Atitit 日志分析与方法调用链

发出命令与接收，，接收使用缩进一格来。。。

方法闭合 使用notepad++选中字符串模式单词。。所以 cls\_mthd 模式的输出，来确定一个具体命令。。

如何在日志里面确定方法区域。。Block，，使用唯一命名方法名称模式带类名

>>>> call: util.ShellUtil\_\_main([Ljava.lang.String;@21bcffb5)

>>>> call: redis.clients.jedis.Jedis\_\_Jedis(localhost)

>>>> call: redis.clients.jedis.BinaryJedis\_\_BinaryJedis(localhost)

>>>> call: redis.clients.jedis.Client\_\_Client(localhost)

>>>> call: redis.clients.jedis.BinaryClient\_\_BinaryClient(localhost)

>>>> call: redis.clients.jedis.Connection\_\_Connection(localhost)

<<redis.clients.jedis.Connection\_\_Connection() ret:null

<<redis.clients.jedis.BinaryClient\_\_BinaryClient() ret:null

<<redis.clients.jedis.Client\_\_Client() ret:null

<<redis.clients.jedis.BinaryJedis\_\_BinaryJedis() ret:null

<<redis.clients.jedis.Jedis\_\_Jedis() ret:null

>>>> call: redis.clients.jedis.Jedis\_\_set(LHC\_NEED\_11086, 1)

>>>> call: redis.clients.jedis.BinaryJedis\_\_checkIsInMulti()

>>>> call: redis.clients.jedis.BinaryClient\_\_isInMulti()

<<redis.clients.jedis.BinaryClient\_\_isInMulti() ret:false

package util;

import java.util.LinkedList;

import java.util.List;

import com.alibaba.fastjson.JSON;

import javassist.CannotCompileException;

import javassist.ClassPool;

import javassist.CodeConverter;

import javassist.CtBehavior;

import javassist.CtClass;

import javassist.CtConstructor;

import javassist.CtMethod;

import javassist.NotFoundException;

import javassist.Translator;

public class PrintArgumentsTranslator implements Translator {

public static int *tabb*=0;

public static void main(String[] **args**) {

System.***out***.println( JSON.*toJSONString*("2"));

}

public void start(ClassPool **pool**) {}

*@Override*

public void onLoad(ClassPool **classPool**, String **cname**)

throws NotFoundException, CannotCompileException {

// CodeConverter codeConverter = new CodeConverter();

// classPool.importPackage("com.alibaba.fastjson.JSON");

// classPool.importPackage("com.alibaba.fastjson.\*");

CtClass c = **classPool**.get(**cname**);

// c.instrument(codeConverter );

// CtMethod taget\_ctMethod = ctClass.getDeclaredMethod("setA");

for (CtMethod m : c.getDeclaredMethods())

{

insertLogStatement(c, m);

//获取ConverterTranslator的reportSet方法

// CtMethod reportSet = classPool.get("util.PrintArgumentsTranslator").getDeclaredMethod("reportSet");

//该方法用于设定，在执行setA 方法前执行 reportSet 方法

// codeConverter.insertBeforeMethod(m, reportSet);

}

for (CtConstructor m : c.getConstructors())

insertLogStatement(c, m);

}

private void insertLogStatement(CtClass **c**, CtBehavior **m**) {

try {

//c.add

String methodstr = methodSttmtFrg(**m**);

//com.mchange

List<String> inglist=new LinkedList<String>();

inglist.add("com.mchange"); inglist.add("org.apache");

if(contains(inglist,**c**.getName()))

return;

// "----- calling: util.ShellUtil.>main(" + $1+")"

String toPrint =

"\">>>> call: "+**c**.getName() +"\_\_" + **m**.getName() +methodstr +" \"";

if(**m**.isEmpty())

return; //abslt method

**m**.insertBefore("System.out.println("+toPrint+");");

// String retCmd="\"--methret:\"+$\_";

**m**.insertAfter("System.out.println(\"\t<<"+**c**.getName() +"\_\_" + **m**.getName()+"() ret:\"+$\_+\"\\r\\n\");");

//CtBehavior.insertAfter

} catch (Exception e) {

// ignore any exception (we cannot insert log statement)

System.***out***.println( "ex cls:"+**c**+" ,mtd:"+**m**);

e.printStackTrace();

}

}

private String methodSttmtFrg(CtBehavior **m**) throws NotFoundException {

List<String> args = new LinkedList<String>();

for (int i = 0; i < **m**.getParameterTypes().length; i++)

args.add("$" + (i + 1)+"");

// JSON.toJSONString(

String methodstr= args.toString()

.replace("[", "(\" + ")

.replace(",", " + \", \" + ")

.replace("]", "+\")");

return methodstr;

}

private static String encodeJavaMsg(String **msg**) {

char c='\\';

String replacement = String.*valueOf*(c)+""+String.*valueOf*(c);

**msg**=**msg**.replaceAll("\\\\", "\\\\\\\\");

**msg**=**msg**.replaceAll("\n", "\\\\n");

**msg**=**msg**.replaceAll("\r", "\\\\r");

**msg**=**msg**.replaceAll("\"", " \\\" ");

return **msg**;

}

private boolean contains(List<String> **inglist**, String **name**) {

// if(!name.startsWith("util."))

// System.out.println("D");

for (String pkg : **inglist**) {

if(**name**.trim().toLowerCase().startsWith(pkg.trim().toLowerCase()))

return true;

}

return false;

}

}。