

SERVICES WEB & WEB DES SERVICES

Francis.Lapique@epfl.ch, e-pfl

INTRODUCTION

Si la mutualisation des *services Web* au niveau inter-entreprises ou plus généralement dans le Web des services est encore musique d'avenir, au niveau de l'entreprise elle est d'ores et déjà en marche. Pour quelle raison? Parce que la maîtrise des coûts liés aux problèmes de l'interopérabilité des services métiers passe par l'affranchissement de l'intégration d'environnements *middleware* propriétaires. C'est un tournant important que l'EPFL comme toute entreprise devra négocier au mieux dans le cadre de sa politique de systèmes d'information.

Les approches techniques pour aborder l'intégration des services sont pléthoriques. Sur le marché de l'*Open Source*, j'ai décidé d'explorer trois démarches:

- SOAP (*Simple Object Access Protocol*), une spécification de communication entre services Web sur une base d'échange de messages XML
- JXTA, un réseau pair-à-pair (*Peer-to-Peer*)
- Jahia, un serveur d'applications sur la plate-forme J2EE.

Qui dit services dit clients de ces services. C'est le point de vue de cet article. Le côté serveur, par ses aspects

trop spécifiques, n'est pas abordé ou très peu. Pour rendre l'exercice le plus didactique possible, tous les codes sources sont en Java pour les programmeurs qui voudraient expérimenter ce que je décris. Les trois chapitres SOAP, JXTA et Jahia sont indépendants.

Avant de rentrer dans le vif du sujet, quelques mots tout de même du service.

LE SERVICE

Le service **exemple** qui va nous accompagner tout au long de cet article n'est pas un service d'achats, de réservation ni de recherche de vie extra-terrestre mais celui d'un automate à l'affût des dernières nouvelles d'un certain nombre de sujets à caractères scientifiques. Sans rentrer dans trop de détails techniques, qui dépasseraient le cadre de cet article, on dirige l'automate sur un ensemble de sites, qui va catégoriser des documents sur la base d'un modèle d'apprentissage. Les méthodes d'apprentissage sur des données textuelles sont nombreuses et les limites de chacune bien connues. Pour ceux que cela intéresse voici quelques mots-clés pour découvrir une littérature,

SUITE EN PAGE 8

SOMMAIRE FI 8/2002

Services Web & Web des services <i>Francis Lapique</i>	1
Libre-service ne signifie pas abus <i>Jacques Virchaux</i>	2
A quoi sert XSL? <i>Guy de Pourtalès</i>	3
Les enjeux du référencement des locaux <i>Philippe Pichon</i>	7
Programme des cours	15
Flash sur Flash ... dans le Flash ! <i>Hicham Dennaoui</i>	19
Calendrier	20

PROCHAINES PARUTIONS

	délaï	rédaçtion	paraçtion
9	31.10.02		19.11.02
10	28.11.02		17.12.02

comme *naïve Bayes*, *arbres de décision*, *réseaux de neurones (PMC)*, *machines à vecteurs supports (SVM)*, *modèles de Markov cachés (MMC)*, *apprentissage relationnel*. Notre service fait appel à un classifieur SVM. Expliqué de façon simple, imaginez un tas de ronds et de losanges dans un espace à 2 dimensions. Le classifieur va séparer au mieux ces deux tas. Imaginons que seuls les ronds vous intéressent, étant donné un candidat carré, le classifieur va apprécier la distance qui sépare ce candidat des ronds. Avec des documents, on fait la même chose en les plongeant dans un espace de grande dimension (sous-ensemble des termes quelque peu modifiés des documents). A titre indicatif voici quelques-uns des sites qui sont visités plusieurs fois par jour, depuis quelques mois. Toutes les pages visitées sont gardées en mémoire ce qui permet d'identifier les changements d'un passage à l'autre:

www.aldaily.com/
www.scitechdaily.com/
abc.net.au/science/news/default.htm
www.nature.com/nsu/
www.nature.com/nature/insights/index.html
www.scicentral.com/
www.sciencedaily.com/news/headlines.htm
www.nas.edu/headlines
www.nationalacademies.org/topnews/
www.sciencenews.org/index.asp
exn.ca/ScienceNews/index.asp
www.sciam.com/news_directory.cfm
www.space.com/news/index.html
www.wired.com/news/nc_index.html
worldscientist.com/
www.closetotruth.com

LE CLIENT SOAP

SOAP est un protocole de transmission de messages XML. On peut l'utiliser dans le cadre de simples transmissions unidirectionnelles comme dans celui de dialogues de type requête-réponse. Bien que mettant en avant HTTP, SOAP n'est pas lié à un protocole de transport particulier. Il est également indépendant du système d'exploitation et du langage de programmation.

