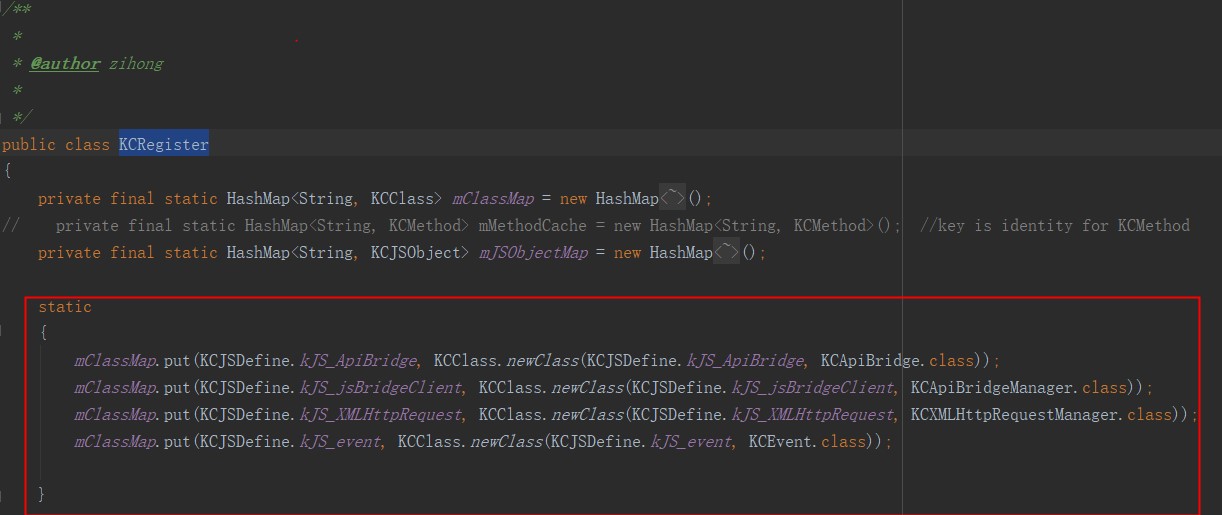
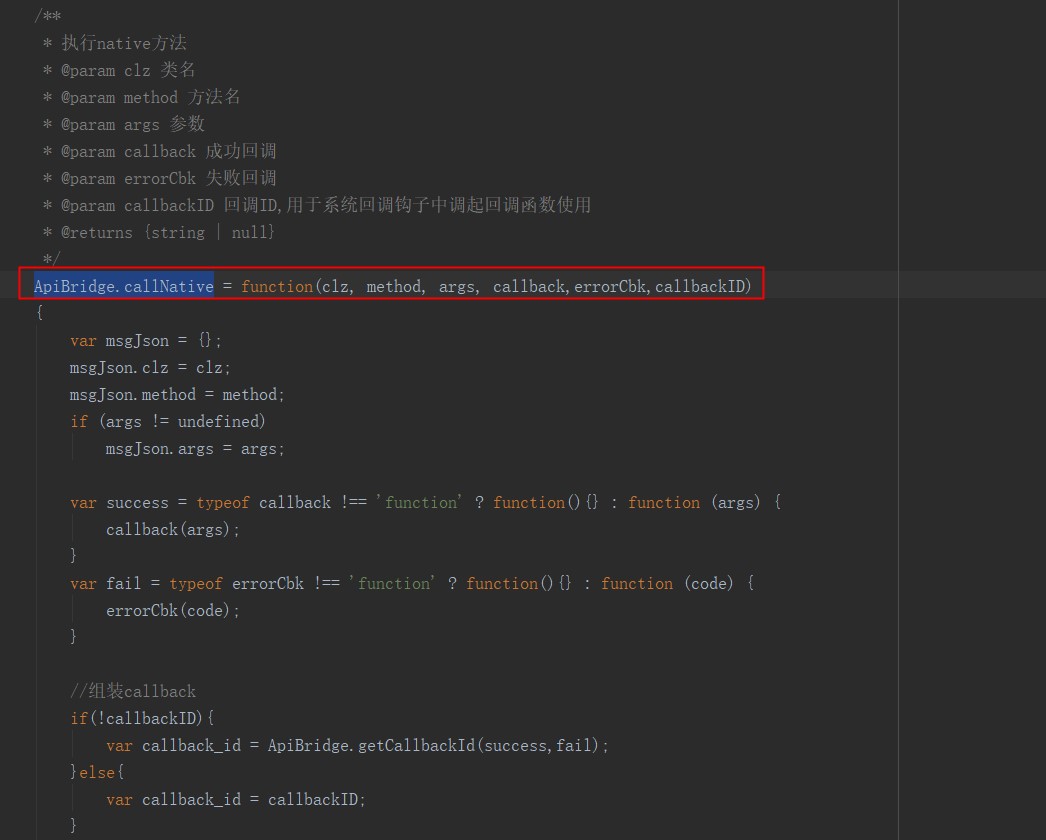
Kerkee 开发文档

官网链接：<http://www.kerkee.com/docs/installation/install_from_packages>

1. 框架(kerkee)核心类介绍
2. com.kercer.kerkee.bridge.KCRegister: 系统的核心类注册类。也包含了拓展其他类注册的方法。
   1. 该注册类代码注册系统类的代码如图所示：



1. com.kercer.kerkee.bridge.KCJSDefine: 常量类，框架自带的注册类的类名和参数主键名称。
2. com.kercer.kerkee.bridge.KCApiBridge: 框架的核心类（桥接类）。
3. com.kercer.kerkee.bridge.KCApiBridgeManager：框架的桥接类管理类，目前没有什么用。
4. com.kercer.kerkee.bridge.xhr.KCXMLHttpRequestManager：框架的网络请求类。
5. com.kercer.kerkee.bridge.event.KCEvent：框架的事件类。
6. com.kercer.kerkee.bridge.KCJSExecutor：框架的执行类，调用js方法的类。
7. 基于官方demo和核心框架（kerkee），二次封装框架。
8. kerkee.js 文件：可以实现跨域请求的文件，在这里封装一些api，供开发的时候使用。
   1. 该文件有个核心方法（ApiBridge.callNative），该方法可以通过已经注册的类方法（即Android那边的方法，或者说是java方法）。
   2. 方法声明如图：



* 1. 到时候可以通过该方法调用已经注册进桥接文件的类的方法。

1. com.kercer.kerkee.api.KCJSObjDefine: 自定义的系统注册类名称和参数主键常量类。
2. com.kercer.kerkee.api.KCRegistMgr：自定义的注册管理类。
3. 开发步骤：
4. 在kerkee.js 文件中编写供 JS 调用的函数。如果需要调用原生方法，可以在方法体使用（ApiBridge.callNative）进行调用。
5. 根据具体的业务，新建和编写提供JS 调用的Native方法（需要静态方法，就是Java中的原生方法）。
6. 在我们的 com.kercer.kerkee.api.KCJSObjDefine 类中根据具体需求定义注册到JS中的类名常量和参数主键产量，或者其他要用到的产量（这样做是为了统一管理）。
7. 在 com.kercer.kerkee.api.KCRegistMgr 类中调用KCJSBridge.registClass 函数进行注册我们第2步中新建的类。