Android Handler的使用

1. 简介

之前讲了关于Handler的原理，这篇要讲一讲关于Handler的使用方法。

2. Handler的初始化

通过下面的代码可知，当msg.what 中的数值为1时，发出Toast通知，内容为msg.obj 中的字符串。
如果为其他，则发出另一种Toast通知。

```
// 实例化Handler
handler = object : Handler() {
        // 重写处理消息的方法
        // 在这里重写的方法即是dispatchMessage中第二优先级的Handler的回调
        // 可以通过msg.what中的数值来判断此消息是从何处发过来的
        override fun handleMessage(msg: Message) {
            when (msg.what) {
                1 -> {
                    Toast.makeText(applicationContext, msg.obj.toString(),
Toast.LENGTH_LONG)
                            .show()
                }
                else -> {
                    Toast.makeText(applicationContext, "others",
Toast.LENGTH_LONG).show()
                }
            }
        }
    }
```

3. 普通的消息传递

普通的消息传递是Handler的最基本的使用方法。通过msg.what 中数据可以判断出信息会从哪里发出来的，以及可以获取消息中存储的数据内容。
闲话不多说，看代码。

```
private fun handlerMessage() {
        Thread(Runnable {
            val message = Message.obtain()
            message.what = 1
            message.obj = "handler message"

            // 通过这个方法，把消息存储到MessageQueue中
            handler.sendMessage(message)
        }).start()
    }
```

4. Runnable消息的传递

除了上面的普通消息的传递，还可以传递一个Runnable，让收到消息的线程执行Runnable中的代码。

```
private fun handlerRunnable() {
        Thread(Runnable {
            // 这里通过post把Runnbale传递给Handler，
            // 实际原理是post方法会创建一个Message，并且把Message的callback设为Runnable
            handler.post(Runnable {
                Toast.makeText(applicationContext, "handler runnable", Toast.LENGTH_LONG).show()
            })
        }).start()
    }
```

除了上面的实时传递Runnable以外还可以延迟执行Runnable。如下面的代码。

```
private fun handlerDelayRunnable() {
        Thread(Runnable {
            // Runnble里面的代码会被延迟2秒执行
            handler.postDelayed(Runnable {
                Toast.makeText(applicationContext, "handler delayRunnable",
Toast.LENGTH_LONG)
                    .show()
            }, 2000)
        }).start()
    }
```

当然除了实时传递和延迟传递以外还有 **定时传递** postAtTime() 方法。具体的使用方法可以看一下文档，用起来很简单。

5. 注意点

1. 同一个实例化的Message对象不要发送（sendMessage ）两次，会报错。
2. Message对象可以通过直接实例化(Message())来获取，但并不推荐。推荐使用val message = Message.obtain() 来获取。
3. 如果在Handler中引用了Activity等有长生命周期的组件，在Activity的onDestory时不要忘了对Handler的取消，以免数据泄漏。取消的方法是handler.removeCallbacksAndMessages(null) 。

关于Android Handler原理：https://juejin.im/post/5e2bfd15e51d454d94423399
github：https://github.com/HyejeanMOON/HandlerDemo