

第 11 章 Android 特色开发 ,

基于位置的服务

现在你已经学会了非常多的 Android 技能,并且通过这些技能你完全可以编写出相当不错的应用程序了。不过从本章开始,我们将要学习一些全新的 Android 技术,这些技术有别于传统的 PC 或 Web 领域的应用技术,是只有在移动设备上才能实现的。

说到只有在移动设备上才能实现的技术,很容易就让人联想到基于位置的服务(Location Based Service)。由于移动设备相比于电脑可以随身携带,我们通过地理定位的技术就可以随时得知自己所在的位置,从而围绕这一点开发出很多有意思的应用。本章中我们就将针对这一点展开进行讨论,学习一下基于位置的服务究竟是如何实现的。

11.1 基于位置的服务简介

基于位置的服务简称 LBS,这个技术随着移动互联网的兴起,在最近的几年里十分火爆。其实它本身并不是什么时髦的技术,主要的工作原理就是利用无线电通讯网络或 GPS 等定位方式来确定出移动设备所在的位置,而这种定位技术早在很多年前就已经出现了。

那为什么 LBS 技术直到最近几年才开始流行呢?这主要是因为,在过去移动设备的功能极其有限,即使定位到了设备所在的位置,也就仅仅是定位到了而已,我们并不能在位置的基础上进行一些其他的操作。而现在就大大不同了,有了 Android 系统作为载体,我们可以利用定位出的位置进行许多丰富多彩的操作。比如说天气预报程序可以根据用户所在的位置自动选择城市,发微博的时候我们可以向朋友们晒一下自己在哪里,不认识路的时候随时打开地图就可以查询路线,等等等等。

介绍了这么多,相信你早已经按耐不住了,那么就让我们马上开始本章的学习之旅吧。

11.2 找到自己的位置

归根结底,其实基于位置的服务所围绕的核心就是要确定出自己所在的位置,这在 Android 中并不困难,主要借助 LocationManager 这个类就可以实现了。下面我们首先学习一下 LocationManager 的基本用法,然后再通过一个例子来尝试获取一下自己当前的位置。

另外需要注意，本章中所写的代码建议你都在手机上运行，DDMS 虽然也提供了在模拟器中模拟地理位置的功能，但在手机上得到真实的位置数据，你的感受会更加深刻。

11.2.1 LocationManager 的基本用法

毫无疑问，要想使用 LocationManager 就必须要先获取到它的实例，我们可以调用 Context 的 getSystemService() 方法获取到。getSystemService() 方法接收一个字符串参数用于确定获取系统的哪个服务，这里传入 Context.LOCATION_SERVICE 即可。因此，获取 LocationManager 的实例就可以写成：

```
LocationManager locationManager = (LocationManager)
```

```
getSystemService(Context.LOCATION_SERVICE);
```

接着我们需要选择一个位置提供器来确定设备当前的位置。Android 中一般有三种位置提供器可供选择，GPS_PROVIDER、NETWORK_PROVIDER 和 PASSIVE_PROVIDER。其中前两种使用的比较多，分别表示使用 GPS 定位和使用网络定位。这两种定位方式各有特点，GPS 定位的精准度比较高，但是非常耗电，而网络定位的精准度稍差，但耗电量比较少。我们应该根据自己的实际情况来选择使用哪一种位置提供器，当位置精度要求非常高的时候，最好使用 GPS_PROVIDER，而一般情况下，使用 NETWORK_PROVIDER 会更加得划算。

需要注意的是，定位功能必须要由用户主动去启用才行，不然任何应用程序都无法获取到手机当前的位置信息。进入手机的设置→定位服务，其中第一个选项表示允许使用网络的方式来对手机进行定位，第二个选项表示允许使用 GPS 的方式来对手机进行定位，如图 11.1 所示。



图 11.1

你并不需要担心一旦启用了这几个选项后，手机的电量就会直线下滑，这些选项只是表明你已经同意让应用程序来对你的手机进行定位了，但只有当定位操作真正开始的时候才会影响到手机的电量。下面我们就来看一看，如何才能真正地开始定位操作。

将选择好的位置提供者传入到 `getLastKnownLocation()` 方法中，就可以得到一个 `Location` 对象，如下所示：

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

这个 `Location` 对象中包含了经度、纬度、海拔等一系列的位置信息，然后从中取出我们所关心的那部分数据即可。

如果有些时候你想让定位的精度尽量高一些，但又不确定 GPS 定位的功能是否已经启用，这个时候就可以先判断一下有哪些位置提供者可用，如下所示：

```
List<String> providerList = locationManager.getProviders(true);
```

可以看到，`getProviders()` 方法接收一个布尔型参数，传入 `true` 就表示只有启用的位置提供者才会被返回。之后再从 `providerList` 中判断是否包含 GPS 定位的功能就行了。

另外，调用 `getLastKnownLocation()` 方法虽然可以获取到设备当前的位置信息，但是用户是完全有可能带着设备随时移动的，那么我们怎样才能在设备位置发生改变的时候获取到最新的位置信息呢？不用担心，`LocationManager` 还提供了一个 `requestLocationUpdates()` 方法，只要传入一个 `LocationListener` 的实例，并简单配置几个参数就可以实现上述功能了，写法如下：

```
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 5000, 10,
    new LocationListener() {
        @Override
        public void onStatusChanged(String provider, int status, Bundle
extras) {
        }

        @Override
        public void onProviderEnabled(String provider) {
        }

        @Override
        public void onProviderDisabled(String provider) {
        }
    })
```

```
@Override
public void onLocationChanged(Location location) {
}

});
```

这里 `requestLocationUpdates()` 方法接收四个参数，第一个参数是位置提供器的类型，第二个参数是监听位置变化的时间间隔，以毫秒为单位，第三个参数是监听位置变化的距离间隔，以米为单位，第四个参数则是 `LocationListener` 监听器。这样的话，`LocationManager` 每隔 5 秒钟会检测一下位置的变化情况，当移动距离超过 10 米的时候，就会调用 `LocationListener` 的 `onLocationChanged()` 方法，并把新的位置信息作为参数传入。

好了，关于 `LocationManager` 的用法基本就是这么多，下面我们就通过一个例子来尝试一下吧。

11.2.2 确定自己位置的经纬度

通过上一小节的学习，你会发现 `LocationManager` 的用法并不复杂，那么本小节中我们来编写一个可以获取当前位置经纬度信息的程序吧。

新建一个 `LocationTest` 项目，修改 `activity_main.xml` 中的代码，如下所示：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <TextView
        android:id="@+id/position_text_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

