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 \*/  
017package org.apache.commons.beanutils.converters;  
018  
019  
020/\*\*  
021 \* {@link org.apache.commons.beanutils.Converter} implementaion that handles conversion  
022 \* to and from <b>java.lang.Character</b> objects.  
023 \* <p>  
024 \* Can be configured to either return a <i>default value</i> or throw a  
025 \* <code>ConversionException</code> if a conversion error occurs.  
026 \*  
027 \* @version $Id$  
028 \* @since 1.3  
029 \*/  
030public final class CharacterConverter extends AbstractConverter {  
031  
032 /\*\*  
033 \* Construct a <b>java.lang.Character</b> <i>Converter</i> that throws  
034 \* a <code>ConversionException</code> if an error occurs.  
035 \*/  
036 public CharacterConverter() {  
037 super();  
038 }  
039  
040 /\*\*  
041 \* Construct a <b>java.lang.Character</b> <i>Converter</i> that returns  
042 \* a default value if an error occurs.  
043 \*  
044 \* @param defaultValue The default value to be returned  
045 \* if the value to be converted is missing or an error  
046 \* occurs converting the value.  
047 \*/  
048 public CharacterConverter(final Object defaultValue) {  
049 super(defaultValue);  
050 }  
051  
052 /\*\*  
053 \* Return the default type this <code>Converter</code> handles.  
054 \*  
055 \* @return The default type this <code>Converter</code> handles.  
056 \* @since 1.8.0  
057 \*/  
058 @Override  
059 protected Class<?> getDefaultType() {  
060 return Character.class;  
061 }  
062  
063 /\*\*  
064 \* <p>Convert a java.lang.Class or object into a String.</p>  
065 \*  
066 \* @param value The input value to be converted  
067 \* @return the converted String value.  
068 \* @since 1.8.0  
069 \*/  
070 @Override  
071 protected String convertToString(final Object value) {  
072 final String strValue = value.toString();  
073 return strValue.length() == 0 ? "" : strValue.substring(0, 1);  
074 }  
075  
076 /\*\*  
077 \* <p>Convert the input object into a java.lang.Character.</p>  
078 \*  
079 \* @param <T> Target type of the conversion.  
080 \* @param type Data type to which this value should be converted.  
081 \* @param value The input value to be converted.  
082 \* @return The converted value.  
083 \* @throws Exception if conversion cannot be performed successfully  
084 \* @since 1.8.0  
085 \*/  
086 @Override  
087 protected <T> T convertToType(final Class<T> type, final Object value) throws Exception {  
088 if (Character.class.equals(type) || Character.TYPE.equals(type)) {  
089 return type.cast(new Character(value.toString().charAt(0)));  
090 }  
091  
092 throw conversionException(type, value);  
093 }  
094  
095}