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;  
018  
019import java.beans.PropertyDescriptor;  
020import java.util.Set;  
021  
022/\*\*  
023 \* <p>  
024 \* A context interface used during introspection for querying and setting  
025 \* property descriptors.  
026 \* </p>  
027 \* <p>  
028 \* An implementation of this interface is passed to {@link BeanIntrospector}  
029 \* objects during processing of a bean class. It allows the  
030 \* {@code BeanIntrospector} to deliver descriptors for properties it has  
031 \* detected. It is also possible to find out which properties have already been  
032 \* found by another {@code BeanIntrospector}; this allows multiple  
033 \* {@code BeanIntrospector} instances to collaborate.  
034 \* </p>  
035 \*  
036 \* @version $Id$  
037 \* @since 1.9  
038 \*/  
039public interface IntrospectionContext {  
040 /\*\*  
041 \* Returns the class that is subject of introspection.  
042 \*  
043 \* @return the current class  
044 \*/  
045 Class<?> getTargetClass();  
046  
047 /\*\*  
048 \* Adds the given property descriptor to this context. This method is called  
049 \* by a {@code BeanIntrospector} during introspection for each detected  
050 \* property. If this context already contains a descriptor for the affected  
051 \* property, it is overridden.  
052 \*  
053 \* @param desc the property descriptor  
054 \*/  
055 void addPropertyDescriptor(PropertyDescriptor desc);  
056  
057 /\*\*  
058 \* Adds an array of property descriptors to this context. Using this method  
059 \* multiple descriptors can be added at once.  
060 \*  
061 \* @param descriptors the array of descriptors to be added  
062 \*/  
063 void addPropertyDescriptors(PropertyDescriptor[] descriptors);  
064  
065 /\*\*  
066 \* Tests whether a descriptor for the property with the given name is  
067 \* already contained in this context. This method can be used for instance  
068 \* to prevent that an already existing property descriptor is overridden.  
069 \*  
070 \* @param name the name of the property in question  
071 \* @return <b>true</b> if a descriptor for this property has already been  
072 \* added, <b>false</b> otherwise  
073 \*/  
074 boolean hasProperty(String name);  
075  
076 /\*\*  
077 \* Returns the descriptor for the property with the given name or  
078 \* <b>null</b> if this property is unknown.  
079 \*  
080 \* @param name the name of the property in question  
081 \* @return the descriptor for this property or <b>null</b> if this property  
082 \* is unknown  
083 \*/  
084 PropertyDescriptor getPropertyDescriptor(String name);  
085  
086 /\*\*  
087 \* Removes the descriptor for the property with the given name.  
088 \*  
089 \* @param name the name of the affected property  
090 \*/  
091 void removePropertyDescriptor(String name);  
092  
093 /\*\*  
094 \* Returns a set with the names of all properties known to this context.  
095 \*  
096 \* @return a set with the known property names  
097 \*/  
098 Set<String> propertyNames();  
099}