布局文件中的内容实在是太简单了，只有一个 `TextView` 控件，用于稍后显示设备位置的经纬度信息。

然后修改 `MainActivity` 中的代码，如下所示：

```
public class MainActivity extends Activity {

    private TextView positionTextView;

    private LocationManager locationManager;

    private String provider;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    positionTextView = (TextView) findViewById(R.id.position_text_view);
    locationManager = (LocationManager) getSystemService(Context.
LOCATION_SERVICE);
    // 获取所有可用的位置提供者
    List<String> providerList = locationManager.getProviders(true);
    if (providerList.contains(LocationManager.GPS_PROVIDER)) {
        provider = LocationManager.GPS_PROVIDER;
    } else if (providerList.contains(LocationManager.NETWORK_PROVIDER)) {
        provider = LocationManager.NETWORK_PROVIDER;
    } else {
        // 当没有可用的位置提供者时，弹出Toast提示用户
        Toast.makeText(this, "No location provider to use",
Toast.LENGTH_SHORT).show();
        return;
    }
    Location location = locationManager.getLastKnownLocation(provider);
    if (location != null) {
        // 显示当前设备的位置信息
        showLocation(location);
    }
    locationManager.requestLocationUpdates(provider, 5000, 1,
locationListener);
}

protected void onDestroy() {
    super.onDestroy();
    if (locationManager != null) {
        // 关闭程序时将监听器移除
        locationManager.removeUpdates(locationListener);
    }
}

LocationListener locationListener = new LocationListener() {
```

```

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

        @Override
        public void onProviderEnabled(String provider) {
        }

        @Override
        public void onProviderDisabled(String provider) {
        }

        @Override
        public void onLocationChanged(Location location) {
            // 更新当前设备的位置信息
            showLocation(location);
        }
    };

    private void showLocation(Location location) {
        String currentPosition = "latitude is " + location.getLatitude() + "\n"
+ "longitude is " + location.getLongitude();
        positionTextView.setText(currentPosition);
    }
}

```

这里并没有什么复杂的逻辑，基本全是我们在上一小节中学到的知识。在 `onCreate()` 方法中首先是获取到了 `LocationManager` 的实例，然后调用 `getProviders()` 方法去得到所有可用的位置提供者，接下来再调用 `getLastKnownLocation()` 方法就可以获取到记录当前位置信息的 `Location` 对象了，这里我们将 `Location` 对象传入到 `showLocation()` 方法中，经度和纬度的值就会显示到 `TextView` 上了。然后为了要能监测到位置信息的变化，下面又调用了 `requestLocationUpdates()` 方法来添加一个位置监听器，设置时间间隔是 5 秒，距离间隔是 1 米，并在 `onLocationChanged()` 方法中时时更新 `TextView` 上显示的经纬度信息。最后当程序关闭时，我们还需要调用 `removeUpdates()` 方法来将位置监听器移除，以保证不会继续耗费手机的电量。

另外，获取设备当前的位置信息也是要声明权限的，因此还需要修改 `AndroidManifest.xml` 中的代码，如下所示：

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.locationtest"
    android:versionCode="1"
    android:versionName="1.0" >
    .....
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    .....
</manifest>
```

现在运行一下程序，就可以看到手机当前位置的经纬度信息了，如图 11.2 所示。

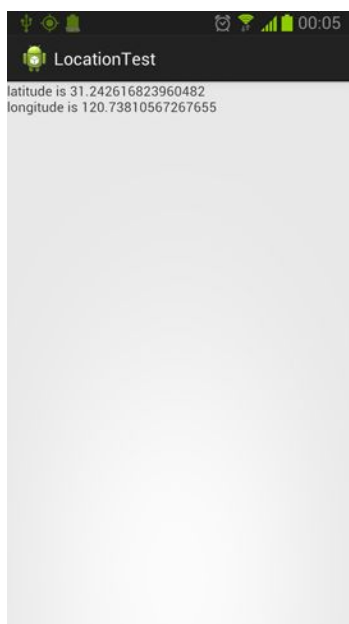


图 11.2

之后如果你拿着手机随处移动，就可以看到界面上的经纬度信息是会变化的。由此证实，我们的程序确实已经在正常工作了。

11.3 反向地理编码，看得懂的位置信息

话说回来，刚才我们虽然成功获取到了设备当前位置的经纬度信息，但遗憾的是，这种经纬值一般人是根本看不懂的，相信谁也无法立刻答出南纬 25 度、东经 148 度是什么地方吧？为了能够更加直观地阅读位置信息，本节中我们就来学习一下，如何通过反向地理编码，将经纬值转换成看得懂的位置信息。

11.3.1 Geocoding API 的用法

其实 Android 本身就提供了地理编码的 API，主要是使用 `Geocoder` 这个类来实现的。它可以非常简单地完成正向和反向的地理编码功能，从而轻松地将一个经纬值转换成看得懂的位置信息。

不过，非常遗憾的是，`Geocoder` 长期存在着一些较为严重的 bug，在反向地理编码的时候会有有一定的概率不能解析出位置的信息，这样就无法保证位置解析的稳定性，因此我们不得不去寻找 `Geocoder` 的替代方案。

还算比较幸运，谷歌又提供了一套 `Geocoding API`，使用它的话也可以完成反向地理编码的工作，只不过它的用法稍微复杂了一些，但稳定性要比 `Geocoder` 强得多。本小节中我们只是学习一下 `Geocoding API` 的简单用法，更详细的用法请参考官方文档：<https://developers.google.com/maps/documentation/geocoding/>。

`Geocoding API` 的工作原理并不神秘，其实就是利用了我们上一章中学习的 HTTP 协议。在手机端我们可以向谷歌的服务器发起一条 HTTP 请求，并将经纬度的值作为参数一同传递过去，然后服务器会帮我们将这个经纬值转换成看得懂的位置信息，再将这些信息返回给手机端，最后手机端去解析服务器返回的信息，并进行处理就可以了。

`Geocoding API` 中规定了很多接口，其中反向地理编码的接口如下：

```
http://maps.googleapis.com/maps/api/geocode/json?latlng=40.714224,-73.961452&sensor=true_or_false
```

我们来仔细看下这个接口的定义，其中 `http://maps.googleapis.com/maps/api/geocode/` 是固定的，表示接口的连接地址。`json` 表示希望服务器能够返回 JSON 格式的数据，这里也可以指定成 `xml`。`latlng=40.714224,-73.96145` 表示传递给服务器去解码的经纬值是北纬 40.714224 度，西经 73.96145 度。`sensor=true_or_false` 表示这条请求是否来自于某个设备的位置传感器，通常指定成 `false` 即可。

如果发送 `http://maps.googleapis.com/maps/api/geocode/json?latlng=40.714224,-73.96145&sensor=false` 这样一条请求给服务器，我们将会得到一段非常长的 JSON 格式的数据，其中会包括如下部分内容：

