



Tehnologii Java

Curs -

Cristian Frăsinaru

`acf@infoiasi.ro`

Facultatea de Informatică

Universitatea "Al. I. Cuza" Iași





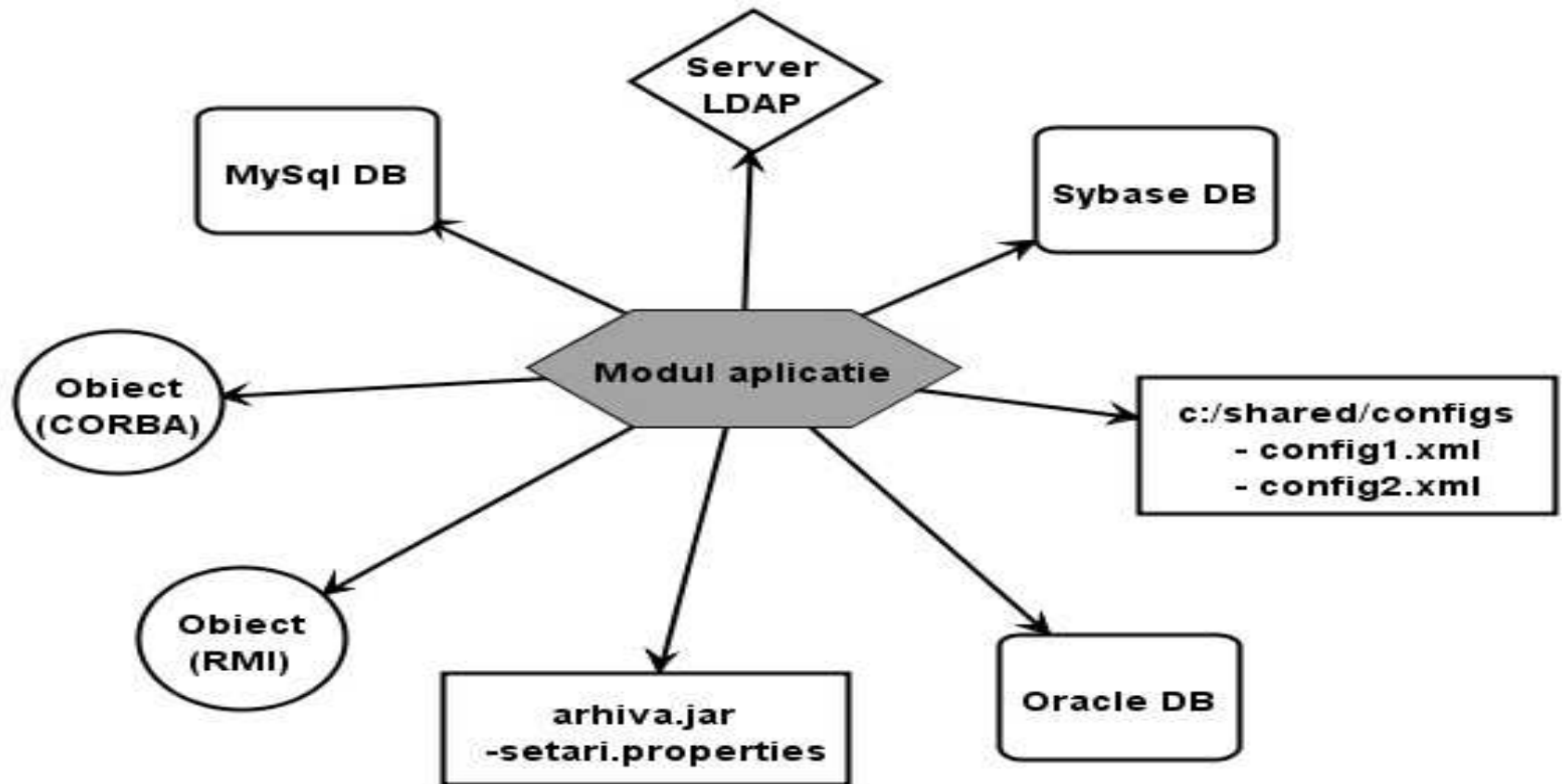
Servicii Java de acces la resurse



Cuprins

- Servicii de nume
- Servicii de directoare
- JNDI
- Exemple RefFS, LDAP, RMI
- Service Locator
- JDBC și JNDI

Contextul de lucru





Servicii de nume și directoare



Ce este un serviciu de nume ?

