

第 24 章

位置服务与地图应用

由于移动设备比电脑更容易随身携带，所以在移动设备上使用位置服务与地图应用，就更加实用了。本章将对 Android 中的位置服务与百度地图应用进行详细的介绍。

24.1 位置服务

在开发 Android 位置服务相关应用时，可以从 GPS（全球定位系统）或者网络获得用户位置。通过 GPS 能获得最精确的信息。例如，在某些外卖软件的“选择收货地址”页面中，不仅定位了当前所在城市，而且附近地址也被列出，界面如图 24.1 所示；再如，某些天气软件能够自动定位当前所在城市，界面如图 24.2 所示。



图 24.1 定位当前位置

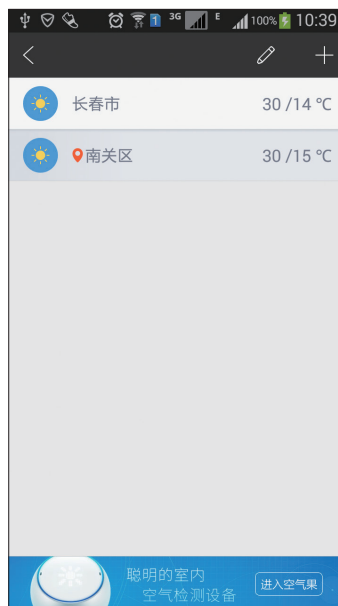


图 24.2 自动定位当前所在城市

对于开发 Android 应用的程序员来说，开发 GPS 功能的应用程序非常简单。在 Android 系统中，开发人员需要使用以下类访问定位服务。

◆ **LocationManager**：该类提供系统定位服务访问功能。

LocationManager 提供的常用方法如表 24.1 所示。

表 24.1 LocationManager 提供的常用方法

方 法 名	描 述
List<String> getAllProviders()	获取所有的 LocationProvider 列表
Location getLastKnownLocation(String provider)	根据 LocationProvider 获取最近一次已知的 Location
LocationProvider getProvider(String name)	根据名称来获取 LocationProvider
void requestLocationUpdates (String provider, long minTime, float minDistance, PendingIntent intent)	通过指定的 LocationProvider 周期性地获取定位信息，并通过 intent 启动相应的组件
void requestLocationUpdates (String provider, long minTime, float minDistance, LocationListener listener)	通过指定的 LocationProvider 周期性地获取定位信息，并触发 listener 所对应的触发器

☑ LocationProvider: 定位组件的抽象表示，通过该类可以获取该定位组件的相关信息。
LocationProvider 提供的常用方法如表 24.2 所示。

表 24.2 LocationProvider 提供的常用方法

方 法 名	描 述
int getAccuracy()	返回 LocationProvider 的精度
String getName()	返回 LocationProvider 的名称
int getPowerRequirement()	获取 LocationProvider 的电源需求

◆ Location: 该类表示特定时间的地理位置信息，位置由经度、纬度、UTC 时间戳以及可选的高度、速度、方向等组成。

Location 提供的常用方法如表 24.3 所示。

表 24.3 Location 提供的常用方法

方 法 名	描 述	方 法 名	描 述
float getAccuracy()	获取定位信息的精度	double getLongitude()	获取定位信息的经度
double getAltitude()	获取定位信息的高度	String getProvider()	获取提供该定位信息的 LocationProvider
float getBearing()	获取定位信息的方向	float getSpeed()	获取定位信息的速度
double getLatitude()	获取定位信息的纬度		

上面的 3 个 API 就是 GPS 定位支持的 3 个核心 API，使用它们获取 GPS 定位信息的一般步骤如下：

- (1) 获取系统的 LocationManager 对象。
- (2) 使用 LocationManager，通过指定 LocationProvider 来获取定位信息，定位信息由 Location 对象来表示。
- (3) 从 Location 对象中获取定位信息。

24.1.1 获取 LocationProvider

LocationProvider 是位置源的意思，用来提供定位信息。在获取定位信息之前，需要先获得 Location Provider 对象。常用的 LocationProvider 如表 24.4 所示。

表 24.4 常用的 LocationProvider

方 法 名	描 述
passive	被动定位方式，即利用其他应用程序使用定位更新了定位信息时，系统会保存下来，该应用接收到消息后直接读取就可以了
gps	通过手机里的 GPS 芯片利用卫星获取定位信息
network	通过网络获取定位信息。通常利用手机基站和 WIFI 节点的地址来大致定位

