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 \*/  
017package org.apache.commons.beanutils;  
018  
019  
020/\*\*  
021 \* <p>Decorates a {@link DynaBean} to provide <code>Map</code> behaviour.</p>  
022 \*  
023 \* <p>The motivation for this implementation is to provide access to {@link DynaBean}  
024 \* properties in technologies that are unaware of BeanUtils and {@link DynaBean}s -  
025 \* such as the expression languages of JSTL and JSF.</p>  
026 \*  
027 \* <p>This can be achieved either by wrapping the {@link DynaBean} prior to  
028 \* providing it to the technolody to process or by providing a <code>Map</code>  
029 \* accessor method on the DynaBean implementation:  
030 \* <pre><code>  
031 \* public Map getMap() {  
032 \* return new DynaBeanMapDecorator(this);  
033 \* }</code></pre>  
034 \* </ul>  
035 \* </p>  
036 \*  
037 \* <p>This, for example, could be used in JSTL in the following way to access  
038 \* a DynaBean's <code>fooProperty</code>:  
039 \* <ul><li><code>${myDynaBean.<b>map</b>.fooProperty}</code></li></ul>  
040 \* </p>  
041 \*  
042 \* <h3>Usage</h3>  
043 \*  
044 \* <p>To decorate a {@link DynaBean} simply instantiate this class with the  
045 \* target {@link DynaBean}:</p>  
046 \*  
047 \* <ul><li><code>Map fooMap = new DynaBeanMapDecorator(fooDynaBean);</code></li></ul>  
048 \*  
049 \* <p>The above example creates a <b><i>read only</i></b> <code>Map</code>.  
050 \* To create a <code>Map</code> which can be modified, construct a  
051 \* <code>DynaBeanMapDecorator</code> with the <b><i>read only</i></b>  
052 \* attribute set to <code>false</code>:</p>  
053 \*  
054 \* <ul><li><code>Map fooMap = new DynaBeanMapDecorator(fooDynaBean, false);</code></li></ul>  
055 \*  
056 \* <h3>Limitations</h3>  
057 \* <p>In this implementation the <code>entrySet()</code>, <code>keySet()</code>  
058 \* and <code>values()</code> methods create an <b><i>unmodifiable</i></b>  
059 \* <code>Set</code> and it does not support the Map's <code>clear()</code>  
060 \* and <code>remove()</code> operations.</p>  
061 \* <p>For reasons of backwards compatibility, the generic types of this  
062 \* {@code Map} implementation are {@code <Object, Object>}. However, the  
063 \* keys of the map are typically strings.</p>  
064 \*  
065 \* @since BeanUtils 1.8.0  
066 \* @version $Id$  
067 \* @deprecated Use {@link DynaBeanPropertyMapDecorator} instead. When adding  
068 \* generics it turned out that it was not possible to use the correct type  
069 \* parameters without breaking backwards compatibility. Therefore, class  
070 \* {@code DynaBeanPropertyMapDecorator} was introduced as a replacement.  
071 \*/  
072@Deprecated  
073public class DynaBeanMapDecorator extends BaseDynaBeanMapDecorator<Object> {  
074 /\*\*  
075 \* Construct a Map for the specified {@link DynaBean}.  
076 \*  
077 \* @param dynaBean The dyna bean being decorated  
078 \* @param readOnly <code>true</code> if the Map is read only  
079 \* otherwise <code>false</code>  
080 \* @throws IllegalArgumentException if the {@link DynaBean} is null.  
081 \*/  
082 public DynaBeanMapDecorator(final DynaBean dynaBean, final boolean readOnly) {  
083 super(dynaBean, readOnly);  
084 }  
085  
086 /\*\*  
087 \* Constructs a read only Map for the specified  
088 \* {@link DynaBean}.  
089 \*  
090 \* @param dynaBean The dyna bean being decorated  
091 \* @throws IllegalArgumentException if the {@link DynaBean} is null.  
092 \*/  
093 public DynaBeanMapDecorator(final DynaBean dynaBean) {  
094 super(dynaBean);  
095 }  
096  
097 @Override  
098 protected Object convertKey(final String propertyName) {  
099 return propertyName;  
100 }  
101}