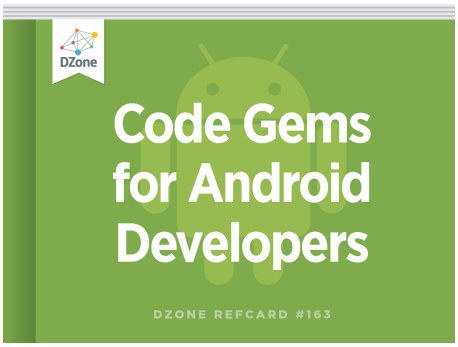
DZone



Refcard #163

Code Gems for Android Developers

Working with the World's Most Popular Mobile OS

Written by **Avi Yehuda**
Developer, RSA

Provides an overview of the API and Tools to develop applications using the Java platform.

SECTION 1

What is Android?

Android is a stack of operating system, middle ware and applications developed by Google that is dedicated to mobile devices. Android relies on Linux kernel 2.6 for core services. The Android platform provides API and Tools to develop applications using Java Platform. It features Dalvik Virtual Machine, SQLite, an integrated browser, application framework, as well as various media and hardware support.

SECTION 2

Basic Concepts

The following table outlines the key concepts in an Android application:

CONCEPT	DESCRIPTION
Activity	Activity is the presenter of a single screen in the application. It has certain abilities, like displaying views, menus, alerts and notifications. It can also call another Activity, which means opening a new screen. Activity is a class that derives from an *android.app.Activity*. An application needs to have at least one Activity. All Activities must be declared in the manifest file.
View	A view is a single user interface element. It handles user events and draws the component on the screen. Views can contain other Views, these are called view groups. A View is a class that derives from *android.view.View*. There are already many existing views. The developer can use them or create his own customized view by extending any of them.
Intent	Intent is the negotiator between two activities or between two applications. It gives the ability to pass messages and data between the two entities. When writing applications for mobile, Intent gives access to OS services like opening the camera, a browser, displaying notifications and so on.
Service	A Service is an application that has the ability to run in the background without displaying any user interface. A Service is a class that derives from *android.app.Service*. All Services must be declared in the manifest file.

SECTION 3

Development

Installing SDK

Download and install the Android SDK
http://developer.android.com/sdk/

Android Development Tools (ADT) Eclipse plugin

Download install and configure ADT for Eclipse.
http://developer.android.com/sdk/eclipse-adt.html

Creating a Project with Eclipse

- Select File > New > Android Project.
- Enter Project Name. Press 'Next'.
- Select Build Target according to the wanted SDK version. Press 'Next'.
- Enter Application name – this is how the user sees the title of your application.
- Enter Package name – Android demands each application to declare its root package.
- Create Activity + Activity name.
- Minimum SDK Version - If you're unsure of the appropriate API Level to use, copy the API Level listed for the Build Target you selected in the Target tab.
- Click Finish

You can also develop Android applications without Eclipse, but since we are developing in Java, Eclipse makes it easier to develop.

SECTION 4

An Android Project Structure

Manifest file
AndroidManifest.xml defines the Android application. It contains the attributes, activities, versions, permissions, and other parameters of the application.

'src' folder
As with any Java project, this folder holds all Java source code and packages.

'res' folder
Contains local resources for the application:

- 'drawable' – image folders according to resolutions. By default there are 3 folders for 3 basic resolutions.
- 'layout' – xml files which represent display layout. By default a main.xml is created.
- 'values' – xml files which define global constants, strings, styles or colors.

SDK jar
Contains the android.jar which is different across versions of the SDK.

'gen' folder
This folder contains the R class which is automatically generated by the Eclipse plugin and gives access to the project resources.

'assets' folder
This folder holds other raw resource files such as movie or sound files. By default, this folder is not created. These resources will not be modified.

Creating Android Virtual Device (AVD)

The developer can test his applications on his own device or on an emulator, which comes along with Android SDK.
But first the developer has to define a virtual device that suits his needs.

To create an AVD from Eclipse:

- Select Window > Android SDK and AVD Manager, or click the Android SDK and AVD Manager icon in the Eclipse toolbar.
- In the Virtual Devices panel, you'll see a list of existing AVDs. Click New to create a new AVD.
- Fill in the details for the AVD and click "Create AVD".

