Les Web Services

Les Web Services

Caractéristiques
Notions XML
SOAP
WSDL
UDDI

Caractéristiques

Pourquoi un nouveau middleware ?

Caractéristiques : besoins

Communiquer dans un environnement réparti & hétérogène :

	Protocole	Format représentation données
DCOM	RPC DCE	binaire
CORBA	IIOP	binaire (CDR)
RMI	JRMP/IIOP	binaire (Java Serialization)

Protocoles + format de représentation données spécifiques

Interopérabilité ok, mais…

Solution liée à l'environnement

Nécessite un protocole de transport (TCP ?)

Configuration du réseau (no de port?)

Caractéristiques : limitation des middleware

Passage à large échelle : Web

Protocoles propriétaires

- IIOP, RMI, DCOM
- Firewall

Pas d’ouverture des services

- Notion de moteur de recherche inexistante

Trop de contraintes sur le client !

- Doit posséder les souches
- Difficulté de construire dynamiquement

Caractéristiques : limitation des middleware

Inconvénients Intrinsèques

- Complexité
  - CORBA : IDL, Mapping, …
  - EJB : Container, JNDI, …
- Pérennité : remise en question
  - CORBA, EJB, .Net, …
- Prix
  - Plates-formes
  - Compétences

Caractéristiques : limitation des middleware

Solutions existantes

Modification du Protocole

- RMI / IIOP

Passerelles

- CORBA vers DCOM

Portage d’applications existantes difficile

Solutions non standards

Caractéristiques

- quel est le protocole le + utilisé sur Internet ?

- quel est le format de données universel ?

    - HTTP: simple, sans état, toutes les chances de traverser les firewalls

    - XML : ASCII (pas binaire), parsing facile, standard W3C

Caractéristiques

Eléments de base : HTTP

- Protocole applicatif (niveau 7)
- Utilise TCP (niveau 4) garantie d'un transport fiable(sans erreur)
- Pas de notion de connexion HTTP
- Toutes les commandes HTTP sont émises en mode texte (ASCII)

⇒Protocole simple, facilement implantable

Version actuelle HTTP 1.1 (RFC 2067) depuis janvier 1997

Apport principal : connexions TCP persistantes
Raison : pour les "petits" fichiers (< 10 Ko, 80 % des documents Web) le coût de l'ouverture de cx TCP est non négligeable / coût du transfert

⇒ gain de temps important

Caractéristiques

Eléments de base : HTTP

3 commandes principales (présentes dans HTTP/1.0 et 1.1)

- GET demande d'un document
- HEAD demande de l'en-tête (HTTP) d'un document
- POST dépose d'information sur le serveur

En-têtes client

• informations sur le client	From, Host, User-Agent
• information sur la page contenant le lien	Referer
• login, mot de passe	Authorization
• préférences pour le document demandé	Accept …
• conditions sur le document demandé	If …

- Accept: liste de types MIME
- Accept-Charset, Accept-Encoding, Aspect-Language
- If-Modified-Since, If-Unmodified-Since

Caractéristiques

Eléments de base : HTTP

commande URL version HTTP comprise par le client
en-tête 1 HTTP: valeur
....
en-tête n HTTP: valeur
informations envoyées par le client

```
GET /index.html HTTP/1.1
Accept-language: fr
User-Agent: Mozilla/4.0
If-modified-since: Tue, 23 Jul 1997 10:24:05
```

```
POST /index.php HTTP/1.1
Accept-language: fr

nom=Roos&prenom=JF
```

Caractéristiques

Eléments de base : HTTP

Réponse du serveur

version HTTP du serveur code retour commentaire
en-tête 1 HTTP: valeur
....
en-tête n HTTP: valeur
document texte (HTML, …) ou binaire (GIF, JPG, …)

```
HTTP/1.1 200 OK
Content-Length: 9872
Content-Type: text/html
<HTML>
....
</HTML>
```

Caractéristiques

Eléments de base : HTTP
code retour : renseigne sur le succès (200) ou l'échec (4xx) de la requête

- 200 : ok
- 404 : document inconnu
- 401 : authentification nécessaire
- 500 : erreur du serveur HTTP dans le traitement requête (servlet, PHP, …)
- 503 : serveur temporairement surchargé
- …