```
"formatted_address" : "277 Bedford Avenue, 布鲁克林纽约州 11211美国"
```

从这段内容中我们就可以看出北纬 40.714224 度，西经 73.96145 度对应的地理位置是在哪里了。如果你想查看服务器返回的完整数据，在浏览器中访问上面的网址即可。

这样的话，使用 `Geocoding API` 进行反向地理编码的工作原理你就已经搞清楚了，那么难点其实就在于如何从服务器返回的数据中解析出我们想要的那部分信息了。而 JSON 格式数据的解析方式我们早在上一章中就牢牢地掌握了，因此我相信这个问题一定是难不倒你的。下面我们就来完善一下 `LocationTest` 这个程序，给它加入反向地理编码的功能吧。

11.3.2 对经纬度进行解析

使用 Geocoding API 进行反向地理编码的流程相信你已经很清楚了，我们先要发送一个 HTTP 请求给谷歌的服务器，然后再对返回的 JSON 数据进行解析。发送 HTTP 请求的方式我们准备使用 HttpClient，解析 JSON 数据的方式使用 JSONObject。修改 MainActivity 中的代码，如下所示：

```
public class MainActivity extends Activity {

    public static final int SHOW_LOCATION = 0;
    .....

    private void showLocation(final Location location) {
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    // 组装反向地理编码的接口地址
                    StringBuilder url = new StringBuilder();

                    url.append("http://maps.googleapis.com/maps/api/geocode/json?latlng=");
                    url.append(location.getLatitude()).append(",");
                    url.append(location.getLongitude());
                    url.append("&sensor=false");
                    HttpClient httpClient = new DefaultHttpClient();
                    HttpGet httpGet = new HttpGet(url.toString());
                    // 在请求消息头中指定语言，保证服务器会返回中文数据
                    httpGet.addHeader("Accept-Language", "zh-CN");
                    HttpResponse httpResponse = httpClient.execute(httpGet);
                    if (httpResponse.getStatusLine().getStatusCode() == 200) {
                        HttpEntity entity = httpResponse.getEntity();
                        String response = EntityUtils.toString(entity,
                            "utf-8");

                        JSONObject jsonObject = new JSONObject(response);
                        // 获取results节点下的位置信息
                        JSONArray resultArray = jsonObject.getJSONArray
                            ("results");

                        if (resultArray.length() > 0) {
                            JSONObject subObject = resultArray.
                                getJSONObject(0);

                                // 取出格式化后的位置信息
```

```

        String address = subObject.getString
("formatted_address");

        Message message = new Message();
        message.what = SHOW_LOCATION;
        message.obj = address;
        handler.sendMessage(message);
    }
}
} catch (Exception e) {
    e.printStackTrace();
}
}
}).start();
}

private Handler handler = new Handler() {
    public void handleMessage(Message msg) {
        switch (msg.what) {
            case SHOW_LOCATION:
                String currentPosition = (String) msg.obj;
                positionTextView.setText(currentPosition);
                break;
            default:
                break;
        }
    }
};
}

```

观察 `showLocation()` 方法，由于我们要在这里发起网络请求，因此必须开启一个子线程。在子线程中首先是通过 `StringBuilder` 组装了一个反向地理编码接口地址的字符串，然后使用 `HttpClient` 去请求这个地址就好了。注意在 `HttpGet` 中我们还添加了一个消息头，消息头中将语言类型指定为简体中文，不然服务器会默认返回英文的位置信息。

接下来就是对服务器返回的 JSON 数据进行解析了。由于一个经纬度的值有可能包含了好几条街道，因此服务器通常会返回一组位置信息，这些信息都是存放在 `results` 结点下的。在得到了这些位置信息后只需要取其中的第一条就可以了，通常这也是最接近我们位置的那一条。之后就可以从 `formatted_address` 结点中取出格式化后的位置信息了，这种位置信息你就完全可以看得懂了。

不过别忘了，目前我们还是在子线程当中的，因此在这里无法直接将得到的位置信息显示到 `TextView` 上。但这个问题也一定难不倒你了，使用异步消息处理机制就可以轻松解决，相信不需要我再进行多余的解释了吧。

由于这里我们使用到了网络功能，因此还需要在 `AndroidManifest.xml` 中添加权限声明，如下所示：

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.locationtest"
    android:versionCode="1"
    android:versionName="1.0" >
    .....
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />
    .....
</manifest>
```

好了，现在可以重新运行一下程序了，结果如图 11.3 所示。

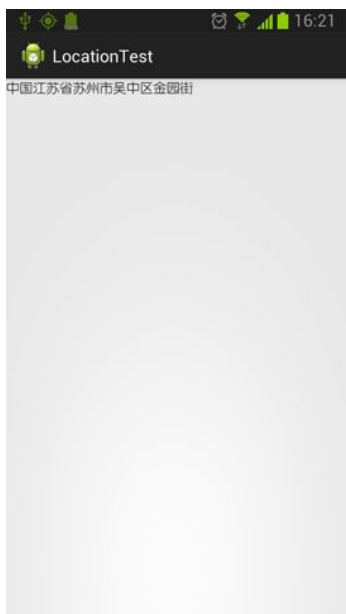


图 11.3

可以看到，手机当前的位置信息已经成功显示出来了！如果你带着手机移动了较远的距离，界面上显示的位置也会跟着一起变化的。

当然，在这个例子中我们只是对服务器返回的 JSON 数据进行了最简单的解析，位置信

息是作为整体取出的，其实你还可以进行更精确的解析，将国家名、城市名、街道名、甚至邮政编码等作为独立的信息取出，更加有趣的功能就等着你自己去进行研究了。

经验值：+30000 目前经验值：232905

级别：头领鸟

获赠宝物：拜访三尺神尊者。获得三尺神尊者馈赠的尊贵宝物，全新恒星级阿尔法三界定位器（TPS）一部、全新夔龙皮 Android 战袍一套、行星级秋风之消亡战斧一把。任何神都有分身，但要说哪位神的分身最多，非三尺神尊者莫属。三尺神尊者时刻位于三界任何高级生灵的头顶上空三尺的位置，因此，他确切地知道每一个生灵在任何时刻的时空位置，你的所作所为他也全部都看在眼里。记者采访时间为什么低等生灵的头顶上就没有呢，三尺神诚恳地回答：“没有办法，确实是忙不过来了”。三尺神是三界中文治武功排名前十的。几乎没有人能够战胜他，他的威震三界的随身武器是超光锥三界之消亡，这件武器在整个宇宙中是独一无二的，其至高无上令神人魔共惧的强大之处在于它不仅可以毁灭当下的存在，而且可以毁灭过去和未来的存在。换句话说，三界中的任何高等生灵犯下了罪孽，无论是位于现在、过去、还是未来，都逃不出三尺神尊者的惩罚，毁灭是注定的。你可以拜访三尺神，但你永远无法拜别三尺神，因为他始终在你头顶上空三尺。当你修行的级别不够时是无法看到三尺神的，三尺神只有被你自身所发出的光芒照到时才能被你看到。当你第一次看到三尺神时，我敢说，整个世界在你眼前都不一样了。我定了定神。继续前进。

