org.oorexx.uno

Class RgfReflectUNO

```
java.lang.Object
  └org.oorexx.uno.RgfReflectUNO
```

public class **RgfReflectUNO**
extends java.lang.Object

This class allows reflecting UNO objects and types using the type descriptions and usually returns results as strings only (to facilitate interaction with scripting languages); originally devised for Open Object Rexx (cf. http://www.ooRexx.org).

```
------------------------ Apache Version 2.0 license -------------------------
    Copyright (C) 2006 Rony G. Flatscher

    Licensed under the Apache License, Version 2.0 (the "License");
    you may not use this file except in compliance with the License.
    You may obtain a copy of the License at

        http://www.apache.org/licenses/LICENSE-2.0

    Unless required by applicable law or agreed to in writing, software
    distributed under the License is distributed on an "AS IS" BASIS,
    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
    See the License for the specific language governing permissions and
    limitations under the License.
 ----------------------------------------------------------------------------
```

Version:
1.0, 2006-01-01

Author:
Rony G. Flatscher

Field Summary	
static java.lang.String	**version** Version string indicating version of this class (majorVersion*100+minorVersion concatenated with a dot and the sorted date of last change.

Method Summary	
static java.lang.Object	**findInterfaceWithMember**(java.lang.Object o, java.lang.String needle, boolean bReturnString, int howMany, boolean bExtendSearch) Looks for the interface in a service object (o) containing a member (a method or attribute) of the given name.
static java.lang.String	**getDefinition**(java.lang.Object o) Returns a blank delimited string containing the UNOIDL definitions of an UNO object (or a string denoting the fully qualified UNOIDL name).
static java.lang.String	**getInterfaceNamesViaReflection**(java.lang.Object o) Returns a blank delimited String of the interface names that are defined in UNOIDL for the service object.
static java.lang.String	**getProperties**(java.lang.Object o) Creates and returns a blank delimited string of property definitions available for the service object o.
static java.lang.String	**getServiceNamesViaReflection**(java.lang.Object o) Returns a blank delimited String of service names that are defined in UNOIDL for the service object.
static java.lang.String	**getTypeName**(java.lang.Object o) Returns string indicating the type of the supplied UNO object.
static java.lang.String	**getXTypeProviderTypeNames**(java.lang.Object o) Returns a blank delimited string of names of the provided interface types.
static java.lang.String	**queryInterfaceName**(java.lang.Object o, java.lang.String name, boolean bExtendSearch) Looks up a (partially qualified) given interface name case-insensitively and returns the fully-qualified mixed case interface name.
static java.lang.Object	**queryInterfaceObjectByName**(java.lang.Object o, java.lang.String name, boolean bExtendSearch) Looks up a (partially qualified) given interface name case-insensitively, carries out the appropriate UnoRuntime.queryInterface() and returns its result.

static java.lang.String	**queryServiceName**(java.lang.Object o, java.lang.String name) Looks up a (partially qualified) given service name case-insensitively and returns the fully-qualified mixed case service name.
static com.sun.star.uno.XComponentContext	**setContext**(com.sun.star.uno.XComponentContext ctxt) Sets the component context and gets singletons and type descriptions from it.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

version

public static java.lang.String **version**

Version string indicating version of this class (majorVersion*100+minorVersion concatenated with a dot and the sorted date of last change.

Method Detail

setContext

public static com.sun.star.uno.XComponentContext **setContext**(com.sun.star.uno.XComponentContext ctxt)

Sets the component context and gets singletons and type descriptions from it. (Will invoke the private method "setUpPropertyAttributesHashTable()".)

Parameters:
ctxt - component context object to be used for this class, if null, then a default component context is created using "com.sun.star.comp.helper.Bootstrap.bootstrap()".
Returns:
The component context object that is used for this class.

---

getTypeName

public static java.lang.String **getTypeName**(java.lang.Object o)

Returns string indicating the type of the supplied UNO object. All of the possible return values are documented at the end of getDefinition(Object o) ("...strings representing UNO-Types...").

Parameters:
o - UNO object or fully qualified string referring to an UNOIDL definition
Returns:
String spelling out the UNO type or empty string, if type cannot be determined

---

getDefinition

public static java.lang.String **getDefinition**(java.lang.Object o)

Returns a blank delimited string containing the UNOIDL definitions of an UNO object (or a string denoting the fully qualified UNOIDL name). Here are the encodings for the different UNOIDL types:

