FlowTracker插件文档

本插件主要面向软件开发者，能够在软件开发活动中**追踪开发者的开发行为**和**采样开发者的工作状态**，主要包括两个部分：

* 集成开发环境（IDE）内部触发的事件采集插件；
* 基于事件采样的主观工作状态报告表。

# 1. 功能介绍

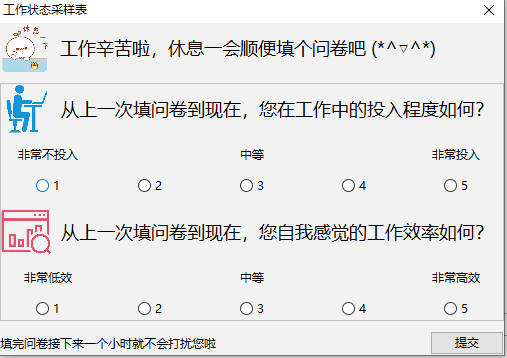
我们目前针对**IntelliJ IDEA**及基于**IntelliJ Platform SDK**开发的**IDE**（例如，**Android Studio**）开发了一个插件，该插件可以以非侵入的方式感知开发者的开发行为。目前我们采集的事件主要包括如下8类：

* **鼠标事件：**包括鼠标点击、鼠标移动、鼠标拖拽、鼠标滚轮滚动等；
* **键盘事件：**记录用户每次键盘活动按下的键，我们会模糊掉具体的字母、数字和标点符号，具体细节可以参考该文档[第4节](#_4._安全与隐私保护)；
* **光标事件：**包括插入光标，移动光标等；
* **文件事件：**包括文件打开，文件保存等；
* **运行事件：**包括运行流程（运行的开始、暂停、恢复和停止，以及exit code）等；
* **动作事件：**IDE内置的动作，通常IDE界面的一个按钮对应一个动作，包括导航、搜索、重构等；
* **调试事件：**包括调试流程（调试的开始、单步跳入/跳出/跳过、停止）、断点的添加/移除等；
* **编译事件：**记录包括编译完成时间、编译结果等，其中编译结果包括是否中断编译流程、警告和错误数目等。

其中记录的日志格式为<time, type, when, [data]>

* time表示Unix时间戳；
* type表示事件类型；
* when表示事件发生的带语义的时机；
* data是可选字段，某些事件会包含若干个键值对。

除了对开发者的行为进行感知，本插件还提供了一套基于事件采样的主观工作状态采样表，具体问题如图1所示：



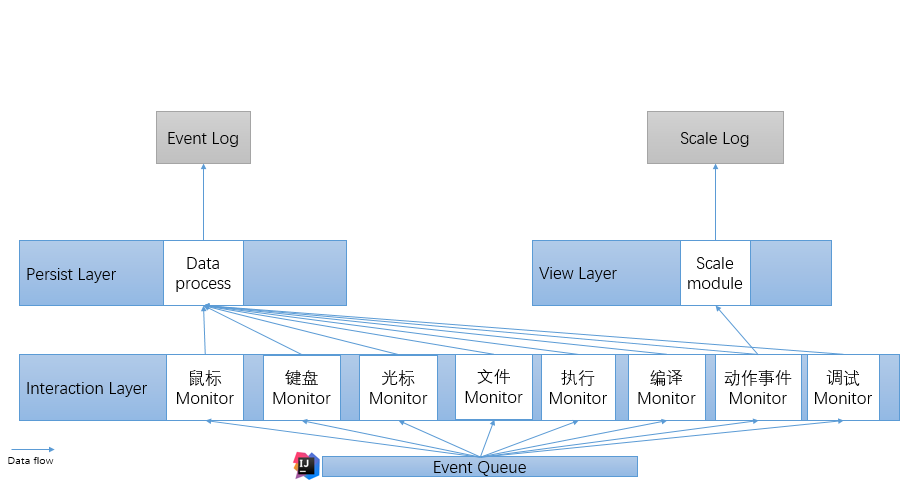
**图1 主观工作状态采样表**

其中有两点值得注意：

* 每次问卷弹出的间隔至少为半个小时；
* 问卷弹出由事件触发，触发事件包括：
  + - 超过30秒没有键盘活动
    - 开始一次运行/调试
    - 代码上传/下载
    - 保存（保存事件弹出问卷的间隔为一个小时）

