**类加载器**

在类加载的5个阶段中，在加载阶段做的第一件事“通过一个类的全限定名来获取此类的二进制字节流”这个操作放到了虚拟机外部去实现，以便让程序自己决定如何获取所需要的类，实现这个动作的代码模块被称为“类加载器”。从Java开发人员来看，系统提供的有以下3中类加载器：

* **启动类加载器（Bootstrap ClassLoader）**

这个类加载器是由C++语言实现的，是虚拟机自身的一部分。它主要加载<JAVA\_HOME>\lib目录中的类（或被-Xbootclasspath参数所指定的路径中的，并且是虚拟机识别的（仅按照文件名识别，如rt.jar）类库加载到虚拟机内存中）。启动类加载器无法被Java程序直接引用，如果自定义类加载器需要把加载操作委托给启动类加载器，直接使用null代替就好。

* **扩展类加载器（Extension ClassLoader）**

由sun.misc.Launcher$ExtClassLoader实现，它负责加载在<JAVA\_HOME>\lib\ext目录中的，或被java.ext.dirs系统变量所指定的路径中的所有类库，开发者可以直接使用扩展类加载器。 它是由Java编写，并继承自java.lang.ClassLoader。

* **应用程序类加载器（Extension ClassLoader）**

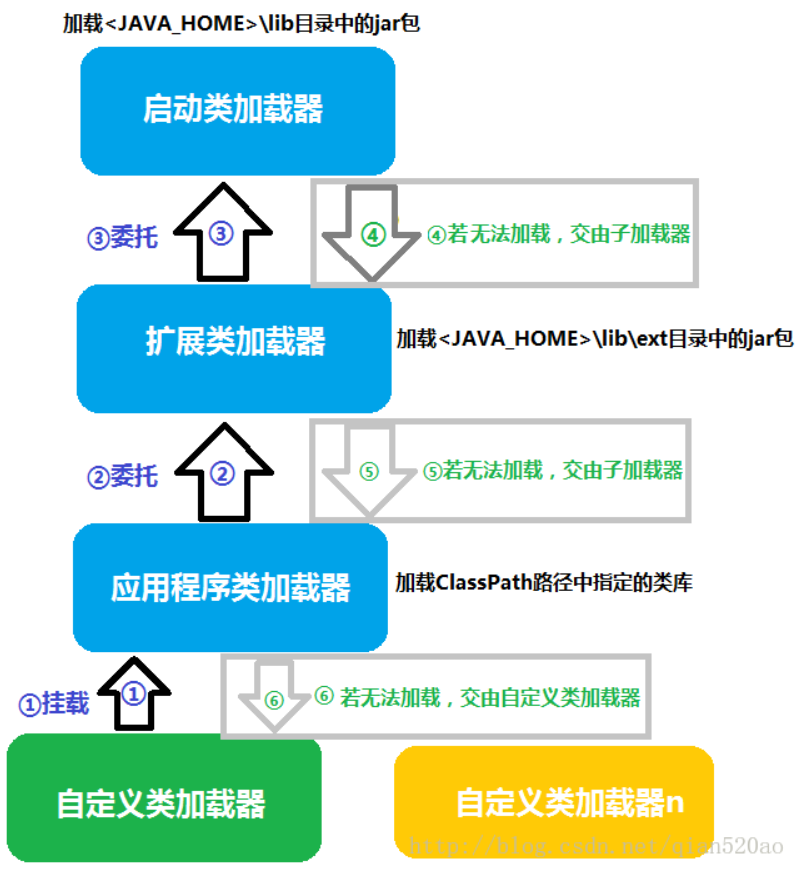
由sun.misc.Launcher$App-ClassLoader实现。由于这个类加载器是ClassLoader中的getSystemClassLoader()方法的返回值，所以一般也称它为系统类加载器。 它负责加载用户路径（ClassPath，即环境变量中配置的）上所指定的类库。如果程序中没有自定义类加载器，则这个就是程序默认的类加载器，开发者可以直接使用它。它是由Java编写，并继承自java.lang.ClassLoader。

* **类与类加载器**

类加载器虽然只是加载一个实现类，但是类加载器对于加载出来的这个类确有很大的影响：比较两个类是否“相等”，只有这两个类被同一个类加载器加载的前提下才有意义，否则，即使这两个类来源于同一个Class文件，但是被不同的类加载器加载，那么这两个类也必不相等（equals()方法，isAssignableForm()方法，isInstance()方法的返回结果），同时也就会影响instanceof关键字做对象所属关系判定情况。

* **双亲委派模型**

首先参考网络上一张图，这张图充分的展示了类加载的双亲委派模型：



**双亲委派模型工作流程**

当一个类加载器加载一个类的时候，它首先会把这个加载操作委托给父类加载器，它本身并不会去加载这个类，然后以此类推，一直将这个操作委托到启动类加载器，当启动类加载器在相应的目录下没有加载到相应的类，则将这个操作传递给他的子类加载器去加载，然后依次类推，直到子类加载器加载成功为止。当传递到发起者类加载器并且发起者类加载器也没有加载成功，则停止传递，抛出ClassNotFoundException异常。

注意：这里除了顶层的启动类加载器外，其他的加载器都应该有自己的福类加载器，这里说的类加载器之间的父子关系指的不是继承关系来实现，而是使用组合关系来复用另一个类加载器的代码而组成的父子关系。

**优点**

双亲委派模型使指定目录下的类只能由指定的类加载器去加载，保证了加载出来的类的唯一性，维护了Java程序的稳定运作。如果没有双亲委派模型，同一个类可以由多个类加载器加载，那么系统中会出现从同一个Class文件中加载出多个不同的类，那么Java程序会一片混乱。

实现双亲委派的代码存放在java.lang.ClassLoader类的loadClass()方法之中，源码如下：

****

* **自定义类加载器**

<https://blog.csdn.net/qian520ao/article/details/79087089>