为什么阿里巴巴禁止使用Apache Beanutils进行属性的copy?

原创 Hollis Hollis 2020-07-29 09:30

收录于合集

#解读阿里开发手册

15个

△Hollis, 一个对Coding有着独特追求的人△

这是Hollis的第 **297** 篇原创分享

作者 | Hollis

来源 | Hollis（ID：hollischuang）

在日常开发中，我们经常需要给对象进行赋值，通常会调用其set/get方法，有些时候，如果我们要转换的两个对象之间属性大致相同，会考虑使用属性拷贝工具进行。

如我们经常在代码中会对一个数据结构封装成DO、SDO、DTO、VO等，而这些Bean中的大部分属性都是一样的，所以使用属性拷贝类工具可以帮助我们节省大量的set和get操作。

市面上有很多类似的工具类，比较常用的有

1、Spring BeanUtils

2、Cglib BeanCopier

3、Apache BeanUtils

4、Apache PropertyUtils

5、Dozer

那么，我们到底应该选择哪种工具类更加合适呢？为什么阿里巴巴Java开发手册中提到禁止使用Apache BeanUtils呢？

2. 【强制】避免用 Apache Beanutils 进行属性的 copy。
   说明：Apache BeanUtils 性能较差，可以使用其他方案比如 Spring BeanUtils, Cglib BeanCopier，注意均是浅拷贝。

由于篇幅优先，关于这几种工具类的用法及区别，还有到底是什么是浅拷贝和深拷贝不在本文的讨论范围内。

本文主要聚焦于对比这几个类库的性能问题。

性能对比

No Data No BB，我们就来写代码来对比下这几种框架的性能情况。

代码示例如下：

首先定义一个PersonDO类：

```
public class PersonDO {
    private Integer id;
    private String name;
    private Integer age;
    private Date birthday;
    //省略setter/getter
}
```

再定义一个PersonDTO类：

```
public class PersonDTO {
    private String name;
    private Integer age;
    private Date birthday;
}
```

然后进行测试类的编写：

使用Spring BeanUtils进行属性拷贝：

```
private void mappingBySpringBeanUtils(PersonDO personDO, int times) {
    StopWatch stopwatch = new StopWatch();
    stopwatch.start();

    for (int i = 0; i < times; i++) {
        PersonDTO personDTO = new PersonDTO();
        org.springframework.beans.BeanUtils.copyProperties(personDO, personDTO);
    }
    stopwatch.stop();
    System.out.println("mappingBySpringBeanUtils cost :" + stopwatch.getTotalTimeMillis());
}
```

其中的StopWatch用于记录代码执行时间，方便进行对比。

使用Cglib BeanCopier进行属性拷贝：

```
private void mappingByCglibBeanCopier(PersonDO personDO, int times) {
    StopWatch stopwatch = new StopWatch();
    stopwatch.start();
    for (int i = 0; i < times; i++) {
        PersonDTO personDTO = new PersonDTO();
        BeanCopier copier = BeanCopier.create(PersonDO.class, PersonDTO.class, false);
```

```
            copier.copy(personDO, personDTO, null);
        }
        stopwatch.stop();
        System.out.println("mappingByCglibBeanCopier cost :" + stopwatch.getTotalTimeMillis());
    }
```

使用Apache BeanUtils进行属性拷贝：

```
    private void mappingByApacheBeanUtils(PersonDO personDO, int times)
        throws InvocationTargetException, IllegalAccessException {
        StopWatch stopwatch = new StopWatch();
        stopwatch.start();
        for (int i = 0; i < times; i++) {
            PersonDTO personDTO = new PersonDTO();
            BeanUtils.copyProperties(personDTO, personDO);
        }
        stopwatch.stop();
        System.out.println("mappingByApacheBeanUtils cost :" + stopwatch.getTotalTimeMillis());
    }
```

使用Apache PropertyUtils进行属性拷贝：

```
    private void mappingByApachePropertyUtils(PersonDO personDO, int times)
        throws InvocationTargetException, IllegalAccessException, NoSuchMethodException {
        StopWatch stopwatch = new StopWatch();
```

```
        stopwatch.start();
        for (int i = 0; i < times; i++) {
            PersonDTO personDTO = new PersonDTO();
            PropertyUtils.copyProperties(personDTO, personDO);
        }
        stopwatch.stop();
        System.out.println("mappingByApachePropertyUtils cost :" + stopwatch.getTotalTimeMillis());
    }
```

然后执行以下代码：

```
    public static void main(String[] args)
        throws InvocationTargetException, IllegalAccessException, NoSuchMethodException {
        PersonDO personDO = new PersonDO();
        personDO.setName("Hollis");
        personDO.setAge(26);
        personDO.setBirthday(new Date());
        personDO.setId(1);

        MapperTest mapperTest = new MapperTest();

        mapperTest.mappingBySpringBeanUtils(personDO, 100);
        mapperTest.mappingBySpringBeanUtils(personDO, 1000);
        mapperTest.mappingBySpringBeanUtils(personDO, 10000);
        mapperTest.mappingBySpringBeanUtils(personDO, 100000);
```

