Android is a mobile operating system that is based on a modified version of Linux. It was originally developed by a startup of the same name, Android, Inc. In 2005, as part of its strategy to enter the mobile space, Google purchased Android and took over its development work (as well as its development team).

Google wanted Android to be open and free; hence, most of the Android code was released under the open-source Apache License, which means that anyone who wants to use Android can do so by downloading the full Android s Moreover, vendors (typically hardware manufacturers) can add their own proprietary extensions to Android and customize Android to differentiate their products from others. This simple development model makes Android very attractive and has thus piqued the interest of many vendors. This has been especially true for companies affected by the phe- nomenon of Apple's iPhone, a hugely successful product that revolutionized the smartphone industry. Such companies include Motorola and Sony Ericsson, which for many years have been developing their own mobile operating systems. When the iPhone was launched, many of these manufacturers had to scramble to find new ways of revitalizing their products. These manufacturers see Android as a solution — they will continue to design their own hardware and use Android as the operating sys- tem that powers it.

The main advantage of adopting Android is that it offers a unified approach to application development. Developers need only develop for Android, and their applications should be able to run on numerous different devices, as long as the devices are powered using Android. In the world of smartphones, applications are the most important part of the success chain. Device manufacturers therefore see Android as their best hope to challenge the onslaught of the iPhone, which already commands a large base of applications.

Features of Android

As Android is open source and freely available to manufacturers for customization, there are no fixed hardware and software configurations. However, Android itself supports the following features:

- **Storage —** Uses SQLite, a lightweight relational database, for data storage. Chapter 6 discusses data storage in more detail.
- **Connectivity** — Supports GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth (includes A2DP and AVRCP), WiFi, LTE, and WiMAX. Chapter 8 discusses networking in more detail.
- **Messaging —** Supports both SMS and MMS. Chapter 8 discusses messaging in more detail.
- **Web browser —** Based on the open-source WebKit, together with Chrome's V8 JavaScript engine
- **Media support** — Includes support for the following media: H.263, H.264 (in 3GP or MP4 container), MPEG-4 SP, AMR, AMR-WB (in 3GP container), AAC, HE-AAC (in MP4 or 3GP container), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP
- **Hardware support —** Accelerometer Sensor, Camera, Digital Compass, Proximity Sensor, and GPS
- **Multi-touch —** Supports multi-touch screens
- **Multi-tasking —** Supports multi-tasking applications
- **Flash support —** Android 2.3 supports Flash 10.1.
- **Tethering —** Supports sharing of Internet connections as a wired/wireless hotspot

In order to understand how Android works, take a look at Figure 1-1, which shows the various layers that make up the Android operating system (OS).

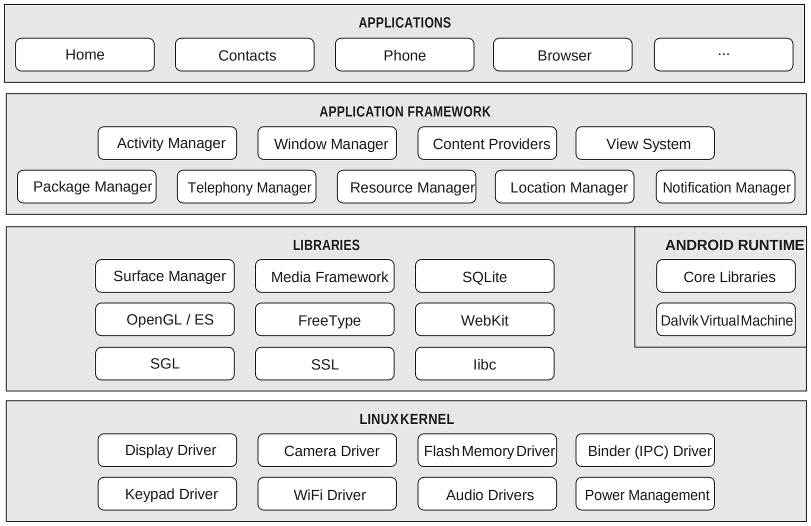


Figure 1-1

The Android OS is roughly divided into five sections in four main layers:

