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.converters;  
019  
020import org.apache.commons.beanutils.locale.BaseLocaleConverter;  
021import org.apache.commons.logging.Log;  
022import org.apache.commons.logging.LogFactory;  
023  
024import java.text.DecimalFormat;  
025import java.text.ParseException;  
026import java.util.Locale;  
027  
028  
029/\*\*  
030 \* <p>Standard {@link org.apache.commons.beanutils.locale.LocaleConverter}  
031 \* implementation that converts an incoming  
032 \* locale-sensitive String into a <code>java.lang.Number</code> object,  
033 \* optionally using a default value or throwing a  
034 \* {@link org.apache.commons.beanutils.ConversionException}  
035 \* if a conversion error occurs.</p>  
036 \*  
037 \* @since 1.7  
038 \* @version $Id$  
039 \*/  
040  
041public class DecimalLocaleConverter extends BaseLocaleConverter {  
042  
043  
044 // ----------------------------------------------------- Instance Variables  
045  
046 /\*\* All logging goes through this logger \*/  
047 private final Log log = LogFactory.getLog(DecimalLocaleConverter.class);  
048  
049 // ----------------------------------------------------------- Constructors  
050  
051 /\*\*  
052 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
053 \* that will throw a {@link org.apache.commons.beanutils.ConversionException}  
054 \* if a conversion error occurs. The locale is the default locale for  
055 \* this instance of the Java Virtual Machine and an unlocalized pattern is used  
056 \* for the convertion.  
057 \*  
058 \*/  
059 public DecimalLocaleConverter() {  
060  
061 this(false);  
062 }  
063  
064 /\*\*  
065 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
066 \* that will throw a {@link org.apache.commons.beanutils.ConversionException}  
067 \* if a conversion error occurs. The locale is the default locale for  
068 \* this instance of the Java Virtual Machine.  
069 \*  
070 \* @param locPattern Indicate whether the pattern is localized or not  
071 \*/  
072 public DecimalLocaleConverter(final boolean locPattern) {  
073  
074 this(Locale.getDefault(), locPattern);  
075 }  
076  
077 /\*\*  
078 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
079 \* that will throw a {@link org.apache.commons.beanutils.ConversionException}  
080 \* if a conversion error occurs. An unlocalized pattern is used for the convertion.  
081 \*  
082 \* @param locale The locale  
083 \*/  
084 public DecimalLocaleConverter(final Locale locale) {  
085  
086 this(locale, false);  
087 }  
088  
089 /\*\*  
090 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
091 \* that will throw a {@link org.apache.commons.beanutils.ConversionException}  
092 \* if a conversion error occurs.  
093 \*  
094 \* @param locale The locale  
095 \* @param locPattern Indicate whether the pattern is localized or not  
096 \*/  
097 public DecimalLocaleConverter(final Locale locale, final boolean locPattern) {  
098  
099 this(locale, (String) null, locPattern);  
100 }  
101  
102 /\*\*  
103 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
104 \* that will throw a {@link org.apache.commons.beanutils.ConversionException}  
105 \* if a conversion error occurs. An unlocalized pattern is used for the convertion.  
106 \*  
107 \* @param locale The locale  
108 \* @param pattern The convertion pattern  
109 \*/  
110 public DecimalLocaleConverter(final Locale locale, final String pattern) {  
111  
112 this(locale, pattern, false);  
113 }  
114  
115 /\*\*  
116 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
117 \* that will throw a {@link org.apache.commons.beanutils.ConversionException}  
118 \* if a conversion error occurs.  
119 \*  
120 \* @param locale The locale  
121 \* @param pattern The convertion pattern  
122 \* @param locPattern Indicate whether the pattern is localized or not  
123 \*/  
124 public DecimalLocaleConverter(final Locale locale, final String pattern, final boolean locPattern) {  
125  
126 super(locale, pattern, locPattern);  
127 }  
128  
129 /\*\*  
130 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
131 \* that will return the specified default value  
132 \* if a conversion error occurs. The locale is the default locale for  
133 \* this instance of the Java Virtual Machine and an unlocalized pattern is used  
134 \* for the convertion.  
