| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/DirContext.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 CLASS**](http://docs.google.com/javax/naming/directory/BasicAttributes.html)   [**NEXT CLASS**](http://docs.google.com/javax/naming/directory/InitialDirContext.html) | [**FRAMES**](http://docs.google.com/index.html?javax/naming/directory/DirContext.html)    [**NO FRAMES**](http://docs.google.com/DirContext.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | CONSTR | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | CONSTR | [METHOD](#3rdcrjn) |

## **javax.naming.directory**

Interface DirContext

**All Superinterfaces:** [Context](http://docs.google.com/javax/naming/Context.html) **All Known Subinterfaces:** [EventDirContext](http://docs.google.com/javax/naming/event/EventDirContext.html), [LdapContext](http://docs.google.com/javax/naming/ldap/LdapContext.html) **All Known Implementing Classes:** [InitialDirContext](http://docs.google.com/javax/naming/directory/InitialDirContext.html), [InitialLdapContext](http://docs.google.com/javax/naming/ldap/InitialLdapContext.html)

public interface **DirContext**extends [Context](http://docs.google.com/javax/naming/Context.html)

The directory service interface, containing methods for examining and updating attributes associated with objects, and for searching the directory.

#### Names

Each name passed as an argument to a DirContext method is relative to that context. The empty name is used to name the context itself. The name parameter may never be null.

Most of the methods have overloaded versions with one taking a Name parameter and one taking a String. These overloaded versions are equivalent in that if the Name and String parameters are just different representations of the same name, then the overloaded versions of the same methods behave the same. In the method descriptions below, only one version is documented. The second version instead has a link to the first: the same documentation applies to both.

See Context for a discussion on the interpretation of the name argument to the Context methods. These same rules apply to the name argument to the DirContext methods.

#### Attribute Models

There are two basic models of what attributes should be associated with. First, attributes may be directly associated with a DirContext object. In this model, an attribute operation on the named object is roughly equivalent to a lookup on the name (which returns the DirContext object), followed by the attribute operation invoked on the DirContext object in which the caller supplies an empty name. The attributes can be viewed as being stored along with the object (note that this does not imply that the implementation must do so).

The second model is that attributes are associated with a name (typically an atomic name) in a DirContext. In this model, an attribute operation on the named object is roughly equivalent to a lookup on the name of the parent DirContext of the named object, followed by the attribute operation invoked on the parent in which the caller supplies the terminal atomic name. The attributes can be viewed as being stored in the parent DirContext (again, this does not imply that the implementation must do so). Objects that are not DirContexts can have attributes, as long as their parents are DirContexts.

JNDI support both of these models. It is up to the individual service providers to decide where to "store" attributes. JNDI clients are safest when they do not make assumptions about whether an object's attributes are stored as part of the object, or stored within the parent object and associated with the object's name.

#### Attribute Type Names

In the getAttributes() and search() methods, you can supply the attributes to return by supplying a list of attribute names (strings). The attributes that you get back might not have the same names as the attribute names you have specified. This is because some directories support features that cause them to return other attributes. Such features include attribute subclassing, attribute name synonyms, and attribute language codes.

In attribute subclassing, attributes are defined in a class hierarchy. In some directories, for example, the "name" attribute might be the superclass of all name-related attributes, including "commonName" and "surName". Asking for the "name" attribute might return both the "commonName" and "surName" attributes.

With attribute type synonyms, a directory can assign multiple names to the same attribute. For example, "cn" and "commonName" might both refer to the same attribute. Asking for "cn" might return the "commonName" attribute.

Some directories support the language codes for attributes. Asking such a directory for the "description" attribute, for example, might return all of the following attributes:

* description
* description;lang-en
* description;lang-de
* description;lang-fr

#### Operational Attributes

Some directories have the notion of "operational attributes" which are attributes associated with a directory object for administrative purposes. An example of operational attributes is the access control list for an object.

In the getAttributes() and search() methods, you can specify that all attributes associated with the requested objects be returned by supply null as the list of attributes to return. The attributes returned do *not* include operational attributes. In order to retrieve operational attributes, you must name them explicitly.

#### Named Context

There are certain methods in which the name must resolve to a context (for example, when searching a single level context). The documentation of such methods use the term *named context* to describe their name parameter. For these methods, if the named object is not a DirContext, NotContextException is thrown. Aside from these methods, there is no requirement that the *named object* be a DirContext.

#### Parameters

An Attributes, SearchControls, or array object passed as a parameter to any method will not be modified by the service provider. The service provider may keep a reference to it for the duration of the operation, including any enumeration of the method's results and the processing of any referrals generated. The caller should not modify the object during this time. An Attributes object returned by any method is owned by the caller. The caller may subsequently modify it; the service provider will not.

#### Exceptions

All the methods in this interface can throw a NamingException or any of its subclasses. See NamingException and their subclasses for details on each exception.

**Since:** 1.3 **See Also:**[Context](http://docs.google.com/javax/naming/Context.html)

| **Field Summary** | |
| --- | --- |
| static int | [**ADD\_ATTRIBUTE**](http://docs.google.com/javax/naming/directory/DirContext.html#ADD_ATTRIBUTE)            This constant specifies to add an attribute with the specified values. |
| static int | [**REMOVE\_ATTRIBUTE**](http://docs.google.com/javax/naming/directory/DirContext.html#REMOVE_ATTRIBUTE)            This constant specifies to delete the specified attribute values from the attribute. |
| static int | [**REPLACE\_ATTRIBUTE**](http://docs.google.com/javax/naming/directory/DirContext.html#REPLACE_ATTRIBUTE)            This constant specifies to replace an attribute with specified values. |

| **Fields inherited from interface javax.naming.**[**Context**](http://docs.google.com/javax/naming/Context.html) |
| --- |
| [APPLET](http://docs.google.com/javax/naming/Context.html#APPLET), [AUTHORITATIVE](http://docs.google.com/javax/naming/Context.html#AUTHORITATIVE), [BATCHSIZE](http://docs.google.com/javax/naming/Context.html#BATCHSIZE), [DNS\_URL](http://docs.google.com/javax/naming/Context.html#DNS_URL), [INITIAL\_CONTEXT\_FACTORY](http://docs.google.com/javax/naming/Context.html#INITIAL_CONTEXT_FACTORY), [LANGUAGE](http://docs.google.com/javax/naming/Context.html#LANGUAGE), [OBJECT\_FACTORIES](http://docs.google.com/javax/naming/Context.html#OBJECT_FACTORIES), [PROVIDER\_URL](http://docs.google.com/javax/naming/Context.html#PROVIDER_URL), [REFERRAL](http://docs.google.com/javax/naming/Context.html#REFERRAL), [SECURITY\_AUTHENTICATION](http://docs.google.com/javax/naming/Context.html#SECURITY_AUTHENTICATION), [SECURITY\_CREDENTIALS](http://docs.google.com/javax/naming/Context.html#SECURITY_CREDENTIALS), [SECURITY\_PRINCIPAL](http://docs.google.com/javax/naming/Context.html#SECURITY_PRINCIPAL), [SECURITY\_PROTOCOL](http://docs.google.com/javax/naming/Context.html#SECURITY_PROTOCOL), [STATE\_FACTORIES](http://docs.google.com/javax/naming/Context.html#STATE_FACTORIES), [URL\_PKG\_PREFIXES](http://docs.google.com/javax/naming/Context.html#URL_PKG_PREFIXES) |

