



SUN YAT-SEN UNIVERSITY

中山大學



手机应用平台软件开发

10、网络访问



网络的重要性

SUN YAT-SEN UNIVERSITY

- 网络化信息化的世界
- 短信、电话
- 无线上网
- 移动，自由，随时随地





Android支持的通信模式

SUN YAT-SEN UNIVERSITY

➤ GSM

➤ BLUETOOTH

➤ EDGE

➤ NFC

➤ 3G

➤ ...

➤ WIFI



GSM

SUN YAT-SEN UNIVERSITY

全球移动通信系统

（**G**lobal **S**ystem for **M**obile Communications）当前应用最为广泛的移动电话标准。全球超过200个国家和地区超过10亿人正在使用GSM电话。



GSM

SUN YAT-SEN UNIVERSITY

- GSM标准的广泛使用使得在移动电话运营商之间签署“漫游协定”后用户的国际漫游变得很平常。
 - GSM较之它以前的标准最大的不同是它的信令和语音信道都是数字的，因此GSM被看作是第二代（2G）移动电话系统。GSM标准当前由3GPP组织负责制定和维护。
-



EDGE

SUN YAT-SEN UNIVERSITY

增强型数据速率GSM演进技术Enhanced Data Rate for GSM Evolution

- 从GSM到3G的过渡技术
 - 能够充分利用现有的GSM资源
 - 弹性优势
 - 工作在TDMA和GSM网络
 - 提高了GPRS信道编码效率及其高速移动数据标准
-



第三代移动通信技术（3rd-generation, 3G）

SUN YAT-SEN UNIVERSITY

- 3G就是指IMT-2000（International Mobile Telecommunications-2000），是国际电信联盟（ITU）定义的第三代无线通信的全球标准。
 - IMT-2000规定移动终端的连接速度
 - 以车速移动时 ——144Kbps
 - 室外静止或步行时 ——384Kbps
 - 室内 ——2Mbps
-



3G 目前存在的几种标准

SUN YAT-SEN UNIVERSITY

- WCDMA (Wideband CDMA) —— 欧洲
 - CDMA2000 —— 美国高通北美公司
 - TD-SCDMA (时分同步CDMA) —— 中国大陆
-



3G——应用

SUN YAT-SEN UNIVERSITY

- 宽带上网
 - 手机购物
 - 视频通话
 - 手机网游
 - 手机电视
 - 无线搜索
 - 手机音乐
-



WiFi

SUN YAT-SEN UNIVERSITY

Wireless Fidelity, 中文译为“无线兼容认证”

- 实质——一种商业认证
- 技术——短程无线传输
- 现状——带WiFi的便携式设备是潮流





WiFi

SUN YAT-SEN UNIVERSITY

常见的WiFi使用形式——无线路由器

- 覆盖范围——70至120米
 - 使用场合——公司、家庭、公共场所
 - 优点——方便的建立局域网、低成本、使用简单
-



WiFi——特点

SUN YAT-SEN UNIVERSITY

WiFi相比其他技术有如下特点:

- 无线电波的覆盖范围广
 - 传输速度高
 - 使用门槛比较低
 - 消除布线的麻烦
 - 发射功率低，健康安全
-



Bluetooth

SUN YAT-SEN UNIVERSITY



Bluetooth®

□ 定义:

1. 开放式无线通讯标准
2. 设备短距离互联解决方案

□ 优势:

1. 无需驱动程序——独特的配置文件
 2. 小型化无线电
 3. 低功率、低成本、安全性、稳固
 4. 易于使用、即时连接
-



□ 蓝牙协议栈

- 核心协议层（HCI、LMP、L2CAP、SDP）
 - 线缆替换协议层（RFCOMM）
 - 电话控制协议层（TCS-BIN）
 - 选用协议层（PPP、TCP、IP、UDP、OBEX、IrMC、WAP、WAE）
-



Bluetooth

SUN YAT-SEN UNIVERSITY

蓝牙规范（**profile**）——为了保证蓝牙设备的互通性而制定的一系列规范：

- 蓝牙立体声音频传输规范（A2DP）
 - 基本图像规范（BIP）
 - 基本打印规范（BPP）
 - 无线电话规范（CTP）
 - 蓝牙耳机规范（HP）
 - 文件传输规范（FTP）
 -
-



Near Field Communication (近场通讯)

SUN YAT-SEN UNIVERSITY

NFC由非接触式射频识别(RFID)及互联互通技术整合演变而来,在单一芯片上结合感应式读卡器、感应式卡片和点对点的功能,能在短距离内与兼容设备进行识别和数据交换。