Serviciu de nume = Mecanism de asociere a unor **nume unice (atomice)** obiectelor unui sistem, pe baza cărora acestea să poată fi regăsite ulterior.

Nume atomic \mapsto **Referință** \mapsto **Obiect**

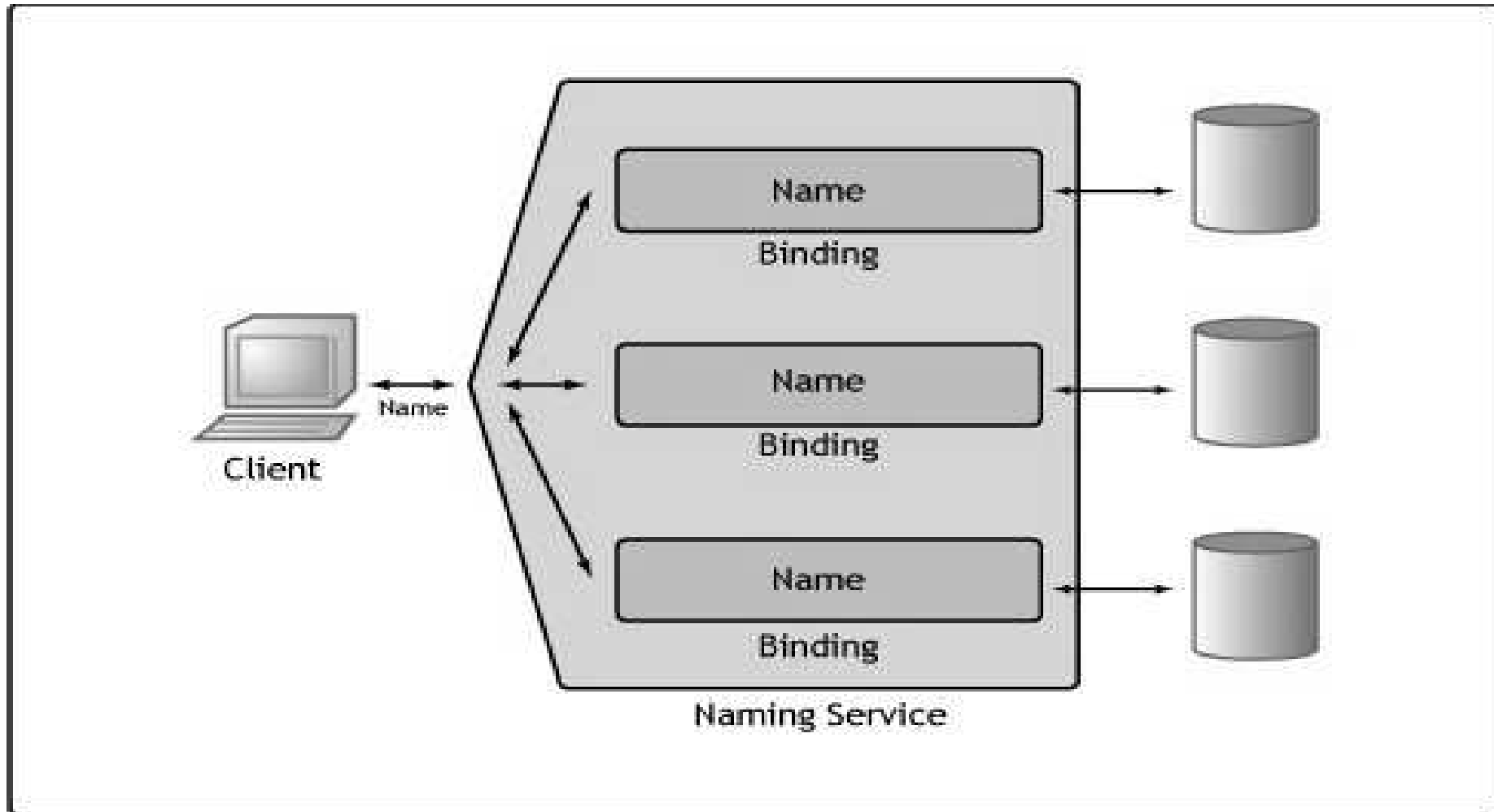
Exemple:

nume fișier relativ/absolut \mapsto fișier

nume domeniu Internet \mapsto adresa IP

nume RMI \mapsto obiect Java

Imagine de ansamblu



Concepte legate de nume

- **Mapare (legătură, binding)** = asocierea dintre un nume și un obiect
- **Context** = o mulțime de legături (bindings)
- **Subcontext** = Un context în cadrul altui context de același tip
- **Sistem de nume** = mulțime de contexte de același tip
- **Serviciu de nume** = serviciu care asigură operații asupra unui sistem de nume
- **Spațiu de nume** = Mulțimea tuturor numelor dintr-un sistem de nume

Ce este un serviciu director ?

Serviciu director = Extensie a unui serviciu de nume ce permite asocierea unor **attribute** obiectelor unui sistem.

Nume \mapsto **Obiect director + Attribute**

Exemplu

LDAP - model ierarhic de bază de date distribuită, obiectele având attribute de tipul:
(uid, cn, sn, l, ou, o, dc, st, c)

Concepte legate de directoare

- **Atribut** = informație asociată cu un obiect director.
Format din *identificator* și o mulțime de *valori*
- **Director** = o mulțime de obiecte directoare
- **Serviciu director** = un serviciu care oferă operații pentru crearea, adăugarea și actualizarea atributelor obiectelor unui director.
- **Căutare** = Procesul de identificare a obiectelor pe baza atributelor sale (*content based search*)

```
cn=John Doe,ou=people,dc=wikipedia,dc=org
```

```
%cn=common name, dc=domain component, ou=organization unit
```

Exemple de servicii

- **DNS (Domain Name System)**
- **NFS (Sun's Network File System)**
- **NIS (Network Information Service)**
- **NDS (Novell's Directory Service)**
- **ADS (Microsoft's Active Directory Services)**
- **RMI Registry**
- **COS (Common Object Service) Naming - CORBA**
- **LDAP (Lightweight Directory Access Protocol)**

