

原 JVM(四)——一道面试题搞懂JVM类加载机制

2017年07月05日 19:35:20

版权声明：本文为博主原创文章

有这样一道面试题：

```
1 class Singleton
2     private static int value1;
3     public static int value2;
4     public static int value3;
5
6     private Singleton() {
7         value1++;
8         value2++;
9     }
10
11     public static Singleton getInstance(){
12         return singleton;
13     }
14
15 }
16
17 class Singleton2{
18     public static int value1;
19     public static int value2 = 0;
20     private static Singleton2 singleton2 = new Singleton2();
21
22     private Singleton2(){
23         value1++;
24         value2++;
25     }
26 }
```



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

🗣 QQ客服 🗣 客服论坛

关于 招聘 广告服务 网站地图

©2018 CSDN版权所有 京ICP证09002463号

🐾 百度提供搜索支持



CSDN APP

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

JVM

jvm 类加载 破坏双亲模型 更多

_wgs/article/details/74489549

();

```
27     public static void main(String[] args) {
28         return 1;
29     }
30
31 }
32
33 public static void main(String[] args) {
34     Singleton1 singleton1 = new Singleton1();
35     System.out.println(singleton1.value1);
36     System.out.println(singleton1.value2);
37
38     Singleton2 singleton2 = new Singleton2();
39     System.out.println(singleton2.value1);
40     System.out.println(singleton2.value2);
41 }
42
```

说出运行的结果：

Singleton1 value1 : 1
Singleton1 value2 : 0
Singleton2 value1 : 1
Singleton2 value2 : 1

稍后会带来分析。

一 类加载机制

联系我们



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

💬 QQ客服 💬 客服论坛

[关于](#) [招聘](#) [广告服务](#) [网站地图](#)

©2018 CSDN版权所有 京ICP证09002463号

🐾 百度提供搜索支持



经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

CSDN APP

Singleton1 value1);

Singleton1 value2);

Singleton2();

Singleton2 value1);

Singleton2 value2);

5

4

JVM类加载分为5个过程

联系我们

加载
Loading



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

💬 QQ客服 💬 客服论坛

关于 招聘 广告服务 网站地图

©2018 CSDN版权所有 京ICP证09002463号

🔍 百度提供搜索支持



CSDN APP

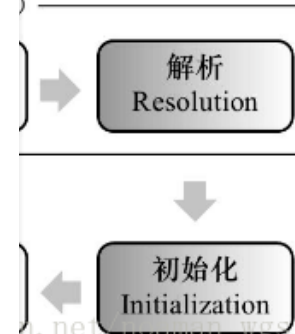
经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

}, 卸载, 如下图所示:



下面来看看加载, 验证,

1.1 加载

加载主要是将.class文件 (并不一定是.class。可以是ZIP包, 网络中获取) 中的二进制字节流读入到JVM中。

在加载阶段, JVM需要完成3件事:

- 1) 通过类的全限定名获取该类的二进制字节流;
- 2) 将字节流所代表的静态存储结构转化为方法区的运行时数据结构;
- 3) 在内存中生成一个该类的java.lang.Class对象, 作为方法区这个类的各种数据的访问入口。

1.2 连接

1.2.1 验证

验证是连接阶段的第一步, 主要确保加载进来的字节流符合JVM规范。

验证阶段会完成以下4个阶段的检验动作:

- 1) 文件格式验证
- 2) 元数据验证(是否符合Java语言规范)
- 3) 字节码验证 (确定程序语义合法, 符合逻辑)
- 4) 符号引用验证 (确保下一步的解析能正常执行)