TypeClass	Encoding
UNO_CONSTANTS	UNO_CONSTANTS|fully-qualified-name|datatype member-name|value… *Remark:* the following UNO "datatype"s are allowed for constants: UNO_BOOLEAN (Java: boolean), UNO_BYTE (Java: byte), UNO_SHORT (Java: short), UNO_UNSIGNED_SHORT (Java: short), UNO_LONG (Java: int), UNO_UNSIGNED_LONG (Java: int), UNO_HYPER (Java: long), UNO_UNSIGNED_HYPER (Java: long), UNO_FLOAT (Java: float), UNO_DOUBLE (Java: double).
UNO_ENUM	UNO_ENUM|fully-qualified-name|default-value member-name|value… *Remark:* this is always of UNO type UNO_LONG (ie. a 32-bit integer, Java type: int)
UNO_EXCEPTION	UNO_EXCEPTION|fully-qualified-name member-name|fullDatatype…
UNO_INTERFACE	UNO_INTERFACE|fully-qualified-name member-name|member-definition… where "member-definition" is one of: • UNO_ATTRIBUTE|[READONLY]|fullDatatype... • UNO_METHOD|[ONEWAY]|return-value-fullDatatype|[argName:fullDatatype[,...]]|[exception[,...]]

UNO_MODULE	`UNO_MODULE|fully-qualified-name fully-qualified-member-name|UNO-Type…`
UNO_SERVICE	`UNO_SERVICE|fully-qualified-name|[implementationName] member-name|member-definition…` where "member-definition" is one of: • UNO_INTERFACE|[OPTIONAL]|defined_by_service • UNO_SERVICE|[OPTIONAL]|defined_by_service • UNO_PROPERTY|[modifier[,...]]|fullDatatype|defined_by_service *Remark:* if a service object is reflected that implements more than one service definition, than the "fully-qualified-name" of that compound service is created by concatenating all service names with the plus sign (+). Each of these constituting service definitions (if available via reflection) is then used to create the entire definition of that "compound service" object in hand, documenting all defined interfaces, services and properties.
UNO_SINGLETON	`UNO_SINGLETON|fully-qualified-name|[old-style-servicename]` *Remark:* old-style-servicename or empty string if an instance of an interface
UNO_STRUCT	`UNO_STRUCT|fully-qualified-name member-name|fullDatatype…`
UNO_TYPEDEF	`UNO_TYPEDEF|fully-qualified-name referenced-type|UNO-Type`

`"fullDatatype"` is encoded as follows:

`fully-qualified-datatype-name:UNO-TypeClass:[referenced-type]:[UNO-TypeClass]`

If a datatype is of type `UNO_TYPEDEF`, then its "referenced-type-name" is given together with its respective UNO-TypeClass.

The following strings representing UNO-Types ("TypeClasses") may be encountered:

`"UNO_ANY", "UNO_ARRAY", "UNO_BOOLEAN", "UNO_BYTE", "UNO_CHAR", "UNO_CONSTANT", "UNO_CONSTANTS", "UNO_DOUBLE", "UNO_ENUM", "UNO_EXCEPTION", "UNO_FLOAT", "UNO_HYPER", "UNO_INTERFACE", "UNO_INTERFACE_ATTRIBUTE" "UNO_INTERFACE_METHOD", "UNO_LONG", "UNO_MODULE", "UNO_PROPERTY", "UNO_SEQUENCE", "UNO_SERVICE", "UNO_SHORT", "UNO_SINGLETON", "UNO_STRING", "UNO_STRUCT", "UNO_TYPE", "UNO_TYPEDEF", "UNO_UNION", "UNO_UNKNOWN", "UNO_UNSIGNED_HYPER", "UNO_UNSIGNED_LONG", "UNO_UNSIGNED_SHORT", "UNO_VOID".`

Remark: The UNO "`TypeClass`" constant names use the respective names above, but without the lead-in string "`UNO_`" (this makes the type information from methods of this class unambiguous).

The UNO datatypes map to Java as follows:

UNO Datatype	Java Datatype
UNO_ANY	com.sun.star.uno.Any or java.lang.Object
UNO_VOID	void
UNO_BOOLEAN	boolean
UNO_BYTE (8-bit)	byte
UNO_CHAR (16-bit)	char
UNO_SHORT (16-bit)	short
UNO_UNSIGNED_SHORT (16-bit)	short
UNO_LONG (32-bit)	int
UNO_UNSIGNED_LONG (32-bit)	int
UNO_HYPER (64-bit)	long
UNO_UNSIGNED_HYPER (64-bit)	long
UNO_FLOAT	float
UNO_DOUBLE	double

Parameters:
o - UNO service object or string fully qualifying a service definition in an UNOIDL module

Returns:
string containing the UNOIDL definitions, each element delimited with a blank, each element's definition parts delimited with a vertical bar (|).

---

getXTypeProviderTypeNames

```
public static java.lang.String getXTypeProviderTypeNames(java.lang.Object o)
```

Returns a blank delimited string of names of the provided interface types. Cf. UNO's `com.sun.star.lang.XTypeProvider` interface which is applied to the service object `o`.