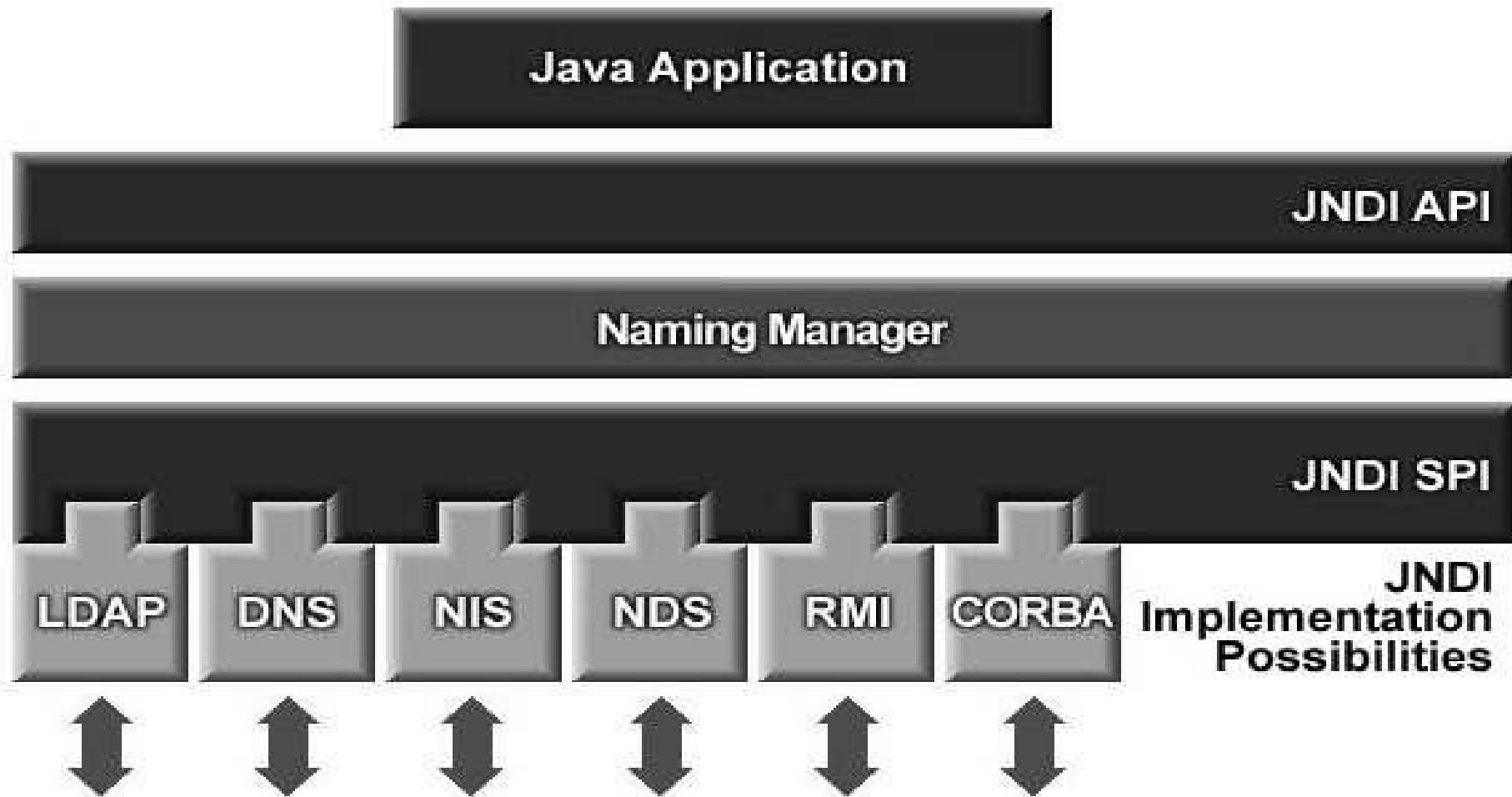
JNDI

Ce reprezintă JNDI ?

JNDI = Java Naming and Directory Interface

- **API** (Application Programming Interface)
Interfață de accesare a serviciilor de nume și directoare, independentă de tipul acestora.
- **SPI** (Service Provider Interface)
Mecanismul prin care un *furnizor de servicii* de nume sau directoare poate dezvolta și integra în arhitectura JNDI o implementare proprie care să poată fi accesată prin API-ul JNDI.

Arhitectura JNDI



Operații uzuale cu nume

- Setarea unui *context inițial*

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

- Căutarea unui obiect după numele său

```
Printer printer = (Printer)ctx.lookup("treekiller");  
printer.print(report);
```

- Crearea de noi legături, listarea sau distrugerea acestora, redenumiri, etc.

```
context.bind(nume, obiect);  
    .rebind  
    .unbind  
    .listBindings
```

