**Служба имен и каталогов JNDI**

**JNDI** (Java Naming and Directory Interface) - это API для доступа к службам имен и каталогов. *Службой имен*, в самом широком смысле, называют систему, управляющую отображением множества имен во множество объектов. Зная имя объекта в системе, можно получить доступ к этому объекту или ассоциировать с этим именем другой объект. Самым наглядным примером *JNDI* является служба доменных имен DNS, которая определяет соответствие между понятными доменными именами (например, localhost) и сетевым IP-адресом (127.0.0.1). Имея определенное доменное имя DNS, мы может узнать соответствующий ему IP-адрес.

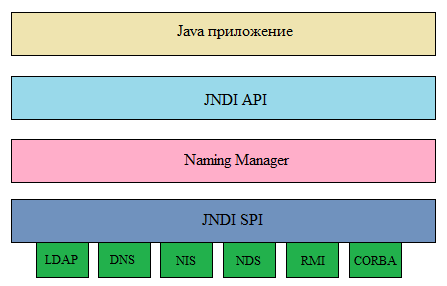
В службе каталогов именованные объекты сгруппированы в древовидную структуру. Кроме того объекты каталога имеют атрибуты. Наиболее близким и понятным примером такой службы является файловая система. Объекты файловой системы - файлы - собраны в каталоги и идентифицируются путями, например, C:\windows\notepad.exe. У файлов есть атрибуты: скрытый, архивный, только для чтения и другие. Передавая файловой системе путь, можно получить содержимое соответствующего файла, записать в него какие-то данные, изменить его атрибуты.

*JNDI* предназначен для единообразного доступа к разнообразным службам имен и каталогов, включая упомянутые выше DNS и файловую систему, а также LDAP, [DataSource](http://java-online.ru/datasource.xhtml). Разные службы каталогов интегрируются с *JNDI* через интерфейс поставщика услуг SPI (Service Provider Interface).

Концепция JNDI основана на двух основных определениях :

* ассоциация (binding) — соответствие JNDI-имени определенному объекту;
* контекст (context) — среда, в которой хранится набор ассоциаций между объектами и именами.

По сути, *JNDI* представляет собой аналог *JDBC* для служб имен и каталогов. Также, как и *JDBC* предоставляет стандартный Java API для доступа к разнообразным базам данных, JNDI стандартизует доступ к службам имен и каталогов. На следующем рисунке представлена архитектура *JNDI*.



Как показано на рисунке *JNDI* представляет обобщенную абстракцию доступа к самым разным службам имен, таким как LDAP (Lightweight Directory Access Protocol), DNS (Domain Naming System), NIS (Network Information Service), NDS (Novell Directory Services), RMI (Remote Method Invocation), CORBA (Common Object Request Broker Architecture). Получив экземпляр контекста JNDI, его можно использовать для поиска ресурсов в любой службе имен, доступной этому контексту. За кулисами **JNDI** взаимодействует со всеми доступными ей службами имен, передавая им имена ресурсов, которые требуется найти и выясняя, где в действительности находится искомый ресурс.

**Инициализация контекста JNDI**

Чтобы воспользоваться хранящимся в контексте *JNDI* ресурсом, необходимо инициализировать контекст *javax.naming.Context* и найти требуемый ресурс. Инициализация *Context'a* напоминает настройку драйвера JDBC при подключении к серверу базы данных.

Чтобы подключиться к службе имен или каталогов, необходимо получить библиотеки *JNDI* для этой службы. Это напоминает выбор соответствующего драйвера JDBC. Если требуется подключиться к LDAP, DNS или файловой системе компьютера, необходимо получить провайдера для соответствующей службы соответственно.

При работе в окружении Java EE (WEB), сервер приложений загружает все необходимые библиотеки для доступа к окружению *JNDI*. В противном случае необходимо настроить свое приложение, указав, какие библиотеки JNDI оно должно использовать. Сделать это можно, например, создав объект *Properties* и передав его конструктору *InitialContext* :

String CONTEXT = "oracle.j2ee.rmi.RMIInitialContextFactory";

String URL = "ormi://<host>:<port>/app";

String LOGIN = "SCOTT";

String PASSWORD = "TIGER";

Properties properties = new Properties();

properties.put(Context.INITIAL\_CONTEXT\_FACTORY, CONTEXT );

properties.put(Context.PROVIDER\_URL , URL );

properties.put(Context.SECURITY\_PRINCIPAL , LOGIN );

properties.put(Context.SECURITY\_CREDENTIALS , PASSWORD);

Context context = new InitialContext(properties);

В представленном примере выполняется настройка объекта *Properties* для доступа к дереву *JNDI* удаленного сервера БД Oracle. Необходимо принять во внимание, что параметры подключения к *JNDI* зависят от конкретного производителя и представленный пример не является универсальным. Следует обращаться к документации с описанием своего сервера приложений, чтобы узнать, как организовать удаленное подключение.

