Android微信抢红包插件源码解析

Android (/lib/tag/Android) 2015-12-03 09:06:44 发布

您的评价: 0.0 还行

收藏 1收藏

阅读目录

这个Android插件可以帮助你在微信群聊抢红包时战无不胜。当检测到红包时，插件会自动点击屏幕，人工点击的速度无法比拟。

你正在查看的是**dev分支** (https://github.com/geeeeeeeeek/WeChatLuckyMoney/tree/dev)，这个分支包含大量实验性的修改，不再更新。如果你希望有一个可以立即使用的插件请切换到**stable分支** (https://github.com/geeeeeeeeek/WeChatLuckyMoney/tree/stable)。

> **注：** stable分支的插件只点击最新的红包，根据目前测试抢红包成功率100%。dev分支在stable分支的基础上尝试了大量修改和优化，能使用但无法保证稳定性。

下面的文档仅针对**dev分支**。

预期特性

1. 可以抢屏幕上显示的所有红包，同类插件往往只能获取最新的一个红包。
2. 智能跳过已经戳过的红包，避免频繁点击影响正常使用。
3. 红包日志 (默认未开启)，方便查看抢过的红包内容。
4. 性能优化，感受不到插件的存在，可一直后台开启，不影响日常聊天。
5. 由于这是一份教学代码，项目的文档和注释都比较完整，代码适合阅读。

实现原理

1. 抢红包流程的逻辑控制

这个插件通过一个Stage类来记录当前对应的阶段。Stage类被设计成单例并惰性实例化，因为一个Service不需要也不应该处在不同的阶段。对外暴露阶段常量和entering和getCurrentStage两个方法，分别记录和获取当前的阶段。

```
public class Stage {

    private static Stage stageInstance;

    public static final int FETCHING_STAGE = 0, OPENING_STAGE = 1, FETCH

    private int currentStage = FETCHED_STAGE;

    private Stage() {}

    public static Stage getInstance() {
        if (stageInstance == null) stageInstance = new Stage();
        return stageInstance;
    }

    public void entering(int _stage) {
        stageInstance.currentStage = _stage;
    }

    public int getCurrentStage() {
        return stageInstance.currentStage;
    }
}
```

1.1 阶段说明

阶段	说明
FETCHING_STAGE	正在读取屏幕上的红包，此时不应有别的操作
FETCHED_STAGE	已经结束一个FETCH阶段，屏幕上的红包都已加入待抢队列
OPENING_STAGE	正在拆红包，此时不应有别的操作
OPENED_STAGE	红包成功抢到，进入红包详情页面

1.程序以FETCHED_STAGE 开始，将屏幕上的红包加入待抢队列：

```
--> FETCHED_STAGE --> FETCHING_STAGE  --> FETCHED_STAGE -->
```

阅读目录

2.处理待抢队列中的红包：

```
--> [CLICK] --> OPENING_STAGE --> [CLICK] --> OPENED_STAGE --> [BACK] --

--> [CLICK] --> OPENING_STAGE --> [BACK] --> FETCHED_STAGE -->(没抢到)
```

3.不断重复流程1和2

1.2 根据阶段选择不同的入口

在每次窗体状态发生变化后，根据当前所在的阶段选择入口。

```
switch (Stage.getInstance().getCurrentStage()) {
    case Stage.OPENING_STAGE:
        // .......
        Stage.getInstance().entering(Stage.FETCHED_STAGE);
        performGlobalAction(GLOBAL_ACTION_BACK);
        break;
    case Stage.OPENED_STAGE:
        Stage.getInstance().entering(Stage.FETCHED_STAGE);
        performGlobalAction(GLOBAL_ACTION_BACK);
        break;
    case Stage.FETCHED_STAGE:
        if (nodesToFetch.size() > 0) {
            AccessibilityNodeInfo node = nodesToFetch.remove(nodesToFetc
            if (node.getParent() != null) {
                // .......
                Stage.getInstance().entering(Stage.OPENING_STAGE);
                node.getParent().performAction(AccessibilityNodeInfo.ACT
            }
            return;
        }
        Stage.getInstance().entering(Stage.FETCHING_STAGE);
        fetchHongbao(nodeInfo);
        Stage.getInstance().entering(Stage.FETCHED_STAGE);
        break;
}
```