下面将介绍如何获得 Location Provider。

1. 获取所有可用的 LocationProvider

在 LocationManager 中，提供了一个 getAllProviders() 方法用来获取系统所有可用的 LocationProvider。下面通过一个例子来演示如何获得当前支持的全部 LocationProvider。

例 24.1 获取所有可用的 LocationProvider 名称

在 Android Studio 中创建一个 Module，名称为“Location Provider”。实现本实例的具体步骤如下：

(1) 修改新建 Module 的 res/layout 目录下的布局文件 activity_main.xml，首先将默认添加的布局管理器修改为相对布局管理器并添加背景图片，然后为默认添加的 TextView 组件设置属性值，最后再添加一个 TextView 组件，用于显示获取的 LocationProvider 名称。

(2) 打开默认添加的 MainActivity 类，让该类继承自 Activity，然后在 onCreate() 方法中完成获取及显示 LocationProvider 名称。具体代码如下：

```

01 public class MainActivity extends Activity {
02     @Override
03     protected void onCreate(Bundle savedInstanceState) {
04         super.onCreate(savedInstanceState);
05         setContentView(R.layout.activity_main);
06         //设置全屏显示
07         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
08             WindowManager.LayoutParams.FLAG_FULLSCREEN);
09         //获取显示LocationProvider名称的TextView组件
10         TextView textView = (TextView) findViewById(R.id.provider);
11         //获取location服务
12         LocationManager locationManager = (LocationManager)
13             getSystemService(LOCATION_SERVICE);
14         //获取系统所有的LocationProvider名称
15         List<String> providersNames = locationManager.getAllProviders();
16         StringBuilder stringBuilder = new StringBuilder();//使用StringBuilder保存数据
17         //遍历获取到的全部LocationProvider名称
18         for (Iterator<String> iterator = providersNames.iterator();
19             iterator.hasNext(); ) {

```

```
20         stringBuilder.append(iterator.next() + "\n");
21     }
22     textView.setText(stringBuilder.toString()); //显示LocationProvider名称
23 }
24 }
```

(3) 运行本实例，将显示如图 24.3 所示的界面效果。



图 24.3 获取所有可用的 LocationProvider 名称

2. 通过名称获得 LocationProvider

在通过调用 LocationManager 的 getAllProviders() 方法获取所有的 LocationProvider 时返回的是 List<String> 集合，集合中的元素为 LocationProvider 的名称。为了获取实际的 LocationProvider 对象，可以通过 LocationManager 的 getProvider() 方法。例如下面的代码可以获取基于 GPS 的 LocationProvider。

```
LocationProvider lp = lm.getProvider(LocationManager.GPS_PROVIDER);
```

3. 通过 Criteria 类获得 LocationProvider

通过 getAllProviders() 方法可以获取系统所有可用的 LocationProvider。但有些时候，应用程序可能希望得到符合指定条件的 LocationProvider，这就需要使用 LocationManager 的 getBestProvider(Criteria criteria, boolean enabledOnly) 方法来获取。在该方法中需要借助于 Criteria 类。

对于位置源而言，有两种用户十分关心的属性：精度和耗电量。在 Criteria 类中，保存了关于精度和耗电量的信息，其说明如表 24.5 所示。

表 24.5 Criteria 类定义的精度和耗电信息

常 量	说 明	常 量	说 明
ACCURACY_COARSE	近似的精度	NO_REQUIREMENT	无要求
ACCURACY_FINE	更精细的精度	POWER_HIGH	高耗电量
ACCURACY_HIGH	高等精度	POWER_MEDIUM	中耗电量

续表

常 量	说 明	常 量	说 明
ACCURACY_MEDIUM	中等精度	POWER_LOW	低耗电量
ACCURACY_LOW	低等精度		

下面通过一个实例来演示通过 Criteria 类获得 LocationProvider 名称。

例 24.2 通过 Criteria 类获得 LocationProvider 名称

在 Android Studio 中创建 Module，名称为“Criteria Location Provider”，实现本实例的具体步骤如下：

(1) 修改布局文件 activity_main.xml，首先将默认添加的布局管理器修改为相对布局管理器，然后为布局管理器添加背景图片，再为默认添加的 TextView 组件与新添加的 TextView 组件设置属性值，用于显示获取的 LocationProvider 名称。

