| | [**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/timer/package-summary.html)   [**NEXT PACKAGE**](http://docs.google.com/javax/naming/directory/package-summary.html) | [**FRAMES**](http://docs.google.com/index.html?javax/naming/package-summary.html)    [**NO FRAMES**](http://docs.google.com/package-summary.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |

## Package javax.naming

Provides the classes and interfaces for accessing naming services.

**See:**

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

| **Interface Summary** | |
| --- | --- |
| [**Context**](http://docs.google.com/javax/naming/Context.html) | This interface represents a naming context, which consists of a set of name-to-object bindings. |
| [**Name**](http://docs.google.com/javax/naming/Name.html) | The Name interface represents a generic name -- an ordered sequence of components. |
| [**NameParser**](http://docs.google.com/javax/naming/NameParser.html) | This interface is used for parsing names from a hierarchical namespace. |
| [**NamingEnumeration<T>**](http://docs.google.com/javax/naming/NamingEnumeration.html) | This interface is for enumerating lists returned by methods in the javax.naming and javax.naming.directory packages. |
| [**Referenceable**](http://docs.google.com/javax/naming/Referenceable.html) | This interface is implemented by an object that can provide a Reference to itself. |

| **Class Summary** | |
| --- | --- |
| [**BinaryRefAddr**](http://docs.google.com/javax/naming/BinaryRefAddr.html) | This class represents the binary form of the address of a communications end-point. |
| [**Binding**](http://docs.google.com/javax/naming/Binding.html) | This class represents a name-to-object binding found in a context. |
| [**CompositeName**](http://docs.google.com/javax/naming/CompositeName.html) | This class represents a composite name -- a sequence of component names spanning multiple namespaces. |
| [**CompoundName**](http://docs.google.com/javax/naming/CompoundName.html) | This class represents a compound name -- a name from a hierarchical name space. |
| [**InitialContext**](http://docs.google.com/javax/naming/InitialContext.html) | This class is the starting context for performing naming operations. |
| [**LinkRef**](http://docs.google.com/javax/naming/LinkRef.html) | This class represents a Reference whose contents is a name, called the link name, that is bound to an atomic name in a context. |
| [**NameClassPair**](http://docs.google.com/javax/naming/NameClassPair.html) | This class represents the object name and class name pair of a binding found in a context. |
| [**RefAddr**](http://docs.google.com/javax/naming/RefAddr.html) | This class represents the address of a communications end-point. |
| [**Reference**](http://docs.google.com/javax/naming/Reference.html) | This class represents a reference to an object that is found outside of the naming/directory system. |
| [**StringRefAddr**](http://docs.google.com/javax/naming/StringRefAddr.html) | This class represents the string form of the address of a communications end-point. |

| **Exception Summary** | |
| --- | --- |
| [**AuthenticationException**](http://docs.google.com/javax/naming/AuthenticationException.html) | This exception is thrown when an authentication error occurs while accessing the naming or directory service. |
| [**AuthenticationNotSupportedException**](http://docs.google.com/javax/naming/AuthenticationNotSupportedException.html) | This exception is thrown when the particular flavor of authentication requested is not supported. |
| [**CannotProceedException**](http://docs.google.com/javax/naming/CannotProceedException.html) | This exception is thrown to indicate that the operation reached a point in the name where the operation cannot proceed any further. |
| [**CommunicationException**](http://docs.google.com/javax/naming/CommunicationException.html) | This exception is thrown when the client is unable to communicate with the directory or naming service. |
| [**ConfigurationException**](http://docs.google.com/javax/naming/ConfigurationException.html) | This exception is thrown when there is a configuration problem. |
| [**ContextNotEmptyException**](http://docs.google.com/javax/naming/ContextNotEmptyException.html) | This exception is thrown when attempting to destroy a context that is not empty. |
| [**InsufficientResourcesException**](http://docs.google.com/javax/naming/InsufficientResourcesException.html) | This exception is thrown when resources are not available to complete the requested operation. |
| [**InterruptedNamingException**](http://docs.google.com/javax/naming/InterruptedNamingException.html) | This exception is thrown when the naming operation being invoked has been interrupted. |
| [**InvalidNameException**](http://docs.google.com/javax/naming/InvalidNameException.html) | This exception indicates that the name being specified does not conform to the naming syntax of a naming system. |
| [**LimitExceededException**](http://docs.google.com/javax/naming/LimitExceededException.html) | This exception is thrown when a method terminates abnormally due to a user or system specified limit. |
| [**LinkException**](http://docs.google.com/javax/naming/LinkException.html) | This exception is used to describe problems encounter while resolving links. |
| [**LinkLoopException**](http://docs.google.com/javax/naming/LinkLoopException.html) | This exception is thrown when a loop was detected will attempting to resolve a link, or an implementation specific limit on link counts has been reached. |
| [**MalformedLinkException**](http://docs.google.com/javax/naming/MalformedLinkException.html) | This exception is thrown when a malformed link was encountered while resolving or constructing a link. |
| [**NameAlreadyBoundException**](http://docs.google.com/javax/naming/NameAlreadyBoundException.html) | This exception is thrown by methods to indicate that a binding cannot be added because the name is already bound to another object. |
| [**NameNotFoundException**](http://docs.google.com/javax/naming/NameNotFoundException.html) | This exception is thrown when a component of the name cannot be resolved because it is not bound. |
| [**NamingException**](http://docs.google.com/javax/naming/NamingException.html) | This is the superclass of all exceptions thrown by operations in the Context and DirContext interfaces. |
| [**NamingSecurityException**](http://docs.google.com/javax/naming/NamingSecurityException.html) | This is the superclass of security-related exceptions thrown by operations in the Context and DirContext interfaces. |
| [**NoInitialContextException**](http://docs.google.com/javax/naming/NoInitialContextException.html) | This exception is thrown when no initial context implementation can be created. |
| [**NoPermissionException**](http://docs.google.com/javax/naming/NoPermissionException.html) | This exception is thrown when attempting to perform an operation for which the client has no permission. |
| [**NotContextException**](http://docs.google.com/javax/naming/NotContextException.html) | This exception is thrown when a naming operation proceeds to a point where a context is required to continue the operation, but the resolved object is not a context. |
| [**OperationNotSupportedException**](http://docs.google.com/javax/naming/OperationNotSupportedException.html) | This exception is thrown when a context implementation does not support the operation being invoked. |
| [**PartialResultException**](http://docs.google.com/javax/naming/PartialResultException.html) | This exception is thrown to indicate that the result being returned or returned so far is partial, and that the operation cannot be completed. |
| [**ReferralException**](http://docs.google.com/javax/naming/ReferralException.html) | This abstract class is used to represent a referral exception, which is generated in response to a *referral* such as that returned by LDAP v3 servers. |
| [**ServiceUnavailableException**](http://docs.google.com/javax/naming/ServiceUnavailableException.html) | This exception is thrown when attempting to communicate with a directory or naming service and that service is not available. |
| [**SizeLimitExceededException**](http://docs.google.com/javax/naming/SizeLimitExceededException.html) | This exception is thrown when a method produces a result that exceeds a size-related limit. |
| [**TimeLimitExceededException**](http://docs.google.com/javax/naming/TimeLimitExceededException.html) | This exception is thrown when a method does not terminate within the specified time limit. |

