

深入理解单例模式：静态内部类单例原理

原创 置顶 走着不语 最后发布于2018-05-26 02:00:54 阅读数 39099 ☆ 收藏

分类专栏： JAVA进阶

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。
本文链接：<https://blog.csdn.net/mnb65482/article/details/80458571>

本文主要介绍java的单例模式，以及详细剖析静态内部类之所以能够实现单例的原理。OK，废话不多说，进入正文。

首先我们要先了解下单例的四大原则：

- 1.构造私有。
- 2.以静态方法或者枚举返回实例。
- 3.确保实例只有一个，尤其是多线程环境。
- 4.确保反序列换时不会重新构建对象。

我们常用的单例模式有：

饿汉模式、懒汉模式、双重锁懒汉模式、静态内部类模式、枚举模式，我们来逐一分析下这些模式的区别。

1.饿汉模式：

```
1 public class Singleton{
2     private static Singleton INSTANCE = new Singleton();
3     private Singleton(){}
4     public static Singleton getInstance(){ return INSTANCE; }}
```

饿汉模式在类被初始化时就已经在内存中创建了对象，以空间换时间，故不存在线程安全问题。

2.懒汉模式：

```
1 public class Singleton{
2     private static Singleton INSTANCE = null;
3     private Singleton(){}
4     public static Singleton getInstance() {
5         if(INSTANCE == null){
6             INSTANCE = new Singleton();
7         }
8         return INSTANCE ;
9     }
10 }
```

懒汉模式在方法被调用后才创建对象，以时间换空间，在多线程环境下存在风险。

3.双重锁懒汉模式(Double Check Lock)

```
1 public class Singleton{
2     private static Singleton INSTANCE = null;
3     private Singleton(){}
4     public static Singleton getInstance(){if(INSTANCE == null){
5         synchronized(Singleton.class){
6             if(INSTANCE == null){
7                 INSTANCE = new Singleton();
8             }
9         }
10         return INSTANCE;
11     }
12 }
13 }
```

👍 65

🔗

💬 38

☆

📱

<

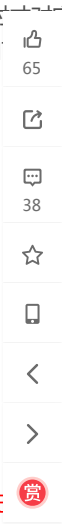
>

👍

🔊

举报

DCL模式的优点就是，只有在对象需要被使用时才创建，第一次判断 `INSTANCE == null` 为了避免非必要加锁，当第一次加载时实例进行加锁再实例既可以节约内存空间，又可以保证线程安全。但是，由于jvm存在乱序执行功能，DCL也会出现线程不安全的情况。具体分析如



```
INSTANCE = new Singleton();
```

这个步骤，其实在jvm里面的执行分为三步：

- 1.在堆内存开辟内存空间。
- 2.在堆内存中实例化Singleton里面的各个参数。
- 3.把对象指向堆内存空间。

由于jvm存在乱序执行功能，所以可能在2还没执行时就先执行了3，如果此时再被切换到线程B上，由于执行了3，`INSTANCE` 空了，会被直接实例化的话，就会出现异常。这个就是著名的DCL失效问题。

不过在JDK1.5之后，官方也发现了这个问题，故而具体化了`volatile`，即在JDK1.6及以后，只要定义为`private volatile static Singleton INSTANCE` 决DCL失效问题。`volatile`确保`INSTANCE`每次均在主内存中读取，这样虽然会牺牲一点效率，但也无伤大雅。

3.静态内部类模式：

```
1 public class Singleton{
2     private Singleton(){}
3
4     private static class SingletonHoler{
5         private static Singleton INSTANCE = new Singleton();
6     }
7
8     public static Singleton getInstance(){
9         return SingletonHoler.INSTANCE;
10    }
11 }
```

静态内部类的优点是：外部类加载时并不需要立即加载内部类，内部类不被加载则不去初始化`INSTANCE`，故而不占内存。即当Singleton第一次被加载要去加载SingletonHoler，只有当`getInstance()`方法第一次被调用时，才会去初始化`INSTANCE`，第一次调用`getInstance()`方法会导致虚拟机加载Singleton，这种方法不仅能确保线程安全，也能保证单例的唯一性，同时也延迟了单例的实例化。

