

# Java Naming and Directory Interface

*Helder da Rocha*  
[www.argonavis.com.br](http://www.argonavis.com.br)


- Este seminário tem como objetivo abordar os **conceitos fundamentais** relativos a sistemas de nomes e diretórios em geral e mostrar como usar o **JNDI** para usar tais sistemas
  - Abrange aspectos essenciais da API para manipulação de nomes, contextos e atributos (não entra em detalhes sobre LDAP, SPI, federações, etc.)
  - Finalidade principal: preencher um pré-requisito essencial para o desenvolvimento de aplicações J2EE.
- O roteiro deste capítulo é baseado nas duas primeiras trilhas do **JNDI Tutorial** da Sun (por Rossana Lee)
  - Veja onde encontrar esta e outras fontes no final desta apresentação.
  - Os exemplos e exercícios (opcionais) sobre diretórios requerem o acesso a um servidor LDAP (com permissão de leitura) e DNS


- *Introdução ao JNDI*
  - *Conceitos básicos sobre nomes e diretórios*
  - *Classes e pacotes da API JNDI*
- *Operações com nomes*
  - *Contexto inicial*
  - *Localização (lookup) de objetos*
  - *Contextos*
  - *Ligações (bindings)*
- *Operações com diretórios*
  - *Atributos*
  - *Pesquisas*
- *Registro de objetos*

# Conceitos fundamentais

- *Conceitos relativos a sistema de nomes*
  - *Serviço de nomes*
  - *Convenções de nomenclatura*
  - *Ligação (binding)*
  - *Referências e endereços*
  - *Contextos e subcontextos*
  - *Sistema de nomes e espaço de nomes*
- *Conceitos relativos a sistemas de diretórios*
  - *Diretórios e serviços de diretórios*
  - *Sistemas de diretórios*
  - *Atributos*
  - *Pesquisas e filtros*
  - *LDAP*

# Serviço de nomes

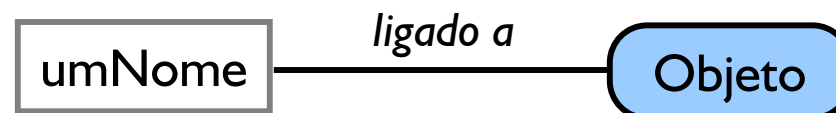
- A principal função de um serviço de nomes é permitir a associação de um **nome** (ou uma outra representação alternativa mais simples) a recursos computacionais como
  - endereços de memória, de rede, de serviços
  - objetos e referências
  - códigos em geral
- Suas duas funções básicas são
  - Associar (mapear) um nome a um recurso
  - Localizar um recurso a partir de seu nome
- Exemplos
  - Sistema de arquivos: liga caminho a bloco(s) de memória:  


```
c:\temp\dados.txt — 16A0:0C00
```
  - Sistema DNS: liga nome de domínio a endereço IP:  


```
www.argonavis.com.br — 200.206.192.193
```

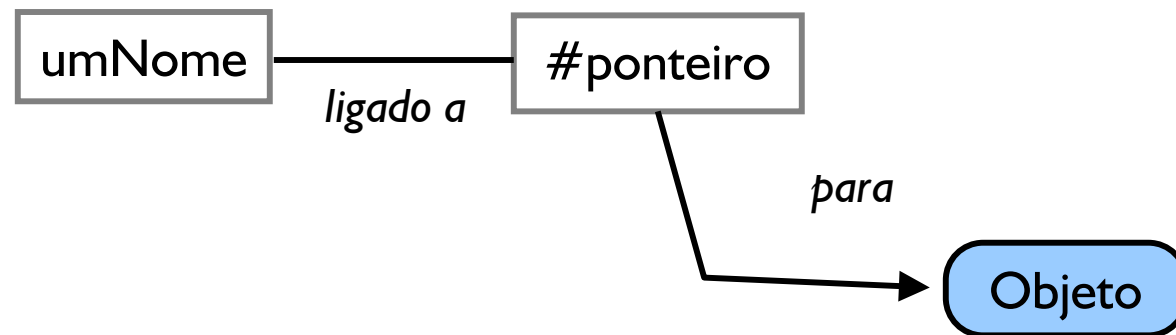
# Ligação ou mapeamento (Binding)

- *É a associação de um nome com um objeto (ou com um localizador do objeto)*
- *Exemplos:*
  - *Nome de arquivo no DOS está **ligado** a um bloco de memória*
  - *Nome de arquivo no Mac está **ligado** a um ou dois blocos de memória (resource e data fork)*
  - *Nome de máquina na internet está **ligado** a endereço IP*
  - *Nome de objeto em ORB está **ligado** a uma instância remota do objeto*



# Ponteiros e referências

- Muitas vezes, o sistema de nomes não armazena o recurso identificado pelo nome mas guarda apenas uma **referência** (ou ponteiro) para ele
  - A referência pode conter apenas a informação necessária para buscar o objeto
  - Informações ou até referências adicionais podem ser obtidas em consultas posteriores usando essa referência