## Package javax.naming Description

Provides the classes and interfaces for accessing naming services.

This package defines the naming operations of the Java Naming and Directory InterfaceTM (JNDI).   JNDI provides naming and directory functionality to applications written in the Java programming language. It is designed to be independent of any specific naming or directory service implementation. Thus a variety of services--new, emerging, and already deployed ones--can be accessed in a common way.

#### Context

This package defines the notion of a *context*, represented by the Context interface. A context consists of a set of name-to-object *bindings*. Context is the core interface for looking up, binding, unbinding, and renaming objects, and for creating and destroying subcontexts.

lookup() is the most commonly used operation. You supply lookup() the name of the object you want to look up, and it returns the object bound to that name. For example, the following code fragment looks up a printer and sends a document to the printer object to be printed:

Printer printer = (Printer)ctx.lookup("treekiller");  
printer.print(report);

#### Names

Every naming method in the Context interface has two overloads: one that accepts a Name argument and one that accepts a string name. Name is an interface that represents a generic name--an ordered sequence of zero of more components. For these methods, Name can be used to represent a *composite name* (CompositeName) so that you can name an object using a name which spans multiple namespaces.

The overloads that accept Name are useful for applications that need to manipulate names: composing them, comparing components, and so on. The overloads that accept string names are likely to be more useful for simple applications, such as those that simply read in a name and look up the corresponding object.

#### Bindings

The Binding class represents a name-to-object binding. It is a tuple containing the name of the bound object, the name of the object's class, and the object itself.

The Binding class is actually a subclass of NameClassPair, which consists simply of the object's name and the object's class name. The NameClassPair is useful when you only want information about the object's class and do not want to pay the extra cost of getting the object.

#### References

Objects are stored in naming and directory services in different ways. If an object store supports storing Java objects, it might support storing an object in its serialized form. However, some naming and directory services do not support the storing of Java objects. Furthermore, for some objects in the directory, Java programs are but one group of applications that access them. In this case, a serialized Java object might not be the most appropriate representation. JNDI defines a *reference*, represented by the Reference class, which contains information on how to construct a copy of the object. JNDI will attempt to turn references looked up from the directory into the Java objects they represent, so that JNDI clients have the illusion that what is stored in the directory are Java objects.

#### The Initial Context

In JNDI, all naming and directory operations are performed relative to a context. There are no absolute roots. Therefore JNDI defines an *initial context*, InitialContext, which provides a starting point for naming and directory operations. Once you have an initial context, you can use it to look up other contexts and objects.

#### Exceptions

JNDI defines a class hierarchy for exceptions that can be thrown in the course of performing naming and directory operations. The root of this class hierarchy is NamingException. Programs interested in dealing with a particular exception can catch the corresponding subclass of the exception. Otherwise, programs should catch NamingException.

## Package Specification

The JNDI API Specification and related documents can be found in the [JNDI documentation](http://docs.google.com/technotes/guides/jndi/index.html).

**Since:** 1.3

| | [**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/timer/package-summary.html)   [**NEXT PACKAGE**](http://docs.google.com/javax/naming/directory/package-summary.html) | [**FRAMES**](http://docs.google.com/index.html?javax/naming/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).