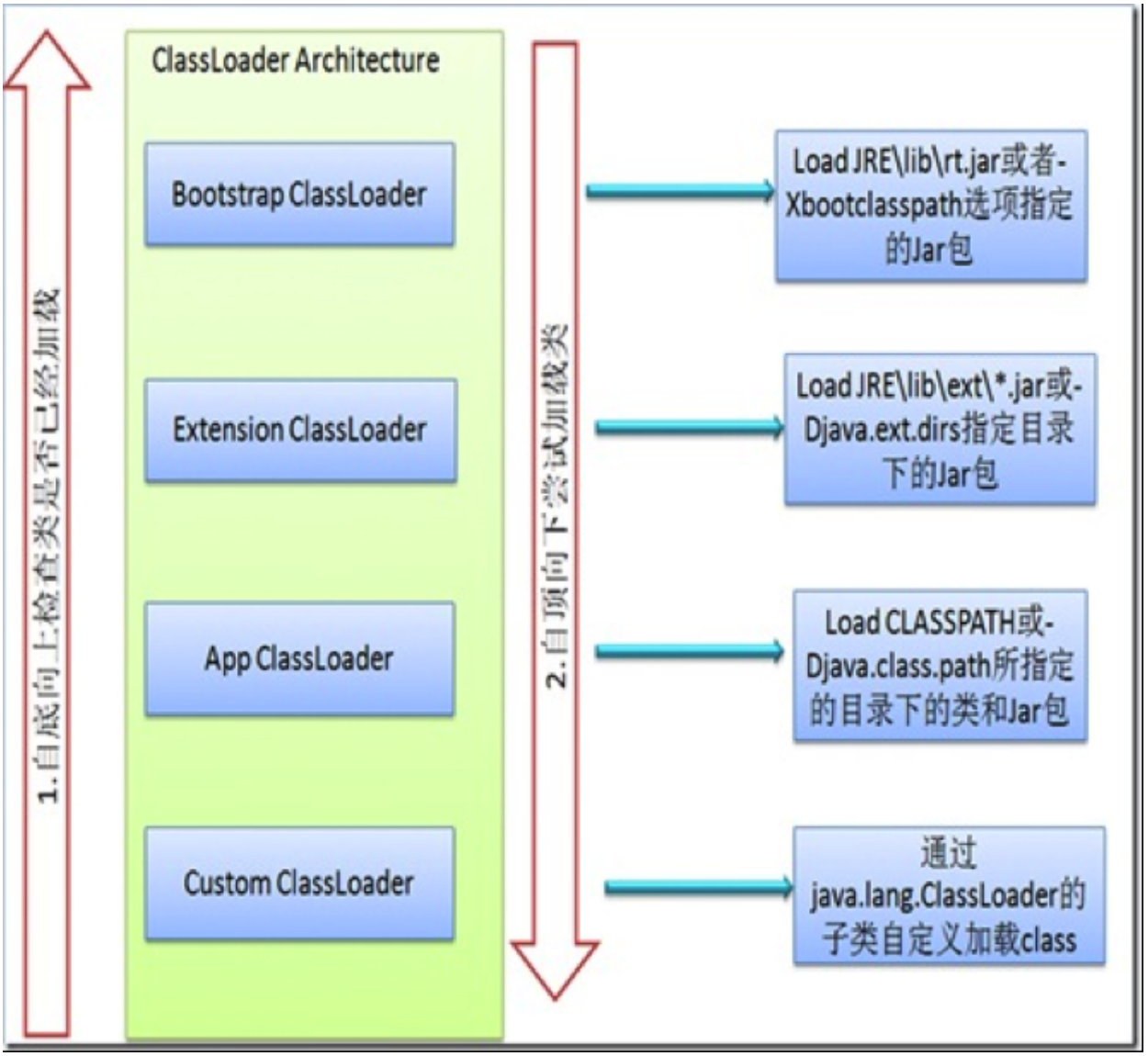
ClassLoader，即java类加载器，主要作用是将class加载到JVM内，同时它还指定class由谁来加载。ClassLoader就是用来动态加载class文件到内存当中用的。

"按需加载，类隔离，资源回收。"