# 2. 程序组织架构

该插件的组织架构如图2所示，插件主要分为3层，交互层（Interaction Layer）、持久层（Persist Layer）和视图层（View Layer）。



**图2 插件组织架构**

交互层（Interaction Layer）: 开发者与IntelliJ IDEA的所有交互行为都会被IntelliJ IDEA加入它自己的事件队列（Event Queue）中去，我们的插件要获取这些数据，就需要在IDE内注册相对应的监听器来监听。为了从事件队列内采集我们所要采集的开发者的8类事件，我们注册了对应的监听器:

鼠标监听器：

**mouseListener**

调试监听器：

**debuggerContextListener**  
**debuggerManagerListener**

**breakpointListener**

动作事件监听器：

**actionListener**

编译监听器:

**compilationStatusListener**

执行监听器:

**executionTargetListener**

**executionListener**

文件监听器:

**fileEditorManagerListener**

**documentListener**;

**virtualFileListener**

光标监听器:

**caretListener**

键盘监听器:

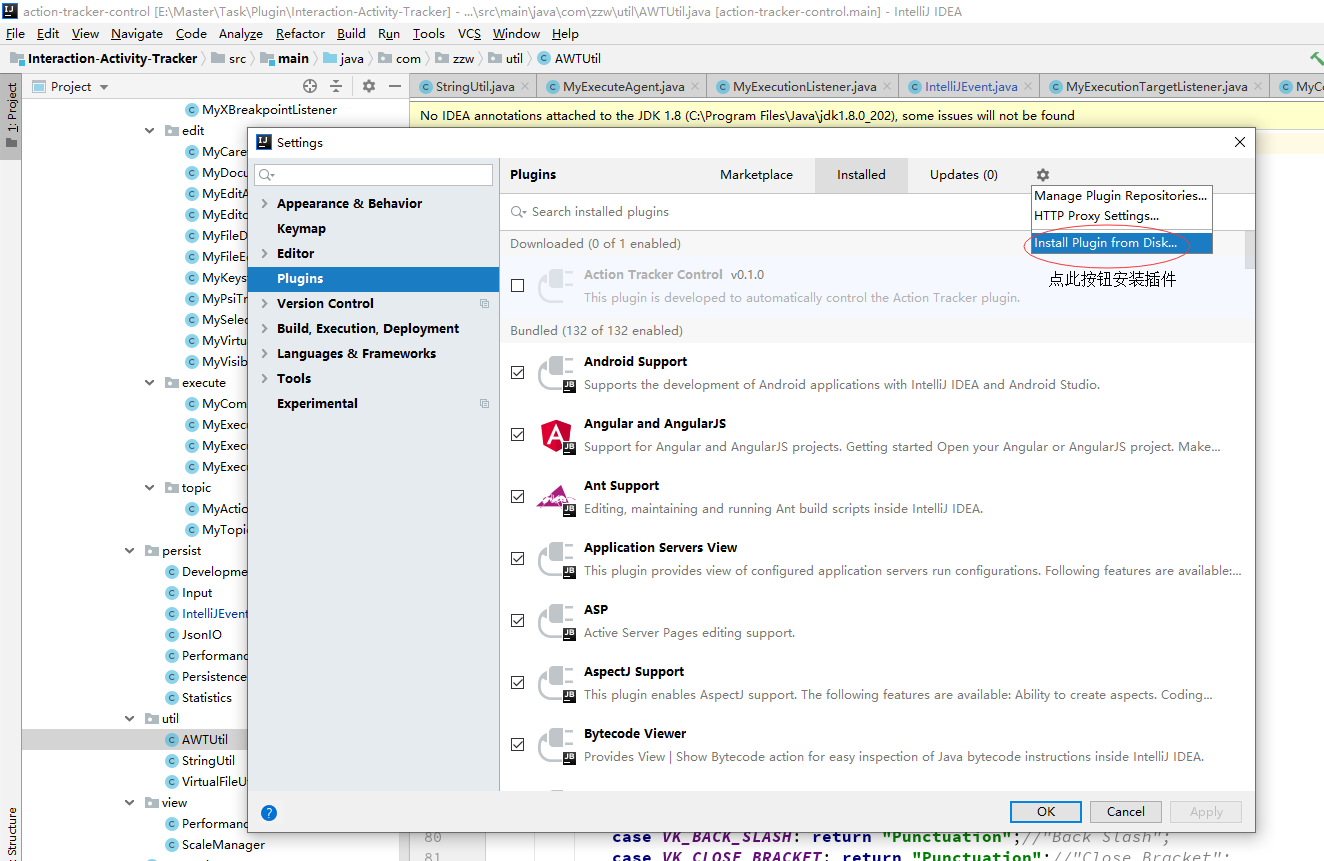
**keystrokeListener**

持久层（Persist Layer）：这层的主要功能就是一些数据处理的工作，包括将键盘事件采集数据中的具体字母、数字、标点信息模糊；将采集到的文件路径进行模糊；将交互层采集的数据规整成统一格式输出到本地日志。

视图层（View Layer）: 这层的主要功能就是检测到当前满足问卷弹出条件时，弹出主观工作状态采样表供开发者填写。

