JNDI

Interfaz de Nombrado y Directorio Java

(Java Naming and Directory Interface)

Pablo Nacarino Mntiel

¿Qué es?

- API de Java para servicios de directorio
  - ¿Qué significa esto?

    - API que permite buscar y usar tanto ficheros como objetos a través de un nombre, organizando dichos datos en una estructura que puede ser similar a una estructura de directorios.
    - Permite asociar un nombre a un fichero u objeto para que pueda ser usado por distintos servicios o aplicaciones evitando problemas de nomenclatura y/o de cambios en los ficheros u objetos

    - JNDI es usada por Java RMI y las APIs de Java EE para buscar objetos en una red

Particularidades de JNDI

- JNDI organiza sus nombres en una estructura jerárquica
- Un nombre puede ser cualquier string. Un nombre también puede ser un objeto que soporte la interfaz Name, sin embargo la forma más común de nombrar a un objeto es un string.
- Los objetos guardados pueden ser de cualquier tipo básico de Java o cualquier objeto Serializado
- Las implementaciones pueden hacer uso de un servidor, un fichero, o una base de datos a elección del desarrollador

Aplicación en caso real

Aplicaciones java, corriendo en el mismo o en distintos ordenadores

Esas aplicaciones tiene cada una sus propios ficheros de configuración con datos como parámetros de conexión a base de datos, direcciones IP de la red en la que pueden encontrar diversos servicios.

Los datos en esos ficheros de configuración o de propiedades son comunes para todas o muchas de estas aplicaciones.

Con JNDI solucionas problema repetir esos datos en muchos ficheros de configuración dentro del mismo ordenador, y el problema es aún mayor si tenemos que ir copiándolos por distintos ordenadores en la red.

Ejemplo Sencillo: levantar servidor JNDI

```
System.setProperty(Context.INITIAL_CONTEXT_FACTORY, "org.jnp.interfaces.NamingContextFactory");

NamingBeanImpl jnpServer = new NamingBeanImpl();
jnpServer.start();

Main main = new Main();
main.setNamingInfo(jnpServer);
main.setPort(5400);
main.setBindAddress(InetAddress.getLocalHost().getHostName());
main.start();
```

Ejemplo sencillo: Introducir datos

```
Hashtable<String, String> env = new Hashtable<String, String>();
env.put(Context.INITIAL_CONTEXT_FACTORY,"org.jnp.interfaces.NamingContextFactory");
env.put(Context.PROVIDER_URL,              "jnp://192.168.1.2:5400");
Context context = new InitialContext(env);

context.createSubcontext("config");
context.bind("/config/applicationName", "MyApp");
context.bind("/config/clase", new SomeData("pedro",4,new Date()));
```

Ejemplo sencillo: Acceso a datos desde cliente

```
final Hashtable<String, String> env = new Hashtable<String, String>();
env.put(Context.INITIAL_CONTEXT_FACTORY, "org.jnp.interfaces.NamingContextFactory");
env.put(Context.PROVIDER_URL, "jnp://192.168.1.2:5400");
Context context = new InitialContext(env);

System.out.println("Application name = "
      + context.lookup("java:/config/applicationName"));
System.out.println("someData = " + context.lookup("java:/config/clase"));
```