

浅谈Java浅层复制和深层复制

Java对象的深层复制是指Java对象A本身被clone成新对象B，同时A的属性也是被clone成新对象，赋值到A的各个属性上去，A与B的相同属性都引用到不同的对象；
Java对象的浅层复制是指Java对象A本身被clone成新对象B,但A的属性没有被clone处理，只是把A的各个属性所指的对象赋值到B对应的属性上，A与B的相同属性都引用到同一个对象。

在java中，默认是浅层复制的，如果要复制的对象中只含有基本数据类型和String类型，

那么浅层复制和浅层复制是没有区别的，所以你可以放心的使用默认的浅层复制，

如果属性有Date或其他自定的数据类，则一定的小心了，因为这时浅层复制后对象B的属性birthday与原始对象A的对应属性birthday，都是引用到同一个对象TestVo，

如果通过B.birthday的方法改了TestVo的值，则修改会影响到A.birthday，这时也就会发生互串的情况

以下三种方法可以实现浅层复制：

(1) 通过调用对象set方法来实现，属性个数比较少时适用



```
public class TestVo implements Cloneable{

    private String name;
    private int age;
    private Date birthday;

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public Date getBirthday() {
        return birthday;
    }
    public void setBirthday(Date birthday) {
        this.birthday = birthday;
    }

    protected TestVo clone() {
        TestVo testVo = null;
        try {
            testVo = (TestVo) super.clone();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return testVo;
    }
}
```



```
public static void test1(){
    TestVo t1 = new TestVo();
    t1.setAge(10);
    t1.setName("刘备");
    t1.setBirthday(DateUtil.strToDate("2016-10-10 12:12:12"));

    TestVo t2 = new TestVo();
    t2.setAge(t1.getAge());
    t2.setName(t1.getName());
    t2.setBirthday(t1.getBirthday());

    System.out.println("t1==" + t1.getAge() + ", " + t1.getName() + ", " + t1.getBirthday());
    System.out.println("t2==" + t2.getAge() + ", " + t2.getName() + ", " + t2.getBirthday());
}
```

公告

昵称：徜徉林
园龄：2年2个月
粉丝：1
关注：6
+加关注

< 2018年4月 >

日	一	二	三	四	五	六
25	26	27	28	29	30	31
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	1	2	3	4	5

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

我的标签

Java(1)
浅层复制(1)
深层复制(1)

随笔分类

hadoop

随笔档案

2017年5月 (2)
2016年10月 (1)

最新评论

1. Re:谈谈我的session跨域处理方法
@笨笨、你好噯，父子域名之间是可以共享session的，如果是父子域名，就不需要这么麻烦...

--徜徉林

```
        t2.setAge(20);
        t2.setName("张飞");
        t2.setBirthday(DateUtil.strToDate("2016-11-11 13:13:13"));
        System.out.println("t3==" + t1.getAge() + ", " + t1.getName() + ", " + t1.getBirthday());
        System.out.println("t4==" + t2.getAge() + ", " + t2.getName() + ", " + t2.getBirthday());
    }
}
```



打印结果如下：

t1==10,刘备,Mon Oct 10 12:12:12 CST 2016
t2==10,刘备,Mon Oct 10 12:12:12 CST 2016
t3==10,刘备,Mon Oct 10 12:12:12 CST 2016
t4==20,张飞,Fri Nov 11 13:13:13 CST 2016

这时候浅层复制和深层复制打到的效果是一样的，所以对t2的值修改，不会影响t1对象的值，
但是如果date类型的属性值，按照以下方法设置值，则会影响到t1中的值

```
public static void test1(){
    TestVo t1 = new TestVo();
    t1.setAge(10);
    t1.setName("刘备");
    t1.setBirthday(DateUtil.strToDate("2016-10-10 12:12:12"));

    TestVo t2 = new TestVo();
    t2.setAge(t1.getAge());
    t2.setName(t1.getName());
    t2.setBirthday(t1.getBirthday());

    System.out.println("t1==" + t1.getAge() + ", " + t1.getName() + ", " + t1.getBirthday());
    System.out.println("t2==" + t2.getAge() + ", " + t2.getName() + ", " + t2.getBirthday());

    t2.setAge(20);
    t2.setName("张飞");
    //t2.setBirthday(DateUtil.strToDate("2016-11-11 13:13:13"));
    t2.getBirthday().setTime(1000);
    System.out.println("t3==" + t1.getAge() + ", " + t1.getName() + ", " + t1.getBirthday());
    System.out.println("t4==" + t2.getAge() + ", " + t2.getName() + ", " + t2.getBirthday());
}
}
```



打印结果如下：

t1==10,刘备,Mon Oct 10 12:12:12 CST 2016
t2==10,刘备,Mon Oct 10 12:12:12 CST 2016
t3==10,刘备,Thu Jan 01 08:00:01 CST 1970
t4==20,张飞,Thu Jan 01 08:00:01 CST 1970