Signing and Generating jars

Android applications are zipped to jar files. The only difference is that Android jar files have a special extension - .apk.
All application jars have to be signed before they are installed.

For more instructions read http://developer.android.com/guide/publishing/app-signing.html

SECTION 5

User Interface

Android generates user interfaces either from XML or from Java code.

Views

Creating and adding a new View to a Layout in XML

```
<?xml version="1.0" encoding="utf-8"?>
  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent" android:layout_height="fill_parent" android:id="@+id/mainLayout" >
  <TextView
  android:text="My text view"
  android:id="@+id/TextViewExample"

  android:layout_width="wrap_content" android:layout_height="wrap_content"></TextView>
 </LinearLayout>
```

View Groups

A ViewGroupis a special view that can contain other views.

List view

```
ListView list = new ListView(this);
  String[] listItems = {"Option 1","Option 2","Option 3"};
  list.setAdapter(new ArrayAdapter<String>(this,
  android.R.layout.simple_list_item_1, listItems ));
 list.setOnItemClickListener(new OnItemClickListener() {...
```

Layouts

A Layout is a type of GroupView. It holds views in a certain layout on the screen.

Adding a Layout - XML example - adding to the layout XML

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

The layout xml is placed in <project_path>/res/layouts

Adding a Layout - Code example
This would be implemented in an Activity class.

```
LinearLayout innerLayout = new LinearLayout(this);
  innerLayout.setPadding(20,20,20,20);
  innerLayout.setGravity(Gravity.CENTER);
 outerLayout.addView(innerLayout );
```

Global Strings

A global string, or an array of strings, are declared in an external xml file in the resource folder:<project>/res/values/strings.xml

Declaring a global string

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

Using a global string while creating a TextView

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

Code usage

```
String hello = context.getString(R.string.hello);

  Global String Array
  Declaration in a Global String Array
  <?xml version="1.0" encoding="utf-8"?>
  <resources>
  <string-array name="planets_array">
  <item>Mercury</item>
  <item>Venus</item>
  <item>Earth</item>
   <item>Mars</item>
  </string-array>
 </resources>
```

Code usage

```
String[] planets = context.getResources().getStringArray(R.array.planets_array);
```

Menus

Options Menu

Options Menu

An Options menu is presented when the user presses the menu button while the Activity is active.
Step 1 - Creating the menu XML

```
<?xml version="1.0" encoding="utf-8"?>
  <menu xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:id="@+id/new_game"
   android:icon="@drawable/ic_new_game"
  android:title="@string/new_game" />
  <item android:id="@+id/quit"
  android:icon="@drawable/ic_quit"
   android:title="@string/quit" />
 </menu>
```

The layout xml is placed in <project_path>/res/menu

Step 2 – Displaying the menu (implemented in Activity class)

```
@Override
  public boolean onCreateOptionsMenu(Menu menu) {
  MenuInflater inflater = getMenuInflater();
   inflater.inflate(R.menu.game_menu, menu);
  return true;
 }
```

Step 3 – Listening to user choice (implemented in Activity class)

```
@Override
  public boolean onOptionsItemSelected(MenuItem item) {
   // Handle item selection
  switch (item.getItemId()) {
  case R.id.new_game:
  newGame();
  return true;
  case R.id.quit:
  quit();
   return true;
   default:
   return super.onOptionsItemSelected(item);
  }
 }
```

Context Menu

A context menu is a floating list of menu items that appears when the user performs a long-press on a View.
Step 1 - Creating a menu XML

```
<?xml version="1.0" encoding="utf-8"?>
  <menu xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:id="@+id/new_game"
   android:icon="@drawable/ic_new_game"
  android:title="@string/new_game" />
  <item android:id="@+id/quit"
  android:icon="@drawable/ic_quit"
  android:title="@string/quit" />
 </menu>
```

Step 2 – Showing the menu (implemented in Activity class)

```
@Override
  public void onCreateContextMenu(ContextMenu menu, View v,
   ContextMenuInfo menuInfo) {
  super.onCreateContextMenu(menu, v, menuInfo);
  MenuInflater inflater = getMenuInflater();
  inflater.inflate(R.menu.context_menu, menu);
 }
```

