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