ItemsActivity.this.updateDisplay();
        }
    }

    public void updateDisplay() {
        if (feed == null) {
            titleTextView.setText("Unable to get RSS feed");
            return;
        }

        // set the title for the feed
        titleTextView.setText(feed.getTitle());

        // get the items for the feed
        ArrayList<RSSItem> items = feed.getAllItems();
```

The ItemsActivity class (continued)

```
        // create a List of Map<String, ?> objects
        ArrayList<HashMap<String, String>> data =
                new ArrayList<HashMap<String, String>>();
        for (RSSItem item : items) {
            HashMap<String, String> map =
                new HashMap<String, String>();
            map.put("date", item.getPubDateFormatted());
            map.put("title", item.getTitle());
            data.add(map);
        }

        // create the resource, from, and to variables
        int resource = R.layout.listview_item;
        String[] from = {"date", "title"};
        int[] to = {R.id.pubDateTextView,
                    R.id.titleTextView};
        // create and set the adapter
        SimpleAdapter adapter =
            new SimpleAdapter(
                this, data, resource, from, to);
        itemsListView.setAdapter(adapter);

        Log.d("News reader", "Feed displayed");
    }
```

The ItemsActivity class (continued)

```
    @Override
    public void onItemClick(AdapterView<?> parent, View v,
            int position, long id) {

        // get the item at the specified position
        RSSItem item = feed.getItem(position);

        // create an intent
        Intent intent = new Intent(this, ItemActivity.class);

        intent.putExtra("pubdate", item.getPubDate());
        intent.putExtra("title", item.getTitle());
        intent.putExtra("description", item.getDescription());
        intent.putExtra("link", item.getLink());

        this.startActivity(intent);
    }
}
```

The FileIO class

```
package com.murach.newsreader;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.URL;

import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.xml.sax.InputSource;
import org.xml.sax.XMLReader;

import android.content.Context;
import android.util.Log;

public class FileIO {

    private final String URL_STRING =
       "http://rss.cnn.com/rss/cnn_tech.rss";
    private final String FILENAME = "news_feed.xml";
    private Context context = null;
```

The FileIO class (continued)

```
public FileIO (Context context) {
    this.context = context;
}

public void downloadFile() {
    try{
        // get the URL
        URL url = new URL(URL_STRING);

        // get the input stream
        InputStream in = url.openStream();

        // get the output stream
        FileOutputStream out =
            context.openFileOutput(FILENAME,
        Context.MODE_PRIVATE);
```

The FileIO class (continued)

```
            // read input and write output
            byte[] buffer = new byte[1024];
            int bytesRead = in.read(buffer);
            while (bytesRead != -1)
            {
                out.write(buffer, 0, bytesRead);
                bytesRead = in.read(buffer);
            }
            out.close();
            in.close();
        }
        catch (IOException e) {
            Log.e("News reader", e.toString());
        }
    }
```

The FileIO class (continued)

```
public RSSFeed readFile() {
    try {
        // get the XML reader
        SAXParserFactory factory = SAXParserFactory.newInstance();
        SAXParser parser = factory.newSAXParser();
        XMLReader xmlreader = parser.getXMLReader();

        // set content handler
        RSSFeedHandler theRssHandler = new RSSFeedHandler();
        xmlreader.setContentHandler(theRssHandler);

        // read the file from internal storage
        FileInputStream in = context.openFileInput(FILENAME);

        // parse the data
        InputSource is = new InputSource(in);
        xmlreader.parse(is);

        // set the feed in the activity
        RSSFeed feed = theRssHandler.getFeed();
        return feed;
    }
```

The FileIO class (continued)

```
        catch (Exception e) {
            Log.e("News reader", e.toString());
            return null;
        }
    }
}
```

The activity_item layout

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >

        <TextView
            android:id="@+id/titleTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingLeft="7dp"
            android:paddingRight="7dp"
            android:paddingTop="5dp"
            android:text="@string/item_title"
            android:textSize="24sp"
            android:textStyle="bold" />
```

The activity_item layout (continued)

```
<TextView
    android:id="@+id/pubDateTextView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingLeft="7dp"
    android:paddingTop="5dp"
    android:text="@string/item_pub_date" />

<TextView
    android:id="@+id/descriptionTextView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingLeft="7dp"
    android:paddingRight="7dp"
    android:paddingTop="5dp"
    android:text="@string/item_description"
    android:textSize="18sp" />
```

The activity_item layout (continued)

```
        <TextView
            android:id="@+id/linkTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:paddingLeft="7dp"
            android:paddingTop="5dp"
            android:text="@string/item_link"
            android:textColor="@color/blue"
            android:textSize="18sp" />

    </LinearLayout>
</ScrollView>
```

The ItemActivity class

```
package com.murach.newsreader;

import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.TextView;
import android.app.Activity;
import android.content.Intent;

public class ItemActivity extends Activity
implements OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_item);

        // get references to widgets
        TextView titleTextView = (TextView)
            findViewById(R.id.titleTextView);
```

The ItemActivity class (continued)

```
        TextView pubDateTextView = (TextView)
            findViewById(R.id.pubDateTextView);
        TextView descriptionTextView = (TextView)
            findViewById(R.id.descriptionTextView);
        TextView linkTextView = (TextView)
            findViewById(R.id.linkTextView);

        // get the intent and its data
        Intent intent = getIntent();
        String pubDate = intent.getStringExtra("pubdate");
        String title = intent.getStringExtra("title");
        String description =
            intent.getStringExtra("description").replace('\n', ' ');

        // display data on the widgets
        pubDateTextView.setText(pubDate);
        titleTextView.setText(title);
        descriptionTextView.setText(description);

        // set the listener
        linkTextView.setOnClickListener(this);
    }
```

The ItemActivity class (continued)

```
    @Override
    public void onClick(View v) {
        // get the intent and create the Uri for the link
        Intent intent = getIntent();
        String link = intent.getStringExtra("link");
        Uri viewUri = Uri.parse(link);

        // create the intent and start it
        Intent viewIntent = new Intent(Intent.ACTION_VIEW, viewUri);
        startActivity(viewIntent);
    }
}
```