**双亲委派机制：（**[1.1 jvm核心类加载器--jdk源码剖析 - 盛开的太阳 - 博客园 (cnblogs.com)](https://www.cnblogs.com/ITPower/p/13197220.html)**）**

1、Java虚拟机对class文件采用的是**按需加载**的方式，也就是说当需要使用该类时才会将它的class文件加载到内存中成成class对象。而且加载某个类的class文件时，Java虚拟机采用的时双亲委派模式，即是把请求交由父类处理，它是一种任务委派模式。

2、问题引出:在项目中建一个java.lang包，然后在其中建造一个String类，现在在测试类中进行新建一个java.lang.String对象该对象是自定义的String还是Java核心API中的String。（事实证明是加载的Java核心API）

（**想想为什么？如果加载的不是加载的Java核心API，而是就近加载，首先可以保证一件事，你必然会使用到一个Java核心API，那么恶意攻击方是不是可以选择传送一个类似的自定义的同名Java核心API类给你，然后在该类中进行恶意修改，进行信息盗取，只要该类一加载进你的内存中，那么你的程序就废了**）

（那么Java 是怎么做到加载是Java核心API而不是自定义的？**这就是采用双亲委派机制，保证Java的安全性**）

**双亲委派机制的工作原理：（类加载请求的识别一般是通过全限定名与负责的加载路径进行比较）**

1、如果一个类加载器收到了类加载请求，它并不会自己先去加载，而是把这个请求委托给父亲的加载器去执行。

2、如果父类加载器还存在其父类加载器，则进一步向上委托，一次递归，请求最终将到达顶层的启动类加载器。

3、如果父类加载器可以完成类加载任务，就成功返回，倘若父类加载器无法完成此加载器任务子加载器才会尝试自己去加载，这就是双亲委派模式。



**双亲委派系统举例：**



双亲委派的优势：

1、避免类的重复加载

2、保护程序安全**，防止核心API被随意篡改（比如java.lang.String）**

3、线程上下文加载器打破双亲委派浅析（[浅谈双亲委派机制的缺陷及打破双亲委派机制 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/185612299)）（这里传统jdbc是通过java提供的DriveMager来获取第三方的具体Driver实例，也就是说明了Java核心类中使用了用户自定义类。）（[(1条消息) 关于Class.forName(“com.mysql.jdbc.Driver”)\_hong2511的博客-CSDN博客\_class.forname(driver)](https://blog.csdn.net/hong2511/article/details/81491193)）

**双亲委派之沙箱安全机制：**

1、自定义String类，但是在加载自定义String类时会率先使用引导类加载器加载，而引导类加载器的过程中回先加载jdk自带的文件。

2、（rt.jar包中java\lang\String.class），报错信息说没有main方法，就是因为加载的是rt.jar包中的String类。这样可以保证对Java核心源代码的保护，这就是沙箱安全机制。（**不被外界同名class替代**）（所谓沙箱就是一个封闭空间，在该空间中进行程序的运行能够保证运行的程序不会影响外界的程序运行，防止可能出现的病毒进行蔓延破坏。比如360沙箱，我们可以在里面运行U盘）

(参考文章：[java中的安全模型(沙箱机制)\_改变ing-CSDN博客\_沙箱安全机制](https://blog.csdn.net/qq_30336433/article/details/83268945))

**Java安全模型的核心就是Java沙箱（sandbox），什么是沙箱？沙箱是一个限制程序运行的环境。沙箱机制就是将 Java 代码限定在虚拟机(JVM)特定的运行范围中，并且严格限制代码对本地系统资源访问，通过这样的措施来保证对代码的有效隔离，防止对本地系统造成破坏。沙箱主要限制系统资源访问，那系统资源包括什么？——CPU、内存、文件系统、网络。不同级别的沙箱对这些资源访问的限制也可以不一样。所有的Java程序运行都可以指定沙箱，可以定制安全策略。**