NFC

SUN YAT-SEN UNIVERSITY

□ 技术优势:

- 轻松、安全、迅速的通信
- 传输范围小——独特的信号衰减技术
- 带宽高、能耗低

□ 应用场合:

- 门禁、公交
 - 手机支付
-



使用网络，应该有相应使用允许，在文件
AndroidManifest添加：

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

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



HTTP连接

SUN YAT-SEN UNIVERSITY

- 超文本传输协议(HTTP, HyperText Transfer Protocol)是互联网上应用最为广泛的一种网络协议。
 - 最常见的从网络传输数据的方式就是使用HTTP
 - HTTP可以封装几乎所有类型的数据
-



从Web读取数据

- 通过一个URL类来读取Web服务器上某个文件的定长部分
 - 适用于仅仅需要从一个Web站点读取一些数据的情况下，例如：当应用程序只需要一个轻量级、不太关键的网络特性
-



从Web读取数据

- 创建一个新的URL对象
- 为这个URL资源打开一个stream
- 读取数据
- 关闭InputStream

```
URL text = new URL("http://www.sysu.edu.cn");
InputStream isText = text.openStream();
byte[] bText = new byte[250];
int readSize = isText.read(bText);
Log.i("HTTP", "readSize = " + readSize);
Log.i("HTTP", "bText = " + new String(bText));
isText.close();
```



从Web读取数据

```
URL text = new URL("http://www.kf6nvr.net/st/index.html");
InputStream isText = text.openStream();
byte[] bText = new byte[250];
int readSize = isText.read(bText);
Log.i("HTTP", "readSize = " + readSize);
Log.i("HTTP", "bText = " + new String(bText));
isText.close();
```

■ LogCat观察输出结果

```
11-23 00:42... I 302 HTTP readSize = 250
11-23 00:42... I 302 HTTP bText = <html xmlns="http://www.w3.org/1999/xhtml" xml:1
```



从Web读取数据

- 这种方法虽简单，但并不严谨
 - 没有很好的错误处理：如手机没有网络、服务器关闭、URL无效、用户操作超时
 - 因此，从一个URL读取数据值之前，往往需要了解更多的信息，例如，需要读取的数据到底有多大
-



使用HttpURLConnection

- 对URL进行侦查，避免错误地传输过多的数据
 - HttpURLConnection获取一些有关URL对象所引用的资源信息
 - 如：HTTP状态、头信息、内容的长度、类型和日期时间等
-



使用HttpURLConnection

- 创建一个新的URL对象
- 为这个URL资源打开一个stream
- 读取数据
- 关闭InputStream

```
URL text = new
URL( "http://api.flickr.com/services/feeds/photos_public.gne?id=
      26648248@N04&lang=en-us&format=atom" );
HttpURLConnection http= (HttpURLConnection) text.openConnection();
Log.i( "HTTP", "length = " + http.getContentLength());
Log.i( "HTTP", "respCode = " + http.getResponseCode());
Log.i( "HTTP", "contentType = "+ http.getContentType());
Log.i( "HTTP", "content = "+http.getContent());
```



□ 使用HttpURLConnection

```
URL text = new
URL( "http://api.flickr.com/services/feeds/photos_public.gne?id=
      26648248@N04&lang=en-us&format=atom" );
HttpURLConnection http= (HttpURLConnection) text.openConnection();
Log.i( "HTTP", "respCode = " + http.getResponseCode());
Log.i( "HTTP", "contentType = "+ http.getContentType());
Log.i( "HTTP", "content = "+http.getContent());
```

➤ LogCat观察输出结果

```
11-23 00:36... I 302 HTTP respCode = 200
11-23 00:36... I 302 HTTP contentType = application/atom+xml; charset=utf-8
11-23 00:36... I 302 HTTP content = org.apache.harmony.luni.internal.net.www.proto..
```



解析从网络获取的XML

- 大部分网络资源的传输存储在一种结构化的形式“可拓展标记语言(Extensible Markup Language, XML)
 - Android提供了一种快速而高效的XML Pull Parse是网络应用程序解析器的首选
-



解析从网络获取的XML

1. START_TAG事件指定了一个XML标记的开始
 2. START_TAG: 找到一个新的标记时（<tag>）返回
 3. TEXT: 当找到文本时返回（即<tag>TEXT</tag>）
 4. END_TAG: 找到标记的结束时（</tag>）返回
 5. END_DOCUMENT: 当到达XML文件末尾时返回
-