Parameters:
`o` - a service object which gets queried of its provided interface types using the XTypeProvider interface
Returns:
blank delimited string of interface names, empty string if none available

---

queryInterfaceObjectByName

```
public static java.lang.Object queryInterfaceObjectByName(java.lang.Object o,
                                                          java.lang.String name,
                                                          boolean bExtendSearch)
```

Looks up a (partially qualified) given interface name case-insensitively, carries out the appropriate `UnoRuntime.queryInterface()` and returns its result.

Parameters:
`name` - which may qualify the desired interface fully or partly, case does not matter either. If `name` denotes an existing Java class, that class is used to carry out the UnoRuntime.queryInterface(...). Otherwise an interface class is searched at runtime using `XTypeProvider` and possibly reflection.
`o` - is the object from which the interface should be queried from
`bExtendSearch` - `true`: if interface cannot be found in the `XTypeProvider` list, then extend search using full reflection; it could be the case that `XTypeProvider` does not provide all available interfaces! (as of OOo 2.0, 2005-12-20)
Returns:
the desired interface object, if successful, `null` else

---

getInterfaceNamesViaReflection

```
public static java.lang.String getInterfaceNamesViaReflection(java.lang.Object o)
```

Returns a blank delimited String of the interface names that are defined in UNOIDL for the service object.

Parameters:
`o` - service object to analyze
Returns:
a blank delimited string of service names (or empty string)

---

getServiceNamesViaReflection

```
public static java.lang.String getServiceNamesViaReflection(java.lang.Object o)
```

Returns a blank delimited String of service names that are defined in UNOIDL for the service object.

Parameters:
`o` - service object to analyze
Returns:
a blank delimited string of service names (or empty string)

---

getProperties

```
public static java.lang.String getProperties(java.lang.Object o)
```

Creates and returns a blank delimited string of property definitions available for the service object `o`. To find out about all available interfaces in a service object at runtime use `getXTypeProviderTypeNames(Object o)`, to find the interface in which a member (a method or an attribute) got defined at runtime use `findInterfaceWithMember(Object o, String needle, boolean bReturnString, int howMany, boolean bExtendSearch)`.

Parameters:
`o` - a service object
Returns:
blank delimited string of property definitions in the form
`property-name|full-datatype-name:UNO-TypeClass:[referenced-type]:[UNTO-TypeClass]`.

---

findInterfaceWithMember

```
public static java.lang.Object findInterfaceWithMember(java.lang.Object o,
                                                       java.lang.String needle,
                                                       boolean bReturnString,
                                                       int howMany,
                                                       boolean bExtendSearch)
```

Looks for the interface in a service object (o) containing a member (a method or attribute) of the given `name`. This method uses the `XTypeProvider` interface to search the available interfaces at runtime. To find out about all properties of a service object at runtime use `getProperties(Object o)`.

Parameters:

o - service object to analyze
`needle` - denotes the name to look for case-independently
`bReturnString` - if `true` then the string encoded INTERFACE information (cf. `getDefinition(Object o)`) is returned having the denoted member(s), otherwise the queried interface object is returned, which can be immediately addressed to invoke the member method or access the member attribute.
`howMany` - determines how many members matching `needle` should be searched and returned. Only relevant, if `bReturnString` is `true`. interfaces that contain the sought for member should be returned. A number less than 1 indicates to return all matching interface definitions (separated by a newline character 0x0A).
`bExtendSearch` - `true`: if interface cannot be found in the `XTypeProvider` list, then extend search using full reflection; it could be the case that `XTypeProvider` does not provide all available interfaces! (as of OOo 2.0, 2005-12-20)

Returns:
either a string (`bReturnString==true`, can be empty, if no definitions were found) or the desired interface object (`bReturnString==false`, can be null `null`) depending on the value of parameter

---

queryInterfaceName

```
public static java.lang.String queryInterfaceName(java.lang.Object o,
                                                   java.lang.String name,
                                                   boolean bExtendSearch)
```

Looks up a (partially qualified) given interface name case-insensitively and returns the fully-qualified mixed case interface name.

Parameters:
o - service object to analyze
`name` - denotes the name to look for case-independently
`bExtendSearch` - `true`: if interface cannot be found in the `XTypeProvider` list, then extend search using full reflection; it could be the case that `XTypeProvider` does not provide all available interfaces! (as of OOo 2.0, 2005-12-20)

Returns:
fully-qualified interface name or empty string, if not found.

---

queryServiceName

```
public static java.lang.String queryServiceName(java.lang.Object o,
                                                 java.lang.String name)
```

Looks up a (partially qualified) given service name case-insensitively and returns the fully-qualified mixed case service name.

Parameters:
o - service object to analyze
`name` - denotes the name to look for case-independently

Returns:
fully-qualified service name or empty string, if not found.