2. 屏幕内容检测和自动化点击的实现

阅读目录

和其他插件一样，这里使用的是Android API提供的AccessibilityService。这个类位于android.accessibilityservice包内，该包中的类用于开发无障碍服务，提供代替或增强的用户反馈。

AccessibilityService 服务在后台运行，等待系统在发生 AccessibilityEvent 事件时回调。这些事件指的是用户界面上发生的状态变化， 比如焦点变更、按钮按下等等。服务可以请求“查询当前窗口中内容”的能力。 开发辅助服务需要继承该类并实现其抽象方法。

2.1 配置AccessibilityService

在这个例子中，我们需要监听的事件是当红包来或者滑动屏幕时引起的屏幕内容变化，和点开红包时窗体状态的变化，因此我们只需要在配置XML的accessibility-service标签中加入一条

```
android:accessibilityEventTypes="typeWindowStateChanged|typeWindowConten
```

或在onAccessibilityEvent回调函数中对事件进行一次类型判断

```
final int eventType = event.getEventType();
if (eventType == AccessibilityEvent.TYPE_WINDOW_STATE_CHANGED
     || eventType == AccessibilityEvent.TYPE_WINDOW_CONTENT_CHANGED) {
         // ...
}
```

除此之外，由于我们只监听微信，还需要指定微信的包名

```
android:packageNames="com.tencent.mm"
```

为了获取窗口内容，我们还需要指定

```
android:canRetrieveWindowContent="true"
```

其他配置请看代码。

2.2 获取红包所在的节点

首先，我们要获取当前屏幕的根节点，下面两种方式效果是相同的：

```
AccessibilityNodeInfo nodeInfo = event.getSource(); AccessibilityNodeInf
```

阅读目录

这里返回的AccessibilityNodeInfo是窗体内容的节点，包含节点在屏幕上的位置、文本描述、子节点id、能否点击等信息。从 AccessibilityService的角度来看，窗体上的内容表示为辅助节点树，虽然和视图的结构不一定一一对应。换句话说，自定义的视图可以自己描述上面的辅助节点信息。当辅助节点传递给AccessibilityService之后就不可更改了，如果强行调用引起状态变化的方法会报错。

在聊天页面，每个红包上面都有“领取红包”这几个字，我们把它作为识别红包的依据。如果你收到了这四个字的文本消息，可能其他的插件会做出误判。因为我们加入了阶段的概念，因此不会出现这个问题。

AccessibilityNodeInfo的API中有一个findAccessibilityNodeInfosByText方法允许我们通过文本来搜索界面中的节点。匹配是大小写敏感的，它会从遍历树的根节点开始查找。API文档中特别指出，为了防止创建大量实例，节点回收是调用者的责任，这一点会在接下来的部分中讲到。

```
List<AccessibilityNodeInfo> node1 = nodeInfo.findAccessibilityNodeInfosB
```

2.3 对节点进行操作

AccessibilityNodeInfo同样暴露了一个API——performAction来对节点进行点击或者其他操作。出于安全性考虑，只有这个操作来自AccessibilityService时才会被执行。

```
nodeInfo.performAction(AccessibilityNodeInfo.ACTION_CLICK);
```

不过，我们在调试时发现"领取红包"的mClickable属性为false，说明点击的监听加在它父辈的节点上。通过getParent获取父节点，这个节点是可以点击的。

我们还需要全局的返回操作，最方便的办法就是performGlobalAction，不过注意这个方法是API 16后才有的。

```
performGlobalAction(GLOBAL_ACTION_BACK);
```

3. 获取屏幕上的所有红包

和其他插件最大的区别是，这个插件的逻辑是获取屏幕上所有的红包节点，去掉已经获取过的之后，将待抢红包加入队列，再将队列中的红包一个个打开。

3.1 判断红包节点是否已被抢过

实现这一点是编写时最大的障碍。对于一般的Java对象实例来说，除非被GC回收，实例的Id都不会变化。我最初的想法是通过正则表达式匹配下面的十六进制对象id来表示一个红包。

```
android.view.accessibility.AccessibilityNodeInfo@2a5a7c; .......
```