- **Linux kernel** — This is the kernel on which Android is based. This layer contains all the low-level device drivers for the various hardware components of an Android device.
- **Libraries** — These contain all the code that provides the main features of an Android OS. For example, the SQLite library provides database support so that an application can use it for data storage. The WebKit library provides functionalities for web browsing.
- **Android runtime** — At the same layer as the libraries, the Android runtime provides a set of core libraries that enable developers to write Android apps using the Java programming language. The Android runtime also includes the Dalvik virtual machine, which enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine (Android applications are compiled into the Dalvik executables). Dalvik is a specialized virtual machine designed specifically for Android and optimized for battery-powered mobile devices with limited memory and CPU.
- **Application framework** — Exposes the various capabilities of the Android OS to application developers so that they can make use of them in their applications.
- **Applications** — At this top layer, you will find applications that ship with the Android device (such as Phone, Contacts, Browser, etc.), as well as applications that you download and install from the Android Market. Any applications that you write are located at this layer.

V

Table 1. Android versions

Code name	**Version**	**API level**
Oreo	8.0	26
Nougat	7.0 – 7.1.1	24 -25
Marshmallow	6.0	23
Lollipop	5.1	22
Lollipop	5.0	21
KitKat	4.4 - 4.4.4	19
Jelly Bean	4.1.x - 4.3.x	16 - 18
Ice Cream Sandwich	4.0.1 - 4.0.4	14 - 15
Honeycomb	3.2.x	13
Honeycomb	3.0 - 3.1	11 - 12
Gingerbread	2.3 - 2.3.7	9-10
Froyo	2.2.x	8
Eclair	2.1	7
Eclair	2.0 - 2.0.1	5 -6
Donut	1.6	4
Cupcake	1.5	3
(no code name)	1.1	2
(no code name)	1.0	1

History of Android

The history and versions of android are interesting to know. The code names of android ranges from AtoJcurrently, such as Aestro, Blender, Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwitch, Jelly Bean, KitKat and Lollipop. Let's understand the android history in a sequence.

1) Initially, **Andy Rubin** founded Android Incorporation in Palo Alto, California, United States in October, 2003.

2) In 17th August 2005, Google acquired android Incorporation. Since then, it is in the subsidiary of Google Incorporation.

3) The key employees of Android Incorporation are **Andy Rubin**, **Rich Miner**, **Chris White**and **Nick Sears**.

4) Originally intended for camera but shifted to smart phones later because of low market for camera only.

5) Android is the nick name of Andy Rubin given by coworkers because of his love to robots.

6) In 2007, Google announces the development of android OS.

7) In 2008, HTC launched the first android mobile.

The Android operating system is published in different versions. The following table gives an overview of the available versions.

Android Applications

Android applications are usually developed in the Java language using the Android Software Development Kit.

Once developed, Android applications can be packaged easily and sold out either through a store such as **Google Play**, **SlideME**, **Opera Mobile Store**, **Mobango**, **F-droid** and the **Amazon Appstore**.

Android powers hundreds of millions of mobile devices in more than 190 countries around the world. It's the largest installed base of any mobile platform and growing fast. Every day more than 1 million new Android devices are activated worldwide.

This tutorial has been written with an aim to teach you how to develop and package Android application. We will start from environment setup for Android application programming and then drill down to look into various aspects of Android applications.

Categories of Android applications

There are many android applications in the market. The top categories are −

Music	News	Multimedia
Sports	Lifestyle	Food & Drink
Travel	Weather	Books
Business	Reference	Navigation
Social Media	Utilities	Finance

Android devices in the market

Android devices come in all shapes and sizes. As of late November 2010, the Android OS can be seen powering the following types of devices:

- Smartphones
- Tablets
- E-reader devices
- Netbooks
- MP4 players
- Internet TVs

Chances are good that you own at least one of the preceding dev amsung Galaxy S, the HTC Desire HD, and the LG Optimus One smartphones.

Another popular category of devices that manufacturers are rushing out is the *tablet*. Tablet sizes typically start at seven inches, measured diagonally. Figure 1-3 shows the Samsung Galaxy Tab and the Dell Streak, which is a five-inch phone tablet.

Besides smartphones and tablets, Android is also beginning to appear in dedicated devices, such as e-book readers. Figure 1-4 shows the Barnes and Noble's NOOKcolor, which is a color e-Book reader running the Android OS.

