网络编程在整个Android开发是非常重要的, 因为严格意义上来说目前为止很少出现所谓的单击系统. 即使有单击系统, 系统的升级本身也是通过网络的实现. 因此学会Android网络开发的重要性不言而喻

单击按钮, 下载互联网图片代码, 此代码在Android4.0以下的版本运行正常

- 此代码要注意添加互联网访问权限：<uses-permission android:name="android.permission.INTERNET"/>
- android已经提供了文件处理的API Bitmap,来处理相关文件流

```
1.   public void downImg(View v) throws Exception {
2.     URL url = new URL("https://www.baidu.com/img/bd_logo1.png");
3.     // 根据URL地址,创建连接对象,注意此时还没有打开连接
4.     HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();
5.     // 对连接进行初始化设置
6.     urlConn.setRequestMethod("GET");
7.     // 设置连接超时时间
8.     urlConn.setConnectTimeout(5000);
9.     // 设置读取超时时间
10.    urlConn.setReadTimeout(5000);
11.    // 发送连接,与服务器建立连接
12.    urlConn.connect();
13.    // 如果响应吗为200,说明请求成功
14.    if (urlConn.getResponseCode() == 200) {
15.     // 获取服务器响应头中的流,流里面的数据就是客户请求的数据
16.     InputStream inputStream = urlConn.getInputStream();
17.     // 读取出流里的数据,并构造成位图对象
18.     Bitmap bitmap = BitmapFactory.decodeStream(inputStream);
19.     ImageView iv = (ImageView) findViewById(R.id.img);
20.     // 把位图对象显示至imageView
21.     iv.setImageBitmap(bitmap);
22.    }
23.   }
```

此代码在 Android4.0以上会出现如下异常信息：Caused by: android.os.NetworkOnMainThreadException

在Android4.0以上的版本. 网络请求不能再主线程中完成. 因为这样会导致用户体验变差, 因此我们需要创建一个子线程完成此操作

- 在此处可以先讲守护线程的概念

```
1.   private class DownImageThread extends Thread {
2.
3.     @Override
4.     public void run() {
5.      URL url;
6.      try {
7.       url = new URL("http://s0.hao123img.com/res/img/jd20151111.jpg");
8.       // 根据URL地址,创建连接对象,注意此时还没有打开连接
9.       HttpURLConnection urlConn = (HttpURLConnection) url
10.        .openConnection();
11.      // 对连接进行初始化设置
12.      urlConn.setRequestMethod("GET");
13.      // 设置连接超时时间
14.      urlConn.setConnectTimeout(5000);
15.      // 设置读取超时时间
16.      urlConn.setReadTimeout(5000);
17.      // 发送连接,与服务器建立连接
18.      urlConn.connect();
19.      // 如果响应吗为200,说明请求成功
20.      if (urlConn.getResponseCode() == 200) {
21.       // 获取服务器响应头中的流,流里面的数据就是客户请求的数据
22.       InputStream inputStream = urlConn.getInputStream();
23.       // 读取出流里的数据,并构造成位图对象
24.       Bitmap bitmap = BitmapFactory.decodeStream(inputStream);
25.       ImageView iv = (ImageView) findViewById(R.id.img);
26.       // 把位图对象显示至imageView
```

```
27.         iv.setImageBitmap(bitmap);
28.        }
29.       } catch (Exception e) {
30.        throw new RuntimeException(e);
31.       }
32.      }
33.
34.     }
35.
36.     public void downImg(View v) throws Exception {
37.      new DownImageThread().start();
38.     }
```

但是测试的时候会发现, 出现如下异常：主线程下载图片可能会导致堵塞问题, 因此在Android4.0以上版本对它进行优化

```
1.    Caused by: android.view.ViewRootImpl$CalledFromWrongThreadException: Only the ori
      ginal thread that created a view hierarchy can touch its views.
```

主线程堵塞的代码测试：(如果主线程堵塞,则其它的按钮则不能被操作, 仿佛死机一般这样用户体验不是很好)