(2) 打开默认添加的 MainActivity 类，该类继承 Activity，在 onCreate() 方法中，完成通过 Criteria 类过滤条件来获取及显示 LocationProvider 名称。具体代码如下：

```

01 public class MainActivity extends Activity {
02     @Override
03     protected void onCreate(Bundle savedInstanceState) {
04         super.onCreate(savedInstanceState);
05         setContentView(R.layout.activity_main);
06         //设置全屏显示
07         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
08             WindowManager.LayoutParams.FLAG_FULLSCREEN);
09         //获取显示最佳LocationProvider的TextView组件
10         TextView textView = (TextView) findViewById(R.id.provider);
11         //获取location服务
12         LocationManager locationManager =
13             (LocationManager) getSystemService(LOCATION_SERVICE);
14         Criteria criteria=new Criteria();           //创建过滤条件
15         criteria.setCostAllowed(false);             //使用不收费的
16         criteria.setAccuracy(Criteria.ACCURACY_FINE); //使用精度最准确的
17         criteria.setPowerRequirement(Criteria.POWER_LOW);//使用耗电量最低的
18         //获取最佳的LocationProvider名称
19         String provider=locationManager.getBestProvider(criteria,true);
20         textView.setText(provider); //显示最佳的LocationProvider名称
21     }
22 }

```

(3) 打开 AndroidManifest.xml 文件，在其中设置获取最佳的 LocationProvider 权限，具体代码如下：

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

(4) 运行本实例，将显示如图 24.4 所示的界面效果。



图 24.4 通过 Criteria 类获得 LocationProvider 名称

24.1.2 获取定位信息

通过手机可以实时获取定位信息，包括用户所在位置的经度、纬度、高度以及方向等。对于位置发生变化的用户，可以在变化后接收到相关的通知。在 `LocationManager` 类中，定义了多个 `requestLocationUpdates()` 方法，用来为当前 `Activity` 注册位置变化通知事件。该方法的声明如下：

```
public void requestLocationUpdates (String provider, long minTime, float minDistance,
LocationListener listener)
```

- ◆ `provider`：注册的 `provider` 的名称，可以是 `GPS_PROVIDER` 等。
- ◆ `minTime`：通知间隔的最小时间，单位是毫秒。系统可能为了省电而延长该时间。
- ◆ `minDistance`：更新通知的最小变化距离，单位是米。
- ◆ `listener`：用于处理通知的监听器。

在 `LocationListener` 接口中，定义了 4 个方法，其说明如表 24.6 所示。

表 24.6 `LocationListener` 接口中方法说明

方 法	说 明
<code>onLocationChanged</code>	当位置发生变化时调用该方法
<code>onProviderDisabled</code>	当 <code>provider</code> 禁用时调用该方法
<code>onProviderEnabled</code>	当 <code>provider</code> 启用时调用该方法
<code>onStatusChanged</code>	当状态发生变化时调用该方法

例 24.3

获取动态定位信息

在 Android Studio 中创建 Module，名称为“Get Location Information”。实现本实例的具体步骤如下：

- （1）修改布局文件 `activity_main.xml`，首先将默认添加的布局管理器修改为相对布局管理器并添加背景图片，然后为默认添加的 `TextView` 组件设置属性值，用于显示获取定位的信息。

(2) 打开 MainActivity 类，该类继承 Activity，定义所需的成员变量，并在 onCreate() 方法中，获取系统的 LocationManager 对象，具体代码如下：

```

01 public class MainActivity extends Activity {
02     private TextView text; //定义用于显示LocationProvider的TextView组件
03     @Override
04     public void onCreate(Bundle savedInstanceState) {
05         super.onCreate(savedInstanceState);
06         setContentView(R.layout.activity_main);
07         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
08             WindowManager.LayoutParams.FLAG_FULLSCREEN); //设置全屏显示
09         //获取显示Location信息的TextView组件
10         text = (TextView) findViewById(R.id.location);
11         //获取系统的LocationManager对象
12         LocationManager locationManager =
13             (LocationManager) getSystemService(LOCATION_SERVICE);
14     }
15 }