La syntaxe d'un message SOAP est décrite en détail dans un document du W3C (www.w3.org/TR/SOAP/).

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://www.w3c.org/..."
xmlns:xsi="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Header></SOAP-ENV:Header>
<SOAP-ENV:Body>
  <ns1:GetNews
    xmlns:ns1="urn:BasicFilteringService">
    <param1 xsi:type="xsd:int">123</param1>
  </ns1:uneMethode>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Un message SOAP est constitué d'une enveloppe contenant un en-tête (*header*) facultatif et le corps du message (*body*). La plupart des packages SOAP que vous utiliserez prendront en charge les détails de syntaxe des messages SOAP, mais pour se fixer les idées, regardons un message de requête dans le cadre d'un appel de service:

Le prologue XML contient seulement une déclaration XML `<?xml version="1.0" encoding="UTF-8" ?>` spécifiant la version de XML et l'encodage des caractères du message XML.

L'enveloppe SOAP est marquée par la balise `<SOAP-ENV:Envelope ... >`. Elle spécifie que le style d'encodage de ce message SOAP suit le schéma défini dans <http://schemas.xmlsoap.org/soap/encoding/>. L'élément en-tête est optionnel. Vient ensuite le corps désigné par la balise `<SOAP-ENV:Body>` qui contient la méthode à appeler `<ns1:getNews... >` accompagné de son namespace `urn:BasicFilteringService`. Cette balise encapsule à son tour un paramètre `<param1 ... >`. Les paramètres n'ont pas de namespace.

Property	Detail
ID	urn:BasicFilteringService
Scope	Application
Provider Type	java
Provider Class	libering.BasicFilteringService
Use Static Class	false
Methods	executeQuery,getItems,getCategories
Type Mapping	
Detail Mapping Registry Class	

figure 1

Vous ne pouvez pas utiliser un service sans avoir son descripteur de déploiement. J'ai choisi d'enregistrer le service dans un environnement Apache SOAP. Apache SOAP (<http://xml.apache.org/soap/index.html>) est une implémentation d'un service SOAP en Java de l'Apache Software Foundation. La figure 1 vous donne le résultat de ce déploiement. Vous y trouvez son nom, plus précisément l'URN du service *urn:BasicFilteringService*, le nom du package et les méthodes qui sont au nombre de trois:

- *GetCategories* pour obtenir la liste des catégories.
- *GetNews* pour obtenir les nouvelles des douze dernières heures pour une catégorie ou pour l'ensemble.
- *ExecuteQuery* pour faire une recherche dans une catégorie ou sur l'ensemble.

Nous avons toutes les informations nécessaires pour nous lancer dans l'écriture de notre premier client SOAP (*SoapClient1.java*). Le but fixé est la recherche des nouvelles du jour dans la catégorie physics. Le programme commence par créer une instance de `java.net.URL` sur le service de routage *rpcrouter* (<http://soap1.epfl.ch:8080/soap> est l'url du service SOAP sur *Tomcat*). La méthode *setTargetObjectURI()* appliquée à un objet *org.apache.soap.rpc.Call* fixe l'URI du service et *setMethodName()* le nom de la méthode qui à la lumière du descriptif est *getNews*:

```
URL url = new URL("http://soap1.epfl.ch:8080/
                 soap/servlet/rpcrouter");
Call call = new Call();
call.setTargetObjectURI
    ("urn:BasicFilteringService");
call.setMethodName("getNews");
```

Le code concernant le passage, plus exactement la sérialisation des paramètres à envoyer se fait de la façon suivante:

```
call.setEncodingStyleURI(Constants.NS_URI_SOAP_ENC);
Object[] multiParams = { categorie };
Vector params = new Vector();
params.addElement(new Parameter("params",
    Object[].class, multiParams, null));
call.setParams(params);
```

On commence par préciser le style d'encodage (ici *Constants.NS_URI_SOAP_ENC* équivalent à <http://schemas.xmlsoap.org/soap/encoding>), puis on range les paramètres (dans cet exemple un seul, le nom de la catégorie) dans un vecteur *params* et on les associe à l'objet *call*.

En invoquant la méthode *invoke()* sur l'objet *call* on capture la réponse du service. Sur cette instance *Response* on applique la méthode *getReturnValue()* pour obtenir une instance de *Parameter*. On récupère la valeur du paramètre de retour par *getValue()*:

```
try {
    call.setMethodName("getNews");
    Response resp = call.invoke(url, "");
    Parameter ret = resp.getReturnValue();
    Object value = ret.getValue();
    System.out.println("Réponses: " + value);
} catch (SOAPException e) {
    System.err.println("Caught SOAPException (" +
        e.getFaultCode() + "): " +
        e.getMessage());
}
```

En ajoutant les imports

```
import java.net.*;
import java.util.*;
import org.apache.soap.*;
import org.apache.soap.rpc.*;
```

et le morceau de code pour saisir la catégorie, vous pouvez passer à la compilation et au test:

```
java SoapClient1 physics
```

```
Réponses: [physics]<a href="http://www.first
science.com/site/articles/blackholes.asp">First
Science.com - Black Holes and Time Machines</a>
```

La réponse se présente sous la forme suivante: le nom de la catégorie entre crochets, une suite de liens (au format HTML) retenus par l'automate qui sont identifiés par leur titre.