那么，静态内部类又是如何实现线程安全的呢？首先，我们先了解下类的加载时机。

类加载时机：JAVA虚拟机在有且仅有的5种场景下会对类进行初始化。

- 1.遇到new、getstatic、setstatic或者invokestatic这4个字节码指令时，对应的java代码场景为：new一个关键字或者一个实例化对象时、读取或设置时(final修饰、已在编译期把结果放入常量池的除外)、调用一个类的静态方法时。
 - 2.使用java.lang.reflect包的方法对类进行反射调用的时候，如果类没进行初始化，需要先调用其初始化方法进行初始化。
 - 3.当初始化一个类时，如果其父类还未进行初始化，会先触发其父类的初始化。
 - 4.当虚拟机启动时，用户需要指定一个要执行的主类(包含main()方法的类)，虚拟机会先初始化这个类。
 - 5.当使用JDK 1.7等动态语言支持时，如果一个java.lang.invoke.MethodHandle实例最后的解析结果REF_getStatic、REF_putStatic、REF_invokeStatic，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。
- 这5种情况被称为是类的主动引用，注意，这里《虚拟机规范》中使用的限定词是"有且仅有"，那么，除此之外的所有引用类都不会对类进行初始化，被动引用。静态内部类就属于被动引用的行列。

我们再回头看下`getInstance()`方法，调用的是`SingletonHoler.INSTANCE`，取的是SingletonHoler里的`INSTANCE`对象，跟上面那个DCL方法不同的是`getInstance()`方法并没有多次去new对象，故不管多少个线程去调用`getInstance()`方法，取的都是同一个`INSTANCE`对象，而不会重新创建。当方法被调用时，SingletonHoler才在Singleton的运行时常量池里，把符号引用替换为直接引用，这时静态对象`INSTANCE`也真正创建，然后再被get方法返回出去，这点同饿汉模式。那么`INSTANCE`在创建过程中又是如何保证线程安全的呢？在《深入理解JAVA虚拟机》中，有这句话：



虚拟机会保证一个类的`<clinit>()`方法在多线程环境中被正确地加锁、同步，如果多个线程同时去初始化一个类，那么只有一个线程去执行这个类的这个方法，其他线程都需要阻塞等待，直到活动线程执行`<clinit>()`方法完毕。如果在一个类的`<clinit>()`方法中有耗时很长的操作，就可能造成多个进程阻塞

是，其他线程虽然会被阻塞，但如果执行<clinit>()方法后，其他线程唤醒之后不会再次进入<clinit>()方法。同一个加载器下，一个类型只会初始化一次应用中，这种阻塞往往是很隐蔽的。

故而，可以看出INSTANCE在创建过程中是线程安全的，所以说静态内部类形式的单例可保证线程安全，也能保证单例的唯一性，但这也延迟了单例的创建。

那么，是不是可以说静态内部类单例就是最完美的单例模式了呢？其实不然，静态内部类也有着一个致命的缺点，就是传参的问题，故外部无法传递参数进去，例如Context这种参数，所以，我们创建单例时，可以在静态内部类与DCL模式里自己斟酌。

最后粗略的介绍下枚举类型的单例吧。

枚举单例：

```
1 public enum Singleton{
2     INSTANCE;
3     public void method(){
4         //TODO
5     }
6 }
```

枚举在java中与普通类一样，都能拥有字段与方法，而且枚举实例创建是线程安全的，在任何情况下，它都是一个单例。我们可直接以

Singleton.INSTANCE

的方式调用。

参考资料：

- 《深入理解JAVA虚拟机》
- 《Android源码设计模式解析与实战》
- 《java虚拟机规范》

点赞 65

收藏


分享

...

走着不语

发布了7 篇原创文章 · 获赞 71 · 访问量 5万+

私信



主机为什么可以那么便宜

主机便宜

想对作者说点什么

OnlyPiglet 6个月前

作者你好 我认为volatile 此处生效的原因 并不是 变量每次都从主存中获取最新变量,因为变量地址已经固定了 再怎么从主存中获取不都是null 原因应该是 volatile 的p告知编译器的在标记的变量前后不使用优化功能 就是说 1.在堆内存开辟内存空间。 2.在堆内存中实例化Singleton里面的各个参数。 3.把对象指向堆内存空间。 1,2,3化 ,所以才避免DCL 失效

u010465325 1天前

DCL失效问题失效是因为指令乱序，前文已经说了，然后volatile可以解决DCL失效的问题，那么volatile应该是可以解决指令乱序的问题，才能解决这个问题，而不是主内存解决的。文章在此处因果不明，逻辑不通。

Lamborrt 3个月前

学习了!