# Convenções de nomenclatura

- *Todo sistema de nomes obedece a uma determinada convenção que determina sua sintaxe*
  - *Estruturas similares com convenções diferentes: incompatibilidade!*
- **Exemplos**
  - *Convenção para nomes de arquivos no Unix requer que os componentes do nome sejam ordenados da esquerda para a direita relativos à raiz do sistema (/) separados pelo caractere "/":*  
***/usr/bin/arquivo.txt***
  - *Convenção para nome de domínio estabelece que componentes do nome sejam ordenados relativos à raiz do domínio (.com, .net, .br) da direita para a esquerda e separados pelo caractere ponto:*  
***maq.subdom.dom.com.br***
  - *Convenção para nome de recurso LDAP requer que os pares nome-valor sejam ordenados da direita para a esquerda e separados por vírgula:* ***tel=43211234, area=11, pais=55***



# Nomes atômicos, compostos e federações

- **Nome atômico** é o componente indivisível de um nome
  - `arquivo.txt` em `c:\textos\arquivo.txt`
  - `usr` em `/usr/local/bin/java`
  - `o=Argo Navis` em `cn=Helder da Rocha, o=Argo Navis, c=BR`
  - `org` em `jakarta.apache.org`
- **Nome composto** (Compound Name) é a combinação de zero ou mais nomes atômicos de acordo com uma certa convenção
  - Os nomes à direita, na lista acima, são nomes compostos
  - Tem **múltiplas ligações**; uma para cada nome atômico
- **Federação** (Federation ou Composite Name) é um nome que mistura nomes de espaços de nome diferentes. Exemplo:
  - `http://java.sun.com/products/j2ee/index.html`
  - Espaço de nomes URL Scheme ID: `http://` (define o protocolo)
  - Espaço de nomes DNS: `java.sun.com`
  - Espaço de nomes do sistema de arquivos: `/products/j2ee/index.html`

# Contextos e subcontextos

- Um **contexto** é um conjunto de ligações nome-objeto
  - *Em outras palavras, é um objeto que tem zero ou mais ligações*
- Se o objeto (referência) contido no contexto for também um contexto ele é um **subcontexto**
  - *O escopo do subcontexto é limitado pelo seu contexto pai*
- A **resolução** de um nome (obtenção do objeto associado) ocorre em um escopo limitado pelo seu contexto
- Cada contexto possui uma **operação** de resolução associada
- Exemplos de contextos e subcontextos:
  - */usr/bin/java/*  
**usr** é o contexto; **bin** é subcontexto de **usr**, ...
  - *www.abc.com.br*  
**br** é o contexto, **com** é subcontexto de **br**, ...

# Sistema e espaço de nomes

- Um **sistema de nomes** é um conjunto interligado de contextos que respeitam a mesma convenção e possuem um conjunto comum de operações
- O sistema de nomes oferece um **serviço de nomes** através de suas operações
  - Exemplo: sistema de arquivos. Oferece sistema que mapeia nomes de arquivo a diretórios e arquivos
- O **espaço de nomes** consiste de todo o conjunto de nomes disponíveis em um sistema de nomes
  - Exemplo: espaço de nomes do sistema de arquivos corresponde a todos os nomes de arquivo e diretório de um sistema
- O espaço de nomes de uma federação é chamado de espaço de nomes composto

# Diretórios e serviços de diretório

- **Diretório**: conjunto interligado de objetos
  - Organização não precisa ser hierárquica (contextual)
  - Para cada item há um **nome** unívoco (chave)
  - Cada item possui um ou mais **atributos**
- Um **serviço de diretório** é oferecido para criar, remover, modificar e principalmente pesquisar atributos associados a objetos em um diretório
  - Diretório = tipo de banco de dados acessível via rede
  - Projetado para ser mais eficiente na recuperação de dados que na gravação ou alteração
  - Atualizações são simples, sem transações e envolvendo pequena quantidade de dados
- Exemplos
  - NIS (Network Information System)
  - X.500 - serviço antigo baseado em OSI, precursor do LDAP

# Sistemas de nomes e diretório

- *Sistemas de nomes são frequentemente estendidos com **sistemas de diretório***
  - *Associa nomes com objetos (como um serviço de nomes)*
  - *Permite que objetos tenham atributos (qualificadores)*
- *Pode-se*
  - *Procurar o objeto pelo nome (serviço de nomes)*
  - *Procurar o objeto pelos seus atributos*
  - *Obter os atributos de um objeto*
- ***Objetos de diretório** representam objetos em um sistema de diretórios assim como nomes representam objetos em um sistema de nomes*
  - *Além de representar um objeto, objetos de diretório possuem atributos que descrevem o objeto associado.*

- *Descrevem objeto associado a um objeto de diretório*
- *Um atributo possui*
  - *Um **identificador** de atributo: permite que o atributo seja localizado e utilizado*
  - ***Conjunto de valores** de atributo: as informações (dados) que estão associadas com o atributo*
  - *Um **tipo**: restringe os dados que um atributo pode receber*
- *Atributos fazem parte do contexto do objeto*
- *Vários serviços também suportam **pesquisas** com base no conteúdo dos objetos (seus atributos)*
  - *Busca é formada por uma **expressão** envolvendo os atributos dentro de um determinado contexto*
  - *Expressão pode resultar em mais de um objeto*