| **Method Summary** | |
| --- | --- |
| void | [**bind**](http://docs.google.com/javax/naming/directory/DirContext.html#bind(javax.naming.Name,%20java.lang.Object,%20javax.naming.directory.Attributes))([Name](http://docs.google.com/javax/naming/Name.html) name, [Object](http://docs.google.com/java/lang/Object.html) obj, [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) attrs)            Binds a name to an object, along with associated attributes. |
| void | [**bind**](http://docs.google.com/javax/naming/directory/DirContext.html#bind(java.lang.String,%20java.lang.Object,%20javax.naming.directory.Attributes))([String](http://docs.google.com/java/lang/String.html) name, [Object](http://docs.google.com/java/lang/Object.html) obj, [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) attrs)            Binds a name to an object, along with associated attributes. |
| [DirContext](http://docs.google.com/javax/naming/directory/DirContext.html) | [**createSubcontext**](http://docs.google.com/javax/naming/directory/DirContext.html#createSubcontext(javax.naming.Name,%20javax.naming.directory.Attributes))([Name](http://docs.google.com/javax/naming/Name.html) name, [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) attrs)            Creates and binds a new context, along with associated attributes. |
| [DirContext](http://docs.google.com/javax/naming/directory/DirContext.html) | [**createSubcontext**](http://docs.google.com/javax/naming/directory/DirContext.html#createSubcontext(java.lang.String,%20javax.naming.directory.Attributes))([String](http://docs.google.com/java/lang/String.html) name, [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) attrs)            Creates and binds a new context, along with associated attributes. |
| [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) | [**getAttributes**](http://docs.google.com/javax/naming/directory/DirContext.html#getAttributes(javax.naming.Name))([Name](http://docs.google.com/javax/naming/Name.html) name)            Retrieves all of the attributes associated with a named object. |
| [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) | [**getAttributes**](http://docs.google.com/javax/naming/directory/DirContext.html#getAttributes(javax.naming.Name,%20java.lang.String%5B%5D))([Name](http://docs.google.com/javax/naming/Name.html) name, [String](http://docs.google.com/java/lang/String.html)[] attrIds)            Retrieves selected attributes associated with a named object. |
| [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) | [**getAttributes**](http://docs.google.com/javax/naming/directory/DirContext.html#getAttributes(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Retrieves all of the attributes associated with a named object. |
| [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) | [**getAttributes**](http://docs.google.com/javax/naming/directory/DirContext.html#getAttributes(java.lang.String,%20java.lang.String%5B%5D))([String](http://docs.google.com/java/lang/String.html) name, [String](http://docs.google.com/java/lang/String.html)[] attrIds)            Retrieves selected attributes associated with a named object. |
| [DirContext](http://docs.google.com/javax/naming/directory/DirContext.html) | [**getSchema**](http://docs.google.com/javax/naming/directory/DirContext.html#getSchema(javax.naming.Name))([Name](http://docs.google.com/javax/naming/Name.html) name)            Retrieves the schema associated with the named object. |
| [DirContext](http://docs.google.com/javax/naming/directory/DirContext.html) | [**getSchema**](http://docs.google.com/javax/naming/directory/DirContext.html#getSchema(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Retrieves the schema associated with the named object. |
| [DirContext](http://docs.google.com/javax/naming/directory/DirContext.html) | [**getSchemaClassDefinition**](http://docs.google.com/javax/naming/directory/DirContext.html#getSchemaClassDefinition(javax.naming.Name))([Name](http://docs.google.com/javax/naming/Name.html) name)            Retrieves a context containing the schema objects of the named object's class definitions. |
| [DirContext](http://docs.google.com/javax/naming/directory/DirContext.html) | [**getSchemaClassDefinition**](http://docs.google.com/javax/naming/directory/DirContext.html#getSchemaClassDefinition(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Retrieves a context containing the schema objects of the named object's class definitions. |
| void | [**modifyAttributes**](http://docs.google.com/javax/naming/directory/DirContext.html#modifyAttributes(javax.naming.Name,%20int,%20javax.naming.directory.Attributes))([Name](http://docs.google.com/javax/naming/Name.html) name, int mod\_op, [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) attrs)            Modifies the attributes associated with a named object. |
| void | [**modifyAttributes**](http://docs.google.com/javax/naming/directory/DirContext.html#modifyAttributes(javax.naming.Name,%20javax.naming.directory.ModificationItem%5B%5D))([Name](http://docs.google.com/javax/naming/Name.html) name, [ModificationItem](http://docs.google.com/javax/naming/directory/ModificationItem.html)[] mods)            Modifies the attributes associated with a named object using an ordered list of modifications. |
| void | [**modifyAttributes**](http://docs.google.com/javax/naming/directory/DirContext.html#modifyAttributes(java.lang.String,%20int,%20javax.naming.directory.Attributes))([String](http://docs.google.com/java/lang/String.html) name, int mod\_op, [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) attrs)            Modifies the attributes associated with a named object. |
| void | [**modifyAttributes**](http://docs.google.com/javax/naming/directory/DirContext.html#modifyAttributes(java.lang.String,%20javax.naming.directory.ModificationItem%5B%5D))([String](http://docs.google.com/java/lang/String.html) name, [ModificationItem](http://docs.google.com/javax/naming/directory/ModificationItem.html)[] mods)            Modifies the attributes associated with a named object using an ordered list of modifications. |
| void | [**rebind**](http://docs.google.com/javax/naming/directory/DirContext.html#rebind(javax.naming.Name,%20java.lang.Object,%20javax.naming.directory.Attributes))([Name](http://docs.google.com/javax/naming/Name.html) name, [Object](http://docs.google.com/java/lang/Object.html) obj, [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) attrs)            Binds a name to an object, along with associated attributes, overwriting any existing binding. |
| void | [**rebind**](http://docs.google.com/javax/naming/directory/DirContext.html#rebind(java.lang.String,%20java.lang.Object,%20javax.naming.directory.Attributes))([String](http://docs.google.com/java/lang/String.html) name, [Object](http://docs.google.com/java/lang/Object.html) obj, [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) attrs)            Binds a name to an object, along with associated attributes, overwriting any existing binding. |
| [NamingEnumeration](http://docs.google.com/javax/naming/NamingEnumeration.html)<[SearchResult](http://docs.google.com/javax/naming/directory/SearchResult.html)> | [**search**](http://docs.google.com/javax/naming/directory/DirContext.html#search(javax.naming.Name,%20javax.naming.directory.Attributes))([Name](http://docs.google.com/javax/naming/Name.html) name, [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) matchingAttributes)            Searches in a single context for objects that contain a specified set of attributes. |
| [NamingEnumeration](http://docs.google.com/javax/naming/NamingEnumeration.html)<[SearchResult](http://docs.google.com/javax/naming/directory/SearchResult.html)> | [**search**](http://docs.google.com/javax/naming/directory/DirContext.html#search(javax.naming.Name,%20javax.naming.directory.Attributes,%20java.lang.String%5B%5D))([Name](http://docs.google.com/javax/naming/Name.html) name, [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) matchingAttributes, [String](http://docs.google.com/java/lang/String.html)[] attributesToReturn)            Searches in a single context for objects that contain a specified set of attributes, and retrieves selected attributes. |
| [NamingEnumeration](http://docs.google.com/javax/naming/NamingEnumeration.html)<[SearchResult](http://docs.google.com/javax/naming/directory/SearchResult.html)> | [**search**](http://docs.google.com/javax/naming/directory/DirContext.html#search(javax.naming.Name,%20java.lang.String,%20java.lang.Object%5B%5D,%20javax.naming.directory.SearchControls))([Name](http://docs.google.com/javax/naming/Name.html) name, [String](http://docs.google.com/java/lang/String.html) filterExpr, [Object](http://docs.google.com/java/lang/Object.html)[] filterArgs, [SearchControls](http://docs.google.com/javax/naming/directory/SearchControls.html) cons)            Searches in the named context or object for entries that satisfy the given search filter. |
| [NamingEnumeration](http://docs.google.com/javax/naming/NamingEnumeration.html)<[SearchResult](http://docs.google.com/javax/naming/directory/SearchResult.html)> | [**search**](http://docs.google.com/javax/naming/directory/DirContext.html#search(javax.naming.Name,%20java.lang.String,%20javax.naming.directory.SearchControls))([Name](http://docs.google.com/javax/naming/Name.html) name, [String](http://docs.google.com/java/lang/String.html) filter, [SearchControls](http://docs.google.com/javax/naming/directory/SearchControls.html) cons)            Searches in the named context or object for entries that satisfy the given search filter. |
| [NamingEnumeration](http://docs.google.com/javax/naming/NamingEnumeration.html)<[SearchResult](http://docs.google.com/javax/naming/directory/SearchResult.html)> | [**search**](http://docs.google.com/javax/naming/directory/DirContext.html#search(java.lang.String,%20javax.naming.directory.Attributes))([String](http://docs.google.com/java/lang/String.html) name, [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) matchingAttributes)            Searches in a single context for objects that contain a specified set of attributes. |
| [NamingEnumeration](http://docs.google.com/javax/naming/NamingEnumeration.html)<[SearchResult](http://docs.google.com/javax/naming/directory/SearchResult.html)> | [**search**](http://docs.google.com/javax/naming/directory/DirContext.html#search(java.lang.String,%20javax.naming.directory.Attributes,%20java.lang.String%5B%5D))([String](http://docs.google.com/java/lang/String.html) name, [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) matchingAttributes, [String](http://docs.google.com/java/lang/String.html)[] attributesToReturn)            Searches in a single context for objects that contain a specified set of attributes, and retrieves selected attributes. |
| [NamingEnumeration](http://docs.google.com/javax/naming/NamingEnumeration.html)<[SearchResult](http://docs.google.com/javax/naming/directory/SearchResult.html)> | [**search**](http://docs.google.com/javax/naming/directory/DirContext.html#search(java.lang.String,%20java.lang.String,%20java.lang.Object%5B%5D,%20javax.naming.directory.SearchControls))([String](http://docs.google.com/java/lang/String.html) name, [String](http://docs.google.com/java/lang/String.html) filterExpr, [Object](http://docs.google.com/java/lang/Object.html)[] filterArgs, [SearchControls](http://docs.google.com/javax/naming/directory/SearchControls.html) cons)            Searches in the named context or object for entries that satisfy the given search filter. |
| [NamingEnumeration](http://docs.google.com/javax/naming/NamingEnumeration.html)<[SearchResult](http://docs.google.com/javax/naming/directory/SearchResult.html)> | [**search**](http://docs.google.com/javax/naming/directory/DirContext.html#search(java.lang.String,%20java.lang.String,%20javax.naming.directory.SearchControls))([String](http://docs.google.com/java/lang/String.html) name, [String](http://docs.google.com/java/lang/String.html) filter, [SearchControls](http://docs.google.com/javax/naming/directory/SearchControls.html) cons)            Searches in the named context or object for entries that satisfy the given search filter. |