Step 3 – Listening to user choice (implemented in Activity class)

```
@Override
  public boolean onContextItemSelected(MenuItem item) {
  AdapterContextMenuInfo info = (AdapterContextMenuInfo) item.getMenuInfo();
  switch (item.getItemId()) {
  case R.id.edit:
  editNote(info.id);
```

```
editNote(info.id);
return true;
case R.id.delete:
 deleteNote(info.id);
 return true;
 default:
return super.onContextItemSelected(item);
}
}
```

Step 4 – Attaching the context menu to a view (implemented in Activity class)

```
registerForContextMenu(findViewById(R.id.Button01));
```

Submenu

A submenu is a floating list of menu items that the user opens by pressing a menu item in the Options Menu or a context menu.

```
<?xml version="1.0" encoding="utf-8"?>
  <menu xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:id="@+id/file"
   android:icon="@drawable/file"
   android:title="@string/file" >
  <!-- "file" submenu -->
   <menu>
  <item android:id="@+id/new"
  android:title="@string/new" />
  <item android:id="@+id/open"
  android:title="@string/open" />
  </menu>
  </item>
 </menu>
```

Alerts/Dialogs

Toast
A toast notification is a message that pops up on the surface of the window.

```
Toast waitToast = Toast.makeText(getApplicationContext(), "Please wait...", Toast.LENGTH_LONG);
  waitToast.setGravity(Gravity.TOP, 0, 0);
 waitToast.show();
```

AlertDialog

```
AlertDialog.Builder builder = new AlertDialog.Builder(context);
  builder.setMessage("Are you sure you want to exit?")
  .setCancelable(false)
   .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
  public void onClick(DialogInterface dialog, int id) {
  MyActivity.this.finish();
   }
   })
  .setNegativeButton("No", new DialogInterface.OnClickListener() {
  public void onClick(DialogInterface dialog, int id) {
  dialog.cancel();
   }
  });
  AlertDialog alert = builder.create();
 alert.show();
```

Status-Bar Notifications

One way to notify an Android user is by displaying notifications in the status bar. You can show a message, play a sound, add an icon and more.

Creating a status bar notification

```
NotificationManager mNotificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
  Notification notification = new Notification(R.drawable.icon, "Notification Test", System.currentTimeMillis());
  Context context = getApplicationContext();
  CharSequence contentTitle = "My notification Title";
  CharSequence contentText = "This is the message";
  Intent notificationIntent = new Intent(NotificationTest.this, NotificationTest.class);
```

```
    //options
    notification.sound = Uri.withAppendedPath(Audio.Media.INTERNAL_CONTENT_URI, "4"); //sound
    notification.vibrate = new long[]{0,100,200,300}; //vibrate
    //auto cancel after select
    notification.flags |= Notification.FLAG_AUTO_CANCEL;

    PendingIntent contentIntent = PendingIntent.getActivity(NotificationTest.this, 0, notificationIntent, 0);
    notification.setLatestEventInfo(context, contentTitle, contentText, contentIntent);
   mNotificationManager.notify(1, notification);
```

Resource Images

Resource images are placed under <project_dir>/res/drawable

Using a resource image in an image view - XML example

```
((ImageView)findViewById(R.id.myImageView)).setImageResource(R.drawable.my_image_name);
```

Turning the resource image to a Bitmap object

```
Bitmap bitmap = BitmapFactory.decodeResource(view.getResources(),R.drawable.my_image_name);
```

Drawing the bitmap on a canvas object

```
canvas.drawBitmap(bitmap, x, y, null);
```

Creating Links Using Linkfy

Linkfy is a class that lets you create links from TextViews.
You can create links not just to web sites, but to also map addresses, emails and even phone numbers.

Creating web links

```
>TextView myWebSite = (TextView) findViewById(R.id.my_web_site);
myWebSite.setText(http://http://www.dzone.com/)
Linkify.addLinks(myWebSite , Linkify.WEB_URLS);
```

You can also create links to phone numbers, map locations or email addresses.