- **Lightweight Directory Access Protocol**
  - *Protocolo leve para acesso a diretórios (padrão aberto)*
  - *Armazena objetos em uma árvore (DIT - directory information tree)*
  - *Define vários atributos, tipos e sintaxes padrão baseados no X.500*
  - *Extensível (pode-se criar novos tipos, atributos, etc.)*
- **Entradas no diretório são formadas de atributos**
  - *Um dos atributos é usado para identificar o objeto em um contexto:*  
**RDN - Relative Distinguished Name** - unívoco no contexto
  - *Uma seqüência ordenada de RDNs identifica o objeto globalmente:*  
**DN - Distinguished Name** - chave unívoca global!
  - *Cada atributo tem um tipo e um valor*
  - *Cada objeto tem um tipo (definido por uma ou mais classes)*
- **Diretórios baseados em LDAP suportam**
  - *qualquer tipo de dados*
  - *várias formas de segurança (criptografia, autenticação, integridade)*

# Uma entrada em um diretório LDAP

```
dn: cn=Niels Bohr,ou=People,o=JNDITutorial,dc=argonavis,dc=com,dc=br
cn: Niels Bohr
sn: Bohr
mail: Niels.Bohr@JNDITutorial.com
telephonenumber: +1 408 555 7562
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
```

Atributos

DN: Distinguished Name (seqüência de RDNs)

Atributos objectclass (informam as classes às quais pertencem esta entrada)

bohr.ldif

- Cada contexto é identificado por um nome (par **atributo=valor**)
- Cada contexto tem atributos
- Obtenção de um contexto