Exemplul 1 (RefFS)



```
// Stabilirea contextului initial
Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY,
        "com.sun.jndi.fscontext.RefFSContextFactory");
env.put(Context.PROVIDER_URL, "file:///c:/jdk1.5.0/bin/");

String name = "java.exe";
try {
    // Crearea contextului initial
    Context ctx = new InitialContext(env);

    // Cautarea obiectului dupa nume
    Object obj = ctx.lookup(name);
    System.out.println("Nume=name, + " Obiect=" + obj);
    // Nume=java.exe, Obiect=c:/jdk1.5.0/bin/java.exe
} catch (NamingException e) {
    System.err.println("Numele "+ name + " nu este in context\n" + e);
}
```



Exemplul 2 (RMI)

Alinierea serviciului rmiregistry la JNDI.

```
jndi.properties (in CLASSPATH)
-----
java.naming.factory.initial=
    com.sun.jndi.rmi.registry.RegistryContextFactory
java.naming.provider.url=
    rmi://adresaServer:port
```

in aplicatia RMI

```
-----
Naming.rebind
    --> context.rebind
Naming.lookup("rmi://adresaServer:port/numeObiect")
    --> context.lookup(numeObiect);
```

Operații uzuale cu directoare

- Operațiile din cadrul serviciului de nume
- Setarea unui *context inițial*

```
DirContext ctx = new InitialDirContext(proprietati);
```

- Obținerea atributelor unui obiect

```
ctx.getAttributes("sn=Nelu, ou=Info");
```

- Adăugarea, modificarea, ștergerea atributelor unui obiect
- Căutarea obiectelor pe baza atributelor sale

Exemplu (LDAP)



```
Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY,
        "com.sun.jndi.ldap.LdapCtxFactory");
env.put(Context.PROVIDER_URL, "ldap://ldap.uaic.ro");
try {
    // Creare context initial
    DirContext ctx = new InitialDirContext(env);

    // Cautarea unui obiect dupa atribute
    Attributes attrs = ctx.getAttributes("cn=gigi, ou=info");

    // Aflarea altor atribute
    System.out.println("sn: " + attrs.get("sn").get());
} catch (NamingException e) {
    System.err.println("Eroare la obtinerea atributelor!\n" + e);
}
```



Evenimente

Evenimentele oferă un mecanism de *notificare* în cazul modificării stării obiectelor serviciului. Folosind clase de tip *listener* pot fi implementate acțiuni care să se execute:

- la adăugarea, ștergerea sau redenumirea unui obiect în cadrul unui context
- la modificare atributelor, etc.

```
interface NamespaceChangeListener {  
    void objectAdded(NamingEvent evt);  
    void objectRemoved(NamingEvent evt);  
    void objectRenamed(NamingEvent evt);  
}
```

"Depozitarea" obiectelor

Un serviciu de nume și directoare poate fi folosit și pentru stocarea obiectelor.

↳ **Interoperabilitatea** sistemelor/aplicațiilor.

Tipuri de obiecte:

- **serializabile**
- **referințe** (Referenceable sau JNDI Reference)
- **cu attribute**
- RMI, CORBA, etc.

Object Factory = Responsabil cu crearea obiectelor stocate într-un serviciu.

JDBC și JNDI

JDBC 3.0

- **Suport pentru JNDI**
- *Connection Pooling*
Eficientizarea procesului de obținere a unei conexiuni la o bază de date
- *RowSet*
Extensie conformă cu specificațiile JavaBeans a interfeței `ResultSet`
- *Distributed Transactions*
Integrare în arhitectura Java EE

Surse de date

O bază de date este abstractizată printr-un obiect de tip **DataSource**.

