2. Android线程机制

- 使用线程访问网络

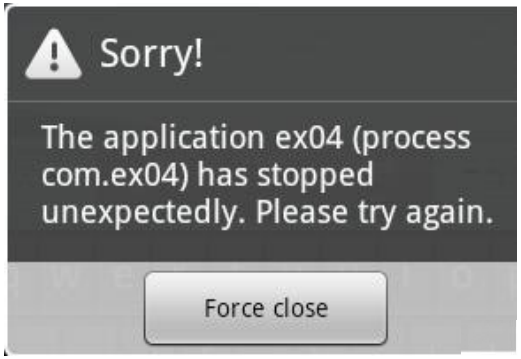
1. 之前所提及到的网络操作方式会造成UI线程阻塞，直到网络操作完成为止
2. 如果在主线程中直接使用HTTP同步访问， AS会抛出 android.os.NetworkOnMainThreadException
3. 把一些耗时的操作从UI线程中移开，重新开启一个新的工作线程来执行这些任务,带给用户流畅的体验

2. Android线程机制

- Android线程模型

1. 单线程模型常常会引起Android应用程序性能低下。如果在单线程下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。

2. 如果阻塞应用程序的时间过长(在Android系统中为5秒钟)，Android会向用户提示如下信息



2. Android线程机制

- Android线程模型

1. 因此需要避免在UI线程中执行耗时的操作

2. 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();
    }
```

2. Android线程机制

- Android线程模型

1. 上页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?

2. 原因是代码违背了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。

3. 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.

2. Android线程机制

- Android线程模型

1. 因此，Andriod提供了几种在其他线程中访问UI线程的方法。

    - Handler
    - AsyncTask
    - RxJava

2. Android线程机制

- Android线程模型

1. 使用上页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();
}
```

2. Android线程机制

- Android线程模型--- AsyncTask

1. AsyncTask Example

```
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...";
    }
    (接下页)
```

2. Android线程机制

- 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中的
                                //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);
    }
}
```

2. Android线程机制

- 使用线程访问网络

1. 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();
```

2. Android线程机制

- 使用线程访问网络

1. 详细代码示例

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