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.locale;  
019  
020import java.util.Locale;  
021  
022import org.apache.commons.collections.FastHashMap;  
023  
024/\*\*  
025 \* <p>Utility methods for converting locale-sensitive String scalar values to objects of the  
026 \* specified Class, String arrays to arrays of the specified Class and  
027 \* object to locale-sensitive String scalar value.</p>  
028 \*  
029 \* <p>The implementations for these method are provided by {@link LocaleConvertUtilsBean}.  
030 \* These static utility method use the default instance. More sophisticated can be provided  
031 \* by using a <code>LocaleConvertUtilsBean</code> instance.</p>  
032 \*  
033 \* @version $Id$  
034 \*/  
035public class LocaleConvertUtils {  
036  
037 // ----------------------------------------------------- Instance Variables  
038  
039 /\*\*  
040 \* <p>Gets the <code>Locale</code> which will be used when  
041 \* no <code>Locale</code> is passed to a method.</p>  
042 \*  
043 \* <p>For more details see <code>LocaleConvertUtilsBean</code></p>  
044 \* @return the default locale  
045 \* @see LocaleConvertUtilsBean#getDefaultLocale()  
046 \*/  
047 public static Locale getDefaultLocale() {  
048  
049 return LocaleConvertUtilsBean.getInstance().getDefaultLocale();  
050 }  
051  
052 /\*\*  
053 \* <p>Sets the <code>Locale</code> which will be used when  
054 \* no <code>Locale</code> is passed to a method.</p>  
055 \*  
056 \* <p>For more details see <code>LocaleConvertUtilsBean</code></p>  
057 \*  
058 \* @param locale the default locale  
059 \* @see LocaleConvertUtilsBean#setDefaultLocale(Locale)  
060 \*/  
061 public static void setDefaultLocale(final Locale locale) {  
062  
063 LocaleConvertUtilsBean.getInstance().setDefaultLocale(locale);  
064 }  
065  
066 /\*\*  
067 \* <p>Gets applyLocalized.</p>  
068 \*  
069 \* <p>For more details see <code>LocaleConvertUtilsBean</code></p>  
070 \*  
071 \* @return <code>true</code> if pattern is localized,  
072 \* otherwise <code>false</code>  
073 \* @see LocaleConvertUtilsBean#getApplyLocalized()  
074 \*/  
075 public static boolean getApplyLocalized() {  
076 return LocaleConvertUtilsBean.getInstance().getApplyLocalized();  
077 }  
078  
079 /\*\*  
080 \* <p>Sets applyLocalized.</p>  
081 \*  
082 \* <p>For more details see <code>LocaleConvertUtilsBean</code></p>  
083 \*  
084 \* @param newApplyLocalized <code>true</code> if pattern is localized,  
085 \* otherwise <code>false</code>  
086 \* @see LocaleConvertUtilsBean#setApplyLocalized(boolean)  
087 \*/  
088 public static void setApplyLocalized(final boolean newApplyLocalized) {  
089 LocaleConvertUtilsBean.getInstance().setApplyLocalized(newApplyLocalized);  
090 }  
091  
092 // --------------------------------------------------------- Methods  
093  
094 /\*\*  
095 \* <p>Convert the specified locale-sensitive value into a String.</p>  
096 \*  
097 \* <p>For more details see <code>LocaleConvertUtilsBean</code></p>  
098 \*  
099 \* @param value The Value to be converted  
100 \* @return the converted value  
101 \* @see LocaleConvertUtilsBean#convert(Object)  
102 \*/  
103 public static String convert(final Object value) {  
104 return LocaleConvertUtilsBean.getInstance().convert(value);  
105 }  
106  
107 /\*\*  
108 \* <p>Convert the specified locale-sensitive value into a String  
109 \* using the conversion pattern.</p>  
110 \*  
111 \* <p>For more details see <code>LocaleConvertUtilsBean</code></p>  
112 \*  
113 \* @param value The Value to be converted  
114 \* @param pattern The convertion pattern  
115 \* @return the converted value  
116 \* @see LocaleConvertUtilsBean#convert(Object, String)  
117 \*/  
118 public static String convert(final Object value, final String pattern) {  
119 return LocaleConvertUtilsBean.getInstance().convert(value, pattern);  
120 }  
121  
122 /\*\*  
123 \* <p>Convert the specified locale-sensitive value into a String  
124 \* using the paticular convertion pattern.</p>  
125 \*  
126 \* <p>For more details see <code>LocaleConvertUtilsBean</code></p>  
127 \*  
128 \* @param value The Value to be converted  
129 \* @param locale The locale  
130 \* @param pattern The convertion pattern  
131 \* @return the converted value  
132 \* @see LocaleConvertUtilsBean#convert(Object, Locale, String)  
133 \*/  
134 public static String convert(final Object value, final Locale locale, final String pattern) {  
135  
136 return LocaleConvertUtilsBean.getInstance().convert(value, locale, pattern);  
137 }  
138  
139 /\*\*  
140 \* <p>Convert the specified value to an object of the specified class (if  
141 \* possible). Otherwise, return a String representation of the value.</p>  
142 \*  
143 \* <p>For more details see <code>LocaleConvertUtilsBean</code></p>  
144 \*  
145 \* @param value The String scalar value to be converted  
146 \* @param clazz The Data type to which this value should be converted.  
