M2 RS

Module : Services Web

Chapitre 03:

Les Technologies Associées :

- http,
- SOAP,
- WSDL,
- UDDI

Sommaire

1. Introduction- Description en couche des services Web

→ Les services Web emploient un ensemble de technologies qui ont été conçues afin de respecter une structure en couches sans être dépendante de façon excessive de la pile des protocoles.

→ Cette structure est formée de quatre couches majeures :

Découverte de services	UDDI
Description de services	WSDL
Communication	SOAP
Transport	HTTP

Couche transport

Cette couche est responsable du transport des messages SOAP (c.à.d. des messages XML) échangés entre les applications. Actuellement, cette couche inclut HTTP, SMTP, FTP, et de nouveaux protocoles tels que BEEP.

Couche communication

Cette couche est responsable du formatage des données échangées de sorte que les messages peuvent être compris à chaque extrémité. SOAP est basé sur XML pour accéder à des services Web.

Couche description de service

Cette couche est responsable de la description de l'interface publique du service Web. Le langage utilisé pour décrire un service Web est WSDL qui est la notation standard basée sur XML pour construire la description de l'interface d'un service.

Couche découverte de service

Cette couche est chargée de centraliser les services dans un registre commun, et de simplifier les fonctionnalités de recherche et de publication des services Web. Actuellement, la découverte des services est assurée par un annuaire UDDI (Universal Description, Discovery, and Integration).

2. http (HyperText Transfer Protocol)

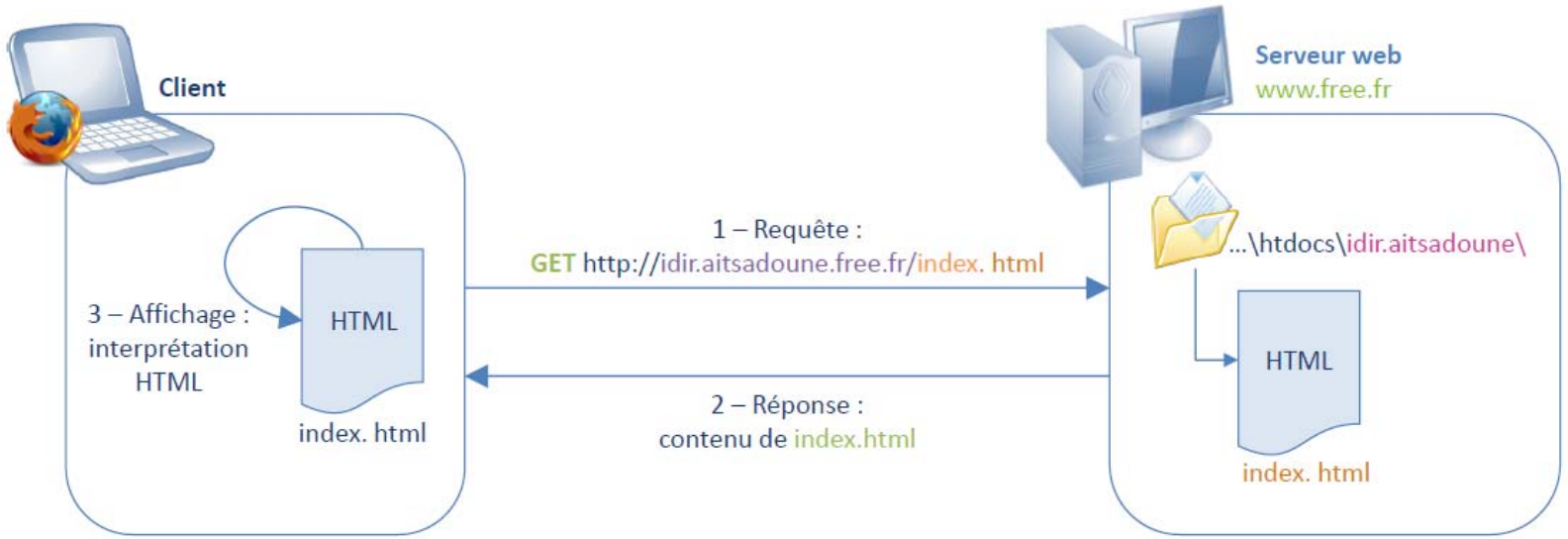
→ Basé sur TCP/IP

→ Disponible sur toutes les plates-formes

→ Protocole d'échanges d'information sur le web

→ Un protocole simple et sans connexion (Peu de paquets sont nécessaires pour échanger des informations)

→ utilise un protocole requête/réponse basé sur du texte



3. SOAP (Simple Object Access Protocol)

→ Norme W3C .