但在测试中，队列中的部分红包没有被戳开。进一步观察发现，新的红包节点和旧的红包节点id出现了重复，且出现概率较大。由于GC日志正常，我推测 AccessibilityNode可能有一个实例池的设计。获取当前窗体节点树的时候，从一个可重用的实例池中获取一个辅助节点信息 (AccessibilityNodeInfo)实例。在接下来的获取时，仍然从实例池中获取节点实例，这时可能会重用之前的实例。这样的设计是有好处的，可以防止每次返回都创建大量的实例，影响性能。AccessibilityNodeProvider的源码表明了这样的设计。

也就是说，为了标识一个唯一的红包，只用实例id是不充分的。这个插件采用的是红包内容+节点实例id的hash来标记。因为同一屏下，同一个节点树下的节点id是一定不会重复的，滑动屏幕后新红包的内容和节点id同时重复的概率已经大大减小。更改标识策略后，实测中几乎没有出现误判。

3.2 将新出现的红包加入待抢队列

我们维护了两个列表，分别记录待抢红包和抢过的红包标识。

```
private List<AccessibilityNodeInfo> nodesToFetch = new ArrayList<>();

private List<String> fetchedIdentifiers = new ArrayList<>();
```

在每次读取聊天屏幕的时候，会检查这个红包是否在fetchedIdentifiers队列中，如果没有，则加入nodesToFetch队列。

```
for (AccessibilityNodeInfo cellNode : fetchNodes) {
    String id = getHongbaoHash(cellNode);
    /* 如果节点没有被回收且该红包没有抢过 */
    if (id != null && !fetchedIdentifiers.contains(id)) {
        nodesToFetch.add(cellNode);
    }
}
```

4. 打开队列中的红包

通过红包打开后显示的文本判断这个红包是否可以抢，进行接下来的操作。

4.1 判断红包节点是否被重用

这也是实现时的一个坑。前面提到了实例池的设计，当我们把红包们加入待抢队列，戳完一个红包再回来时，队列中的其他红包节点可能已被回收重用，如果再去点击这个节点，显然没有什么卵用。

阅读目录

为了解决这个问题，我们只能退而求其次，在点开前做一次检查。如果发现被重用了，就舍弃这个节点，在下一轮fetch的阶段重新加入待抢队列。确认没有重用立即打开，并把节点hash加入fetchedIdentifiers队列。这里如果node失效getHongbaoHash会返回null。

```
AccessibilityNodeInfo node = nodesToFetch.remove(nodesToFetch.size() - 1
if (node.getParent() != null) {
    String id = getHongbaoHash(node);
    if (id == null) return;
    fetchedIdentifiers.add(id);
    Stage.getInstance().entering(Stage.OPENING_STAGE);
    node.getParent().performAction(AccessibilityNodeInfo.ACTION_CLICK);
}
```

可以看出这并不是很有效率的解决方案。在实测中，有时队列中中红包失效后被舍弃，但没有新的AccessibilityEvent发生，接下来的操作都被挂起了。在戳过较多红包之后，这种情况表现得尤为明显，必须要显式地改变窗体内容才能解决。

4.2 根据红包类型选择操作

红包被戳开前会进行查重。戳开后如果界面上出现了“拆红包”几个字，说明红包还没有被别人抢走，立刻点击“拆红包”并将stage标记为OPENED_STAGE。

此时，另三种情况表明抢红包失败了，直接返回，接下来状态会被标记为FETCHED_STAGE。

1. “过期”，说明红包超过有效时间
2. “手慢了”，说明红包发完但没抢到
3. “红包详情”，说明你已经抢到过

4.3 防止加载红包时返回

戳开红包和红包加载完之间有一个“正在加载”的过渡动画，会触发onAccessibilityEvent回调方法。如果在加载完之前判断，上述文本还没出现，会被默认标记为FETCHED_STAGE并触发返回。因此，我们要在返回前特殊判定这种情形。

我们引入了TTL来记录尝试次数，并返回错误值-1。如果到达MAX_TTL时红包还没有加载出来就舍弃这个红包。

```
Stage.getInstance().entering(Stage.OPENING_STAGE);
ttl += 1; return -1;
```