# 3. 使用说明

IntelliJ IDEA插件安装包是Interaction-Activity-Tracker.zip，安装插件时，可以参考[该链接](https://jingyan.baidu.com/article/3d69c5513e5953f0cf02d7b4.html)，如果IntelliJ IDEA的版本在2018以上，安装本地插件的方法如图3，点击如图按钮，选择本地的插件安装包，随后按照提示重启IDE即可完成安装：



**图3：IntelliJ IDEA安装插件**

本插件目前支持的IntelliJ IDEA版本包括：2018.1、2018.2、2018.3和2019.1，暂不支持更低版本的IntelliJ IDEA。插件安装完成并重启IntelliJ IDEA后，检测C盘“C:\Users\用户名”路径是否存在interaction\_traces文件夹，如果存在，表示插件安装成功。插件安装成功后，每次打开IDEA插件都会自动进行数据采集。

如果需要本地编译插件代码，则只能用IntellliJ IDEA导入项目。由于项目是通过Gradle开发，如果你的IDEA没有配置Gradle，会帮你自动配置，这可能需要等待几分钟时间。配置完成后，就可以对插件进行运行和调试了。也可以直接运行项目，这样随后会弹出一个新的IDEA窗口，你可以在这个窗口里试用插件。

# 4. 安全与隐私保护

该插件在独立的JVM中运行，异步接收IDE的事件数据。经过长时间测试，不会影响开发者的IDE使用体验；即使插件崩溃，也不会造成IDE崩溃。

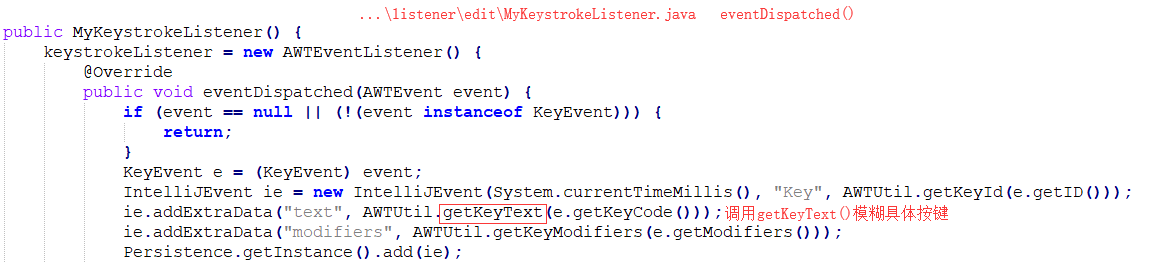
我们的所有监听器，都是向IntelliJ IDEA的Disposer注册的，我们需要的所有的信息都是从IntelliJ IDEA的事件队列获得的，因此不会获取IDEA以外的所有信息。

