

公告

昵称：along-JL
园龄：3年
粉丝：28
关注：6
[+加关注](#)

<	2018年5月							>
日	一	二	三	四	五	六		
29	30	1	2	3	4	5		
6	7	8	9	10	11	12		
13	14	15	16	17	18	19		
20	21	22	23	24	25	26		
27	28	29	30	31	1	2		
3	4	5	6	7	8	9		

搜索

找找看

谷歌搜索

常用链接

[我的随笔](#)
[我的评论](#)
[我的参与](#)
[最新评论](#)
[我的标签](#)

JAVA不可变类(immutable)机制与String的不可变性

一、不可变类简介

不可变类：所谓的不可变类是指这个类的实例一旦创建完成后，就不能改变其成员变量值。如JDK内部自带的很多不可变类：Interger、Long和String等。
可变类：相对于不可变类，可变类创建实例后可以改变其成员变量值，开发中创建的大部分类都属于可变类。

二、不可变类的优点

说完可变类和不可变类的区别，我们需要进一步了解为什么要有不可变类？这样的特性对JAVA来说带来怎样的好处？

1. 线程安全
- 不可变对象是线程安全的，在线程之间可以相互共享，不需要利用特殊机制来保证同步问题，因为对象的值无法改变。可以降低并发错误的可能性，因为不需要用一些锁机制等保证内存一致性问题也减少了同步开销。
2. 易于构造、使用和测试
3. ...

三、不可变类的设计方法

对于设计不可变类，个人总结出以下原则：

1. 类添加final修饰符，保证类不被继承。
如果类可以被继承会破坏类的不可变性机制，只要继承类覆盖父类的方法并且继承类可以改变成员变量值，那么一旦子类以父类的形式出现时，不能保证当前类是否可变。
2. 保证所有成员变量必须私有，并且加上final修饰
通过这种方式保证成员变量不可改变。但只做到这一步还不够，因为如果是对象成员变量有可能再外部改变其值。所以第4点弥补这个不足。
3. 不提供改变成员变量的方法，包括setter
避免通过其他接口改变成员变量的值，破坏不可变特性。

我的标签

Spring(4)
算法(2)
Java(2)
学习(1)

随笔分类

JDK源码解析
Spring学习笔记(3)
Spring源码分析
架构&设计(2)
算法练笔(1)
细品JAVA(2)
一叶一森林(1)
杂序(2)

随笔档案

2017年3月 (1)
2017年2月 (1)
2016年8月 (1)
2016年7月 (1)
2016年1月 (2)
2015年11月 (2)
2015年10月 (3)

文章分类

杂序

最新评论

1. Re:Spring Security 入门详解
That's good,thanks

--DwarfZebra
2. Re:Spring Security 入门详解
很好

--阿斯顿上
3. Re:Spring Security 入门详解
总结的很到位。

--qwfys200
4. Re:高性能页面加载技术--
BigPipe设计原理及Java简单实现
这个好用吗？

--战斗可乐

4.通过构造器初始化所有成员，进行深拷贝(deep copy)

如果构造器传入的对象直接赋值给成员变量，还是可以通过对传入对象的修改进而导致改变内部变量的值。例如：

```
public final class ImmutableDemo {  
    private final int[] myArray;  
    public ImmutableDemo(int[] array) {  
        this.myArray = array; // wrong  
    }  
}
```

这种方式不能保证不可变性，myArray和array指向同一块内存地址，用户可以在ImmutableDemo之外通过修改array对象的值来改变myArray内部的值。

为了保证内部的值不被修改，可以采用深度copy来创建一个新内存保存传入的值。正确做法：

```
public final class MyImmutableDemo {  
    private final int[] myArray;  
    public MyImmutableDemo(int[] array) {  
        this.myArray = array.clone();  
    }  
}
```

5. 在getter方法中，不要直接返回对象本身，而是克隆对象，并返回对象的拷贝

这种做法也是防止对象外泄，防止通过getter获得内部可变成员对象后对成员变量直接操作，导致成员变量发生改变。

四、String对象的不可变性

string对象在内存创建后就不可改变，不可变对象的创建一般满足以上5个原则，我们看看String代码是如何实现的。

```
public final class String  
    implements java.io.Serializable, Comparable<String>, CharSequence  
{  
  
    /** The value is used for character storage. */  
    private final char value[];  
    /** The offset is the first index of the storage that is used. */  
    private final int offset;  
    /** The count is the number of characters in the String. */  
    private final int count;  
    /** Cache the hash code for the string */  
    private int hash; // Default to 0  
    ....  
    public String(char value[]) {  
        this.value = Arrays.copyOf(value, value.length); // deep copy操作  
    }  
    ...  
    public char[] toCharArray() {  
        // Cannot use Arrays.copyOf because of class initialization order issues  
        char result[] = new char[value.length];  
        System.arraycopy(value, 0, result, 0, value.length);  
        return result;  
    }  
}
```