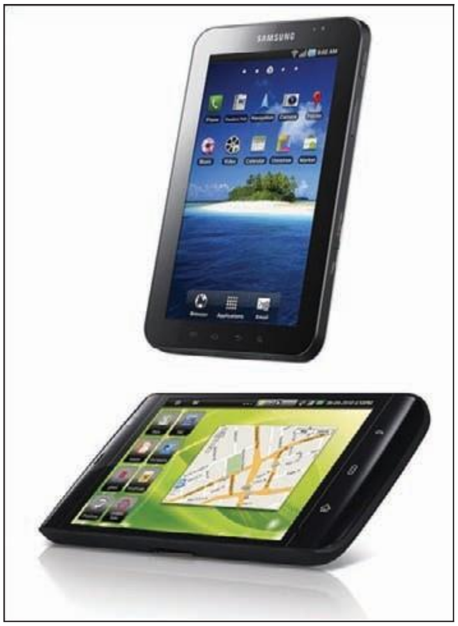


Figure 1-3

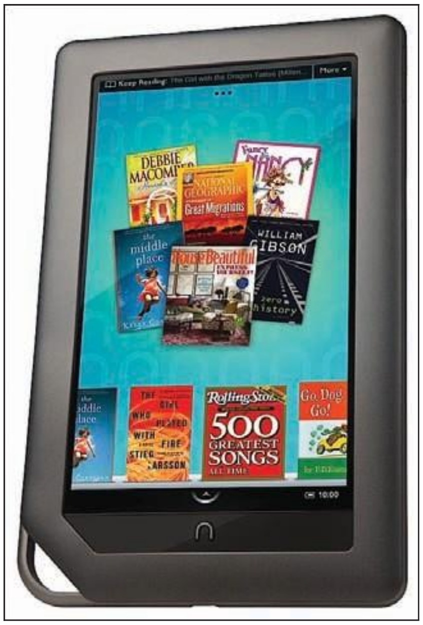


Figure 1-4

In addition to these popular mobile devices, Android is also slowly finding its way into your living room. People of Lava, a Swedish company, has developed an Android-based TV, call the Scandinavia Android TV (see Figure 1-5).

Google has also ventured into a proprietary smart TV platform based on Android and co-developed with companies such as Intel, Sony, and Logitech. Figure 1-6 shows Sony's Google TV.

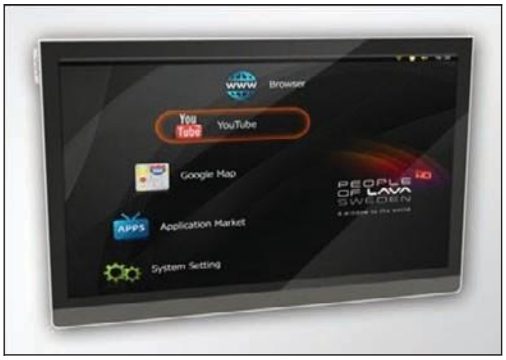


Figure 1-5

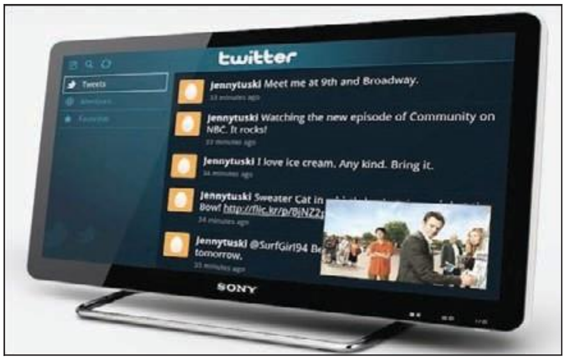


Figure 1-6

oBtAining the reQuiredtoolS

Now that start writing some applications! Before you write your first app, however, you need to download the required tools and SDKs.

For Android development, you can use a Mac, a Windows PC, or a Linux machine. All the tools needed are free and can be downloaded from the Web. Most of the examples provided in this book should work fine with the Android emulator, with the exception of a few examples that require access to the hardware. For this book, I will be using a Windows 7 computer to demonstrate all the code samples. If you are using a Mac or Linux computer, the screenshots should look similar; some minor differences may be present, but you should be able to follow along without problems.

So, let the fun begin!

jAvA jdk

The Android SDK makes use of the Java SE Development Kit (JDK). Hence, if your computer does not have the JDK installed, you should start by downloading the JDK from `www.oracle.com/technetwork/java/javase/downloads/index.html` and installing it prior to moving to the next section.

Android Sdk

The next important piece of software you need to download is, of course, the Android SDK. The Android SDK contains a debugger, libraries, an emulator, documentation, sample code, and tutorials.