permite acesso a

- atributos
- subcontextos
- atributos dos subcontextos

Atributos X-500

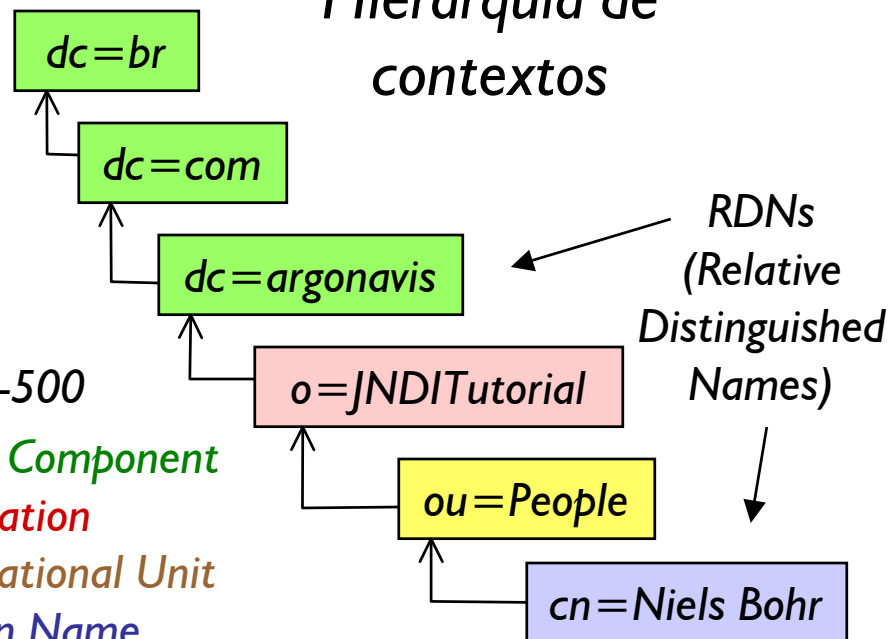
dc: Domain Component

o: Organization

ou: Organizational Unit

cn: Common Name

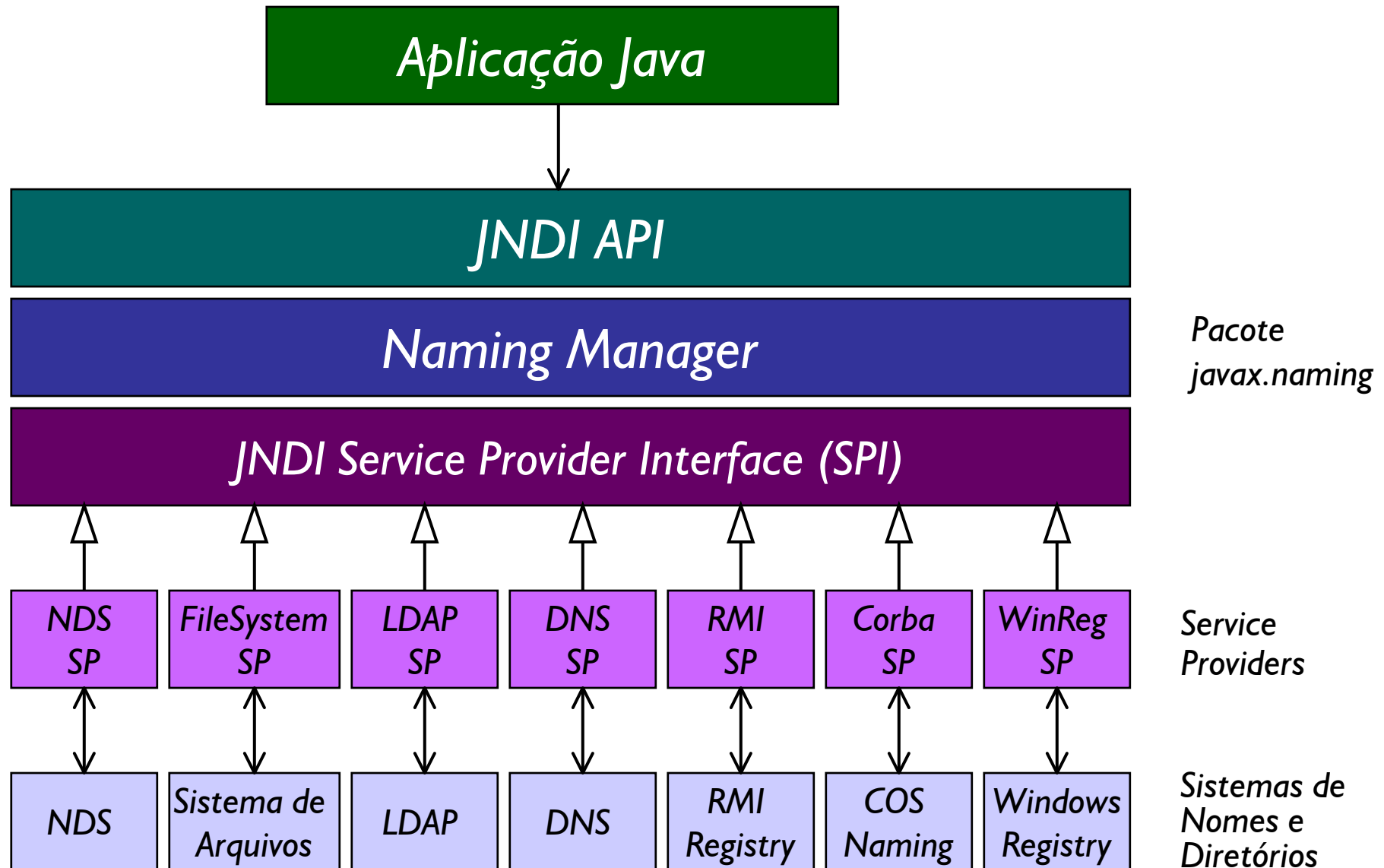
Hierarquia de contextos





- **Java Naming and Directory Interface** é uma *ponte* sobre os diversos serviços de nomes e diretórios diferentes
- **Vantagens**
  - *Só é preciso aprender uma única API para acessar vários tipos de informação de serviços de diretório*
  - *Isola a aplicação dos detalhes específicos do protocolo*
  - *Pode ser usada para ler objetos Java (serializados) que estejam armazenados em um diretório*
  - *Pode combinar diferentes tipos de diretório (federação) e tratá-los como um diretório único*
- **Componentes**
  - **API** - *Application Programming Interface*
  - **SPI** - *Service Provider Interface* que permite que novos serviços sejam *plugados* transparentemente

# Arquitetura JNDI



- A API JNDI está incluída no J2SDK 1.3 ou posterior nos pacotes e subpacotes descendentes de **javax.naming**.
- Para usar JNDI é preciso ter
  - As classes e interfaces do JNDI (pacotes **javax.naming.\***)
  - Pelo menos um provedor de serviços JNDI (driver)
- O Java 2 SDK inclui provedores de serviço (SPs) para
  - **LDAP** - Lightweight Directory Access Protocol
  - **CORBA** - Common ORB Architecture e COS name service
  - **Java RMI Registry**
- Outros provedores de serviços (para sistema de arquivos, para DNS, para JDBC, para Windows Registry, etc.) podem ser encontrados a partir do site

<http://java.sun.com/products/jndi/serviceproviders.html>

# Principais classes

- A API JNDI consiste de cinco pacotes
- **javax.naming** contém as principais **classes** e **interfaces**
  - **Context**: interface onde se pode recuperar, ligar, desligar e renomear objetos, e criar e destruir contextos
  - **InitialContext**: ponto de partida (raiz) para todas as operações
  - **Name**: abstração de um nome. Contém geralmente um String de texto que corresponde ao nome do objeto ou contexto
  - **NameClassPair**: contém nome do objeto e de sua classe
  - **Binding**: contém nome do objeto ligado, nome da classe do objeto e o próprio objeto
  - **Reference**: abstração de uma referência para um objeto
  - **NamingEnumeration**: um tipo de `java.util.Enumeration` usado para colecionar componentes de um contexto
  - **NamingException**: principal exceção do JNDI