11.4 使用百度地图

现在手机地图的应用真的可以算得上是非常广泛了，和 PC 上的地图相比，手机地图能够随时随地进行查看，并且轻松构建出行路路线，使用起来明显更加地方便。但是你有没有想过，其实我们在自己的应用程序里也是可以加入地图功能的。

在手机地图领域做得最好的就当数谷歌地图和百度地图了，并且这两种地图都提供了丰富的 API，使得任何开发者都可以轻松地将地图功能引入到自己的应用程序当中。只不过谷歌地图在 2013 年 3 月的时候全面停用了第一版的 API Key，而第二版的 API Key 在中国使用的时候又有诸多限制，因此这里我们就不准备使用谷歌地图了。相比之下，百度地图的使用就没有任何限制，而且用法也非常方便，那么它自然就成为我们本节的主题了。

11.4.1 申请 API Key

要想在自己的应用程序里加入百度地图的功能，首先必须申请一个 API Key。你得拥有一个百度账号才能进行申请，我相信大多数人早就已经拥有了吧？如果你还没有的话，赶快去注册一个吧。

下面我们需要注册成为一名百度开发者。登录你的百度账号，并打开 <http://developer.baidu.com/user/reg> 这个网址，在这里填写一些注册信息即可，如图 11.4 所示。

* 类型：☒ 个人 ☐ 公司

* 开发者来源：

* 开发者姓名：

* 开发者简介：

* Email地址： [修改](#)

* 手机号： [重新发送 \(5秒\)](#)

* 验证码：

开发者官方网站：

品牌LOGO：

☒ 我已阅读并同意[百度开放云平台注册协议](#)

[提交](#)

图 11.4

只需填写有“*”号的那部分内容就足够了，接下来点击提交，会显示如图 11.5 所示的界面。



图 11.5

到此一切顺利！这样你就已经成为一名百度开发者了。接着点击去创建应用，然后选择我的应用，会看到如图 11.6 所示的界面。



图 11.6

这里会显示所有你申请过的 API Key，由于这是一个刚刚注册的账号，所以目前只有一个系统默认的 API Key。接下来点击创建应用就可以去申请新的 API Key 了，应用名称可以随便填，应用类型选择 for mobile，如图 11.7 所示。



图 11.7

那么剩下一个安全码是什么意思呢？这是我们申请 API Key 所必须填写的一个字段，它的组成方式是数字签名+;+包名。这里数字签名指的是我们打包程序时所用 keystore 的 SHA1 指纹，可以在 Eclipse 中查看到。点击 Eclipse 导航栏的 Window→Preferences→Android→Build，界面如图 11.8 所示。

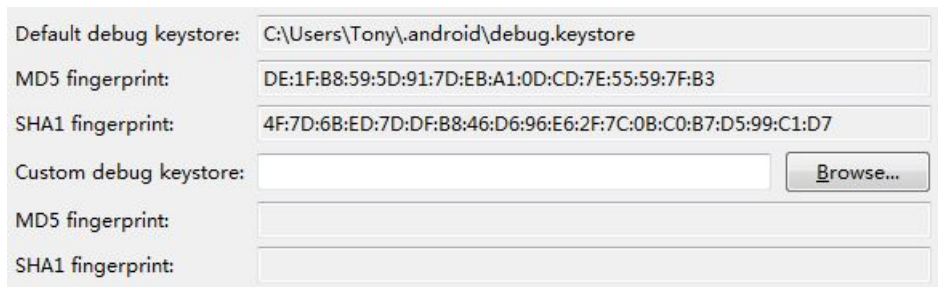


图 11.8

其中，4F:7D:6B:ED:7D:DF:B8:46:D6:96:E6:2F:7C:0B:C0:B7:D5:99:C1:D7 就是我们所需的 SHA1 指纹了，当然你的 Eclipse 中显示的指纹肯定和我是不一样的。另外需要注意，目前我们使用的是 debug.keystore 所生成的指纹，这是 Android 自动生成的一个用于测试的 keystore。而当你的应用程序发布时还需要创建一个正式的 keystore，如果要得到它的指纹，就需要在 cmd 中输入如下命令：

keytool -list -v -keystore <keystore文件名>

然后输入正确的密码就可以了。创建 keystore 的方法我们将在第 15 章中学习。

那么数字签名的值已经得到了，接下来需要连接一个;号，然后加上应用程序的包名。目前我们的应用程序还不存在，但可以先将包名预定下来，比如就叫 com.example.baidumaptest。因此，一个完整的安全码就是：4F:7D:6B:ED:7D:DF:B8:46:D6:96:E6:2F:7C:0B:C0:B7:D5:99:C1:D7; com.example.baidumaptest。我们将这个值填入到图 11.7 的安全码输入框中，然后点击确定。这样的话就已经申请成功了，如图 11.9 所示。

应用Id	应用名称	访问应用(ak)	应用类别	备注信息(双击更改)	应用配置
系统默认AK	系统默认AK	2a9*****45a	移动端	当前系统所用AK	当前系统所用AK
1873934	百度地图测试	SHVPoTtlpzfonPD3HCkc5sIt	移动端		设置 删除

[创建应用\(2/20\)](#) [查看回收站](#)

注：若您申请的密钥用于Android SDK或iOS SDK，创建密钥后，请进行相关的配置工作(查看配置方法：[Android](#) [iOS](#))。

图 11.9

其中，SHVPoTtlpzfonPD3HCkc5sIt 就是申请到的 API Key，有了它就可以进行后续的地图开发工作了，那么我们马上开始吧。

11.4.2 让地图显示出来

现在正是趁热打铁的好时机，新建一个 BaiduMapTest 项目，并将包命名为 com.example.

