Université Claude Bernard Lyon 1

Conception d'Applications Hétérogènes Distribuées

Master 2 Traitement de l'Information

Lionel Médini

Septembre-novembre 2015

Plan du cours

- Outils de programmation avancés
  - Retour sur les patrons de conception
    - Inversion de contrôle (conteneurs d'objets)
    - Contexte et annuaires (communication dans un conteneur)
    - Réserve d'instances (gestion du cycle de vie)
- Systèmes d'information distribués
  - Appel de méthodes distantes (CORBA, RMI)
- Frameworks Java
  - Objets transactionnels distribués (EJB)
  - Composants déployables "à chaud" (OSGi)
- Introduction à l'urbanisation des SI

Le pattern Contexte

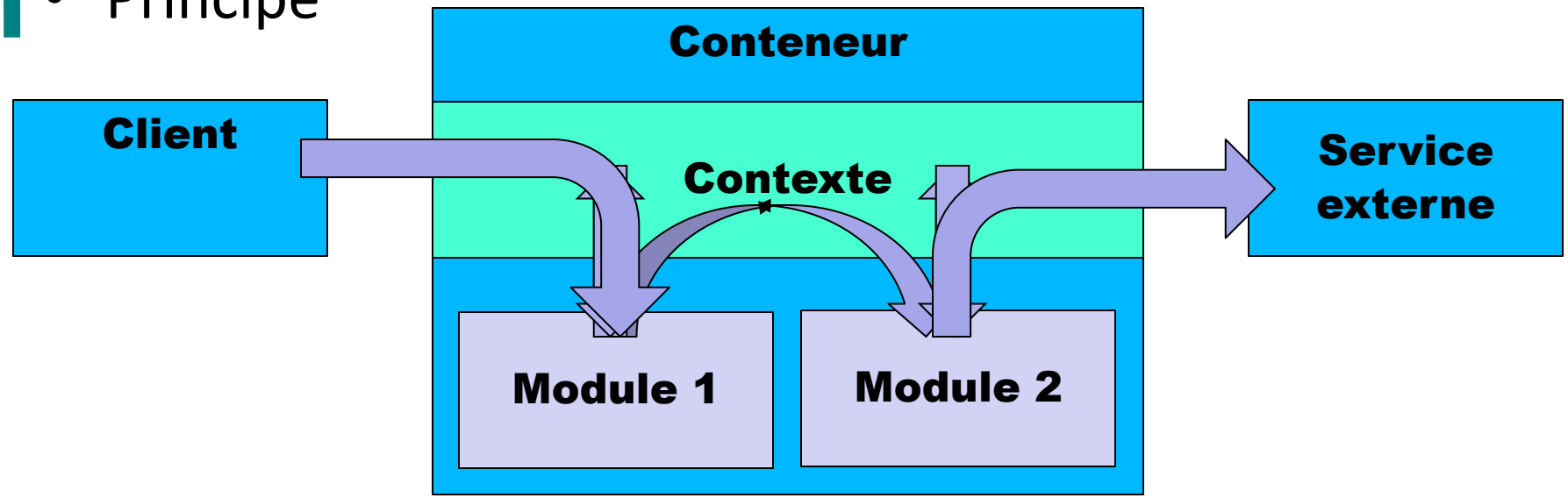
- “Contexte”
  - Un conteneur diminue le couplage entre ses composants en les isolant les uns des autres et de l’extérieur
  - Le conteneur n’a pas accès aux classes d’implémentation des composants
  - Les composants ne connaissent pas le type de conteneur
- Problème
  - Les composants ont besoin de communiquer entre eux et avec l’extérieur
  - Le framework doit pouvoir contrôler les communications avec et entre les composants

Le pattern Contexte

- Principe
  - Ajout d'un niveau d'indirection entre conteneur et composants
    - Spécifique aux types de modules (servlets, EJB, POJO...)
    - Spécifique au conteneur (Web, EJB...)
  - Rediriger les communications vers cet objet
- Exemples
  - javax.servlet.ServletContext (Java EE) : environnement des servlets d'un conteneur Web
  - javax.ejb.EJBContext (Java EE) : informations contextuelles liées à un EJB
- Remarque
  - Un composant peut avoir plusieurs contextes associés

