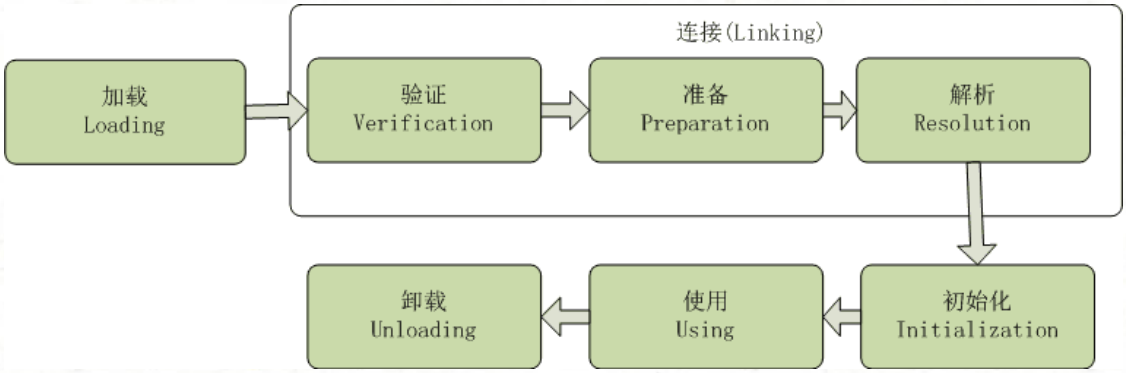
http://www.jianshu.com/p/7a55a8d06e63

虚拟机如何加载Class文件?
Class文件中的信息进入到虚拟机后会发生什么变化?
**虚拟机把描述类的数据从Class文件加载到内存，并对数据进行校验、转换解析和初始化，最终形成可以被虚拟机直接使用的Java类型，这就是虚拟机的类加载机制。**




类的生命周期

**何时加载?**

- 1) 遇到new、getstatic、putstatic或invokestatic这4条字节码指令时，如果类没有进行过初始化，则需要先触发初始化。

场景：new 实例化对象时候、读取或设置一个类的静态字段的时候，以及调用一个类的静态方法的时候。

- 2)使用java.lang.reflect包的方法对类进行反射调用的时候，如果类没有进行过初始化，则需要先触发期初始化。
- 3)当初始化一个类的时候，如果发现其父类还没有进行过初始化，则需要先触发其父类的初始化。
- 4)当虚拟机启动时，用户需要指定一个要执行的主类（包含main()方法的那个类），虚拟机会先初始化这个主类。
- 5)当使用JDK1.7的动态语言支持的，如果一个java.lang.invoke.MethodHandle实例最后的解析结果REF_getStatic、REF_putStatic、REF_invokeStatic的方法句柄，并且这个方法句柄所

对应的类没有进行初始化，则需要先触发其初始化。

# 类加载过程

## 1.加载

在该阶段，虚拟机需要完成以下3件事情:

- 1)通过一个类的全限定名来获取定义此类的**二进制字节流**
- 2)将这个字节流所代表的静态存储结构转化为**方法区**的运行时数据结构。
- 3)在内存中生成一个代表这个类的**java.lang.Class对象**，作为方法区这个类的各种数据的访问入口。

获取类的二进制字节流是开发人员可控性最强的，开发人员可以定义自己的类加载器去控制字节流的获取方式（重写loadClass()方法）。

**数组类加载**

- 1)如果数组的组件类型是引用类型，则递归加载这个组件类型，数组将在加载该组件类型的类加载器的类名称空间上被标识
- 2)如果数组的组件类型不是引用类型(如int[]数组)，Java虚拟机将会把数组标记与引导类加载器关联。
- 3)数组类的可见性与它的组件类型的可见性一致，如果组件类型不是引用类型，那数组类的可见性将默认为public。

## 2.验证

验证是连接阶段的第一步，这阶段的目的是为了确保Class文件的字节流中包含的信息符合当前虚拟机的要求，并且不会危害虚拟机自身的安全。

- 1)文件格式验证
- 2)元数据验证
- 3)字节码验证
- 4)符号引用验证

## 3.准备

准备阶段是正式为类变量**分配内存**并**设置类变量初始值**的阶段，这些变量

所使用的内存都将在**方法区**中进行分配。

| 数据类型 | 零值 |
| --- | --- |
| int | 0 |
| long | 0L |
| short | (short)0 |
| char | '\u0000' |
| byte | (byte)0 |
| boolean | false |
| float | 0.0f |
| double | 0.0d |
| reference | null |