baidumaptest。在开始编码之前，我们还需要先将百度地图 Android 版的 SDK 准备好，下载地址是：<http://developer.baidu.com/map/sdkandev-download.htm>，然后点击全部下载按钮就可以了。

下载完成后对压缩包解压，应该可以看到其中有三个压缩包。其中，Docs 包中含有百度地图的使用文档，Sample 包中含有一个使用百度地图的工程样例，Lib 包中含有使用百度地图所必须依赖的库文件。解压 Lib 包，这里面就是我们所需要的一切了，如图 11.10 所示。




名称	类型	大小
 baidumapapi_v2_3_1.jar	Executable Jar File	856 KB
 libBaiduMapSDK_v2_3_1.so	SO 文件	1,190 KB
 readme.txt	文本文档	7 KB

图 11.10

baidumapapi_v2_3_1.jar 和 libBaiduMapSDK_v2_3_1.so 这两个文件都是使用百度地图所必不可少的。现在将 baidumapapi_v2_3_1.jar 拷贝到项目的 libs 目录下，然后在 libs 目录下新建一个 armeabi 目录，并将 libBaiduMapSDK_v2_3_1.so 拷贝到新建的目录下，所图 11.11 所示。

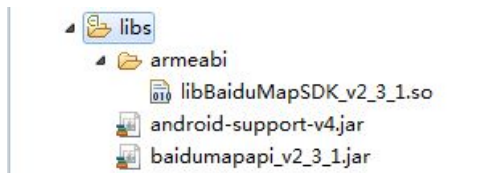


图 11.11

libs 目录你已经知道是专门用于存放第三方 Jar 包的地方，而 armeabi 目录则是专门用于存放 so 文件的地方。so 文件是用 C/C++ 语言进行编写，然后再用 NDK 编译出来的。libBaiduMapSDK_v2_3_1.so 这个文件已经由百度帮我们编译好了，因此直接放到 armeabi 目录下就可以使用了。

接下来修改 activity_main.xml 中的代码，如下所示：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <com.baidu.mapapi.map.MapView
        android:id="@+id/map_view"
        android:layout_width="match_parent"
```



```
        android:layout_height="match_parent"
        android:clickable="true" />
```

```
</LinearLayout>
```

在布局文件中我们只是放置了一个 MapView，并让它填满整个屏幕。这个 MapView 是由百度提供的自定义控件，所以在使用它的时候需要将完整的包名加上。

然后修改 MainActivity 中的代码，如下所示：

```
public class MainActivity extends Activity {

    private BMapManager manager;

    private MapView mapView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        manager = new BMapManager(this);
        manager.init("SHVPoTtIpzfonPD3HCkc5sIt", null);
        setContentView(R.layout.activity_main);
        mapView = (MapView) findViewById(R.id.map_view);
        mapView.setBuiltInZoomControls(true);
    }

    @Override
    protected void onResume() {
        mapView.onResume();
        if (manager != null) {
            manager.start();
        }
        super.onResume();
    }

    @Override
    protected void onPause() {
        mapView.onPause();
        if (manager != null) {
            manager.stop();
        }
    }
}
```

```

        super.onPause();
    }

    @Override
    protected void onDestroy() {
        mapView.destroy();
        if (manager != null) {
            manager.destroy();
            manager = null;
        }
        super.onDestroy();
    }
}

```

可以看到，这里的代码也是非常简单。首先需要创建一个 BMapManager 对象，然后调用它的 init() 方法来进行初始化操作。init() 方法接收两个参数，第一个参数就是在上一小节中我们申请到的 API Key，第二个参数传入 null 即可。注意初始化操作一定要在 setContentView() 方法前调用，不然的话就会出错。接下来我们获取到了 MapView 的实例，然后调用它的 setBuiltInZoomControls() 方法并传入 true，表示启用内置的缩放控制功能。

另外还需要重写 onResume()、onPause() 和 onDestroy() 这三个方法，在这里对百度地图的 API 进行管理，以保证资源能够及时地得到释放。

到此为止，我们的代码都十分简练，但下面的部分就十分繁杂了。相信你已经猜到，使用百度地图也需要在 AndroidManifest.xml 中声明权限的，不过不同于以往，这次我们要声明好多个权限才能保证百度地图的所有功能都可以正常使用。修改 AndroidManifest.xml 中的代码，如下所示：

```

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.baidumaptest"
    android:versionCode="1"
    android:versionName="1.0" >
    .....

    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_SETTINGS" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />

```

```
<uses-permission android:name="android.permission.CALL_PHONE" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_MOCK_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_GPS" />
.....
</manifest>
```

其中虽然有一些权限在我们当前的例子中是用不到的，但全部添加进来也不见得是一件坏事，这样就不会时不时因为权限不足的问题导致程序崩溃了。

现在运行一下程序，百度地图就应该成功显示出来了，如图 11.12 所示。



图 11.12

由于我们启用了内置的缩放控制功能，因此屏幕的右下角会有两个用于放大和缩小的按钮，除此之外，使用多点触控的方式也可以对地图进行缩放。

11.4.3 定位到我的位置

地图是成功显示出来了，但也许这并不是你想要的。因为这是一张世界地图的全貌，而你可能希望看到更加精细的地图信息，比如说自己所在位置的周边环境。显然，通过缩放的方式来慢慢找到自己的位置是一种很愚蠢的做法。那么本小节我们就来学习一下，如何才能

在地图中快速定位到自己的位置。

百度地图的 API 中提供了一个 MapController 类，它是地图的总控制器，调用 MapView 的 getController() 方法就能获取到 MapController 的实例，如下所示：

```
MapController controller = mapView.getController();
```

有了 MapController 后，我们就能对地图进行各种各样的操作了，比如设置地图的缩放级别就可以这样写：

```
controller.setZoom(12);
```

其中 12 就表示一个缩放级别，其取值范围是 3 到 19，级别越高，地图显示的信息就越精细。

那么怎样才能让地图定位到某一个经纬度上呢？这就需要借助 GeoPoint 类了。其实 GeoPoint 并没有什么太多的用法，主要就是用于存放经纬度值的，它的构造方法接收两个参数，第一个参数是纬度值，第二参数是经度值。但是需要注意，GeoPoint 是以微度为单位的，因此我们还要把经纬度的值乘以 10 的 6 次方再传给 GeoPoint。之后调用 MapController 的 setCenter() 方法，并把 GeoPoint 的实例传入就可以了，写法如下：

