Java bean 和 Map<String,Object> 互转工具类

作者：iMLe0n
原文链接：https://ld246.com/article/1544771470952
来源网站：链滴

好多都不符合业务需求自己手写了个

```
package com.xxx.xxx.commons.util;

import java.beans.BeanInfo;
import java.beans.IntrospectionException;
import java.beans.Introspector;
import java.beans.PropertyDescriptor;
import java.lang.reflect.Field;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.math.BigDecimal;
import java.util.HashMap;
import java.util.Map;

/**
 * @Author: imle0n
 * @Date: 18-12-14 11:27
 * @Description: javaBean 和 Map 的互转
  */
public class BeanMapTransUtil {

  public static Map<String, Object> tranferBean2Map(Object obj) throws Exception {
  //obj为空，结束方法
  if (obj == null) {
  return null;
  }

  Map<String, Object> map = new HashMap<String, Object>();
  /* Introspector 类为通过工具学习有关受目标 Java Bean 支持的属性、事件和方法的知识提供了一
标准方法。
```

原文链接：Java bean 和 Map<String,Object> 互转工具类

```
* java的自省机制
* */  BeanInfo beanInfo = Introspector.getBeanInfo(obj.getClass());
 PropertyDescriptor[] ps = beanInfo.getPropertyDescriptors();
 for (PropertyDescriptor propertyDescriptor : ps) {
 String key = propertyDescriptor.getName();

 if (!"class".equals(key)) {
 Method getter = propertyDescriptor.getReadMethod();
 Object value = getter.invoke(obj);
 map.put(key, value);
 }
}  return map;
 }

 public static  T transferMap2Bean(Class clazz, Map map) throws IntrospectionException, Illeg
lAccessException, InstantiationException, InvocationTargetException {
 //获取类属性
 BeanInfo beanInfo = Introspector.getBeanInfo(clazz);
 T obj = clazz.newInstance();
 // 给 JavaBean 对象的属性赋值
 PropertyDescriptor[] propertyDescriptors = beanInfo.getPropertyDescriptors();
 for (int i = 0; i < propertyDescriptors.length; i++) {
 PropertyDescriptor descriptor = propertyDescriptors[i];
 String propertyName = descriptor.getName();

 if (map.containsKey(propertyName)) {
 // 下面一句可以 try 起来，这样当一个属性赋值失败的时候就不会影响其他属性赋值。
 Object value = map.get(propertyName);

 Object[] args = new Object[1];
 args[0] = value;
 Field privateField = getPrivateField(propertyName, clazz);
 if (privateField == null) {
} privateField.setAccessible(true);
 String type = privateField.getGenericType().toString();
 if ("class java.lang.String".equals(type)) {
 if (value == null) {
privateField.set(obj, null);
 } else {
privateField.set(obj, value);
 }

} else if ("class java.lang.Boolean".equals(type)) {
 if (value == null) {
privateField.set(obj, null);
 } else {
privateField.set(obj, Boolean.parseBoolean(String.valueOf(value)));
 }
} else if ("class java.lang.Long".equals(type)) {
 if (value == null) {
privateField.set(obj, null);
 } else {
privateField.set(obj, Long.parseLong(String.valueOf(value)));
 }
```

```
} else if ("class java.lang.Integer".equals(type)) {
 if (value == null) {
privateField.set(obj, null);
 } else {
privateField.set(obj, Integer.parseInt(String.valueOf(value)));
 }

} else if ("class java.lang.Double".equals(type)) {
 if (value == null) {
privateField.set(obj, null);
 } else {
privateField.set(obj, Double.parseDouble(String.valueOf(value)));
 }

} else if ("class java.lang.Float".equals(type)) {
 if (value == null) {
privateField.set(obj, null);
 } else {
privateField.set(obj, Float.parseFloat(String.valueOf(value)));
 }

} else if ("class java.math.BigDecimal".equals(type)) {
 if (value == null) {
privateField.set(obj, null);
 } else {
privateField.set(obj, new BigDecimal(String.valueOf(value)));
 }

}//可继续追加类型
 }
}  return obj;
 }

 /*拿到反射父类私有属性*/
 private static Field getPrivateField(String name, Class cls) {
 Field declaredField = null;
 try {
declaredField = cls.getDeclaredField(name);
 } catch (NoSuchFieldException ex) {

 if (cls.getSuperclass() == null) {
 return declaredField;
 } else {
declaredField = getPrivateField(name, cls.getSuperclass());
 }
}  return declaredField;
 }

}
```