Application Servers G22.3033-011

Session 5 - Sub-Topic 1
Java Naming and Directory Interface (JNDI)

Dr. Jean-Claude Franchitti

New York University
Computer Science Department
Courant Institute of Mathematical Sciences

Agenda

- Naming and directory services
- JNDI
  naming
  directory
  service provider interface
- Q & A

Naming & Directory Services

- Naming services map names (people-friendly)
  to addresses or objects (machine-friendly)
  e.g: www.sun.com => 192.9.48.5
- Directory services add attributes and attribute-based searching
  e.g: find the two-sided printers in my hotel

Wide Range of Scale

- Global
  DNS, X.500
- Enterprise
  NIS, NIS+, LDAP, NDS, Active Directory
- Applications and services
  spreadsheet, calendars, file system, ...

Usage Examples

- Locating network resources
  printers, databases,
  Enterprise JavaBeans™ components
- Enterprise-wide namespace
  share file systems and other network services
- Security

Usage Examples (cont.)

- Accessing attributes of people and resources
  e-mail, calendars, find nearby color printers
- Support for distributed computing
  RMI registry, CORBA object references

What Is JNDI?

- A naming and directory interface for Java applications
- Enables access to existing and emerging naming and directory services
- Java language-centric design

JNDI Architecture

- Java Application
- JNDI API
- Naming Manager
- JNDI SPI
- LDAP, DNS, NIS, NDS, RMI, CORBA
- (JNDI Implementation Possibilities)

Naming Interface

- Names are relative

  Context

  Binding

- Operations include:
  lookup

Example: Lookup

-
  ```
  Printer p = (Printer)
      ctx.lookup("speedy");
  p.print(instream);
  ```

- Application gets back the object directly

- Naming service implementation(s) are hidden from application

Example: Bind

-
  ```
  Calendar c;
  ...
  ctx.bind("alice/cal", c);
  ```

- Naming service determines types of objects

- *Reference*s maximize object portability

Example: Browsing

-
  ```
  void traverse(Context ctx) {
      NamingEnumeration bindings =
          ctx.listBindings("");
      while (bindings.hasMore()) {
          Binding binding = (Binding)
              bindings.next();
          Object o = binding.getObject();

          // Do something with object...

          if (o instanceof Context)
              traverse((Context)o);}
  }
  ```

Initial Context

- Starting point for name resolution
- May contain a variety of bindings to useful and shared contexts
- Contents dynamically configurable

-
  ```
  Context ictx =
      new InitialContext(environment);
  ```

URLs as Names

- URLs may be used as names in initial context

  ```
  ictx.lookup("ldap://svr/o=Sun
  ,c=US");
  ```

Composite Names

- A name can span multiple namespaces
  ctx.lookup("eng.sun.com/printer/speedy")

Directory Interface

- *DirContext*
- *Attribute*
- Operations include:
  get and set attributes
  attribute-based search
  examine schema

Example: Get Attribute

-
  ```
  DirContext ctx;
  ...
  Attributes attrs =

  ctx.getAttributes("speedy");
  Attribute size =
  attrs.get("paperSize");
  …
  ```

Example: Set Attribute

-
  ```
  DirContext ctx;
  ...
  Attributes attrs =
      ctx.getAttributes("speedy");
  Attribute size = attrs.get("paperSize");
  ...
  size.add("legal");

  // directory has not yet been updated

  ctx.modifyAttributes("speedy",
                       REPLACE_ATTRIBUTE,
                       attrs);
  ```

Example: Search

- Find password of user "Bob":

```
NamingEnumeration results =
  ctx.search("user", "(uid=Bob)",
null);

SearchResult r = (SearchResult)
  results.next();

Attribute password =

r.getAttributes().get("userPassword");
```

Service Provider Interface

- Plug in support for naming and directory services
- Plug in support for new object types
- Supports federation of multiple systems

Once Again

- Share and manage network resources using naming and directory services
- Use JNDI to access these services, either individually or in federation

Status & Where To Go Next

- JNDI 1.1 software shipped February '98
- Service providers available for:
  LDAP, NDS, NIS, NIS+,
  CORBA, SLP, file system, ...
- Learn more, download, or send feedback:
  http://java.sun.com/jndi

Summary

- JNDI Provides network-wide sharing of a variety of information about users, machines, networks, services, and applications.
- JNDI is an API specified in Java^{tm} that provides naming and directory functionality to applications written in Java.
- JNDI is designed especially for Java by using Java's object model

Summary (continued)

- Using JNDI, Java applications can store and retrieve named Java objects of any type.
- JNDI provides methods for performing standard directory operations, such as associating attributes with objects and searching for objects using their attributes.
- JNDI is also defined independent of any specific naming or directory service implementation.

Summary (continued)

- JNDI enables Java applications to access different, possibly multiple, naming and directory services using a common API.
- Different naming and directory service providers can be plugged in seamlessly behind this common API.
- This allows Java applications to take advantage of information in a variety of existing naming and directory services, such as LDAP, NDS, DNS, and NIS(YP).

Summary (continued)

- This also allows Java applications to coexist with legacy applications and systems.
- Using JNDI as a tool, the Java application developer can build new powerful and portable applications that not only take advantage of Java's object model but are also well-integrated with the environment in which they are deployed.