登录查看 38 条热评

举报

静态内部类实现单例模式

阅读数 779

StaticInnerClass静态内部类的外部调用静态内部类可以直接创建对象new B.C();如果内部类不是静态的，那就得这样... 博文 来自： hqw11的博客

单例模式和静态类的区别对比

阅读量 6182

什么是单例模式单例模式指的是在应用***整个生命周期内只能存在一个实例。***单例模式是一种被广泛使用的设计... 博文 来自： baidu_41878679...

静态类实现单例模式

public class Singleton { private Singleton() { } static cla...

博文 来自： qq_14955 博客

Java笔记之单例模式及原理

单例设计模式

博文 来自： stephen...



海外cdn加速哪家好
免费的cdn加速

单例模式的几种用法比较

最近在看何红辉、关爱民著的《Android源码设计模式解析与实战》，一边学习，一边理解，一边记笔记。1.定义确... 博文 来自： 旭日的芬芳...

单例模式简单原理

阅读量 3568

单例模式：保证一个类有且仅有一个实例，并提供一个访问它的全局访问点。以下为简单的模拟实现：public class S... 博文 来自： sssbbryj的专栏

【设计模式一】单例模式，静态内部类单例，枚举单例

阅读量 1014

单例模式的优点： 1.在内存中只有一个对象，节约内存 2.避免频繁的创建和销毁对象，可以提高性能 3.避免对共享资... 博文 来自： yu540135101的博客

Java单例---静态内部类

阅读量 281

之前写过一篇双重锁校验单例，这是延迟加载的一种单例模式，俗称懒汉模式，这次写一个静态内部类的单例，这个... 博文 来自： 影公子的博客

静态内部类写单例的好处

阅读量 691

以前写单例这样的public class Demo { private static Demo instance; private Demo() {} public static D... 博文 来自： xiexiaotian11的博客



软文找写手,80一篇
软文发稿网站

单例的静态内部类实现方式

阅读量 70

网上虽然有单例的七八种实现方式，但是说实话我有点理解不了写这些博客的人的想法，既然其它模式要么是多线程... 博文 来自： 西雅图的风的博客

设计模式----单例模式的实现以及实现原理

阅读量 264

一.简介单例模式是一个十分常见的设计模式，一个对象实例如果在创建的过程十分消耗资源，且整个 app 系统只需... 博文 来自： 海盗的帽子的博客



hqw11
49篇文章
排名:千里之外



南故笙烟归期何夕
53篇文章
排名:千里之外



公众号火炎一笑倾城
99篇文章
排名:千里之外



子沐、一念
33篇文章
排名:千里之外

设计模式--单例模式原理

阅读量 1144

本文继续介绍23种设计模式系列之单例模式。概念： Java中单例模式是一种常见的设计模式，单例模式的写法有... 博文 来自： xuehuagongzi000...

单例模式之静态内部类实现

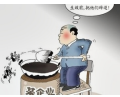
阅读量 297

单例模式，是一种常用的软件设计模式。在它的核心结构中只包含一个被称为单例的特殊类。通过单例模式可以保证... 博文 来自： qq_35939148的博客

JAVA三种实现单例模式方法（二）：使用静态内部类实现单例设计模式

阅读量 2238

静态代码块和静态内部类的加载顺序：当调用外部类的构造函数是，外部类的静态代码块同时被加载，但是其内部类... 博文 来自： Venus的专栏



工位出租600元/月
租工位办公室

使用静态内部类实现单例

单例模式之 利用静态类实现单例利用静态类在jvm内存模型中存储在静态块且只有一个实例的属性，可以轻松实现单... 博文 来自： 马以的专...

