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.collections4.functors;  
018  
019import java.lang.reflect.Constructor;  
020import java.lang.reflect.InvocationTargetException;  
021  
022import org.apache.commons.collections4.FunctorException;  
023import org.apache.commons.collections4.Transformer;  
024  
025/\*\*  
026 \* Transformer implementation that creates a new object instance by reflection.  
027 \* <p>  
028 \* <b>WARNING:</b> from v4.1 onwards this class will <b>not</b> be serializable anymore  
029 \* in order to prevent potential remote code execution exploits. Please refer to  
030 \* <a href="https://issues.apache.org/jira/browse/COLLECTIONS-580">COLLECTIONS-580</a>  
031 \* for more details.  
032 \* </p>  
033 \*  
034 \* @since 3.0  
035 \*/  
036public class InstantiateTransformer<T> implements Transformer<Class<? extends T>, T> {  
037  
038 /\*\* Singleton instance that uses the no arg constructor \*/  
039 @SuppressWarnings("rawtypes")  
040 private static final Transformer NO\_ARG\_INSTANCE = new InstantiateTransformer<>();  
041  
042 /\*\* The constructor parameter types \*/  
043 private final Class<?>[] iParamTypes;  
044 /\*\* The constructor arguments \*/  
045 private final Object[] iArgs;  
046  
047 /\*\*  
048 \* Get a typed no-arg instance.  
049 \*  
050 \* @param <T> the type of the objects to be created  
051 \* @return Transformer<Class<? extends T>, T>  
052 \*/  
053 public static <T> Transformer<Class<? extends T>, T> instantiateTransformer() {  
054 return NO\_ARG\_INSTANCE;  
055 }  
056  
057 /\*\*  
058 \* Transformer method that performs validation.  
059 \*  
060 \* @param <T> the type of the objects to be created  
061 \* @param paramTypes the constructor parameter types  
062 \* @param args the constructor arguments  
063 \* @return an instantiate transformer  
064 \* @throws IllegalArgumentException if paramTypes does not match args  
065 \*/  
066 public static <T> Transformer<Class<? extends T>, T> instantiateTransformer(final Class<?>[] paramTypes,  
067 final Object[] args) {  
068 if (((paramTypes == null) && (args != null))  
069 || ((paramTypes != null) && (args == null))  
070 || ((paramTypes != null) && (args != null) && (paramTypes.length != args.length))) {  
071 throw new IllegalArgumentException("Parameter types must match the arguments");  
072 }  
073  
074 if (paramTypes == null || paramTypes.length == 0) {  
075 return new InstantiateTransformer<>();  
076 }  
077 return new InstantiateTransformer<>(paramTypes, args);  
078 }  
079  
080 /\*\*  
081 \* Constructor for no arg instance.  
082 \*/  
083 private InstantiateTransformer() {  
084 super();  
085 iParamTypes = null;  
086 iArgs = null;  
087 }  
088  
089 /\*\*  
090 \* Constructor that performs no validation.  
091 \* Use <code>instantiateTransformer</code> if you want that.  
092 \* <p>  
093 \* Note: from 4.0, the input parameters will be cloned  
094 \*  
095 \* @param paramTypes the constructor parameter types  
096 \* @param args the constructor arguments  
097 \*/  
098 public InstantiateTransformer(final Class<?>[] paramTypes, final Object[] args) {  
099 super();  
100 iParamTypes = paramTypes != null ? paramTypes.clone() : null;  
101 iArgs = args != null ? args.clone() : null;  
102 }  
103  
104 /\*\*  
105 \* Transforms the input Class object to a result by instantiation.  
106 \*  
107 \* @param input the input object to transform  
108 \* @return the transformed result  
109 \*/  
110 @Override  
111 public T transform(final Class<? extends T> input) {  
112 try {  
113 if (input == null) {  
114 throw new FunctorException(  
115 "InstantiateTransformer: Input object was not an instanceof Class, it was a null object");  
116 }  
117 final Constructor<? extends T> con = input.getConstructor(iParamTypes);  
118 return con.newInstance(iArgs);  
119 } catch (final NoSuchMethodException ex) {  
120 throw new FunctorException("InstantiateTransformer: The constructor must exist and be public ");  
121 } catch (final InstantiationException ex) {  
122 throw new FunctorException("InstantiateTransformer: InstantiationException", ex);  
123 } catch (final IllegalAccessException ex) {  
124 throw new FunctorException("InstantiateTransformer: Constructor must be public", ex);  
125 } catch (final InvocationTargetException ex) {  
126 throw new FunctorException("InstantiateTransformer: Constructor threw an exception", ex);  
127 }  
128 }  
129  
130}