河北师范大学软件学院
Software College of Hebei Normal University

Android 基础开发

第三章 第二节
Handler

教学目标

- 掌握在Android中多线程的使用方法

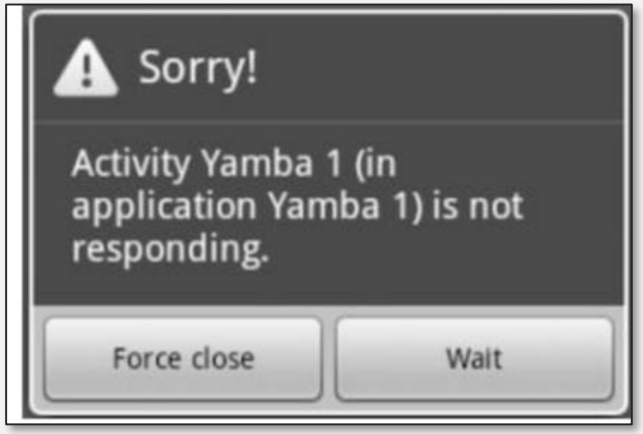
目录

Android中的多线程问题

- Android用户界面是与用户交互的接口，对于用户的操作，Android迅速响应用户输入（200ms内）是一个重要目标。
  - 如果Activity中的应用程序在5s之内未做出响应，可能会出现"应用程序无响应，是否关闭？"的对话框。
  - 若用户界面长时间对用户的操作未作出响应，那么这样的应用程序，肯定不会受到用户的广泛青睐。
  - 此类问题案例很多：如后台下载、异步加载图片等等。
- 对于这类耗时比较多的工作，一般是使用多线程的方法来解决的。

ANR错误页面

- 在Android上，如果你的应用程序有一段时间响应不够灵敏，系统会向用户显示一个对话框，这个对话框称作应用程序无响应（ANR：Application Not Responding）对话框。



Android中的多线程问题

- 模拟耗时工作
  - 当耗时工作在执行时，检验用户当前是否可以执行其它操作？
    - ✓点击其它按钮是否有反应？
    - ✓执行其它视图元素绑定的事件是否有反应？
  - Logcat提示：
    Android too much work in main thread.

Android中的多线程问题

- Android应用中的主线程（UI线程）：
  - Android应用刚启动时，会在当前应用所对应的进程中启动一个主线程（也叫UI线程）；
  - 该UI线程处理与UI相关的事件，如：用户的按键事件，把相关的事件分派到对应的组件进行处理等。
- 对于UI线程中比较耗时的工作，开启一个子线程来处理这些工作：
  - 首先创建一个Thread对象，然后调用start( )方法启动新的子线程。

Android中的多线程问题

- 使用子线程解决异步执行带来的新问题：
  - 在Android中，只有UI线程（即主线程）才可以更新主UI界面，而其子线程不能更新UI视图。
- 对于这类既需要异步执行，又需要更新UI界面的问题，Android提供了多种解决方案：
  - 使用多线程实现：Thread+Handler

