# 单例与序列化的那些事儿

2016/02/25 | 分类： 基础技术 | 0 条评论 | 标签： 单例, 序列化
分享到：
原文出处： Hollis

本文将通过实例+阅读Java源码的方式介绍序列化是如何破坏单例模式的，以及如何避免序列化对单例的破坏。

> 单例模式，是设计模式中最简单的一种。通过单例模式可以保证系统中一个类只有一个实例而且该实例易于外界访问，从而方便对实例个数的控制并节约系统资源。如果希望在系统中某个类的对象只能存在一个，单例模式是最好的解决方案。关于单例模式的使用方式，可以阅读单例模式的七种写法

但是，单例模式真的能够实现实例的唯一性吗？

答案是否定的，很多人都知道使用反射可以破坏单例模式，除了反射以外，使用序列化与反序列化也同样会破坏单例。

## 序列化对单例的破坏

首先来写一个单例的类：

code 1

```
1  package com.hollis;
2  import java.io.Serializable;
3  /**
4   * Created by hollis on 16/2/5.
5   * 使用双重校验锁方式实现单例
6   */
7  public class Singleton implements Serializable{
8      private volatile static Singleton singleton;
9      private Singleton (){}
10     public static Singleton getSingleton() {
```

```
        if (singleton == null) {
            synchronized (Singleton.class) {
                if (singleton == null) {
                    singleton = new Singleton();
                }
            }
        }
        return singleton;
    }
}
```

接下来是一个测试类：

code 2

```
package com.hollis;
import java.io.*;
/**
 * Created by hollis on 16/2/5.
 */
public class SerializableDemo1 {
    //为了便于理解，忽略关闭流操作及删除文件操作。真正编码时千万不要忘记
    //Exception直接抛出
    public static void main(String[] args) throws IOException, ClassNotFoundException {
        //Write Obj to file
        ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("tempFile"));
        oos.writeObject(Singleton.getSingleton());
        //Read Obj from file
        File file = new File("tempFile");
        ObjectInputStream ois =  new ObjectInputStream(new FileInputStream(file));
        Singleton newInstance = (Singleton) ois.readObject();
        //判断是否是同一个对象
        System.out.println(newInstance == Singleton.getSingleton());
    }
}
//false
```

输出结构为false，说明：

> 通过对Singleton的序列化与反序列化得到的对象是一个新的对象，这就破坏了Singleton的单例性。

这里，在介绍如何解决这个问题之前，我们先来深入分析一下，为什么会这样？在反序列化的过程中到底发生了什么。

# ObjectInputStream

对象的序列化过程通过ObjectOutputStream和ObjectInputputStream来实现的，那么带着刚刚的问题，分析一下ObjectInputputStream 的`readObject` 方法执行情况到底是怎样的。

为了节省篇幅，这里给出ObjectInputStream的`readObject`的调用栈：

```
readObject--->readObject0--->readOrdinaryObject--->checkResolve
```

这里看一下重点代码，`readOrdinaryObject`方法的代码片段：
code 3

```
private Object readOrdinaryObject(boolean unshared)
        throws IOException
    {
        //此处省略部分代码

        Object obj;
        try {
            obj = desc.isInstantiable() ? desc.newInstance() : null;
        } catch (Exception ex) {
            throw (IOException) new InvalidClassException(
```

```
                desc.forClass().getName(),
                "unable to create instance").initCause(ex);
        }

        //此处省略部分代码

        if (obj != null &&
            handles.lookupException(passHandle) == null &&
            desc.hasReadResolveMethod())
        {
            Object rep = desc.invokeReadResolve(obj);
            if (unshared && rep.getClass().isArray()) {
                rep = cloneArray(rep);
            }
            if (rep != obj) {
                handles.setObject(passHandle, obj = rep);
            }
        }

        return obj;
    }
```

code 3 中主要贴出两部分代码。先分析第一部分：

code 3.1

```
Object obj;
try {
    obj = desc.isInstantiable() ? desc.newInstance() : null;
} catch (Exception ex) {
    throw (IOException) new InvalidClassException(desc.forClass().getName(),"unable to create instance").initCause(ex);
}
```

这里创建的这个obj对象，就是本方法要返回的对象，也可以暂时理解为是ObjectInputStream的`readObject`返回的对象。

`isInstantiable`：如果一个serializable/externalizable的类可以在运行时被实例化，那么该方法就返回true。针对serializable和externalizable我会在其他文章中介绍。