- Responsabil cu obținerea unei conexiuni
- Alternativa preferată la `DriverManager`
- Oferit de producătorul bazei de date la nivel de driver (driver JDBC 3.0)
- Integrat în arhitectura JNDI

Conectarea 'clasică'

Implementată uzual printr-o clasă *Singleton*.

```
public class DBConnector {
    private static Connection connection = null;

    public static Connection getConnection() {
        if (connection != null) {
            return connection;
        }
        try {
            Class.forName("org.hsqldb.jdbcDriver").newInstance();
            connection = DriverManager.
                getConnection("jdbc:hsqldb:hsqldb://localhost/test");
        } catch (Exception e) {
            return null;
        }
        return connection;
    }
}
```

Conectarea folosind *DataSource*



```
// Instantiate a DataSource object
org.hsqldb.jdbc.jdbcDataSource ds;
ds = new org.hsqldb.jdbc.jdbcDataSource();

// Set connection properties.
ds.setUser("user");
ds.setPassword("passwd");
ds.setDatabaseName("test");
ds.setServerName("localhost");
ds.setPortNumber(9001);

// Open connection
Connection conn = ds.getConnection();
System.out.println("Connection successful!");
```



Conectarea prin JNDI



1. Inregistrarea bazei în serviciul de directoare

```
org.hsqldb.jdbc.jdbcDataSource ds;  
...  
Context ctx = new InitialContext(env);  
ds = new org.hsqldb.jdbc.jdbcDataSource();  
ds.setDatabase("jdbc:hsqldb:hsqldb://localhost/test");  
ctx.bind("db/test", ds);  
ctx.close();
```

2. Obținerea unei conexiuni

```
...  
Context ctx = new InitialContext(env);  
DataSource ds = (DataSource)ctx.lookup("db/test");  
connection = ds.getConnection();
```



JDBC-JNDI în servere de aplicații

Serverele de aplicații care implementează specificațiile Java EE permit configurarea într-un mod standard a surselor de date.

GlassFish

Admin Console

- Resources

 - Connection Pools

 - *postgres/timetable_pool

- JDBC Resources

 - *postgres/timetable

Utilizarea resursei JDBC

```
try {
    //-----
    InitialContext ic = new InitialContext();
    //-----

    String jndiName = "postgres/timetable";

    DataSource ds = (DataSource) ic.lookup(jndiName);

    Connection connection = ds.getConnection();
    ...

} catch (NamingException ne) {
    throw new RuntimeException(ne);
}
```

