Finding Your Location

The purpose of location-based services is to fi nd the physical location of the device.

Access to the location-based services is handled using the Location Manager system Service. To access the Location Manager, request an instance of the LOCATION_SERVICE using the getSystemService method, as shown in the following snippet:

```
String serviceString = Context.LOCATION_SERVICE;
LocationManager locationManager;
locationManager = (LocationManager)getSystemService(serviceString);
```

Before you can use the Location Manager, you need to add one or more uses-permission tags to your manifest to support access to the LBS hardware.

The following snippet shows the *fi ne* and *coarse* permissions. Of the default providers, the GPS provider requires fi ne permission, while the Network provider requires only coarse. An application that has been granted fi ne permission will have coarse permission granted implicitly.

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

You can fi nd the last location fi x determined by a particular Location Provider using the getLastKnownLocation method, passing in the name of the Location Provider. The following example fi nds the last location fi x taken by the GPS provider:

```
String provider = LocationManager.GPS_PROVIDER;
Location location = locationManager.getLastKnownLocation(provider);
```

Note that getLastKnownLocation *does not ask the Location Provider to update the current position. If the device has not recently updated the current position this value may be be out of date.*

The Location object returned includes all the position information available from the provider that supplied it. This can include latitude, longitude, bearing, altitude, speed, and the time the location fi x was taken. All these properties are available using get methods on the Location object. In some instances, additional details will be included in the extras Bundle.

"Where Am I?" Example

The following example — "Where Am I?" — features a new Activity that fi nds the device's current location using the GPS Location Provider. You will expand on this example throughout the chapter as you learn new geographic functionality.

This example assumes that you have enabled the GPS_PROVIDER *Location Provider using the techniques shown previously in this chapter, or that you're running it on a device that supports GPS and has that hardware enabled.*

1. Create a new WhereAmI project with a WhereAmI Activity. This example uses the GPS provider (either mock or real), so modify the manifest fi le to include the uses-permission tag for ACCESS_FINE_LOCATION.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.paad.whereami">
<application
android:icon="@drawable/icon">
<activity
android:name=".WhereAmI"
android:label="@string/app_name">
<intent-fi lter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-fi lter>
</activity>
</application>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
</manifest>
```

2. Modify the main.xml layout resource to include an android:ID attribute for the TextView control so that you can access it from within the Activity.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fi ll_parent"
android:layout_height="fi ll_parent">
<TextView
android:id="@+id/myLocationText"
android:layout_width="fi ll_parent"
android:layout_height="wrap_content"
android:text="@string/hello"
/>
</LinearLayout>
```

3. Override the onCreate method of the WhereAmI Activity to get a reference to the Location Manager. Call getLastKnownLocation to get the last location fi x value, and pass it in to the updateWithNewLocation method stub.

```
package com.paad.whereami;
import android.app.Activity;
import android.content.Context;
import android.location.Location;
import android.location.LocationManager;
import android.os.Bundle;
import android.widget.TextView;
public class WhereAmI extends Activity {
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main);
LocationManager locationManager;
String context = Context.LOCATION_SERVICE;
locationManager = (LocationManager)getSystemService(context);
String provider = LocationManager.GPS_PROVIDER;
Location location = locationManager.getLastKnownLocation(provider);
updateWithNewLocation(location);
}
private void updateWithNewLocation(Location location) {
}
}
```

4. Fill in the updateWithNewLocation method to display the passed-in Location in the Text View by extracting the latitude and longitude values.

```
private void updateWithNewLocation(Location location) {
String latLongString;
TextView myLocationText;
myLocationText = (TextView)findViewById(R.id.myLocationText);
if (location != null) {
double lat = location.getLatitude();
double lng = location.getLongitude();
} else {
latLongString = "No location found";
}
myLocationText.setText("Your Current Position is:\n" +
latLongString);
        }
```

5. When running, your Activity should look like Figure 7-3.

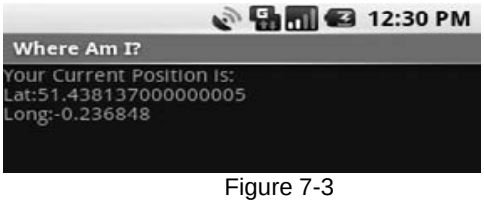

Figure 7-3

Tracking Movement

Most location-sensitive applications will need to be reactive to user movement. Simply polling the Location Manager will not force it to get new updates from the Location Providers.