解析从网络获取的XML

- 创建URL实例
- 从XmlPullParserFactory中获取一个XmlPullParser实例

```
URL text = new URL("http://.....");  
XmlPullParserFactory parserCreator=xmlPullParserFactory.newInstance();  
XmlPullParser parser = parserCreator.newPullParser();  
parser.setInput(text.openStream(), null);  
status.setText("Parsing...");
```



解析从网络获取的XML

- 若要寻找在<link>下，当rel="enclosure"、
type="image/*"，href中的值

```
<link rel="enclosure" type="image/jpeg"  
href="http://farm5.static.flickr.com/4126/4965796746_a9b9d6dd37_m.jpg" />
```



解析从网络获取的XML

```
<link rel="enclosure" type="image/jpeg"
href="http://farm5.static.flickr.com/4126/4965796746_a9b9d6dd37_m.jpg" />
```

```
int parserEvent = parser.getEventType();
while (parserEvent != XmlPullParser.END_DOCUMENT) {
    switch(parserEvent) {
        case XmlPullParser.START_TAG:
            String tag = parser.getName();
            if (tag.compareTo("link") == 0) {
                String relType = parser.getAttributeValue(null, "rel");
                if (relType.compareTo("enclosure") == 0) {
                    String encType = parser.getAttributeValue(null, "type");
                    if (encType.startsWith("image/")) {
                        String imageSrc = parser.getAttributeValue(null, "href");
                        Log.i("HTTP", "image source = " + imageSrc);
                    }
                }
            }
            break;
    }
    parserEvent = parser.next();
}
```

11-23 01:11... I 362 HTTP image source = http://farm5.static.flickr.com/4126/49657.



使用线程访问网络

- 之前所提及到的网络操作方式会造成UI线程阻塞，直到网络操作完成为止
 - 把一些耗时的操作从UI线程中移开，重新开启一个新的工作线程来执行这些任务，带给用户流畅的体验
-



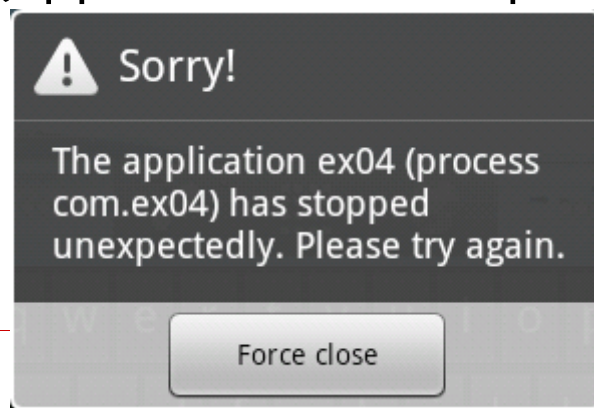
Android线程模型

- 当第一次启动一个Android程序时，Android会自动创建一个称为“main”主线程的线程。这个主线程(也称为UI线程) 负责把事件分派到相应的控件。
 - 例如，当你在屏幕上按下一个按钮后，UI线程会把这个事件分发给刚按得那个按钮，紧接着按钮设置它自身为被按下状态并向事件队列发送一个无效(invalidate)请求。UI线程会把这个请求移出事件队列并通知按钮在屏幕上重新绘制自身。
-



Android线程模型

- 单线程模型常常会引起Android应用程序性能低下，因为所有的任务都在同一个线程中执行，如果执行一些耗时的操作，如访问网络或查询数据库，会阻塞整个用户界面。
- 如果阻塞应用程序的时间过长(在Android系统中为5秒钟)，Android会向用户提示一些信息，即打开一个“应用程序没有相应(application not responding)”的对话框。





Android线程模型

- 因此，需要避免在UI线程中执行耗时的操作
- 在后台线程或工作者线程中执行这些耗时的任务是否可以呢？
- 请看以下代码：按钮的单击事件从网络上下载一副图片并使用ImageView来展现这幅图片。

```
public void onClick(View v) {  
    new Thread(new Runnable() {  
        public void run() {  
            Bitmap b = loadImageFromNetwork();  
            mImageView.setImageBitmap(b);  
        }  
    }).start();  
}
```



Android线程模型

- 这段代码好像很好地解决了遇到的问题，因为它不会阻塞UI线程。
- 然而运行时，Android会提示程序因为异常而终止。
- Why?!

```
public void onClick(View v) {  
    new Thread(new Runnable() {  
        public void run() {  
            Bitmap b = loadImageFromNetwork();  
            mImageView.setImageBitmap(b);  
        }  
    }).start();  
}
```