147 \* @return the converted value  
148 \* @see LocaleConvertUtilsBean#convert(String, Class)  
149 \*/  
150 public static Object convert(final String value, final Class<?> clazz) {  
151  
152 return LocaleConvertUtilsBean.getInstance().convert(value, clazz);  
153 }  
154  
155 /\*\*  
156 \* <p>Convert the specified value to an object of the specified class (if  
157 \* possible) using the convertion pattern. Otherwise, return a String  
158 \* representation of the value.</p>  
159 \*  
160 \* <p>For more details see <code>LocaleConvertUtilsBean</code></p>  
161 \*  
162 \* @param value The String scalar value to be converted  
163 \* @param clazz The Data type to which this value should be converted.  
164 \* @param pattern The convertion pattern  
165 \* @return the converted value  
166 \* @see LocaleConvertUtilsBean#convert(String, Class, String)  
167 \*/  
168 public static Object convert(final String value, final Class<?> clazz, final String pattern) {  
169  
170 return LocaleConvertUtilsBean.getInstance().convert(value, clazz, pattern);  
171 }  
172  
173 /\*\*  
174 \* <p>Convert the specified value to an object of the specified class (if  
175 \* possible) using the convertion pattern. Otherwise, return a String  
176 \* representation of the value.</p>  
177 \*  
178 \* <p>For more details see <code>LocaleConvertUtilsBean</code></p>  
179 \*  
180 \* @param value The String scalar value to be converted  
181 \* @param clazz The Data type to which this value should be converted.  
182 \* @param locale The locale  
183 \* @param pattern The convertion pattern  
184 \* @return the converted value  
185 \* @see LocaleConvertUtilsBean#convert(String, Class, Locale, String)  
186 \*/  
187 public static Object convert(final String value, final Class<?> clazz, final Locale locale, final String pattern) {  
188  
189 return LocaleConvertUtilsBean.getInstance().convert(value, clazz, locale, pattern);  
190 }  
191  
192 /\*\*  
193 \* <p>Convert an array of specified values to an array of objects of the  
194 \* specified class (if possible) using the convertion pattern.</p>  
195 \*  
196 \* <p>For more details see <code>LocaleConvertUtilsBean</code></p>  
197 \*  
198 \* @param values Value to be converted (may be null)  
199 \* @param clazz Java array or element class to be converted to  
200 \* @param pattern The convertion pattern  
201 \* @return the converted value  
202 \* @see LocaleConvertUtilsBean#convert(String[], Class, String)  
203 \*/  
204 public static Object convert(final String[] values, final Class<?> clazz, final String pattern) {  
205  
206 return LocaleConvertUtilsBean.getInstance().convert(values, clazz, pattern);  
207 }  
208  
209 /\*\*  
210 \* <p>Convert an array of specified values to an array of objects of the  
211 \* specified class (if possible).</p>  
212 \*  
213 \* <p>For more details see <code>LocaleConvertUtilsBean</code></p>  
214 \*  
215 \* @param values Value to be converted (may be null)  
216 \* @param clazz Java array or element class to be converted to  
217 \* @return the converted value  
218 \* @see LocaleConvertUtilsBean#convert(String[], Class)  
219 \*/  
220 public static Object convert(final String[] values, final Class<?> clazz) {  
221  
222 return LocaleConvertUtilsBean.getInstance().convert(values, clazz);  
223 }  
224  
225 /\*\*  
226 \* <p>Convert an array of specified values to an array of objects of the  
227 \* specified class (if possible) using the convertion pattern.</p>  
228 \*  
229 \* <p>For more details see <code>LocaleConvertUtilsBean</code></p>  
230 \*  
231 \* @param values Value to be converted (may be null)  
232 \* @param clazz Java array or element class to be converted to  
233 \* @param locale The locale  
234 \* @param pattern The convertion pattern  
235 \* @return the converted value  
236 \* @see LocaleConvertUtilsBean#convert(String[], Class, Locale, String)  
237 \*/  
238 public static Object convert(final String[] values, final Class<?> clazz, final Locale locale, final String pattern) {  
239  
240 return LocaleConvertUtilsBean.getInstance().convert(values, clazz, locale, pattern);  
