Android消息机制-Handler(中篇)

Dec 27, 2015

- HandlerThread
  - 1. 创建
  - 2. getLooper
  - 3. 运行
  - 4. 退出
  - 用法

> 本文基于Android 6.0的源代码，来分析Handler的用法

相关源码

```
framework/base/core/java/andorid/os/HandlerThread.java
```

HandlerThread

Handler的用法，往往是在一个线程中运行Looper，其他线程通过Handler来发送消息到Looper所在线程，这里涉及线程间的通信。既然涉及多个线程的通信，会有同步的问题，Android对此直接提供了HandlerThread类，下面来讲讲HandlerThread类的设计。

1. 创建

HandlerThread 继承于 Thread类

```
public HandlerThread(String name) {
    super(name);
    mPriority = Process.THREAD_PRIORITY_DEFAULT; //默认优先级
}

public HandlerThread(String name, int priority) {
    super(name);
    mPriority = priority;
}
```

2. getLooper

获取HandlerThread线程中的Looper对象

```
public Looper getLooper() {
    // 当线程没有启动或者已经结束时，则返回null
    if (!isAlive()) {
        return null;
    }

    //当线程已经启动，则等待直到Looper创建完成
    synchronized (this) {
        while (isAlive() && mLooper == null) {
            try {
                wait(); //休眠等待
            } catch (InterruptedException e) {
            }
        }
    }
    return mLooper;
}
```

3. 运行

```
@Override
public void run() {
    mTid = Process.myTid();  //获取线程的tid
    Looper.prepare();   // 创建Looper对象
    synchronized (this) {
        mLooper = Looper.myLooper(); //获取Looper对象
        notifyAll(); //唤醒等待线程
    }
    Process.setThreadPriority(mPriority);
    onLooperPrepared();  // 该方法可通过覆写，实现自己的逻辑
    Looper.loop();   //进入循环模式
    mTid = -1;
}
```

4. 退出

```
public boolean quit() {
    Looper looper = getLooper();
    if (looper != null) {
        looper.quit(); //普通退出
        return true;
    }
    return false;
}

public boolean quitSafely() {
    Looper looper = getLooper();
    if (looper != null) {
        looper.quitSafely(); //安全退出
        return true;
    }
    return false;
}
```

quit()与quitSafely()的区别，仅仅在于是否移除当前正在处理的消息。移除当前正在处理的消息可能会出现不安全的行为。

用法

很多时候，在HandlerThread线程中运行Loop()方法，在其他线程中通过Handler发送消息到HandlerThread线程。 通过wait/notifyAll的方式，有效地解决了多线程的同步问题。

下面再来讲解用法：

1. 创建并启动HandlerThread线程，内部包含Looper

   ```
   HandlerThread handlerThread = new HandlerThread("yuanhh.com");
   handlerThread.start()
   ```

2. 创建Handler

   ```
   Handler handler = new Handler(handlerThread.getLooper());
   ```

3. 发送消息

   handler.post(new Runnable() {

```
    @Override
    public void run() {
        System.out.println("thread id="+Thread.currentThread().g
etId());
        handler.sendEmptyMessage(10);
    }
 });
```

喜欢

0条评论 最新 最早 最热

还没有评论，沙发等你来抢

嘿嘿参北斗哇 (http://www.baidu.com/p/嘿嘿参北斗哇) 帐号管理


(http://duoshuo.com/settings/avatar/)

说点什么吧...

分享到: 发布

多说 (http://duoshuo.com)

gityuan@gmail.com (mailto:gityuan@gmail.com) · Github

(https://github.com/yuanhuihui) · 天道酬勤 · © 2015 Yuanhh · Jekyll

(https://github.com/jekyll/jekyll) theme by HyG (https://github.com/Gaohaoyang)