关于我们的插件采集到的一些可能涉及安全与隐私的数据，我们在此进行一些简单的说明：

关于采集到的IDE编辑区域的**键盘按键信息**，我们只记录按键键区（例如记录按下的是字母、数字还是标点），不记录具体的按键，例如输入的字母都记录为“Letter”，数字都记录为“Number”，标点都记录为“Punctuation”，这在我们程序util包里的AWTUtil.java文件里实现，图4展示了部分代码。模糊方法的调用在我们程序listener.edit包里的MyKeystrokeListener.java文件里可以找到，如图5所示。



**图4 模糊键盘按键信息**

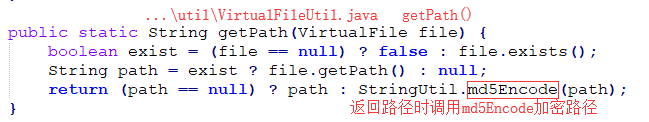


**图5 调用getKeyText模糊按键**

关于采集到的**文件名和路径**，我们利用MD5对其进行加密。MD5是一种不可逆的加密算法，加上其强大的抗修改性，即使文件名只变动了一个字节，得到的MD5值还是有很大的区别，因此保证了无法通过加密后的值恢复原本的文件名和路径，也无法推算出项目文件的大致架构。这在我们程序util包里的StringUtil.java文件里实现，图6展示了所用的MD5加密方法。对于MD5的使用如图7所示，在获取路径之后直接对其进行加密，这在程序util包里的VirtualFileUtil.java文件里实现。

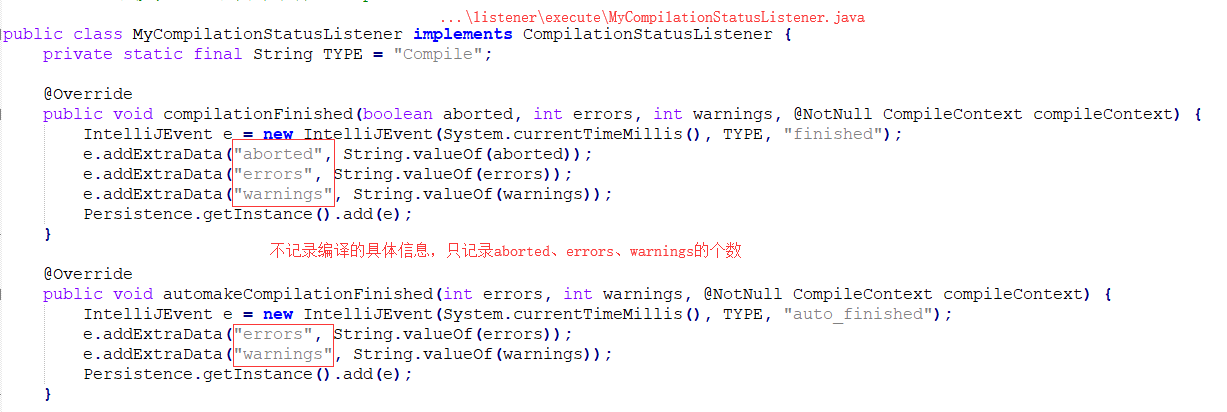


**图6 MD5加密方法**



**图7 MD5的使用**

关于**编译报错等信息**，我们不采集具体的编译信息，只记录error, warning的个数。在程序listener.execute包里的MyCompilationStatusListener.java文件里可以找到，如图8所示。



**图8 编译信息记录**