| | [**Overview**](http://docs.google.com/overview-summary.html) | **Package** | Class | [**Use**](http://docs.google.com/package-use.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV PACKAGE**](http://docs.google.com/javax/management/openmbean/package-summary.html)   [**NEXT PACKAGE**](http://docs.google.com/javax/management/remote/package-summary.html) | [**FRAMES**](http://docs.google.com/index.html?javax/management/relation/package-summary.html)    [**NO FRAMES**](http://docs.google.com/package-summary.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |

## Package javax.management.relation

Provides the definition of the Relation Service.

**See:**

[**Description**](#3znysh7)

| **Interface Summary** | |
| --- | --- |
| [**Relation**](http://docs.google.com/javax/management/relation/Relation.html) | This interface has to be implemented by any MBean class expected to represent a relation managed using the Relation Service. |
| [**RelationServiceMBean**](http://docs.google.com/javax/management/relation/RelationServiceMBean.html) | The Relation Service is in charge of creating and deleting relation types and relations, of handling the consistency and of providing query mechanisms. |
| [**RelationSupportMBean**](http://docs.google.com/javax/management/relation/RelationSupportMBean.html) | A RelationSupport object is used internally by the Relation Service to represent simple relations (only roles, no properties or methods), with an unlimited number of roles, of any relation type. |
| [**RelationType**](http://docs.google.com/javax/management/relation/RelationType.html) | The RelationType interface has to be implemented by any class expected to represent a relation type. |

| **Class Summary** | |
| --- | --- |
| [**MBeanServerNotificationFilter**](http://docs.google.com/javax/management/relation/MBeanServerNotificationFilter.html) | Filter for [MBeanServerNotification](http://docs.google.com/javax/management/MBeanServerNotification.html). |
| [**RelationNotification**](http://docs.google.com/javax/management/relation/RelationNotification.html) | A notification of a change in the Relation Service. |
| [**RelationService**](http://docs.google.com/javax/management/relation/RelationService.html) | The Relation Service is in charge of creating and deleting relation types and relations, of handling the consistency and of providing query mechanisms. |
| [**RelationSupport**](http://docs.google.com/javax/management/relation/RelationSupport.html) | A RelationSupport object is used internally by the Relation Service to represent simple relations (only roles, no properties or methods), with an unlimited number of roles, of any relation type. |
| [**RelationTypeSupport**](http://docs.google.com/javax/management/relation/RelationTypeSupport.html) | A RelationTypeSupport object implements the RelationType interface. |
| [**Role**](http://docs.google.com/javax/management/relation/Role.html) | Represents a role: includes a role name and referenced MBeans (via their ObjectNames). |
| [**RoleInfo**](http://docs.google.com/javax/management/relation/RoleInfo.html) | A RoleInfo object summarises a role in a relation type. |
| [**RoleList**](http://docs.google.com/javax/management/relation/RoleList.html) | A RoleList represents a list of roles (Role objects). |
| [**RoleResult**](http://docs.google.com/javax/management/relation/RoleResult.html) | Represents the result of a multiple access to several roles of a relation (either for reading or writing). |
| [**RoleStatus**](http://docs.google.com/javax/management/relation/RoleStatus.html) | This class describes the various problems which can be encountered when accessing a role. |
| [**RoleUnresolved**](http://docs.google.com/javax/management/relation/RoleUnresolved.html) | Represents an unresolved role: a role not retrieved from a relation due to a problem. |
| [**RoleUnresolvedList**](http://docs.google.com/javax/management/relation/RoleUnresolvedList.html) | A RoleUnresolvedList represents a list of RoleUnresolved objects, representing roles not retrieved from a relation due to a problem encountered when trying to access (read or write) the roles. |

| **Exception Summary** | |
| --- | --- |
| [**InvalidRelationIdException**](http://docs.google.com/javax/management/relation/InvalidRelationIdException.html) | This exception is raised when relation id provided for a relation is already used. |
| [**InvalidRelationServiceException**](http://docs.google.com/javax/management/relation/InvalidRelationServiceException.html) | This exception is raised when an invalid Relation Service is provided. |
| [**InvalidRelationTypeException**](http://docs.google.com/javax/management/relation/InvalidRelationTypeException.html) | Invalid relation type. |
| [**InvalidRoleInfoException**](http://docs.google.com/javax/management/relation/InvalidRoleInfoException.html) | This exception is raised when, in a role info, its minimum degree is greater than its maximum degree. |
| [**InvalidRoleValueException**](http://docs.google.com/javax/management/relation/InvalidRoleValueException.html) | Role value is invalid. |
| [**RelationException**](http://docs.google.com/javax/management/relation/RelationException.html) | This class is the superclass of any exception which can be raised during relation management. |
| [**RelationNotFoundException**](http://docs.google.com/javax/management/relation/RelationNotFoundException.html) | This exception is raised when there is no relation for a given relation id in a Relation Service. |
| [**RelationServiceNotRegisteredException**](http://docs.google.com/javax/management/relation/RelationServiceNotRegisteredException.html) | This exception is raised when an access is done to the Relation Service and that one is not registered. |
| [**RelationTypeNotFoundException**](http://docs.google.com/javax/management/relation/RelationTypeNotFoundException.html) | This exception is raised when there is no relation type with given name in Relation Service. |
| [**RoleInfoNotFoundException**](http://docs.google.com/javax/management/relation/RoleInfoNotFoundException.html) | This exception is raised when there is no role info with given name in a given relation type. |
| [**RoleNotFoundException**](http://docs.google.com/javax/management/relation/RoleNotFoundException.html) | This exception is raised when a role in a relation does not exist, or is not readable, or is not settable. |

## Package javax.management.relation Description

Provides the definition of the Relation Service. The Relation Service is used to record relationships between MBeans in an MBean Server. The Relation Service is itself an MBean. More than one instance of a [RelationService](http://docs.google.com/javax/management/relation/RelationService.html) MBean can be registered in an MBean Server.

