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  
019/\*\*  
020 \* <p>A <strong>DynaClass</strong> is a simulation of the functionality of  
021 \* <code>java.lang.Class</code> for classes implementing the  
022 \* <code>DynaBean</code> interface. DynaBean instances that share the same  
023 \* DynaClass all have the same set of available properties, along with any  
024 \* associated data types, read-only states, and write-only states.</p>  
025 \*  
026 \* @version $Id$  
027 \*/  
028  
029public interface DynaClass {  
030  
031  
032 /\*\*  
033 \* Return the name of this DynaClass (analogous to the  
034 \* <code>getName()</code> method of <code>java.lang.Class</code), which  
035 \* allows the same <code>DynaClass</code> implementation class to support  
036 \* different dynamic classes, with different sets of properties.  
037 \*  
038 \* @return the name of the DynaClass  
039 \*/  
040 public String getName();  
041  
042  
043 /\*\*  
044 \* Return a property descriptor for the specified property, if it exists;  
045 \* otherwise, return <code>null</code>.  
046 \*  
047 \* @param name Name of the dynamic property for which a descriptor  
048 \* is requested  
049 \* @return The descriptor for the specified property  
050 \*  
051 \* @throws IllegalArgumentException if no property name is specified  
052 \*/  
053 public DynaProperty getDynaProperty(String name);  
054  
055  
056 /\*\*  
057 \* <p>Return an array of <code>ProperyDescriptors</code> for the properties  
058 \* currently defined in this DynaClass. If no properties are defined, a  
059 \* zero-length array will be returned.</p>  
060 \*  
061 \* <p><strong>FIXME</strong> - Should we really be implementing  
062 \* <code>getBeanInfo()</code> instead, which returns property descriptors  
063 \* and a bunch of other stuff?</p>  
064 \*  
065 \* @return the set of properties for this DynaClass  
066 \*/  
067 public DynaProperty[] getDynaProperties();  
068  
069  
070 /\*\*  
071 \* Instantiate and return a new DynaBean instance, associated  
072 \* with this DynaClass.  
073 \*  
074 \* @return A new <code>DynaBean</code> instance  
075 \*  
076 \* @throws IllegalAccessException if the Class or the appropriate  
077 \* constructor is not accessible  
078 \* @throws InstantiationException if this Class represents an abstract  
079 \* class, an array class, a primitive type, or void; or if instantiation  
080 \* fails for some other reason  
081 \*/  
082 public DynaBean newInstance()  
083 throws IllegalAccessException, InstantiationException;  
084  
085  
086}