Modifions légèrement *SoapClient1.java* pour exploiter la méthode *executeQuery*. Dans ce cas nous devons sérialiser deux paramètres qui sont le nom de la catégorie et la chaîne de recherche:

```
Object[] multiParams = { name, request };
```

et invoquer un nouveau service *executeQuery*:

```
call.setMethodName("executeQuery");
```

La requête *Quelles sont les références à des articles retenus par l'automate sur l'ensemble des catégories où les mots human et francis sont présents?* s'exprime de la manière suivante:

```
java SoapClient2 "human francis"
```

```
Réponses: [biology]<a href="http://www.nhgri.
nih.gov/NEWS/sequencing.html">NHGRI Prioritizes
Next Organisms to Sequence</a>[biology]<a href=
"http://www.sciencedaily.com/releases/2002/05/
020508073140.htm">ScienceDaily Magazine -
International Team Of Researchers Assembles
Draft Sequence Of Mouse Genome</a>[biology]
<a href="http://www.newscientist.com/news/news.
jsp?id=ns99992253">New Scientist</a>
[consciousness]<a href="http://cogprints.soton.
ac.uk/view-evol-psy.html">Cogprints - Subject:
Evolutionary Psychology</a>.....
```

À la date d'aujourd'hui il existe une bonne douzaine d'implémentations de SOAP. La question de l'interopérabilité est ouverte. Comment un client Apache SOAP discute-t-il avec un service sur .NET ou GLUE ? Il existe une tentative de proposer une grammaire XML pour décrire ces services Web. Cette proposition soutenue par Ariba, IBM et Microsoft auprès du W3C s'appelle WSDL (Web Services Description Language). WSDL va dans la bonne direction, mais la génération des URIs qui ne sont pas globalement uniques dans le temps indique que le travail sur WSDL n'est pas tout à fait abouti.

Signalons le **Java Web Services Developer Pack** (WSDP) qui va dans cette direction, il est téléchargeable (32 Mo) directement depuis le site de Sun. Il contient un ensemble d'APIs et d'outils qui étendent la plate-forme Java 2 pour lui permettre de créer, de déployer et d'exécuter des services Web conformes aux spécifications SOAP, WSDL et UDDI.

L'image suivante est tirée de la collection O'Reilly: *Java & XML*, elle résume bien l'esprit d'un service Web dans la culture SOAP.

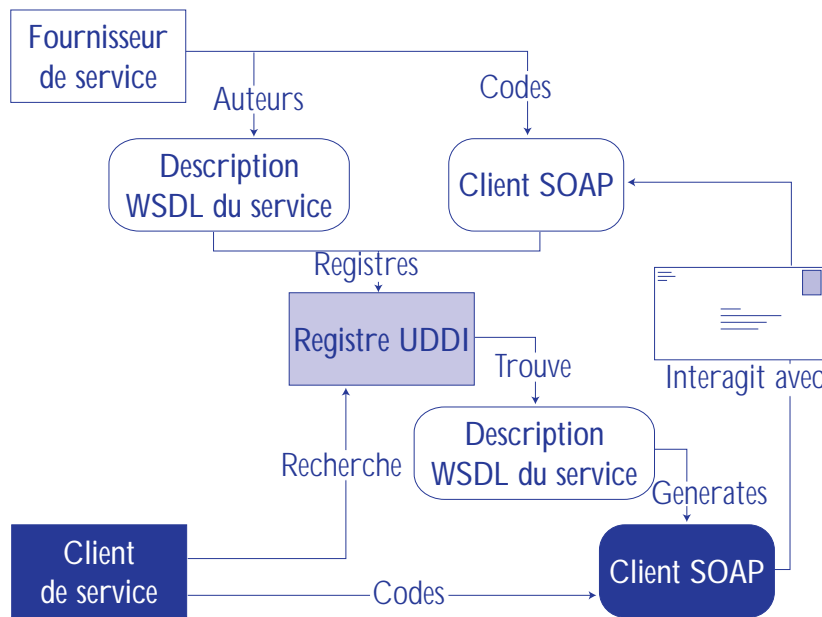


figure 2

Sans aller plus loin avec SOAP, passons au réseau pair-à-pair.