| **Methods inherited from interface javax.naming.**[**Context**](http://docs.google.com/javax/naming/Context.html) |
| --- |
| [addToEnvironment](http://docs.google.com/javax/naming/Context.html#addToEnvironment(java.lang.String,%20java.lang.Object)), [bind](http://docs.google.com/javax/naming/Context.html#bind(javax.naming.Name,%20java.lang.Object)), [bind](http://docs.google.com/javax/naming/Context.html#bind(java.lang.String,%20java.lang.Object)), [close](http://docs.google.com/javax/naming/Context.html#close()), [composeName](http://docs.google.com/javax/naming/Context.html#composeName(javax.naming.Name,%20javax.naming.Name)), [composeName](http://docs.google.com/javax/naming/Context.html#composeName(java.lang.String,%20java.lang.String)), [createSubcontext](http://docs.google.com/javax/naming/Context.html#createSubcontext(javax.naming.Name)), [createSubcontext](http://docs.google.com/javax/naming/Context.html#createSubcontext(java.lang.String)), [destroySubcontext](http://docs.google.com/javax/naming/Context.html#destroySubcontext(javax.naming.Name)), [destroySubcontext](http://docs.google.com/javax/naming/Context.html#destroySubcontext(java.lang.String)), [getEnvironment](http://docs.google.com/javax/naming/Context.html#getEnvironment()), [getNameInNamespace](http://docs.google.com/javax/naming/Context.html#getNameInNamespace()), [getNameParser](http://docs.google.com/javax/naming/Context.html#getNameParser(javax.naming.Name)), [getNameParser](http://docs.google.com/javax/naming/Context.html#getNameParser(java.lang.String)), [list](http://docs.google.com/javax/naming/Context.html#list(javax.naming.Name)), [list](http://docs.google.com/javax/naming/Context.html#list(java.lang.String)), [listBindings](http://docs.google.com/javax/naming/Context.html#listBindings(javax.naming.Name)), [listBindings](http://docs.google.com/javax/naming/Context.html#listBindings(java.lang.String)), [lookup](http://docs.google.com/javax/naming/Context.html#lookup(javax.naming.Name)), [lookup](http://docs.google.com/javax/naming/Context.html#lookup(java.lang.String)), [lookupLink](http://docs.google.com/javax/naming/Context.html#lookupLink(javax.naming.Name)), [lookupLink](http://docs.google.com/javax/naming/Context.html#lookupLink(java.lang.String)), [rebind](http://docs.google.com/javax/naming/Context.html#rebind(javax.naming.Name,%20java.lang.Object)), [rebind](http://docs.google.com/javax/naming/Context.html#rebind(java.lang.String,%20java.lang.Object)), [removeFromEnvironment](http://docs.google.com/javax/naming/Context.html#removeFromEnvironment(java.lang.String)), [rename](http://docs.google.com/javax/naming/Context.html#rename(javax.naming.Name,%20javax.naming.Name)), [rename](http://docs.google.com/javax/naming/Context.html#rename(java.lang.String,%20java.lang.String)), [unbind](http://docs.google.com/javax/naming/Context.html#unbind(javax.naming.Name)), [unbind](http://docs.google.com/javax/naming/Context.html#unbind(java.lang.String)) |

| **Field Detail** |
| --- |

### ADD\_ATTRIBUTE

static final int **ADD\_ATTRIBUTE**

This constant specifies to add an attribute with the specified values.