You can download the Android SDK from `http://developer.android.com/sdk/index.html`.

Once the SDK is downloaded, unzip its content (the `android-sdk-windows` folder) into the `C:\Android\` folder, or whatever name you have given to the folder you just created.

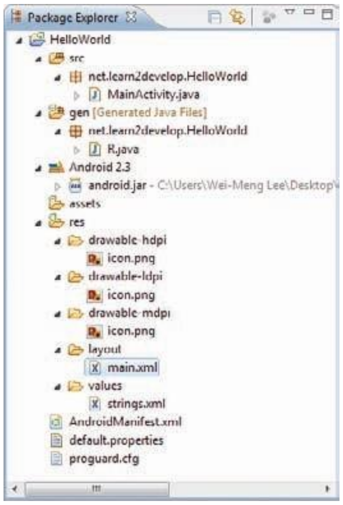
Now that you have created your first Hello World Android application, it is time to dissect the innards of the Android project and examine all the parts that make everything work.

First, note the various files that make up an Android project in the Package Explorer in Eclipse (see Figure 1-30).

The various folders and their files are as follows:

- `src` — Contains the `.java` source files for your project. In this example, there is one file, `MainActivity.java`. The `MainActivity.java` file is the source file for your activity. You will write the code for your application in this file.
- `Android 2.3` library — This item contains one file, `android.jar`, which contains all the class libraries needed for an Android application.

- gen — Contains the `R.java` file, a compiler-generated file that references all the resources found in your project. You should not modify this file.
- `assets` — This folder contains all the assets used by your application, such as HTML, text files, databases, etc.



```
Package Explorer
HelloWorld
  src
    net.learn2develop.HelloWorld
      MainActivity.java
  gen [Generated Java Files]
    net.learn2develop.HelloWorld
      R.java
  Android 2.3
    android.jar - C:\Users\Wei-Meng Lee\Desktop\
  assets
  res
    drawable-hdpi
      icon.png
    drawable-ldpi
      icon.png
    drawable-mdpi
      icon.png
    layout
      main.xml
    values
      strings.xml
  AndroidManifest.xml
  default.properties
  proguard.cfg
```

Figure 1-30

- res — This folder contains all the resources used in your application. It also contains a few other subfolders: drawable-<*resolution*>, layout, and values. Chapter 3 talks more about how you can support devices with different screen resolutions and densities.
- AndroidManifest.xml — This is the manifest file for your Android application. Here you specify the permissions needed by your application, as well as other features (such as intent-filters, receivers, etc.). Chapter 2 discusses the use of the AndroidManifest.xml file in more details.

The main.xml file defines the user interface for your activity. Observe the following in bold:

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello" />
```

The @string in this case refers to the **STRINGS.XML** file located in the res/values folder. Hence, @string/hello refers to the hello string defined in the strings.xml file, which is "Hello World, MainActivity!":

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, MainActivity!</string>
    <string name="app_name">HelloWorld</string>
</resources>
```

It is recommended that you store all the string constants in your application in this strings.xml file and reference these strings using the @string identifier. That way, if you ever need to localize your application to another language, all you need to do is replace the strings stored in the strings.xml file with the targeted language and recompile your application.

Observe the content of the AndroidManifest.xml file:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.learn2develop.HelloWorld"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="9" />
</manifest>
```

The AndroidManifest.xml file contains detailed information about the application:

- It defines the package name of the application as net.learn2develop.HelloWorld.
- The version code of the application is 1. This value is used to identify the version number of your application. It can be used to programmatically determine whether an application needs to be upgraded.
- The version name of the application is 1.0. This string value is mainly used for display to the user. You should use the format: *<major>.<minor>.<point>* for this value.

- The application uses the image named `icon.png` located in the `drawable` folder

The name of this application is the string named `app_name` defined in the `strings.xml` file.

- There is one activity in the application represented by the `MainActivity.java` file. The label displayed for this activity is the same as the application name.
- Within the definition for this activity, there is an element named `<intent-filter>`:
  - The action for the intent filter is named `android.intent.action.MAIN` to indicate that this activity serves as the entry point for the application.
  - The category for the intent-filter is named `android.intent.category.LAUNCHER` to indicate that the application can be launched from the device's Launcher icon. Chapter 2 discusses intents in more details.
- Finally, the `android:minSdkVersion` attribute of the `<uses-sdk>` element specifies the minimum version of the OS on which the application will run.

Finally, the code that connects the activity to the UI (`main.xml`) is the `setContentView()` method, which is in the `MainActivity.java` file:

```
package net.learn2develop.HelloWorld;

import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
        { super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Here, `R.layout.main` refers to the `main.xml` file located in the `res/layout` folder. As you add additional XML files to the `res/layout` folder, the filenames will automatically be generated in the `R.java` file. The `onCreate()` method is one of many methods that are fired when an activity is loaded. Chapter 2 discusses the life cycle of an activity in more detail.

