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 java.util.Locale;  
021import java.text.ParseException;  
022import org.apache.commons.beanutils.ConversionException;  
023  
024  
025/\*\*  
026 \* <p>Standard {@link org.apache.commons.beanutils.locale.LocaleConverter}  
027 \* implementation that converts an incoming  
028 \* locale-sensitive String into a <code>java.lang.Short</code> object,  
029 \* optionally using a default value or throwing a  
030 \* {@link org.apache.commons.beanutils.ConversionException}  
031 \* if a conversion error occurs.</p>  
032 \*  
033 \* @version $Id$  
034 \*/  
035  
036public class ShortLocaleConverter extends DecimalLocaleConverter {  
037  
038 // ----------------------------------------------------------- Constructors  
039  
040 /\*\*  
041 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
042 \* that will throw a {@link org.apache.commons.beanutils.ConversionException}  
043 \* if a conversion error occurs. The locale is the default locale for  
044 \* this instance of the Java Virtual Machine and an unlocalized pattern is used  
045 \* for the convertion.  
046 \*  
047 \*/  
048 public ShortLocaleConverter() {  
049  
050 this(false);  
051 }  
052  
053 /\*\*  
054 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
055 \* that will throw a {@link org.apache.commons.beanutils.ConversionException}  
056 \* if a conversion error occurs. The locale is the default locale for  
057 \* this instance of the Java Virtual Machine.  
058 \*  
059 \* @param locPattern Indicate whether the pattern is localized or not  
060 \*/  
061 public ShortLocaleConverter(final boolean locPattern) {  
062  
063 this(Locale.getDefault(), locPattern);  
064 }  
065  
066 /\*\*  
067 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
068 \* that will throw a {@link org.apache.commons.beanutils.ConversionException}  
069 \* if a conversion error occurs. An unlocalized pattern is used for the convertion.  
070 \*  
071 \* @param locale The locale  
072 \*/  
073 public ShortLocaleConverter(final Locale locale) {  
074  
075 this(locale, false);  
076 }  
077  
078 /\*\*  
079 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
080 \* that will throw a {@link org.apache.commons.beanutils.ConversionException}  
081 \* if a conversion error occurs.  
082 \*  
083 \* @param locale The locale  
084 \* @param locPattern Indicate whether the pattern is localized or not  
085 \*/  
086 public ShortLocaleConverter(final Locale locale, final boolean locPattern) {  
087  
088 this(locale, (String) null, locPattern);  
089 }  
090  
091 /\*\*  
092 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
093 \* that will throw a {@link org.apache.commons.beanutils.ConversionException}  
094 \* if a conversion error occurs. An unlocalized pattern is used for the convertion.  
095 \*  
096 \* @param locale The locale  
097 \* @param pattern The convertion pattern  
098 \*/  
099 public ShortLocaleConverter(final Locale locale, final String pattern) {  
100  
101 this(locale, pattern, false);  
102 }  
103  
104 /\*\*  
105 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
106 \* that will throw a {@link org.apache.commons.beanutils.ConversionException}  
107 \* if a conversion error occurs.  
108 \*  
109 \* @param locale The locale  
110 \* @param pattern The convertion pattern  
111 \* @param locPattern Indicate whether the pattern is localized or not  
112 \*/  
113 public ShortLocaleConverter(final Locale locale, final String pattern, final boolean locPattern) {  
114  
115 super(locale, pattern, locPattern);  
116 }  
117  
118 /\*\*  
119 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
120 \* that will return the specified default value  
121 \* if a conversion error occurs. The locale is the default locale for  
122 \* this instance of the Java Virtual Machine and an unlocalized pattern is used  
123 \* for the convertion.  
124 \*  
125 \* @param defaultValue The default value to be returned  
126 \*/  
127 public ShortLocaleConverter(final Object defaultValue) {  
128  
129 this(defaultValue, false);  
130 }  
131  
132 /\*\*  
133 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
134 \* that will return the specified default value  
135 \* if a conversion error occurs. The locale is the default locale for  
136 \* this instance of the Java Virtual Machine.  
137 \*  
138 \* @param defaultValue The default value to be returned  
139 \* @param locPattern Indicate whether the pattern is localized or not  
140 \*/  
141 public ShortLocaleConverter(final Object defaultValue, final boolean locPattern) {  
142  
143 this(defaultValue, Locale.getDefault(), locPattern);  
144 }  
145  
146 /\*\*  
147 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
148 \* that will return the specified default value  
149 \* if a conversion error occurs. An unlocalized pattern is used for the convertion.  
150 \*  
151 \* @param defaultValue The default value to be returned  
152 \* @param locale The locale  
153 \*/  
154 public ShortLocaleConverter(final Object defaultValue, final Locale locale) {  
155  
156 this(defaultValue, locale, false);  
157 }  
158  
159 /\*\*  
160 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
161 \* that will return the specified default value  
162 \* if a conversion error occurs.  
163 \*  
164 \* @param defaultValue The default value to be returned  
165 \* @param locale The locale  
166 \* @param locPattern Indicate whether the pattern is localized or not  
167 \*/  
168 public ShortLocaleConverter(final Object defaultValue, final Locale locale, final boolean locPattern) {  
169  
170 this(defaultValue, locale, null, locPattern);  
171 }  
172  
173 /\*\*  
174 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
175 \* that will return the specified default value  
176 \* if a conversion error occurs. An unlocalized pattern is used for the convertion.  
177 \*  
178 \* @param defaultValue The default value to be returned  
179 \* @param locale The locale  
180 \* @param pattern The convertion pattern  
181 \*/  
182 public ShortLocaleConverter(final Object defaultValue, final Locale locale, final String pattern) {  
183  
184 this(defaultValue, locale, pattern, false);  
185 }  
186  
187 /\*\*  
188 \* Create a {@link org.apache.commons.beanutils.locale.LocaleConverter}  
189 \* that will return the specified default value  
190 \* if a conversion error occurs.  
191 \*  
192 \* @param defaultValue The default value to be returned  
193 \* @param locale The locale  
194 \* @param pattern The convertion pattern  
195 \* @param locPattern Indicate whether the pattern is localized or not  
196 \*/  
197 public ShortLocaleConverter(final Object defaultValue, final Locale locale, final String pattern, final boolean locPattern) {  
198  
199 super(defaultValue, locale, pattern, locPattern);  
200 }  
201  
202 /\*\*  
203 \* Convert the specified locale-sensitive input object into an output object of the  
204 \* specified type. This method will return values of type Short.  
205 \*  
206 \* @param value The input object to be converted  
207 \* @param pattern The pattern is used for the convertion  
208 \* @return The converted value  
209 \*  
210 \* @throws org.apache.commons.beanutils.ConversionException if conversion cannot be performed  
211 \* successfully  
212 \* @throws ParseException if an error occurs parsing a String to a Number  
213 \* @since 1.8.0  
214 \*/  
215 @Override  
216 protected Object parse(final Object value, final String pattern) throws ParseException {  
217  
218 final Object result = super.parse(value, pattern);  
219  
220 if (result == null || result instanceof Short) {  
221 return result;  
222 }  
223  
224 final Number parsed = (Number)result;  
225 if (parsed.longValue() != parsed.shortValue()) {  
226 throw new ConversionException("Supplied number is not of type Short: " + parsed.longValue());  
227 }  
228  
229 // now returns property Short  
230 return new Short(parsed.shortValue());  
231 }  
232  
233}  
234