1. SDK包中缺少com.tencent.smtt.sdk.ui.dialog.TBSActivityPicker.DialogButton类

答：可以更新官网最新的混淆规则文件，或者不混淆sdk jar包。

1. 如何打开在线文档

答：目前TSB还不支持在线预览功能，只支持本地文件打开，打开方法请参考官网文件版本sdk内的使用说明当Android程序启动时，Android系统会为每个程序创建一个 Application 类的对象，并且只创建一个，application对象的生命周期是整个程序中最长的，它的生命周期就等于这个程序的生命周期。在默认情况下应用系统会自动生成 Application 对象，但是如果我们自定义了 Application，那就需要在 AndroidManifest.xml 中声明告知系统，实例化的时候，是实例化我们自定义的，而非默认的。

但是我们在组件化开发的时候，可能为了数据的问题每一个组件都会自定义一个Application类，如果我们在自己的组件中开发时需要获取 **全局的Context**，一般都会直接获取 application 对象，但是当所有组件要打包合并在一起的时候就会出现问题，因为最后程序只有一个 Application，我们组件中自己定义的 Application 肯定是没法使用的，因此我们需要想办法再任何一个业务组件中都能获取到全局的 Context，而且这个 Context 不管是在组件开发模式还是在集成开发模式都是生效的。

在 组件化工程模型图中，功能组件集合中有一个 **Common 组件**， Common 有公共、公用、共同的意思，所以这个组件中主要封装了项目中需要的基础功能，并且每一个业务组件都要依赖Common组件，Common 组件就像是万丈高楼的地基，而业务组件就是在 Common 组件这个地基上搭建起来我们的APP的，Common 组件会专门在一个章节中讲解，这里只讲 Common组件中的一个功能，在Common组件中我们封装了项目中用到的各种Base类，这些基类中就有**BaseApplication 类**。

BaseApplication 主要用于各个业务组件和app壳工程中声明的 Application 类继承用的，只要各个业务组件和app壳工程中声明的Application类继承了 BaseApplication，当应用启动时 BaseApplication 就会被动实例化，这样从 BaseApplication 获取的 Context 就会生效，也就从根本上解决了我们不能直接从各个组件获取全局 Context 的问题；

这时候大家肯定都会有个疑问？不是说了业务组件不能有自己的 Application 吗，怎么还让他们继承 BaseApplication 呢？其实我前面说的是业务组件不能在集成模式下拥有自己的 Application，但是这不代表业务组件也不能在组件开发模式下拥有自己的Application，其实业务组件在组件开发模式下必须要有自己的 Application 类，一方面是为了让 BaseApplication 被实例化从而获取 Context，还有一个作用是，**业务组件自己的 Application 可以在组件开发模式下初始化一些数据**，例如在组件开发模式下，A组件没有登录页面也没法登录，因此就无法获取到 Token，这样请求网络就无法成功，因此我们需要在A组件这个 APP 启动后就应该已经登录了，这时候组件自己的 Application 类就有了用武之地，我们在组件的 Application的 onCreate 方法中模拟一个登陆接口，在登陆成功后将数据保存到本地，这样就可以处理A组件中的数据业务了；另外我们也**可以在组件Application中初始化一些第三方库**。

但是，实际上业务组件中的Application在最终的集成项目中是没有什么实际作用的，组件自己的 Application 仅限于在组件模式下发挥功能，因此我们需要在将项目从组件模式转换到集成模式后将组件自己的Application剔除出我们的项目；在 AndroidManifest 合并问题小节中介绍了如何在不同开发模式下让 Gradle 识别组件表单的路径，这个方法也同样适用于Java代码；