To create an activity, you create a Java class that extends the `Activity` base class:

```
package  net.learn2develop.Activities;


import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
        { super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Your activity class would then load its UI component using the XML file defined in your `res/layout` folder. In this example, you would load the UI from the `main.xml` file:

```
        setContentView(R.layout.main);
```

Every activity you have in your application must be declared in your `AndroidManifest.xml` file, like this:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.learn2develop.Activities"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon"
        android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
                    android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="9" />
</manifest>
```

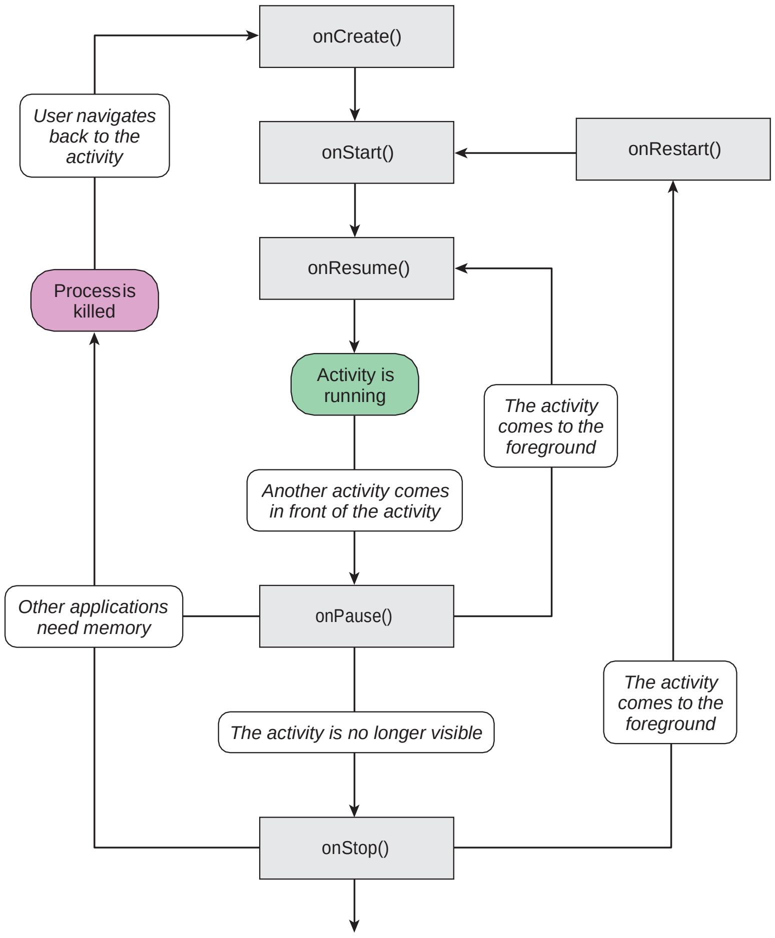
The `Activity` base class defines a series of events that governs the life cycle of an activity. The `Activity` class defines the following events:

- `onCreate()` — Called when the activity is first created
- `onStart()` — Called when the activity becomes visible to the user
- `onResume()` — Called when the activity starts interacting with the user
- `onPause()` — Called when the current activity is being paused and the previous activity is being resumed
- `onStop()` — Called when the activity is no longer visible to the user
- `onDestroy()` — Called before the activity is destroyed by the system (either manually or by the system to conserve memory)
- `onRestart()` — Called when the activity has been stopped and is restarting again

By default, the activity created for you contains the `onCreate()` event. Within this event handler is the code that helps to display the UI elements of your screen.
Figure 2-1 shows the life cycle of an activity and the various stages it goes through — from when the activity is started until it ends.