版权说明

阅读目录

本项目源自小米今年秋季发布会时演示的抢红包测试源码(https://github.com/XiaoMi/LuckyMoneyTool)。stable分支基于此代码继续开发，dev分支重写了几乎所有的逻辑代码。应用的包名com.miui.hongbao未变。

由于插件可能会改变自然的微信交互方式，这份代码仅可用于教学目的，不得更改后用于其他用途。

项目使用MIT许可证。在上述前提下，你可以将代码用于任何用途。

项目主页：http://www.open-open.com/lib/view/home/1449104557530 (http://www.open-open.com/lib/view/home/1449104557530)

扩展阅读

Android微信抢红包外挂源码 (/lib/view/open1454732047167.html)
Android微信红包插件 (/lib/view/open1449754717801.html)
Android微信抢红包外挂 (/lib/view/open1424576626728.html)
QQ抢红包插件实现 (/lib/view/open1452598631292.html)
Android 开发小贴士 (/lib/view/open1453898074230.html)

为您推荐

响应式Web设计实战总结 (/lib/view/open1435632371778.html)
利用 JavaScript 数据绑定实现一个简单的 MVVM 库 (/lib/view/open1460163338795.html)
为什么 ContentEditable 很恐怖 (/lib/view/open1416385094336.html)
深度分析Twitter Heron (/lib/view/open1433473314119.html)
H5移动端知识点总结 (/lib/view/open1454548816823.html)

更多

Android (https://www.baidu.com/s?wd=site:open-open.com Android)
Android开发 (https://www.baidu.com/s?wd=site:open-open.com Android开发)

阅读目录

同类热门新闻

1. Android开发周报：Google I/O大会日期确定、Bitmap内存详解 (http://www.open-open.com/news/view/e1146f)
2. [第11期]Android周报 (http://www.open-open.com/news/view/fe8090)
3. Android开发技术周报 Issue#23 (http://www.open-open.com/news/view/1e4063d)
4. Android开发技术周报 Issue#39 (http://www.open-open.com/news/view/b3be35)
5. Android开发周报：Play Store搜索广告推出、ListView源码解析 (http://www.open-open.com/news/view/97d6e5)
6. Android开发周报：Google I/O 2015回顾、Context源码解析 (http://www.open-open.com/news/view/14bda29)

同类热门经验

1. 上百个Android开源项目分享 (/lib/view/open1328063267889.html)
2. android json解析及简单例子 (/lib/view/open1326376799874.html)
3. Android 软件自动更新功能的实现 (/lib/view/open1322529319718.html)
4. 自定义 Android 对话框 (AlertDialog) 的样式 (/lib/view/open1325635738437.html)
5. adb shell 命令详解 (/lib/view/open1327557366686.html)
6. android定位和地图开发实例 (/lib/view/open1328756009421.html)

阅读目录

相关文档 — 更多 (http://www.open-open.com/doc)

- android 开发的微信小程序初体验.pdf (http://www.open-open.com/doc/view/d26e83f21f464b7ea6f96f90ef3c5146)
- android源码编译.doc (http://www.open-open.com/doc/view/29d03a17f8d441ca9efa3f47f7162304)
- Android定位源码.doc (http://www.open-open.com/doc/view/264f9fc05be44cb38c5d6b14e3e35f75)
- Android 源码编译指南.pdf (http://www.open-open.com/doc/view/c7d46b0dce884556926acf71796834b0)
- Android 通讯录源码.pdf (http://www.open-open.com/doc/view/5bbd27d3ebcd4550a75c6ba441402639)

相关经验 — 更多 (http://www.open-open.com/lib)

- Android微信抢红包外挂源码 (/lib/view/open1454732047167.html)
- Android微信红包插件 (/lib/view/open1449754717801.html)
- Android微信抢红包外挂 (/lib/view/open1424576626728.html)
- QQ抢红包插件实现 (/lib/view/open1452598631292.html)

相关讨论 — 更多 (http://www.open-open.com/solution)

- Android Fragment完全解析 (http://www.open-open.com/solution/view/1434936049419)
- 高效稳定的全平台IM支持pc手机音视频寻求合作源码出售 (http://www.open-open.com/solution/view/1457150803687)
- android笔记--JSON数据解析 (http://www.open-open.com/solution/view/1319448346593)