Using a regular expressionfor filtering strings

```
TextView myCustomLink = new TextView(this);
  Pattern pattern = Pattern.compile("[a-zA-Z]+&");
  myCustomLink.setText("press Linkify& or on Android& to search it on google");
  Linkify.addLinks(myCustomLink,pattern, "http://www.google.ie/search?q=");
  mainLayout.addView(myCustomLink);
```

SECTION 6

System APIs

Using Multithreading for Background Jobs

Regular Thread

```
(new Thread(new Runnable() {

  @Override
 public void run() {
 doLongOperation();
 }
})).start();
```

Thread With Handler for UI operations

A new thread cannot update the user interface, so you need to use a handler. The Handler is the middleman between a new thread and the UI message queue.

```
final Handler myHandler = new Handler(){
  @Override
  public void handleMessage(Message msg) {
  updateUI((String)msg.obj);
```

```
  }

  };

  (new Thread(new Runnable() {

  @Override
  public void run() {
 Message msg = myHandler.obtainMessage();
msg.obj = doLongOperation();
  myHandler.sendMessage(msg);
  }
 })).start();
```

Using AsyncTask

An AsyncTask is a thread that can handle user interface operations.

```
class MyAsyncTask extends AsyncTask<Integer, String, Long> {
@Override
  protected Long doInBackground(Integer... params) {
long start = System.currentTimeMillis();
  for (Integer integer : params) {
  publishProgress("start processing "+integer);
  doLongOperation();
  publishProgress("done processing "+integer);
  }
return start - System.currentTimeMillis();
 }
@Override
  protected void onProgressUpdate(String... values) {         updateUI(values[0]);  }
@Override
  protected void onPostExecute(Long time) {
  updateUI("Done,it took:"+ time +"millisecondes"); }
@Override
  protected void onPreExecute() {
  updateUI("Starting the process"); }
  }
MyAsyncTask aTask = new MyAsyncTask();
  aTask.execute(1, 2, 3, 4, 5);
```

AsyncTask defines 3 generic types: AsyncTask<{type of the input}, {type of the update unit}, {type of the result}>. You don't have to use all of them – simply use 'Void' for any of them.

Using a Timer to Schedule Jobs

A Timer is a comfortable way to dispatch a thread in the future, be it once or more.

```
TimerTask timerTask = new TimerTask() {
  @Override
 public void run() { doSomething(); }
};
Timer timer = new Timer();
  timer.schedule(timerTask, 2000,2000);
```

Opening a new screen

Each screen is a different Activity class. You have to declare each activity in the manifest file.

Opening a new activity

```
Intent in = new Intent(myactivity.this, MyActivity2.class);
in.putExtra("myKey", "new1"); //passing a parameter
startActivity(in);
```

Receiving a parameter from from the original activity

```
String s= getIntent().getExtras().get("myKey").toString()
```

Going back from the new activity to the original activity

```
finish();
```

Open a new Activity to get theresult

```
Intent in = new Intent(myactivity.this, MyActivity2.class);
startActivityForResult(in, 0);
```

Returning a result to the original activity

```
Intent in = getIntent();
in.putExtra("result", "some parameter");
setResult(RESULT_OK,in);
finish();
```

Getting the result from the new activityafter it finishes

```
@Override
 protected void onActivityResult(int requestCode, int resultCode, Intent data) ...
```

SECTION 7

Network

Opening a browser using intent

```
startActivity(new Intent(Intent.ACTION_VIEW,  Uri.parse("http://www.dzone.com")));
```

Enabling Internet permission in the manifest file

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

Reading HTTP using Apache HttpClient

```
HttpClient client = new DefaultHttpClient();
HttpGet request = new HttpGet();
request.setURI(new URI("http://www.dzone.com/"));
HttpResponse response = client.execute(request);
BufferedReader inBuff = new BufferedReader (new InputStreamReader(response.getEntity().getContent()));
  String line = null;
  while ((line = inBuff.readLine()) != null) {
  doSomethingWithHTML(line);
  }
  inBuff.close();
```