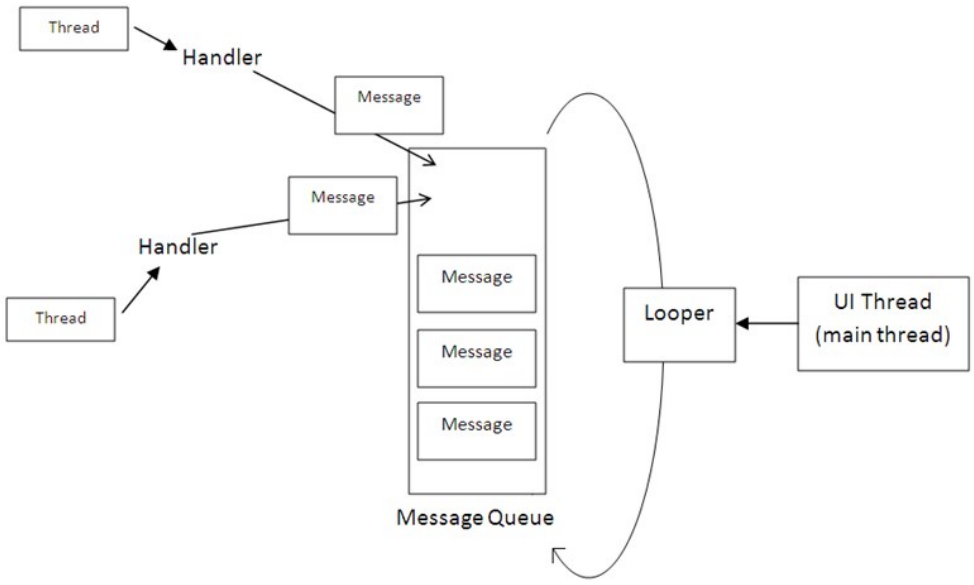
```
1.    public void btn_down(View view) throws Exception {
2.     Thread.sleep(3000);
3.    }
```

大概的意思是：只有创建UI的线程才能绘制UI. 这个比较好理解, 如果子线程可以修改UI或者View状态的话, 在游戏中很多的时候会出现线程安全的问题
出问题的代码就
是：`ImageView iv = (ImageView) findViewById(R.id.img); 此行代码已经放到了自定义的线程中, 而findViewByID是属于Activity`的
总结如下：

- 通过案例了解守护线程的概念
- Android中的主线程又称UI线程, 因为只有在主线程中,才能刷新UI操作
- UI停止刷新, 应用程序无法响应用户操作
- 所有耗时的操作都不应该在主线程进行

解决方案如下：Android消息队列机制：(主要原理是线程通信机制)

- 主线程创建, 系统会同时创建消息队列队对象(MessageQueue) 和消息轮训对象
- 轮询器的作用, 就是不停的检测消息队列中是否有消息(Message)
- 消息队列一旦有消息, 轮询器就会把消息对象传递给消息处理器(Handler) 处理器会调用handlerMessage方法来处理这条消息.



采用消息队列机制解决图片下载问题：

首先在MainActivity 中创建Handler, 因为我们知道UI的绘制需要在主线程中执行

```
1.  public class MainActivity extends Activity {
2.
3.   private Handler handler = new Handler() {
4.    @Override
5.    public void handleMessage(Message msg) {
6.     ImageView iv = (ImageView) findViewById(R.id.img_view);
7.     iv.setImageBitmap((Bitmap) msg.obj);
8.    }
9.
10.   };
11.
12.   @Override
13.   protected void onCreate(Bundle savedInstanceState) {
14.    super.onCreate(savedInstanceState);
15.    setContentView(R.layout.activity_main);
16.    this.auto_data();
17.    // 每一个主线程都有一个唯一的Looper,每一个Looper都有一个MessageQueue
18.    Log.i("demo", Looper.myQueue() + "," + Looper.getMainLooper());
19.   }
20.  }
```

在子线程中传入handler对象, 一旦读取下载完毕,可以通过message对象来进行消息通信

```
1.  public void btn_down(View view) {
2.    // 获取下载图片的地址
3.    new Thread() {
4.     @Override
5.     public void run() {
6.      try {
7.       URL uri = new URL("https://www.baidu.com/img/bd_logo1.png");
8.       HttpsURLConnection httpUri = (HttpsURLConnection) uri
9.         .openConnection();
10.       httpUri.setRequestMethod("GET");
11.       httpUri.setConnectTimeout(5000);
12.       // 与服务器建立连接
13.       httpUri.connect();
14.       if (httpUri.getResponseCode() == 200) {
15.        InputStream stream = httpUri.getInputStream();
16.        Bitmap bitmap = BitmapFactory.decodeStream(stream);
17.        Message message = handler.obtainMessage();
18.        message.obj = bitmap;
```

```
19.          handler.sendMessage(message);
20.         }
21.        } catch (Exception e) {
22.         e.printStackTrace();
23.         throw new RuntimeException();
24.        }
25.       }
26.
27.      }.start();
28.     }
```