If attribute does not exist, create the attribute. The resulting attribute has a union of the specified value set and the prior value set. Adding an attribute with no value will throw InvalidAttributeValueException if the attribute must have at least one value. For a single-valued attribute where that attribute already exists, throws AttributeInUseException. If attempting to add more than one value to a single-valued attribute, throws InvalidAttributeValueException.

The value of this constant is 1.

**See Also:**[ModificationItem](http://docs.google.com/javax/naming/directory/ModificationItem.html), [modifyAttributes(javax.naming.Name, int, javax.naming.directory.Attributes)](http://docs.google.com/javax/naming/directory/DirContext.html#modifyAttributes(javax.naming.Name,%20int,%20javax.naming.directory.Attributes)), [Constant Field Values](http://docs.google.com/constant-values.html#javax.naming.directory.DirContext.ADD_ATTRIBUTE)

### REPLACE\_ATTRIBUTE

static final int **REPLACE\_ATTRIBUTE**

This constant specifies to replace an attribute with specified values.

If attribute already exists, replaces all existing values with new specified values. If the attribute does not exist, creates it. If no value is specified, deletes all the values of the attribute. Removal of the last value will remove the attribute if the attribute is required to have at least one value. If attempting to add more than one value to a single-valued attribute, throws InvalidAttributeValueException.

The value of this constant is 2.

**See Also:**[ModificationItem](http://docs.google.com/javax/naming/directory/ModificationItem.html), [modifyAttributes(javax.naming.Name, int, javax.naming.directory.Attributes)](http://docs.google.com/javax/naming/directory/DirContext.html#modifyAttributes(javax.naming.Name,%20int,%20javax.naming.directory.Attributes)), [Constant Field Values](http://docs.google.com/constant-values.html#javax.naming.directory.DirContext.REPLACE_ATTRIBUTE)

### REMOVE\_ATTRIBUTE

static final int **REMOVE\_ATTRIBUTE**

This constant specifies to delete the specified attribute values from the attribute.

The resulting attribute has the set difference of its prior value set and the specified value set. If no values are specified, deletes the entire attribute. If the attribute does not exist, or if some or all members of the specified value set do not exist, this absence may be ignored and the operation succeeds, or a NamingException may be thrown to indicate the absence. Removal of the last value will remove the attribute if the attribute is required to have at least one value.

The value of this constant is 3.

**See Also:**[ModificationItem](http://docs.google.com/javax/naming/directory/ModificationItem.html), [modifyAttributes(javax.naming.Name, int, javax.naming.directory.Attributes)](http://docs.google.com/javax/naming/directory/DirContext.html#modifyAttributes(javax.naming.Name,%20int,%20javax.naming.directory.Attributes)), [Constant Field Values](http://docs.google.com/constant-values.html#javax.naming.directory.DirContext.REMOVE_ATTRIBUTE)

| **Method Detail** |
| --- |

### getAttributes

[Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) **getAttributes**([Name](http://docs.google.com/javax/naming/Name.html) name)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Retrieves all of the attributes associated with a named object. See the class description regarding attribute models, attribute type names, and operational attributes.

**Parameters:**name - the name of the object from which to retrieve attributes **Returns:**the set of attributes associated with name. Returns an empty attribute set if name has no attributes; never null. **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered**See Also:**[getAttributes(String)](http://docs.google.com/javax/naming/directory/DirContext.html#getAttributes(java.lang.String)), [getAttributes(Name, String[])](http://docs.google.com/javax/naming/directory/DirContext.html#getAttributes(javax.naming.Name,%20java.lang.String%5B%5D))

### getAttributes

[Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) **getAttributes**([String](http://docs.google.com/java/lang/String.html) name)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Retrieves all of the attributes associated with a named object. See [getAttributes(Name)](http://docs.google.com/javax/naming/directory/DirContext.html#getAttributes(javax.naming.Name)) for details.

**Parameters:**name - the name of the object from which to retrieve attributes **Returns:**the set of attributes associated with name **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### getAttributes

[Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) **getAttributes**([Name](http://docs.google.com/javax/naming/Name.html) name,  
 [String](http://docs.google.com/java/lang/String.html)[] attrIds)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Retrieves selected attributes associated with a named object. See the class description regarding attribute models, attribute type names, and operational attributes.

If the object does not have an attribute specified, the directory will ignore the nonexistent attribute and return those requested attributes that the object does have.

A directory might return more attributes than was requested (see **Attribute Type Names** in the class description), but is not allowed to return arbitrary, unrelated attributes.

See also **Operational Attributes** in the class description.

**Parameters:**name - the name of the object from which to retrieve attributesattrIds - the identifiers of the attributes to retrieve. null indicates that all attributes should be retrieved; an empty array indicates that none should be retrieved. **Returns:**the requested attributes; never null **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### getAttributes

[Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) **getAttributes**([String](http://docs.google.com/java/lang/String.html) name,  
 [String](http://docs.google.com/java/lang/String.html)[] attrIds)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Retrieves selected attributes associated with a named object. See [getAttributes(Name, String[])](http://docs.google.com/javax/naming/directory/DirContext.html#getAttributes(javax.naming.Name,%20java.lang.String%5B%5D)) for details.

**Parameters:**name - The name of the object from which to retrieve attributesattrIds - the identifiers of the attributes to retrieve. null indicates that all attributes should be retrieved; an empty array indicates that none should be retrieved. **Returns:**the requested attributes; never null **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### modifyAttributes

void **modifyAttributes**([Name](http://docs.google.com/javax/naming/Name.html) name,  
 int mod\_op,  
 [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) attrs)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Modifies the attributes associated with a named object. The order of the modifications is not specified. Where possible, the modifications are performed atomically.

**Parameters:**name - the name of the object whose attributes will be updatedmod\_op - the modification operation, one of: ADD\_ATTRIBUTE, REPLACE\_ATTRIBUTE, REMOVE\_ATTRIBUTE.attrs - the attributes to be used for the modification; may not be null **Throws:** [AttributeModificationException](http://docs.google.com/javax/naming/directory/AttributeModificationException.html) - if the modification cannot be completed successfully [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered**See Also:**[modifyAttributes(Name, ModificationItem[])](http://docs.google.com/javax/naming/directory/DirContext.html#modifyAttributes(javax.naming.Name,%20javax.naming.directory.ModificationItem%5B%5D))

### modifyAttributes

void **modifyAttributes**([String](http://docs.google.com/java/lang/String.html) name,  
 int mod\_op,  
 [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) attrs)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Modifies the attributes associated with a named object. See [modifyAttributes(Name, int, Attributes)](http://docs.google.com/javax/naming/directory/DirContext.html#modifyAttributes(javax.naming.Name,%20int,%20javax.naming.directory.Attributes)) for details.

