Android Handler事件分析处理

|  |  |  |
| --- | --- | --- |
| 版本号 | 描述 | 日期(更新日期)/更新人 |
| 1.0 | Android Handler事件分析 | 2018/08/07 刘志保 |
|  |  |  |

Handler在android中地位很重要,下面梳理一下这个知识点.

预备知识 : ThreadLocal使用方式.

<1> 涉及的对象

Handler

Looper

Message

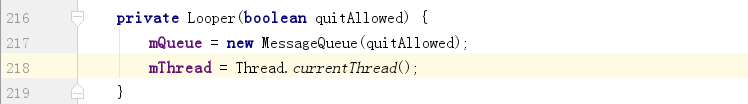
MessageQueue

ThreadLocal

Thread

<2> 根据上面从最小的单元开始介绍:

Thread : 这个就是java的用于开辟线程的类,但是在Looper是特指当前线程,并不是新开辟的.参考Looper.java源码:



<a> : Message : 这是消息传送的载体数据结构体,参考Message.java源码

**public int what**;

**public int arg1**;

**public int arg2**;

**public** Object **obj**;

Handler **target**;

Runnable **callback**;

其中前面四个参数很容易理解,即Handler调用下面的方法

**public final** Message obtainMessage(**int** what, **int** arg1, **int** arg2, Object obj)

在Handler调用这个接口时,即把这四个参数传给Message数据体.另外两个如下:

**public static** Message obtain(Handler h, Runnable callback) {  
 Message m = *obtain*();  
 m.target = h;  
 m.callback = callback;  
  
 **return** m;  
}

这里注意,如果传递了callback参数,那么回调就会回调这个,而不会HandleMessage.

<b> : MessageQueue是Message数据结构体的队列,不过这个队列核心是native层实现的,不是java层实现的.

<c> : ThreadLocal这个是为每一个线程保存一个全局变量的副本,这个副本在不同的线程之中可能都不一样,在同一个线程中可以读写这个副本,而不会影响其他线程.在这里它是保存这样的,建立Looper副本:

sThreadLocal.set(**new** Looper(quitAllowed));

<d> : Looper : 封装了消息循环和消息队列,即MessageQueue和线程对象的载体.相当于一个消息动力设备.

**private static** Looper *sMainLooper*; *// guarded by Looper.class***final** MessageQueue **mQueue**;  
**final** Thread **mThread**;

其中sMainLooper是个全局静态Looper,目前android系统特指为UI主Main线程对应的消息,这么设定,应该是大部分的消息处理只需要放到主Main线程中处理就可以了.平时在开发过程中大部分也是默认在主Main线程中处理(也行是我很少自定义线程),如果使用Main线程sMainLooper作为消息处理,那么Looper是对开发隐藏的.即只需要调用Handler工具类发送消息和回调处理消息就可以了.如果消息需要在自定义线程中处理,那么就需要定制Looper,那就要在自己的线程中添加:

Public class AThread extend Thread{

Public void run(){

Looper.prepare() // 第一步

// todo your work here

// 通过工具类发送消息 // 第二步

Looper.loop(); // 第三步

}

}

那么Looper类中的mThread即是AThread的对象,为什么?看源码:

**private** Looper(**boolean** quitAllowed) {  
 **mQueue** = **new** MessageQueue(quitAllowed);  
 **mThread** = Thread.*currentThread*();  
}

Looper.prepare() 做了什么?

***sThreadLocal***.set(**new** Looper(quitAllowed)); // 记住这个地方,将Looper放到***sThreadLocal***中.

其中new Looper就是上面的部分.

也就是说自定义的线程需要创建自己的消息体mQueue,以及mThread是自定义当前的线程对象.

Looper.loop() 做了什么?这个是消息循环,拿出消息来处理的过程:截取部分源码Looper.java

**final** Looper me = *myLooper*();  
**if** (me == **null**) {  
 **throw new** RuntimeException(**"No Looper; Looper.prepare() wasn't called on this thread."**);  
}  
**final** MessageQueue queue = me.mQueue;  
  
*// Make sure the identity of this thread is that of the local process,  
// and keep track of what that identity token actually is.*Binder.*clearCallingIdentity*();  
**final long** ident = Binder.*clearCallingIdentity*();  
  
**for** (;;) {  
 Message msg = queue.next(); *// might block*

其中myLooper拿出Looper创建时添加到***sThreadLocal***的副本

**public static** @Nullable Looper myLooper() {  
 **return *sThreadLocal***.get();  
}

通俗的说 : 在一个线程中***sThreadLocal***设置进去的和get出来的是同一个对象.这一句话很重要.这也是为什么即使在N个线程中创建N个Looper,在使用时去获取的时候都是各自的Looper.

<e> : Handler : 它只是个工具类,用于开发发送消息和最终回调处理消息结果.

总结上面 :

sMainLooper[static]全局的

Handler

线程:主线程或者自定义线程

自定义线程

Handler

产生消息

消息处理回调

Looper

MessageQueue

消息循环

自定义线程

主线程

消息循环

自定义线程

MessageQueue

Handler

发送消息入队列

Message

Message

Message

Message

Message处理

消息处理回调

消息队列

消息泵工作原理 :