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.expression;  
018  
019/\*\*  
020 \* Property Name Expression Resolver.  
021 \* <p>  
022 \* Methods such as PropertyUtilsBean's <code>setNestedProperty()</code> method  
023 \* use a <code>Resolver</code> to process a <i>property name</i>  
024 \* expression and resolve <i>nested</i>, <i>indexed</i> and <i>mapped</i>  
025 \* property names. The following code provides an example usage  
026 \* demonstrating all the methods:  
027 \*  
028 \* <pre>  
029 \* // Iterate through a nested property expression  
030 \* while (resolver.hasNested(name)) {  
031 \*  
032 \* // isolate a single property from a nested expresion  
033 \* String next = resolver.next(name);  
034 \*  
035 \* // Process...  
036 \* String property = resolver.getProperty(next);  
037 \* if (resolver.isIndexed(next)) {  
038 \*  
039 \* int index = resolver.getIndex(next);  
040 \* bean = getIndexedProperty(bean, property, index);  
041 \*  
042 \* } else if (resolver.isMapped(next)) {  
043 \*  
044 \* String key = resolver.getKey(next);  
045 \* bean = getMappedProperty(bean, property, key);  
046 \*  
047 \* } else {  
048 \*  
049 \* bean = getSimpleProperty(bean, property);  
050 \*  
051 \* }  
052 \*  
053 \* // remove the processed property from the expression  
054 \* name = resolver.remove(name);  
055 \* }  
056 \* </pre>  
057 \*  
058 \* In order to create an implementation, it is important to understand how  
059 \* BeanUtils/PropertyUtils uses the <code>resolver</code>. The following are  
060 \* the main methods that use it:  
061 \* <ul>  
062 \* <li>{@link org.apache.commons.beanutils.PropertyUtilsBean}</li>  
063 \* <ul>  
064 \* <li>{@link org.apache.commons.beanutils.PropertyUtilsBean#getIndexedProperty(Object, String)}</li>  
065 \* <li>{@link org.apache.commons.beanutils.PropertyUtilsBean#getMappedProperty(Object, String)}</li>  
066 \* <li>{@link org.apache.commons.beanutils.PropertyUtilsBean#getNestedProperty(Object, String)}</li>  
067 \* <li>{@link org.apache.commons.beanutils.PropertyUtilsBean#getPropertyDescriptor(Object, String)}</li>  
068 \* <li>{@link org.apache.commons.beanutils.PropertyUtilsBean#getSimpleProperty(Object, String)}</li>  
069 \* <li>{@link org.apache.commons.beanutils.PropertyUtilsBean#setIndexedProperty(Object, String, Object)}</li>  
070 \* <li>{@link org.apache.commons.beanutils.PropertyUtilsBean#setMappedProperty(Object, String, Object)}</li>  
071 \* <li>{@link org.apache.commons.beanutils.PropertyUtilsBean#setNestedProperty(Object, String, Object)}</li>  
072 \* <li>{@link org.apache.commons.beanutils.PropertyUtilsBean#setSimpleProperty(Object, String, Object)}</li>  
073 \* </ul>  
074 \* <li>{@link org.apache.commons.beanutils.BeanUtilsBean}</li>  
075 \* <ul>  
076 \* <li>{@link org.apache.commons.beanutils.BeanUtilsBean#copyProperty(Object, String, Object)}</li>  
077 \* <li>{@link org.apache.commons.beanutils.BeanUtilsBean#setProperty(Object, String, Object)}</li>  
078 \* </ul>  
079 \* <li>{@link org.apache.commons.beanutils.locale.LocaleBeanUtilsBean}</li>  
080 \* <ul>  
081 \* <li>{@link org.apache.commons.beanutils.locale.LocaleBeanUtilsBean#setProperty(Object,  
082 \* String, Object, String)}</li>  
083 \* </ul>  
084 \* </ul>  
085 \*  
086 \* @version $Id$  
087 \* @see org.apache.commons.beanutils.PropertyUtilsBean#setResolver(Resolver)  
088 \* @since 1.8.0  
089 \*/  
090public interface Resolver {  
091  
092 /\*\*  
093 \* Extract the index value from the property expression or -1.  
094 \*  
095 \* @param expression The property expression  
096 \* @return The index value or -1 if the property is not indexed  
097 \* @throws IllegalArgumentException If the indexed property is illegally  
098 \* formed or has an invalid (non-numeric) value  
099 \*/  
100 int getIndex(String expression);  
101  
102 /\*\*  
103 \* Extract the map key from the property expression or <code>null</code>.  
104 \*  
105 \* @param expression The property expression  
106 \* @return The index value  
107 \* @throws IllegalArgumentException If the mapped property is illegally formed  
108 \*/  
109 String getKey(String expression);  
110  
111 /\*\*  
112 \* Return the property name from the property expression.  
113 \*  
114 \* @param expression The property expression  
115 \* @return The property name  
116 \*/  
117 String getProperty(String expression);  
118  
119 /\*\*  
120 \* Indicates whether or not the expression  
121 \* contains nested property expressions or not.  
122 \*  
123 \* @param expression The property expression  
124 \* @return The next property expression  
125 \*/  
126 boolean hasNested(String expression);  
127  
128 /\*\*  
129 \* Indicate whether the expression is for an indexed property or not.  
130 \*  
131 \* @param expression The property expression  
132 \* @return <code>true</code> if the expresion is indexed,  
133 \* otherwise <code>false</code>  
134 \*/  
135 boolean isIndexed(String expression);  
136  
137 /\*\*  
138 \* Indicate whether the expression is for a mapped property or not.  
139 \*  
140 \* @param expression The property expression  
141 \* @return <code>true</code> if the expresion is mapped,  
142 \* otherwise <code>false</code>  
143 \*/  
144 boolean isMapped(String expression);  
145  
146 /\*\*  
147 \* Extract the next property expression from the  
148 \* current expression.  
149 \*  
150 \* @param expression The property expression  
151 \* @return The next property expression  
152 \*/  
153 String next(String expression);  
154  
155 /\*\*  
156 \* Remove the last property expresson from the  
157 \* current expression.  
158 \*  
159 \* @param expression The property expression  
160 \* @return The new expression value, with first property  
161 \* expression removed - null if there are no more expressions  
162 \*/  
163 String remove(String expression);  
164  
165}