Le pattern Contexte

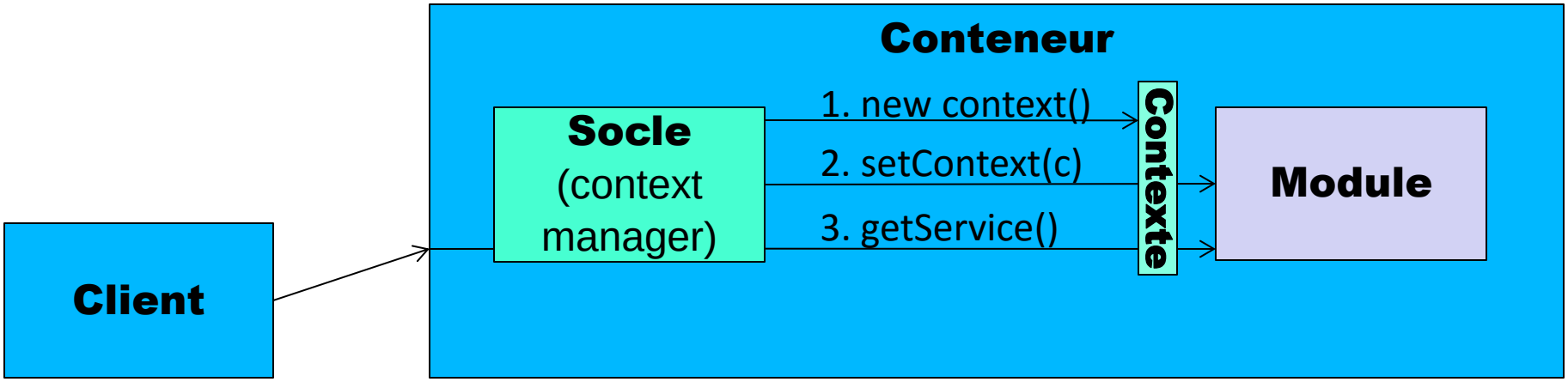
- Principe



- Communication entre les composants et l'extérieur
  - Les modules ne communiquent pas directement avec l'extérieur