P2P

introduction

Un constat s'impose. L'exploitation de ressources massivement distribuées est aujourd'hui une préoccupation majeure qui va au-delà des services Web. Suivant les centres d'intérêts on distingue plusieurs approches qui vont toutes dans ce sens:

- Calcul global (Global Computing) qui consiste à récupérer, à l'image du projet SETI@home, RSA-155 ou Decrypton, des cycles de nœuds de calcul distribués sur le réseau.

- Calcul sur grille (Grid Computing) qui consiste à conduire, à l'image de Globus (www.globus.org), de très gros calcul de modélisation ou de simulation en utilisant les ressources de grands centres de calcul distribués dans le monde.
- Calcul pair-à-pair (Peer-to-Peer, P2P) qui consiste à établir une interconnexion d'égal à égal ou de pair à pair, entre plusieurs ordinateurs. Il complète le modèle classique client/serveur, en symétrisant la relation des nœuds qui interagissent. La relation client-serveur n'est plus associée aux nœuds mais aux transactions: chaque nœud peut être client dans une transaction et serveur dans une autre (figure 3).

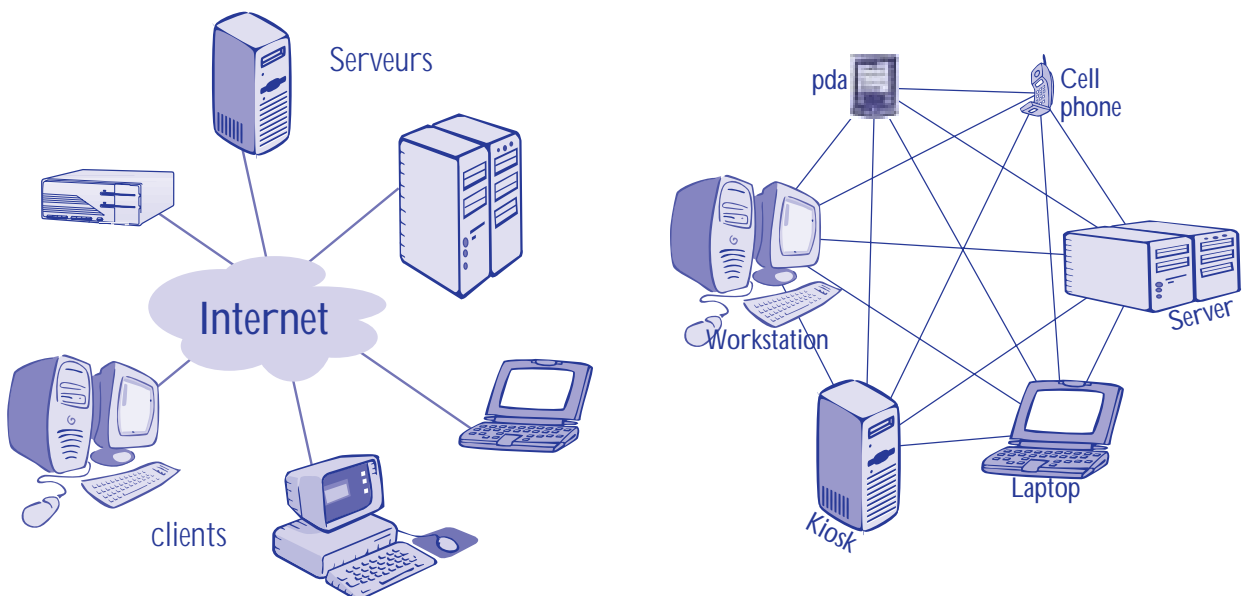


Figure 3

Ce constat fait, nous allons, à l'image de l'exercice précédent avec SOAP, choisir une implémentation dans le cadre d'un réseau pair-à-pair. Le choix s'est porté sur JXTA. Nous allons commencer par introduire les mécanismes de base de JXTA puis passer au codage du client du service *BasicFilteringService*.

Pour celles et ceux qui voudraient aller plus loin avec JXTA, je vous conseille l'excellent livre de Brendon Wilson dont vous pouvez télécharger gratuitement le fichier PDF du Web ou bien l'acheter sous forme papier (<http://www.brendonwilson.com>).

JXTA, une infrastructure pour des services pair-à-pair

Le projet JXTA (<http://www.jxta.org>) propose des concepts et mécanismes de base génériques propres aux réseaux pair-à-pair. Initié par Sun dans une approche Open Source ce projet identifie plusieurs protocoles permettant de gérer un certain nombre de fonctionnalités communes comme la gestion des pairs, la découverte des ressources, l'invocation de services, la communication inter-pairs, la sécurité, etc. Ces protocoles servent de briques de base pour la mise en œuvre de services pair-à-pair.

