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  
022import java.sql.ResultSet;  
023import java.sql.SQLException;  
024import java.util.Iterator;  
025  
026  
027/\*\*  
028 \* <p>Implementation of <code>DynaClass</code> for DynaBeans that wrap the  
029 \* <code>java.sql.Row</code> objects of a <code>java.sql.ResultSet</code>.  
030 \* The normal usage pattern is something like:</p>  
031 \* <pre>  
032 \* ResultSet rs = ...;  
033 \* ResultSetDynaClass rsdc = new ResultSetDynaClass(rs);  
034 \* Iterator rows = rsdc.iterator();  
035 \* while (rows.hasNext()) {  
036 \* DynaBean row = (DynaBean) rows.next();  
037 \* ... process this row ...  
038 \* }  
039 \* rs.close();  
040 \* </pre>  
041 \*  
042 \* <p>Each column in the result set will be represented as a DynaBean  
043 \* property of the corresponding name (optionally forced to lower case  
044 \* for portability).</p>  
045 \*  
046 \* <p><strong>WARNING</strong> - Any {@link DynaBean} instance returned by  
047 \* this class, or from the <code>Iterator</code> returned by the  
048 \* <code>iterator()</code> method, is directly linked to the row that the  
049 \* underlying result set is currently positioned at. This has the following  
050 \* implications:</p>  
051 \* <ul>  
052 \* <li>Once you retrieve a different {@link DynaBean} instance, you should  
053 \* no longer use any previous instance.</li>  
054 \* <li>Changing the position of the underlying result set will change the  
055 \* data that the {@link DynaBean} references.</li>  
056 \* <li>Once the underlying result set is closed, the {@link DynaBean}  
057 \* instance may no longer be used.</li>  
058 \* </ul>  
059 \*  
060 \* <p>Any database data that you wish to utilize outside the context of the  
061 \* current row of an open result set must be copied. For example, you could  
062 \* use the following code to create standalone copies of the information in  
063 \* a result set:</p>  
064 \* <pre>  
065 \* ArrayList results = new ArrayList(); // To hold copied list  
066 \* ResultSetDynaClass rsdc = ...;  
067 \* DynaProperty[] properties = rsdc.getDynaProperties();  
068 \* BasicDynaClass bdc =  
069 \* new BasicDynaClass("foo", BasicDynaBean.class,  
070 \* rsdc.getDynaProperties());  
071 \* Iterator rows = rsdc.iterator();  
072 \* while (rows.hasNext()) {  
073 \* DynaBean oldRow = (DynaBean) rows.next();  
074 \* DynaBean newRow = bdc.newInstance();  
075 \* PropertyUtils.copyProperties(newRow, oldRow);  
076 \* results.add(newRow);  
077 \* }  
078 \* </pre>  
079 \*  
080 \* @version $Id$  
081 \*/  
082  
083public class ResultSetDynaClass extends JDBCDynaClass implements DynaClass {  
084  
085  
086 // ----------------------------------------------------------- Constructors  
087  
088  
089 /\*\*  
090 \* <p>Construct a new ResultSetDynaClass for the specified  
091 \* <code>ResultSet</code>. The property names corresponding  
092 \* to column names in the result set will be lower cased.</p>  
093 \*  
094 \* @param resultSet The result set to be wrapped  
095 \*  
096 \* @throws NullPointerException if <code>resultSet</code>  
097 \* is <code>null</code>  
098 \* @throws SQLException if the metadata for this result set  
099 \* cannot be introspected  
100 \*/  
101 public ResultSetDynaClass(final ResultSet resultSet) throws SQLException {  
102  
103 this(resultSet, true);  
104  
105 }  
106  
107  
108 /\*\*  
109 \* <p>Construct a new ResultSetDynaClass for the specified  
110 \* <code>ResultSet</code>. The property names corresponding  
111 \* to the column names in the result set will be lower cased or not,  
112 \* depending on the specified <code>lowerCase</code> value.</p>  
113 \*  
114 \* <p><strong>WARNING</strong> - If you specify <code>false</code>  
115 \* for <code>lowerCase</code>, the returned property names will  
116 \* exactly match the column names returned by your JDBC driver.  
117 \* Because different drivers might return column names in different  
118 \* cases, the property names seen by your application will vary  
119 \* depending on which JDBC driver you are using.</p>  
120 \*  
121 \* @param resultSet The result set to be wrapped  
122 \* @param lowerCase Should property names be lower cased?  