↓

Method	Description
onCreate	called when activity is first created.
onStart	called when activity is becoming visible to the user.
onResume	called when activity will start interacting with the user.
onPause	called when activity is not visible to the user.
onStop	called when activity is no longer visible to the user.
onRestart	called after your activity is stopped, prior to start.
onDestroy	called before the activity is destroyed.


onCreate()
User navigates back to the activity
onStart()
onRestart()
onResume()
Process is killed
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
Other applications need memory
onPause()
The activity is no longer visible
The activity comes to the foreground
onStop()

```
package net.learn2develop.Activities;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;

public class MainActivity extends Activity {
    String tag = "Events";

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
        { super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
       Log.d(tag, "In the onCreate() event");




          }
          public void onStart()
          {
              super.onStart();
              Log.d(tag, "In the onStart()  event");
          }
          public void onRestart()
          {
              super.onRestart();
              Log.d(tag, "In the onRestart() event");
          }
          public void onResume()
          {
              super.onResume();
              Log.d(tag, "In the onResume() event");
          }
          public void onPause()
          {
              super.onPause();
              Log.d(tag, "In the onPause()  event");
          }
          public void onStop()
          {
              super.onStop();
              Log.d(tag, "In the onStop()  event");
          }
          public void onDestroy()
          {
              super.onDestroy();
              Log.d(tag, "In the onDestroy() event");
          }
      }
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello" />
<Button
    android:id="@+id/btn_dialog"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Click to display a dialog" />
</LinearLayout>
```

```
package net.learn2develop.Dialog;

import android.app.Activity;
import android.os.Bundle;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity{
    CharSequence[] items = { "Google", "Apple", "Microsoft" };
    boolean[] itemsChecked = new boolean [items.length];

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
        { super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button btn = (Button) findViewById(R.id.btn_dialog);
        btn.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v)
                { showDialog(0);
            }
        });
    }

    @Override
    protected Dialog onCreateDialog(int id)
        { switch (id) {
        case 0:
            return new AlertDialog.Builder(this)
            .setIcon(R.drawable.icon)
            .setTitle("This is a dialog with some simple text...")
            .setPositiveButton("OK", new
                    DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog,
                    int whichButton)
                    {
                       Toast.makeText(getBaseContext(),
                              "OK clicked!", Toast.LENGTH_SHORT).show();
                    }
                })
                .setNegativeButton("Cancel", new
                    DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog,
                         int whichButton)
                    {
                       Toast.makeText(getBaseContext(),
                                 "Cancel clicked!", Toast.LENGTH_SHORT).show();
                    }
                })



          .setMultiChoiceItems(items, itemsChecked, new
              DialogInterface.OnMultiChoiceClickListener() {
                  @Override
                  public void onClick(DialogInterface dialog, int which,
                  boolean isChecked) {
                      Toast.makeText(getBaseContext(),
                          items[which] + (isChecked ? " checked!":
                          " unchecked!"),
                          Toast.LENGTH_SHORT).show();
                  }
              }
          )
          .create();
      }
      return null;
  }
}
```

```
package net.learn2develop.Dialog;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import android.app.ProgressDialog;
import android.os.Handler;
import android.os.Message;
```

```
public class MainActivity extends Activity{
    CharSequence[] items = { “Google”, “Apple”, “Microsoft” };
    boolean[] itemsChecked = new boolean [items.length];

    private ProgressDialog _progressDialog;
    private int _progress =0;
    private Handler _progressHandler;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
        { super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button btn = (Button) findViewById(R.id.btn_dialog);
        btn.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                showDialog(1);
                _progress = 0;
                _progressDialog.setProgress(0);
                _progressHandler.sendEmptyMessage(0);
            }
        });

        _progressHandler = new Handler() {
            public void handleMessage(Message msg)
                { super.handleMessage(msg);
                if (_progress >= 100) {
                    _progressDialog.dismiss();
                } else {
                    _progress++;
                    _progressDialog.incrementProgressBy(1);
                    _progressHandler.sendEmptyMessageDelayed(0, 100);
                }
            }
        };
    }

    @Override
    protected Dialog onCreateDialog(int id)
        { switch (id) {
        case 0:
            return new AlertDialog.Builder(this)
            //...
            //...
            .create();
        case 1:
            _progressDialog = new ProgressDialog(this);
            _progressDialog.setIcon(R.drawable.icon);
            _progressDialog.setTitle("Downloading files...");
            _progressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
            _progressDialog.setButton(DialogInterface.BUTTON_POSITIVE, "Hide", new
                DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog,
                    int whichButton)
                {
                    Toast.makeText(getBaseContext(),
                            "Hide clicked!", Toast.LENGTH_SHORT).show();
                }
            });
            _progressDialog.setButton(DialogInterface.BUTTON_NEGATIVE, "Cancel", new
                DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog,
                    int whichButton)
                {
                    Toast.makeText(getBaseContext(),
                            "Cancel clicked!", Toast.LENGTH_SHORT).show();
                }
            });
            return _progressDialog;
        }
        return null;
    }
}
```

2.