Android线程模型

- LogCat中打印的日志信息就会发现这样的错误日志：
`android.view.ViewRoot$CalledFromWrongThreadException: Only the original thread that created a view hierarchy can touch its views.`
 - 从错误信息不难看出Android禁止其他子线程来更新由UI thread创建的View。
 - 上述代码违背了Android单线程模型的原则：Android UI操作并不是线程安全的，并且**这些操作必须在UI线程中执行**。
-



Android线程模型

- 因此，Android提供了几种在其他线程中访问UI线程的方法。

Activity.runOnUiThread(Runnable)

View.post(Runnable)

View.postDelayed(Runnable, long)

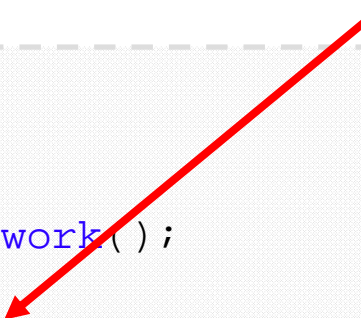
Handler



Android线程模型

- 使用这些类和方法中的任何一种纠正前面的代码示例
- 例如，把Runnable添加至消息队列，并由UI线程来处理

```
public void onClick(View v) {  
    new Thread(new Runnable() {  
        public void run() {  
            final Bitmap b = loadImageFromNetwork();  
            mImageView.post(new Runnable() {  
                public void run() {  
                    mImageView.setImageBitmap(b);  
                }  
            });  
        }  
    }).start();  
}
```





Android线程模型--- AsyncTask

- 在Android1.5中，Android.os包引入了一个新的类，称为AsyncTask
 - 它是一个抽象的辅助类，用来管理后台操作，并最终返回到UI线程。
 - 开发人员创建一个AsyncTask的子类并实现相应的事件方法，这与为后台处理创建线程并使用消息机制更新UI不同
-



Android线程模型--- AsyncTask

- doInBackground()方法会自动地在工作者线程中执行
 - onPreExecute()、onPostExecute()和onProgressUpdate()方法会在UI线程中被调用
 - doInBackground()方法的返回值会被传递给onPostExecute()方法
 - 在doInBackground()方法中你可以调用publishProgress()方法，每一次调用都会使UI线程执行一次onProgressUpdate()方法
-



Android线程模型--- AsyncTask

```
private class ImageLoader extends AsyncTask<URL, String, String>
{
    @Override
    protected String doInBackground(URL... params) {
        try {
            URL text = params[0];
            //执行代码, 如网络访问, 结果解析等
            publishProgress("Test");//调用onProgressUpdate
        } catch (Exception e) {
            Log.e("Net", "Failed", e);
            return "Finished with failure.";
        }
        return "Done...";
    }
}
```



Android线程模型--- AsyncTask

```
protected void onCancelled() {
    Log.e("Net", "Async task Cancelled");
}

protected void onPostExecute(String result) {
    mStatus.setText(result); //result是doInBackground中
                           //return的值,本例中为" Done..."
}

protected void onPreExecute() {
    mStatus.setText("About to load URL");
}

protected void onProgressUpdate(String... values) {
    mStatus.setText(values[0]); //values[0]为doInBackg中
                              //publishProgress(String)中的String
    super.onProgressUpdate(values);
}
```



使用线程访问网络

➤ 新建线程

```
new Thread() {  
    public void run() {  
        try {  
            //执行网络连接代码以及解析代码  
            mHandler.post(new Runnable() {  
                public void run() {  
                    //把有关用户界面更新的内容提交回主线程  
                }  
            });  
        } catch (Exception e) { //异常处理  
        }  
    }.start();  
}
```



使用线程访问网络

```
new Thread() {  
    public void run() {  
        try {  
            //执行网络连接代码 参考PPT P13  
            mHandler.post(new Runnable() {  
                public void run() {  
                    status.setText("Parsing...");  
                }  
            });  
            //执行解析代码 参考PPT P15  
            mHandler.post(new Runnable() {  
                public void run() {  
                    status.setText("Done...");  
                }  
            });  
        } catch (Exception e) {  
            //异常处理  
        }  
    }  
}.start();
```