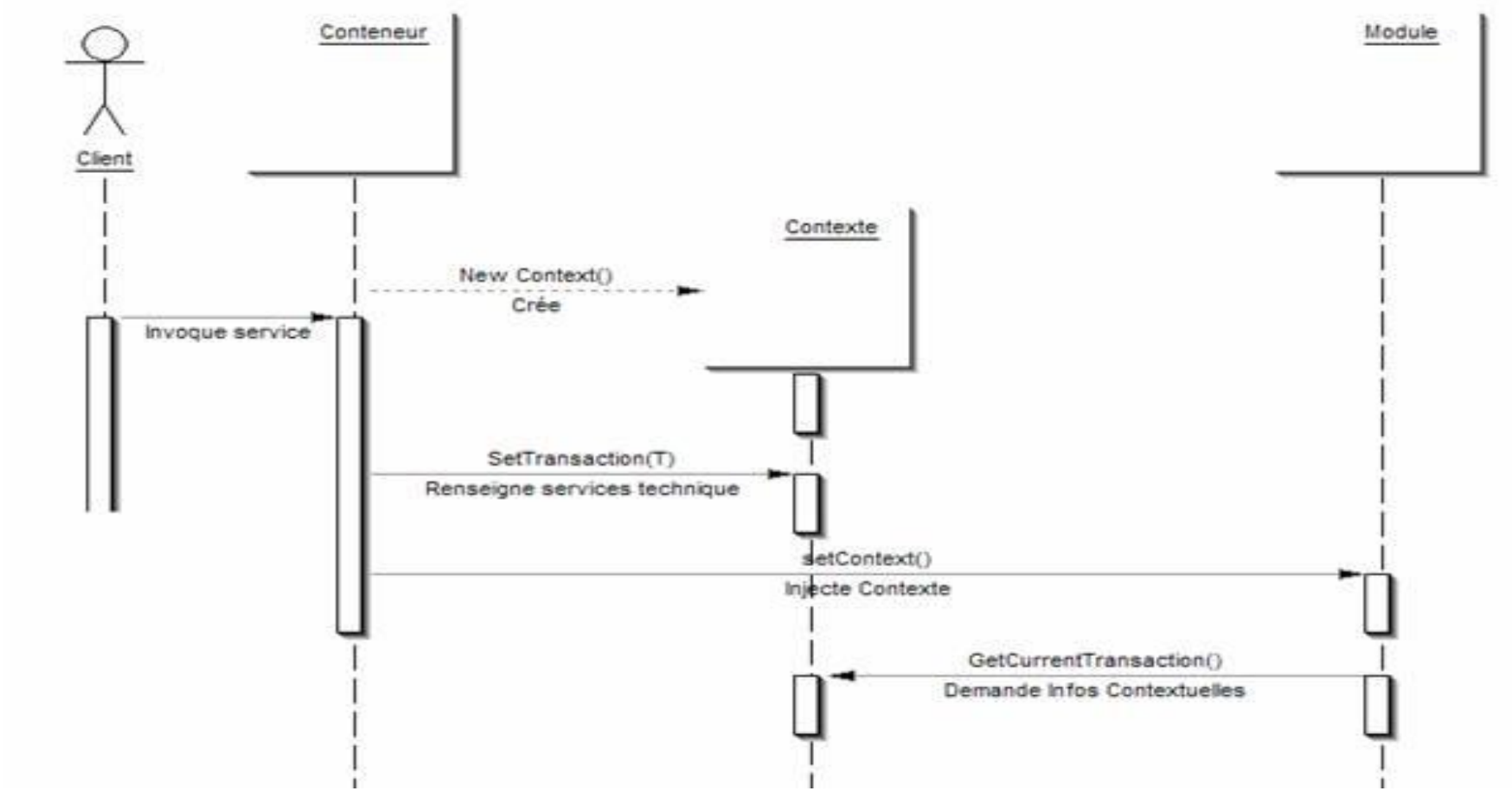
Le pattern Contexte

- Initialisation et service



Le pattern Contexte

Plan
- Rappels sur les patrons de conception
- > Contexte
- Annuaire
- JNDI



Source : http://www.dotnetguru.org/articles/articlets/contextPattern/Contexte.htm

Le pattern Contexte

- Exemple d'utilisation (ServletCont

```
@WebServlet(name = "Test", urlPatterns = {"/test"}
    @WebInitParam(name= "Site", value="http://univ
    @WebInitParam(name= "Author", value= "Medini")
public class Test extends HttpServlet {
 protected void doGet([…]) throws […]
  PrintWriter out = response.getWriter();
  ServletContext context = request.getServletConte
  try {
   out.println("Context path: " + context.getConte
   out.println("Server info: " + context.getServer
   out.println("Registered servlets");
   for (String servletName: context.getServletRegi
    ServletRegistration sr = context.getServletReg
    out.println(servletName + " : " + sr.getClassN
    for (String param: sr.getInitParameters().keyS
     out.println("Parameter:"+ paramName+ " value:
   }
  } finally {
   out.close();
  }
 }
}
```

Context path: /TestServletContext

Server info: Apache Tomcat/7.0.26

Registered servlets

Test : test.Test

Parameter: Site - value: http://univ-lyon1.fr/

Parameter: Author - value: Medini

jsp : org.apache.jasper.servlet.JspServlet

Parameter: fork - value: false

Parameter: xpoweredBy - value: false

Parameter: mappedfile - value: true

default : org.apache.catalina.servlets.DefaultServlet

Parameter: debug - value: 0

Parameter: listings - value: false

Le pattern Contexte

- Avantages
  - Isolation des composants
  - Contrôle des communications
  - Évite l'anti-pattern "ne pas parler aux inconnus"

    ```
    a.b.c.callMethod()
    ```
- Inconvénients
  - Rajoute un niveau d'indirection (de complexité)
  - Peut être en contradiction avec la loi de Demeter (least knowledge)
    - S'il est mal utilisé, beaucoup d'objets peuvent se retrouver à disposition de tous les composants

    ➔problèmes de sécurité et de performances

Le pattern Contexte

- Variantes
  - Encapsulated context
    - intégré au conteneur
    - applications “simples”
  - Ambiant context (aka. Scope, Context Object)
    - tenir compte du scope des objets (-> annuaire JNDI)
    - isolation des couches de l’application
  - Navigational context
    - orienté Web (HTML / JS)