```
GeoPoint point = new GeoPoint((int) (39.915 * 1E6), (int) (116.404 * 1E6));  
mMapController.setCenter(point);
```

上述代码就实现了将地图定位到北纬 39.915 度、东经 116.404 度这个位置的功能。

了解了这些知识之后，接下来再去实现在地图中快速定位自己位置的功能就变得非常简单了。首先我们可以利用在 11.2 节中所学的定位技术来获得自己当前位置的经纬度，之后将经纬度的值传入到 GeoPoint 的构造方法中，再调用 MapController 的 setCenter() 方法来设置地图的中心点就完成了。

那么下面我们就来继续完善 BaiduMapTest 这个项目，加入定位到我的位置这个功能。

修改 MainActivity 中的代码，如下所示：

```
public class MainActivity extends Activity {  
    .....  
  
    private LocationManager locationManager;  
  
    private String provider;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        manager = new BMapManager(this);  
        manager.init("SHVPoTtIpzfonPD3HCkc5sIt", null);  
    }  
}
```

```

        setContentView(R.layout.activity_main);
        mapView = (MapView) findViewById(R.id.map_view);
        mapView.setBuiltInZoomControls(true);
        locationManager = (LocationManager) getSystemService(Context.
LOCATION_SERVICE);
        // 获取所有可用的位置提供器
        List<String> providerList = locationManager.getProviders(true);
        if (providerList.contains(LocationManager.GPS_PROVIDER)) {
            provider = LocationManager.GPS_PROVIDER;
        } else if (providerList.contains(LocationManager.NETWORK_PROVIDER)) {
            provider = LocationManager.NETWORK_PROVIDER;
        } else {
            // 当没有可用的位置提供器时，弹出Toast提示用户
            Toast.makeText(this, "No location provider to use",
Toast.LENGTH_SHORT).show();
            return;
        }
        Location location = locationManager.getLastKnownLocation(provider);
        if (location != null) {
            navigateTo(location);
        }
    }

    private void navigateTo(Location location) {
        MapController controller = mapView.getController();
        controller.setZoom(16); // 设置缩放级别
        GeoPoint point = new GeoPoint((int) (location.getLatitude() * 1E6),
                                      (int) (location.getLongitude() * 1E6));
        controller.setCenter(point); // 设置地图中心点
    }
    .....
}

```

这里大部分的代码你应该非常熟悉了，基本上使用的都是我们在 11.2.2 节编写的定位代码，只不过在获取到了 Location 对象后，我们将它传入到了 navigateTo() 方法中。那么 navigateTo() 方法又做了什么呢？其实也非常简单，先是获取到了 MapController 的实例，并将地图的缩放级别设置成 16，然后把 Location 中存储的经纬度值乘以 10 的 6 次方后传入 GeoPoint 的构造函数，最后调用了 setCenter() 方法。

现在重新运行一下程序，结果如图 11.13 所示。



图 11.13

怎么样？这时地图上的信息看起来就要比世界地图丰富得多了吧。

11.4.4 使用覆盖物来增加更多功能

除了普通的地图展示之外，百度地图还提供了一种叫作覆盖物的功能，所有叠加或覆盖到地图上的内容都被统称为地图覆盖物，如标注、矢量图形元素、定位图标等。覆盖物拥有自己的地理坐标，当我们拖动或缩放地图时，它们会自动进行相应地移动。

百度地图提供了很多种类型的覆盖物，开发人员可以根据自己的实际需求来选择使用哪些覆盖物，本小节中我们就来学习其中的两种。

在百度地图所有的覆盖物中，最常用的就是 `MyLocationOverlay` 了，它主要的作用是在地图中添加一个图层，以标注出设备当前的位置。而 `MyLocationOverlay` 的用法也是非常简单，我们直接就在 `BaiduMapTest` 项目的基础上继续编写了，修改 `MainActivity` 中的代码，如下所示：

```
public class MainActivity extends Activity {
    .....

    private void navigateTo(Location location) {
        MapController controller = mapView.getController();
        // 设置缩放级别
        controller.setZoom(16);
    }
}
```

```
GeoPoint point = new GeoPoint((int) (location.getLatitude() * 1E6),
                                (int) (location.getLongitude() * 1E6));
// 设置地图中心点
controller.setCenter(point);
MyLocationOverlay myLocationOverlay = new MyLocationOverlay(mapView);
LocationData locationData = new LocationData();
// 指定我的位置
locationData.latitude = location.getLatitude();
locationData.longitude = location.getLongitude();
myLocationOverlay.setData(locationData);
mapView.getOverlays().add(myLocationOverlay);
mapView.refresh(); // 刷新使新增覆盖物生效
}
.....
}
```

可以看到，这里首先是创建了一个 `MyLocationOverlay` 的实例，然后通过 `LocationData` 对象指定了当前的经纬度数据，并调用 `setData()` 方法将 `LocationData` 存放到了 `MyLocationOverlay` 中。之后通过 `MapView` 的 `getOverlays()` 方法可以得到一个用于管理覆盖物的集合，再调用 `add()` 方法将 `MyLocationOverlay` 这个覆盖物添加到集合中。最后，还需要调用一下 `MapView` 的 `refresh()` 方法使新增的覆盖物生效。

就是这么简单，现在重新运行一下程序，结果如图 11.14 所示。



图 11.14

这样的话，用户就可以非常清晰地看出自己当前是在哪里了。

MyLocationOverlay 的用法确实非常简单，那么下面我们再学习一下 PopupOverlay 这种覆盖物的用法吧。相比于 MyLocationOverlay，PopupOverlay 允许我们自己指定覆盖物上显示的图片，并且还可以响应图片的点击事件，每个 PopupOverlay 上最多可以显示三张图片。

那么为了要尝试一下 PopupOverlay 的用法，我就事先准备好了三张图片存放在 drawable 目录下，分别命名为 left.png、middle.png 和 right.png。然后修改 MainActivity 中的代码，如下所示：

```
public class MainActivity extends Activity {
    .....

    private void navigateTo(Location location) {
        MapController controller = mapView.getController();
        // 设置缩放级别
        controller.setZoom(16);
        GeoPoint point = new GeoPoint((int) (location.getLatitude() * 1E6),
                                      (int) (location.getLongitude() * 1E6));
        // 设置地图中心点
        controller.setCenter(point);
        .....