这时候t2值修改后，t1也跟着修改了，Date是一个可变的类，这样浅层复制就出现属性值互串的情况，
通过监听引用地址t1.getBirthday() == t2.getBirthday()，发现：

t2.setBirthday(DateUtil.strToDate("2016-11-11 13:13:13"));这样设置值，t1和t2的引用地址不一样，所有t2修改不会对t1产生影响
t2.getBirthday().setTime(1000);这样设置值，t1.getBirthday() == t2.getBirthday()控制台返回true,说明t1和t2的引用同一个地址，所有会相互影响，
如有不对之处，请不吝赐教，欢迎指正

(2) 通过复写object.clone来实现

```
TestVo t1 = new TestVo();
t1.setAge(10);
t1.setName("刘备");
t1.setBirthday(DateUtil.strToDate("2016-10-10 12:12:12"));

TestVo t2 = t1.clone();
```

(3)通过工具类，BeanUtils, 属性个数很多时候适用

```
TestVo t1 = new TestVo();
t1.setAge(10);
t1.setName("刘备");
t1.setBirthday(DateUtil.strToDate("2016-10-10 12:12:12"));

TestVo t2 = new TestVo();
BeanUtils.copyProperties(t2, t1);
```



2. Re:谈谈我的session跨域处理方法
主域名和子域名之间是可以实现session共享的，不用一参数的形式来实现

--笨笨、你好

3. Re:谈谈我的session跨域处理方法
可以使用无状态token验证替代session

--我不是浩天天

4. Re:谈谈我的session跨域处理方法
学习了。

--坤坤

5. Re:浅谈Java浅层复制和深层复制
@2188谢谢，学习了...

--徜徉林

阅读排行榜

1. 谈谈我的session跨域处理方法(9517)
2. 浅谈Java浅层复制和深层复制(1391)
3. Eclipse中启动tomcat: java.lang.OutOfMemoryError: PermGen space的解决方法(404)

评论排行榜

1. 浅谈Java浅层复制和深层复制(7)
2. 谈谈我的session跨域处理方法(4)

推荐排行榜

1. 谈谈我的session跨域处理方法(1)

(2)(3)和(1)的打印结果是一样的，同样，如果(2)(3)的date类型也按照如下修改值，也会影响t1的值，和(1)的一样

t2.getBirthday().setTime(1000);

再看下面的例子：



```
public class TestVoB {

    private int sex;
    public TestVoB(int sex){
        this.sex = sex;
    }

    public void sumValue(){
        this.sex += 10;
    }

    public String toString(){
        return Integer.toString(sex);
    }

    public int getSex() {
        return sex;
    }

    public void setSex(int sex) {
        this.sex = sex;
    }

}
```



```
public class TestVo implements Cloneable{

    private String name;
    private int age;
    private Date birthday;
    TestVoB testVoB = new TestVoB(222);

    public TestVoB getTestVoB() {
        return testVoB;
    }
    public void setTestVoB(TestVoB testVoB) {
        this.testVoB = testVoB;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public Date getBirthday() {
        return birthday;
    }
    public void setBirthday(Date birthday) {
        this.birthday = birthday;
    }

    protected TestVo clone() {
        TestVo testVo = null;
        try {
            testVo = (TestVo) super.clone();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return testVo;
    }

}
```



```
public static void test2() throws CloneNotSupportedException{
    TestVo t1 = new TestVo();
}
```

```
t1.setAge(10);
t1.setName("刘备");
t1.setBirthday(DateUtil.strToDate("2016-10-10 12:12:12"));

TestVo t2 = t1.clone();
System.out.println("t1=="+t1.getAge()+" "+t1.getName()+" "+t1.getBirthday());
System.out.println("t2=="+t2.getAge()+" "+t2.getName()+" "+t2.getBirthday());

t2.setAge(20);
t2.setName("张飞");
t2.setBirthday(DateUtil.strToDate("2016-11-11 13:13:13"));
t2.testVoB.sumValue();

System.out.println("t3=="+t1.getAge()+" "+t1.getName()+" "+t1.getBirthday()+" "+t1.testVoB);

System.out.println("t4=="+t2.getAge()+" "+t2.getName()+" "+t2.getBirthday()+" "+t2.testVoB);
}
```

打印结果如下：

t1==10,刘备,Mon Oct 10 12:12:12 CST 2016
t2==10,刘备,Mon Oct 10 12:12:12 CST 2016
t3==10,刘备,Mon Oct 10 12:12:12 CST 2016,232
t4==20,张飞,Fri Nov 11 13:13:13 CST 2016,232