```

(3) 在 MainActivity 中，创建 locationUpdates() 方法，用于获取指定的查询信息并显示。具体代码如下：

```

01 public void locationUpdates(Location location) { //获取指定的查询信息
02     //如果location不为空时
03     if (location != null) {
04         StringBuilder stringBuilder = new StringBuilder();//使用StringBuilder保存数据
05         //获取经度、纬度、等属性值
06         stringBuilder.append("您的位置信息: \n");
07         stringBuilder.append("经度: ");
08         stringBuilder.append(location.getLongitude());
09         stringBuilder.append("\n纬度: ");
10         stringBuilder.append(location.getLatitude());
11         stringBuilder.append("\n精确度: ");
12         stringBuilder.append(location.getAccuracy());
13         stringBuilder.append("\n高度: ");
14         stringBuilder.append(location.getAltitude());
15         stringBuilder.append("\n方向: ");
16         stringBuilder.append(location.getBearing());
17         stringBuilder.append("\n速度: ");
18         stringBuilder.append(location.getSpeed());
19         stringBuilder.append("\n时间: ");
20         //设置日期时间格式
21         SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH mm ss");
22         stringBuilder.append(dateFormat.format(new Date(location.getTime())));
23         text.setText(stringBuilder); //显示获取的信息
24     } else {
25         //否则输出空信息
26         text.setText("没有获取到GPS信息");

```



```

27     }
28 }


```

(4) 在 onCreate() 方法中，通过 locationManager.requestLocationUpdates() 方法与位置监听器来设置每秒获取一次 location 信息，并将最新的定位信息传递给创建的 locationUpdates() 方法中。关键代码如下：

```

01 //设置每秒获取一次location信息
02 locationManager.requestLocationUpdates(
03     locationManager.GPS_PROVIDER,    //GPS定位提供者
04     1000,                               //更新数据时间为1000毫秒
05     1,                                  //位置间隔为1米
06     //位置监听器
07     new LocationListener() { //GPS定位信息发生改变时触发，用于更新位置信息
08         @Override
09         public void onLocationChanged(Location location) {
10             //GPS信息发生改变时，更新位置
11             locationUpdates(location);
12         }
13
14         @Override
15         //位置状态发生改变时触发
16         public void onStatusChanged(String provider, int status, Bundle extras) {
17         }
18         @Override
19         //定位提供者启动时触发
20         public void onProviderEnabled(String provider) {
21         }
22         @Override
23         //定位提供者关闭时触发
24         public void onProviderDisabled(String provider) {
25         }
26     });
27 //从GPS获取最新的定位信息
28 Location location = locationManager.getLastKnownLocation
29     (LocationManager.GPS_PROVIDER);
30 locationUpdates(location); //将最新的定位信息传递给创建的locationUpdates()方法中

```

 **说明** 添加上面的代码后，在代码的下方将出现红色浪线，此时按下〈Alt+Enter〉键，选择Add Permission ACCESS_FINE_LOCATION选项，然后再次按下〈Alt+Enter〉键，选择Add permission check选项，解决红色浪线问题。此时，Android Studio将自动添加以下权限检查的代码。

```

01 if (ActivityCompat.checkSelfPermission(this,
02     Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
03     && ActivityCompat.checkSelfPermission(this,
04     Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
05     return;
06 }

```


(5) 在 AndroidManifest.xml 文件中, 添加以下代码设置访问 LocationProvider 的相关权限, 具体代码如下:

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

(6) 运行本实例, 将显示如图 24.5 所示。



图 24.5 获取动态定位信息

说明 在运行本实例时, 需要开启手机的GPS功能, 同时需要开启该应用的定位权限。本实例在测试的过程中建议在室外进行测试, 保证GPS的信号质量。

24.2 百度地图服务

百度地图服务是目前很常用的地图服务。为了便于 Android 开发者使用百度地图, 百度提供了百度地图 Android SDK。它是一套基于 Android 2.1 及以上版本设备的应用程序接口, 适用于 Android 系统移动设备的地图应用开发。通过调用地图 SDK 接口, 可以轻松访问百度地图服务和数据, 构建功能丰富、交互性强的地图类应用程序。

24.2.1 获得地图 API 密钥

在使用百度地图 Android SDK 时, 首先需要申请密钥 (API Key)。如果没有 API 密钥, 就不会向应用程序返回任何地图图块 (map tile)。由于该密钥与百度账号相关联, 而且还与创建的过程名称有关, 因此, 在申请密钥前, 必须先注册百度账号。获取地图 API 密钥的具体步骤如下:

(1) 在浏览器的地址栏中输入密钥的申请地址 <http://lbsyun.baidu.com/apiconsole/key>, 百度账号注册完成后, 输入账号和密码进行百度账号的登录, 如果您没有注册为百度开发者, 还需要注册为百度开发者。这时, 页面将自动跳转到百度地图开放平台开发者激活页面。