**Parameters:**name - the name of the object whose attributes will be updatedmod\_op - the modification operation, one of: ADD\_ATTRIBUTE, REPLACE\_ATTRIBUTE, REMOVE\_ATTRIBUTE.attrs - the attributes to be used for the modification; may not be null **Throws:** [AttributeModificationException](http://docs.google.com/javax/naming/directory/AttributeModificationException.html) - if the modification cannot be completed successfully [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### modifyAttributes

void **modifyAttributes**([Name](http://docs.google.com/javax/naming/Name.html) name,  
 [ModificationItem](http://docs.google.com/javax/naming/directory/ModificationItem.html)[] mods)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Modifies the attributes associated with a named object using an ordered list of modifications. The modifications are performed in the order specified. Each modification specifies a modification operation code and an attribute on which to operate. Where possible, the modifications are performed atomically.

**Parameters:**name - the name of the object whose attributes will be updatedmods - an ordered sequence of modifications to be performed; may not be null **Throws:** [AttributeModificationException](http://docs.google.com/javax/naming/directory/AttributeModificationException.html) - if the modifications cannot be completed successfully [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered**See Also:**[modifyAttributes(Name, int, Attributes)](http://docs.google.com/javax/naming/directory/DirContext.html#modifyAttributes(javax.naming.Name,%20int,%20javax.naming.directory.Attributes)), [ModificationItem](http://docs.google.com/javax/naming/directory/ModificationItem.html)

### modifyAttributes

void **modifyAttributes**([String](http://docs.google.com/java/lang/String.html) name,  
 [ModificationItem](http://docs.google.com/javax/naming/directory/ModificationItem.html)[] mods)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Modifies the attributes associated with a named object using an ordered list of modifications. See [modifyAttributes(Name, ModificationItem[])](http://docs.google.com/javax/naming/directory/DirContext.html#modifyAttributes(javax.naming.Name,%20javax.naming.directory.ModificationItem%5B%5D)) for details.

**Parameters:**name - the name of the object whose attributes will be updatedmods - an ordered sequence of modifications to be performed; may not be null **Throws:** [AttributeModificationException](http://docs.google.com/javax/naming/directory/AttributeModificationException.html) - if the modifications cannot be completed successfully [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### bind

void **bind**([Name](http://docs.google.com/javax/naming/Name.html) name,  
 [Object](http://docs.google.com/java/lang/Object.html) obj,  
 [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) attrs)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Binds a name to an object, along with associated attributes. If attrs is null, the resulting binding will have the attributes associated with obj if obj is a DirContext, and no attributes otherwise. If attrs is non-null, the resulting binding will have attrs as its attributes; any attributes associated with obj are ignored.

**Parameters:**name - the name to bind; may not be emptyobj - the object to bind; possibly nullattrs - the attributes to associate with the binding **Throws:** [NameAlreadyBoundException](http://docs.google.com/javax/naming/NameAlreadyBoundException.html) - if name is already bound [InvalidAttributesException](http://docs.google.com/javax/naming/directory/InvalidAttributesException.html) - if some "mandatory" attributes of the binding are not supplied [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered**See Also:**[Context.bind(Name, Object)](http://docs.google.com/javax/naming/Context.html#bind(javax.naming.Name,%20java.lang.Object)), [rebind(Name, Object, Attributes)](http://docs.google.com/javax/naming/directory/DirContext.html#rebind(javax.naming.Name,%20java.lang.Object,%20javax.naming.directory.Attributes))

### bind

void **bind**([String](http://docs.google.com/java/lang/String.html) name,  
 [Object](http://docs.google.com/java/lang/Object.html) obj,  
 [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) attrs)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Binds a name to an object, along with associated attributes. See [bind(Name, Object, Attributes)](http://docs.google.com/javax/naming/directory/DirContext.html#bind(javax.naming.Name,%20java.lang.Object,%20javax.naming.directory.Attributes)) for details.

**Parameters:**name - the name to bind; may not be emptyobj - the object to bind; possibly nullattrs - the attributes to associate with the binding **Throws:** [NameAlreadyBoundException](http://docs.google.com/javax/naming/NameAlreadyBoundException.html) - if name is already bound [InvalidAttributesException](http://docs.google.com/javax/naming/directory/InvalidAttributesException.html) - if some "mandatory" attributes of the binding are not supplied [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### rebind

void **rebind**([Name](http://docs.google.com/javax/naming/Name.html) name,  
 [Object](http://docs.google.com/java/lang/Object.html) obj,  
 [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) attrs)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Binds a name to an object, along with associated attributes, overwriting any existing binding. If attrs is null and obj is a DirContext, the attributes from obj are used. If attrs is null and obj is not a DirContext, any existing attributes associated with the object already bound in the directory remain unchanged. If attrs is non-null, any existing attributes associated with the object already bound in the directory are removed and attrs is associated with the named object. If obj is a DirContext and attrs is non-null, the attributes of obj are ignored.

**Parameters:**name - the name to bind; may not be emptyobj - the object to bind; possibly nullattrs - the attributes to associate with the binding **Throws:** [InvalidAttributesException](http://docs.google.com/javax/naming/directory/InvalidAttributesException.html) - if some "mandatory" attributes of the binding are not supplied [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered**See Also:**[Context.bind(Name, Object)](http://docs.google.com/javax/naming/Context.html#bind(javax.naming.Name,%20java.lang.Object)), [bind(Name, Object, Attributes)](http://docs.google.com/javax/naming/directory/DirContext.html#bind(javax.naming.Name,%20java.lang.Object,%20javax.naming.directory.Attributes))

### rebind

void **rebind**([String](http://docs.google.com/java/lang/String.html) name,  
 [Object](http://docs.google.com/java/lang/Object.html) obj,  
 [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) attrs)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Binds a name to an object, along with associated attributes, overwriting any existing binding. See [rebind(Name, Object, Attributes)](http://docs.google.com/javax/naming/directory/DirContext.html#rebind(javax.naming.Name,%20java.lang.Object,%20javax.naming.directory.Attributes)) for details.

**Parameters:**name - the name to bind; may not be emptyobj - the object to bind; possibly nullattrs - the attributes to associate with the binding **Throws:** [InvalidAttributesException](http://docs.google.com/javax/naming/directory/InvalidAttributesException.html) - if some "mandatory" attributes of the binding are not supplied [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### createSubcontext

[DirContext](http://docs.google.com/javax/naming/directory/DirContext.html) **createSubcontext**([Name](http://docs.google.com/javax/naming/Name.html) name,  
 [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) attrs)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Creates and binds a new context, along with associated attributes. This method creates a new subcontext with the given name, binds it in the target context (that named by all but terminal atomic component of the name), and associates the supplied attributes with the newly created object. All intermediate and target contexts must already exist. If attrs is null, this method is equivalent to Context.createSubcontext().