        PopupOverlay pop = new PopupOverlay(mapView, new PopupClickListener() {
            @Override
            public void onClickedPopup(int index) {
                // 相应图片的点击事件
                Toast.makeText(MainActivity.this,
                              "You clicked button " + index, Toast.LENGTH_SHORT).
show();
            }
        });
        // 创建一个长度为3的Bitmap数组
        Bitmap[] bitmaps = new Bitmap[3];
        try {
            // 将三张图片读取到内存中
            bitmaps[0] = BitmapFactory.decodeResource(getResources(),
R.drawable.left);
            bitmaps[1] = BitmapFactory.decodeResource(getResources(),
R.drawable.middle);
            bitmaps[2] = BitmapFactory.decodeResource(getResources(),
R.drawable.right);
        } catch (Exception e) {
```



```
        e.printStackTrace();
    }
    pop.showPopup(bitmap, point, 18);
}
.....
}
```

令人高兴的是，PopupOverlay 的用法也非常简单，总共没几行代码。首先同样是需要创建一个 PopupOverlay 的实例，注意在构造方法的参数里面可以传入一个 PopupClickListener 的实现，它是用于处理图片的点击事件的，简单起见，我们只是在图片被点击的时候弹出一个 Toast。接下来创建了一个长度为 3 的 Bitmap 数组，然后调用 BitmapFactory 的 decodeResource() 方法来加载 left、middle 和 right 这三张图片，并把它们存放到 Bitmap 数组当中。最后调用 PopupOverlay 的 showPopup() 方法将这个覆盖物显示出来，showPopup() 方法接收三个参数，第一个参数就是前面创建的 Bitmap 数组，第二个参数是一个用于指定地理位置的 GeoPoint 对象，第三个参数是覆盖物在垂直方法上的偏移距离。现在重新运行一下代码，应该就会看到有一个 popup 窗口显示在我的位置上方，并且窗口上的图片都是可以点击的，如图 11.15 所示。



图 11.15

关于百度地图的用法我就准备介绍这么多，现在你已经算是成功入门了。如果想要更加深入地研究百度地图的用法，可以到官方网站上参考开发指南：<http://developer.baidu.com/map/sdk-android.htm>。

好了，本章的主体内容到这里就结束了。下面我们将再次进入本书的特殊环节，学习一下关于 Git 的高级用法。

经验值：+22000 目前经验值：254905

级别：头领鸟

赢得宝物：战胜百度地图守护者（内层高阶守护者）。拾取百度地图守护者掉落的宝物，一本书《孙子兵法》、一本神界自驾游旅游手册，还有一辆 SUV。是的！一辆 SUV！你能想象我当时震惊的样子么？！一辆 SUV 从一个人身上“掉落”，而不是反过来。在我没有打败他之前，我完全看不出他会是那种在身上藏了一辆 SUV 的人。那么大的 SUV，他能藏哪儿呢？！我使劲地揪头发，我快疯了。停！我告诉自己，冷静。这里是神界，是神的地盘，在这里，任何事情都可能发生。此外，百度地图守护者也不是人，他是神，尽管在身材上只比普通入健硕一点。开车会错过很多东西，况且我驾驶技能也很一般，因此，我决定继续选择步行。我卖掉了 SUV。继续前进。

11.5 Git 时间，版本控制工具的高级用法

现在的你对于 Git 应该完全不会感到陌生了吧，通过了之前两节内容的学习，你已经掌握了 Git 中很多的常用命令，像提交代码这种简单的操作相信肯定是难不倒你的。

那么打开 Git Bash，并进入到 BaiduMapTest 这个项目的根目录，然后执行提交操作：

```
git init
git add .
git commit -m "First Commit."
```

这样就将准备工作完成了，下面就让我们开始学习关于 Git 的高级用法。

11.5.1 分支的用法

分支是版本控制工具中比较高级且比较重要的一个概念，它主要的作用就是在现有代码的基础上开辟一个分叉口，使得代码可以在主干线和分支线上同时进行开发，且相互之间不会互相影响。分支的工作原理示意图如图 11.16 所示。

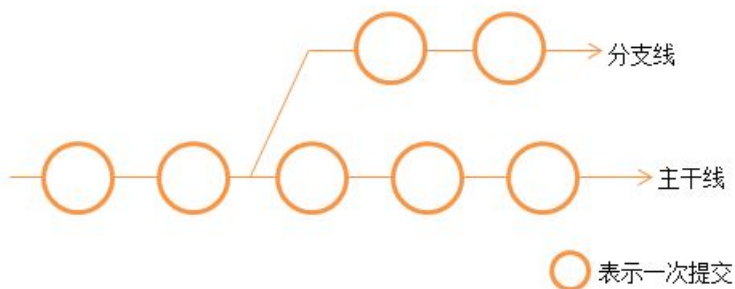


图 11.16

你也许会有疑惑，为什么需要建立分支呢，只在主干线上进行开发不是挺好的吗？没错，通常情况下，只在主干线上进行开发是完全没有问题的，不过一旦涉及到出版本的情况，如果不建立分支的话，你就会非常地头疼。举个简单的例子吧，比如说你们公司研发了一款不错的软件，最近刚刚完成，并推出了 1.0 版本。但是领导是不会让你们闲着的，马上提出了新的需求，让你们投入到了 1.1 版本的开发工作当中。过了几个星期，1.1 版本的功能已完成了一半，但是这个时候有用户反馈，之前上线的 1.0 版本发现了几个重大的 bug，严重影响软件的正常使用。领导也相当重视这个问题，要求你们立刻修复这些 bug，并重新发布 1.0 版本，但这个时候你就非常为难了，你会发现根据没法去修复这些 bug。因为现在 1.1 版本已开发一半了，如果在现有代码的基础上修复这些 bug，那么更新的 1.0 版本将会带有一半 1.1 版本的功能！

进退两难了是不是？但是如果你使用了分支的话，就完全不会存在这个让人头疼的问题。你只需要在发布 1.0 版本的时候建立一个分支，然后在主干线上继续开发 1.1 版本的功能。当 1.0 版本上发现任何 bug 的时候，就在分支线上进行修改，然后发布新的 1.0 版本，并记得将修改后的代码合并到主干线上。这样的话，不仅可以轻松解决掉 1.0 版本存在的 bug，而且保证了主干线上的代码也已经修复了这些 bug，当 1.1 版本发布时就不会有同样的 bug 存在了。

说了这么多，相信你也已经意识到分支的重要性了，那么我们马上来学习一下如何在 Git 中操作分支吧。

分支的英文名是 **branch**，如果想要查看当前的版本库当中有哪些分支，可以使用 `git branch -a` 这个命令，结果如图 11.17 所示。