(2) 百度地图开放平台开发者激活后, 单击“申请密钥”按钮, 将进入到如图 24.6 所示的创建应用页面。



图 24.6 创建应用页面

安全码的组成规则为：Android 签名证书的 SHA1 值 + “;” + packagename（即：数字签名 + 英文状态下的分号 + 包名）。在安全码的组成规则中，主要有以下两部分内容需要获取。

◆ 获取 Android 签名证书的 SHA1 值

Android 签名证书的 SHA1 值可以在 Android Studio 的 signingReport 中获取，具体操作方法如图 24.7 所示。

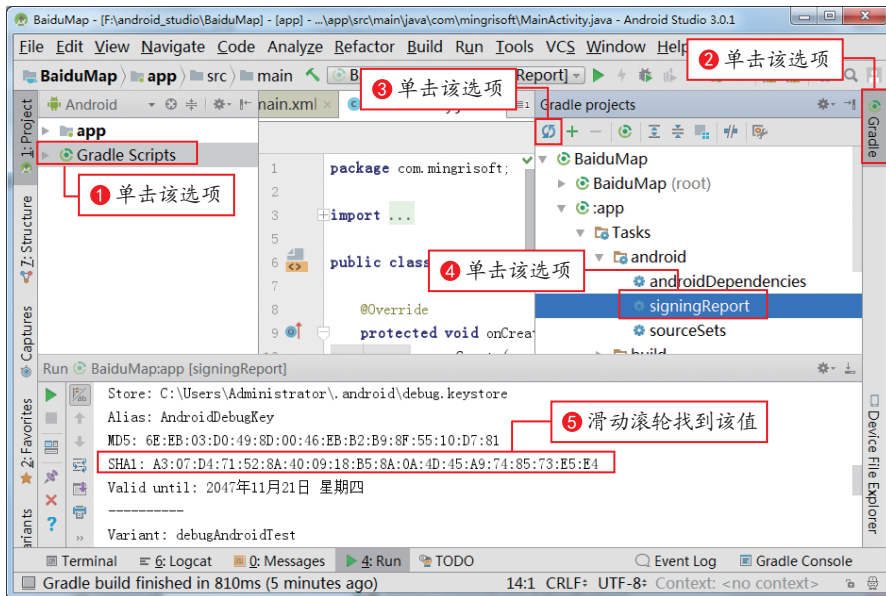


图 24.7 获取 Android 签名证书的 SHA1 值

◆ 获取包名

包名是在 AndroidManifest.xml 中，通过 package 属性定义的名称，如图 24.8 所示。

```
<?xml version="1.0" encoding="utf-8" package="com.mingrisoft">
```

包名

图 24.8 定义的包名

在图 24.6 所示的页面中输入“发布版 SHA1”的值和包名后会自动生成安全码，如图 24.9 所示。



图 24.9 自动生成安全码

(3) 单击“提交”按钮，返回到如图 24.10 所示的应用列表页面，在该页面中将显示刚刚创建的应用。



图 24.10 显示申请到的密钥

24.2.2 下载 SDK 开发包

要开发百度地图应用，需要下载百度地图 SDK 开发包，该包可以到百度地图 API 网站下载，具体下载步骤如下：

(1) 在浏览器的地址栏中输入网址 <http://lbsyun.baidu.com>，进入到百度地图 API 首页，将鼠标移动到“开发文档”超链接上将显示对应子菜单，单击“Android 地图 SDK”超链接进入到“Android 地图 SDK”页面，单击该页面左侧的“产品下载”超链接进入到如图 24.11 所示的相关下载页面。



