一、LocationManager

LocationMangager，位置管理器。要想操作定位相关设备，必须先定义个LocationManager。我们可以通过如下代码创建LocationManger对象。

[java] view plaincopyprint?

LocationManger locationManager=(LocationManager)this.getSystemService(Context.LOCATION\_SERVICE);

二、LocationListener

LocationListener，位置监听，监听位置变化，监听设备开关与状态。

[java] view plaincopyprint?

private LocationListener locationListener=new LocationListener() {

        /\*\*

         \* 位置信息变化时触发

         \*/

        public void onLocationChanged(Location location) {

            updateView(location);

            Log.i(TAG, "时间："+location.getTime());

            Log.i(TAG, "经度："+location.getLongitude());

            Log.i(TAG, "纬度："+location.getLatitude());

            Log.i(TAG, "海拔："+location.getAltitude());

        }

        /\*\*

         \* GPS状态变化时触发

         \*/

        public void onStatusChanged(String provider, int status, Bundle extras) {

            switch (status) {

            //GPS状态为可见时

            case LocationProvider.AVAILABLE:

                Log.i(TAG, "当前GPS状态为可见状态");

                break;

            //GPS状态为服务区外时

            case LocationProvider.OUT\_OF\_SERVICE:

                Log.i(TAG, "当前GPS状态为服务区外状态");

                break;

            //GPS状态为暂停服务时

            case LocationProvider.TEMPORARILY\_UNAVAILABLE:

                Log.i(TAG, "当前GPS状态为暂停服务状态");

                break;

            }

        }

        /\*\*

         \* GPS开启时触发

         \*/

        public void onProviderEnabled(String provider) {

            Location location=lm.getLastKnownLocation(provider);

            updateView(location);

        }

        /\*\*

         \* GPS禁用时触发

         \*/

        public void onProviderDisabled(String provider) {

            updateView(null);

        }

    };

三、Location

Location，位置信息，通过Location可以获取时间、经纬度、海拔等位置信息。上面采用locationListener里面的onLocationChanged()来获取location，下面讲述如何主动获取location。

[java]

Location location=locationManager.getLastKnownLocation(LocationManager.GPS\_PROVIDER);   system.out.println("时间："+location.getTime());   system.out.println("经度："+location.getLongitude());

注意：Location location=new Location(LocationManager.GPS\_PROVIDER)方式获取的location的各个参数值都是为0。

四、GpsStatus.Listener

GpsStatus.Listener ，GPS状态监听，包括GPS启动、停止、第一次定位、卫星变化等事件。

[java]

//状态监听

    GpsStatus.Listener listener = new GpsStatus.Listener() {

        public void onGpsStatusChanged(int event) {

            switch (event) {

            //第一次定位

            case GpsStatus.GPS\_EVENT\_FIRST\_FIX:

                Log.i(TAG, "第一次定位");

                break;

            //卫星状态改变

            case GpsStatus.GPS\_EVENT\_SATELLITE\_STATUS:

                Log.i(TAG, "卫星状态改变");

                //获取当前状态

                GpsStatus gpsStatus=lm.getGpsStatus(null);

                //获取卫星颗数的默认最大值

                int maxSatellites = gpsStatus.getMaxSatellites();

                //创建一个迭代器保存所有卫星

                Iterator<GpsSatellite> iters = gpsStatus.getSatellites().iterator();

                int count = 0;

                while (iters.hasNext() && count <= maxSatellites) {

                    GpsSatellite s = iters.next();

                    count++;

                }

                System.out.println("搜索到："+count+"颗卫星");

                break;

            //定位启动

            case GpsStatus.GPS\_EVENT\_STARTED:

                Log.i(TAG, "定位启动");

                break;

            //定位结束

            case GpsStatus.GPS\_EVENT\_STOPPED:

                Log.i(TAG, "定位结束");

                break;

            }

        };

    };

//绑定监听状态

lm.addGpsStatusListener(listener);

五、GpsStatus

GpsStatus，GPS状态信息，上面在卫星状态变化时，我们就用到了GpsStatus。

[java]

//实例化

GpsStatus gpsStatus = locationManager.getGpsStatus(null); // 获取当前状态

//获取默认最大卫星数

int maxSatellites = gpsStatus.getMaxSatellites();

//获取第一次定位时间（启动到第一次定位）

int costTime=gpsStatus.getTimeToFirstFix();

//获取卫星

Iterable<GpsSatellite> iterable=gpsStatus.getSatellites();

//一般再次转换成Iterator

Iterator<GpsSatellite> itrator=iterable.iterator();