CONCEPTS DE BASE

Pair

Un pair est l'unité structurelle de base de tout réseau P2P. Un pair peut être toute ressource capable de se connecter au réseau et de fournir un service: poste de travail, grappe de machines ou même un assistant personnel électronique de type PDA. JXTA distingue plusieurs types de pairs:

Pairs minimaux

Ces pairs sont réduits à des fonctionnalités élémentaires de communication: envoi et réception de messages. Ils ne sont capables ni de stocker de l'information ni de *router* les messages.

Pairs simples

Ces pairs peuvent envoyer et recevoir des messages et en plus stocker dans un cache local des informations sur les ressources du réseau sous forme d'annonces. Ces annonces leur permettent de répondre à des requêtes d'informations et de participer ainsi au protocole de découverte de ressources.

Pairs de rendez-vous

Ces pairs sont essentiels à la constitution du réseau. En plus des fonctionnalités décrites plus haut ces pairs permettent de faire suivre les requêtes vers d'autres pairs connus, pairs simples ou pairs de rendez-vous.

Pairs de routage

Ces pairs stockent des informations de routage vers les autres pairs. En plus ils permettent de transmettre les requêtes des pairs isolés du réseau par des mécanismes de pare-feu.

Groupe de pairs

Les groupes de pairs correspondent à des ensembles de pairs réunis par un intérêt commun: une application de collaboration, un ensemble de services ou un niveau de sécurité.

Service

Les services sont des traitements fournis au sein d'un réseau P2P (recherche, partage, etc.). Ce sont eux qui motivent la constitution du réseau. On distingue deux types de services, des services individuels fournis par des pairs individuels et des services de groupe fournis par un groupe à tous ses membres. Plusieurs membres du groupe peuvent contribuer, de manière redondante, à ce dernier type de service qui reste disponible tant qu'au moins un membre du groupe reste connecté.

Annonce

Une annonce est une représentation structurée d'une entité présente sur le réseau. En particulier, toutes les entités décrites jusqu'ici (pair, groupe de pairs, service) ou toute autre ressource peut être décrite par une annonce.

Protocoles de base

Plusieurs protocoles définissent la structure des échanges d'informations pour un ensemble d'interactions fréquentes dans un réseau P2P. JXTA définit 6 protocoles que nous décrivons dans la section suivante.

Protocoles JXTA

Les protocoles JXTA sont définis sous forme de séries de messages XML et correspondent aux principaux types d'interaction présents dans un réseau P2P. Tous les protocoles sont asynchrones et reposent sur un modèle requête/réponse. Typiquement un pair envoie une requête à un ou plusieurs pairs de son groupe et peut recevoir zéro, une ou plusieurs réponses. Il n'est pas requis que tous les pairs implémentent tous les protocoles.

Protocole de découverte

Ce protocole PDP (Peer Discovery Protocol) est utilisé par les pairs pour faire connaître leurs propres ressources et pour découvrir les ressources fournies par les autres pairs.

Protocole d'information

Ce protocole PIP (Peer Information Protocol) permet d'obtenir des informations sur les pairs (état, trafic, etc.).

Protocole d'invocation de service

Ce protocole PRP (Peer Resolver Protocol) permet de transmettre des requêtes génériques sur le réseau. Les requêtes peuvent être adressées à un pair spécifique ou bien peuvent être propagées à tous les membres d'un groupe via les pairs de rendez-vous.

Protocole de communication par canaux

Le protocole PBP (Pipe Binding Protocol) permet d'établir des canaux virtuels de communication pipes entre un ou plusieurs pairs.

Protocole de routage

Ce protocole ERP (Endpoint Routing Protocol) de plus bas niveau est utilisé par les pairs pour déterminer la route d'accès à une destination donnée.

Protocole de rendez-vous

Le protocole RVP (Rendez-vous Protocol) sert à propager des messages au sein d'un groupe de pairs. Il permet aux pairs de se connecter au service et contrôle la propagation des messages.

Ces six protocoles offrent une couche *middleware* pour l'implémentation de services P2P.

CODAGE DU CLIENT

Qu'il s'agisse de SOAP ou de JXTA, le client doit s'attacher un service, formuler une requête sous la forme d'un message ou d'une annonce XML, attendre la réponse. A la lumière de ce qui précède le protocole PRP d'invocation de service répond exactement à notre besoin. Le protocole PRP prend en charge l'envoi par un pair d'une annonce de type Resolver Query Message à un *handler* qui se trouve quelque part sur le réseau. Ce *handler* va traiter cette requête et renvoyer une réponse de type Resolver Response Message.