Reading a TCP packet using a Socket

```
Socket requestSocket = new Socket("remote.servername.com", 13);
  InputStreamReader isr = new InputStreamReader(requestSocket.getInputStream(), "ISO-8859-1");
  while ((this.ch = isr.read()) != -1) {
 myStringBuffer.append((char) this.ch);
This may result in following ring formation, where "collin", "owen", and "lisa" are rowkeys.
```

Embedding a browser inside your application

```
WebView  webView = ((WebView)findViewById(R.id.webview));
webView.getSettings().setJavaScriptEnabled(true);
webView.getSettings().setPluginsEnabled(true); //plugins/flash
webView.loadUrl(url);
```

Loading your own html to the embedded browser

```
webView.loadData(htmlString, text/html, utf-8");
```

Using the system email client

```
final Intent emailIntent = new Intent(android.content.Intent.ACTION_SEND);
  emailIntent.setType("plain/html");    //for html messages
  emailIntent.putExtra(android.content.Intent.EXTRA_EMAIL, new String[]{"someine@website.com"});
  emailIntent.putExtra(android.content.Intent.EXTRA_SUBJECT, "email  Subject");
  emailIntent.putExtra(android.content.Intent.EXTRA_TEXT, "<H1>email body</H1>");
 startActivity(Intent.createChooser(emailIntent, "Send mail..."));
```

SECTION 8

Media (Audio, Video)

Using the media player to play local media files

```
MediaPlayer mp = MediaPlayer.create(context, R.raw.sound_file_1);
 mp.start();
```

The media files are placed under res/raw

Playing a media file from a path

```
MediaPlayer mp = new MediaPlayer();
  mp.setDataSource("http://www.somepath.com/file.mp3");
  mp.prepare();
 mp.start();
```

Opening the system media player using intent

```
Intent intent = new Intent(Intent.ACTION_VIEW);
  intent.setDataAndType(Uri.parse( "http://somepath.com/video.mp4" ),"video/mp4");
 startActivity(intent);
```

Displaying video inside your application with VideoView

```
getWindow().setFormat(PixelFormat.TRANSLUCENT);
  VideoView videoHolder = (VideoView) findViewById(R.id.VideoView01);
  videoHolder.setMediaController(new MediaController(this));
  videoHolder.setVideoURI(Uri.parse(path+"/"+fileName));
  videoHolder.requestFocus();
 videoHolder.start();
```

SECTION 9

Storage

Storing in shared preferences

Share preferences are a simple key-value storage mechanism of primitive types and Strings. These values are stored even if the program is terminated.

```
// Restore preferences
SharedPreferences settings =
PreferenceManager.getDefaultSharedPreferences(context)
boolean booleanParam = settings.getBoolean("booleanParam", false);
// Change preferences
  SharedPreferences.Editor editor = settings.edit();
  editor.putBoolean("booleanParam", true);
  editor.commit()
```

Storing internally
Reading

```
int ch;
  StringBuilder strContent = new StringBuilder();
  FileInputStream fis = openFileInput("my_file");
  while ((ch = fis.read()) != -1)
 strContent.append((char) ch);
```

Writing

```
FileOutputStream fop = openFileOutput("my_file", Context.MODE_PRIVATE);
  fop.write("Data to be written".getBytes());
  fop.flush();
 fop.close();
```

Storing on the SD card

```
File path = Environment.getExternalStorageDirectory();
```

SQLite

Android has a built-in SQLite DB that you can use.

Creating a DB table

Creating a DB table

```
public abstract class DBManager extends SQLiteOpenHelper{
public DBManager(Context context,String tableName, String [] columns) {
  super(context, "test.db", null, 1);
  }
  ...
```

Executing SQL code

```
db.execSQL( "create table ..." );
```

Inserting Data

```
getWritableDatabase().insert(tableName, null, values);
```

GettingData

```
getReadableDatabase().query(...);
```

Camera

Permission in manifest file

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

Launching the Camera using an intent

```
private void cameraIntent(){
Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
startActivityForResult(cameraIntent, 1);
}
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
if(data != null && data.getExtras() != null){
Bitmap bitmap = (Bitmap) data.getExtras().get("data");
  }
```