5

4

1.2.2 准备

准备是连接阶段的第二步

1.2.3 解析

解析是连接阶段的第三步

1.3 初始化

初始化阶段是类加载过程注：

- 1) 当有父类且父类为初始
- 2) 再进行子类初始化语

什么时候需要对类进行初

- 1) 使用new该类实例化
- 2) 读取或设置类静态字段的时候（包括main方法调用的子段，在编译时就被放入常量池的静态字段除外static final）；
- 3) 调用类静态方法的时候；
- 4) 使用反射Class.forName(“xxxx”)对该类进行反射调用的时候，该类需要初始化；
- 5) 初始化一个类的时候，有父类，先初始化父类（注：1. 接口除外，父接口在调用的时候才会被初始化；2.子类引用父类静态字段，只会引发父类初始化）；
- 6) 被标明为启动类的类（即包含main()方法的类）要初始化；
- 7) 当使用JDK1.7的动态语言支持时，如果一个java.invoke.MethodHandle实例最后的解析结果REF_getStatic、REF_putStatic、REF_invokeStatic的方法句柄，并且这个方法句柄所对应的类没有进行过初始化，则需要先触发其初始化。

以上情况称为对一个类进行主动引用，且有且只要以上几种情况需要对类进行初始化。

再回过头来分析一开始的面试题：

Singleton输出结果：1 0

原因：

- 1 首先执行main中的Singleton singleton = Singleton.getInstance()；
- 2 类的加载：加载类Singleton
- 3 类的验证
- 4 类的准备：为静态变量分配内存，设置默认值。这里为singleton(引用类型)设置为null,value1,value2（基本数据类型）设置默认值0

联系我们



请扫描二维码联系客服
webmaster@csdn.net
400-660-0108
QQ客服 客服论坛

关于 招聘 广告服务 网站地图
©2018 CSDN版权所有 京ICP证09002463号
百度提供搜索支持



经营性网站备案信息
网络110报警服务
中国互联网举报中心
北京互联网违法和不良信息举报中心

CSDN APP

设置默认初始值。

直接引用的过程。

可主动为类变量赋值。

5

4

5 类的初始化（按照赋值顺序）
执行private static Singleton singleton = new Singleton();
执行Singleton的构造器：
执行

```
public static int value1 = 1;  
public static int value2 = 0;
```

此时value1=1, value2=0

联系我们



请扫描二维码联系客服

 webmaster@csdn.net

 400-660-0108

 QQ客服  客服论坛

关于 招聘 广告服务 网站地图

©2018 CSDN版权所有 京ICP证09002463号

 百度提供搜索支持



CSDN APP

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

Singleton2输出结果：1
原因：

1 首先执行main中的Singleton2
2 类的加载：加载类Singleton2
3 类的验证
4 类的准备：为静态变量value1,value2分配空间
5 类的初始化（按照赋值顺序）
执行

```
public static int value2 = 0;
```

此时value2=0(value1不变，依然是0);
执行

```
private static Singleton singleton = new Singleton();
```