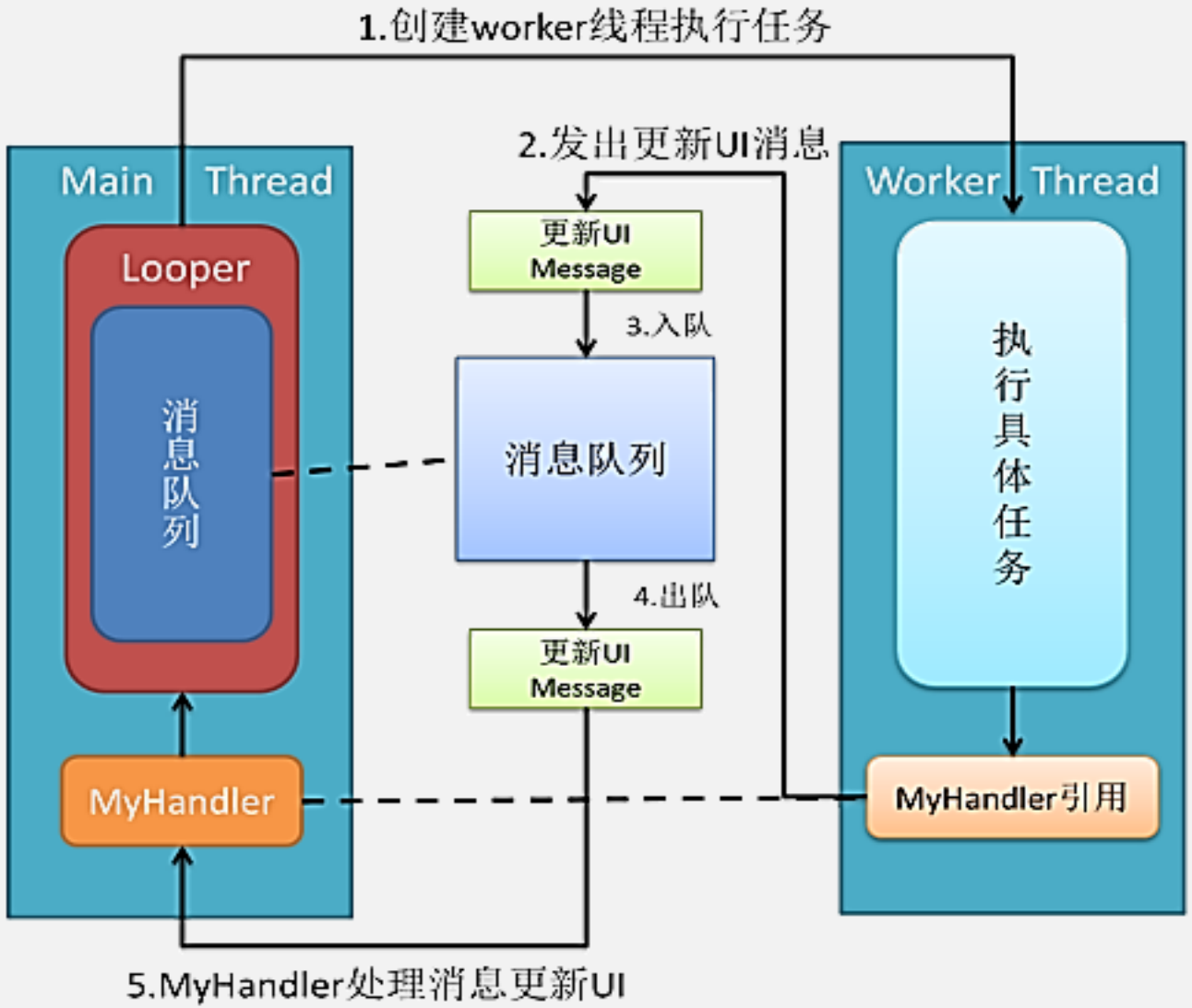
    http://www.cnblogs.com/playing/archive/2011/03/24/1993583.html
  - 使用AsyncTask实现。

目录

Handler简介

- Handler：接受子线程发送的数据，并用此数据配合主线程更新UI。
  - Handler定义在主线程中（UI线程中）；
  - Handler充当主线程和子线程之间交互的中介：
    - Handler在新启动的子线程中发送消息；
    - Handler在主线程中获取并处理子线程所发送的消息。

Handler的使用方法



重要的类

- UI线程：创建UI线程时，初始化一个Looper对象以及与其关联的MessageQueue；
- Handler：发送与处理信息，在当前线程中有一个Looper对象；
- Message：Handler接收与处理的消息对象；
- MessageQueue：消息队列，管理Message；
- Looper：管理MessageQueue, 取出Message分发给对应的Handler处理，每个线程只有一个Looper。

Handler的使用方法

- 使用Handler实现主线程与子线程的通信主要使用如下三个类：
  - Message类：发送带有附加参数的消息，其处理方法由handlerMessage()方法处理。
  - Handler：MQ上添加消息和处理消息，通知MQ它要执行一个任务(sendMessage)，并在loop到自己的时候执行该任务(handleMessage)，整个过程是异步的。
  - Looper：循环工作的线程。
    - Message Queue

Handler的使用方法

- Handler类的常用方法如下表所列。

方法名	描述
post(Runnable r)	添加 r 到 Handler 的消息队列中，等待发送
postDelayed(Runnable r, long mSec)	延迟 mSec 毫秒后，添加 r 对象到 Handler 消息队列中，等待发送
removeCallbacks(Runnable r)	取消消息队列中未发送的 r 对象
sendMessage(Message msg)	添加消息对象 msg 到 Handler 的消息队列中，等待发送
sendMessageDelayed(Message msg, long mSec)	延迟 mSec 毫秒，添加 msg 对象到 Handler 消息队列中，等待发送
removeCallbacks(Message msg)	取消消息队列中未发送的 msg 对象
handleMessage(Message msg)	处理消息队列中当前的一条 msg 消息