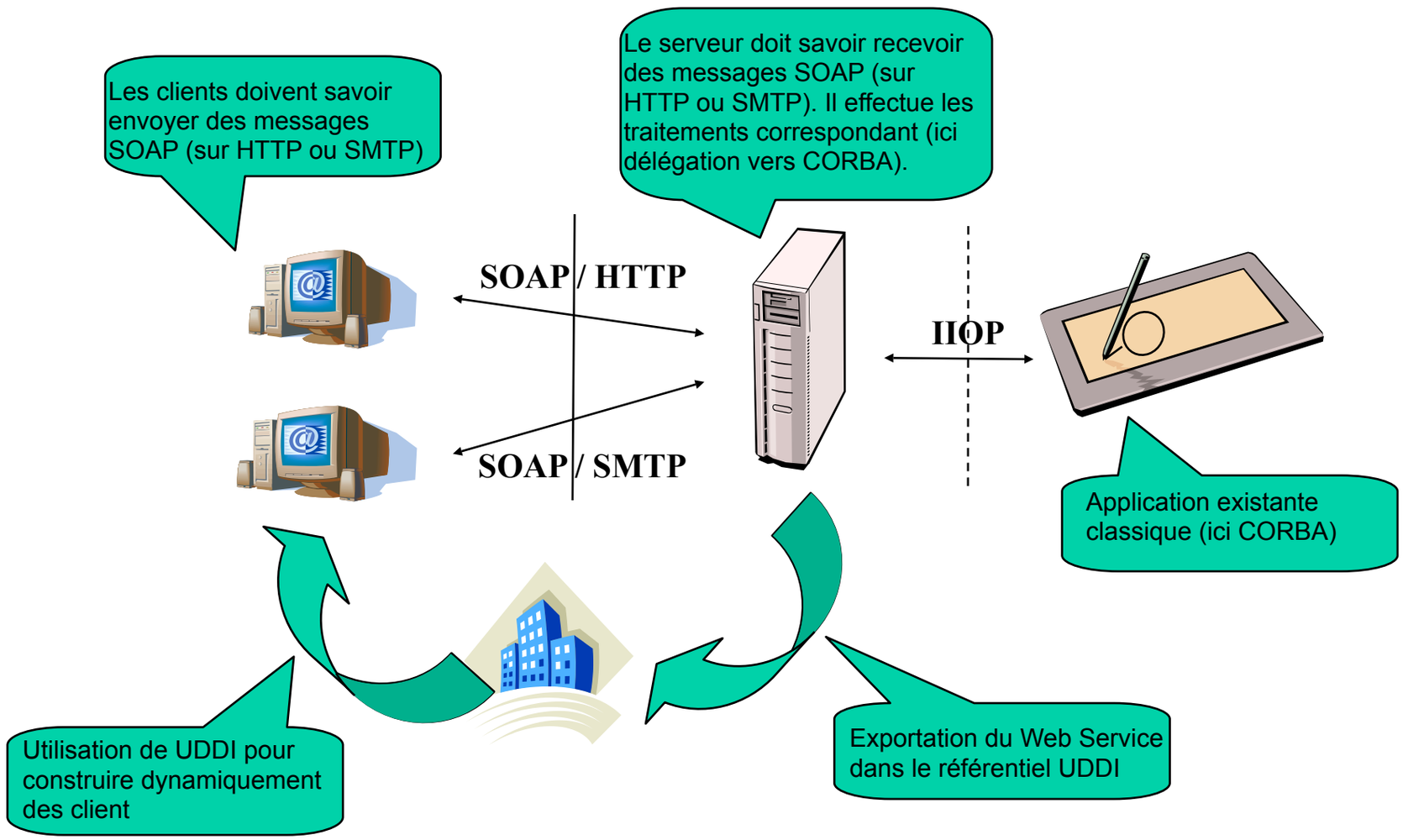
en-têtes HTTP : informations transmises par le serveur sur le document envoyé

- Content-Length : taille du document
- Last-Modified : date de dernière modification du document
- Server : nom du logiciel serveur
- Expire : date d'expiration du document
- Content-Type : type (MIME) du document
- … nombreux autres en-têtes possibles

Caractéristiques : approche envisagée

- Représentation des données : SOAP
  - Basé sur XML
    - Portabilité, Hétérogénéité
  - Porté sur des protocoles large échelle existants
    - HTTP, SMTP, …
- Langage de définition de services : WSDL
  - Définition de services offerts (en XML)
- Annuaire des services : UDDI
  - Référentiel de Web Service
    (Pages Jaunes, Vertes, Blanches)