169

单例模式(静态内部类实现)

静态内部类实现方式(也是一种懒加载方式) 要点：外部类没有static属性，则不会像饿汉式那样立即加载对象只有真... 博文 来自： qq_35289747的博客

767

请教个内部类实现的单例模式，访问不到内部类的问题。

有一个类，用了单例模式：用内部类实现static初始化。这个类同时实现了一个接口。接口声明的时候，调用static 的get...

65

单例模式 种静态方法有点搞不懂 求帮助

public static readonly Static_Singleton instance static Static_Singleton() { instance = new Static_S

38

静态方法、单例模式区别

关于这个问题，下面是一些同仁的观点：观点一：（单例）单例模式比静态方法有很多优势：首先，单例可以继承类... 博文 来自： sooxin的私人博客

21

雷达测速仪 雷达测速仪价格 报价

激光多普勒测速仪多钱

单例模式的五种写法

设计模式（Designpattern），提供了在软件开发过程中面临的一些问题的最佳解决方案，是Java开发者必修的一门... 博文 来自： absolute_chen的...

1万+

java记录-单例模式的原理

1.单例模式是什么意思：我们自己定义的类，默认情况下是可以被外界的代码随意创建任意多个实例的，但是有时候... 博文 来自： fzh595408240的...

142

双检锁DCL与静态内部类实现单例模式的原理解析

双检锁实现单例模式将上锁粒度降低到了仅仅是初始化实例的那部分，保证线程安全，提高执行效率。双检锁的机制... 博文 来自： 马迪奥的小窗

382

为什么静态内部类的单例模式是线程安全的？

public class Singleton { private Singleton() {} private static class SingletonHolder{ private sta... 博文 来自： ren421259121的...

847

【java_设计模式】单例模式 静态内部类实现线程安全

学习路径：https://coding.imooc.com/learn/list/270.html前言加锁可以解决饿汉式单例模式的线程安全。【饿汉... 博文 来自： chenghan_yang的...

176

单例模式四（静态内部类单例）

这里有一种优雅的单例写法，采用内部类来创建实例，跟懒汉式一样，只有在调用的时候才进行创建实例public clas... 博文 来自： yxking1的专栏

23

单例模式静态内部类

前言： 仔细思考了一下这个模式，往里面深挖了一下，发现关于静态内部类未知的太多。尤其是java虚拟机classloa... 博文 来自： 好大的小飞虫的博客

275

内部类——实现单例

使用内部类方式实现单例，既可以做到延迟加载，有不必使用同步关键字，是一种比较完善的实现——Java程序性能... 博文 来自： 脚步不停，信念不...

1399

单例设计模式-静态内部类-基于类初始化的延迟加载解决方案及原理解析

刚刚线程1看不到线程0的重排序,我们创建一个类,这个方案是使用静态内部类来解决,一会我们也会分析一下原理,我们... 博文 来自： Leon_Jinhai_Sun...

245

单例静态内部类

/** * 描述: * 静态内部类 * * @author 小纸人 * @create 2019-03-10 22:46 */public class StaticInClassSingleton... 博文 来自： 赵智任的博客

111

spring怎么实现单例模式

在Spring中，bean可以被定义为两种模式：prototype（多例）和singleton（单例） singleton（单例）：只有一个... 博文 来自： 弹指天下

4059

Java单例模式私有静态内部类实现并测试

package org.vincent;import java.util.concurrent.Callable;import java.util.concurrent.ExecutionExcept... 博文 来自： Vincent

733

android之静态内部类单例模式

public class Singleton{ private Singleton(){} public static Singleton getInstance(){ return S... 博文 来自： teleger的私人博客

307

单例模式之内部静态类

利用内部静态类实现单例模式[java] view plain copypublic class Singleton { private Singleton(){ ... 博文 来自： 山鹰的专栏

578

单例模式原理

阅读数 611

单例设计模式：意图:保证一个类仅有一个实例，并提供一个访问它的全局访问点。使用性：当前只能有一个实例而且... 博文 来自： DAO1024

【单例深思】枚举实现单例原理

单例的枚举实现在《Effective Java》中有提到，因为其功能完整、使用简洁、无偿地提供了序列化机制、在面对复杂... 博文 来自： weixin_3

单例模式的基本原理以及需要注意的点

public class Singleton { private Singleton() {} //关键点0：构造函数是私有的 private ... 博文 来自： 一只正在