– 详细信息请参考：

http://developer.android.com/reference/android/os/Handler.html#pubmethods

使用Message消息解决多线程问题

- 基本流程：
  - 创建Handler，并添加handleMessage方法。
    - ✓使用自定义的匿名子类的方法创建Handler对象，并重写handleMessage方法实现消息的处理。

```
Handler handler = new Handler() {
    public void handleMessage (Message msg) {
        switch (msg.what) {
            case MSG_CURRENT: // TODO
            break;
        }
    }
};
```

使用Message消息解决多线程问题

- 基本流程：
  - 创建Thread对象，在Thread对象的run方法中发送消息。
    - 使用自定义的匿名子类的方法创建Thread对象，并重写run方法实现消息的参数设置和添加到消息队列中等操作。

```
Thread backgroundThread = new Thread() {
    public void run() {
        Message msg = handler.obtainMessage();
        msg.what = MSG_CURRENT;
        handler.sendMessage(msg);
    }
};
```

使用Message消息解决多线程问题

- 基本流程：
  - 启动Thread对象：

```
backgroundThread.start();
```

    - Message对象的常用方法和属性参考：

      http://developer.android.com/reference/android/os/Message.html

典型应用

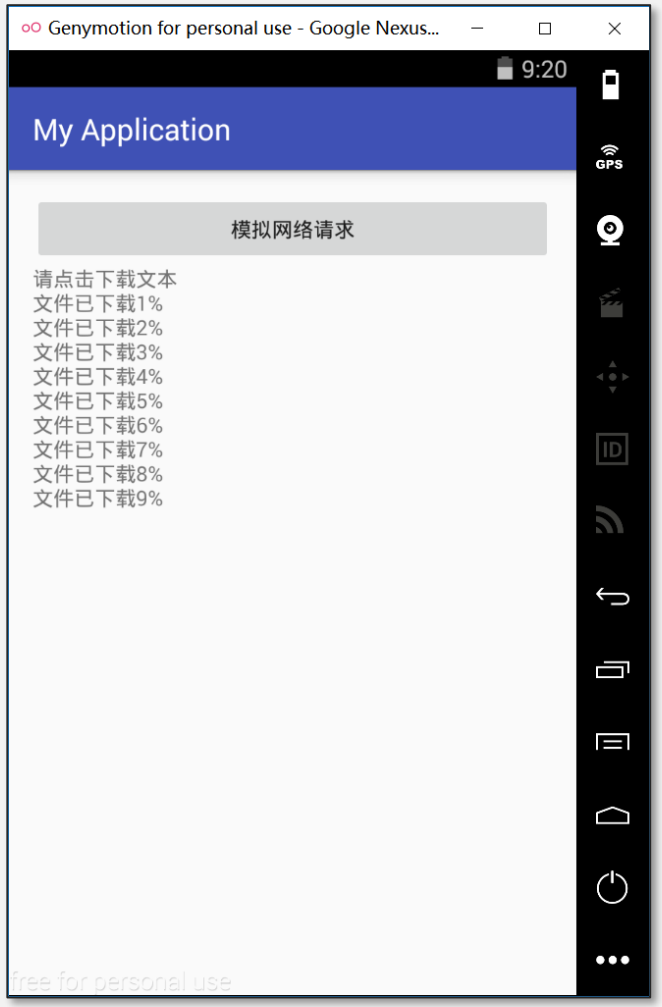
- 模拟图片的异步加载。
  - 在一个GridView视图中，实现图片的异步加载功能。
  - 图片地址来源于一个字符串数组。

典型应用

- 模拟微博的“更多”功能。
  - 在一个ListView视图中，模拟微博客户端的“更多”功能。
  - 当用户向上滚动时，显示更多的记录信息。
  - 记录信息存在在字符串数组中。

课堂练习


Genymotion for personal use - Google Nexus...
9:13
My Application
模拟网络请求
文件已下载9%
free for personal use


Genymotion for personal use - Google Nexus...
9:20
My Application
模拟网络请求
请点击下载文本
文件已下载1%
文件已下载2%
文件已下载3%
文件已下载4%
文件已下载5%
文件已下载6%
文件已下载7%
文件已下载8%
文件已下载9%
free for personal use


Genymotion for personal use - Google Nexus...
9:27
My Application
点击下载文件1:
点击下载文件2:
请点击下载文本
文件1已下载1%
文件2已下载1%
文件1已下载2%
文件2已下载2%
文件2已下载3%
文件1已下载3%
文件2已下载4%
文件2已下载5%
文件1已下载4%
文件2已下载6%
文件2已下载7%
文件1已下载5%
文件2已下载8%
文件2已下载9%
文件1已下载6%
文件1已下载7%
文件1已下载8%
文件1已下载9%
free for personal use