执行Singleton2的构造器：value1++;value2++;
此时value1, value2均等于1,即为最后结果

1	5
2	4
3	
4	
5	
6	
7	
8	
9	
10	

基本数据类型) 设置默认值0,singleton2(引用类型)设置为null,

二 类加载器

类加载器实现的功能是即为加载阶段获取二进制字节流的时候。

JVM提供了以下3种系统的类加载器：

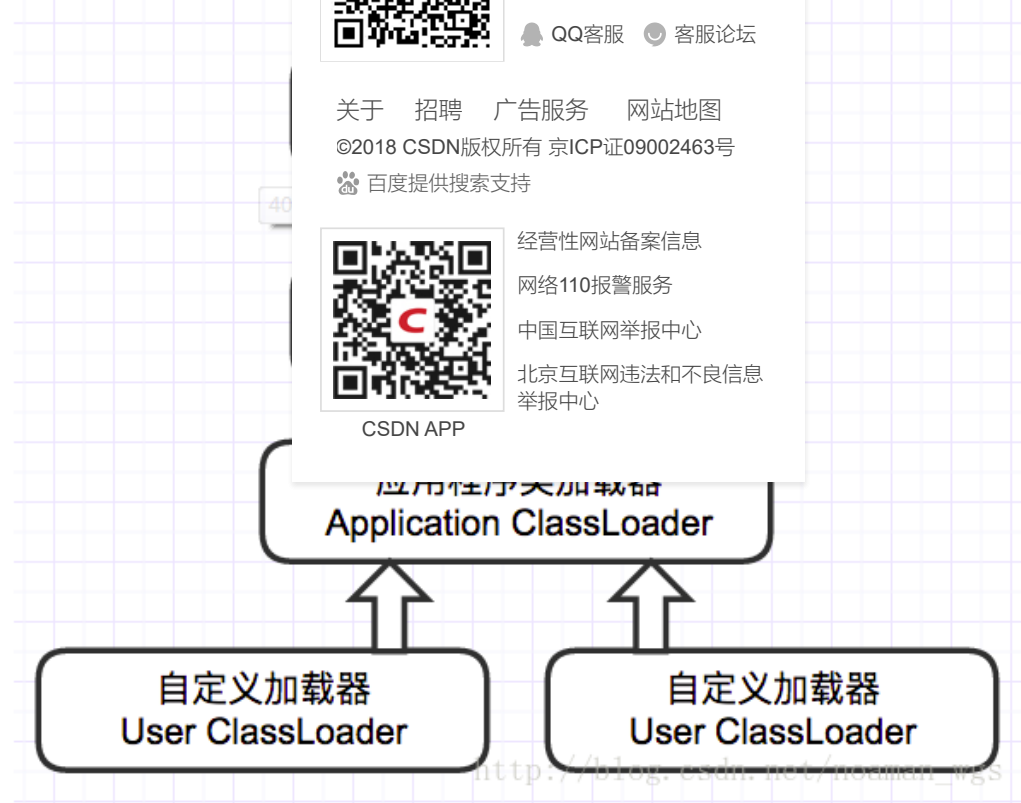
- **启动类加载器 (Bootstrap ClassLoader)**：最顶层的类加载器，负责加载 JAVA_HOME\lib 目录中的，或通过-Xbootclasspath参数指定路径中的，且被虚拟机认可（按文件名识别，如rt.jar）的类。
- **扩展类加载器(Extension ClassLoader)**：负责加载 JAVA_HOME\lib\ext 目录中的，或通过java.ext.dirs系统变量指定路径中的类库。

https://blog.csdn.net/noaman_wgs/article/details/74489549

5/16

- 应用程序类加载器(Application ClassLoader)
一般这个就是默认类加载器

类加载器之间的层次关系



可以通过getSystemClassLoader()获取，负责加载用户路径（classpath）上的类

如果没有自定义类加载器，一

5
4

照片来源：<http://www.importnew.com/25295.html>

类加载器之间的这种层次关系叫做双亲委派模型。

双亲委派模型要求除了顶层的启动类加载器（Bootstrap ClassLoader）外，其余的类加载器都应当有自己的父类加载器。这里的类加载器之间的父子关系一般不是以继承关系实现的，而是用组合实现的。

双亲委派模型的工作过程

如果一个类接受到类加载器 (ClassLoader)。

只有当父类加载器无法加载时，才会由子类加载器加载。

双亲委派模型的代码实现

双亲委派模型的代码实现如下：

- 1) 首先检查类是否被加载。
- 2) 若父类加载器为空，则直接使用启动类加载器加载。
- 3) 若父类加载失败，则委派给父类加载器加载。

loadClass源代码如下：

```

1  protected synchronized
2      //1 首先检查类是否被加载
3      Class c = findClass(name);
4      if (c == null) {
5          try {
6              if (parent != null) {
7                  //2 没有则调用父类加载器的loadClass()方法；
8                  c = parent.loadClass(name, false);
9              } else {
10                 //3 若父类加载器为空，则默认使用启动类加载器作为父加载器；
11                 c = findBootstrapClass0(name);
12             }
13         } catch (ClassNotFoundException e) {
14             //4 若父类加载失败，抛出ClassNotFoundException 异常后
15             c = findClass(name);
16         }
17     }
18     if (resolve) {
19         //5 再调用自己的findClass() 方法。
20         resolveClass(c);
21     }
22     return c;
23 }