六、GpsSatellite

GpsSatellite，定位卫星，包含卫星的方位、高度、伪随机噪声码、信噪比等信息。

[java]

//获取卫星

Iterable<GpsSatellite> iterable=gpsStatus.getSatellites();

//再次转换成Iterator

Iterator<GpsSatellite> itrator=iterable.iterator();

//通过遍历重新整理为ArrayList

ArrayList<GpsSatellite> satelliteList=new ArrayList<GpsSatellite>();

int count=0;

int maxSatellites=gpsStatus.getMaxSatellites();

while (itrator.hasNext() && count <= maxSatellites) {

    GpsSatellite satellite = itrator.next();

    satelliteList.add(satellite);

    count++;

}

System.out.println("总共搜索到"+count+"颗卫星");

//输出卫星信息

for(int i=0;i<satelliteList.size();i++){

    //卫星的方位角，浮点型数据

    System.out.println(satelliteList.get(i).getAzimuth());

    //卫星的高度，浮点型数据

    System.out.println(satelliteList.get(i).getElevation());

    //卫星的伪随机噪声码，整形数据

    System.out.println(satelliteList.get(i).getPrn());

    //卫星的信噪比，浮点型数据

    System.out.println(satelliteList.get(i).getSnr());

    //卫星是否有年历表，布尔型数据

    System.out.println(satelliteList.get(i).hasAlmanac());

    //卫星是否有星历表，布尔型数据

    System.out.println(satelliteList.get(i).hasEphemeris());

    //卫星是否被用于近期的GPS修正计算

    System.out.println(satelliteList.get(i).hasAlmanac());

}

为了便于理解，接下来模拟一个案例，如何在程序代码中使用GPS获取位置信息。

第一步：新建一个Android工程项目，命名为mygps，目录结构如下

第二步：修改main.xml布局文件，修改内容如下：

[[html](http://www.2cto.com/kf/qianduan/css/)] view plaincopyprint?

<?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">

    <EditText android:layout\_width="fill\_parent"

        android:layout\_height="wrap\_content"

        android:cursorVisible="false"

        android:editable="false"

        android:id="@+id/editText"/>

</LinearLayout>

第三步：实用Adnroid平台的GPS设备，需要添加上权限

[html]

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION"/>

 <uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION"/>

第四步：修改核心[组件](http://www.2cto.com/kf/all/zujian/)activity，修改内容如下

[java] view plaincopyprint?

package jason.gprs;

import java.util.Iterator;

import android.app.Activity;

import android.content.Context;

import android.content.Intent;

import android.location.Criteria;

import android.location.GpsSatellite;

import android.location.GpsStatus;

import android.location.Location;

import android.location.LocationListener;

import android.location.LocationManager;

import android.location.LocationProvider;

import android.os.Bundle;

import android.provider.Settings;

import android.util.Log;

import android.widget.EditText;

import android.widget.Toast;

public class MainActivity extends Activity {

    private EditText editText;

    private LocationManager lm;

    private static final String TAG = "GpsActivity";

    @Override

    protected void onDestroy() {

        // TODO Auto-generated method stub

        super.onDestroy();

        lm.removeUpdates(locationListener);

    }

    @Override

    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

        editText = (EditText) findViewById(R.id.editText);

        lm = (LocationManager) getSystemService(Context.LOCATION\_SERVICE);

        // 判断GPS是否正常启动

        if (!lm.isProviderEnabled(LocationManager.GPS\_PROVIDER)) {

            Toast.makeText(this, "请开启GPS导航...", Toast.LENGTH\_SHORT).show();

            // 返回开启GPS导航设置界面

            Intent intent = new Intent(Settings.ACTION\_LOCATION\_SOURCE\_SETTINGS);

            startActivityForResult(intent, 0);

            return;

        }

        // 为获取地理位置信息时设置查询条件

        String bestProvider = lm.getBestProvider(getCriteria(), true);

        // 获取位置信息

        // 如果不设置查询要求，getLastKnownLocation方法传人的参数为LocationManager.GPS\_PROVIDER

        Location location = lm.getLastKnownLocation(bestProvider);

        updateView(location);

        // 监听状态

        lm.addGpsStatusListener(listener);