Service Locator

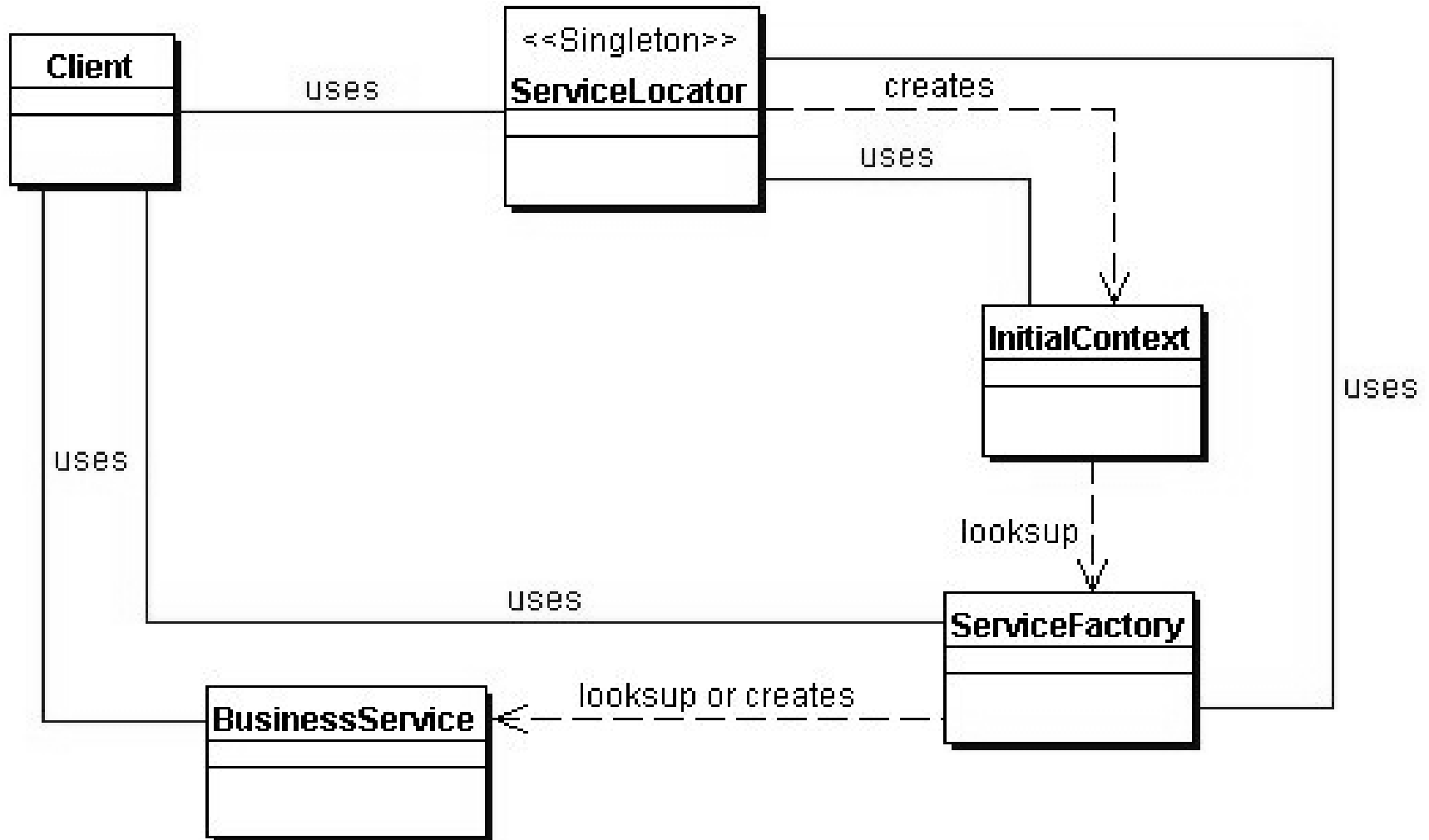
Problema

Căutarea serviciilor folosind JNDI poate implica executarea unor operații de rețea costisitoare.

Service Locator

- abstractizarea utilizării JNDI
- ascunderea complexității creării contextului inițial
- reducerea complexității codului
- asigurarea unui punct unic de control
- îmbunătățirea performanțelor prin folosirea unor structuri *cache*

Structura unui *Service Locator*



Exemplu *ServiceLocator*

```
public class MyServiceLocator {
    private InitialContext ic;

    public MyServiceLocator() {
        try {
            ic = new InitialContext();
        } catch (NamingException ne) {
            throw new RuntimeException(ne);
        }
    }

    private Object lookup(String jndiName)
        throws NamingException {
        return ic.lookup(jndiName);
    }

    public DataSource getDataSource(String dataSourceName)
        throws NamingException {
        return (DataSource) lookup(dataSourceName);
    }
}
```

Exemplu *MyCachingServiceLocator*

```
public class MyCachingServiceLocator {
    private InitialContext ic;
    private Map<String, Object> cache;

    private MyCachingServiceLocator() throws NamingException {
        ic = new InitialContext();
        cache = Collections.synchronizedMap(
            new HashMap<String, Object>());
    }
    private Object lookup(String jndiName) throws NamingException {
        Object cachedObj = cache.get(jndiName);
        if (cachedObj == null) {
            cachedObj = ic.lookup(jndiName);
            cache.put(jndiName, cachedObj);
        }
        return cachedObj;
    }
}
```

Dependency Injection

```
public class MyServlet extends HttpServlet {
    @Resource(mappedName = "postgres/timetable")
    private DataSource orar;
    ...
}
```

Indirectare

```
<sun-web-app>
  <resource-ref>
    <res-ref-name>timetable</res-ref-name>
    <jndi-name>postgres/timetable</jndi-name>
  </resource-ref>
  ...
</sun-web-app>
```

```
@Resource(name = "timetable")
private DataSource orar;
```