|  |
| --- |
| ASM 4.0 |
| 一个Java字节码工程库 |

|  |  |
| --- | --- |
| 作者 | Eric Bruneton |
| 译者 | 王俊超 |

# 目　录

# 介绍

## 动机

程序分析、生成和转换是非常有用的技术，它可以应用在许多的场景中。

* 程序分析，其范围可以从一个简单的句法分析到全语义分析，可以用来发现应用中潜在的错误，检测未用的代码，进行逆向工程代码等。
* 程序生成被用于编译器中。包括传统的编译器，也包括分布式编程中的存根（stub）或者骨架（skeleton）编译器，实时（Just in Time）编译器等。
* 程序转换可用于优化或混淆程序，将调试或性能监视代码到应用程序中，面向切面编程等。

所有这些技术可以用于任何编程语言，但这或多或少取决于使用的语言。就Java而言，它们可以在Java源代码或编译的Java类中使用。在编译后的class类中使用很明显的优点之一是，不需要代码。程序转换，因此可以应用于任何应用，包括闭源代码和商业的。另一个优点是，使用编译的代码，使得在运行时分析，生成或变换类成为可能，类在被装载到Java虚拟机（生成和编译源代码在运行时是可能的，但是这一过程非常缓慢，且需要一个完整的Java编译器）之前完成这些动作。其优点在于一些工具如存根编译器（stub compilers）或切面织入器（aspect weavers）变得对用户透明。

由于程序分析，生成和转换技术的多种可能的用法，对各种编程语言，许多工具已经被实现，用来分析、生成和转换程序，其中就包括Java。ASM就是这些工具中的一个，用于Java语言，它被设计成用于运行时（也可以用于脱机）类生成和转换。因此，ASM 1被设计工作于编译好的Java类，同进也尽可能的快和小。应用为了实现动态类生成或者转换而使用了ASM，为了不过多的拖慢应用的速度，ASM设计得尽可能快是非常重要的。为了让ASM能应于于内存受限的环境和避免应用或者类库膨涨，ASM设计得尽可能的小也是非常重要的。

ASM不仅仅是生成和转换编译后的Java类的的工具，它也是最新最有效的一个。可以从<http://asm.objectweb.org>上下载ASM，它包括以下几个方面的优点：

* 它有一个简单，精心设计和模块化的API，易于使用。
* 这是有据可查的，并有一个相关的Eclipse插件。
* 它提供了最新的Java版本，Java 7的支持。
* 它体积小，速度快，而且非常强大。
* 其广大的用户群体可以为新用户的支持。
* 它的开源许可，您可以在几乎任何你想要的方式使用它。

## 综述

### 范围

ASM库的目标是生成，转换和分析编译的Java类，这些Java类以字节数组形式表示（因为它们存储在磁盘上，并被Java虚拟机加载）。为此，ASM提供了工具来读，写和转换这此字节数组，通过高层概念而不是字节来实现，高层概念包括

使用更高级别的概念不是字节，如数字常量字符串，Java的标识符，Java类型，Java类结构元件等。数字常量、字符串、Java的标识符、Java类型、Java类结构元无素等。注意，ASM库的范围被严格限制于读，写，转化和分析类。尤其是类加载过程是超出ASM关注的范围。

### 模块

ASM库提供了两套API用于生成和转换编译后的类：核心API提供了类基于事件的表示，树形API提供类基于对象的表示。

基于事件的模型，一个类代表了一系列的事件，每一个事件代表了一个类元素，比如类的头部、字段、方法声明以及指令等等。基于事件的API定义了一组可能的事件以及这些事件必然发生的顺序，并提供了一类解析器(class parser)为每一个被解析的元素生成一个事件，同时也提供了一个类写者（class witer）从这一系列的事件中生成编译后的类。

基于对象的模型，一个类以一个树型对象进行表示，每一个对象代表了类的一部分，比如类本身、字段、方法和指令等；同时每一个对象也包含代表它成分的引用对象。