【每日一问】Android 中如何采集 Fragment 页面浏览事件

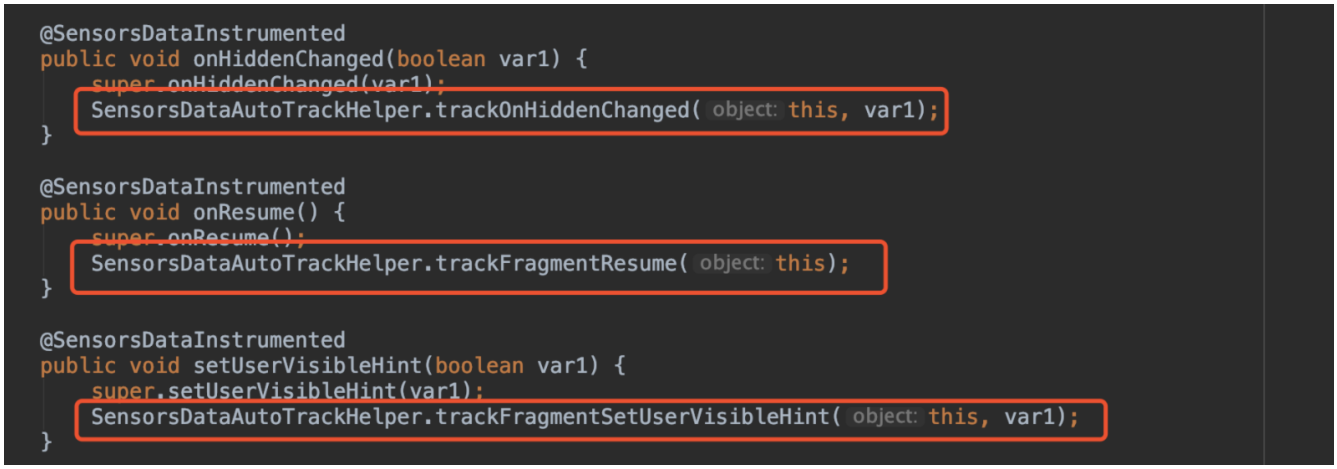
# 1. Fragment 页面浏览事件采集时机

在 Android 系统中，Fragment 页面浏览其实就是指切换不同的 Fragment 。Fragment 本身是没有生命周期的监听的，后期在 Android Support 25.1.0 和 AndroidX 中的 FragmentManager 增加了 FragmentLifecycleCallbacks 用来监听 Fragment 的生命周期，如果不依赖于 Support 库和 AndroidX 库，就无法使用 FragmentLifecycleCallbacks 来监听 Fragment 的生命周期。对于一个 Fragment，通过对 Fragment 生命周期的了解可知，Fragment 可见时会调用 onResume()、setUserVisibleHint()、onHiddenChanged() 这几方法，我们只需要在对应的这几个方法中，添加相应的逻辑判断在 Fragment 可见时，触发页面浏览事件即可实现 Fragment 页面浏览事件的全埋点采集。

# 2. 神策 Android SDK Fragment 页面浏览事件采集实现

神策通过全埋点插件在编译时期插入代码来实现 Fragment 的页面浏览采集，关于全埋点插件的实现原理可参考全埋点插件源码，这里不在介绍，我们下面来看看具体的采集原理。

下图是反编译后的源码，这里使用 ASM 在编译期间通过在 Fragment 系统生命周期方法中分别插入了对应的神策 SDK 方法。




从中可以看到神策全埋点插件 Hook 了 Fragment 的 onResume()、setUserVisibleHint()、onHiddenChanged() 这几方法生命周期方法。