L'interface QueryHandler est le mécanisme proposé par JXTA pour écrire son propre *handler*. Comme ceci concerne le service, nous ne le détaillerons pas explicitement ici mais sachez que cela se résume à écrire la classe

```
class BasicFilteringService implements
                                QueryHandler
```

J'ai extrait 3 lignes de ce code pour illustrer les trois étapes du service:

```
// Parse the message from the query string.
eq = new MyQueryMsg(new ByteArrayInputStream(
    (query.getQuery()).getBytes());
// Perform the Query
    answer = executeQuery(eq.getCat(),
        eq.getReq());

// Create the response message.
MyResponseMsg er = new MyResponseMsg(answer);
response = new ResolverResponse
    ("BasicFilteringService",
    null, query.getQueryId(), er.toString());

return response;
}
```

Le corps du client se ramène à la création d'une annonce de type Resolver Query Message,

```
MyQueryMsg equery = new MyQueryMsg (categorie,
                                    requete);
```

de récupérer l'identifiant du pair,

```
String localPeerId = currentGroup.getPeerID().
    toString();
```

d'invoquer le service

```
ResolverQuery query = new ResolverQuery("Basic
    FilteringService",
    null, localPeerId, equery.toString(), 0);
```

toString() est une méthode de la classe TheQueryMsg

```
public String toString()
{
    try
    {
        StringWriter out = new StringWriter();
```

```
StructuredTextDocument doc =
    (StructuredTextDocument)
    getDocument(new MimeMediaType("text/xml"));
doc.sendToWriter(out);

return out.toString();
}
catch (Exception e)
{
    return "";
}
}
```

et d'envoyer l'annonce à l'ensemble des pairs:

```
resolver.sendQuery(null, query);
```

Comment tester tout ceci ? Installer JXTA sur votre machine (ça prend 1 minute). Lancer le Shell JXTA:

- sous Windows Start->Programs->JXTA->JXTAShell,
- sous linux allez dans le répertoire *shell* et exécutez *run.sh*.

Si tout est en ordre une fenêtre du genre de celle de la figure 4 apparaît. Donnez un nom à votre pair, faites ok et rentrez un *user name* et *password*. Une autre fenêtre apparaît, c'est celle du Shell JXTA (figure 4), lancer la commande *exit*.

Téléchargez l'exemple *jxta_demo.jar* (<http://...>), installez les classes à l'endroit où vous avez lancé la commande *run.sh* et n'oubliez pas de modifier le CLASSPATH de *run.sh* en conséquence. Vous venez d'ajouter une nouvelle commande au Shell JXTA intitulé *ClientJXTA*. Lancer de nou-

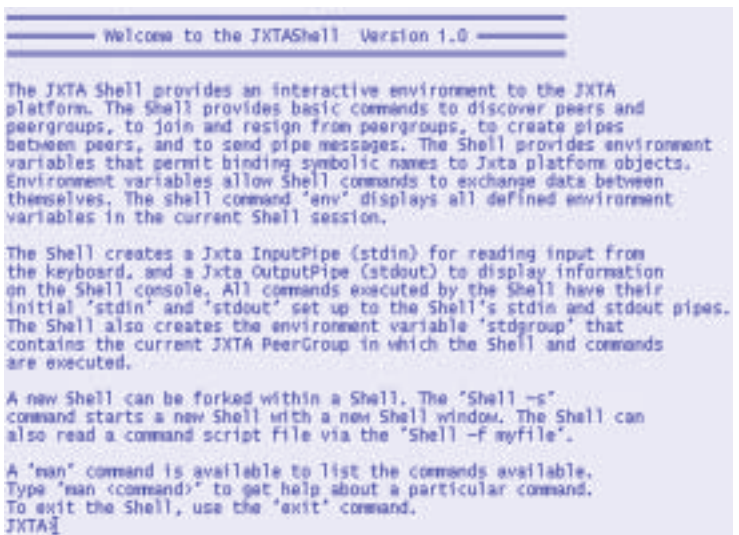
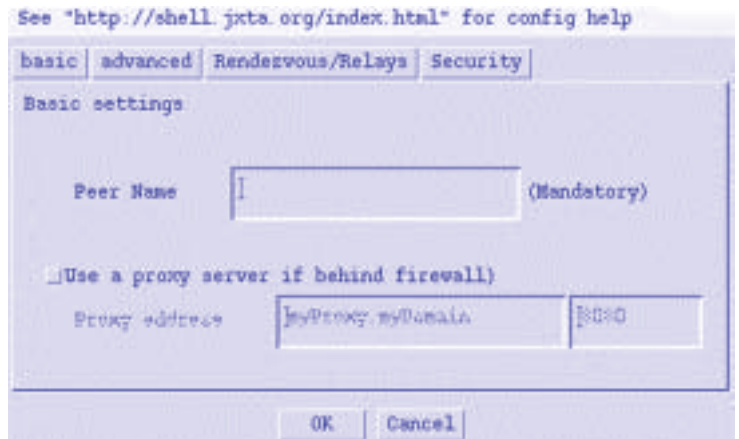


figure 4

veau *run.sh*, *user* et *password* rentrés auparavant , passez la commande *man* , la nouvelle commande *ClientJXTA* doit être dans la liste des commandes. Si ce n'est pas le cas vous avez un problème avec votre CLASSPATH. Si c'est bon lancer la commande *ClientJXTA -c biology -q adn* .

