ASMSupport使用手册

*——java字节码操作类库*

版本：0.4

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. 首先分析下操作流程，和我们常规编写代码一样，首先我需创建一个类，那么我们就需要一个类创建器，在ASMSupport中他是cn.wensiqun.asmsupport.creator.ClassCreator，我们通过new操作获取一个类创建器；随之而来的就是创建一个main方法，由于main方法是一个static类型的，那么我们将会调用他的createStaticMethod方法来帮助我们创建main方法；光有方法名没有方法的实际内容可不行，如果没有方法体，那么就是创建一个接口来，而这里我们还希望能够打印出Hello World呢，所以在调用createStaticMethod的时候我们会传递一个方法体对象cn.wensiqun.asmsupport.block.method.common.StaticMethodBody，这个对象才是核心，我们需要通过匿名类的方式来实现具体的方法体中的内容。那么如何使用ASMSupport去实现呢，代码送上：

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"));

//don't forget return.

runReturn();

}

});

代码2-2

2.2 原理分析

上面我们分析过ASMSupport的操作流程，接下来我们从字节码层面上分析下ASMSupport的实现原理。首先我们先来看看上面main方法中的内容在字节码层面上是如何表示的：

GETSTATIC System.out

LDC "HelloWorld"

INVOKEIRTUAL PrintStream.println(String): void

RETURN

代码2-3

首先是获取一个System的一个静态成员变量out.然后再将”Hello World”压入操作栈，再调用out的println方法，并且将栈顶的”Hello World”传入println方法，最后再调用一个RETURN操作跳出。整个过程很简单，而前面三条指令在我们代码中表现的行为其实就是一句话System.out.println("Hello World!"),而后一条指令RETURN实际上就是main方法中没有显示的写明的return操作。所以在ASMSupport中为了更符合代码编写的习惯，将前三条指令封装成一个类cn.wensiqun.asmsupport.operators.method.StaticMethodInvo-ker。RETURN指令也封装成一个类cn.wensiqun.asmsupport.operators.Return.然后将这两个类放到当前方法的一个执行队列中。然后按照顺序执行队列中的各个对象，进而调用底层的ASM原始字节码操作实现方法中字节码的动态编写。下图能够简单描述这一过程。

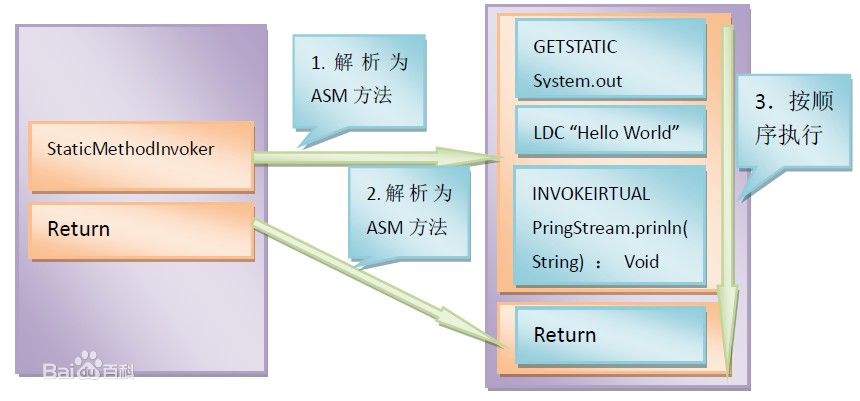


图2-1

第2章 概念篇

这一章主要介绍ASMSupport中的常用类和包结构。是一个比较枯燥的章节，可以将这一章当作是一个工具篇，在实际开发过程中遇到不明白的类能在此章找到相对应的解决方法。如果是初学者也可以在粗略了解本章中介绍的AClass,Value, ClassCreator, ProgramBlock之后直接进入第三章实战篇，在实战篇遇到不明白的类之后进入本章节再细节上了解该类。