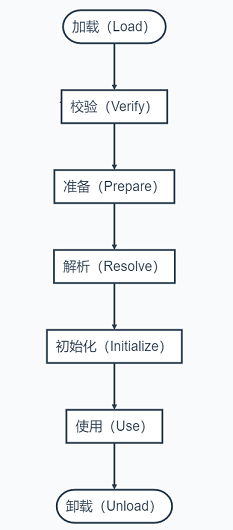
Java 类加载过程

原文地址：<https://www.cnblogs.com/williamjie/p/11167902.html>

**Class的生命周期**

一个Class在虚拟机中的完整生命周期如下图所示：



需要说明的是，上述的流程只是描述了逻辑上各个阶段的开始顺序，实际过程中，各个阶段可能是交错进行，并不是一个阶段等到另一个阶段完全完成才开始执行。

**加载**

加载一个Class需要完成以下3件事：

* 通过Class的全限定名获取Class的二进制字节流
* 将Class的二进制内容加载到虚拟机的方法区
* 在内存中生成一个java.lang.Class对象表示这个Class

获取Class的二进制字节流这个步骤有多种方式：

* 从zip中读取，如：从jar、war、ear等格式的文件中读取Class文件内容
* 从网络中获取，如：Applet
* 动态生成，如：动态代理、ASM框架等都是基于此方式
* 由其他文件生成，典型的是从jsp文件生成相应的Class

**校验**

验证一个Class的二进制内容是否合法，主要包括4个阶段：

* 文件格式验证，确保文件格式符合Class文件格式的规范。如：验证魔数、版本号等。
* 元数据验证，确保Class的语义描述符合Java的Class规范。如：该Class是否有父类、是否错误继承了final类、是否一个合法的抽象类等。
* 字节码验证，通过分析数据流和控制流，确保程序语义符合逻辑。如：验证类型转换是合法的。
* 符号引用验证，发生于符号引用转换为直接引用的时候（转换发生在解析阶段）。如：验证引用的类、成员变量、方法的是否可以被访问（IllegalAccessError），当前类是否存在相应的方法、成员等（NoSuchMethodError、NoSuchFieldError）。

**准备**

在准备阶段，虚拟机会在方法区中为Class分配内存，并设置static成员变量的初始值为默认值。注意这里仅仅会为static变量分配内存（static变量在方法区中），并且初始化static变量的值为其所属类型的默认值。如：int类型初始化为0，引用类型初始化为null。即使声明了这样一个static变量：

public static int a = 123;

在准备阶段后，a在内存中的值仍然是0, 赋值123这个操作会在中初始化阶段执行，因此在初始化阶段产生了对应的Class对象之后a的值才是123 。

**解析**

解析阶段，虚拟机会将常量池中的符号引用替换为直接引用，解析主要针对的是类、接口、方法、成员变量等符号引用。在转换成直接引用后，会触发校验阶段的符号引用验证，验证转换之后的直接引用是否能找到对应的类、方法、成员变量等。这里也可见类加载的各个阶段在实际过程中，可能是交错执行。

**初始化**

初始化阶段即开始在内存中构造一个Class对象来表示该类，即执行类构造器<clinit>()的过程。需要注意下，<clinit>()不等同于创建类实例的构造方法<init>()

* <clinit>()方法中执行的是对static变量进行赋值的操作，以及static语句块中的操作。
* 虚拟机会确保先执行父类的<clinit>()方法。
* 如果一个类中没有static的语句块，也没有对static变量的赋值操作，那么虚拟机不会为这个类生成<clinit>()方法。
* 虚拟机会保证<clinit>（）方法的执行过程是线程安全的。  
  因此，存在如下一种最简单的单例模式的实现：

参考资料： 《深入理解Java虚拟机》

转载：https://www.cnblogs.com/jqctop1/p/7823629.html