## Principais classes (2)

- **javax.naming.directory** contém recursos para diretórios
  - **DirContext**: é uma extensão de **Context**. Herda todos os seus métodos e acrescenta métodos específicos para diretórios como
    - **getAttributes()** que recupera todos os atributos,
    - **search()** que oferece várias formas de busca e
    - **modifyAttributes()** que permite remover, criar e redefinir atributos
  - **Attribute** e **Attributes** representam um atributo e uma coleção deles
- **javax.naming.event** tem como principais componentes a classe **NamingEvent** e a interface **NamingListener** usadas para responder à alterações em objetos e atributos
- **javax.naming ldap** contém interfaces e classes para recursos específicos do LDAP v.3
- **javax.naming.spi** oferece classes e interfaces usados pelos fabricantes de provedores de serviço

# Exemplos: configuração

- *Para executar os exemplos que serão mostrados a seguir, seu ambiente deve estar configurado*
- *Software necessário (além do J2SDK)*
  - *Provedores de serviços **fscontext** e **dns** para o sistema de arquivos e DNS (pegue-os no site do JNDI, em [java.sun.com](http://java.sun.com))*
  - *Acesso a um servidor **LDAP** para os exemplos que usam diretórios (será preciso povoá-lo)*
- *Os exemplos estão disponíveis abaixo de **cap02/src** e podem ser compilados e executados com o Ant*
  - *É preciso editar o código-fonte e adequá-lo ao seu ambiente*
- *Os arquivos de configuração usados neste capítulo não são os mesmos do JNDI Tutorial*
  - *Foram simplificados para evitar configurações trabalhosas (principalmente os que usam LDAP)*

# Exemplo 1: Sistema de Nomes

Lookup.java

```
1:import javax.naming.Context;
2:import javax.naming.InitialContext;
3:import javax.naming.NamingException;
4:import java.util.Properties;
5:
6:class Lookup {
7:    public static void main(String[] args) {
8:        Properties env = System.getProperties();
9:        env.put(Context.INITIAL_CONTEXT_FACTORY,
10:               "com.sun.jndi.fscontext.RefFSContextFactory");
11:        try {
12:            Context ctx = new InitialContext(env);
13:            Object obj = ctx.lookup(args[0]);
14:            System.out.println(args[0]+"esta ligado a: " + obj);
15:            ctx.close();
16:        } catch (NamingException e) {
17:            System.err.println("Incapaz de achar "+args[0]+": "+e);
18:        }
19:    }
20:}
```

Define provedor de serviços

Escolhe contexto inicial

Procura objeto pelo nome

# Exemplo 1: execução

- Para compilar, rode "ant" no diretório cap02/
- Para executar a partir do diretório cap02/build/, use
  - > `java -cp .;../lib/fscontext.jar;../lib/providerutil.jar jnditut.intro.Lookup nome_do_arquivo`
  - ou rode o target `run` do buildfile com o ant:
    - > `ant run -Dclass=jnditut.intro.Lookup -Darg=arquivo`
- Contexto default: configure-o em `lib/jndi.properties!`
  - Pode ser alterado passando propriedade via linha de comando:
    - `-Djava.naming.url.provider=file:///caminho`
- Se nome passado for um arquivo, será impresso seu caminho:  
`autoexec.bat` ligado a `C:\autoexec.bat`
- Se nome for de diretório, será impresso o valor da referência:  
`windows\system` ligado a:  
`com.sun.jndi.fscontext.RefFSContext@9931f5`



## Exemplo 2: acesso a diretórios LDAP

- Para este exemplo é necessário **configurar** um servidor LDAP
- No diretório **lib/** há um arquivo **exemplos.ldif** (LDAP Data Interchange Format) para povoar o diretório
  - O contexto raiz default é **o=JNDITutorial**. Se necessário, altere-o em **exemplos.ldif** (faça uma busca e substituição) para colocá-lo no contexto do seu domínio LDAP. Por exemplo troque-o por:  
**o=JNDITutorial,dc=empresa,dc=com,dc=br**
- Para importar os dados em um servidor OpenLDAP rodando em Linux, use:
  - > **ldapadd -x -f exemplos.ldif**
- Para executar\* use, a partir do subdiretório **build/** local:
  - > **java jnditut.intro.Getattr**ou rode o target **run** do Ant
  - > **ant run -Dclass=jnditut.intro.Getattr**

# Exemplo 2: código-fonte

Getattr.java

```
1:import javax.naming.*;
2:import javax.naming.directory.InitialDirContext;
3:import javax.naming.directory.DirContext;
4:import javax.naming.directory.Attributes;
5:import java.util.Properties;
6:
7:class Getattr {
8:  public static final String LDAP="localhost:389";
9:  public static void main(String[] args) {
10:   Properties env = System.getProperties();
11:   env.put(Context.INITIAL_CONTEXT_FACTORY,
12:           "com.sun.jndi.ldap.LdapCtxFactory");
13:   env.put(Context.PROVIDER_URL, "ldap://" + LDAP + "/o=JNDITutorial");
14:   try {
15:     DirContext ctx = new InitialDirContext(env);
16:     Attributes at = ctx.getAttributes("cn=Niels Bohr, ou=People");
17:     System.out.println("sn: " + at.get("sn").get());
18:     ctx.close();
19:   } catch (NamingException e) {
20:     System.err.println("Incapaz de obter atributo: " + e);
21:   }
22: }
23:}
```