**Parameters:**name - the name of the context to create; may not be emptyattrs - the attributes to associate with the newly created context **Returns:**the newly created context **Throws:** [NameAlreadyBoundException](http://docs.google.com/javax/naming/NameAlreadyBoundException.html) - if the name is already bound [InvalidAttributesException](http://docs.google.com/javax/naming/directory/InvalidAttributesException.html) - if attrs does not contain all the mandatory attributes required for creation [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered**See Also:**[Context.createSubcontext(Name)](http://docs.google.com/javax/naming/Context.html#createSubcontext(javax.naming.Name))

### createSubcontext

[DirContext](http://docs.google.com/javax/naming/directory/DirContext.html) **createSubcontext**([String](http://docs.google.com/java/lang/String.html) name,  
 [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) attrs)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Creates and binds a new context, along with associated attributes. See [createSubcontext(Name, Attributes)](http://docs.google.com/javax/naming/directory/DirContext.html#createSubcontext(javax.naming.Name,%20javax.naming.directory.Attributes)) for details.

**Parameters:**name - the name of the context to create; may not be emptyattrs - the attributes to associate with the newly created context **Returns:**the newly created context **Throws:** [NameAlreadyBoundException](http://docs.google.com/javax/naming/NameAlreadyBoundException.html) - if the name is already bound [InvalidAttributesException](http://docs.google.com/javax/naming/directory/InvalidAttributesException.html) - if attrs does not contain all the mandatory attributes required for creation [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### getSchema

[DirContext](http://docs.google.com/javax/naming/directory/DirContext.html) **getSchema**([Name](http://docs.google.com/javax/naming/Name.html) name)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Retrieves the schema associated with the named object. The schema describes rules regarding the structure of the namespace and the attributes stored within it. The schema specifies what types of objects can be added to the directory and where they can be added; what mandatory and optional attributes an object can have. The range of support for schemas is directory-specific.

This method returns the root of the schema information tree that is applicable to the named object. Several named objects (or even an entire directory) might share the same schema.

Issues such as structure and contents of the schema tree, permission to modify to the contents of the schema tree, and the effect of such modifications on the directory are dependent on the underlying directory.

**Parameters:**name - the name of the object whose schema is to be retrieved **Returns:**the schema associated with the context; never null **Throws:** [OperationNotSupportedException](http://docs.google.com/javax/naming/OperationNotSupportedException.html) - if schema not supported [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### getSchema

[DirContext](http://docs.google.com/javax/naming/directory/DirContext.html) **getSchema**([String](http://docs.google.com/java/lang/String.html) name)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Retrieves the schema associated with the named object. See [getSchema(Name)](http://docs.google.com/javax/naming/directory/DirContext.html#getSchema(javax.naming.Name)) for details.

**Parameters:**name - the name of the object whose schema is to be retrieved **Returns:**the schema associated with the context; never null **Throws:** [OperationNotSupportedException](http://docs.google.com/javax/naming/OperationNotSupportedException.html) - if schema not supported [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### getSchemaClassDefinition

[DirContext](http://docs.google.com/javax/naming/directory/DirContext.html) **getSchemaClassDefinition**([Name](http://docs.google.com/javax/naming/Name.html) name)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Retrieves a context containing the schema objects of the named object's class definitions.

One category of information found in directory schemas is *class definitions*. An "object class" definition specifies the object's *type* and what attributes (mandatory and optional) the object must/can have. Note that the term "object class" being referred to here is in the directory sense rather than in the Java sense. For example, if the named object is a directory object of "Person" class, getSchemaClassDefinition() would return a DirContext representing the (directory's) object class definition of "Person".

The information that can be retrieved from an object class definition is directory-dependent.

Prior to JNDI 1.2, this method returned a single schema object representing the class definition of the named object. Since JNDI 1.2, this method returns a DirContext containing all of the named object's class definitions.

**Parameters:**name - the name of the object whose object class definition is to be retrieved **Returns:**the DirContext containing the named object's class definitions; never null **Throws:** [OperationNotSupportedException](http://docs.google.com/javax/naming/OperationNotSupportedException.html) - if schema not supported [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### getSchemaClassDefinition

[DirContext](http://docs.google.com/javax/naming/directory/DirContext.html) **getSchemaClassDefinition**([String](http://docs.google.com/java/lang/String.html) name)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Retrieves a context containing the schema objects of the named object's class definitions. See [getSchemaClassDefinition(Name)](http://docs.google.com/javax/naming/directory/DirContext.html#getSchemaClassDefinition(javax.naming.Name)) for details.

**Parameters:**name - the name of the object whose object class definition is to be retrieved **Returns:**the DirContext containing the named object's class definitions; never null **Throws:** [OperationNotSupportedException](http://docs.google.com/javax/naming/OperationNotSupportedException.html) - if schema not supported [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### search

[NamingEnumeration](http://docs.google.com/javax/naming/NamingEnumeration.html)<[SearchResult](http://docs.google.com/javax/naming/directory/SearchResult.html)> **search**([Name](http://docs.google.com/javax/naming/Name.html) name,  
 [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) matchingAttributes,  
 [String](http://docs.google.com/java/lang/String.html)[] attributesToReturn)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Searches in a single context for objects that contain a specified set of attributes, and retrieves selected attributes. The search is performed using the default SearchControls settings.

For an object to be selected, each attribute in matchingAttributes must match some attribute of the object. If matchingAttributes is empty or null, all objects in the target context are returned.

An attribute *A*1 in matchingAttributes is considered to match an attribute *A*2 of an object if *A*1 and *A*2 have the same identifier, and each value of *A*1 is equal to some value of *A*2. This implies that the order of values is not significant, and that *A*2 may contain "extra" values not found in *A*1 without affecting the comparison. It also implies that if *A*1 has no values, then testing for a match is equivalent to testing for the presence of an attribute *A*2 with the same identifier.

The precise definition of "equality" used in comparing attribute values is defined by the underlying directory service. It might use the Object.equals method, for example, or might use a schema to specify a different equality operation. For matching based on operations other than equality (such as substring comparison) use the version of the search method that takes a filter argument.

When changes are made to this DirContext, the effect on enumerations returned by prior calls to this method is undefined.

If the object does not have the attribute specified, the directory will ignore the nonexistent attribute and return the requested attributes that the object does have.

A directory might return more attributes than was requested (see **Attribute Type Names** in the class description), but is not allowed to return arbitrary, unrelated attributes.

See also **Operational Attributes** in the class description.

