https://blog.csdn.net/qbg19881206/article/details/8993562

**Javassist**

**1.读写字节码**

Javassist是用来处理java字节码的类库。字节码保存在二进制文件中称为类文件。每个类文件夹包括一个java类或接口。

Javasssist.CtClass这个类是一个类文件的抽象表示。一个CtClass(compile-time class编译时类)对象处理一个类文件。下面是个简单的例子：

ClassPool pool = ClassPool.getDefault();CtClass cc = pool.get("test.Rectangle");cc.setSuperclass(pool.get("test.Point"));cc.writeFile();

这个程序首先包括了一个ClassPool对象，它javassist是用来修改字节码的。ClassPool对象是一个CtClass对象容器。它读取类文件根据需要构建CtClass对象，并且记录构建的对象以便响应最新的访问。修改类的定义，用户首先必须从ClassPool对象引用到一个CtClass对象表示。ClassPool中的get()方法就是这个作用。上面的例子展示，CtClass对象表示了ClassPool中的test.Rectangle对象。ClassPool的getDefault()搜索默认的系统搜索路径。

从实现的角度说，ClassPool是一个CtClass对象的hash表，类名做为key。ClassPool的get()搜索hash表找到与指定key关联的CtClass对象。如果没有找到CtClass对象,get()读一个类文件构建新的CtClass对象，它是被记录在hash表中然后返回这个对象。

在ClassPool对象中的CtClass对象可以被编辑修改（如何修改在后面说明）。上面的例子中，把test.Rectangle的父类改为了test.Point。这个改变在执行writeFile()后实现。

writeFile()转换CtClass对象到一个类文件中并写到本地磁盘。Javassist还提供了一个方法直接得到修改的字节码。得到字节码，执行toBytecode()：

byte[] b = cc.toBytecode(); 也以直接载入CtClass:Class clazz = cc.toClass();toClass()请求当前线程的类载入器载入CtClass表示的类文件。它返回一个java.lang.Class对象来表示载入的类。 定义一个新类 定义新类，在ClassPool中执行makeClass().ClassPool pool = ClassPool.getDefault();CtClass cc = pool.makeClass("Point"); 这个例子定义了没有成员的Point类。CtNewMethod的工厂方法可以构建Point的成员方法，用CtClass的addMethod()加入到Point中。    makeClass()不能建接口,ClassPool的makeInterface()可以。CtNewMethod的abstractMethod()可以为接口构建方法。

**2.冻结的类**

如果CtClass通过writeFile(),toClass(),toBytecode()转换了类文件，javassist冻结了CtClass对象。以后是不允许修改这个 CtClass对象。这是为了警告开发人员当他们试图修改一个类文件时，已经被JVM载入的类不允许被重新载入。

一个冻结的CtClass可以被解冻，因为修改类定义可以被允许。例如：

CtClasss cc = ...;

   :

cc.writeFile();

cc.defrost();

cc.setSuperclass(...);   // OK since the class is not frozen.

Defrost()执行后，CtClass对象将可以再次修改。

如果ClassPool.doPruning被设置为true，javassist将修剪被冻结的CtClass对象的数据结构，以减少内存消耗。修剪会丢掉对象中不需要的属性(attribute\_info structures)。例如，Code\_attribute结构（method bodies）。因此，CtClass对象修剪后，方法的字节码除了方法名，方法签名和注释，其它将不可访问。修剪的CtClass对象并不能再解冻。ClassPool.doPruning默认的值为false。

    不允许修剪执行了CtClass.stopPruning()的对象：

CtClasss cc = ...;

cc.stopPruning(true);

:

cc.writeFile();         // convert to a class file.

// cc is not pruned.

**注意：**调试的时候，你可能想暂时的停止修剪和冻结并写一个类文件到磁盘，debugWriteFile()是为这个目的做的方法。它停止修剪，写一个类文件，解冻它，然后再次修剪（如果是初始化）。

**3.类搜索路径**

默认的ClassPool通过static方法ClassPool.getDefault()返回与JVM一样的搜索路径。如果程序运行在一个web服务器，如JBoss,Tomcat，ClassPool对象可以找不到用户的类，因为web 服务器用多个类载入器(class loader)作为载入器。这种情况，附加的类路径必须注册到ClassPool。应该为pool 提供一个ClassPool 对象：

pool.insertClassPath(new ClassClassPath(this.getClass()));

这段程序注册了用来载入类对象的类路径。可以用任何类对象作为参数替换this.getClass()。注册的路径用来载入类 。

可以注册一个目录名作为类搜索路径。例如，下面的例子加入了一个目录/usr/local/javalib作为搜索路径：

       ClassPool pool = ClassPool.getDefault();

       pool.insertClassPath("/usr/local/javalib");

这个搜索路径不仅可以是目录，还可以是URL：

       ClassPool pool = ClassPool.getDefault();

       ClassPath cp = new URLClassPath("www.javassist.org", 80, "/java/", "org.javassist.");

       pool.insertClassPath(cp);

这个程序加入”http://www.javassist.org:80/java/” 到类搜索路径。这个URL只是搜索属于org.javassist包的类。例如，载入org.javassist.test.Main,它的类文件将被得到：

http://www.javassist.org:80/java/org/javassist/test/Main.class

另外，可以直接把一个byte数组给一个ClassPool对象，构建一个CtClass对象从这个数组。用ByteArrayClassPath，例如：

ClassPool cp = ClassPool.getDefault();

byte[] b = a byte array;

String name = class name;

cp.insertClassPath(new ByteArrayClassPath(name, b));

CtClass cc = cp.get(name);

得到的CtClass对象表示通过b指定的一个类，ClassPool从ByteArrayClassPath读一个类文件,如果执行get()，需要用类名。

如果不知道类的全名，可以用ClassPool中的makeClass()：ClassPool cp = ClassPool.getDefault();

InputStream ins = an input stream for reading a class file;

CtClass cc = cp.makeClass(ins);

makeClass()从输入流中返回CtClass对象。可以用makeClass()把想要修改的类放入ClassPool。如果搜索路径包括一个大的jar文件，它可以提高性能。当ClassPool需要读取一个类文件，它可以为每个类文件重复搜索整个jar文件夹。makeClass()可以有效的搜索。makeClass()构建的CtClass，保存在ClassPool，并且类文件不再被读。

用户可以扩展类搜索路径。可以定义一个实现ClassPath接口的新类，ClassPool的insertClassPath()给了一个类的实例。它允许一个非标准资源包含在搜索路径中。