Le pattern Annuaire

- Contexte
  - Le conteneur + le pattern contexte isolent les composants
  - Le conteneur fait office de serveur (les composants peuvent recevoir des requêtes)
- Problème
  - Permettre l'accès aux composants dans un conteneur
  - Conserver un faible couplage entre les composants

Le pattern Annuaire

- Principe
  - Rajouter un niveau d'indirection qui
    - centralise les références sur les composants accessibles
    - stocke et renvoie les références (en général sous forme de paires clés-valeurs)
  - Éventuellement, hiérarchiser les références pour permettre différents types d'accès
    - En fonction du scope des composants
- Autres noms
  - Registre (*registry*)
  - Serveur de noms (*Naming Directory*)

Le pattern Annuaire

- Fonctionnement
  - À l'instanciation d'un composant
    - Le serveur enregistre le composant dans l'annuaire
    - Le composant attend les requêtes
  - À la première requête du client
    - Le client interroge l'annuaire pour récupérer une référence sur l'objet enregistré
    - Le client effectue un cast de cette référence en fonction du type de l'objet
    - Le client peut appeler les méthodes de l'objet

Le pattern Annuaire

- Exemple d'utilisation (binding d'un stub en RMI)

```
public class ComputeEngine implements Compute {
  public <T> T executeTask(Task<T> t) {
    return t.execute();
  }

  public static void main(String[] args) {
    if (System.getSecurityManager() == null) {
      System.setSecurityManager(new SecurityManager());
    }
    try {
      String name = "Compute";
      Compute engine = new ComputeEngine();
      Compute stub = (Compute) UnicastRemoteObject.exportObject(engine, 0);
      Registry registry = LocateRegistry.getRegistry();
      registry.rebind(name, stub);
      System.out.println("ComputeEngine bound");
    } catch (Exception e) {
      System.err.println("ComputeEngine exception:");
      e.printStackTrace();
    }
  }
}
```

Source : http://docs.oracle.com/javase/tutorial/rmi/implementing.html

Le pattern Annuaire

- Exemple d'utilisation (récupération d'une référence par le client)

```
public class ComputePi {
    public static void main(String args[]) {
        if (System.getSecurityManager() == null) {
            System.setSecurityManager(new SecurityManager());
        }
        try {
            String name = "Compute";
            Registry registry = LocateRegistry.getRegistry(args[0]);
            Compute comp = (Compute) registry.lookup(name);
            Pi task = new Pi(Integer.parseInt(args[1]));
            BigDecimal pi = comp.executeTask(task);
            System.out.println(pi);
        } catch (Exception e) {
            System.err.println("ComputePi exception:");
            e.printStackTrace();
        }
    }
}
```