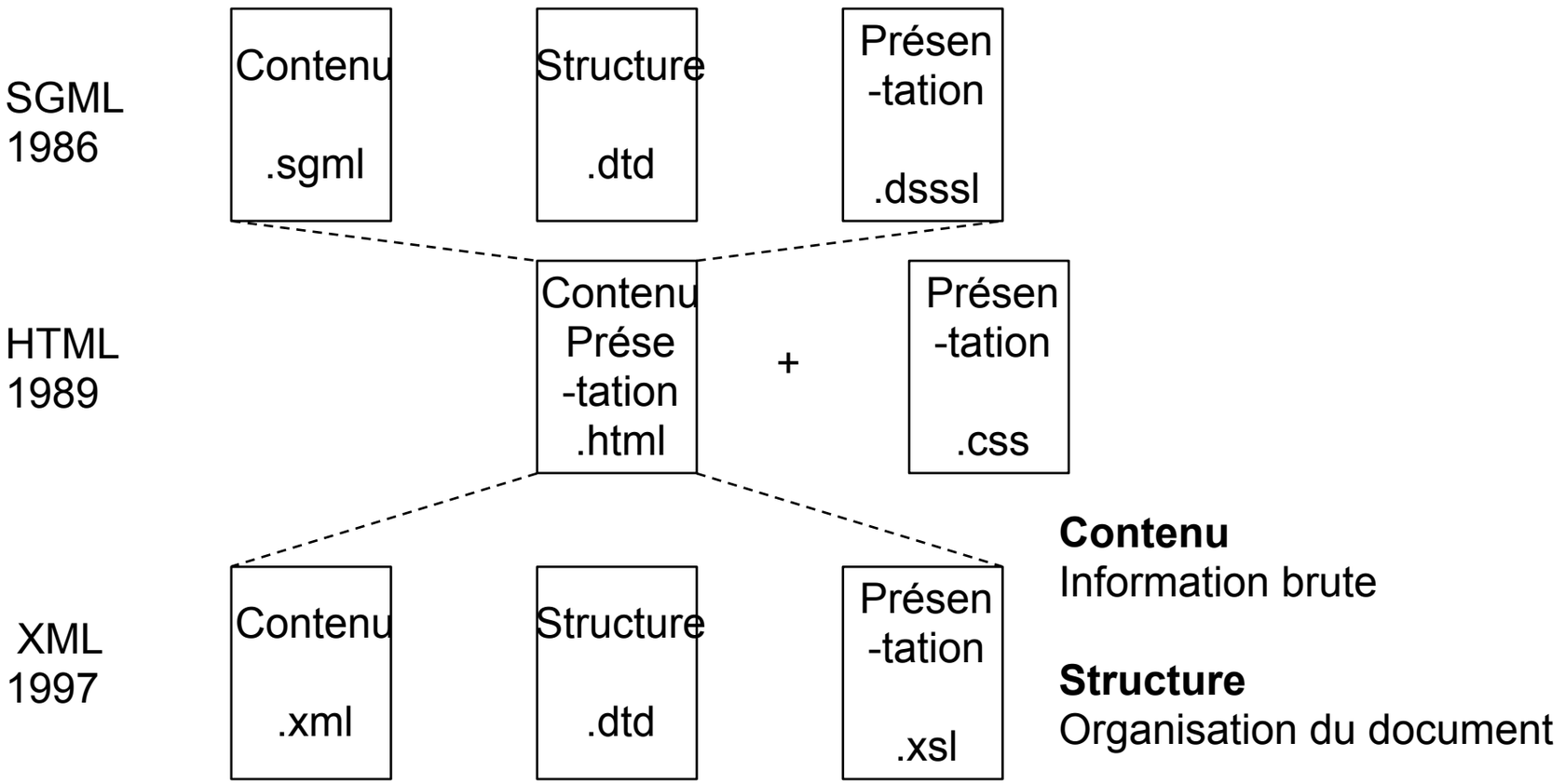
Caractéristiques : exemple



XML

Notions nécessaires pour les Web Services

XML



Contenu
Information brute

Structure
Organisation du document

Présentation
Rendu graphique

XML : principes

- Ensemble non fini de balises
  - L’utilisateur peut créer de nouvelles balises
- Définition de grammaires : XML est un Meta-Langage
  - MathML, NewsML, XMI, Doc, Slides, …
- Séparation de la forme et du fond

XML

1.DTD
2.XML Schema
3.XML Namespace

Déclaration version XML utilisée
DTD utilisée pour ce document
Corps du document

```
<?xml version="1.0" ?>
<!DOCTYPE individu SYSTEM "individu.dtd" >
<individu>
  <nom>Roos</nom>
  <prenom>JF</prenom>
</individu>
```

valide syntaxiquement
conforme à sa DTD

XML

DTD grammaire (balises) du document

1. Définition des balises autorisées <!ELEMENT ... >
2. Définition de leurs attributs <!ATTLIST ... >

```
<balise attribut="type attr" ...> type balise </balise>

<!ELEMENT graphe (noeud|arc)* >
<!ELEMENT noeud EMPTY >
<!ELEMENT arc EMPTY >
<!ATTLIST noeud numero ID #REQUIRED >
<!ATTLIST arc source IDREF #REQUIRED >
<!ATTLIST arc destin IDREF #REQUIRED >

<graphe>
<noeud numero="1"/> <noeud numero="2"/>
<arc source="2" destin="1"></arc>
</graphe>
```

XML

XML Schema

Document XML pour la définition de DTD

les DTD XML sont définies comme des documents XML

Avantages :

- un seul et même langage pour les documents et la définition de leur DTD
- types de données plus évolués (entier,réel,chaîne, date,liste, ...)
- grammaires définissables potentiellement plus évoluées

XML

XML Schema

```
<!ELEMENT promotion (individu)+ >
<!ELEMENT individu ( nom , prenom ) >
<!ELEMENT nom (#PCDATA)> <!ELEMENT prenom (#PCDATA)>
<!ATTLIST individu noSecuriteSociale ID #REQUIRED >
```

```
<?xml version="1.0" ?>
  <element name="promotion" type="PromotionType" />
  <complexType name="PromotionType">
    <element name="individu" type="IndividuType"
           minOccurs="1" maxOccurs="unbounded" />
    <attribute name="noSecuriteSociale" type="ID" use="required" />
  </complexType>
  <complexType name="IndividuType">
    <sequence>
      <element name="nom" type="string">
      <element name="name="prenom" type="string">
    </sequence>
  </complexType>
```

XML

XML Namespace

Utilisation des balises provenant de plusieurs DTD dans un document XML

- attribut réservé xmlns fournissant un nom et l'URL de sa DTD associée
- peut être ajouté à n'importe quelle balise(engénéral, la 1ère du document)

```
<balisexmlns:nomDEspace="URL associée" ... >
<html xmlns:m="http://www.w3.org/1998/Math/MathML"
xmlns:s="http://www.w3.org/2000/svg" >
```