Coloque aqui o endereço do seu servidor LDAP

Contexto raiz (mude se preciso)

Subcontextos

# Contexto inicial

- Precisa ser obtido antes de qualquer operação. Passos:

- 1: selecionar o provedor de serviços

```
Properties env = new Properties();  
env.put(Context.INITIAL_CONTEXT_FACTORY,  
        "classe.do.ProvedorDeServicos");
```

- 2: configurar o acesso ao serviço

```
env.put(Context.PROVIDER_URL, "ldap://xyz.com:389");  
env.put(Context.OUTRA_PROPRIEDADE, "valor"); (...)
```

- 3: criar um objeto para representar o contexto

```
Context ctx = new InitialContext(env);
```

- A configuração (1, 2) pode ser feita via propriedades do sistema

- passadas em linha de comando via agrumento **-Dprop=valor**
- carregados via arquivos de propriedades

- Principais propriedades

```
java.naming.factory.initial: Context.INITIAL_CONTEXT_FACTORY  
java.naming.provider.url: Context.PROVIDER_URL
```

# Recuperação de objetos (lookup)

- Para obter a referência para um objeto de um contexto usa-se o método **lookup()**
  - Para usar o objeto retornado é preciso conhecer o seu tipo e fazer o `cast` (ou `narrow`, se objeto remoto) para promover a referência
  - Se o objeto for um contexto, `lookup()` age como um método para mudar de contexto (como o `chdir`, em Unix)

## Exemplo

- O método **lookup()** usando com o provedor de serviço **fscontext** retorna um **java.io.File** pelo nome de arquivo

```
env.put(Context.INITIAL_CONTEXT_FACTORY,  
        "com.sun.jndi.fscontext.RefFSContextFactory");  
  
env.put(Context.PROVIDER_URL,  
        "file:/cap02/lab/filesys");  
  
Context ctx = new InitialContext(env);  
File f = (File)ctx.lookup("report.txt");
```

← Serviço de nomes para sistema de arquivos

← Diretório raiz do serviço

← Arquivo localizado na raiz do serviço

# Listagem do conteúdo de contextos

- Em vez de obter um objeto de cada vez via `lookup()`, pode-se obter uma lista deles
  - Método **`list()`** retorna uma lista (**`NamingEnumeration`**) de pares nome / nome\_da\_classe (objetos do tipo **`NameClassPair`**)

```
NamingEnumeration lista = ctx.list("awt");
while (lista.hasMore()) {
    NameClassPair nc = (NameClassPair)lista.next();
    System.out.println(nc);
}
```

Trecho de List.java

- Método **`listBindings()`** retorna uma lista de ligações nome / objeto (**`Binding`**)

```
NamingEnumeration lista = ctx.listBindings("awt");
while (lista.hasMore()) {
    Binding bd = (Binding)lista.next();
    System.out.println(bd.getName() + ": " + bd.getObject());
}
```

Trecho de ListBindings.java

- Rode os exemplos\*: `ant run -Dclass=jnditut.naming.List`

\* Antes de rodar os exemplos em `cap02/` rode `ant setupfs` para configurar o ambiente

# Modificação de ligações, objetos e contextos

## ■ Adicionando ligações

```
Fruit fruit = new Fruit("orange");  
ctx.bind("favorite", fruit);
```

*Bind.java*

## ■ Substituindo ligações

```
Fruit fruit = new Fruit("lemon");  
ctx.rebind("favorite", fruit);
```

*Rebind.java*

## ■ Removendo ligações

```
ctx.unbind("favorite");
```

*Unbind.java*

## ■ Renomeando objetos

```
ctx.rename("report.txt", "old_report.txt");
```

*Rename.java*

## ■ Criando novos contextos

```
Context result = ctx.createSubcontext("new");
```

*Create.java*

## ■ Destruindo contextos

```
ctx.destroySubcontext("new");
```

*Destroy.java*

Rode *List* ou *ListBindings* após cada operação para ver os resultados

# Leitura de Atributos (em diretório LDAP)

- Recupera **todos** os atributos do objeto representado pelo RDN "**cn=Niels Bohr, ou=People**"

```
Attributes answer =  
    ctx.getAttributes("cn=Niels Bohr, ou=People");  
  
for (NamingEnumeration ae = answer.getAll(); ae.hasMore(); ) {  
    Attribute attr = (Attribute)ae.next();  
    System.out.print("attribute: " + attr.getID());  
    for (NamingEnumeration e = attr.getAll();  
         e.hasMore();  
         System.out.println("; value: " + e.next()) ) ;  
}
```

