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 \* {@link org.apache.commons.beanutils.Converter} implementaion that handles conversion  
021 \* to and from <b>java.lang.Class</b> objects.  
022 \* <p>  
023 \* The class will be loaded from the thread context class  
024 \* loader (if it exists); otherwise the class loader that loaded this class  
025 \* will be used.  
026 \* <p>  
027 \* Can be configured to either return a <i>default value</i> or throw a  
028 \* <code>ConversionException</code> if a conversion error occurs.  
029 \*  
030 \* @version $Id$  
031 \* @since 1.4  
032 \*/  
033public final class ClassConverter extends AbstractConverter {  
034  
035 /\*\*  
036 \* Construct a <b>java.lang.Class</b> <i>Converter</i> that throws  
037 \* a <code>ConversionException</code> if an error occurs.  
038 \*/  
039 public ClassConverter() {  
040 super();  
041 }  
042  
043 /\*\*  
044 \* Construct a <b>java.lang.Class</b> <i>Converter</i> that returns  
045 \* a default value if an error occurs.  
046 \*  
047 \* @param defaultValue The default value to be returned  
048 \* if the value to be converted is missing or an error  
049 \* occurs converting the value.  
050 \*/  
051 public ClassConverter(final Object defaultValue) {  
052 super(defaultValue);  
053 }  
054  
055 /\*\*  
056 \* Return the default type this <code>Converter</code> handles.  
057 \*  
058 \* @return The default type this <code>Converter</code> handles.  
059 \* @since 1.8.0  
060 \*/  
061 @Override  
062 protected Class<?> getDefaultType() {  
063 return Class.class;  
064 }  
065  
066 /\*\*  
067 \* <p>Convert a java.lang.Class or object into a String.</p>  
068 \*  
069 \* @param value The input value to be converted  
070 \* @return the converted String value.  
071 \* @since 1.8.0  
072 \*/  
073 @Override  
074 protected String convertToString(final Object value) {  
075 return (value instanceof Class) ? ((Class<?>)value).getName() : value.toString();  
076 }  
077  
078 /\*\*  
079 \* <p>Convert the input object into a java.lang.Class.</p>  
080 \*  
081 \* @param <T> Target type of the conversion.  
082 \* @param type Data type to which this value should be converted.  
083 \* @param value The input value to be converted.  
084 \* @return The converted value.  
085 \* @throws Throwable if an error occurs converting to the specified type  
086 \* @since 1.8.0  
087 \*/  
088 @Override  
089 protected <T> T convertToType(final Class<T> type, final Object value) throws Throwable {  
090 if (Class.class.equals(type)) {  
091 ClassLoader classLoader = Thread.currentThread()  
092 .getContextClassLoader();  
093 if (classLoader != null) {  
094 try {  
095 return type.cast(classLoader.loadClass(value.toString()));  
096 } catch (final ClassNotFoundException ex) {  
097 // Don't fail, carry on and try this class's class loader  
098 // (see issue# BEANUTILS-263)  
099 }  
100 }  
101  
102 // Try this class's class loader  
103 classLoader = ClassConverter.class.getClassLoader();  
104 return type.cast(classLoader.loadClass(value.toString()));  
105 }  
106  
107 throw conversionException(type, value);  
108 }  
109  
110}