- l'espace de noms reste valide jusqu'à la balise fermante(ici</html>)
- les balises des ≠ DTD doivent être préxifées par nomDEspace :

```
<s:svg width="2cm" height="0.6cm">
```

XML

XML Namespace

Exemple
Utilisation conjointe de 3 DTD
XHTML, SVG (figures géométriques) et MathML (formules mathématiques)

```
<?xml version="1.0" ?>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:s="http://www.w3.org/2000/svg"
      xmlns:m="http://www.w3.org/1998/Math/MathML">
<head> <title>Exemple d'utilisation des espaces de noms</title> </head>
<body>
  <h1>Les espaces de noms</h1>
  <p>Un rectangle en SVG</p>
  <s:svg width="120" height="35">
   <s:rect width="120" height="35" rx="12" fill="blue"stroke="#4C00E5" />
  </s:svg>
  <p>Une formule de math enMathML</p>
    <m:math> <m:msqrt>x+y</m:msqrt> </m:math>
</body> </html>
```

SOAP

Simple Object Access Protocol

SOAP

3 points de vue

- RPC objet
- Protocole d’échange de messages
- Format d’échange de documents

3 qualités

- Indépendant des langages
- Indépendant des OS
- Indépendant des protocoles de communication (mais HTTP ;-)

Objectif : invoquer un service distant

Spécification SOAP

1. SOAP envelope spécification quelle méthode? Paramètre? Retour?erreur?
2. Data encoding rules règles de représentation des types de données

SOAP

v1.1 SOAP = Simple Object Access Protocol

depuis v1.2 SOAP = SOAP

SOAP n'a aucune notion orientée objet

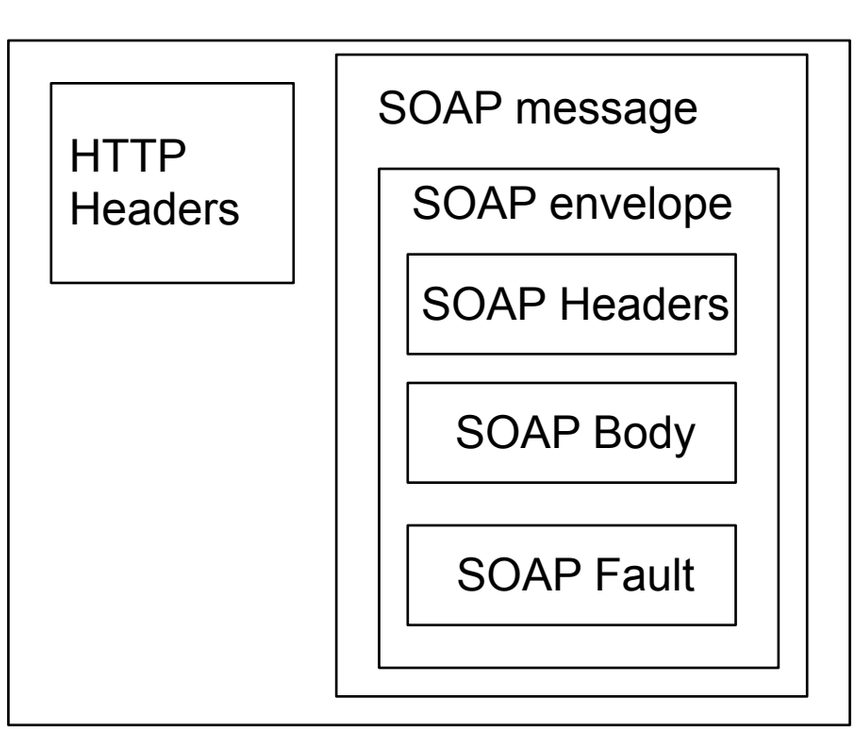
- pas de référence d'objet
- pas d'instanciation
- pas de cycle de vie
- pas de GC
- pas de variables d'instances
- pas d'état "conversationnel" i.e. session (+/-= EJB stateless bean)

SOAP : technologie centrée sur document XML

SOAP

Structure du message SOAP

- 1 invocation par message
- La réponse a la même structure



Envelope+ Body: Obligatoires
Headers+Fault : facultatifs

SOAP

Enveloppe contient

- Les informations du message
- Document XML
- Balise racine `<Enveloppe>`
- 2 balises principales : `<Header> et <Body>`
- balise utilisateur (méthode & param) : ex `<euroToDollar> <value>`

Invocation du service `euroToDollar` avec la valeur `12.34`

```
<Envelope>
 <Body>
   <euroToDollar>
     <value>12.34</value>
   </euroToDollar>
 </Body>
</Envelope>
```

SOAP

Enveloppe

Risque conflits balises SOAP & utilisateur de noms utilisation de namespace

- SOAP-ENV namespace XML pour SOAP
  http://schemas.xmlsoap.org/soap/envelope

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope">
  <SOAP-Env:Body>
    <euroToDollar>
      <value>12.34</value>
    </euroToDollar>
  </SOAP-Env:Body>
</SOAP-ENV:Envelope>Enveloppe
```

SOAP

Enveloppe
Exemple HTTP

```
POST /convertisseur HTTP/1.1
Content-Type: text/xml
Content-Length: 999
SOAPAction: http://some.host/SOAPServer/convertisseur