类加载机制--->双亲委派模型（避免重复加载）优势就是避免Java核心API篡改。

每个ClassLoader实例都有一个父类加载器的引用（不是继承的关系，是一个包含的关系），虚拟机内置的类加载器（Bootstrap ClassLoader）本身没有父类加载器，但可以用作其它ClassLoader实例的的父类加载器。当一个ClassLoader实例需要**加载某个类时**，它会试图亲自搜索某个类之前，先把这个任务委托给它的父类加载器，这个过程是**由上至下依次检查**的，首先由最顶层的类加载器Bootstrap ClassLoader试图加载，如果没加载到，则把任务转交给Extension ClassLoader 试图加载，如果也没加载到，则转交给App ClassLoader 进行加载，如果它也没有加载得到的话，则返回给委托的发起者，由它到指定的文件系统或网络等URL中加载该类。如果它们都没有加载到这个类时，则抛出ClassNotFoundException异常。否则将这个找到的类生成一个类的定义，并将它加载到内存当中，最后返回**这个类在内存中的Class实例对象**。

JVM根据 类名+包名+ClassLoader实例ID 来判定两个类是否相同，是否已经加载过。JVM在判定两个class是否相同时，不仅要判断两个类名是否相同，而且要判断是否由同一个类加载器实例加载的。只有两者同时满足的情况下，JVM才认为这两个class是相同的。



java默认提供前三个ClassLoader。

BootStrapClassLoader启动类加载器　　最顶层的类加载器，由C++编写而成，内嵌到了JVM中，在JVM启动时会初始化该ClassLoader。主要用来**加载Java的核心类库**JRE/lib/rt.jar中所有的class文件，这个jar文件中包含了java规范定义的所有接口及实现。eg.rt.jar、resources.jar。加载完核心类库后，并构造Extension ClassLoader和App ClassLoader类加载器。

ExtensionClassLoader扩展类加载器　　用来读取Java的一些扩展类库，eg.读取JRE/lib/ext/*.jar中的包等（有些版本没有ext这个目录的）。

AppClassLoader系统类加载器　　用来读取CLASSPATH下指定的所有jar包或目录的类文件，一般情况下这个就是程序中默认的类加载器。

CustomClassLoader用户自定义　　用户自定义编写，用来读取指定类文件 。必须继承自java.lang.ClassLoader类。基于自定义的ClassLoader可用于加载非Classpath中（如从网络上下载的jar或二进制）的jar及目录、还可以在加载前对class文件优一些动作，如解密、编码等。

Extension ClassLoader和App ClassLoader继承自ClassLoader类

严格来说，ExtClassLoader的父类加载器是null，只不过在默认的ClassLoader的loadClass 方法中，当parent为null时，是交给BootStrapClassLoader来处理的，而且ExtClassLoader 没有重写默认的loadClass方法，所以，ExtClassLoader也会调用BootStrapLoader类加载器来加载，这就导致“BootStrapClassLoader具备了ExtClassLoader父类加载器的功能”。

三个重要的方法

loadClass　　classloader加载类的入口，此方法负责加载指定名字的类，ClassLoader的实现方法为先从已经加载的类中寻找，如没有则继续从父ClassLoader中寻找，如仍然没找到，则从BootstrapClassLoader中寻找，最后再调用findClass方法来寻找，如要改变类的加载顺序，则可覆盖此方法，如加载顺序相同，则可通过覆盖findClass来做特殊的处理，例如解密、固定路径寻找等，当通过整个寻找类的过程仍然未获取到Class对象时，则抛出ClassNotFoundException。如类需要resolve，则调用resolveClass进行链接。

findClass 此方法直接抛出ClassNotFoundException，因此需要通过覆盖loadClass或此方法来以自定义的方式加载相应的类。

defineClass 此方法负责将二进制的字节码转换为Class对象，这个方法对于自定义加载类而言非常重要，如二进制的字节码的格式不符合JVM Class文件的格式，抛出ClassFormatError；如需要生成的类名和二进制字节码中的不同，则抛出NoClassDefFoundError；如需要加载的class是受保护的、采用不同签名的或类名是以java.开头的，则抛出SecurityException；如需加载的class在此ClassLoader中已加载，则抛出LinkageError。

自定义类加载器