Cette commande va lancer une requête pour le service *BasicFilteringService* qui se trouve quelque part sur le réseau P2P (sur le pair *f101*) et renvoyer dans une fenêtre console une réponse:

```
Processing response...
Searched the category biology for adn:
[biology]<a href="http://human-nature.com/nibbs/02/caldararo.html">Ancient DNA and Human origins: The role of gene sequence variation in the species concept by Niccolo Caldararo</a>
```

Si vous n'avez aucune réponse, vérifiez ma présence sur le réseau P2P (pair *f101*), avec la commande *peers*. En cas d'absence faites moi parvenir un mail pour que j'inscrive le service (fig. 4).

Si vous comparez la figure 5 tirée de l'ouvrage de Brendon Wilson, à celle du monde SOAP vous avez une bonne idée des approches différentes d'un *peer* et SOAP services.

Jahia

Jahia est un serveur d'applications pour la gestion de contenu Web (CMS). En termes moins techniques c'est un environnement de développement collaboratif pour créer et gérer des sites dynamiques. Jahia est un *framework* 100% Java sur une plate-forme J2EE (figure 6) . Les applicatifs Portal Server, CMS et Admin Center s'appuient sur un ensemble de spécifications (méthodes et données) de la couche Jahia Foundation Server.

Le Portal Server prend en charge l'intégration d'applications et services Web , le CMS grâce à l'utilisation de composants de présentation (templates) facilite la création et la mise à jour de sites et l'Admin Center offre un certain nombre d'outils de gestion (utilisateurs, templates, applications).

L'écriture du service se résume à celle d'une *servlet* dans laquelle on définit par la méthode *doGet()*.

```
public class UnService extends
HttpServlet
{
    public void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
    {
        .....
    }
}
```

En tant qu'administrateur d'un site Jahia, vous pouvez enregistrer un nouveau service ou composant au moyen du menu *Manage components* (figure 7). C'est ce que j'ai fait pour notre service *BasicFilteringService* dans le cadre du site *democms.epfl.ch/test*.