`desc.newInstance`：该方法通过反射的方式调用无参构造方法新建一个对象。

所以。到目前为止，也就可以解释，为什么序列化可以破坏单例了？

答：序列化会通过反射调用无参数的构造方法创建一个新的对象。

那么，接下来我们再看刚开始留下的问题，如何防止序列化/反序列化破坏单例模式。

## 防止序列化破坏单例模式

先给出解决方案，然后再具体分析原理：

只要在Singleton类中定义`readResolve`就可以解决该问题：

code 4

```
package com.hollis;
import java.io.Serializable;
/**
 * Created by hollis on 16/2/5.
 * 使用双重校验锁方式实现单例
 */
public class Singleton implements Serializable{
    private volatile static Singleton singleton;
    private Singleton (){}
    public static Singleton getSingleton() {
```

```
        if (singleton == null) {
            synchronized (Singleton.class) {
                if (singleton == null) {
                    singleton = new Singleton();
                }
            }
        }
        return singleton;
    }

    private Object readResolve() {
        return singleton;
    }
}
```

具体原理，我们回过头继续分析code 3中的第二段代码:

code 3.2

```
if (obj != null &&
            handles.lookupException(passHandle) == null &&
            desc.hasReadResolveMethod())
        {
            Object rep = desc.invokeReadResolve(obj);
            if (unshared && rep.getClass().isArray()) {
                rep = cloneArray(rep);
            }
            if (rep != obj) {
                handles.setObject(passHandle, obj = rep);
            }
        }
```

`hasReadResolveMethod`:如果实现了serializable 或者 externalizable接口的类中包含`readResolve`则返回true

`invokeReadResolve`:通过反射的方式调用要被反序列化的类的readResolve方法。

所以，原理也就清楚了，主要在Singleton中定义readResolve方法，并在该方法中指定要返回的对象的生成策略，就可以方式单例被破坏。

## 总结

在涉及到序列化的场景时，要格外注意他对单例的破坏。

## 推荐阅读

深入分析Java的序列化与反序列化



**相关文章**

- 面试中单例模式有几种写法
- 关于 Java 中的 double check lock
- 深入学习 Java 序列化
- Hi，我们再来聊一聊Java的单例吧
- 如何防止单例模式被JAVA反射攻击
- 开发可正确序列化和反序列化的guava table组件
- 浅析若干Java序列化工具
- singleton模式四种线程安全的实现
- Java对象的序列化与反序列化
- 深入分析Java的序列化与反序列化

## 发表评论

Comment form

Name*

姓名

邮箱*

请填写邮箱

网站 (请以 http://开头)

请填写网站地址

评论内容*

请填写评论内容

(*) 表示必填项

提交评论

还没有评论。

« Maven和Gradle对比
深入分析Java的序列化与反序列化 »

Search for:

Search

Search



- 本周热门文章
- 本月热门
- 热门标签

0 用 Maven 实现一个 protobuf 的 ...

1 使用 watch 帮你重复执行命令

2 干货！一次 kafka 卡顿事故排查过...

3 直播一次问题排查过程


## 最新评论

- Re: Java代码优化
  建议针对接口编程，如：Map hm = new HashMap();而不是HashMap hm = ... 陈新宇
- Re: 高并发风控技术解密（上）
  来补充能量了！ www.wuliaokankan.cn
- Re: Java泛型详解
  深度好文~码了以后慢慢看~没有使用场景，一下子不能全部掌握 yountreeg
- Re: 你真的了解volatile关键字吗？
  是无效的啊，但是i++你已经执行完第一步和第二步了，第三步是不用重新读取i的，直接修改i的值，所以不... JustArgo

- Re: 单点登录原理与简单实现
  Hi，请到伯乐在线的小组发帖提问，支持微信登录。链接是： http://group.jobbole.... 唐尤华
- Re: 你真的了解volatile关键字吗？
  Hi，请到伯乐在线的小组发帖提问，支持微信登录。链接是： http://group.jobbole.... 唐尤华
- Re: 攻破JAVA NIO技术壁垒
  是的，已更新 唐尤华
- Re: Java泛型详解
  Hi，请到伯乐在线的小组发帖提问，支持微信登录。链接是： http://group.jobbole.... 唐尤华



## 关于ImportNew

ImportNew 专注于 Java 技术分享。于2012年11月11日 11:11正式上线。是的，这是一个很特别的时刻 :)

ImportNew 由两个 Java 关键字 import 和 new 组成，意指：Java 开发者学习新知识的网站。 import 可认为是学习和吸收， new 则可认为是新知识、新技术圈子和新朋友.....

## 联系我们

Email：ImportNew.com@gmail.com
新浪微博：@ImportNew
推荐微信号


反馈建议：ImportNew.com@gmail.com
广告与商务合作QQ：2302462408

## 推荐关注

小组 – 好的话题、有启发的回复、值得信赖的圈子
头条 – 写了文章？看干货？去头条！
相亲 – 为IT单身男女服务的征婚传播平台
资源 – 优秀的工具资源导航
翻译 – 活跃 & 专业的翻译小组
博客 – 国内外的精选博客文章
设计 – UI,网页，交互和用户体验
前端 – JavaScript, HTML5, CSS
安卓 – 专注Android技术分享
iOS – 专注iOS技术分享
Java – 专注Java技术分享
Python – 专注Python技术分享