Embedding a camera in your application

Step 1-Preview - extending a surface view

```
class Preview extends SurfaceView implements SurfaceHolder.Callback{
  SurfaceHolder mHolder;
Camera mCamera;
Preview(Context context) {
  super(context);
mHolder = getHolder();
  mHolder.addCallback(this);
  mHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
  }

  public void surfaceCreated(SurfaceHolder holder) {
  mCamera = Camera.open();
  mCamera.setPreviewDisplay(holder);
  }
public void surfaceDestroyed(SurfaceHolder holder) {
  mCamera.stopPreview();
  mCamera.release();
  }

  public void surfaceChanged(SurfaceHolder holder, int format, int w, int h) {
  mCamera.startPreview();
  }
 }
```

Step 2 – Taking pictures

The camera needs to be in preview mode while taking a picture - otherwise the action will fail

```
camera.takePicture(null, null, new PictureCallback() {

  @Override
  public void onPictureTaken(byte[] data, Camera camera) {
  try{
```

```
Bitmap bitmap = BitmapFactory.decodeByteArray(data, 0, data.length);
} finally{
camera.release();
}
}
});
```

Contacts

Permission in manifest

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

Executing the contacts application using intent

```
Intent intent = new Intent(Intent.ACTION_VIEW);
  intent.setData(Uri.parse("content://contacts/people/" ));
 startActivity(intent);
```

Using intent to display a contact with id

```
Intent intent = new Intent(Intent.ACTION_VIEW);
 intent.setData(Uri.parse("content://contacts/people/"+id )); startActivity(intent);
```

Phone calls

Permission in manifest

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

Execute a phone call

```
Intent intent = new Intent(Intent.ACTION_CALL);
  intent.setData(Uri.parse("tel:+436641234567"));
 startActivity(intent);
```

Call a contact

```
Intent intent = new Intent(Intent.ACTION_CALL);
  intent.setData(Uri.parse("content://contacts/people/"+contact_id));
 startActivity(intent);
```

Display the dialer with a number

```
Intent intent = new Intent(Intent.ACTION_DIAL);
  intent.setData(Uri.parse("tel:+436641234567"));
 startActivity(intent);
```

SMS, MMS

Permission in manifest file

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

Open SMS application

```
Intent sendIntent = new Intent(Intent.ACTION_VIEW);
  sendIntent.putExtra("sms_body", "Content of the SMS goes here...");
  sendIntent.setType("vnd.android-dir/mms-sms");
 startActivity(sendIntent);
```

Open MMS application

```
:Intent sendIntent = new Intent(Intent.ACTION_SEND);
sendIntent.putExtra("sms_body", "additional text");
//attaching an image
  File img = new File("/sdcard/a.png");
  sendIntent.putExtra(Intent.EXTRA_STREAM, Uri.fromFile(img));
  sendIntent.setType("image/png");
  startActivity(sendIntent);
```

Send SMS from your code

```
PendingIntent pi = PendingIntent.getActivity(this, 0,
  new Intent(this, this.getClass()), 0)
  SmsManager sms = SmsManager.getDefault();
 sms.sendTextMessage(phoneNumber, null, message, pi, null);
```

Geo-location

Required permission in manifest file

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

Listening to user location updates

Location updates are received from GPS, Network or Passive (other applications' data).

```
LocationManager locationManager =
 (LocationManager) context.getSystemService(LOCATION_SERVICE);
LocationListener myLocationListener = new LocationListener() {
  ...
  };
locationManager.requestLocationUpdates(
   LocationManager.NETWORK_PROVIDER, 0, 0, myLocationListener);
  locationManager.requestLocationUpdates(
   LocationManager.GPS_PROVIDER, 0, 0, myLocationListener);
  locationManager.requestLocationUpdates(
   LocationManager.PASSIVE_PROVIDER, 0, 0, myLocationListener);
```

Fetching Screen Properties

```
Display display = getWindowManager().getDefaultDisplay();
int screenHeight = display.getHeight();
int screenWidth = display.getWidth();
int orientation = getResources().getConfiguration().orientation;
  if (orientation == Configuration.ORIENTATION_PORTRAIT) {
 // portrait
  } else if (orientation == Configuration.ORIENTATION_LANDSCAPE) {
 // Landscape
  }
```