使用线程访问网络

```
new Thread() {
    public void run() {
        try {
            //执行网络连接代码 参考PPT P13
            mHandler.post(new Runnable() {
                public void run() {
                    status.setText("Parsing...");
                }
            });
            //执行解析代码 参考PPT P15
            mHandler.post(new Runnable() {
                public void run() {
                    status.setText("Done...");
                }
            });
        } catch (Exception e) {
            //异常处理
        }
    }
}.start();
```




Android Wifi开发

SUN YAT-SEN UNIVERSITY

Android SDK提供的相关包：android.net.wifi

ScanResult

用于描述一个已经被检测到的wifi接入点。

WifiConfiguration

该类代表了一个已经配置好的wifi网络，包括了该网络的一些安全设置。例如接入点密码，接入点通讯所采用的安全标准。

WifiConfiguration. AuthAlgorithm

公认的IEEE 802.11标准认证算法。



Android Wifi相关类介绍

SUN YAT-SEN UNIVERSITY

WifiConfiguration.GroupCipher 公认的组密码。

WifiConfiguration.KeyMgmt 公认的密钥管理方案。

WifiConfiguration.PairwiseCipher 公认的用于WPA的成对密码标准。

WifiConfiguration.Protocol 公认的安全协议

WifiConfiguration.Status 网络所可能存在的状态。



Android Wifi相关类介绍

SUN YAT-SEN UNIVERSITY

WifiInfo

描述了各个wifi连接的状态，该连接是否处于活动状态或者是否处于识别过程中。

WifiManager

这个类比较重要。它提供了用于管理wifi连接的各种主要API。详见表后说明。

WifiManager.MulticastLock

允许应用程序接收wifi的多播数据包。

WifiManager.WifiLock

允许应用程序永久地保持wifi连接（防止系统自动回收）。



Android Wifi开发

SUN YAT-SEN UNIVERSITY

Android 操作WiFi的重要类——WifiManager，这个类提供了最主要的用于管理wifi连接的API。通过调用Context.getSystemService(Context.WIFI_SERVICE)方法来得到系统提供的WifiManager，代码如下：

```
WifiManager mWifiManager = (WifiManager)  
context.getSystemService(Context.WIFI_SERVICE);
```



WifiManager主要有如下功能:

- 已经配置好的网络连接列表。这个列表可以被用户查看或者更新，而且可以通过它来修改个别接入点的属性；
 - 如果当前有连接存在的话，可以得到当前正处于活动状态的wifi连接的控制权，可以通过它建立或者断开连接，并且可以查询该网络连接的动态信息；
 - 通过对已经扫描到的接入点的足够信息来进行判断，得出一个最好的接入点进行连接。
 - 定义了很多用于系统广播通知的常量，它们分别代表了WiFi状态的改变。
-



Android Wifi开发

SUN YAT-SEN UNIVERSITY

Android网络连接管理类——ConnectivityManager，该类用于管理抽象意义上的“网络连接”，它的主要作用是：

- 监控网络连接（包括WiFi，GPRS，UMTS等等）；
 - 当网络连接发生改变时，向系统广播这一改变；
 - 当失去了当前的网络连接时，尝试切换到另外一个连接；
 - 提供了允许其他应用程序调用的API让应用程序可以方便地查询当前的网络状态。
-



Android Wifi权限获取

SUN YAT-SEN UNIVERSITY

要在应用程序中对Android系统的WiFi设备进行相关操作，需要在项目中的AndroidManifest.xml中选择性地添加如下几句用于声明权限的语句：

```
<uses-permission  
android:name="android.permission.ACCESS_WIFI_STATE">  
</uses-permission>  
<uses-permission  
android:name="android.permission.ACCESS_CHECKIN_PROPERTIES">  
</uses-permission>  
<uses-permission  
android:name="android.permission.WAKE_LOCK"></uses-permission>  
<uses-permission  
android:name="android.permission.CHANGE_WIFI_STATE">  
</uses-permission>
```



Android Wifi开发——代码示例

SUN YAT-SEN UNIVERSITY

```
//取得WifiManager对象
mWifiManager = (WifiManager)
context.getSystemService(Context.WIFI_SERVICE);
//取得WifiInfo对象
mWifiInfo = mWifiManager.getConnectionInfo();
//打开WIFI
public void openWifi()
{
    if (!mWifiManager.isWifiEnabled())
    {
        mWifiManager.setWifiEnabled(true);
    }
}
```



Android Wifi开发——代码示例

SUN YAT-SEN UNIVERSITY

//关闭WIFI

```
public void closeWifi()
```

```
{
```

```
    if (!mWifiManager.isWifiEnabled())
```

```
    {
```

```
        mWifiManager.setWifiEnabled(false);
```

```
    }
```

```
}
```