图 24.11 产品下载页面

(2) 单击“自定义下载”按钮，进入到如图 24.12 所示的页面，在该页面中根据自己的项目需要勾选相应的功能。

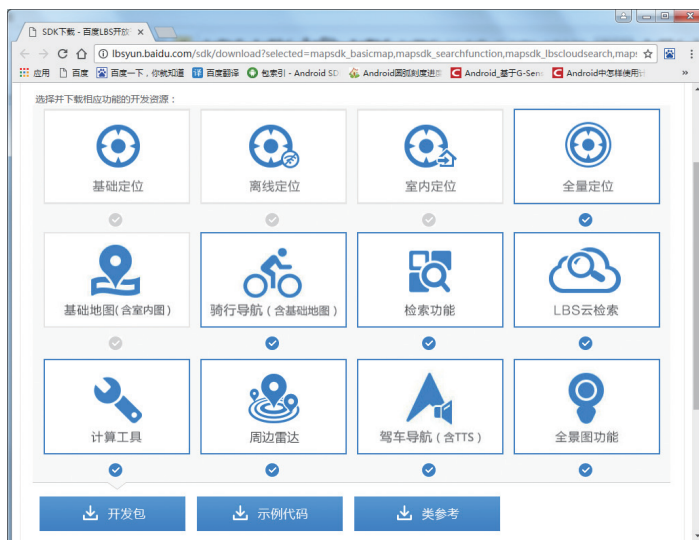


图 24.12 选择要下载的资源

(4) 在 AndroidManifest.xml 文件的 <application> 标签中添加子标签 <meta-data> 用于指定开发密钥, 格式如图 24.16 所示。

```
<application android:allowBackup="true" android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name" android:roundIcon="@mipmap/ic_launcher_round"
    android:supportRtl="true" android:theme="@style/AppTheme">
    <meta-data
        android:name="com.baidu.lbsapi.API_KEY"
        android:value="cmjo5n9tAbSnYSiM4Fol9aKsLbrOZNv5" />
</application>
```

图 24.16 指定开发密钥

例如, 申请的密钥为 cmjo5n9tAbSnYSiM4Fol9aKsLbrOZNv5, 可以使用下面的代码:

```
01 <meta-data
02     android:name="com.baidu.lbsapi.API_KEY"
03     android:value="cmjo5n9tAbSnYSiM4Fol9aKsLbrOZNv5" />
```

(5) 在 AndroidManifest.xml 文件的 <manifest> 标签中添加子标签 <uses-permission> 允许所需权限。通常情况下, 使用百度 API 需要允许以下权限。

```
01 <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
02 <uses-permission android:name="android.permission.INTERNET"/>
03 <uses-permission android:name="com.android.launcher.permission.READ_SETTINGS" />
04 <uses-permission android:name="android.permission.WAKE_LOCK"/>
05 <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
06 <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
07 <uses-permission android:name="android.permission.GET_TASKS" />
08 <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
09 <uses-permission android:name="android.permission.WRITE_SETTINGS" />
```

(6) 在布局文件 activity_main.xml 中, 首先将默认添加的布局管理器修改为相对布局管理器, 然后添加地图组件, 并删除 TextView 组件, 关键代码如下:

```
01 <com.baidu.mapapi.map.MapView
02     android:id="@+id/bmapview"
03     android:layout_width="match_parent"
04     android:layout_height="match_parent"
05     android:clickable="true" />
```

(7) 打开主活动 MainActivity, 在该文件中声明所需成员变量, 关键代码如下:

```
private MapView mMapView; //百度地图组件
```

(8) 在 MainActivity 的 onCreate() 方法中, 初始化 SDK 引用的 Context 全局变量, 然后获取布局文件中添加的百度地图组件, 以及百度地图对象, 关键代码如下:

```
01 @Override
02 protected void onCreate(Bundle savedInstanceState) {
03     super.onCreate(savedInstanceState);
04     SDKInitializer.initialize(getApplicationContext()); //初始化地图SDK
05     setContentView(R.layout.activity_main);
```

```

06     mMapView = (MapView) findViewById(R.id.bmapview);    //获取地图组件
07 }

```

注意 在initialize()方法中必须传入ApplicationContext，如果传入this，或者MainActivity.this都会在运行时报错，所以建议把该方法放到Application的初始化方法中。

(9) 重写 activity 的生命周期的几个方法来管理地图的生命周期。关键代码如下：

```

01 //实现地图生命周期管理
02 @Override
03 protected void onResume() {
04     super.onResume();
05     mMapView.onResume();
06 }
07
08 @Override
09 protected void onPause() {
10     super.onPause();
11     mMapView.onPause();
12 }
13
14 @Override
15 protected void onDestroy() {
16     mMapView.onDestroy();
17     mMapView = null;
18     super.onDestroy();
19 }