Subcontexto  
↙

- Recupera **apenas** atributos "sn" e "mail" se existirem

```
String[] attrIDs = {"sn", "mail"};  
Attributes answer =  
    ctx.getAttributes("cn=Niels Bohr, ou=People", attrIDs);
```

# Leitura de Atributos (em diretório DNS)

## ■ Inicialização do contexto

```
env.put(Context.INITIAL_CONTEXT_FACTORY,  
         "com.sun.jndi.dns.DnsContextFactory");  
env.put(Context.PROVIDER_URL, "dns://192.168.1.1/com.br");
```

## ■ Leitura dos atributos

```
DirContext ctx = new InitialDirContext(env);  
String[] attrIDs = {"A", "MX", "NS", "CNAME"};  
  
Attributes atts =  
    ctx.getAttributes("www.argonavis", attrIDs);  
  
System.out.println("Host: argonavis.com.br: ");  
for (NamingEnumeration ae = atts.getAll(); ae.hasMore(); ) {  
    Attribute attr = (Attribute)ae.next();  
    System.out.print("attribute: " + attr.getID());  
    for (NamingEnumeration e = attr.getAll(); e.hasMore(); ) {  
        System.out.println("; value: " + e.next());  
    }  
}  
}
```

Servidor DNS

Raiz do contexto  
pequisado

Subcontexto

Atributos DNS  
selecionados



# Modificação de atributos (exemplos LDAP)

- *Tarefa pouco freqüente em um diretório típico*

```
modifyAttributes(String rdn, int tipo, Attributes atributos)
```

```
modifyAttributes(String rdn, ModificationItem[] selecao)
```

- *Constante numérica indica o tipo de modificação:*

- `DirContext.ADD_ATTRIBUTE`
- `DirContext.REPLACE_ATTRIBUTE`
- `DirContext.REMOVE_ATTRIBUTE`

- *Exemplo:*

- *Define um novo **mail** no lugar do antigo para **cn=Niels Bohr***

```
Attribute att = new BasicAttribute("mail", "bohr@novo.com")
```

```
ModificationItem[] mods = new ModificationItem[1];
```

```
mods[0] =
```

```
    new ModificationItem(DirContext.REPLACE_ATTRIBUTE, att);
```

```
ctx.modifyAttributes("cn=Niels Bohr, ou=People", mods);
```

# Pesquisa simples

- Principal operação em um diretório é a **pesquisa**
- Forma mais simples localiza objeto informando subcontexto e lista de atributos (e valores)
  - Exemplo: Procura objetos contendo atributo **sn** com valor **Bohr** e atributo **mail** (de conteúdo não especificado)

```
Attributes matchAttrs = new BasicAttributes(true);
matchAttrs.put( new BasicAttribute("sn", "Bohr") );
matchAttrs.put( new BasicAttribute("mail") );
```

```
NamingEnumeration answer =
    ctx.search("ou=People", matchAttrs);
```

Faz a busca dentro do subcontexto **ou=People**

```
while ( answer.hasMore() ) {
    SearchResult sr = (SearchResult) answer.next();
    System.out.println( ">>>" + sr.getName() );
    GetattrsAll.printAttrs( sr.getAttributes() );
}
```

Navega  
pelos  
objetos  
do conjunto  
de resultados

implementação de loops for para ler atributos (veja GetAttrsAll.java)

# Filtros de pesquisa

- Permitem reduzir o escopo da pesquisa realizada em um contexto através de **expressões lógicas** sobre atributos

- Exemplo

```
( &(sn=Bohr) (mail=*) )
```

- Objeto que possui atributo **sn** contendo **Bohr** e atributo **mail**

- Vários **operadores** (veja livro-texto para mais detalhes)

- **&**, **|**, **!**, **=**, **~**, **<=**, **>=**, **=\***, **\***, **\**

- Exemplo

```
// Controles de pesquisa default
SearchControls ctls = new SearchControls();

// Define filtro
String filtro = "&(sn=Geisel) (mail=*)";

// Pesquisa no contexto ou=People usando o filtro
NamingEnumeration res =
    ctx.search("ou=People", filtro, ctls);
```

# Controles de pesquisa

- *Permitem controlar abrangência da pesquisa*
  - *Veja exemplos no livro-texto*
- **Limite de *Escopo*:**
  - *SearchControls.ONELEVEL\_SCOPE: default - pesquisa ocorre apenas no nível atual do contexto*
  - *SearchControls.SUBTREE\_SCOPE: pesquisa ocorre na subárvore de descendentes (inclusive o contexto)*
  - *SearchControls.OBJECT\_SCOPE: pesquisa limita-se ao objeto*
- **Limite de *Quantidade de respostas***
  - *Permite definir um limite máximo de respostas (provoca exceção **SizeLimitExceededException** caso limite seja ultrapassado)*
- **Limite de *Tempo***
  - *Define o tempo máximo que uma operação poderá esperar pelas respostas (provoca exceção **TimeLimitExceededException**)*