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope">
  <SOAP-Env:Body>
    <euroToDollar>
      <value>12.34</value>
    </euroToDollar>
  </SOAP-Env:Body>
</SOAP-ENV:Envelope>
```

SOAPAction(URI du service web invoqué) doit être présent
mais peut-être vide URL POST suffisante pour identifier le service

SOAP

Enveloppe
Exemple HTTP

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: 999
```

```
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
envelope">
  <SOAP-ENV:Body>
    <euroToDollarResponse>
      <return>10.07</return>
    </euroToDollarResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP

En-tête

- Informations supplémentaires sur le message : balises
- Non liées à l'invocation proprement dite
- En relation avec son contexte d'exécution

Exemple

```
<SOAP-ENV:Header>
  <Transaction SOAP-ENV:mustUnderstand="1">
    5
  </Transaction>
</SOAP-ENV:Header>
```

2 attributs facultatifs

- `mustUnderstand="1"` Le serveur doit être capable de traiter le message
- `Actor="url"` destinataire du header en cas d'appels en cascade

SOAP

- Le Body contient le message à échanger
- La balise Body est obligatoire
- La balise Body doit être le premier fils de la balise Envelope (ou le deuxième si il existe une balise Header)
- La balise Body contient des entrées
- Une entrée est n’importe quelle balise incluse optionnellement dans un namespace
- Une entrée peut être une Fault.

SOAP

pour signaler une erreur d'exécution la balise `<fault>`

4 sous- balises

- `faultcode` : type d'erreur
- `faultstring` : message d'erreur pour l'utilisateur
- `faultactor` : émetteur de l'erreur en cas d'appel en cascade
- `Detail` : message détaillé pour l'application (stack trace)

4 valeurs principales pour faultcode

- `Client` erreur provenant de la requête du client
- `Server` erreur provenant du serveur
- `MustUnderstand` incapacité à traiter un header mustUnderstand
- `VersionMismatch` namespace de l'enveloppe incorrect

mais valeurs extensibles (même esprit que les code 4xx pour HTTP)
ex : `Client.Authentication`

SOAP

Exemple

```
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV=« http://schemas.xmlsoap.org/soap/envelop »>

  <SOAP-ENV:Fault>

    <SOAP-ENV:faultcode>Client</SOAP-ENV:faultcode>

    <SOAP-ENV:faultstring>Methode inexistante</SOAP-
ENV:faultstring>

  </SOAP-ENV:FAULT>
</SOAP-ENV:Envelope>
```

SOAP

Règles de représentation des types de données
But : typer les données échangées

- Types simples : string,int,double,boolean, date, time, enum, tableaux d'octets
- Types composés : structures, tableaux

Positionner l'attribut
`encodingStyle` `http://schemas.xmlsoap.org/soap/encoding`
(aussi utilisé comme namespace)

- Typage en référençant un XML Schema
- Typage directement avec les données transmises
  - attribut type ->namespace `xsi`
  - typage XML Schema -> namespace `xsd`
  - `xsi` `http://www.w3.org/2001/XMLSchema-instance`
  - `xsd` `http://www.w3.org/2001/XMLSchema`

SOAP

Types simples
Exemple de message avec données typées

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<SOAP-ENV:Body>

<euroToDollarResponse
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/
encoding">
<return xsi:type="xsd:double">10.07</return>
</euroToDollarResponse>

</SOAP-ENV:Body> </SOAP-ENV:Envelope>
```

SOAP

Types simples

Exemple de message avec typage externe

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope">
    <SOAP-ENV:Body>
      <euroToDollarResponse
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding">
        <foo:return xmlns:foo="uri du schema">10.07</foo:return>
      </euroToDollarResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Schéma

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="return" type="xsd:double" />
</xsd:schema>
```

SOAP

Types simples
Enumération

- cas particulier de chaîne avec une liste de valeurs autorisées

Schéma

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="return">
    <simpleType base="xsd:string">
      <enumeration value="mineur" />
      <enumeration value="majeur" />
    </simpleType>
  </xsd:element>
</xsd:schema>
```

Message

```
<foo:return xmlns:foo="uri du schema">majeur</foo:return>
```

SOAP

Types simples
Tableau d'octets

- type base64

```
<picture xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
  xmlns:SOAP-ENC="http://www.w3.org/soap/encoding"
     xsi:type="SOAP-ENC:base64" >
    aG93IG5vDyBicm73biBjb3cNCg==
</picture>
```

SOAP

Types composés
Structures

```
struct Personne { string nom; int age; }
```

```
<foo:Personne xmlns:foo="uri du schema" >
  <foo:nom>Bob</foo:nom>
  <foo:age>45</foo:age>
</foo:Personne>
```

- Schéma

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Personne" >
    <xsd:complexType base="xsd:string">
      <xsd:element name="nom" type="xsd:string" />
      <xsd:element name="age" type="xsd:int" />
    </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

SOAP

Types composés
Tableaux

- type arrayType

```
<myFavoriteNumbers
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:SOAP-ENC="http://www.w3.org/2001/06/soap-encoding"
  SOAP-ENC:arrayType="xsd:int[2]" >
  <number>3</number>
  <number>4</number>
</myFavoriteNumbers>
```