        // 绑定监听，有4个参数

        // 参数1，设备：有GPS\_PROVIDER和NETWORK\_PROVIDER两种

        // 参数2，位置信息更新周期，单位毫秒

        // 参数3，位置变化最小距离：当位置距离变化超过此值时，将更新位置信息

        // 参数4，监听

        // 备注：参数2和3，如果参数3不为0，则以参数3为准；参数3为0，则通过时间来定时更新；两者为0，则随时刷新

        // 1秒更新一次，或最小位移变化超过1米更新一次；

        // 注意：此处更新准确度非常低，推荐在service里面启动一个Thread，在run中sleep(10000);然后执行handler.sendMessage(),更新位置

        lm.requestLocationUpdates(LocationManager.GPS\_PROVIDER, 1000, 1, locationListener);

    }

    // 位置监听

    private LocationListener locationListener = new LocationListener() {

        /\*\*

         \* 位置信息变化时触发

         \*/

        public void onLocationChanged(Location location) {

            updateView(location);

            Log.i(TAG, "时间：" + location.getTime());

            Log.i(TAG, "经度：" + location.getLongitude());

            Log.i(TAG, "纬度：" + location.getLatitude());

            Log.i(TAG, "海拔：" + location.getAltitude());

        }

        /\*\*

         \* GPS状态变化时触发

         \*/

        public void onStatusChanged(String provider, int status, Bundle extras) {

            switch (status) {

            // GPS状态为可见时

            case LocationProvider.AVAILABLE:

                Log.i(TAG, "当前GPS状态为可见状态");

                break;

            // GPS状态为服务区外时

            case LocationProvider.OUT\_OF\_SERVICE:

                Log.i(TAG, "当前GPS状态为服务区外状态");

                break;

            // GPS状态为暂停服务时

            case LocationProvider.TEMPORARILY\_UNAVAILABLE:

                Log.i(TAG, "当前GPS状态为暂停服务状态");

                break;

            }

        }

        /\*\*

         \* GPS开启时触发

         \*/

        public void onProviderEnabled(String provider) {

            Location location = lm.getLastKnownLocation(provider);

            updateView(location);

        }

        /\*\*

         \* GPS禁用时触发

         \*/

        public void onProviderDisabled(String provider) {

            updateView(null);

        }

    };

    // 状态监听

    GpsStatus.Listener listener = new GpsStatus.Listener() {

        public void onGpsStatusChanged(int event) {

            switch (event) {

            // 第一次定位

            case GpsStatus.GPS\_EVENT\_FIRST\_FIX:

                Log.i(TAG, "第一次定位");

                break;

            // 卫星状态改变

            case GpsStatus.GPS\_EVENT\_SATELLITE\_STATUS:

                Log.i(TAG, "卫星状态改变");

                // 获取当前状态

                GpsStatus gpsStatus = lm.getGpsStatus(null);

                // 获取卫星颗数的默认最大值

                int maxSatellites = gpsStatus.getMaxSatellites();

                // 创建一个迭代器保存所有卫星

                Iterator<GpsSatellite> iters = gpsStatus.getSatellites()

                        .iterator();

                int count = 0;

                while (iters.hasNext() && count <= maxSatellites) {

                    GpsSatellite s = iters.next();

                    count++;

                }

                System.out.println("搜索到：" + count + "颗卫星");

                break;

            // 定位启动

            case GpsStatus.GPS\_EVENT\_STARTED:

                Log.i(TAG, "定位启动");

                break;

            // 定位结束

            case GpsStatus.GPS\_EVENT\_STOPPED:

                Log.i(TAG, "定位结束");

                break;

            }

        };

    };

    /\*\*

     \* 实时更新文本内容

     \*

     \* @param location

     \*/

    private void updateView(Location location) {

        if (location != null) {

            editText.setText("设备位置信息\n\n经度：");

            editText.append(String.valueOf(location.getLongitude()));

            editText.append("\n纬度：");

            editText.append(String.valueOf(location.getLatitude()));

        } else {

            // 清空EditText对象

            editText.getEditableText().clear();

        }

    }

    /\*\*

     \* 返回查询条件

     \*

     \* @return

     \*/

    private Criteria getCriteria() {

        Criteria criteria = new Criteria();

        // 设置定位精确度 Criteria.ACCURACY\_COARSE比较粗略，Criteria.ACCURACY\_FINE则比较精细

        criteria.setAccuracy(Criteria.ACCURACY\_FINE);

        // 设置是否要求速度

        criteria.setSpeedRequired(false);

        // 设置是否允许运营商收费

        criteria.setCostAllowed(false);

        // 设置是否需要方位信息

        criteria.setBearingRequired(false);

        // 设置是否需要海拔信息

        criteria.setAltitudeRequired(false);

        // 设置对电源的需求

        criteria.setPowerRequirement(Criteria.POWER\_LOW);

        return criteria;

    }

}

第五步：运行效果