可以看出t2中的testVoB值的修改影响了t1中的testVoB，说明两个引用指向同一个对象。

一般情况下，用浅层复制就够了，但是在特殊情况下，浅层复制不能满足我们的业务需求，这时候就需要深层复制，实现深层复制只需要在上面的例子中修改一下就可以：

- 1.让TestVoB 类也实现和TestVo类一样的clone功能（实现Cloneable接口，重载clone()方法），
- 2.在TestVo的clone()方法中加入一句testVo.testVoB = testVoB.clone();

```
public class TestVoB implements Cloneable{

    private int sex;

    public TestVoB clone() {
        TestVoB testVoB = null;
        try {
            testVoB = (TestVoB) super.clone();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return testVoB;
    }

    public TestVoB(int sex){
        this.sex = sex;
    }

    public void sumValue(){
        this.sex += 10;
    }

    public String toString(){
        return Integer.toString(sex);
    }

    public int getSex() {
        return sex;
    }

    public void setSex(int sex) {
        this.sex = sex;
    }

}

public class TestVo implements Cloneable{

    private String name;
    private int age;
    private Date birthday;
    TestVoB testVoB = new TestVoB(222);

    public TestVoB getTestVoB() {
        return testVoB;
    }

    public void setTestVoB(TestVoB testVoB) {
```

```
        this.testVoB = testVoB;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public Date getBirthday() {
        return birthday;
    }
    public void setBirthday(Date birthday) {
        this.birthday = birthday;
    }

    protected TestVo clone() {
        TestVo testVo = null;
        try {
            testVo = (TestVo) super.clone();
        } catch (Exception e) {
            e.printStackTrace();
        }
        testVo.testVoB = testVoB.clone();
        return testVo;
    }
}
```



这时候再去执行，打印结果如下：

t1==10,刘备,Mon Oct 10 12:12:12 CST 2016
t2==10,刘备,Mon Oct 10 12:12:12 CST 2016
t3==10,刘备,Mon Oct 10 12:12:12 CST 2016,222
t4==20,张飞,Fri Nov 11 13:13:13 CST 2016,232

可以看到t2中的testVoB值的修改，t1中的testVoB没有变化，说明两个对象引用指向了不同的对象，实现了深层复制

以上只是本人自己的一些理解，如有不对的地方，请不吝赐教，共同学习

标签： 深层复制 ， 浅层复制 ， Java

好文要顶

关注我

收藏该文



徜徉林
关注 - 6
粉丝 - 1

+加关注

0

0

» 下一篇：[Eclipse中启动tomcat: java.lang.OutOfMemoryError: PermGen space的解决方法](#)

posted @ 2016-10-14 11:06 徜徉林 阅读(1391) 评论(7) 编辑 收藏

评论列表

#1楼 2016-10-14 11:15 旷视科技/face++

受教了谢谢

支持(0) 反对(0)

#2楼 2016-10-14 14:11 597967460

至于为什么两种写法导致的结果不一样	
string虽然是引用对象，但是是不可变对象。 修改克隆后对象的string属性只修改了引用，未修改值，因此原对象不影响。	
支持(0) 反对(0)	
# 3楼[楼主] 2016-10-14 14:17 徜徉林	
@ 597967460 谢谢，那Date类型的属性， t2.setBirthday(DateUtil.strToDate("2016-11-11 13:13:13")); t2.getBirthday().setTime(1000); 这两种写法导致的结果不一样，是什么原因呢？	
支持(0) 反对(0)	
# 4楼 2016-10-14 14:20 597967460	
@ 徜徉林	
t2.setBirthday(DateUtil.strToDate("2016-11-11 13:13:13"));///修改了对象的引用，t1和t2在这句之后指向了两个不同的对象	
支持(0) 反对(0)	
# 5楼[楼主] 2016-10-14 14:29 徜徉林	
@ 597967460 谢谢，明白了，Date是可变类型， t2.getBirthday().setTime(1000) 这样设置值，等于在原引用上直接修改值，两个对象中的Date属性引用还是指向同一个对象	
支持(0) 反对(0)	
# 6楼 2016-10-14 16:29 2188	
http://blog.csdn.net/a220315410/article/details/27743607	
String是引用类型，但某些情况会有值特性。上方地址有详解	
支持(0) 反对(0)	
# 7楼[楼主] 2016-10-14 16:44 徜徉林	
@ 2188 谢谢，学习了	
支持(0) 反对(0)	
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](#)。

 腾讯云

助力开发者快速搭建小程序

一站式配置主机和域名
套餐11元/月起

立即抢购



最新IT新闻：

- 马化腾：腾讯不要进入各行各业取而代之 要成为数字化转型的连接器
- 沉痛悼念 “DPDK之父” Venky Venkatesan
- SpaceX总裁自曝公司成功秘诀：在“空白纸”上自由设计火箭
- 全民公敌携程
- 纽约时报评扎克伯格听证会第二日：众议院的问题更刁钻

» [更多新闻...](#)

 阿里云

新购满返 ¥6000 封顶

广告



最新知识库文章：

- [写给自学者入门指南](#)
- [和程序员谈恋爱](#)
- [学会学习](#)
- [优秀技术人的管理陷阱](#)
- [作为一个程序员，数学对你到底有多重要](#)
- » [更多知识库文章...](#)