**Parameters:**name - the name of the context to searchmatchingAttributes - the attributes to search for. If empty or null, all objects in the target context are returned.attributesToReturn - the attributes to return. null indicates that all attributes are to be returned; an empty array indicates that none are to be returned. **Returns:**a non-null enumeration of SearchResult objects. Each SearchResult contains the attributes identified by attributesToReturn and the name of the corresponding object, named relative to the context named by name. **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered**See Also:**[SearchControls](http://docs.google.com/javax/naming/directory/SearchControls.html), [SearchResult](http://docs.google.com/javax/naming/directory/SearchResult.html), [search(Name, String, Object[], SearchControls)](http://docs.google.com/javax/naming/directory/DirContext.html#search(javax.naming.Name,%20java.lang.String,%20java.lang.Object%5B%5D,%20javax.naming.directory.SearchControls))

### search

[NamingEnumeration](http://docs.google.com/javax/naming/NamingEnumeration.html)<[SearchResult](http://docs.google.com/javax/naming/directory/SearchResult.html)> **search**([String](http://docs.google.com/java/lang/String.html) name,  
 [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) matchingAttributes,  
 [String](http://docs.google.com/java/lang/String.html)[] attributesToReturn)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Searches in a single context for objects that contain a specified set of attributes, and retrieves selected attributes. See [search(Name, Attributes, String[])](http://docs.google.com/javax/naming/directory/DirContext.html#search(javax.naming.Name,%20javax.naming.directory.Attributes,%20java.lang.String%5B%5D)) for details.

**Parameters:**name - the name of the context to searchmatchingAttributes - the attributes to search forattributesToReturn - the attributes to return **Returns:**a non-null enumeration of SearchResult objects **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### search

[NamingEnumeration](http://docs.google.com/javax/naming/NamingEnumeration.html)<[SearchResult](http://docs.google.com/javax/naming/directory/SearchResult.html)> **search**([Name](http://docs.google.com/javax/naming/Name.html) name,  
 [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) matchingAttributes)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Searches in a single context for objects that contain a specified set of attributes. This method returns all the attributes of such objects. It is equivalent to supplying null as the atributesToReturn parameter to the method search(Name, Attributes, String[]).

See [search(Name, Attributes, String[])](http://docs.google.com/javax/naming/directory/DirContext.html#search(javax.naming.Name,%20javax.naming.directory.Attributes,%20java.lang.String%5B%5D)) for a full description.

**Parameters:**name - the name of the context to searchmatchingAttributes - the attributes to search for **Returns:**an enumeration of SearchResult objects **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered**See Also:**[search(Name, Attributes, String[])](http://docs.google.com/javax/naming/directory/DirContext.html#search(javax.naming.Name,%20javax.naming.directory.Attributes,%20java.lang.String%5B%5D))

### search

[NamingEnumeration](http://docs.google.com/javax/naming/NamingEnumeration.html)<[SearchResult](http://docs.google.com/javax/naming/directory/SearchResult.html)> **search**([String](http://docs.google.com/java/lang/String.html) name,  
 [Attributes](http://docs.google.com/javax/naming/directory/Attributes.html) matchingAttributes)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Searches in a single context for objects that contain a specified set of attributes. See [search(Name, Attributes)](http://docs.google.com/javax/naming/directory/DirContext.html#search(javax.naming.Name,%20javax.naming.directory.Attributes)) for details.

**Parameters:**name - the name of the context to searchmatchingAttributes - the attributes to search for **Returns:**an enumeration of SearchResult objects **Throws:** [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### search

[NamingEnumeration](http://docs.google.com/javax/naming/NamingEnumeration.html)<[SearchResult](http://docs.google.com/javax/naming/directory/SearchResult.html)> **search**([Name](http://docs.google.com/javax/naming/Name.html) name,  
 [String](http://docs.google.com/java/lang/String.html) filter,  
 [SearchControls](http://docs.google.com/javax/naming/directory/SearchControls.html) cons)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Searches in the named context or object for entries that satisfy the given search filter. Performs the search as specified by the search controls.

The format and interpretation of filter follows RFC 2254 with the following interpretations for attr and value mentioned in the RFC.

attr is the attribute's identifier.

value is the string representation the attribute's value. The translation of this string representation into the attribute's value is directory-specific.

For the assertion "someCount=127", for example, attr is "someCount" and value is "127". The provider determines, based on the attribute ID ("someCount") (and possibly its schema), that the attribute's value is an integer. It then parses the string "127" appropriately.

Any non-ASCII characters in the filter string should be represented by the appropriate Java (Unicode) characters, and not encoded as UTF-8 octets. Alternately, the "backslash-hexcode" notation described in RFC 2254 may be used.

If the directory does not support a string representation of some or all of its attributes, the form of search that accepts filter arguments in the form of Objects can be used instead. The service provider for such a directory would then translate the filter arguments to its service-specific representation for filter evaluation. See search(Name, String, Object[], SearchControls).

RFC 2254 defines certain operators for the filter, including substring matches, equality, approximate match, greater than, less than. These operators are mapped to operators with corresponding semantics in the underlying directory. For example, for the equals operator, suppose the directory has a matching rule defining "equality" of the attributes in the filter. This rule would be used for checking equality of the attributes specified in the filter with the attributes of objects in the directory. Similarly, if the directory has a matching rule for ordering, this rule would be used for making "greater than" and "less than" comparisons.

Not all of the operators defined in RFC 2254 are applicable to all attributes. When an operator is not applicable, the exception InvalidSearchFilterException is thrown.

The result is returned in an enumeration of SearchResults. Each SearchResult contains the name of the object and other information about the object (see SearchResult). The name is either relative to the target context of the search (which is named by the name parameter), or it is a URL string. If the target context is included in the enumeration (as is possible when cons specifies a search scope of SearchControls.OBJECT\_SCOPE or SearchControls.SUBSTREE\_SCOPE), its name is the empty string. The SearchResult may also contain attributes of the matching object if the cons argument specified that attributes be returned.

If the object does not have a requested attribute, that nonexistent attribute will be ignored. Those requested attributes that the object does have will be returned.

A directory might return more attributes than were requested (see **Attribute Type Names** in the class description) but is not allowed to return arbitrary, unrelated attributes.

See also **Operational Attributes** in the class description.

**Parameters:**name - the name of the context or object to searchfilter - the filter expression to use for the search; may not be nullcons - the search controls that control the search. If null, the default search controls are used (equivalent to (new SearchControls())). **Returns:**an enumeration of SearchResults of the objects that satisfy the filter; never null **Throws:** [InvalidSearchFilterException](http://docs.google.com/javax/naming/directory/InvalidSearchFilterException.html) - if the search filter specified is not supported or understood by the underlying directory [InvalidSearchControlsException](http://docs.google.com/javax/naming/directory/InvalidSearchControlsException.html) - if the search controls contain invalid settings [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered**See Also:**[search(Name, String, Object[], SearchControls)](http://docs.google.com/javax/naming/directory/DirContext.html#search(javax.naming.Name,%20java.lang.String,%20java.lang.Object%5B%5D,%20javax.naming.directory.SearchControls)), [SearchControls](http://docs.google.com/javax/naming/directory/SearchControls.html), [SearchResult](http://docs.google.com/javax/naming/directory/SearchResult.html)

### search

[NamingEnumeration](http://docs.google.com/javax/naming/NamingEnumeration.html)<[SearchResult](http://docs.google.com/javax/naming/directory/SearchResult.html)> **search**([String](http://docs.google.com/java/lang/String.html) name,  
 [String](http://docs.google.com/java/lang/String.html) filter,  
 [SearchControls](http://docs.google.com/javax/naming/directory/SearchControls.html) cons)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Searches in the named context or object for entries that satisfy the given search filter. Performs the search as specified by the search controls. See [search(Name, String, SearchControls)](http://docs.google.com/javax/naming/directory/DirContext.html#search(javax.naming.Name,%20java.lang.String,%20javax.naming.directory.SearchControls)) for details.