下面分别介绍对每个生命周期插入的代码。

在 Fragment 的 onResume 方法中通过插入 trackFragmentResume(Object object) 方法完成 Fragment 的页面浏览事件的采集。

```
public static void trackFragmentResume(Object object) {
    if (SensorsDataAPI.sharedInstance().isAutoTrackEventTypeIgnored(SensorsDa
{
        return;
    }
    if (!SensorsDataAPI.sharedInstance().isTrackFragmentAppViewScreenEnabled(
{
        return;
    }
    if (!isFragment(object)) {
        return;
    }
    try {
        Method getParentFragmentMethod =
object.getClass().getMethod("getParentFragment");
        if (getParentFragmentMethod != null) {
            Object parentFragment =
getParentFragmentMethod.invoke(object);
            if (parentFragment == null) {
                if (!fragmentIsHidden(object) &&
fragmentGetUserVisibleHint(object)) {
                    trackFragmentAppViewScreen(object);
```

```
                }
            } else {
                if (!fragmentIsHidden(object) &&
fragmentGetUserVisibleHint(object) && !fragmentIsHidden(parentFragment) &&
fragmentGetUserVisibleHint(parentFragment)) {
                    trackFragmentAppViewScreen(object);
                }
            }
        }
    } catch (Exception e) {
        //ignored
    }
}
```

在 Fragment 的 onHiddenChanged 方法中通过插入 trackOnHiddenChanged(Object object, boolean hidden) 方法完成 Fragment 的页面浏览事件的采集。

```
public static void trackOnHiddenChanged(Object object, boolean hidden) {
        if (SensorsDataAPI.sharedInstance().isAutoTrackEventTypeIgnored(Sensor
{
            return;
        }
        if (!SensorsDataAPI.sharedInstance().isTrackFragmentAppViewScreenEnabl
{
            return;
        }
        if (!isFragment(object)) {
            return;
        }
        Object parentFragment = null;
        try {
            Method getParentFragmentMethod =
object.getClass().getMethod("getParentFragment");
            if (getParentFragmentMethod != null) {
                parentFragment = getParentFragmentMethod.invoke(object);
            }
        } catch (Exception e) {
            //ignored
        }
        if (parentFragment == null) {
```

```
            if (!hidden) {
                if (fragmentIsResumed(object)) {
                    if (fragmentGetUserVisibleHint(object)) {
                        trackFragmentAppViewScreen(object);
                    }
                }
            }
        } else {
            if (!hidden && !fragmentIsHidden(parentFragment)) {
                if (fragmentIsResumed(object) &&
fragmentIsResumed(parentFragment)) {
                    if (fragmentGetUserVisibleHint(object) &&
fragmentGetUserVisibleHint(parentFragment)) {
                        trackFragmentAppViewScreen(object);
                    }
                }
            }
        }
    }
```

在 Fragment 的 setUserVisibleHint 方法中通过插入 trackFragmentSetUserVisibleHint(Object object, boolean isVisibleToUser) 方法完成 Fragment 的页面浏览事件的采集。

```
public static void trackFragmentSetUserVisibleHint(Object object, boolean
isVisibleToUser) {
    if (SensorsDataAPI.sharedInstance().isAutoTrackEventTypeIgnored(SensorsDa
{
        return;
    }
    if (!SensorsDataAPI.sharedInstance().isTrackFragmentAppViewScreenEnabled(
{
        return;
    }
    if (!isFragment(object)) {
        return;
    }
    Object parentFragment = null;
    try {
        Method getParentFragmentMethod =
object.getClass().getMethod("getParentFragment");
```

```
            if (getParentFragmentMethod != null) {
                parentFragment = getParentFragmentMethod.invoke(object);
            }
        } catch (Exception e) {
            //ignored
        }
        if (parentFragment == null) {
            if (isVisibleToUser) {
                if (fragmentIsResumed(object)) {
                    if (!fragmentIsHidden(object)) {
                        trackFragmentAppViewScreen(object);
                    }
                }
            }
        } else {
            if (isVisibleToUser && fragmentGetUserVisibleHint(parentFragment))
    {
                if (fragmentIsResumed(object) &&
    fragmentIsResumed(parentFragment)) {
                    if (!fragmentIsHidden(object) &&
    !fragmentIsHidden(parentFragment)) {
                        trackFragmentAppViewScreen(object);
                    }
                }
            }
        }
    }
```

这几个方法是根据 Fragment 不同状态，通过一系列的判断，最终调用 trackFragmentAppViewScreen 方法来采集页面浏览事件。

每日一问的答案中可能无法全完解读这个问题，如果您是相关技术专家或者是对本问题有自己的见解，欢迎带着「批判性」的态度阅读，指正其中的不足。

By 宗 燕山 | 4月 21st, 2020 | 每日一问已关闭评论