- éléments de types non homogènes,
- éléments de types structure ou tableaux
- tableaux multi-dimensionnels
- creux (seuls certains éléments sont transmis)

SOAP

Implantation
Axis ws.apache.org/axis/

Architecture

- serveur HTTP
- Tomcat
- servlet RPC Router
  - "comprend" SOAP
  - aiguille les requêtes

Services SOAP en. Java + descripteur de service XML

```
public class EssaiDeWS {
  public double euroToDollar(double euro) { return euro/1.12; }
}
```

Client SOAP en Java avec API org.apache.soap

SOAP

Conclusion

- protocole simple, facilement implantable
- sécurité basée sur la sécurité du protocole sous-jacent (ex. HTTPS)
- indépendant langages, OS

mais

- pas de passage d'objets par référence
- pas d'activation à la demande
- pas de gestion de l'exécution des requêtes
- pas de ramasse-miettes
- pas de transaction entre +sieurs invocations (voir extensions)
- fiabilité essentiellement basée sur TCP

juste un protocole de transport d'invocation de services

WSDL

Web Service Description Language

WSDL

Description de services Web

- Contrat d’utilisation
- Description XML

4 informations

- Interface du service (méthodes disponibles)
- Types de données
- Mode de connexion
- Localisation du service

WSDL

Balises WSDL

- <definition> Racine
- <types> les types de données échangées
- <message> les messages transmis
- <portType> regroupement d’opérations
- <operation> des opérations
- <binding> les modes de transmission des messages.
- <service> la localisation des services

WSDL

Types : description sous forme XML Schema.

```
<wsdl:types>
     <xsd:schema
        targetNameSpace="http://www.exemple.fr/personne.xsd"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
         <xsd:element name="personne">
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:element name="nom" type="xsd:string" />
                  <xsd:element name="prenom" type="xsd:string"/>
               </xsd:sequence>
            </xsd:complexType>
         </xsd:element>
     </xsd:schema>
</wsdl:types>
```

WSDL

Message

- défini un profil de paramètres
- un message est composé de part
  - élément de type simple
  - référence un type précédemment défini

```
<wsdl:message name="AddPersonneRequest">
  <wsdl:part name="nom" type="xsd:string" />
  <wsdl:part name="age" type="xsd:int" />
</wsdl:message>

<wsdl:message name="AddPersonneResponse">
</wsdl:message>

<wsdl:message name="RemovePersonneRequest">
  <wsdl:part name="nom" element="PersonneType" />
</wsdl:message>
```

WSDL

Opération

- un message invocable (input)
- et/ou un message émis (output) ou un message d'erreur (fault)

input

- one-way 1 seul message en input
- request-response 1 message en input + 1 en output

output

- solicit-response 1 seul message en output + 1 en input
- notification 1 message en output

```
<wsdl:operation name="addPersonne">
  <wsdl:input message="AddPersonneRequest" />
  <wsdl:output message="AddPersonneResponse" />
  <wsdl:fault message="AddPersonneFault" />
</wsdl:operation>
```

WSDL

Port

- regroupement d'opérations
- comme une interface

```
<wsdl:portType name="PersonPortType">
  <wsdl:operation name="addPersonne">
    ...
  </wsdl:operation>
  <wsdl:operation name="removePersonne">
    ...
  </wsdl:operation>
</wsdl:portType>
```

WSDL

Binding

- spécifie la liaison entre un port et un protocole
- précise le
  - type de transport (ex. HTTP)
  - la façon dont sont créés les messages (style)

```
<wsdl:binding name="PersonneBinding" type="PersonPortType">
  <soap:binding
    transport="http://schemas.xmlsoap.org/soap/http"
    style="rpc" />
  ...
</wsdl:binding>
```

WSDL

Binding

- pour chaque opération l'URI associée (voir en-tête HTTP SOAPAction)
- pour chaque message le style d'encodage

```
<wsdl:operation name="addPersonne">
<soap:operation soapAction="http://somehost/SOAPServer/conv"/>
  <wsdl:input>
   <soap:body
    use="encoded"
    encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
  </wsdl:input>
  <wsdl:output>
    ...
  </wsdl:output>
  <wsdl:fault>
    ...
  </wsdl:fault>
</wsdl:operation>
```

WSDL

Service

- un ensemble de liaisons (i.e. de ports)
- pour chaque liaison, sa localisation

```
<wsdl:service name="ServicePersonne">
 <wsdl:port binding="PersonneBinding">
 <soap:location="http://localhost:8080/soap/servlet/
rpcrouter"/>
 </wsdl:port>
 ...
</wsdl:service>
```

WSDL

Conclusion WSDL

- langage de définition d'interfaces de services web
- type ⊂message ⊂operation ⊂port ⊂liaison ⊂service
- fournit le lien entre ces éléments et SOAP

- XML
- génération automatique de WSDL àpartir de Java, EJB, ...