241 }  
242  
243 /\*\*  
244 \* <p>Register a custom {@link LocaleConverter} for the specified destination  
245 \* <code>Class</code>, replacing any previously registered converter.</p>  
246 \*  
247 \* <p>For more details see <code>LocaleConvertUtilsBean</code></p>  
248 \*  
249 \* @param converter The LocaleConverter to be registered  
250 \* @param clazz The Destination class for conversions performed by this  
251 \* Converter  
252 \* @param locale The locale  
253 \* @see LocaleConvertUtilsBean#register(LocaleConverter, Class, Locale)  
254 \*/  
255 public static void register(final LocaleConverter converter, final Class<?> clazz, final Locale locale) {  
256  
257 LocaleConvertUtilsBean.getInstance().register(converter, clazz, locale);  
258 }  
259  
260 /\*\*  
261 \* <p>Remove any registered {@link LocaleConverter}.</p>  
262 \*  
263 \* <p>For more details see <code>LocaleConvertUtilsBean</code></p>  
264 \*  
265 \* @see LocaleConvertUtilsBean#deregister()  
266 \*/  
267 public static void deregister() {  
268  
269 LocaleConvertUtilsBean.getInstance().deregister();  
270 }  
271  
272  
273 /\*\*  
274 \* <p>Remove any registered {@link LocaleConverter} for the specified locale.</p>  
275 \*  
276 \* <p>For more details see <code>LocaleConvertUtilsBean</code></p>  
277 \*  
278 \* @param locale The locale  
279 \* @see LocaleConvertUtilsBean#deregister(Locale)  
280 \*/  
281 public static void deregister(final Locale locale) {  
282  
283 LocaleConvertUtilsBean.getInstance().deregister(locale);  
284 }  
285  
286  
287 /\*\*  
288 \* <p>Remove any registered {@link LocaleConverter} for the specified locale and Class.</p>  
289 \*  
290 \* <p>For more details see <code>LocaleConvertUtilsBean</code></p>  
291 \*  
292 \* @param clazz Class for which to remove a registered Converter  
293 \* @param locale The locale  
294 \* @see LocaleConvertUtilsBean#deregister(Class, Locale)  
295 \*/  
296 public static void deregister(final Class<?> clazz, final Locale locale) {  
297  
298 LocaleConvertUtilsBean.getInstance().deregister(clazz, locale);  
299 }  
300  
301 /\*\*  
302 \* <p>Look up and return any registered {@link LocaleConverter} for the specified  
303 \* destination class and locale; if there is no registered Converter, return  
304 \* <code>null</code>.</p>  
305 \*  
306 \* <p>For more details see <code>LocaleConvertUtilsBean</code></p>  
307 \*  
308 \* @param clazz Class for which to return a registered Converter  
309 \* @param locale The Locale  
310 \* @return The registered locale Converter, if any  
311 \* @see LocaleConvertUtilsBean#lookup(Class, Locale)  
312 \*/  
313 public static LocaleConverter lookup(final Class<?> clazz, final Locale locale) {  
314  
315 return LocaleConvertUtilsBean.getInstance().lookup(clazz, locale);  
316 }  
317  
318 /\*\*  
319 \* <p>Look up and return any registered FastHashMap instance for the specified locale.</p>  
320 \*  
321 \* <p>For more details see <code>LocaleConvertUtilsBean</code></p>  
322 \*  
323 \* @param locale The Locale  
324 \* @return The FastHashMap instance contains the all {@link LocaleConverter} types for  
325 \* the specified locale.  
326 \* @see LocaleConvertUtilsBean#lookup(Locale)  
327 \* @deprecated This method will be modified to return a Map in the next release.  
328 \*/  
329 @Deprecated  
330 protected static FastHashMap lookup(final Locale locale) {  
331 return LocaleConvertUtilsBean.getInstance().lookup(locale);  
332 }  
333  
334 /\*\*  
335 \* <p>Create all {@link LocaleConverter} types for specified locale.</p>  
336 \*  
337 \* <p>For more details see <code>LocaleConvertUtilsBean</code></p>  
338 \*  
339 \* @param locale The Locale  
340 \* @return The FastHashMap instance contains the all {@link LocaleConverter} types  
341 \* for the specified locale.  
342 \* @see LocaleConvertUtilsBean#create(Locale)  
343 \* @deprecated This method will be modified to return a Map in the next release.  
344 \*/  
345 @Deprecated  
346 protected static FastHashMap create(final Locale locale) {  
347  
348 return LocaleConvertUtilsBean.getInstance().create(locale);  
349 }  
350}