

<https://blog.csdn.net/u013206465/article/details/47170253> (看评论)

## 类加载器:

类加载器就是java运行时环境(java Runtime Environment)的一部分, 负责动态加载Java类到Java虚拟机的内存空间中。意味着平常的.class文件就是通过这个类加载器加载到内存中的。

## 类加载器的种类以及作用:

### 1. Bootstrap ClassLoader

负责加载Java\_HOME中jre/lib/rt.jar里所有的class, 由C++实现, 不是ClassLoader子类

### 2. Extension ClassLoader

负责加载java平台中扩展功能的一些jar包, 包括JAVA\_HOME中jre/lib/\*.jar

### 3. App ClassLoader

负责加载classpath中指定的jar包及目录中class

### 4. Custom ClassLoader

属于应用程序根据自身需要自定义的ClassLoader

```
1 public String() {  
2     this.value = new char[0];  
3     System.out.println("=====");  
4 }
```

也就是说只要调用了我们自己写的String类的话应该是有输出的，接下来我们来试试：

```
1 import java.lang.String;  
2  
3 public class Test {  
4  
5     public static void main(String[] args) {  
6         String test = new String();  
7         test = "测试";  
8         System.out.println(test);  
9     }  
10  
11 }  
12  
13
```

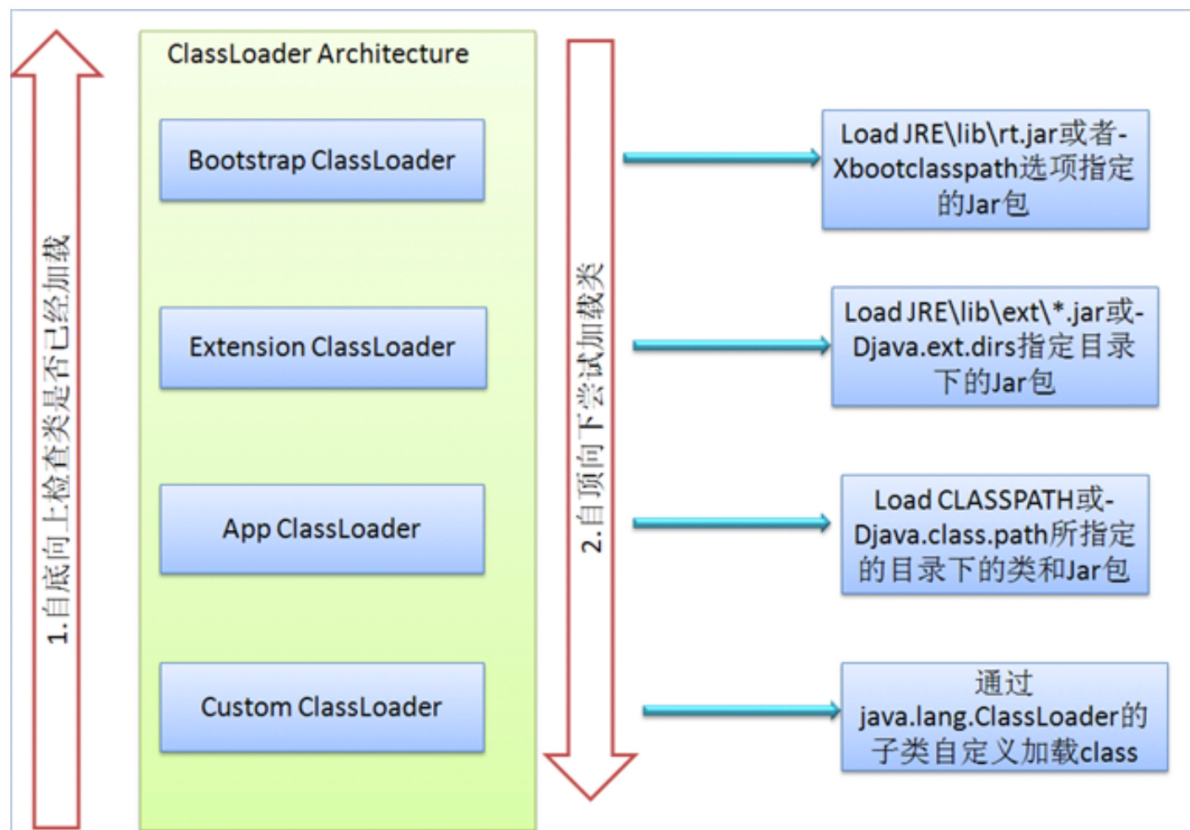
运行结果如下：

```
<已终止> Test ( 2 ) [Java 应用程序] D:\jre7\bin\javaw.exe ( 2015年7月31日 下午3:42:11 )  
1
```

可以看到调用的是系统的String类，没有输出。

解释：因为从jdk1.2开始，类的加载过程采用父亲委托机制。这种机制能更好的保证java平台的安全。在此委托机制中，除了java虚拟机自带的根类加载器以外，其余的类加载器都有且只有一个父类加载器。当java程序请求加载器loader1加载Sample类时，loader1首先委托自己的父加载器去加载Sample类，若父加载器能加载，则由父加载器完成加载任务，否则才由加载器loader1本身加载Sample类。

好吧~~~这下看明白了类加载器有个加载顺序我们来看一下这个加载顺序



加载过程中会首先检查类是否被加载，检查顺序是自底向上，从Custom ClassLoader到BootStrap ClassLoader逐层检查，只要某个ClassLoader已加载就被视为已加载此类，保证此类只被ClassLoader加载一次。而加载的顺序是自顶向下，也就是说当发现这个类没有的时候会先让自己的父类去加载，父类没有再让儿子去加载，那么在这个例子中我们自己的String应该是被Bootstrap ClassLoader加载了，所以App ClassLoader就不会再去加载我们自己写的String类了，导致我们写的类是没有被加载的。