```
Tony@TONY-PC /f/codes/AndroidFirstLine/BaiduMapTest (master)
$ git branch -a
* master
```

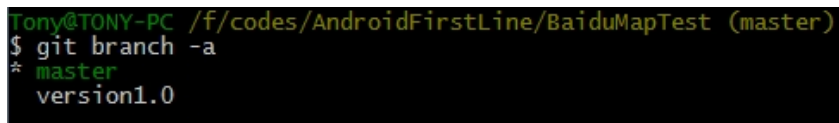
图 11.17

由于目前 BaiduMapTest 项目中还没有创建过任何分支，因此只有一个 master 分支存在，

这也就是前面所说的主干线。接下来我们尝试去创建一个分支，命令如下：

```
git branch version1.0
```

这样就创建了一个名为 version1.0 的分支，我们再次输入 `git branch -a` 这个命令来检查一下，结果如图 11.18 所示。



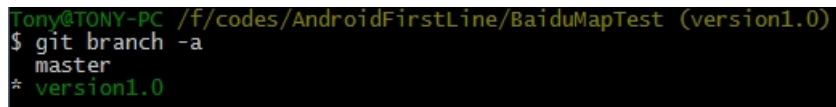
```
Tony@TONY-PC /f/codes/AndroidFirstLine/BaiduMapTest (master)
$ git branch -a
* master
  version1.0
```

图 11.18

可以看到，果然有一个叫作 version1.0 的分支出现了。你会发现，master 分支的前面有一个*号，说明目前我们的代码还是在 master 分支上的，那么怎样才能切换到 version1.0 这个分支上呢？其实也很简单，只需要使用 `checkout` 命令即可，如下所示：

```
git checkout version1.0
```

再次输入 `git branch -a` 来进行检查，结果如图 11.19 所示。



```
Tony@TONY-PC /f/codes/AndroidFirstLine/BaiduMapTest (version1.0)
$ git branch -a
  master
* version1.0
```

图 11.19

可以看到，我们已经把代码成功切换到 version1.0 这个分支上了。

需要注意的是，在 version1.0 分支上修改并提交的代码将不会影响到 master 分支。同样的道理，在 master 分支上修改并提交的代码也不会影响到 version1.0 分支。因此，如果我们在 version1.0 分支上修复了一个 bug，在 master 分支上这个 bug 仍然是存在的。这时将修改的代码一行行复制到 master 分支上显然不是一种聪明的做法，最好的办法就是使用 `merge` 命令来完成合并操作，如下所示：

```
git checkout master
git merge version1.0
```

仅仅这样简单的两行命令，就可以把在 version1.0 分支上修改并提交的内容合并到 master 分支上了。当然，在合并分支的时候还有可能出现代码冲突的情况，这个时候你就需要静下心来慢慢地找出并解决这些冲突，Git 在这里就无法帮助你了。

最后，当我们不再需要 version1.0 这个分支的时候，可以使用如下命令将这个分支删除掉：

```
git branch -D version1.0
```

11.5.2 与远程版本库协作

可以这样说，如果你是一个人在开发，那么使用版本控制工具就远远无法发挥出它真正强大的功能。没错，所有版本控制工具最重要的一个特点就是可以使用它来进行团队合作开发。每个人的电脑上都会有一份代码，当团队的某个成员在自己的电脑上编写完成了某个功能后，就将代码提交到服务器，其他的成员只需要将服务器上的代码同步到本地，就能保证整个团队所有人的代码都相同的。这样的话，每个团队成员就可以各司其职，大家共同来完成一个较为庞大的项目。

那么如何使用 Git 来进行团队合作开发呢？这就需要有一个远程的版本库，团队的每个成员都从这个版本库中获取到最原始的代码，然后各自进行开发，并且以后每次提交的代码都同步到远程版本库上就可以了。另外，团队中的每个成员最好都要养成经常从版本库中获取最新代码的习惯，不然的话，大家的代码就很有可能经常出现冲突。

比如说现在有一个远程版本库的 Git 地址是 `https://github.com/exmaple/test.git`，就可以使用如下的命令将代码下载到本地：

```
git clone https://github.com/exmaple/test.git
```

之后你在这份代码的基础进行了一些修改和提交，那么怎样才能把本地修改的内容同步到远程版本库上呢？这就需要借助 `push` 命令来完成了，用法如下所示：

```
git push origin master
```

其中 `origin` 部分指定的是远程版本库的 Git 地址，`master` 部分指定的是同步到哪一个分支上，上述命令就完成了将本地代码同步到 `https://github.com/exmaple/test.git` 这个版本库的 `master` 分支上的功能。

知道了将本地的修改同步到远程版本库上的方法，接下来我们看一下如何将远程版本库上的修改同步到本地。Git 提供了两种命令来完成此功能，分别是 `fetch` 和 `pull`，`fetch` 的语法规则和 `push` 是差不多的，如下所示：

```
git fetch origin master
```

执行这个命令后，就会将远程版本库上的代码同步到本地，不过同步下来的代码并不会合并到任何分支上去，而是会存放在到一个 `origin/master` 分支上，这时我们可以通过 `diff` 命令来查看远程版本库上到底修改了哪些东西：

```
git diff origin/master
```

之后再调用 `merge` 命令将 `origin/master` 分支上的修改合并到主分支上即可，如下所示：

```
git merge origin/master
```

而 `pull` 命令则是相当于将 `fetch` 和 `merge` 这两个命令放在一起执行了，它可以从远程版

本库上获取最新的代码并且合并到本地，用法如下所示：

```
git pull origin master
```

也许你现在对远程版本库的使用还会感觉比较抽象，没关系，因为暂时我们只是了解一下命令的用法，还没进行实践，在第 14 章当中，你将会对远程版本库的用法有更深一层的认识。

11.6 小结与点评

不得不说，本章中学到的知识应该还算是蛮有趣的吧？在这次的 Android 特色开发环节中，我们主要学习了基于位置服务的工作原理和用法，借助手机内置的定位功能，我们就可以随时确定自己当前位置的经纬度，并且通过反向地理编码的方式，还能将经纬度转换成看得懂的位置信息。之后我们又学习了百度地图的用法，不仅成功地将地图信息显示了出来，还综合利用了前面所学到的定位技术实现了一个较为完整的例子。

除了基于位置的服务之外，本章 Git 时间中继续对 Git 的用法进行了更深一步的探究，使得我们对分支和远程版本库的使用都有了一定层次的了解。

既然 Android 特色开发环节这么有意思，仅仅一章显然不够过瘾，那么下一章中我们将继续学习 Android 中的特色开发技术，研究一下传感器的用法。

经验值：+10000 目前经验值：264905

级别：头领鸟

获赠宝物：拜会 Git 领主父亲的属地。获赠不少盘缠。