//得到WifiLock, 以便应用程序保持wifi连接

```
public void acquireWifiLock()
```

```
{
```

```
    mWifiLock.acquire();
```

```
}
```



Android Wifi开发——代码示例

SUN YAT-SEN UNIVERSITY

//解锁WifiLock

```
public void releaseWifiLock()  
{
```

```
    if (mWifiLock.isHeld()) //判断是否被锁定  
    {  
        mWifiLock.acquire();  
    }
```

```
}
```

//创建WifiLock

```
public void creatWifiLock()  
{
```

```
    mWifiLock = WifiManager.createWifiLock("Lock");  
}
```




Android Wifi开发——代码示例

SUN YAT-SEN UNIVERSITY

//得到已经配置好的网络列表

```
public List<WifiConfiguration> getConfiguration()  
{  
    return mWifiConfiguration;  
}
```

//选择一个已配置好的网络进行连接

```
public void connectConfiguration(int index)  
{  
    //索引大于配置好的网络索引返回  
    if(index > mWifiConfiguration.size())  
    {  
        return;  
    }  
    //连接配置好的指定ID的网络  
    mWifiManager.enableNetwork(mWifiConfiguration.get(index)  
        .networkId, true);  
}
```



Android Wifi开发——代码示例

SUN YAT-SEN UNIVERSITY

//扫描接入点

```
public void startScan()
```

```
{
```

```
    mWifiManager.startScan();
```

//得到扫描结果

```
    mWifiList = mWifiManager.getScanResults();
```

//得到已经配置好的网络列表

```
    mWifiConfiguration =
```

```
    mWifiManager.getConfiguredNetworks();
```

```
}
```

//得到网络连接列表

```
public List<ScanResult> getWifiList()
```

```
{
```

```
    return mWifiList;
```

```
}
```



Android 蓝牙开发

SUN YAT-SEN UNIVERSITY

□ Android SDK提供的相关包： `android.bluetooth`

□ API主要为应用程序提供如下几个功能：

1. 搜寻有效范围内的蓝牙设备；
 2. 通过本地的蓝牙适配器来查询到与之配对的蓝牙设备；
 3. 在配对的蓝牙设备之间建立RFCOMM信道；
 4. 连接到其他设备的指定端口；
 5. 在设备之间传输数据。
-



Android 蓝牙API

SUN YAT-SEN UNIVERSITY

❑ android.bluetooth包括了以下两个接口：

| 接口名 | 描 述 |
|---|--|
| BluetoothProfile | 蓝牙规范的公用API接口，所有的蓝牙规范都必须实现这个接口。Profile目的是要确保Bluetooth设备间的互通性。 |
| BluetoothProfile.ServiceListener | 用于在蓝牙客户设备连接或者断开连接时给它们发出通知的接口。 |



Android 蓝牙API

SUN YAT-SEN UNIVERSITY

android.bluetooth包括了以下一些类:

| 类 名 | 描 述 |
|--------------------------|--|
| BluetoothA2dp | 这个类作为对BluetoothProfile接口实现的实例，这是对蓝牙的A2DP规范的API实现类。 |
| BluetoothAdapter | 代表了本地的蓝牙适配器。 |
| BluetoothAssignedNumbers | 蓝牙的指令编号。 |
| BluetoothClass | 代表蓝牙的类，这个类描述了蓝牙设备的特征和性能参数。 |



Android 蓝牙API

SUN YAT-SEN UNIVERSITY

| 类 名 | 描 述 |
|-----------------------------|-----------------------|
| BluetoothClass.Device | 定义了所有的device类所用的常量。 |
| BluetoothClass.Device.Major | 定义了所有主要的device类所用的常量。 |
| BluetoothClass.Service | 定义了所有的service类所用的常量。 |
| BluetoothHeadset | 实现蓝牙耳机服务的公共API。 |



android.bluetooth中用于建立连接的类

| 类 名 | 描 述 |
|------------------------------|----------------------|
| BluetoothServerSocket | 用于监听socket连接请求的类。 |
| BluetoothSocket | 一个已连接的或正在连接的socket类。 |

类似于Java API中的ServerSocket和Socket类



Android 蓝牙权限获取

SUN YAT-SEN UNIVERSITY

要在应用程序中对Android系统的蓝牙设备进行相关操作，需要在项目中的AndroidManifest.xml中添加：

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

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

Questions?



ANDROID