Use the requestLocationUpdates method to get updates whenever the current location changes, using a LocationListener. Location Listeners also contain hooks for changes in a provider's status and availability.

The requestLocationUpdates method accepts either a specifi c Location Provider name or a set of Criteria to determine the provider to use.

To optimize effi ciency and minimize cost and power use, you can also specify the minimum time and the minimum distance between location change updates.

The following snippet shows the skeleton code for requesting regular updates based on a minimum time and distance.

```
String provider = LocationManager.GPS_PROVIDER;
int t = 5000; // milliseconds
int distance = 5; // meters
LocationListener myLocationListener = new LocationListener() {
public void onLocationChanged(Location location) {
// Update application based on new location.
}
public void onProviderDisabled(String provider){
// Update application if provider disabled.
}
public void onProviderEnabled(String provider){
// Update application if provider enabled.
}
public void onStatusChanged(String provider, int status,
Bundle extras){
// Update application if provider hardware status changed.
}
};

locationManager.requestLocationUpdates(provider, t, distance, myLocationListener);
```

When the minimum time and distance values are exceeded, the attached Location Listener will execute its onLocationChanged event.

You can request multiple location updates pointing to different Location Listeners and using different minimum thresholds. A common design pattern is to create a single listener for your application that broadcasts Intents to notify other components of location changes. This centralizes your listeners and ensures that the Location Provider hardware is used as effi ciently as possible.

To stop location updates, call removeUpdates, as shown below. Pass in the Location Listener instance you no longer want to have triggered.

```
locationManager.removeUpdates(myLocationListener);
```

Most GPS hardware incurs signifi cant power cost. To minimize this, you should disable updates whenever possible in your application, specifi cally when location changes are being used to update an Activity's User Interface. You can improve performance further by extending the minimum time between updates as long as possible.

Updating Your Location in "Where Am I?"

In the following example, "Where Am I?" is enhanced to track your current location by listening for location changes. Updates are restricted to one every 2 seconds, and only when movement of more than 10 meters has been detected.

Rather than explicitly selecting the GPS provider, in this example, you'll create a set of Criteria and let Android choose the best provider available.

1. Start by opening the WhereAmI Activity in the WhereAmI project. Update the onCreate method to fi nd the best Location Provider that features high accuracy and draws as little power as possible.

```
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main);
LocationManager locationManager;
String context = Context.LOCATION_SERVICE;
locationManager = (LocationManager)getSystemService(context);
Criteria criteria = new Criteria();
criteria.setAccuracy(Criteria.ACCURACY_FINE);
criteria.setAltitudeRequired(false);
criteria.setBearingRequired(false);
criteria.setCostAllowed(true);
criteria.setPowerRequirement(Criteria.POWER_LOW);
String provider = locationManager.getBestProvider(criteria, true);
Location location = locationManager.getLastKnownLocation(provider);
updateWithNewLocation(location);}
```

2. Create a new LocationListener instance variable that fi res the existing updateWithNewLocation method whenever a location change is detected.

```
private fi nal LocationListener locationListener = new LocationListener() {
public void onLocationChanged(Location location) {
updateWithNewLocation(location);
}
public void onProviderDisabled(String provider){
updateWithNewLocation(null);
}
public void onProviderEnabled(String provider){ }
public void onStatusChanged(String provider, int status,
Bundle extras){ }
};
```

3. Return to onCreate and execute requestLocationUpdates, passing in the new Location Listener object. It should listen for location changes every 2 seconds but fi re only when it detects movement of more than 10 meters.

```
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main);
LocationManager locationManager;
String context = Context.LOCATION_SERVICE;
locationManager = (LocationManager)getSystemService(context);
Criteria criteria = new Criteria();
criteria.setAccuracy(Criteria.ACCURACY_FINE);
criteria.setAltitudeRequired(false);
criteria.setBearingRequired(false);
criteria.setCostAllowed(true);
criteria.setPowerRequirement(Criteria.POWER_LOW);
String provider = locationManager.getBestProvider(criteria, true);
Location location = locationManager.getLastKnownLocation(provider);
updateWithNewLocation(location);
locationManager.requestLocationUpdates(provider, 2000, 10,
locationListener);
}
```

If you run the application and start changing the device location, you will see the Text View update accordingly.