#### Java虚拟机类加载器

1. **JVM三种预定义类型类加载器**

我们首先看一下JVM预定义的三种类型类加载器，当一个 JVM启动的时候，Java缺省开始使用如下三种类型类装入器：

**启动（Bootstrap）类加载器**：引导类装入器是用本地代码实现的类装入器，它负责将 <Java\_Runtime\_Home>/lib下面的核心类库或-Xbootclasspath选项指定的jar包加载到内存中。由于引导类加载器涉及到虚拟机本地实现细节，开发者无法直接获取到启动类加载器的引用，所以不允许直接通过引用进行操作。

**扩展（Extension）类加载器**：扩展类加载器是由Sun的ExtClassLoader实现的。它负责将< Java\_Runtime\_Home >/lib/ext或者由系统变量-Djava.ext.dir指定位置中的类库加载到内存中。开发者可以直接使用标准扩展类加载器。

**系统（System）类加载器**：系统类加载器是由 Sun的 AppClassLoader实现的。它负责将系统类路径java -classpath或-Djava.class.path变量所指的目录下的类库加载到内存中。开发者可以直接使用系统类加载器。

除了以上列举的三种类加载器，还有一种比较特殊的类型就是线程上下文类加载器。

1. **类加载双亲委派机制介绍和分析**

Java类加载器的作用就是在运行时加载类。Java类加载器基于三个机制：委托、可见性和单一性。委托机制是指将加载一个类的请求交给父类加载器，如果这个父类加载器不能够找到或者加载这个类，那么再加载它。可见性的原理是子类的加载器可以看见所有的父类加载器加载的类，而父类加载器看不到子类加载器加载的类。单一性原理是指仅加载一个类一次，这是由委托机制确保子类加载器不会再次加载父类加载器加载过的类。也可以说，JVM在加载类时默认采用的是双亲委派机制。通俗的讲，就是某个特定的类加载器在接到加载类的请求时，首先将加载任务委托给父类加载器，依次递归，如果父类加载器可以完成类加载任务，就成功返回；只有父类加载器无法完成此加载任务时，才自己去加载。父类加载器在加载类之前，先判断该类是否加载，如果没有加载去加载该类。

**委托机制**

当一个类加载和初始化的时候，类仅在有需要加载的时候被加载。假设你有一个应用需要的类叫作Abc.class，首先加载这个类的请求由Application类加载器委托给它的父类加载器Extension类加载器，然后再委托给Bootstrap类加载器。Bootstrap类加载器会先看看rt.jar中有没有这个类，因为并没有这个类，所以这个请求由回到Extension类加载器，它会查看jre/lib/ext目录下有没有这个类，如果这个类被Extension类加载器找到了，那么它将被加载，而Application类加载器不会加载这个类；而如果这个类没有被Extension类加载器找到，那么再由Application类加载器从classpath中寻找。记住classpath定义的是类文件的加载目录，而PATH是定义的是可执行程序如javac，java等的执行路径。

**可见性机制**

子类加载器可以看到父类加载器加载的类，而反之则不行。

**单一性机制**

根据这个机制，父加载器加载过的类不能被子加载器加载第二次。虽然重写违反委托和单一性机制的类加载器是可能的，但这样做并不可取。你写自己的类加载器的时候应该严格遵守这三条机制。

1. **附录**

<https://www.cnblogs.com/zyzcj/p/8073500.html>

<http://www.importnew.com/6581.html>

https://blog.csdn.net/javazejian/article/details/73413292