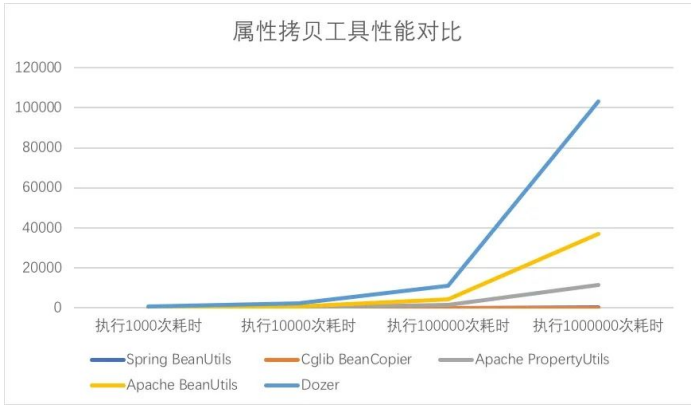
```
        mapperTest.mappingBySpringBeanUtils(personDO, 1000000);
        mapperTest.mappingByCglibBeanCopier(personDO, 100);
        mapperTest.mappingByCglibBeanCopier(personDO, 1000);
        mapperTest.mappingByCglibBeanCopier(personDO, 10000);
        mapperTest.mappingByCglibBeanCopier(personDO, 100000);
        mapperTest.mappingByCglibBeanCopier(personDO, 1000000);
        mapperTest.mappingByApachePropertyUtils(personDO, 100);
        mapperTest.mappingByApachePropertyUtils(personDO, 1000);
        mapperTest.mappingByApachePropertyUtils(personDO, 10000);
        mapperTest.mappingByApachePropertyUtils(personDO, 100000);
        mapperTest.mappingByApachePropertyUtils(personDO, 1000000);
        mapperTest.mappingByApacheBeanUtils(personDO, 100);
        mapperTest.mappingByApacheBeanUtils(personDO, 1000);
        mapperTest.mappingByApacheBeanUtils(personDO, 10000);
        mapperTest.mappingByApacheBeanUtils(personDO, 100000);
        mapperTest.mappingByApacheBeanUtils(personDO, 1000000);
    }
```

得到结果如下：

工具类	执行1000次耗时	执行10000次耗时	执行100000次耗时	执行1000000次耗时

工具类	执行1000次耗时	执行10000次耗时	执行100000次耗时	执行1000000次耗时
Spring BeanUtils	5ms	10ms	45ms	169ms
Cglib BeanCopier	4ms	18ms	45ms	91ms
Apache PropertyUtils	60ms	265ms	1444ms	11492ms
Apache BeanUtils	138ms	816ms	4154ms	36938ms
Dozer	566ms	2254ms	11136ms	102965ms

画了一张折线图更方便大家进行对比



综上，我们基本可以得出结论，**在性能方面，Spring BeanUtils和Cglib BeanCopier表现比较不错，而Apache PropertyUtils、Apache BeanUtils以及Dozer则表现的很不好。**

所以，如果考虑性能情况的话，建议大家不要选择Apache PropertyUtils、Apache BeanUtils以及Dozer等工具类。

很多人会不理解，为什么大名鼎鼎的Apache开源出来的的类库性能确不高呢？这不像是Apache的风格呀，这背后导致性能低下的原因又是什么呢？

其实，是因为Apache BeanUtils力求做得完美，在代码中增加了非常多的校验、兼容、日志打印等代码，过度的包装导致性能下降严重。

总结

本文通过对比几种常见的属性拷贝的类库，分析得出了这些工具类的性能情况，最终也验证了《阿里巴巴Java开发手册》中提到的"Apache BeanUtils 效率低"的事实。

但是本文只是站在性能这一单一角度进行了对比，我们在选择一个工具类的时候还会有其他方面的考虑，比如使用成本、理解难度、兼容性、可扩展性等，对于这种拷贝类工具类，我们还会考虑其功能是否完善等。

就像虽然Dozer性能比较差，但是他可以很好的和Spring结合，可以通过配置文件等进行属性之间的映射等，也受到了很多开发者的喜爱。

本文用到的第三方类库的maven依赖如下：

```
<!--Apache PropertyUtils、Apache BeanUtils-->
<dependency>
    <groupId>commons-beanutils</groupId>
    <artifactId>commons-beanutils</artifactId>
    <version>1.9.4</version>
</dependency>

<dependency>
    <groupId>commons-logging</groupId>
    <artifactId>commons-logging</artifactId>
    <version>1.1.2</version>
</dependency>

<!--Spring PropertyUtils-->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>org.springframework.beans</artifactId>
    <version>3.1.1.RELEASE</version>
</dependency>

<!--cglib-->
```

```
<dependency>
    <groupId>cglib</groupId>
    <artifactId>cglib-nodep</artifactId>
    <version>2.2.2</version>
</dependency>

<!--dozer-->
<dependency>
    <groupId>net.sf.dozer</groupId>
    <artifactId>dozer</artifactId>
    <version>5.5.1</version>
</dependency>

<!--日志相关-->
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>1.7.7</version>
</dependency>

<dependency>
    <groupId>org.slf4j</groupId>
```

```
    <artifactId>jul-to-slf4j</artifactId>
    <version>1.7.7</version>
</dependency>

<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>jcl-over-slf4j</artifactId>
    <version>1.7.7</version>
</dependency>

<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>log4j-over-slf4j</artifactId>
    <version>1.7.7</version>
</dependency>

<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-jdk14</artifactId>
    <version>1.7.7</version>
</dependency>
```