**Parameters:**name - the name of the context or object to searchfilter - the filter expression to use for the search; may not be nullcons - the search controls that control the search. If null, the default search controls are used (equivalent to (new SearchControls())). **Returns:**an enumeration of SearchResults for the objects that satisfy the filter. **Throws:** [InvalidSearchFilterException](http://docs.google.com/javax/naming/directory/InvalidSearchFilterException.html) - if the search filter specified is not supported or understood by the underlying directory [InvalidSearchControlsException](http://docs.google.com/javax/naming/directory/InvalidSearchControlsException.html) - if the search controls contain invalid settings [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

### search

[NamingEnumeration](http://docs.google.com/javax/naming/NamingEnumeration.html)<[SearchResult](http://docs.google.com/javax/naming/directory/SearchResult.html)> **search**([Name](http://docs.google.com/javax/naming/Name.html) name,  
 [String](http://docs.google.com/java/lang/String.html) filterExpr,  
 [Object](http://docs.google.com/java/lang/Object.html)[] filterArgs,  
 [SearchControls](http://docs.google.com/javax/naming/directory/SearchControls.html) cons)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Searches in the named context or object for entries that satisfy the given search filter. Performs the search as specified by the search controls.

The interpretation of filterExpr is based on RFC 2254. It may additionally contain variables of the form {i} -- where i is an integer -- that refer to objects in the filterArgs array. The interpretation of filterExpr is otherwise identical to that of the filter parameter of the method search(Name, String, SearchControls).

When a variable {i} appears in a search filter, it indicates that the filter argument filterArgs[i] is to be used in that place. Such variables may be used wherever an *attr*, *value*, or *matchingrule* production appears in the filter grammar of RFC 2254, section 4. When a string-valued filter argument is substituted for a variable, the filter is interpreted as if the string were given in place of the variable, with any characters having special significance within filters (such as '\*') having been escaped according to the rules of RFC 2254.

For directories that do not use a string representation for some or all of their attributes, the filter argument corresponding to an attribute value may be of a type other than String. Directories that support unstructured binary-valued attributes, for example, should accept byte arrays as filter arguments. The interpretation (if any) of filter arguments of any other type is determined by the service provider for that directory, which maps the filter operations onto operations with corresponding semantics in the underlying directory.

This method returns an enumeration of the results. Each element in the enumeration contains the name of the object and other information about the object (see SearchResult). The name is either relative to the target context of the search (which is named by the name parameter), or it is a URL string. If the target context is included in the enumeration (as is possible when cons specifies a search scope of SearchControls.OBJECT\_SCOPE or SearchControls.SUBSTREE\_SCOPE), its name is the empty string.

The SearchResult may also contain attributes of the matching object if the cons argument specifies that attributes be returned.

If the object does not have a requested attribute, that nonexistent attribute will be ignored. Those requested attributes that the object does have will be returned.

A directory might return more attributes than were requested (see **Attribute Type Names** in the class description) but is not allowed to return arbitrary, unrelated attributes.

If a search filter with invalid variable substitutions is provided to this method, the result is undefined. When changes are made to this DirContext, the effect on enumerations returned by prior calls to this method is undefined.

See also **Operational Attributes** in the class description.

**Parameters:**name - the name of the context or object to searchfilterExpr - the filter expression to use for the search. The expression may contain variables of the form "{i}" where i is a nonnegative integer. May not be null.filterArgs - the array of arguments to substitute for the variables in filterExpr. The value of filterArgs[i] will replace each occurrence of "{i}". If null, equivalent to an empty array.cons - the search controls that control the search. If null, the default search controls are used (equivalent to (new SearchControls())). **Returns:**an enumeration of SearchResults of the objects that satisfy the filter; never null **Throws:** [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if filterExpr contains {i} expressions where i is outside the bounds of the array filterArgs [InvalidSearchControlsException](http://docs.google.com/javax/naming/directory/InvalidSearchControlsException.html) - if cons contains invalid settings [InvalidSearchFilterException](http://docs.google.com/javax/naming/directory/InvalidSearchFilterException.html) - if filterExpr with filterArgs represents an invalid search filter [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered**See Also:**[search(Name, Attributes, String[])](http://docs.google.com/javax/naming/directory/DirContext.html#search(javax.naming.Name,%20javax.naming.directory.Attributes,%20java.lang.String%5B%5D)), [MessageFormat](http://docs.google.com/java/text/MessageFormat.html)

### search

[NamingEnumeration](http://docs.google.com/javax/naming/NamingEnumeration.html)<[SearchResult](http://docs.google.com/javax/naming/directory/SearchResult.html)> **search**([String](http://docs.google.com/java/lang/String.html) name,  
 [String](http://docs.google.com/java/lang/String.html) filterExpr,  
 [Object](http://docs.google.com/java/lang/Object.html)[] filterArgs,  
 [SearchControls](http://docs.google.com/javax/naming/directory/SearchControls.html) cons)  
 throws [NamingException](http://docs.google.com/javax/naming/NamingException.html)

Searches in the named context or object for entries that satisfy the given search filter. Performs the search as specified by the search controls. See [search(Name, String, Object[], SearchControls)](http://docs.google.com/javax/naming/directory/DirContext.html#search(javax.naming.Name,%20java.lang.String,%20java.lang.Object%5B%5D,%20javax.naming.directory.SearchControls)) for details.

**Parameters:**name - the name of the context or object to searchfilterExpr - the filter expression to use for the search. The expression may contain variables of the form "{i}" where i is a nonnegative integer. May not be null.filterArgs - the array of arguments to substitute for the variables in filterExpr. The value of filterArgs[i] will replace each occurrence of "{i}". If null, equivalent to an empty array.cons - the search controls that control the search. If null, the default search controls are used (equivalent to (new SearchControls())). **Returns:**an enumeration of SearchResults of the objects that satisfy the filter; never null **Throws:** [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if filterExpr contains {i} expressions where i is outside the bounds of the array filterArgs [InvalidSearchControlsException](http://docs.google.com/javax/naming/directory/InvalidSearchControlsException.html) - if cons contains invalid settings [InvalidSearchFilterException](http://docs.google.com/javax/naming/directory/InvalidSearchFilterException.html) - if filterExpr with filterArgs represents an invalid search filter [NamingException](http://docs.google.com/javax/naming/NamingException.html) - if a naming exception is encountered

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/DirContext.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 CLASS**](http://docs.google.com/javax/naming/directory/BasicAttributes.html)   [**NEXT CLASS**](http://docs.google.com/javax/naming/directory/InitialDirContext.html) | [**FRAMES**](http://docs.google.com/index.html?javax/naming/directory/DirContext.html)    [**NO FRAMES**](http://docs.google.com/DirContext.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | CONSTR | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | CONSTR | [METHOD](#3rdcrjn) |

[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).