123 \*  
124 \* @throws NullPointerException if <code>resultSet</code>  
125 \* is <code>null</code>  
126 \* @throws SQLException if the metadata for this result set  
127 \* cannot be introspected  
128 \*/  
129 public ResultSetDynaClass(final ResultSet resultSet, final boolean lowerCase)  
130 throws SQLException {  
131  
132 this(resultSet, lowerCase, false);  
133  
134 }  
135  
136  
137 /\*\*  
138 \* <p>Construct a new ResultSetDynaClass for the specified  
139 \* <code>ResultSet</code>. The property names corresponding  
140 \* to the column names in the result set will be lower cased or not,  
141 \* depending on the specified <code>lowerCase</code> value.</p>  
142 \*  
143 \* <p><strong>WARNING</strong> - If you specify <code>false</code>  
144 \* for <code>lowerCase</code>, the returned property names will  
145 \* exactly match the column names returned by your JDBC driver.  
146 \* Because different drivers might return column names in different  
147 \* cases, the property names seen by your application will vary  
148 \* depending on which JDBC driver you are using.</p>  
149 \*  
150 \* @param resultSet The result set to be wrapped  
151 \* @param lowerCase Should property names be lower cased?  
152 \* @param useColumnLabel true if the column label should be used, otherwise false  
153 \*  
154 \* @throws NullPointerException if <code>resultSet</code>  
155 \* is <code>null</code>  
156 \* @throws SQLException if the metadata for this result set  
157 \* cannot be introspected  
158 \* @since 1.8.3  
159 \*/  
160 public ResultSetDynaClass(final ResultSet resultSet, final boolean lowerCase, final boolean useColumnLabel)  
161 throws SQLException {  
162  
163 if (resultSet == null) {  
164 throw new NullPointerException();  
165 }  
166 this.resultSet = resultSet;  
167 this.lowerCase = lowerCase;  
168 setUseColumnLabel(useColumnLabel);  
169 introspect(resultSet);  
170  
171 }  
172  
173  
174 // ----------------------------------------------------- Instance Variables  
175  
176  
177 /\*\*  
178 \* <p>The <code>ResultSet</code> we are wrapping.</p>  
179 \*/  
180 protected ResultSet resultSet = null;  
181  
182  
183 // --------------------------------------------------------- Public Methods  
184  
185  
186 /\*\*  
187 \* <p>Return an <code>Iterator</code> of {@link DynaBean} instances for  
188 \* each row of the wrapped <code>ResultSet</code>, in "forward" order.  
189 \* Unless the underlying result set supports scrolling, this method  
190 \* should be called only once.</p>  
191 \* @return An <code>Iterator</code> of {@link DynaBean} instances  
192 \*/  
193 public Iterator<DynaBean> iterator() {  
194  
195 return (new ResultSetIterator(this));  
196  
197 }  
198  
199  
200 /\*\*  
201 \* Get a value from the {@link ResultSet} for the specified  
202 \* property name.  
203 \*  
204 \* @param name The property name  
205 \* @return The value  
206 \* @throws SQLException if an error occurs  
207 \* @since 1.8.0  
208 \*/  
209 public Object getObjectFromResultSet(final String name) throws SQLException {  
210 return getObject(getResultSet(), name);  
211 }  
212  
213 // -------------------------------------------------------- Package Methods  
214  
215  
216 /\*\*  
217 \* <p>Return the result set we are wrapping.</p>  
218 \*/  
219 ResultSet getResultSet() {  
220  
221 return (this.resultSet);  
222  
223 }  
224  
225  
226 // ------------------------------------------------------ Protected Methods  
227  
228 /\*\*  
229 \* <p>Loads the class of the given name which by default uses the class loader used  
230 \* to load this library.  
231 \* Dervations of this class could implement alternative class loading policies such as  
232 \* using custom ClassLoader or using the Threads's context class loader etc.  
233 \* </p>  
234 \* @param className The name of the class to load  
235 \* @return The loaded class  
236 \* @throws SQLException if the class cannot be loaded  
237 \*/  
238 @Override  
239 protected Class<?> loadClass(final String className) throws SQLException {  
240  
241 try {  
242 return getClass().getClassLoader().loadClass(className);  
243 }  
244 catch (final Exception e) {  
245 throw new SQLException("Cannot load column class '" +  
246 className + "': " + e);  
247 }  
248 }  
249}