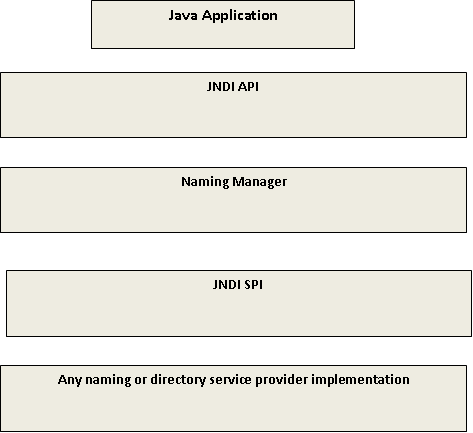
JNDI

JNDI is the acronym of **Java Naming and  Directory Interface.**JNDI is  a Java API which is using  by Java applications to find objects or data  with a specific name.In this section ,we are discussing a simple JNDI tutorial with simple examples.JNDI supports  variety of naming and directory services.Because JNDI architecture is independent of  any particular naming or directory service implementation. JNDI is inbuilt in Java 2  SDK from v1.3 onwards.

**JNDI Architecture**

JNDI has an API and an SPI(Service Provider Interface).  The API enables the Java applications  to access the naming and directory services. The SPI enables to plug in various naming and directory services. The Java applications can use these services using the API. This concept is explained below as a schematic.

The  service providers **LDAP**(**Lightweight Directory Access Protocol**),**RMI**(**Remote Method Invocation** ) **Registry** , **CORBA** (**Common Object Request Broker Architecture**)are inbuilt with Java  from  SDK v1.3  onwards.These are inbuilt by default.If we need other providers , then we can download it from this [**list**](http://www.oracle.com/technetwork/java/index.html).

**Overview to Naming Service**

A naming service maps developer friendly names to objects . So that the applications can access the same object with the bound name.

***Names*** *–*To do the  look up , one must know the name of object. The syntax of naming is determined by the naming service.It is known as the ***naming convention .***

***Bindings –*** The association between a name and an object is referred as *binding .*If  an application need to access an object using a name then the object should bound to the name before.

***Context*** – A context is a set of name to object bindings. The context provides the lookup.  It also provides options to bind ,unbind and rebind objects .

**InitilalContext** –It is the starting point of all naming and directory operations.

**Exceptions** : JNDI defines a hierarchy of Exceptions . The super class is **NamingExceptions**. All other exceptions are deriving from **NamingException.**

The ***javax.naming*** package contains the classes and interfaces  for accessing naming services.

**Overview to DirectoryService**

Similar to naming service here also it is possible to access objects with its name. In addition to this , it is possible to get the attributes of objects.Also it is possible to search for objects with attributes.

**Directory and Directory Service**

A ***directory*** is a connected set of  directory objects.A ***directory service*** is a service  which provides options for creating , adding ,removing and modifying attributes  associated with directory.

There are many Directory service implementation available now a days. Examples are DNS(Domain Name System) and LDAP(Light weight Directory Access Protocol).In the coming sections we will be discussing various operations on directory services using JNDI. In all those examples , we will be taking LDAP as reference .

To use JNDI concepts in our application , we should have the naming classes along with at least one service providers. Java provides inbuilt support for:

***a)Lightweight Directory Access Protocol (LDAP)***

***b)Common Object Request Broker Architecture (CORBA) Common Object Services (COS) name service***

***c)Java Remote Method Invocation (RMI) Registry***

If we need to use other services in our application , then we need to use the JNDI implementation for the same.

Example:

import javax.naming.Context;

import javax.naming.InitialContext;

import javax.naming.NamingException;

import java.util.Hashtable;

class Lookup {

public static void main(String[] args) {

// Check that user has supplied name of file to lookup

if (args.length != 1) {

System.err.println("usage: java Lookup <filename>");

System.exit(-1);

}

String name = args[0];

// Identify service provider to use

Hashtable env = new Hashtable(11);

env.put(Context.INITIAL\_CONTEXT\_FACTORY,

"com.sun.jndi.fscontext.RefFSContextFactory");

try {

// Create the initial context

Context ctx = new InitialContext(env);

// Look up an object

Object obj = ctx.lookup(name);

// Print it out

System.out.println(name + " is bound to: " + obj);

// Close the context when we're done

ctx.close();

} catch (NamingException e) {

System.err.println("Problem looking up " + name + ": " + e);

}

}

}