推荐 130 个令你眼前一亮的网站，总有一个用得着

总结了大学生活两年来，发现的 130 余个黑科技网站，总有一个会让你眼前一亮，赶紧收藏！... 博文 来自： 爪白白的

在中国程序员是青春饭吗？

今年，我也32了，为了不给大家误导，咨询了猎头、圈内好友，以及年过35岁的几位老程序员.....舍了老脸去揭人家... 博文 来自： 启舰

我在支付宝花了1分钟，查到了女朋友的开房记录！

在大数据时代下，不管你做什么都会留下蛛丝马迹，只要学会把各种软件运用到极致，捉奸简直轻而易举。今天就来... 博文 来自： zandaoguang的博...

程序员请照顾好自己，周末病魔差点一套带走我。

程序员在一个周末的时间，得了重病，差点当场去世，还好及时挽救回来了。... 博文 来自： 敖丙

python json java mysql pycharm android linux json格式



走着不语

TA的个人主页 >

原创7

粉丝16

获赞71

评论39

访问5万+

等级: 博客 已

周排名: 23万+

积分: 394

总排名: 17万+

关注

私信



可视化数据图

最新文章

清除AndroidStudio缓存的代理配置

AndroidStudio无法关联C++源码处理方式。

关于Android架构组件Room的简单使用

关于使用打包命令gradle、gradlew以及使用AndroidStudio右侧gradle菜单栏打包/install-run的区别

Android Studio使用protobuf协议开发

分类专栏



Android进阶之路

1篇



JAVA进阶

1篇



开发笔记

5篇

归档

2019年5月

1篇

2019年4月

1篇

2018年7月

1篇

2018年6月

1篇

2018年5月

2篇

2016年8月

1篇

展开

热门文章

深入理解单例模式：静态内部类单例原理

阅读数 38965

Android Studio使用protobuf协议开发

阅读数 5878

关于使用打包命令gradle、gradlew以及使用AndroidStudio右侧gradle菜单栏打

阅读数 3669

实现无限自动循环的ViewPager,广告轮询页面

阅读数 646

清除AndroidStudio缓存的代理配置

阅读数 644

最新评论

深入理解单例模式：静态内部类单例原理

u010465325：DCL失效问题失效是因为指令乱序，前文已经说了，然后volatile可以解决DCL ...

深入理解单例模式：静态内部类单例原理

weixin_43495019：[reply]weixin_43495019/[reply]JVM在对单例类加载的过程中执行了加载 ...

深入理解单例模式：静态内部类单例原理

kbocbre：[reply]weixin_43495019/[reply]我也是这个疑问，不管是用饿汉模式还是静态变量。 ...

深入理解单例模式：静态内部类单例原理

SUNBOYmxbsH：[reply]u013693703[/reply]volatile会在编译时加lock，禁止了指令重排序，i ...

深入理解单例模式：静态内部类单例原理

sinat_34654467：[reply]u013693703[/reply]你是对的，synchronized本身monitorEnter，n ...

1 gtx显卡排行

2 多线程

3 cpu至强天梯图

4 可视化分析平台

5 广州皮肤管理培

6 房屋买卖协议

7 CPU天梯榜

8 工作流程 英文

9 丁香茶的副作用

10 显卡天梯 2019

11 可视化数据展示

12 租赁女友

13 下巴痘痘为什么

14 cisco ios

15 mx4

16 服务器cpu天梯

17 sql有什么用

18 什么是泛型

19 高度近视怎么办


20 监理报名


21 面部皮肤管理加

22 主流的crm系统


23 学皮肤管理学费


24 用户画像是什么


 QQ客服


 客服论坛


工作时间 8:30-22:00

 kefu@csdn.net

 400-660-0108

 65

 38






举报

[关于我们](#)[招聘](#)[广告服务](#)[网站地图](#)


京ICP备19004658号 经营性网站备案信息


 公安备案号 11010502030143


©1999-2020 北京创新乐知网络技术有限
公司 网络110报警服务


北京互联网违法和不良信息举报中心


中国互联网举报中心 家长监护 版权申诉


65





38













举报