# Provedores de serviços para objetos

- *Pode-se usar JNDI para mapear nomes a objetos*
  - *Objetos localizáveis por nome podem ser abstraídos do contexto ou até linguagem em que são usados*
  - *Aplicações diferentes podem compartilhar objetos*
- *Dois drivers JNDI estão disponíveis para acesso a objetos distribuídos no J2SDK*
  - **SPI CORBA** (COS Naming): *permite localização de objetos CORBA (serializados em um formato independente de linguagem)*  
`com.sun.jndi.cosnaming.CNContextFactory`
  - **SPI RMI**: *permite a localização de objetos Java serializados (objetos pode ser usados por outras aplicações Java)*  
`com.sun.jndi.rmi.registry.RegistryContextFactory`

## ■ *Propriedades a definir*

`java.naming.factory.initial` ou `Context.INITIAL_CONTEXT_FACTORY`  
`com.sun.jndi.rmi.registry.RegistryContextFactory`

`java.naming.provider.url` ou `Context.PROVIDER_URL`  
`rmi://nome_do_host:1099` (endereço e porta do RMI Registry)

## ■ *Mapeamento*

```
Context ctx = new InitialContext();  
Fruit fruit = new Fruit("orange");  
ctx.bind("favorite", fruit);
```

## ■ *Localização*

```
Context ctx = new InitialContext();  
Fruit fruit = (Fruit) ctx.lookup("favorite");
```

# Objetos CORBA (RMI sobre IIOP)

## ■ Propriedades a definir

```
java.naming.factory.initial ou Context.INITIAL_CONTEXT_FACTORY  
com.sun.jndi.cosnaming.CNCtxFactory
```

```
java.naming.provider.url ou Context.PROVIDER_URL  
iiop://nome_do_host:1900 (endereço e porta do ORB)
```

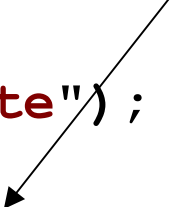
## ■ Mapeamento

```
Context ctx = new InitialContext();  
Fruit fruit = new Fruit("orange");  
ctx.bind("favorite", fruit);
```

## ■ Localização

```
Context ctx = new InitialContext();  
Object corbaFruit = ctx.lookup("favorite");  
Object javaFruit =  
    javax.rmi.PortableRemoteObject.narrow(corbaFruit,  
                                           Fruit.class);  
Fruit fruit = (Fruit) javaFruit;
```

Converte objeto CORBA  
em objeto Java



- *Em servidores J2EE, pools de conexões de bancos de dados são acessíveis através de objetos DataSource, publicadas no JNDI abaixo no namespace java:/*
  - `java:/CloudscapeDB`
  - `java:/OracleDB`
- *O mapeamento dos nomes (bind) não faz parte da especificação J2EE*
  - *Geralmente é realizada através das ferramentas administrativas do servidor ou XMLs proprietários*
- *Para acessar um banco*

```
Context ctx = new InitialContext();  
javax.sql.DataSource ds = (javax.sql.DataSource)  
    ctx.lookup("java:/Cloudscape");  
java.sql.Connection = ds.getConnection();
```



- *Sistemas de nomes e diretórios abstraem conceitos específicos a um domínio de problema*
  - *Maior facilidade do uso de recursos*
  - *Maior desacoplamento*
- *JNDI fornece uma interface genérica para diversos sistemas de nomes e diretórios diferentes*
  - *Permite que recursos compartilhados, localizados através de diferentes sistemas de nomes e diretórios possam ser armazenados e localizados usando uma interface uniforme*
  - *A mesma interface pode ser usada para interagir com sistemas de arquivos, LDAP, registro do Windows, DNS, ORBs, RMI Registry e outros repositórios de objetos, etc.*

- [1] Rossana Lee. *The JNDI Tutorial*, Sun Microsystems, 2002  
<http://java.sun.com/products/jndi/tutorial/>  
*A maior parte deste capítulo é baseada nas primeiras duas seções (trilhas) do JNDI Tutorial.*
- [2] David N. Blank-Edelman. *Perl for System Administration*. Appendix B: *The 10 minute LDAP Tutorial*. O'Reilly and Associates, 2000  
*Introdução à estrutura de registros LDAP.*
- [3] Jeff Hodges. *Introduction to Directories and the Lightweight Directory Access Protocol (LDAP)*. Stanford University, 1997  
<http://www.stanford.edu/~hodges/talks/mactivity.ldap.97>  
*Pequena introdução a serviços de diretórios e LDAP*
- [4] *OpenLDAP*: [www.openldap.org](http://www.openldap.org). *Vários documentos sobre configuração e uso do servidor OpenLDAP.*
- [5] Ed Roman et al. *Mastering EJB 2.0*, Wiley, 2001  
<http://www.theserverside.com/books/masteringEJB/index.jsp>  
*Apêndice A tem um breve e objetivo tutorial sobre JNDI*

*helder@argonavis.com.br*

***argonavis.com.br***

*Revisão 2.0 (Junho 2002)*

*Tutorial JNDI, dezembro 2001*

*J500 - Curso de Java 2 Enterprise Edition, Janeiro 2002*

*Atualizado em Junho 2002*