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  
019import org.apache.commons.beanutils.Converter;  
020  
021/\*\*  
022 \* Provides a facade for {@link Converter} implementations  
023 \* preventing access to any public API in the implementation,  
024 \* other than that specified by {@link Converter}.  
025 \* <p />  
026 \* This implementation can be used to prevent registered {@link Converter}  
027 \* implementations that provide configuration options from being  
028 \* retrieved and modified.  
029 \*  
030 \* @version $Id$  
031 \* @since 1.8.0  
032 \*/  
033public final class ConverterFacade implements Converter {  
034  
035 private final Converter converter;  
036  
037 /\*\*  
038 \* Construct a converter which delegates to the specified  
039 \* {@link Converter} implementation.  
040 \*  
041 \* @param converter The converter to delegate to  
042 \*/  
043 public ConverterFacade(final Converter converter) {  
044 if (converter == null) {  
045 throw new IllegalArgumentException("Converter is missing");  
046 }  
047 this.converter = converter;  
048 }  
049  
050 /\*\*  
051 \* Convert the input object into an output object of the  
052 \* specified type by delegating to the underlying {@link Converter}  
053 \* implementation.  
054 \*  
055 \* @param <T> The result type of the conversion  
056 \* @param type Data type to which this value should be converted  
057 \* @param value The input value to be converted  
058 \* @return The converted value.  
059 \*/  
060 public <T> T convert(final Class<T> type, final Object value) {  
061 return converter.convert(type, value);  
062 }  
063  
064 /\*\*  
065 \* Provide a String representation of this facade implementation  
066 \* sand the underlying {@link Converter} it delegates to.  
067 \*  
068 \* @return A String representation of this facade implementation  
069 \* sand the underlying {@link Converter} it delegates to  
070 \*/  
071 @Override  
072 public String toString() {  
073 return "ConverterFacade[" + converter.toString() + "]";  
074 }  
075  
076}