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.collections4.properties;  
019  
020import java.io.File;  
021import java.io.FileInputStream;  
022import java.io.FileNotFoundException;  
023import java.io.IOException;  
024import java.io.InputStream;  
025import java.io.Reader;  
026import java.net.URI;  
027import java.net.URL;  
028import java.nio.file.Files;  
029import java.nio.file.Path;  
030import java.nio.file.Paths;  
031import java.util.Properties;  
032  
033/\*\*  
034 \* Subclasses create and load {@link Properties} and subclasses of {@link Properties} like {@link SortedProperties}.  
035 \*  
036 \* @param <T> {@link Properties} or a subclass like {@link SortedProperties}.  
037 \* @see Properties  
038 \* @since 4.4  
039 \*/  
040public abstract class AbstractPropertiesFactory<T extends Properties> {  
041  
042 /\*\*  
043 \* Constructs an instance.  
044 \*/  
045 protected AbstractPropertiesFactory() {  
046 // no init.  
047 }  
048  
049 /\*\*  
050 \* Subclasses override to provide customized properties instances.  
051 \*  
052 \* @return a new Properties instance.  
053 \*/  
054 protected abstract T createProperties();  
055  
056 /\*\*  
057 \* Creates and loads properties from the given file.  
058 \*  
059 \* @param classLoader the class loader to use to get the named resource.  
060 \* @param name the location of the properties file.  
061 \* @return a new properties object.  
062 \* @throws IOException Thrown if an error occurred reading the input stream.  
063 \* @throws IllegalArgumentException Thrown if the input contains a malformed Unicode escape sequence.  
064 \*/  
065 public T load(final ClassLoader classLoader, final String name) throws IOException {  
066 try (final InputStream inputStream = classLoader.getResourceAsStream(name)) {  
067 return load(inputStream);  
068 }  
069 }  
070  
071 /\*\*  
072 \* Creates and loads properties from the given file.  
073 \*  
074 \* @param file the location of the properties file.  
075 \* @return a new properties object.  
076 \* @throws IOException Thrown if an error occurred reading the input stream.  
077 \* @throws IllegalArgumentException Thrown if the input contains a malformed Unicode escape sequence.  
078 \* @throws FileNotFoundException Thrown if the file does not exist, is a directory, or cannot be opened for  
079 \* reading.  
080 \* @throws SecurityException Thrown if a security manager's {@code checkRead} method denies read access to  
081 \* the file.  
082 \*/  
083 public T load(final File file) throws FileNotFoundException, IOException {  
084 try (final FileInputStream inputStream = new FileInputStream(file)) {  
085 return load(inputStream);  
086 }  
087 }  
088  
089 /\*\*  
090 \* Creates and loads properties from the given input stream.  
091 \*  
092 \* @param inputStream the location of the properties file.  
093 \* @return a new properties object.  
094 \* @throws IOException Thrown if an error occurred reading the input stream.  
095 \* @throws IllegalArgumentException Thrown if the input contains a malformed Unicode escape sequence.  
096 \*/  
097 public T load(final InputStream inputStream) throws IOException {  
098 if (inputStream == null) {  
099 return null;  
100 }  
101 final T properties = createProperties();  
102 properties.load(inputStream);  
103 return properties;  
104 }  
105  
106 /\*\*  
107 \* Creates and loads properties from the given path.  
108 \*  
109 \* @param path the location of the properties file.  
110 \* @return a new properties object.  
111 \* @throws IOException Thrown if an error occurred reading the input stream.  
112 \* @throws IllegalArgumentException Thrown if the input contains a malformed Unicode escape sequence.  
113 \*/  
114 public T load(final Path path) throws IOException {  
115 try (final InputStream inputStream = Files.newInputStream(path)) {  
116 return load(inputStream);  
117 }  
118 }  
119  
120 /\*\*  
121 \* Creates and loads properties from the given reader.  
122 \*  
123 \* @param reader the location of the properties file.  
124 \* @return a new properties object.  
125 \* @throws IOException Thrown if an error occurred reading the input stream.  
126 \* @throws IllegalArgumentException Thrown if the input contains a malformed Unicode escape sequence.  
127 \*/  
128 public T load(final Reader reader) throws IOException {  
129 final T properties = createProperties();  
130 properties.load(reader);  
131 return properties;  
132 }  
133  
134 /\*\*  
135 \* Creates and loads properties from the given file name.  
136 \*  
137 \* @param name the location of the properties file.  
138 \* @return a new properties object.  
139 \* @throws IOException Thrown if an error occurred reading the input stream.  
140 \* @throws IllegalArgumentException Thrown if the input contains a malformed Unicode escape sequence.  
141 \*/  
142 public T load(final String name) throws IOException {  
143 try (final FileInputStream inputStream = new FileInputStream(name)) {  
144 return load(inputStream);  
145 }  
146 }  
147  
148 /\*\*  
149 \* Creates and loads properties from the given URI.  
150 \*  
151 \* @param uri the location of the properties file.  
152 \* @return a new properties object.  
153 \* @throws IOException Thrown if an error occurred reading the input stream.  
154 \* @throws IllegalArgumentException Thrown if the input contains a malformed Unicode escape sequence.  
155 \*/  
156 public T load(final URI uri) throws IOException {  
157 return load(Paths.get(uri));  
158 }  
159  
160 /\*\*  
161 \* Creates and loads properties from the given URL.  
162 \*  
163 \* @param url the location of the properties file.  
164 \* @return a new properties object.  
165 \* @throws IOException Thrown if an error occurred reading the input stream.  
166 \* @throws IllegalArgumentException Thrown if the input contains a malformed Unicode escape sequence.  
167 \*/  
168 public T load(final URL url) throws IOException {  
169 try (final InputStream inputStream = url.openStream()) {  
170 return load(inputStream);  
171 }  
172 }  
173  
174}