A *relation type* defines a relationship between MBeans. It contains *roles* that the MBeans play in the relationship. Usually there are at least two roles in a relation type.

A *relation* is a named instance of a relation type, where specific MBeans appear in the roles, represented by their [ObjectName](http://docs.google.com/javax/management/ObjectName.html)s.

For example, suppose there are Module MBeans, representing modules within an application. A DependsOn relation type could express the relationship that some modules depend on others, which could be used to determine the order in which the modules are started or stopped. The DependsOn relation type would have two roles, dependent and dependedOn.

Every role is *typed*, meaning that an MBean that appears in that role must be an instance of the role's type. In the DependsOn example, both roles would be of type Module.

Every role has a *cardinality*, which provides lower and upper bounds on the number of MBeans that can appear in that role in a given relation instance. Usually, the lower and upper bounds are both 1, with exactly one MBean appearing in the role. The cardinality only limits the number of MBeans in the role per relation instance. The same MBean can appear in the same role in any number of instances of a relation type. In the DependsOn example, a given module can depend on many other modules, and be depended on by many others, but any given relation instance links exactly one dependent module with exactly one dependedOn module.

A relation type can be created explicitly, as an object implementing the [RelationType](http://docs.google.com/javax/management/relation/RelationType.html) interface, typically a [RelationTypeSupport](http://docs.google.com/javax/management/relation/RelationTypeSupport.html). Alternatively, it can be created implicitly using the Relation Service's [createRelationType](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#createRelationType(java.lang.String,%20javax.management.relation.RoleInfo%5B%5D)) method.

A relation instance can be created explicitly, as an object implementing the [Relation](http://docs.google.com/javax/management/relation/Relation.html) interface, typically a [RelationSupport](http://docs.google.com/javax/management/relation/RelationSupport.html). (A RelationSupport is itself a valid MBean, so it can be registered in the MBean Server, though this is not required.) Alternatively, a relation instance can be created implicitly using the Relation Service's [createRelation](http://docs.google.com/javax/management/relation/RelationServiceMBean.html#createRelation(java.lang.String,%20java.lang.String,%20javax.management.relation.RoleList)) method.

The DependsOn example might be coded as follows.

import java.util.\*;  
import javax.management.\*;  
import javax.management.relation.\*;  
  
// ...  
MBeanServer mbs = ...;  
  
// Create the Relation Service MBean  
ObjectName relSvcName = new ObjectName(":type=RelationService");  
RelationService relSvcObject = new RelationService(true);  
mbs.registerMBean(relSvcObject, relSvcName);  
  
// Create an MBean proxy for easier access to the Relation Service  
RelationServiceMBean relSvc =  
 MBeanServerInvocationHandler.newProxyInstance(mbs, relSvcName,  
 RelationServiceMBean.class,  
 false);  
  
// Define the DependsOn relation type  
RoleInfo[] dependsOnRoles = {  
 new RoleInfo("dependent", Module.class.getName()),  
 new RoleInfo("dependedOn", Module.class.getName())  
};  
relSvc.createRelationType("DependsOn", dependsOnRoles);  
  
// Now define a relation instance "moduleA DependsOn moduleB"  
  
ObjectName moduleA = new ObjectName(":type=Module,name=A");  
ObjectName moduleB = new ObjectName(":type=Module,name=B");  
  
Role dependent = new Role("dependent", Collections.singletonList(moduleA));  
Role dependedOn = new Role("dependedOn", Collections.singletonList(moduleB));  
Role[] roleArray = {dependent, dependedOn};  
RoleList roles = new RoleList(Arrays.asList(roleArray));  
relSvc.createRelation("A-DependsOn-B", "DependsOn", roles);  
  
// Query the Relation Service to find what modules moduleA depends on  
Map<ObjectName,List<String>> dependentAMap =  
 relSvc.findAssociatedMBeans(moduleA, "DependsOn", "dependent");  
Set<ObjectName> dependentASet = dependentAMap.keySet();  
// Set of ObjectName containing moduleB

**Since:** 1.5 **See Also:** [Java SE 6 Platform documentation on JMX technology](http://docs.google.com/technotes/guides/jmx/), in particular the  [JMX Specification, version 1.4](http://docs.google.com/technotes/guides/jmx/JMX_1_4_specification.pdf)

| | [**Overview**](http://docs.google.com/overview-summary.html) | **Package** | Class | [**Use**](http://docs.google.com/package-use.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV PACKAGE**](http://docs.google.com/javax/management/openmbean/package-summary.html)   [**NEXT PACKAGE**](http://docs.google.com/javax/management/remote/package-summary.html) | [**FRAMES**](http://docs.google.com/index.html?javax/management/relation/package-summary.html)    [**NO FRAMES**](http://docs.google.com/package-summary.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).