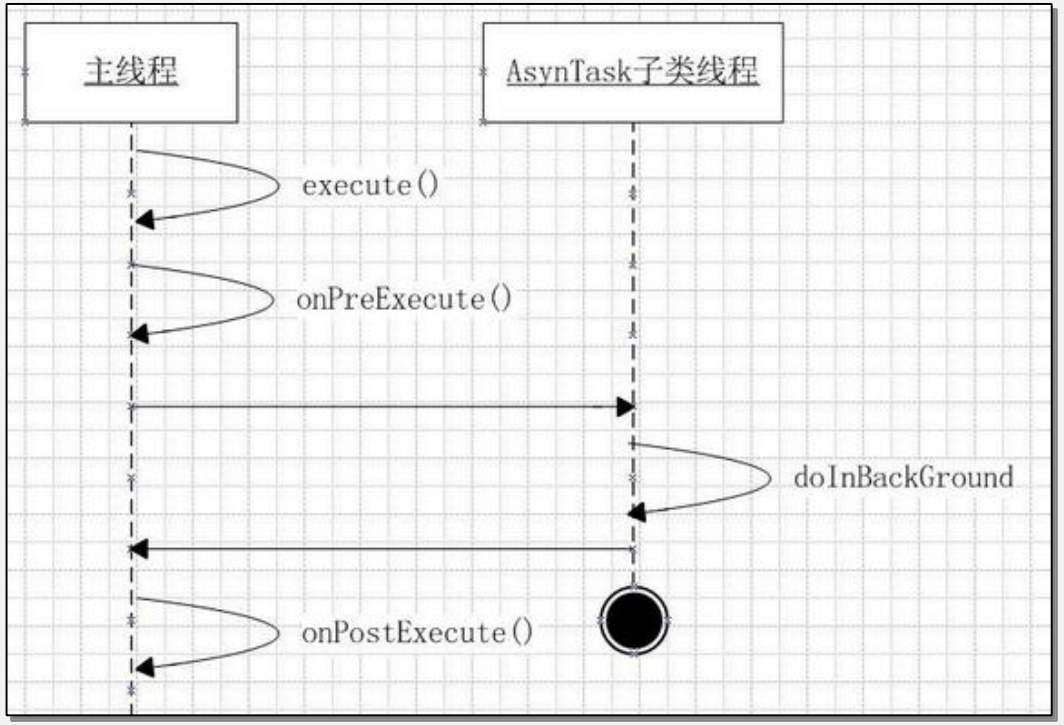
课堂练习


Genymotion for personal use - Google Nexus...
9:45
My Application
请输入用户名：
请输入密码：
登录
正在网络请求数据
Btn1 click!
free for personal use

目录

AsyncTask（异步任务）

- AsyncTask能够适当地、简单地用于UI线程，这个类准许执行后台操作。
- 异步任务的定义是一个在后台线程上运行，其结果是在UI线程上显示的一种类。



AsyncTask的四个方法

- 异步任务执行时对应的四个方法：
  - onPreExecute()：任务被执行之前调用UI线程。
    - 这步通常被用于设置任务，例如在用户界面显示一个进度条。
  - doInBackground(Params...)：onPreExecute()执行完成，立刻调用后台线程。
    - 这步被用于执行较长时间的后台任务；
    - 异步任务的参数也被传到这步；
    - 计算的结果在这步返回，将传回到上一步。

AsyncTask的四个方法

- onProgressUpdate(Progress...)：一次呼叫publishProgress(Progress...)后调用UI线程，执行的时机没有定义。
  - 这个方法用于在用户界面显示进度，当后台计算还在进行时。
  - 例如：这个方法可以被用于一个进度条动画或在文本域显示日志。
- onPostExecute(Result)：当后台计算结束时，调用UI线程。
  - 后台计算结果作为一个参数传递到此方法中。

AsyncTask的创建

- 异步任务的创建：
  - AsyncTask必须被子类继承。
    - 子类至少重写其中的doInBackground(Params...)方法，一般还会重写onPostExecute(Result)。
  - 任务实例必须创建在UI线程。
  - execute(Prams...)必须在UI线程上调用。
  - 不要手动调用onPreExecute(), onPostExecute(), doInBackground(), onProgressUpdate()。
  - 不能在doInBackground(Params... params)中更改UI界面。

AsyncTask的创建

- 异步任务中的三个泛型：
  - Params：任务执行器需要的数据类型。
  - Progress：后台计算中使用的进度单位数据类型。
  - Result：后台计算返回结果的数据类型。

AsyncTask的示例

内容回顾

1. Android中的多线程问题
2. 使用Handler处理多线程问题
3. 使用AsyncTask处理多线程问题

Thank you!