ASMSupport使用手册

*——java字节码操作类库*

版本：4.0

作者:温斯群

第1章 关于ASMSupport

1.1 简介

ASMSupport是一个java的字节码操作类库，底层采用的是ASM。那么相比于直接是用ASM，ASMSupport的优势在于它让开发者不需要了解底层指令的情况下能够快速的开发出动态生成字节码的业务。通过实践归结出以下几个asmsupport的优势：

1. 屏蔽了JVM底层指令。
2. 自动堆栈管理，无需像使用asm一般操作堆栈。
3. 采用匿名类的方式模拟java代码的各程序块，使得变量操作更安全。
4. 使用简单，上手快，可以快速的创建class

1.2 Maven

<dependency>

<groupId>cn.wensiqun</groupId>

<artifactId>asmsupport</artifactId>

<version>0.4</version>

</dependency>

第2章 从Hello World开始

2.1 第一个ASMSupport程序

在介绍ASMSupport的一大篇长篇大论之前，先喝口咖啡，通过一个Hello World实例来感受下ASMSupport是如何创建一个类的。虽然你现在还不明白是如何使用asmsupport，但是Hello World嘛，你懂的。

先看下我们想要生成的代码具体是什么样的：

**public** **class** HelloWorldExample

{

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

{

System.*out*.println("Hello World");

}

}

代码2-1

很简单的一段代码.那么接下来就使用ASMSupport生成它。

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

**final** GlobalVariable systemOut =

AClassFactory.*getProductClass*(System.**class**)

.getGlobalVariable("out");

ClassCreator creator = **new** ClassCreator(

Opcodes.*V1\_5*,

Opcodes.*ACC\_PUBLIC* ,

"generated.helloworld.HelloWorldExample", **null**, **null**);

creator.createStaticMethod("main",

**new** AClass[]{AClassFactory.*getProductClass*(String[].**class**)},

**new** String[]{"args"}, **null**, **null**,

Opcodes.*ACC\_PUBLIC* + Opcodes.*ACC\_STATIC*,

**new** StaticMethodBody(){

@Override

**public** **void** generateBody(LocalVariable... argus) {

invoke(systemOut, "println", Value.*value*("Hello World"));

runReturn();

}

});

creator.setClassOutPutPath(".//target//");

Class<?> cls = creator.startup();

cls.getMethod("main", String[].**class**).invoke(cls, **new** Object[]{**null**});

}

代码2-2