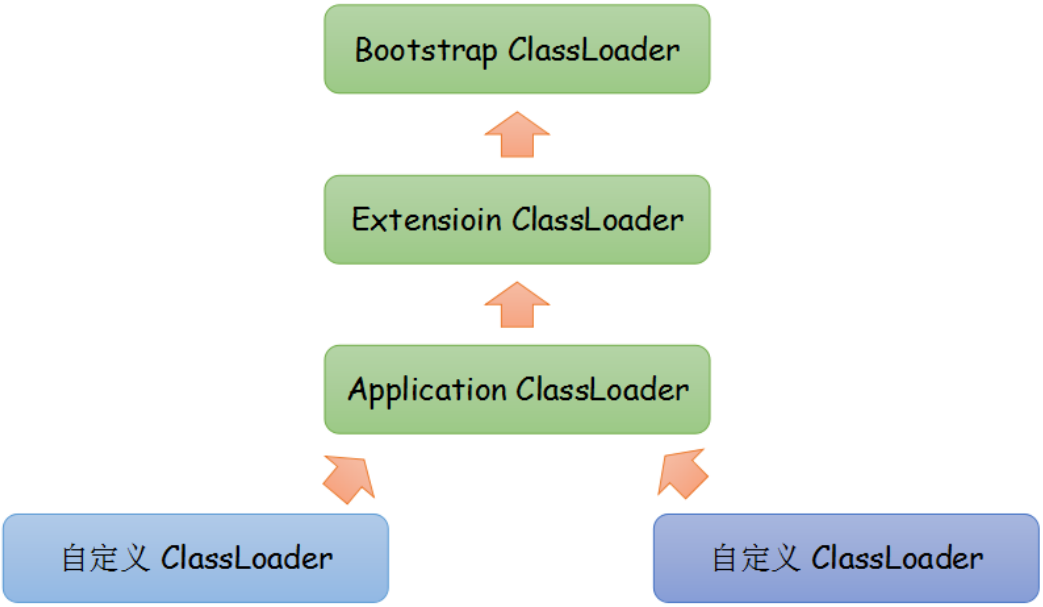
类加载器的功能：通过一个类的全限定名来获取描述此类的二进制字节流，以便让应用程序自己决定如何去获取所需要的类

1. 类与类加载器：
类加载器虽然只是实现了类加载动作，但是在java程序中起到的作用却远远不限于类加载阶段。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起在java虚拟机中的唯一性。即比较两个类是否“相等”，只有在这个两个类由同一个类加载器加载的前提下才又一次，否则即便是同一个类，由不同的加载器加载，那么这两个类也不相等【这里的相等指的是equals, isAssignableFrom以及inInstance方法】

2. 双亲委托模型：
从java虚拟机角度来看有两种类加载器：一种是启动类加载器（Bootstrap classloader），这个类加载器是由c++实现，是虚拟机自身的一部分，另一种就是其他的加载器，都是由java语言实现，独立于虚拟机外部，并且完全继承于java.lang.ClassLoader
从java程序员的角度，可以分的更细致一些，绝大java程序都会使用以来三种系统提供的类加载器
(1) 启动类加载器(BootStrap classLoader)：这个类加载器负责将存在java_home\lib目录中的或者被-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，名字不符合类库即使放在Lib下也不会被加载)，无法被java程序直接引用，用户在编写自定义类加载器的时候，如果需要把在家请求委托给引导类加载器，那么直接使用null代替即可
(2) 扩展类加载器(Extension ClassLoader)：这个类加载器由sun.misc.Launcher$ExtClassLoader实现，负责加载Java_Home\lib\ext中的，或者被Java.ext.dirs系统变量所指路径的类库，开发者可以直接使用扩展类加载器
(3) 应用程序类加载器(Application ClassLoader)：这个加载器由sun.misc.Launcher$AppClassLoader实现。由于这个类加载器是ClassLoader.getSystemClassLoader()方法的返回值，所以一般也称为系统类加载器，他负责记载用户类路径(classpath)上所指定的类库，可以直接使用，如果应用程序中没有自定义过自己的类加载器，一般情况下这个就是程序中默认的类加载器

双亲委托模型：



双亲委托模型要求除了顶层的启动类加载器外，其余的类加载器都应当有自己的父类加载器，这里的父子关系不是通过继承，而是通过组合实现的。

工作流程：

如果一个类加载器收到了类加载的请求，首先判断这个类有没有被加载，如果没有被加载则是把这个请求委托给父类去完成，每一层的类加载器都是如此。因此所有的加载请求都会送到最顶层的启动类加载器中，只有当父类加载器反馈无法完成加载这个请求时，才会有子类加载器去尝试自己完成。如果依旧无法加载则抛出ClassNotFoundException

这样的优点就是，java类随着它类加载器一起具备了一种带有优先级的层次关系，这样保证了无论那个类在各种类加载器环境中都是同一个类