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  
018  
019package org.apache.commons.beanutils;  
020  
021  
022/\*\*  
023 \* <p>A <strong>DynaBean</strong> is a Java object that supports properties  
024 \* whose names and data types, as well as values, may be dynamically modified.  
025 \* To the maximum degree feasible, other components of the BeanUtils package  
026 \* will recognize such beans and treat them as standard JavaBeans for the  
027 \* purpose of retrieving and setting property values.</p>  
028 \*  
029 \* @version $Id$  
030 \*/  
031  
032public interface DynaBean {  
033  
034  
035 /\*\*  
036 \* Does the specified mapped property contain a value for the specified  
037 \* key value?  
038 \*  
039 \* @param name Name of the property to check  
040 \* @param key Name of the key to check  
041 \* @return <code>true<code> if the mapped property contains a value for  
042 \* the specified key, otherwise <code>false</code>  
043 \*  
044 \* @throws IllegalArgumentException if there is no property  
045 \* of the specified name  
046 \*/  
047 public boolean contains(String name, String key);  
048  
049  
050 /\*\*  
051 \* Return the value of a simple property with the specified name.  
052 \*  
053 \* @param name Name of the property whose value is to be retrieved  
054 \* @return The property's value  
055 \*  
056 \* @throws IllegalArgumentException if there is no property  
057 \* of the specified name  
058 \*/  
059 public Object get(String name);  
060  
061  
062 /\*\*  
063 \* Return the value of an indexed property with the specified name.  
064 \*  
065 \* @param name Name of the property whose value is to be retrieved  
066 \* @param index Index of the value to be retrieved  
067 \* @return The indexed property's value  
068 \*  
069 \* @throws IllegalArgumentException if there is no property  
070 \* of the specified name  
071 \* @throws IllegalArgumentException if the specified property  
072 \* exists, but is not indexed  
073 \* @throws IndexOutOfBoundsException if the specified index  
074 \* is outside the range of the underlying property  
075 \* @throws NullPointerException if no array or List has been  
076 \* initialized for this property  
077 \*/  
078 public Object get(String name, int index);  
079  
080  
081 /\*\*  
082 \* Return the value of a mapped property with the specified name,  
083 \* or <code>null</code> if there is no value for the specified key.  
084 \*  
085 \* @param name Name of the property whose value is to be retrieved  
086 \* @param key Key of the value to be retrieved  
087 \* @return The mapped property's value  
088 \*  
089 \* @throws IllegalArgumentException if there is no property  
090 \* of the specified name  
091 \* @throws IllegalArgumentException if the specified property  
092 \* exists, but is not mapped  
093 \*/  
094 public Object get(String name, String key);  
095  
096  
097 /\*\*  
098 \* Return the <code>DynaClass</code> instance that describes the set of  
099 \* properties available for this DynaBean.  
100 \*  
101 \* @return The associated DynaClass  
102 \*/  
103 public DynaClass getDynaClass();  
104  
105  
106 /\*\*  
107 \* Remove any existing value for the specified key on the  
108 \* specified mapped property.  
109 \*  
110 \* @param name Name of the property for which a value is to  
111 \* be removed  
112 \* @param key Key of the value to be removed  
113 \*  
114 \* @throws IllegalArgumentException if there is no property  
115 \* of the specified name  
116 \*/  
117 public void remove(String name, String key);  
118  
119  
120 /\*\*  
121 \* Set the value of a simple property with the specified name.  
122 \*  
123 \* @param name Name of the property whose value is to be set  
124 \* @param value Value to which this property is to be set  
125 \*  
126 \* @throws ConversionException if the specified value cannot be  
127 \* converted to the type required for this property  
128 \* @throws IllegalArgumentException if there is no property  
129 \* of the specified name  
130 \* @throws NullPointerException if an attempt is made to set a  
131 \* primitive property to null  
132 \*/  
133 public void set(String name, Object value);  
134  
135  
136 /\*\*  
137 \* Set the value of an indexed property with the specified name.  
138 \*  
139 \* @param name Name of the property whose value is to be set  
140 \* @param index Index of the property to be set  
141 \* @param value Value to which this property is to be set  
142 \*  
143 \* @throws ConversionException if the specified value cannot be  
144 \* converted to the type required for this property  
145 \* @throws IllegalArgumentException if there is no property  
146 \* of the specified name  
147 \* @throws IllegalArgumentException if the specified property  
148 \* exists, but is not indexed  
149 \* @throws IndexOutOfBoundsException if the specified index  
150 \* is outside the range of the underlying property  
151 \*/  
152 public void set(String name, int index, Object value);  
153  
154  
155 /\*\*  
156 \* Set the value of a mapped property with the specified name.  
157 \*  
158 \* @param name Name of the property whose value is to be set  
159 \* @param key Key of the property to be set  
160 \* @param value Value to which this property is to be set  
161 \*  
162 \* @throws ConversionException if the specified value cannot be  
163 \* converted to the type required for this property  
164 \* @throws IllegalArgumentException if there is no property  
165 \* of the specified name  
166 \* @throws IllegalArgumentException if the specified property  
167 \* exists, but is not mapped  
168 \*/  
169 public void set(String name, String key, Object value);  
170  
171  
172}