135 \*  
136 \* @param defaultValue The default value to be returned  
137 \*/  
138 public DecimalLocaleConverter(final Object defaultValue) {  
139  
140 this(defaultValue, false);  
141 }  
142  
143 /\*\*  
144 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
145 \* that will return the specified default value  
146 \* if a conversion error occurs. The locale is the default locale for  
147 \* this instance of the Java Virtual Machine.  
148 \*  
149 \* @param defaultValue The default value to be returned  
150 \* @param locPattern Indicate whether the pattern is localized or not  
151 \*/  
152 public DecimalLocaleConverter(final Object defaultValue, final boolean locPattern) {  
153  
154 this(defaultValue, Locale.getDefault(), locPattern);  
155 }  
156  
157 /\*\*  
158 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
159 \* that will return the specified default value  
160 \* if a conversion error occurs. An unlocalized pattern is used for the convertion.  
161 \*  
162 \* @param defaultValue The default value to be returned  
163 \* @param locale The locale  
164 \*/  
165 public DecimalLocaleConverter(final Object defaultValue, final Locale locale) {  
166  
167 this(defaultValue, locale, false);  
168 }  
169  
170 /\*\*  
171 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
172 \* that will return the specified default value  
173 \* if a conversion error occurs.  
174 \*  
175 \* @param defaultValue The default value to be returned  
176 \* @param locale The locale  
177 \* @param locPattern Indicate whether the pattern is localized or not  
178 \*/  
179 public DecimalLocaleConverter(final Object defaultValue, final Locale locale, final boolean locPattern) {  
180  
181 this(defaultValue, locale, null, locPattern);  
182 }  
183  
184 /\*\*  
185 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
186 \* that will return the specified default value  
187 \* if a conversion error occurs. An unlocalized pattern is used for the convertion.  
188 \*  
189 \* @param defaultValue The default value to be returned  
190 \* @param locale The locale  
191 \* @param pattern The convertion pattern  
192 \*/  
193 public DecimalLocaleConverter(final Object defaultValue, final Locale locale, final String pattern) {  
194  
195 this(defaultValue, locale, pattern, false);  
196 }  
197  
198 /\*\*  
199 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
200 \* that will return the specified default value  
201 \* if a conversion error occurs.  
202 \*  
203 \* @param defaultValue The default value to be returned  
204 \* @param locale The locale  
205 \* @param pattern The convertion pattern  
206 \* @param locPattern Indicate whether the pattern is localized or not  
207 \*/  
208 public DecimalLocaleConverter(final Object defaultValue, final Locale locale, final String pattern, final boolean locPattern) {  
209  
210 super(defaultValue, locale, pattern, locPattern);  
211  
212 }  
213  
214 // --------------------------------------------------------- Methods  
215  
216 /\*\*  
217 \* Convert the specified locale-sensitive input object into an output  
218 \* object of the specified type.  
219 \*  
220 \* @param value The input object to be converted  
221 \* @param pattern The pattern is used for the convertion  
222 \* @return The converted value  
223 \*  
224 \* @throws org.apache.commons.beanutils.ConversionException if conversion  
225 \* cannot be performed successfully  
226 \* @throws ParseException if an error occurs parsing a String to a Number  
227 \*/  
228 @Override  
229 protected Object parse(final Object value, final String pattern) throws ParseException {  
230  
231 if (value instanceof Number) {  
232 return value;  
233 }  
234  
235 // Note that despite the ambiguous "getInstance" name, and despite the  
236 // fact that objects returned from this method have the same toString  
237 // representation, each call to getInstance actually returns a new  
238 // object.  
239 final DecimalFormat formatter = (DecimalFormat) DecimalFormat.getInstance(locale);  
240  
241 // if some constructors default pattern to null, it makes only sense  
242 // to handle null pattern gracefully  
243 if (pattern != null) {  
244 if (locPattern) {  
245 formatter.applyLocalizedPattern(pattern);  
246 } else {  
247 formatter.applyPattern(pattern);  
248 }  
249 } else {  
250 log.debug("No pattern provided, using default.");  
251 }  
252  
253 return formatter.parse((String) value);  
254 }  
255}