To apply a dialog theme to an activity, simply modify the `<Activity>` element in the `AndroidManifest.xml` file by adding the `android:theme` attribute:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.learn2develop.Activities"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon"
       android:label="@string/app_name">
       <activity android:name=".MainActivity"
                 android:label="@string/app_name"
                 android:theme="@android:style/Theme.Dialog" >
           <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
                    android:name="android.intent.category.LAUNCHER" />
           </intent-filter>
       </activity>
    </application>
    <uses-sdk android:minSdkVersion="9" />
</manifest>
```

Until this point, you have seen how to call activities within your own application. One of the key aspects of Android programming is using the intent to call activities from other applications. In particular, your application can call the many built-in applications that are included with an Android device. For example, if your application needs to enable a user to call a particular person saved in the Contacts application, you can simply use an `Intent` object to bring up the Contacts application, from which the user can select the person to call. This enables your application to present a consistent user experience, and enables you to avoid building another application to retrieve all the contacts in the Contacts application.

The following Try It Out demonstrates how to call some of the built-in applications commonly found on an Android device.

try it out Calling Built-in Applications Using intents

codefile Intents.zip available for download at Wrox.com

1. Using Eclipse, create a new Android project and name it **Intents**.

2. Add the following statements in bold to the `main.xml` file:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
<Button
    android:id="@+id/btn_webbrowser"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Web Browser" />
<Button
    android:id="@+id/btn_makecalls"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Make Calls" />
<Button
    android:id="@+id/btn_showMap"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Show Map" />
<Button
    android:id="@+id/btn_chooseContact"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Choose Contact" />
</LinearLayout>
```

3. Add the following statements in bold to the `MainActivity.java` file:

```
package net.learn2develop.Intents;

import android.app.Activity;
import android.os.Bundle;

import android.content.Intent;
import android.net.Uri;
import android.provider.ContactsContract;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity {

    Button b1, b2, b3, b4;
    int request_Code = 1;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
        { super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

       //---Web browser button---
    b1 = (Button) findViewById(R.id.btn_webbrowser);
    b1.setOnClickListener(new OnClickListener()
    {
        public void onClick(View
            arg0){ Intent i = new
                Intent(android.content.Intent.ACTION_VIEW,
                  Uri.parse("http://www.amazon.com"));
            startActivity(i);
        }
    });

    //---Make calls button---
    b2 = (Button) findViewById(R.id.btn_makecalls);
    b2.setOnClickListener(new OnClickListener()
    {
        public void onClick(View
            arg0){ Intent i = new
                Intent(android.content.Intent.ACTION_DIAL,
              Uri.parse("tel:+651234567"));
            startActivity(i);
        }
    });

    //---Show Map button---
    b3 = (Button) findViewById(R.id.btn_showMap);
    b3.setOnClickListener(new OnClickListener()
    {
        public void onClick(View
            arg0){ Intent i = new
                Intent(android.content.Intent.ACTION_VIEW,
              Uri.parse("geo:37.827500,-122.481670"));
            startActivity(i);
        }
    });

    //---Choose Contact button---
    b4 = (Button) findViewById(R.id.btn_chooseContact);
    b4.setOnClickListener(new OnClickListener()
    {
        public void onClick(View
            arg0){ Intent i = new
            Intent(android.content.Intent.ACTION_PICK);
            i.setType(ContactsContract.Contacts.CONTENT_TYPE);
            startActivityForResult(i,request_Code);
        }
    });
}

public void onActivityResult(int requestCode, int resultCode, Intent data)
{
    if (requestCode == request_Code)
    {
         if (resultCode == RESULT_OK)
           {
```

```
                          Toast.makeText(this,data.getData().toString(),
                              Toast.LENGTH_SHORT).show();
                          Intent i = new Intent(
                                  android.content.Intent.ACTION_VIEW,
                                  Uri.parse(data.getData().toString()));
                         startActivity(i);
                      }
              }
          }
}
```