Java Naming and Directory Interface

The **Java Naming and Directory Interface** (**JNDI**) is a Java API for a directory service that allows Java software clients to discover and look up data and objects via a name. Like all Java APIs that interface with host systems, JNDI is independent of the underlying implementation. Additionally, it specifies a service provider interface (SPI) that allows directory service implementations to be plugged into the framework. It may make use of a server, a flat file, or a database; the choice is up to the vendor.

Background

The JNDI API is used by the Java RMI and Java EE APIs to look up objects in a network. Jini has its own lookup service and does not use the JNDI API.

The API provides:

- a mechanism to bind an object to a name
- a directory lookup interface that allows general queries
- an event interface that allows clients to determine when directory entries have been modified
- LDAP extensions to support the additional capabilities of an LDAP service

The SPI portion allows support for practically any kind of naming or directory service including:

- LDAP
- DNS
- NIS
- CORBA name service
- File system

The JNDI specification was first released by Sun Microsystems on March 10, 1997.[1] As of 2006, the current version is JNDI 1.2.

Basic lookup

JNDI (Java Naming and Directory Interface) organizes its names into a hierarchy. A name can be any string such as "com.mydomain.ejb.MyBean". A name can also be an object that supports the Name interface, however a string is the most common way to name an object. A name is bound to an object in the directory by storing either the object or a reference to the object in the directory service identified by the name.

The JNDI API defines a context that specifies where to look for an object. The initial context is typically used as a starting point.

In the simplest case, an initial context must be created using the specific implementation and extra parameters required by the implementation. The initial context will be used to look up a name. The **initial context is analogous to the root or top of a directory tree for a file system. Below is an example of creating an initial context**:

```
Hashtable args = new Hashtable();

// First you must specify the context factory.
// This is how you choose between jboss implementation
// vs. an implementation from Sun or other vendors.
args.put( Context.INITIAL_CONTEXT_FACTORY,
"com.jndiprovider.TheirContextFactory" );

// The next argument is the URL specifying where the data store is:
```

```
args.put( Context.PROVIDER_URL, "jndiprovider-database" );

// (You may also have to provide security credentials)

// Next you create the initial context
Context myCurrentContext = new InitialContext(args);
```

A context is then used to look up previously bound names in that context. For example:

```
MyBean myBean = (MyBean)
myCurrentContext.lookup("com.mydomain.MyBean");
```

Alternative to above code is as below:

The Context object can also be configured by adding jndi.properties file in classpath containing initial context factory class name and provider URL. The above code will be reduced as shown below:

```
//just need to create initial context object, it will try to read
jndi.properties file from //classpath.
Context myCurrentContext = new InitialContext();
```

A context is then used to look up previously bound names in that context. For example:

```
MyBean myBean = (MyBean)
myCurrentContext.lookup("com.mydomain.MyBean");
```

Searching

Attributes may be attached to special entries called directories. Directories enable searching for objects by their associated attributes. Directories are a type of context; they restrict the name space much like a directory structure on a file system does.

External links

- Oracle's JNDI page [2]

References

[1] http://www.sun.com/smi/Press/sunflash/1997-03/sunflash.970310.10204.html
[2] http://java.sun.com/products/jndi/

Article Sources and Contributors

Java Naming and Directory Interface *Source*: http://en.wikipedia.org/w/index.php?oldid=409350462 *Contributors*: Afsarkhan05, Antonrojo, Arjant, Bovineone, CanisRufus, Cedric dlb, Doug Bell, EagleOne, Fred Bradstadt, Harlan brown, Harryboyles, Helicasey, Hervegirod, Isnow, Jay, Julesd, LilHelpa, Lunacrescens, Mark T, Mjording, NathanBeach, Nicolas1981, Otisjimmy1, Raymondwinn, Raysonho, Rich Farmbrough, Sfmontyo, SkyWalker, Techi2ee, The Elves Of Dunsimore, The wub, Uriber, Uzume, Vijaykhambe, Welsh, Wikibofh, Wikimsd, Zumbo, 62 anonymous edits

License