```

(10) 运行本实例，将显示如图 24.17 所示的界面效果。

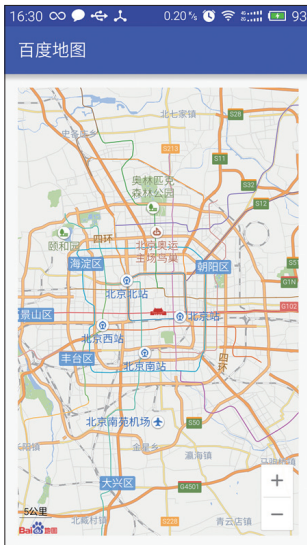


图 24.17 显示百度地图

说明 在运行使用百度地图API的程序时，需要连接互联网。

24.2.4 定位到我的位置

在实现定位到我的位置时，首先要开启定位图层，可以使用 `com.baidu.mapapi.map.BaiduMap` 对象的 `setMyLocationEnabled()` 方法实现。该方法的语法格式如下：

```
Public final void set MyLocationEnabled(boolean enabled)
```

其中，`enabled` 参数用于指定是否允许定位图层，值为 `true` 时表示允许，否则为不允许。开启定位图层的代码如下：

```
mBaiduMap.setMyLocationEnabled(true);
```

创建 `MyLocationData` 对象，用于构造定位数据，包括 GPS 定位时方向角度、纬度坐标、经度坐标、定位精度和时速等。例如，构造定位数据，指定 GPS 定位时方向角度为 100、坐标位置为当前位置，可以使用下面的代码。

```
01 MyLocationData locData = new MyLocationData.Builder()
02     .accuracy(location.getAccuracy()) //设置精度
03     .direction(100) //此处设置开发者获取到的方向信息，顺时针0-360
04     .latitude(location.getLatitude()) //设置纬度坐标
05     .longitude(location.getLongitude()) //设置经度坐标
06     .build();
```

 **说明** 此处代码将使用 Android 原有定位方法。

设置定位数据，并配置定位图层的一些信息。代码如下：


```
01 //设置定位数据
02 mBaiduMap.setMyLocationData(locData);
03 //设置自定义定位图标
04 BitmapDescriptor mCurrentMarker = BitmapDescriptorFactory
05     .fromResource(R.drawable.icon_geo);
06 mCurrentMode = MyLocationConfiguration.LocationMode.NORMAL; //设置定位模式
07 //位置构造方式，将定位模式，定义图标添加到该位置
08 MyLocationConfiguration config = new
09     MyLocationConfiguration(mCurrentMode, true, mCurrentMarker);
10 mBaiduMap.setMyLocationConfiguration(config); //地图显示定位图标
```

最后，在不需要定位图层时关闭定位图层。代码如下：

```
mBaiduMap.setMyLocationEnabled(false); //关闭定位图层
```

下面通过一个实例来演示如何在百度地图上定位我的位置。

例 24.5 百度地图定位我的位置

 **说明** 该实例在本章实例 24.4 基础上进行修改。

在 Android Studio 中创建 Module，名称为“My Location”。实现本实例的具体步骤如下：

(1) 打开主活动 `MainActivity`，在该文件中声明所需成员变量，关键代码如下：

```

01 private MapView mMapView;           //百度地图组件
02 private BaiduMap mBaiduMap;         //定义百度地图对象
03 private boolean isFirstLoc = true;   //定义第一次启动
04 private MyLocationConfiguration.LocationMode mCurrentMode; //定义当前定位模式

```

(2) 在 MainActivity 的 onCreate() 方法中, 首先获取百度地图对象, 然后设置地图缩放级别, 再获取系统的 LocationManager 对象, 关键代码如下:

```

01 mBaiduMap = mMapView.getMap();           //获取百度地图对象
02 mBaiduMap.setMapStatus(MapStatusUpdateFactory.newMapStatus
03     (new MapStatus.Builder().zoom(13).build())); //设置缩放级别
04 //获取系统的LocationManager对象
05 LocationManager locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);