Source : http://docs.oracle.com/javase/tutorial/rmi/client.html

Le pattern Annuaire

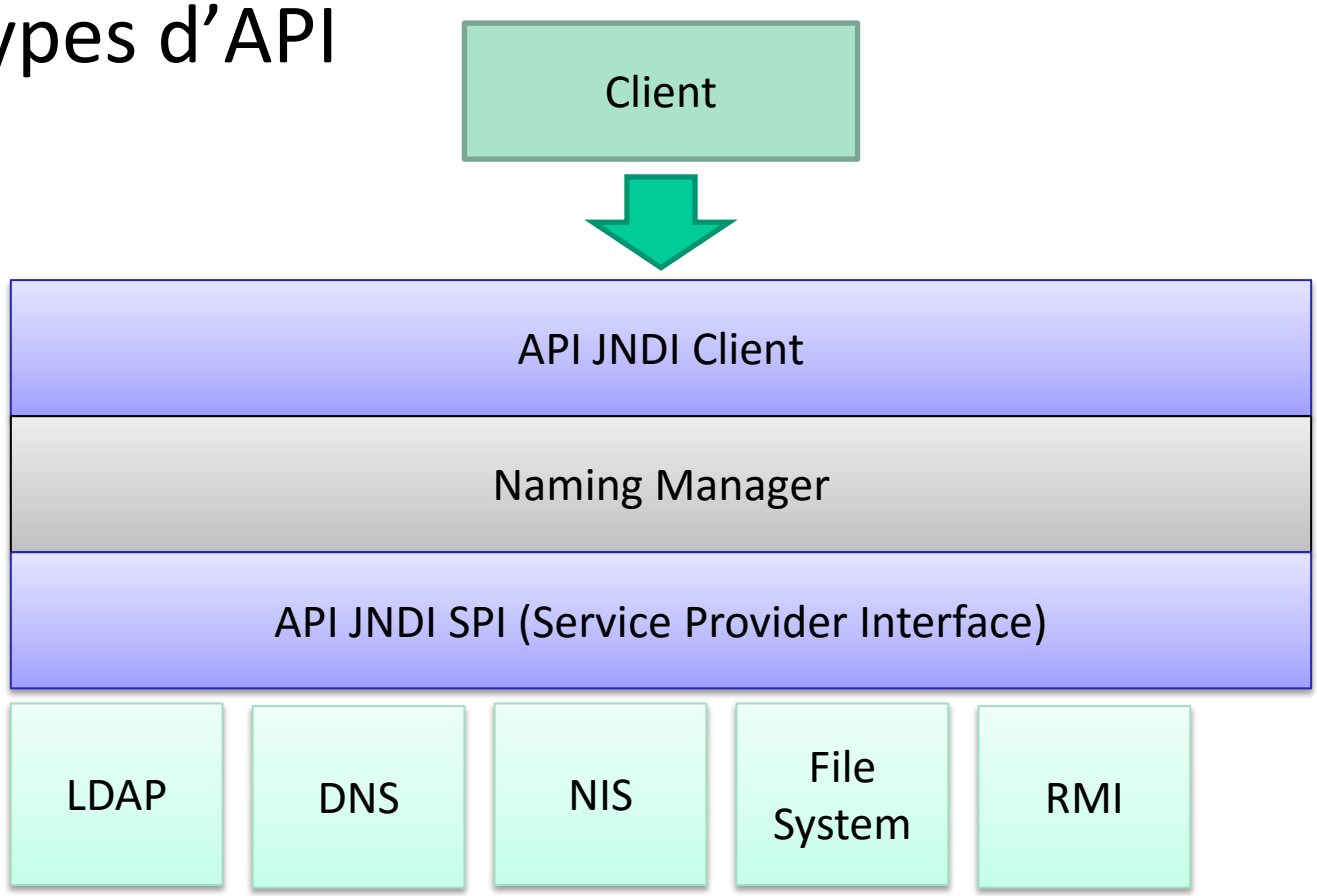
- Remarques
  - Pour favoriser le faible couplage, le client doit connaître l'interface de l'objet et non son implémentation
  - Le conteneur peut avoir la responsabilité de l'enregistrement d'un composant dans l'annuaire
  - Un même objet peut servir à la fois de contexte et d'annuaire
- Exemples (Java SE)
  - RMI : java.rmi.registry (nommage "à plat")
  - JNDI : javax.naming (serveur de noms, hiérarchique)

Java Naming Directory Interface

- API Java de référence pour
  - L'accès à des objets distants (annuaire distribué)
  - La séparation des comportements applicatifs (hiérarchie de contextes)
- Caractéristiques
  - Intégré à Java SE depuis le JDK 1.3
  - Standard utilisé pour de nombreuses technologies
    - LDAP, DNS, RMI, EJB…
  - Packages : `javax.naming.*`

Java Naming Directory Interface

- 2 types d'API



Java Naming Directory Interface

- API « client »
  - Utilisation d'un annuaire indépendamment du SPI
    - Accéder à un contexte applicatif initial (lié à l'application)
    - Enregistrer un objet (serveur)
    - Explorer/modifier la hiérarchie de contextes (client & serveur)
    - Retrouver un objet (client)
  - Paramétrage
    - Fichier de configuration : jndi.properties
    - Programmatique

Java Naming Directory Interface

- API « client »
  - Binding
    - Opérations
      - bind(), rebind(), unbind(), rename()
    - Remarque
      - En pratique, il est plus sûr d'utiliser la méthode rebind()
  - Recherche
    - Opération
      - lookup()
    - Remarque
      - Nécessite de connaître le contexte initial de l'objet cherché

Java Naming Directory Interface

- API « client »
  - Exploration
    - Opérations
      - Context.list(javax.naming.Name) : renvoie un NameClassPair (nom « bindé » et nom de la classe)
      - Context.listBindings(javax.naming.Name) : renvoie une énumération de Binding
  - Modification de l'arborescence
    - Opérations
      - createSubcontext(), destroySubcontext()
    - Remarque
      - Revient à créer/supprimer un répertoire