```

联系我们



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

💬 QQ客服 🗣 客服论坛

关于 招聘 广告服务 网站地图

©2018 CSDN版权所有 京ICP证09002463号

🔍 百度提供搜索支持



CSDN APP

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

将这个类加载请求委派给父类加载器，这样一层一层传送，直到到达启动类加载器 (Bootstrap ClassLoader)。

()方法当中。

方法；

用自己的findClass() 方法。

```
boolean resolve) throws ClassNotFoundException {
```

破坏双亲委派模型

联系我们

双亲委派模型很好的解决了若加载的基础类中需要用到父类的问题。下面介绍两个例子来讲解

1. JNDI破坏双亲委派

JNDI是Java标准服务。为了解决这个问题，利用线程上下文类

2. Spring破坏双亲委派

Spring要对用户程序类加载器加载。那么Spring是如何使用线程上下文类

ClassLoader类加载器（对应Tomcat中的WebAppClassLoader类加载器）：setContextClassLoader(AppClassLoader)。利用这个来加载用户程序。即任何一个线程都可通过getContextClassLoader()获取到WebAppClassLoader。

三 附上Tomcat类加载架构：



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

💬 QQ客服 💬 客服论坛

[关于](#) [招聘](#) [广告服务](#) [网站地图](#)

©2018 CSDN版权所有 京ICP证09002463号

🐾 百度提供搜索支持



CSDN APP

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

即越基础的类由越上层的加载器进行加载。

只别这些用户代码，怎么办呢？这时就需要破坏双亲委派模型了。

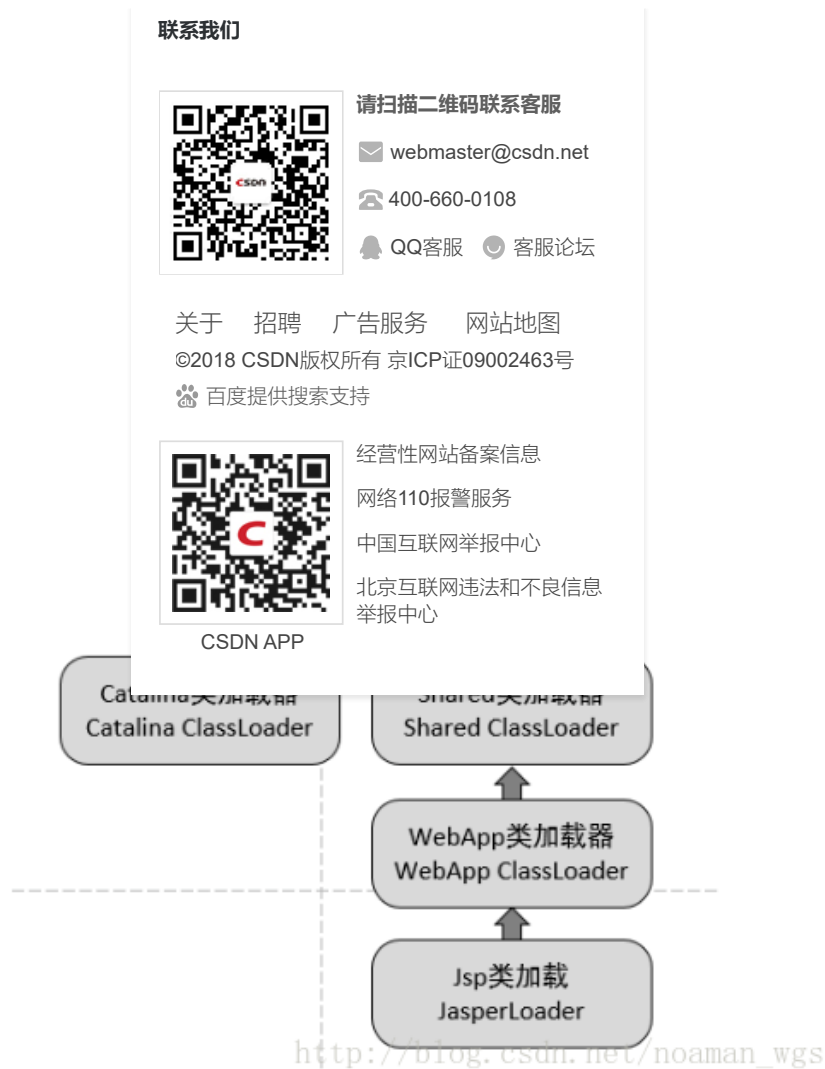
是JNDI需要回调独立厂商实现的代码，而类加载器无法识别这些回调代码（如JNDI的lookup()）。