Другой способ определения настроек *JNDI* связан с созданием файла *jndi.properties* и определением пути в CLASSPATH приложения. Содержимое файла настроек представляет те же самые пары имя/значение, что помещаются в объект Properties. После этого данный файл будет использоваться автоматически при создании контекста вызовом *InitialContext* :

Context context = new InitialContext();

В таблице перечислены наиболее часто используемые свойства **JNDI**, необходимые для подключения к удаленному провайдеру службы JNDI.

|  |  |  |
| --- | --- | --- |
| **Свойство** | **Описание** | **Пример** |
| java.naming.factory.initial Context.INITIAL\_CONTEXT\_FACTORY | Имя фабричного класса, который будет использоваться для создания контекста. | oracle.j2ee.rmi.RMIInitialContextFactory |
| java.naming.provider.url Context.PROVIDER\_URL | Адрес URL службы JNDI - определение расположения службы (формат описания URL зависит от службы). | ormi://localhost:98765/chapter1 |
| java.naming.security.principal Context.SECURITY\_PRINCIPAL | Идентификационная информация (учетная запись), позволяющая аутентифицировать вызывающую программу в службе JNDI. | SCOTT |
| java.naming.security.credentials Context.SECURITY\_CREDENTIALS | Используемый для аутентификации пароль пользователя. | TIGER |

javax.naming.Context.INITIAL\_CONTEXT\_FACTORY - определяет класс, который будет формировать экземпляр JNDI Context. Если работаете с DNS, то необходимо указывать класс, который создает Context для взаимодействия с DNS сервером (см. пример ниже).

**Поиск ресурсов в JNDI, lookup**

Метод **lookup** интерфейса javax.naming.Context возвращает ресурс с именем name, который следует привести к требуемому типу.

Object lookup (String name)

Если в аргументе *name* передать пустую строку, то возвращается новый экземпляр Context.

Чтобы отыскать ресурс, необходимо знать его имя. Допустим, что в JNDI компонент BidService зарегистрирован под именем “/ejb/bid/BidService”. Чтобы найти его, можно выполнить поиск непосредственно :

Context context = new InitialContext();

BidService service = (BidService) context.lookup("/ejb/bid/BidService");

или последовательно :

Context newContext = new InitialContext();

Context bidContext = (Context) newContext.lookup("/ejb/bid/");

BidService service = (BidService) bidContext.lookup("BidService");

**Пример JNDI**

Широкое распространение *JNDI* получила при подключении к серверу БД. Пример использования *JNDI* для настройки подключения *DataSource* к серверу базы Oracle в сервере приложений JBoss рассмотрен [здесь](http://java-online.ru/datasource.xhtml).

Ниже приведен пример, в котором используется *JNDI* для просмотра содержимого корня контекста DNS сервера. В примере использован открытый адрес URL сайта Yandex, размещенный в Википедии.

import javax.naming.Context;

import javax.naming.NameClassPair;

import javax.naming.InitialContext;

import javax.naming.NamingException;

import javax.naming.NamingEnumeration;

import java.util.Properties;

public class ExampleJNDI

{

static String DNS\_CONTEXT = "com.sun.jndi.dns.DnsContextFactory";

static String DNS\_URL = "dns://77.88.8.8"; // yandex

ExampleJNDI() {

Properties props = new Properties ();

props.put (Context.INITIAL\_CONTEXT\_FACTORY, DNS\_CONTEXT);

props.put (Context.PROVIDER\_URL , DNS\_URL );

try {

Context context = new InitialContext (props);

NamingEnumeration<NameClassPair> names = context.list ("");

while ( names.hasMoreElements ())

System.out.println (names.nextElement());

} catch (NamingException e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

new ExampleJNDI();

System.exit(0);

}

}

В результате работы программы в консоли можно увидеть следующие сообщения :

ltd: javax.naming.directory.DirContext

casino: javax.naming.directory.DirContext

alfaromeo: javax.naming.directory.DirContext

amsterdam: javax.naming.directory.DirContext

cat: javax.naming.directory.DirContext

car: javax.naming.directory.DirContext

photography: javax.naming.directory.DirContext

cam: javax.naming.directory.DirContext

aquarelle: javax.naming.directory.DirContext

theatre: javax.naming.directory.DirContext

media: javax.naming.directory.DirContext

total: javax.naming.directory.DirContext

diet: javax.naming.directory.DirContext

today: javax.naming.directory.DirContext

actor: javax.naming.directory.DirContext

fans: javax.naming.directory.DirContext

career: javax.naming.directory.DirContext

...

В Enterprise JavaBeans *JNDI* используется для нахождения всех видов ресурсов, включая компонент [EJB](http://java-online.ru/ejb.xhtml), пул соединений с базой данных, информацию об окружении и многое другое. Таким образом, из окна контейнера EJB можно «увидеть остальной мир» посредством **JNDI**. Клиентское приложение также использует *JNDI* для получения соединения с фабрикой EJB.

Что представляет собой поставщик услуг при использовании *JNDI* с EJB? Ответ заключается в том, что контейнер EJB определяет свою собственную службу **JNDI**, специализированную для работы с ресурсами, управляемыми контейнером. Иными словами, служба JNDI в этом случае позволяет клиентам и Enterprise JavaBeans'ам находить ресурсы по JNDI именам. Следует помнить, что когда вы стартует Ваш контейнер EJB, то также неявно запускается EJB JNDI служба, которая доступна через стандартный JNDI API. О формате именования компонентов EJB подробно представлено [здесь](http://java-online.ru/ejb.xhtml#jndi-ejb).