Java Naming Directory Interface

- API « client » : exemples
  - Binding

```
import javax.naming.*;
  public void registerFruit() throws NamingException {
    //Get the context object
    Context context = new InitialContext();

    // Create the object to be bound
    Fruit fruit = new Fruit("orange");
    // Perform the bind
    context.bind("favorite", fruit);

    // Bind a new object
    Fruit fruit2 = new Fruit("orange");
    context.rebind("favorite", fruit2);
}
```

Java Naming Directory Interface

- API « client » : exemples
  - Recherche d'objet

```
import javax.naming.*;
  public Fruit findFruit() throws NamingException {
    Context context = new InitialContext();
    return (Fruit) context.lookup("favorite");
  }
```

    - Remarque
      - Le contexte initial pour Java EE est « java:comp/env »
      - Par défaut, les EJB sont enregistrés dans le contexte « java:comp/env/ejb »

Java Naming Directory Interface

- API « client » : configuration
  - Par fichier de configuration

```
java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory
java.naming.provider.url=ldap://monserveur:389
...
```

    - Remarque : d'autres paramètres peuvent être nécessaires en fonction du type du service d'annuaire
  - Par programmation

```
Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY,
    "com.sun.jndi.fscontext.RefFSContextFactory");
env.put(Context.PROVIDER_URL, "file:/tmp/tutorial");
Context ctx = new InitialContext(env);
```

Java Naming Directory Interface

- Nommage
  - Il existe deux types de classes utilisées pour gérer les URL de nommage
    - javax.naming.CompoundName : URLs « classiquement » séparées par des slashs
    - javax.naming.CompositeName : URLs dont la structure est définie par une classe spécifique
      - Dans ce cas, il faut fournir le nom du package dans la configuration

Source : http://docs.oracle.com/javase/7/docs/technotes/guides/jndi/spec/spi/jndispi.fm.html

Java Naming Directory Interface

- API SPI
  - Permet d'implémenter un support qui s'interface avec l'annuaire
    - Context
    - ContextFactory
    - …
  - Nécessite l'utilisation d'un type d'URL pour l'accès aux contextes
    - Java Naming Provider : jnp://monserveur
    - Peut nécessiter un paramétrage : java.naming.factory.url.pkgs
  - Pour l'utilisation du service, il « suffit » de configurer le client pour qu'il accède à la factory et à l'URL du contexte initial

Source : http://docs.oracle.com/javase/7/docs/technotes/guides/jndi/spec/spi/jndispi.fm.html

Références

- Contexte
  - Ouvrage
    - Deepak Alur, Dan Malks, John Crupi (2003), Core J2EE Patterns: Best Practices and Design Strategies, Prentice Hall, ISBN : 978-0131422469
  - Sites
    - http://stackoverflow.com/questions/986865/can-you-explain-the-context-design-pattern-a-bit
    - http://www.allankelly.net/patterns/encapsulatedcontext.html
    - http://www.cs.wustl.edu/~schmidt/PDF/Context-Object-Pattern.pdf
    - http://www.cs.unibo.it/fabio/VD99/lima/lima.html
    - http://en.wikipedia.org/wiki/Law_Of_Demeter
    - http://www.dotnetguru.org/articles/articlets/contextPattern/Contexte.htm

Références

- Annuaire

  http://martinfowler.com/eaaCatalog/registry.html

  http://pagesperso-systeme.lip6.fr/Gael.Thomas/cours/mdoc/cours/mdoc4.1-jndi.pdf

- JNDI

  http://www.jmdoudoux.fr/java/dej/chap-jndi.htm

  http://docs.oracle.com/javase/7/docs/api/javax/naming/package-summary.html

  http://docs.oracle.com/javase/jndi/tutorial/basics/naming/index.html

  http://www.javaworld.com/javaworld/jw-04-2002/jw-0419-jndi.html

Références

- JNDI SPI

  http://docs.oracle.com/javase/7/docs/technotes/guides/jndi/spec/spi/jndispi.fm.html

- JNDI JNP

  http://stackoverflow.com/questions/192718/jndi-without-a-j2ee-container-with-jnp-maybe-some-other-provider

  http://mail-archive.ow2.org/jonas-team/2002-01/msg00063.html