001/\*  
002 \* Licensed to the Apache Software Foundation (ASF) under one or more  
003 \* contributor license agreements. See the NOTICE file distributed with  
004 \* this work for additional information regarding copyright ownership.  
005 \* The ASF licenses this file to You under the Apache License, Version 2.0  
006 \* (the "License"); you may not use this file except in compliance with  
007 \* the License. You may obtain a copy of the License at  
008 \*  
009 \* http://www.apache.org/licenses/LICENSE-2.0  
010 \*  
011 \* Unless required by applicable law or agreed to in writing, software  
012 \* distributed under the License is distributed on an "AS IS" BASIS,  
013 \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
014 \* See the License for the specific language governing permissions and  
015 \* limitations under the License.  
016 \*/  
017  
018package org.apache.commons.beanutils;  
019  
020import java.lang.reflect.InvocationTargetException;  
021  
022/\*\*  
023 \* <p>Implementation of <code>DynaBean</code> that wraps a standard JavaBean  
024 \* instance, so that DynaBean APIs can be used to access its properties,  
025 \* though this implementation allows type conversion to occur when properties are set.  
026 \* This means that (say) Strings can be passed in as values in setter methods and  
027 \* this DynaBean will convert them to the correct primitive data types.</p>  
028 \*  
029 \* <p><strong>IMPLEMENTATION NOTE</strong> - This implementation does not  
030 \* support the <code>contains()</code> and <code>remove()</code> methods.</p>  
031 \*  
032 \* @version $Id$  
033 \*/  
034  
035public class ConvertingWrapDynaBean extends WrapDynaBean {  
036  
037  
038  
039 /\*\*  
040 \* Construct a new <code>DynaBean</code> associated with the specified  
041 \* JavaBean instance.  
042 \*  
043 \* @param instance JavaBean instance to be wrapped  
044 \*/  
045 public ConvertingWrapDynaBean(final Object instance) {  
046  
047 super(instance);  
048  
049 }  
050  
051  
052 /\*\*  
053 \* Set the value of the property with the specified name  
054 \* performing any type conversions if necessary. So this method  
055 \* can accept String values for primitive numeric data types for example.  
056 \*  
057 \* @param name Name of the property whose value is to be set  
058 \* @param value Value to which this property is to be set  
059 \*  
060 \* @throws IllegalArgumentException if there are any problems  
061 \* copying the property.  
062 \*/  
063 @Override  
064 public void set(final String name, final Object value) {  
065  
066 try {  
067 BeanUtils.copyProperty(instance, name, value);  
068 } catch (final InvocationTargetException ite) {  
069 final Throwable cause = ite.getTargetException();  
070 throw new IllegalArgumentException  
071 ("Error setting property '" + name +  
072 "' nested exception - " + cause);  
073 } catch (final Throwable t) {  
074 final IllegalArgumentException iae = new IllegalArgumentException  
075 ("Error setting property '" + name +  
076 "', exception - " + t);  
077 BeanUtils.initCause(iae, t);  
078 throw iae;  
079 }  
080  
081 }  
082}