- invocation dynamique d'un service àpartir de sa réprésentation WSDL
- voir :
  - Web Service Invocation Framework : http://ws.apache.org/wsif/
  - API javax.jws Java SE 6

UDDI

Universal Description Discovery and Integration

UDDI

Annuaire de WS

- Pages blanches WS classés par fournisseurs
- Pages jaunes WS classés par catégorie
- Pages vertes informations techniques sur les WS

Annuaire UDDI en ligne

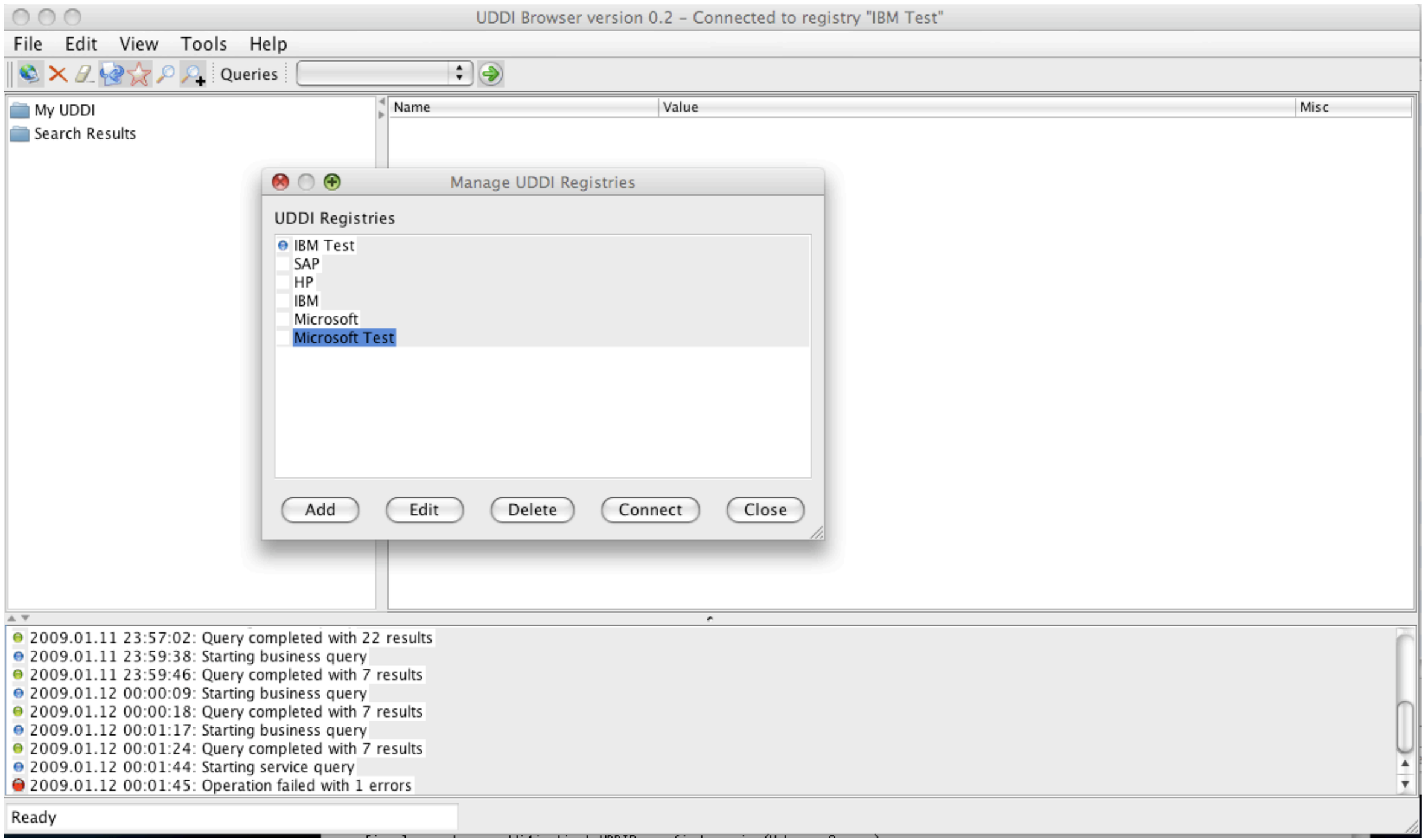
- uddi.microsoft.com
- www.ibm.com/services/uddi
- uddi.hp.com
- udditest.sap.com