## 4.解析

解析阶段是虚拟机将常量池内的**符号引用**替换为**直接引用**的过程。

- **符号引用**:以一组符号来描述所引用的目标，符号可以是任何形式的字面量，只要使用时能无歧义地定位到目标即可。
- **直接引用**:可以是直接指向目标的指针、相对偏移量或是一个能间接定位到目标的句柄。
- 1)类或接口的解析
- 2)字段解析
- 3)类方法解析
- 4)接口方法解析

## 5初始化

初始化阶段是执行类构造器<clinit>()方法的过程。

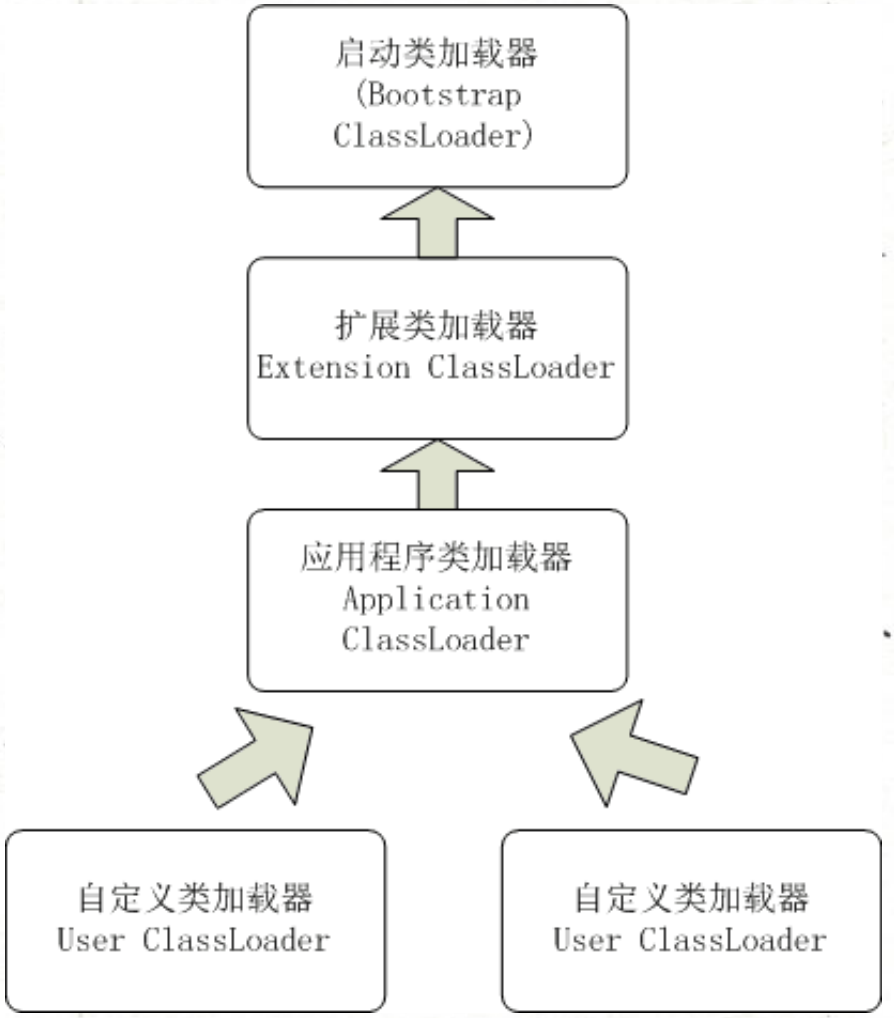
## 类加载器

类加载器实现的是类加载过程中的第一步**“通过一个类的全限定名来获取定义此类的二进制字节流”**。类加载器在类的层次划分、OSGi、热部署、代码加密等领域大放异彩。

对于任意一个类，都需要由加载它的类加载器和这个类本身一同确立其在Java虚拟机中的唯一性，每一个类加载器，都拥有一个独立的类名称空间。

## 1.双亲委派模型

- 1)启动类加载器（Bootstrap ClassLoader）一般加载类库
- 2)扩展类加载器(Extension ClassLoader)一般加载扩展库
- 3)应用程序类加载器（Application ClassLoader）




classloader.png

父子之间的关系一般不会用继承的关系来实现，而是都使用组合关系来复用父加载器的代码。

双亲委派模型的工作过程是:如果一个类加载器收到了类加载的请求，它首先不会自己去尝试加载这个类，而是把这个请求委派给父类加载器去完成，每一层次的类加载器都是如此，因此所有的加载请求最终都应该传递到顶层的启动类加载器中，只有当父加载器反馈自己无法完成这个请求时，子加载器才会去尝试自己去加载, 父类已经加载过的的类(例如

java.lang.System)子类不会去加载(所以手动命名一个java.lang.System类无效)。

## 2.破坏双亲委派模型

OSGi中对类加载器的使用时很值得学习的，弄懂了OSGi的实现，就可以算是掌握了类加载的精髓。