通过Thread.setContextClassLoader()设置。

加载器请求子类加载器去完成类加载的过程，而破坏了双亲委派模型。

WEB-INF目录下，由WebAppClassLoader类加载器加载，而Spring由ClassLoader类加载器或Shared类加

都是通过Thread.currentThread().getContextClassLoader()获取的。当线程创建时会默认创建一个App



5

4

(图片来源: <http://lib.csdn.net/article/java/60356>)


Tomcat目录下有4组目录:


- /common目录下: 类库可以被Tomcat和Web应用程序共同使用; 由 Common ClassLoader类加载器加载目录下的类库;
- /server目录: 类库只能被Tomcat可见; 由 Catalina ClassLoader类加载器加载目录下的类库;
- /shared目录: 类库对所有Web应用程序可见, 但对Tomcat不可见; 由 Shared ClassLoader类加载器加载目录下的类库;

- /WebApp/WEB-INF目
- 每一个JSP文件对应一

参考：《深入理解Java虚拟机》
MyBlog: <https://nomicon.com>
2017/07/05 In NJ

Python正确的学习路线
如何从8K提至20K月薪，你

想对作者说点什么

Iq凌风：例子特别好，

联系我们



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

🗣 QQ客服 🗣 客服论坛

关于 招聘 广告服务 网站地图

©2018 CSDN版权所有 京ICP证09002463号

🔍 百度提供搜索支持



经营性网站备案信息

网络110报警服务



中国互联网举报中心


北京互联网违法和不良信息举报中心

CSDN APP


ClassLoader类加载器加载目录下的类库；

3B%E5%8A%A0%E8%BD%BD%E6%9C%BA%E5%88%B6/


- KingWang_WHU：上面的Singleton输出结果：1 1，整个的分析应该错了吧，执行 public static int value1; public static int value2 = 0; 之后，应该是value1=0,value2=0 (03-17 22:59 #1楼) [收起回复](#)
- 是Guava不是瓜娃 回复 KingWang_WHU：已经改正错误，感谢提醒！ (03-18 12:32)

面试官：请你谈谈Java的类加载过程  5084

刚刚走出校门的应届毕业生，如果在去寻求一份Java开发的工作时，你的面试官很有可能一边看着你的简历，一边漫不经心地问你：了...

JVM类加载机制-思维导图版(65题)  105

转载请注明链接：介绍JVM类加载机制相关的内容，包括7大阶段，双亲委派模型等内容。JVM类加载机制 版本：2018/8/14-1(19:16...

今天做到一道面试题：JVM的工作原理  2849

JVM的工作原理 原文链接：jamesdbloom 翻译：ImportNew.com - 挖坑的张师傅 译文链接：http://www.importnew.com/17770.html ...

联系我们



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

💬 QQ客服 💬 客服论坛

关于 招聘 广告服务 网站地图

©2018 CSDN版权所有 京ICP证09002463号

 百度提供搜索支持



CSDN APP

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心



程序员老司机教你

《深入理解JVM》--J

概述：虚拟机的类加载机制主要是指：虚拟机把描述类的数据从Class文件加载到内存，并对数据进行校验，转换解析和初始化，最终形...

Jvm类加载机制详解--类加载的几个阶段
虚拟机把描述类的数据从Class文件加载到内存，并对数据进行校验、转换解析和初始化，最终形成可以被虚拟机直接使用的Java类型， ...

JVM类加载机制
JVM加载class文件 定义 JVM将编译生成的.class文件按照需求和一定的规则加载到内存中，并组织成为一个完整的Java应用程序。这个...