UDDI

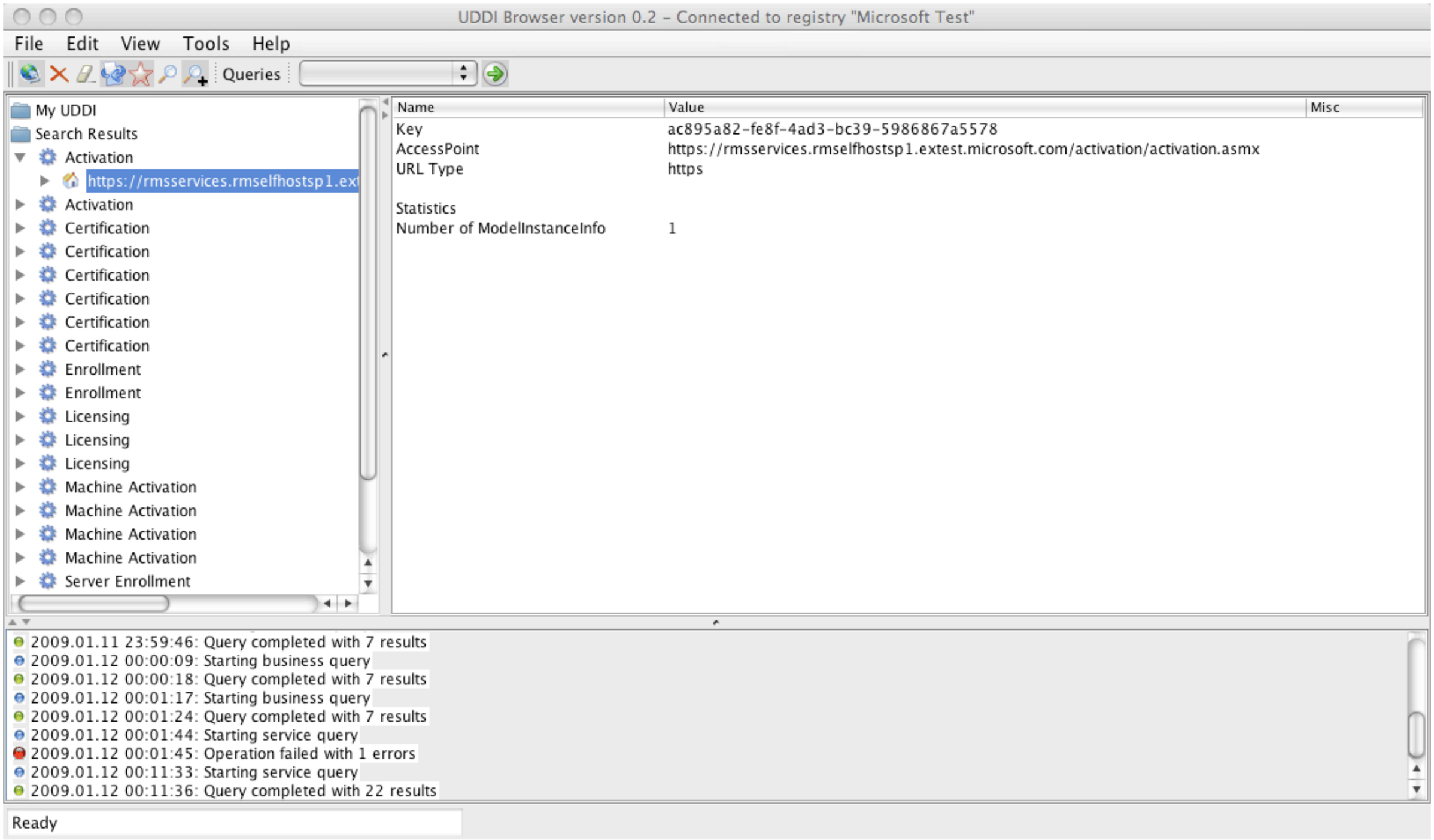
- Fonctionnalités :
    - Ajout
    - Suppression
    - Recherche
    - Navigation

Ex : API Sun JAXR

UDDI


UDDI Browser version 0.2 – Connected to registry "IBM Test"
File
Edit
View
Tools
Help
Queries
My UDDI
Search Results
Name
Value
Misc
Manage UDDI Registries
UDDI Registries
IBM Test
SAP
HP
IBM
Microsoft
Microsoft Test
Add
Edit
Delete
Connect
Close
2009.01.11 23:57:02: Query completed with 22 results
2009.01.11 23:59:38: Starting business query
2009.01.11 23:59:46: Query completed with 7 results
2009.01.12 00:00:09: Starting business query
2009.01.12 00:00:18: Query completed with 7 results
2009.01.12 00:01:17: Starting business query
2009.01.12 00:01:24: Query completed with 7 results
2009.01.12 00:01:44: Starting service query
2009.01.12 00:01:45: Operation failed with 1 errors
Ready

UDDI


UDDI Browser version 0.2 - Connected to registry "Microsoft Test"
File Edit View Tools Help
Queries
My UDDI
Search Results
Activation
https://rmsservices.rmselfhostsp1.ext
Activation
Certification
Certification
Certification
Certification
Certification
Enrollment
Enrollment
Licensing
Licensing
Licensing
Machine Activation
Machine Activation
Machine Activation
Machine Activation
Server Enrollment
Name
Value
Misc
Key
ac895a82-fe8f-4ad3-bc39-5986867a5578
AccessPoint
https://rmsservices.rmselfhostsp1.extest.microsoft.com/activation/activation.asmx
URL Type
https
Statistics
Number of ModelInstanceInfo
1
2009.01.11 23:59:46: Query completed with 7 results
2009.01.12 00:00:09: Starting business query
2009.01.12 00:00:18: Query completed with 7 results
2009.01.12 00:01:17: Starting business query
2009.01.12 00:01:24: Query completed with 7 results
2009.01.12 00:01:44: Starting service query
2009.01.12 00:01:45: Operation failed with 1 errors
2009.01.12 00:11:33: Starting service query
2009.01.12 00:11:36: Query completed with 22 results
Ready

Conclusion

Avantages :

- Des standards simples (SOAP, WSDL, UDDI)
- Multi Protocole / Multi OS / Multi Langage
- Paradigme de Service
- Des outils (éditeurs et moteurs)

Inconvénients :

- Typage (pas de consensus)
- Performance
- Jeunesse (Sécurité, Transaction,…)