我们开发的java应用中包含大量的class，实现的功能都封装在这些class文件当中。Class从加载到内存到从内存卸载经历了这些阶段：加载、连接、初始化、使用、卸载。其中连接有分为验证、准备、解析三个阶段，如下图。



我们重点研究下加载、连接、初始化这三个过程中发生了什么。

一、自定义类加载器加载类

先让我们自定义一个类加载器，并初始化这个class对象，来初步体验下类加载到使用的过程，首先实现一个简单的class类

public class MgDemoSample {  
  
 public void startSample()  
 {  
 System.*out*.println("MgDemoSample");  
 }  
}

使用javac MgDemoSample.java 编译出 class文件，将class文件放到E:/cltmp/目录下，然后创建一个URLClassLoader加载MgDemoSample，并实例化对象，通过反射执行startSample这个函数

URL url = new URL("file:/E:/cltmp/");  
URLClassLoader loader = new URLClassLoader(new URL[]{url});  
Class cl = loader.loadClass("MgDemoSample");  
*log*.info("....{}",cl.getName());  
Object instance = cl.newInstance();  
Method mainMethod = cl.getMethod("startSample");

这样我们就把类的，加载、链接、初始化、使用，走了一遍。

扩展一下，定义两个类加载器，分别加载一个class文件，并获取Class对象cl1和cl2，并用equals比较返回true还是false呢？

接下来让我们一起研究下，加载、连接、初始化，这三个过程。

二、加载

在加载过程中主要做了什么呢

1、获取类的二进制字节流

2、将字节流转换为静态存储结构存储到方法区

3、生成class对象

注意：class的连接这个过程并不是在加载结束后才会开始，这两个阶段是并行的，例如需要验证阶段对文件格式验证通过以后，字节流才能存储到方法区中。

字节流不是必须从class文件中获取，只要是符合class文件结构的二进制流都能加载，例如Spring中IOC就是对class类进行重写并直接生成二进制流，没有子类的实体文件。常见的二进制数据来源有以下几种：

1、class文件中读取

2、jar包中加载class文件

3、动态代理生成字节流

4、通过网络获取字节流

二、连接

连接分为三部分：验证、准备、解析，让我们看下它们都是做了什么吧

1、验证

验证是为了保证获取到的字节流的结构jvm的要求，并且保证流是安全的。验证是类加载中很重要的一步，不过同样也不是必须的一步，如果已经确定字节流的安全性，可以通过参数-Xverify：none设置关闭验证。验证过程中需要验证一下内容：

1、文件格式验证，验证的是class文件结构是否符合要求，是否能被jvm加载，验证的内容如：文件开头是否魔数（0XCAFEBABE），版本号是否符合虚拟机要求，常量是否符合类型要求等。

2、元数据验证，进行语义分析，保证字节码符合java语言规范，如：类是否有父类，是否实现了父类需要实现的所有方法等

3、字节码验证，这个是验证方法在运行时不会做出违反jvm安全的操作。如：保证类型转换的有效等。

4、符号引用验证，这个在解析阶段符号引用转换为直接引用时发生，保证引用能被访问到。如：通过类名能否找到类等。

2、准备

在这个阶段会将类的静态变量分配到内存中，并设置初始值。这里设置的初始值只是默认值，例如int默认是0，static int no=10；，no在这个阶段会被赋值为0，而不是10。

静态的基本类型会被设置为默认值，引用类型设置为null。

3、解析

解析将常量池中字节码的引用转换为直接饮用的过程。在class文件中，符号引用

四、初始化