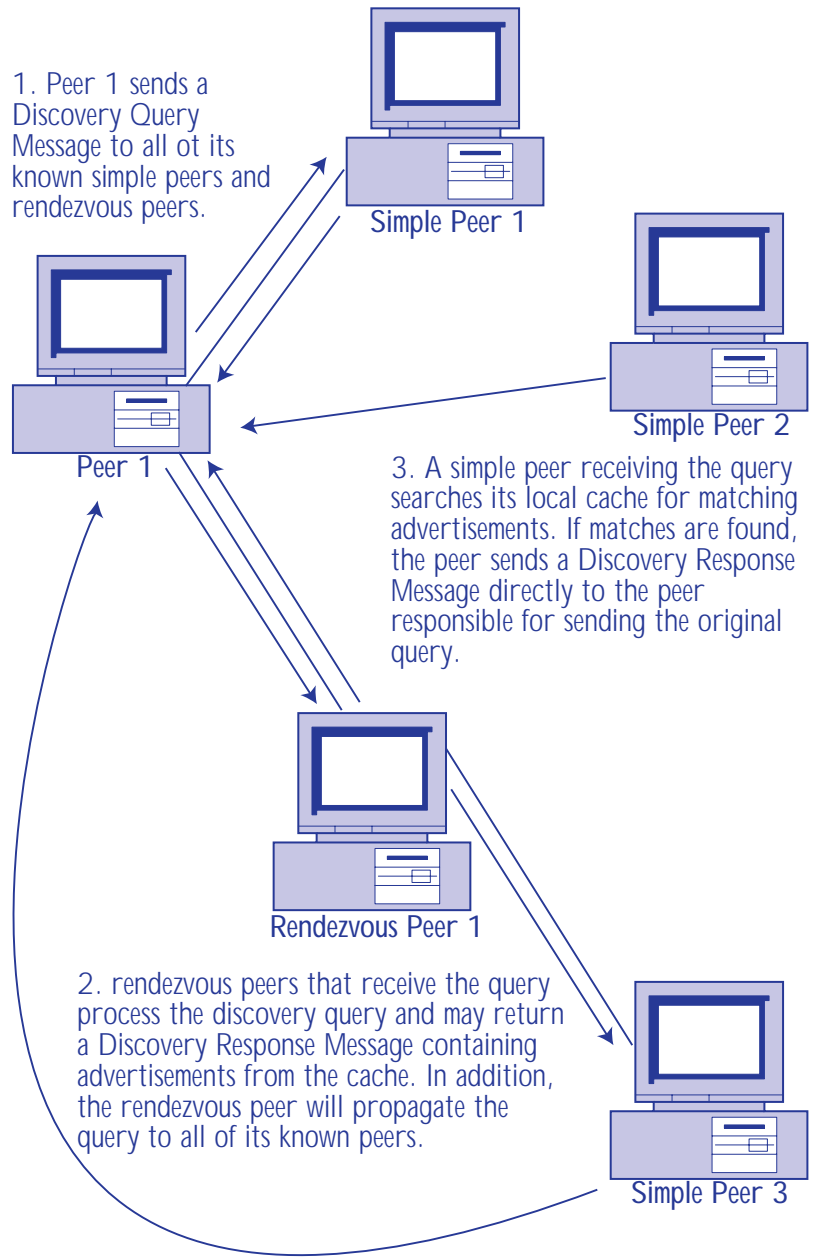


figure 5

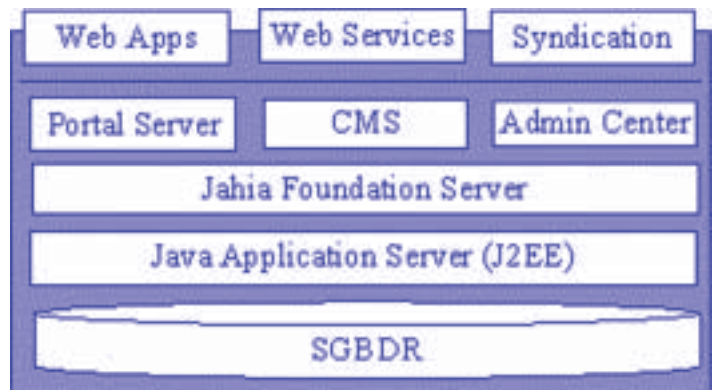


figure 6



figure 7

Voilà pour le côté serveur. Côté client, imaginons que vous êtes éditeur d'un ensemble de pages et que vous voulez vous attacher les services de *BasicFilteringService*. Pour se faire sélectionnez le mode *Edit* dans la barre d'outil. Cliquez *Ajouter une boîte* qui doit contenir le service. Donnez lui un nom Filtering et choisissez le type *Webapps* (figure 8).

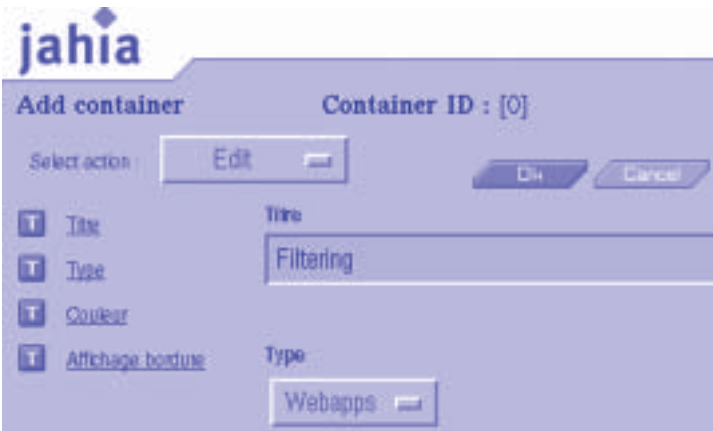


figure 8

En cliquant OK vous devez trouver dans la page la boîte ci-contre:



Cliquez le signe + pour ajouter le service (figure 9):



figure 9

Choisir le service dans la liste qui vous est proposée. Si le service offre des rôles (administrateur, guest,...) cliquez le bouton *Manage* pour l'attribution de ceux-ci. Quitter le menu, et normalement des nouvelles du jour dans la catégorie *Génétique* doivent s'afficher:



(résultat du 18/09/02 après-midi)

Conclusion

Le but de cet article, au-delà de son aspect didactique, est de souligner l'urgence d'organiser tous les projets d'intérêt général autour de cette idée de mutualisation de services. Suivant notre capacité de réaction trois arithmétique nous attendent:

- 1+1=1 (le tout vaut moins que la somme des parties: **régression**)
- 1+1=2 (**état stationnaire**)
- 1+1=3 (le tout vaut plus que la somme des parties: **évolution**) ■