JVM类加载机制详解（二）类加载器与双亲委派模型
在上一篇JVM类加载机制详解（一）JVM类加载过程中说到，类加载机制的第一个阶段加载做的工作有： 1、通过一个类的全限定名（...

相关热词 jvm中跑jvm jvm语言 多jvm jvm开发语言 安卓的jvm

2009

lass文件中加载到内存中，并对数据进行校验、解析和初始...

2438

垃圾回收GC原理、JVM内存...
长的内容拿来认真学习一遍，整理成自己能够快速消化的提...

3.2万

钱？类加载的时机？ 2、什么是类初始化？什么时候进行类...

976

335

224

1.3万

深入理解Java类加载器

【版权申明】未经博主同意

个人资料



是Guava不是瓜娃

原创 200 粉丝 430 喜欢 286

等级: 博客 5 访问: 积分: 4591 排名: 勋章: 恒

联系我们



请扫描二维码联系客服
webmaster@csdn.net
400-660-0108
QQ客服 客服论坛

关于 招聘 广告服务 网站地图
©2018 CSDN版权所有 京ICP证09002463号
百度提供搜索支持



CSDN APP

经营性网站备案信息
网络110报警服务
中国互联网举报中心
北京互联网违法和不良信息举报中心

联系方式

吴操的Blog: 吴操的Blog
My Github: My github
我的公众号:

7.3万

http://blog.csdn.net/javazejian/article/details/73413292 出...

5

4

我的公众号中会定时发送一些Java技术文章，欢迎关注！

联系我们



微信扫一扫
关注该公众



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

💬 QQ客服 💬 客服论坛

[关于](#) [招聘](#) [广告服务](#) [网站地图](#)

©2018 CSDN版权所有 京ICP证09002463号

🐾 百度提供搜索支持



CSDN APP

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

最新文章

记录一次MySQL死锁的分析与解决过程

Maven中 jar包冲突原理与解决办法

定时任务框架Quartz-(一)Quartz入门与Demo搭建

Spring原理与源码分析系列（七） - Spring AOP实现过程与实战

Spring原理与源码分析系列（六） - Spring AOP入门与概述

博主专栏



Spring Boot学习笔记

阅读量：1167 3 篇

5

4



Spring原理与源码

阅读量：2385

个人分类

剑指offer笔试题

转载文章

Java面试题

JavaWeb

数据库

展开

归档

2018年9月

2018年7月

2018年3月

2018年1月

2017年12月

展开

热门文章

Dubbo入门---搭建一个最简单的Demo框架

阅读量：294760

Intellij IDEA中使用MyBatis-generator 自动生成MyBatis代码

阅读量：23773

SpringMVC中出现”HTTP Status 405 - Request method 'PUT' not supported”

联系我们



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

💬 QQ客服 💬 客服论坛

关于 招聘 广告服务 网站地图

©2018 CSDN版权所有 京ICP证09002463号

🐾 百度提供搜索支持



CSDN APP

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

5
4

https://blog.csdn.net/noaman_wgs/article/details/74489549

15/16

阅读量: 12711

Redis应用3-基于Redis消息队
操作

阅读量: 10471

Echarts图表中动态数据显示

阅读量: 8501

最新评论

Dubbo入门---搭建一个最简单

ght886: 感谢分享

Java SSM练手小项目-手把手.

noaman_wgs: [reply]qq_3720029

什么意思, 可以私信我

Java SSM练手小项目-手把手.

qq_37200290: [reply]noaman_w

目名的, 我又新建了一个项目把文

Java SSM练手小项目-手把手...

noaman_wgs: [reply]qq_37200290[/reply] http://l

ocalhost:8080/...

Java SSM练手小项目-手把手...

qq_37200290: [reply]noaman_wgs[/reply]就是点

完登陆没反应, 只能显示登录页面, f12里输...

联系我们



请扫描二维码联系客服

 webmaster@csdn.net

 400-660-0108

 QQ客服

 客服论坛

关于 招聘 广告服务 网站地图

©2018 CSDN版权所有 京ICP证09002463号

 百度提供搜索支持



CSDN APP

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

5
4