```

(3) 在 onCreate() 方法中, 通过 locationManager.requestLocationUpdates() 方法与位置监听器来设置每秒获取一次 location 信息, 并将最新的定位信息传递给创建的 locationUpdates() 方法中。关键代码如下:

```

01 //设置每一秒获取一次location信息
02 locationManager.requestLocationUpdates(
03     locationManager.GPS_PROVIDER,           //GPS定位提供者
04     1000,                                     //更新数据时间为1000毫秒
05     1,                                       //位置间隔为1米
06     //位置监听器
07     new LocationListener() { //GPS定位信息发生改变时触发, 用于更新位置信息
08         @Override
09         public void onLocationChanged(Location location) {
10             //GPS信息发生改变时, 更新位置
11             locationUpdates(location);
12         }
13         @Override
14         //位置状态发生改变时触发
15         public void onStatusChanged(String provider, int status, Bundle extras) {
16         }
17         @Override
18         //定位提供者启动时触发
19         public void onProviderEnabled(String provider) {
20         }
21         @Override
22         //定位提供者关闭时触发
23         public void onProviderDisabled(String provider) {
24         }
25     });
26 //从GPS获取最新的定位信息
27 Location location = locationManager.getLastKnownLocation
28     (locationManager.GPS_PROVIDER);
29 locationUpdates(location); //将最新的定位信息传递给创建的locationUpdates()方法中

```

 **说明** 添加上的代码后，在代码的下方将出现红色浪线，此时按下〈Alt+Enter〉键，然后选择Add Permission ACCESS_FINE_LOCATION选项，添加该权限解决红色浪线问题。此时，Android Studio将自动添加以下权限检查的代码：

```
01 //添加权限检查
02 if (ActivityCompat.checkSelfPermission(this,
03     Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
04     && ActivityCompat.checkSelfPermission(this,
05     Manifest.permission.ACCESS_COARSE_LOCATION)
06     != PackageManager.PERMISSION_GRANTED) {
07     return;
08 }
```

(4) 在 MainActivity 中，创建 locationUpdates() 方法，用于获取当前的经纬度，并设置定位。当获取的信息不为空时，首先设置第一次定位的中心点为当前位置，然后构造和设置定位数据，最后在地图上显示定位图标。关键代码如下：

```
01 public void locationUpdates(Location location) { //获取指定的查询信息
02     //如果location不为空时
03     if (location != null) {
04         //获取用户当前经纬度
05         LatLng ll = new LatLng(location.getLatitude(), location.getLongitude());
06         Log.i("Location", "纬度: "+
07             location.getLatitude()+" | 经度: "+location.getLongitude());
08         if (isFirstLoc) { //如果是第一次定位,就定位到以自己为中心
09             MapStatusUpdate u = MapStatusUpdateFactory.newLatLng(ll); //更新坐标位置
10             mBaiduMap.animateMapStatus(u); //设置地图位置
11             isFirstLoc = false; //取消第一次定位
12         }
13         //构造定位数据
14         MyLocationData locData = new MyLocationData.Builder().
15             accuracy(location.getAccuracy()) //设置精度
16             .direction(100) //此处设置开发者获取到的方向信息,顺时针0-360
17             .latitude(location.getLatitude()) //设置纬度坐标
18             .longitude(location.getLongitude()) //设置经度坐标
19             .build();
20
21         //设置定位数据
22         mBaiduMap.setMyLocationData(locData);
23         //设置自定义定位图标
24         BitmapDescriptor mCurrentMarker = BitmapDescriptorFactory
25             .fromResource(R.drawable.icon_geo);
26         mCurrentMode = MyLocationConfiguration.LocationMode.NORMAL; //设置定位模式
27         //位置构造方式,将定位模式,定义图标添加到该位置
28         MyLocationConfiguration config = new
29             MyLocationConfiguration(mCurrentMode, true, mCurrentMarker);
30         mBaiduMap.setMyLocationConfiguration(config); //地图显示定位图标
```

```

31     } else {
32         //否则输出空信息
33         Log.i("Location","没有获取到GPS信息");
34     }
35 }

```

(5) 重写 Activity 的 onStart() 方法和 onStop() 方法，在 onStart() 方法中启动地图定位，在 onStop() 方法中停止地图定位。关键代码如下：

```

01 @Override
02 protected void onStart() {                //启动地图定位
03     super.onStart();
04     mBaiduMap.setMyLocationEnabled(true);    //启动定位图层
05 }
06 @Override
07 protected void onStop() {                  //停止地图定位
08     super.onStop();
09     mBaiduMap.setMyLocationEnabled(false);   //关闭定位图层
10 }

```

(6) 在 AndroidManifest.xml 文件中，添加以下代码设置访问 LocationProvider 的相关权限，具体代码如下：

```

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

```

(7) 运行本实例，将显示如图 24.18 所示。



图 24.18 在百度地图上显示我的位置

说明 运行本实例需要在手机中进入“设置”→“应用”→你的App应用名称→“权限”开启位置权限，即可进行定位。