阅读排行榜

- 1. Spring Security 入门详解 (15326)
- 2. JAVA不可变类(immutable)机制与String的不可变性(8888)
- 3. 高性能页面加载技术--BigPipe设计原理及Java简单实现(2985)
- 4. Spring 入门知识点笔记整理 (1861)
- 5. 【面试笔记系列】排序算法汇总 (1516)

评论排行榜

- 1. Spring Security 入门详解(3)
- 2. Spring 入门知识点笔记整理(2)
- 3. 如何高效学习(1)
- 4. 高性能页面加载技术--BigPipe设计原理及Java简单实现(1)

推荐排行榜

- 1. 【面试笔记系列】排序算法汇总 (8)
- 2. Spring Security 入门详解(5)
- 3. JAVA不可变类(immutable)机制与String的不可变性(3)
- 4. 如何高效学习(3)
- 5. [细品java]ThreadLocal源码学习 (1)

```
...  
}
```

如上代码所示，可以观察到以下设计细节:

- 1. String类被final修饰，不可继承
- 2. string内部所有成员都设置为私有变量
- 3. 不存在value的setter
- 4. 并将value和offset设置为final。
- 5. 当传入可变数组value[]时，进行copy而不是直接将value[]复制给内部变量.
- 6. 获取value时不是直接返回对象引用，而是返回对象的copy.

这都符合上面总结的不变类型的特性，也保证了String类型是不可变的类。

五、String对象的不可变性的优缺点

从上一节分析，String数据不可变类，那设置这样的特性有什么好处呢？我总结为以下几点：

1.字符串常量池的需要.

字符串常量池可以将一些字符常量放在常量池中重复使用，避免每次都重新创建相同的对象、节省存储空间。但如果字符串是可变的，此时相同内容的String还指向常量池的同一个内存空间，当某个变量改变了该内存的值时，其他遍历的值也会发生改变。所以不符合常量池设计的初衷。

2. 线程安全考虑。

同一个字符串实例可以被多个线程共享。这样便不用因为线程安全问题而使用同步。字符串自己便是线程安全的。

3. 类加载器要用到字符串，不可变性提供了安全性，以便正确的类被加载。譬如你想加载java.sql.Connection类，而这个值被改成了myhacked.Connection，那么会对你的数据库造成不可知的破坏。

4. 支持hash映射和缓存。

因为字符串是不可变的，所以在它创建的时候hashcode就被缓存了，不需要重新计算。这就使得字符串很适合作为Map中的键，字符串的处理速度要快过其它的键对象。这就是HashMap中的键往往都使用字符串。

缺点：

- 1. 如果有对String对象值改变的需求，那么会创建大量的String对象。
- 2.

六、String对象的是否真的不可变

虽然String对象将value设置为final,并且还通过各种机制保证其成员变量不可改变。但是还是可以通过反射机制的手段改变其值。例如：

```
//创建字符串"Hello World", 并赋给引用s  
String s = "Hello World";  
System.out.println("s = " + s); //Hello World  
  
//获取String类中的value字段  
Field valueFieldOfString = String.class.getDeclaredField("value");  
//改变value属性的访问权限  
valueFieldOfString.setAccessible(true);  
  
//获取s对象上的value属性的值
```

```
char[] value = (char[]) valueFieldOfString.get(s);
//改变value所引用的数组中的第5个字符
value[5] = '_';
System.out.println("s = " + s); //Hello_World
```

打印结果为：

```
s = Hello World
s = Hello_World
```

发现String的值已经发生了改变。也就是说，通过反射是可以修改所谓的“不可变”对象的

总结

不可变类是实例创建后就不可以改变成员遍历的值。这种特性使得不可变类提供了线程安全的特性但同时也带来了对象创建的开销，每更改一个属性都是重新创建一个新的对象。JDK内部也提供了很多不可变类如Integer、Double、String等。String的不可变特性主要为了满足常量池、线程安全、类加载的需求。合理使用不可变类可以带来极大的好处。

参考资料

- [1] <http://my.oschina.net/zzw922cn/blog/487610>
- [2] java的String 为什么是不可变的:<http://www.codeceo.com/article/why-java-string-immutable.html>
- [3] <http://www.importnew.com/7535.html>

分类: 细品JAVA

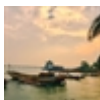
好文要顶

关注我

收藏该文







along-JL

关注 - 6

粉丝 - 28

+加关注

« 上一篇：如何高效学习
» 下一篇：【面试笔记系列】排序算法汇总

posted @ 2016-07-30 19:30 along-JL 阅读(8889) 评论(0) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。



最新IT新闻:

- 母亲节“最特别的礼物”：梁建章号召女随母姓
 - Android P手势操作首秀获赞：谷歌又进行了一番新优化
 - 周鸿祎晒汶川地震救灾照片：曾个人捐款100万
 - 《我的世界》Switch版将免费升级基岩版：新增游戏模式
 - 苏宁易购半价推员工持股计划 1600名员工参与
- » 更多新闻...



最新知识库文章:

- 如何高效学习
 - 如何成为优秀的程序员？
 - 菜鸟工程师的超神之路 -- 从校园到职场
 - 如何识别别人的技术能力和水平？
 - 写给自学者的入门指南
- » 更多知识库文章...