Overview of JNDI

Introduction:

If a distributed application's components can't locate one another, then they can't work together. Therefore, distributed applications require, almost by definition, something to help the components to find each other. The Java Naming and Directory Interface (JNDI) provides just this capability.

# Uses of JNDI

1. Using JNDI, applications based on Java technology can store and retrieve named Java objects of any type.
2. In addition, JNDI provides methods for performing standard directory operations, such as associating attributes with objects and searching for objects using their attributes.

Divided into 2 parts

1. naming service
2. Directory service

# Naming Service

1. Maps names to objects. Refer Diagram : http://www.nyu.edu/classes/jcf/g22.3033-007\_sp01/handouts/g22\_3033\_h83.htm
2. A naming service maintains a set of bindings. Bindings relate names to objects. All objects in a naming system are named in the same way (that is, they subscribe to the same naming convention). Clients use the naming service to locate objects by name.
3. The primary function of a naming system is to bind names to objects (or, in some cases, to references to objects
4. In order to be a naming service, a service must at the very least provide the ability to bind names to objects and to look up objects by name.

Eg : A card catalog, for the uninitiated, maps the names of books to their location in the library. As surprising as it may seem, the notion of a card catalog is quite handy in the world of computing, as well. In computing, we call it a naming service, which associates names with the locations of services and with information. It provides computer programs with a single location where they can find the resources they need.

Binding

A naming service maintains a set of bindings. Bindings relate names to objects. All objects in a naming system are named in the same way (that is, they subscribe to the same naming convention). Clients use the naming service to locate objects by name.

# Context

A context is a set of name-to-object bindings. Every context has an associated naming convention. A context provides a lookup (resolution) operation that returns the object and may provide operations such as those for binding names, unbinding names, and listing bound names.

# Example of some existing naming Services

Some existing naming services : COS, DNS, LDAP, NIS

**Unix file system Naming Service**: The UNIX file system has a namespace consisting of all of the names of files and directories in that file system.

**DNS**: The DNS namespace contains names of DNS domains and entries. The DNS contains bindings that map machine names to IP addresses.

**The Internet's naming service** : maps people-friendly names (such as www.etcee.com) into computer-friendly IP (Internet Protocol) addresses in dotted-quad notation (207.69.175.36). Interestingly, DNS is a distributed naming service, meaning that the service and its underlying database is spread across many hosts on the Internet.

**LDAP**: The LDAP namespace contains names of LDAP entries. An LDAP name is bound to an LDAP entry. Developed by the University of Michigan . As its name implies, it is a lightweight version of DAP (Directory Access Protocol), which in turn is part of X.500, a standard for network directory services. Currently, over 40 companies endorse LDAP.

# How to use some existing Naming services and do lookup and get the required objects (context) for ‘Unix file system naming service’



# Example of using it in application server

Java EE components locate objects by invoking the JNDI lookup method. The names are bound to their objects by the naming and directory service that is provided by a Java EE server. Because Java EE components access this service through the JNDI API, the object usually uses its JNDI name. When it starts up, the Enterprise Server reads information from the configuration file and automatically adds JNDI database names to the name space.

# Example to do JDBC connectivity from JSP using the JNDI name

Refer : <http://docs.oracle.com/cd/E17952_01/refman-5.5-en/connector-j-usagenotes-glassfish-config-jsp.html>

# Access an ejb(deployed in glassfish and having jndi annotated mapped-name) from a remote class using jndi

**Remote Interface :**

****

**Stateless Ejb :**

****

**Access this ejb :**

****

# EJB JNDI Glassfish using deployment-descriptor configuration

**Ref :** <http://docs.oracle.com/cd/E18930_01/html/821-2418/beans.html>

**Configuration in Deployment Descriptor** :



**How to get the reference :**

****

Directory Service

A directory service associates names with objects and also allows such objects to have attributes. Thus, you not only can look up an object by its name but also get the object's attributes or search for the object based on its attributes.