[双亲委派模型 | 日拱一兵](https://dayarch.top/p/java-parents-delegation-model.html)

<https://dayarch.top/p/java-parents-delegation-model.html>

通俗易懂的双亲委派机制

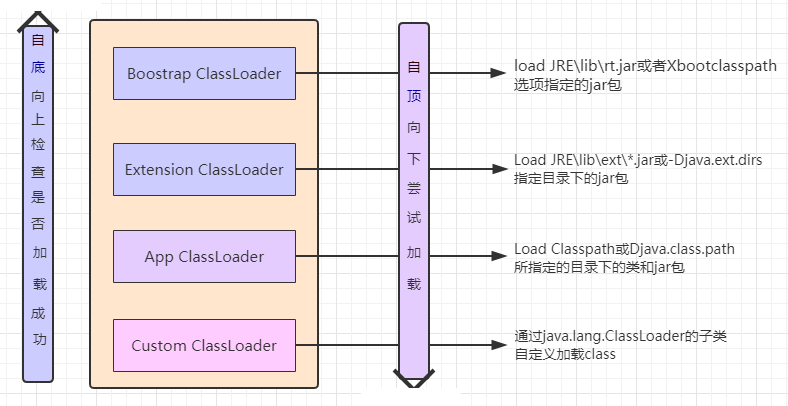
<https://blog.csdn.net/codeyanbao/article/details/82875064>

## 类加载器Classloader

ClassLoader翻译过来就是类加载器，一般开发时其实用得不多，但对于某些框架开发来说却非常常见；

ClassLoader的具体作用就是将class文件加载到jvm虚拟机中去，程序就可以正确运行了，但是jvm启动的时候，并不会一次性加载所有的class文件，而是根据需要动态加载，如果一次性加载那么多jar包那么多class，那内存消耗极大；

## 双亲委托机制



## Java语言自带有三个类加载器

**Bootstrap ClassLoader** 引导类加载器，主要加载核心类库，%JRE\_HOME%\lib下的rt.jar、resources.jar、charsets.jar和class等;

Bootstrap ClassLoader是C语言实现的；

**Extention ClassLoader** 扩展类加载器，加载目录%JRE\_HOME%\lib\ext目录下的jar包和class文件；

**Appclass Loader** 应用类加载器， 加载当前应用的classpath下的所有类；

那么这3个类加载器的加载顺序是怎么样的，具体哪个先行？顺序是：

1. Bootstrap CLassloder
2. Extention ClassLoader
3. AppClassLoader

我们自己写的类是由AppClassLoader加载，jdk api提供的那些类，比如String，Object，Date是由Bootstrap CLassloder加载；

System.***out***.println(Test.**class**.getClassLoader());  
System.***out***.println(String.**class**.getClassLoader());

每个类加载器都有一个父加载器，比如加载Test.class是由AppClassLoader完成，那么AppClassLoader也有一个父加载器，通过getParent方法获取；

ClassLoader cl = Test.**class**.getClassLoader();  
System.***out***.println(**"ClassLoader is:"**+cl.toString());  
System.***out***.println(**"ClassLoader's parent is:"**+cl.getParent());  
System.***out***.println(**"ClassLoader's parent is:"**+cl.getParent().getParent());

ExtClassLoader的父加载器是null（因为是C语言实现的，所以打不出名字）；