The Java Naming and Directory Interface (JNDI)

Fred Long
(Originally prepared by Helen Fuell)

Naming and Directory Services

- Naming and directory services are abstractions of the way in which we commonly attach names to objects, and arrange them in hierarchical structures (called directories)
- For example, file systems, the Domain Name System (DNS)
- Note that we often attach attributes to objects, as well as their name

X.500 and LDAP

- X.500 is a directory services standard developed by ISO and CCITT
- LDAP (Lightweight Directory Access Protocol) is a slimmed down version of X.500
- Both support compound names like:
  "cn=John Smith,o=Acme Products Ltd"
- Here, "cn" is an attribute, "John Smith" its value

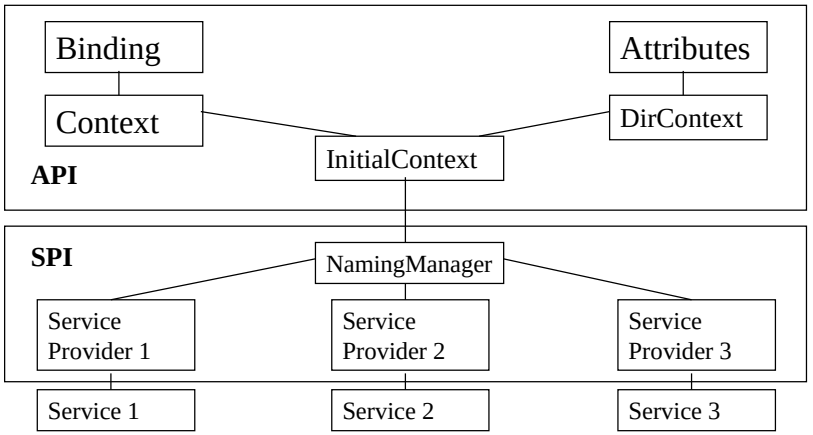
Common Key Attributes

c	A country
o	An organisation
ou	A division of an organisation
cn	The common name of an entry
sn	The surname of a user

JNDI

- JNDI provides a standard way of using naming and directory services in Java
- See: http://java.sun.com/products/jndi/index.html
- JNDI is released as part of the J2SE and J2EE

JNDI Architecture



Supported Services

- JNDI 1.1 service providers:

LDAP	Sun's support for LDAP versions 2 and 3
NIS	Sun's Network Information Service support
COS Naming	Sun's support for the CORBA naming service
RMI Registry	Sun's Remote Method Invocation support
File System	Sun's support for accessing the file system
Tengah	WebLogic's SPI for RMI, JDBC, EJB, Events
PNDS	Personal Naming and Directory Service for smart cards from GemPlus

- JNDI 1.2 (upgrade release) service providers:

LDAP	Sun's support for LDAP versions 2 and 3
NIS	Sun's Network Information Service support
COS Naming	Sun's support for the CORBA naming service
RMI Registry	Sun's Remote Method Invocation support
File System	Sun's support for accessing the file system
DNS	Sun's support for the Domain Name System
DSML	Sun's support for the Directory Service Markup Language (DSML)
Novell	Support for the Novell File System, NDS, NetWare 3X's Bindery, etc.

API Classes for Naming Services

- The following classes comprise the JNDI API for accessing naming services:
  - Binding
  - Context
  - InitialContext

Common Operations on Naming Services

- The following methods perform some common operations carried out on naming services:
  - lookup(String name)
  - list(String name)
  - listBindings(String name)
  - bind(String name, Object obj)
  - rebind(String name, Object obj)
  - unbind(String name)
  - destroySubcontext(String name)
  - createSubcontext(String name)
  - getEnvironment()
  - addToEnvironment(String propName, Object propVal)
  - removeFromEnvironment(String propName)

Example Code

- Setting up the initial context:

```
Properties props = new Properties ( );
props.put (Context.INITIAL_CONTEXT_FACTORY,
        "com.sun.jndi.fscontext.RefFSContextFactory");
props.put (Context.PROVIDER_URL, file:///);

Context initialContext = new InitialContext (props);
```

- Looking up an object:

```
Object obj = initialContext.lookup (name);
```


- Listing the children of a context:

```
NamingEnumeration children = initialContext.list ("");
```

- list returns an enumeration of NameClassPair objects; each NameClassPair object contains the name and class of a single child of the context which can be extracted using getName and getClassName (resp.)
- Listing the bindings of a context:

```
NamingEnumeration bindings = initialContext.listBindings ("");
```

- Now, we get an enumeration of Binding objects, containing the object and its name and class

API Classes for Directory Services

- The following classes comprise the JNDI API for accessing directory services:
  - Attributes
  - DirContext
  - InitialDirContext

Common Operations on Directory Services

- The following methods perform some common operations carried out on directory services:
  - getAttributes(String name) **or**
    getAttributes(String name, String [] attrIds)
  - modifyAttributes(String name, ModificationItem mods)
  - Search(String name, Attributes matchingAttributes)

Searching a Directory

- JNDI allows a directory to be searched using a filter, e.g.:

```
(objectclass=user)
(objectclass=*)
(cn=John*)
(revision<24)
(&(objectclass=user)(cn=John*))
(&(|(objectclass=user)(objectclass=computer))(cn=J*))
```

- Also, searches can be controlled in other ways

Example use of JNDI

```
mport java.util.Properties;
mport javax.naming.*;
ublic class Lookup {
        public static void main (String [] args) throws NamingException {
                String name = args[0];

                Properties props = new Properties();
                props.put(Context.INITIAL_CONTEXT_FACTORY,
                                "com.sun.jndi.fscontext.FSContextFactory");
                props.put(Context.PROVIDER_URL, file:///);

                Context initialContext = new InitialContext(props);

                Object obj = initialContext.lookup(name);
                System.out.println(name + " is bound to " + obj);
```