→ Est un protocole de transmission de messages (échange d'informations structurées).

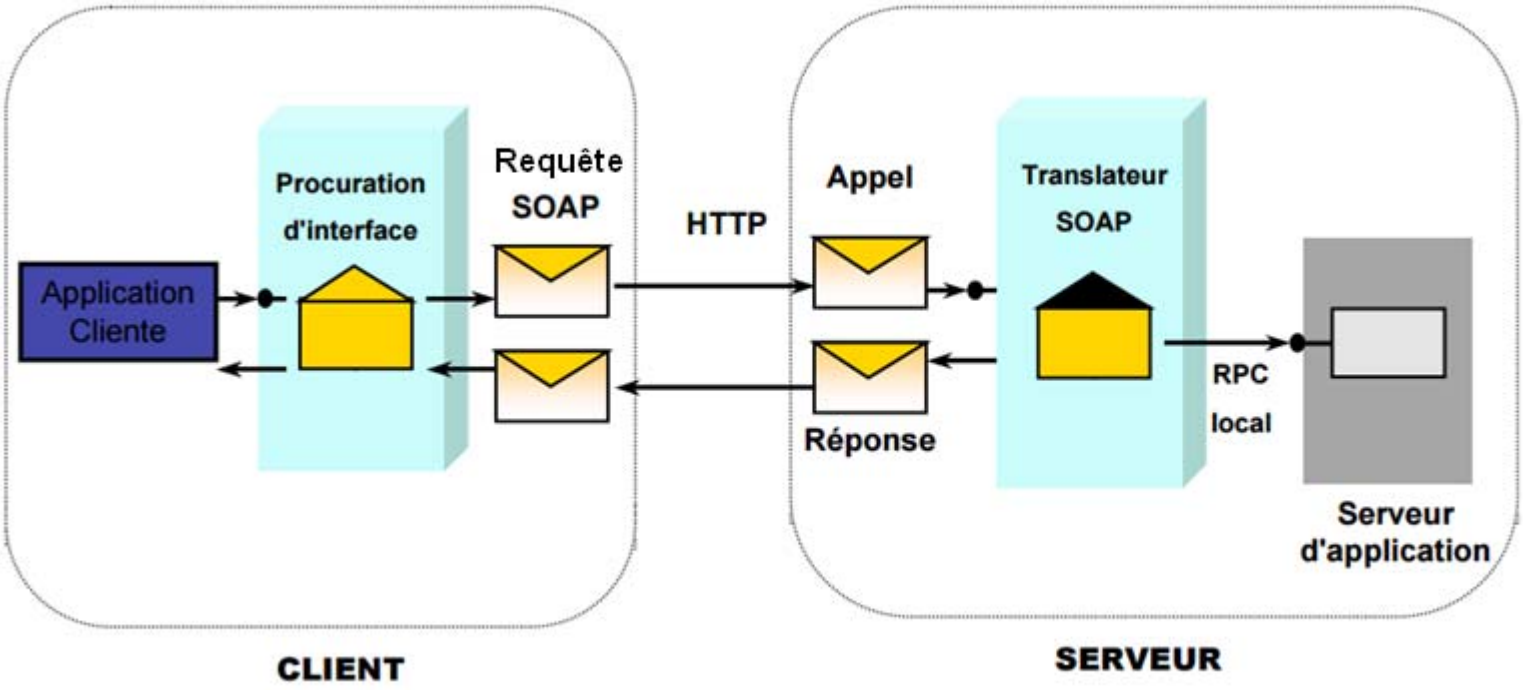
→ Permet des appels de procédures à distance (RPC) s'appuyant principalement sur le protocole HTTP et sur XML.

→ Il est indépendant d'un modèle particulier de programmation : les plates-formes et les technologies existantes.

→ Historique de SOAP

- SOAP 1.0: 1999, basé sur HTTP
- SOAP 1.1: 2000, plus générique, autres protocoles
- SOAP 1.2: recommandation W3C, 2007: https://www.w3.org/TR/soap/

→ En gros, c'est un protocole pour échanger des informations en envoyant des messages



SOAP Côté client

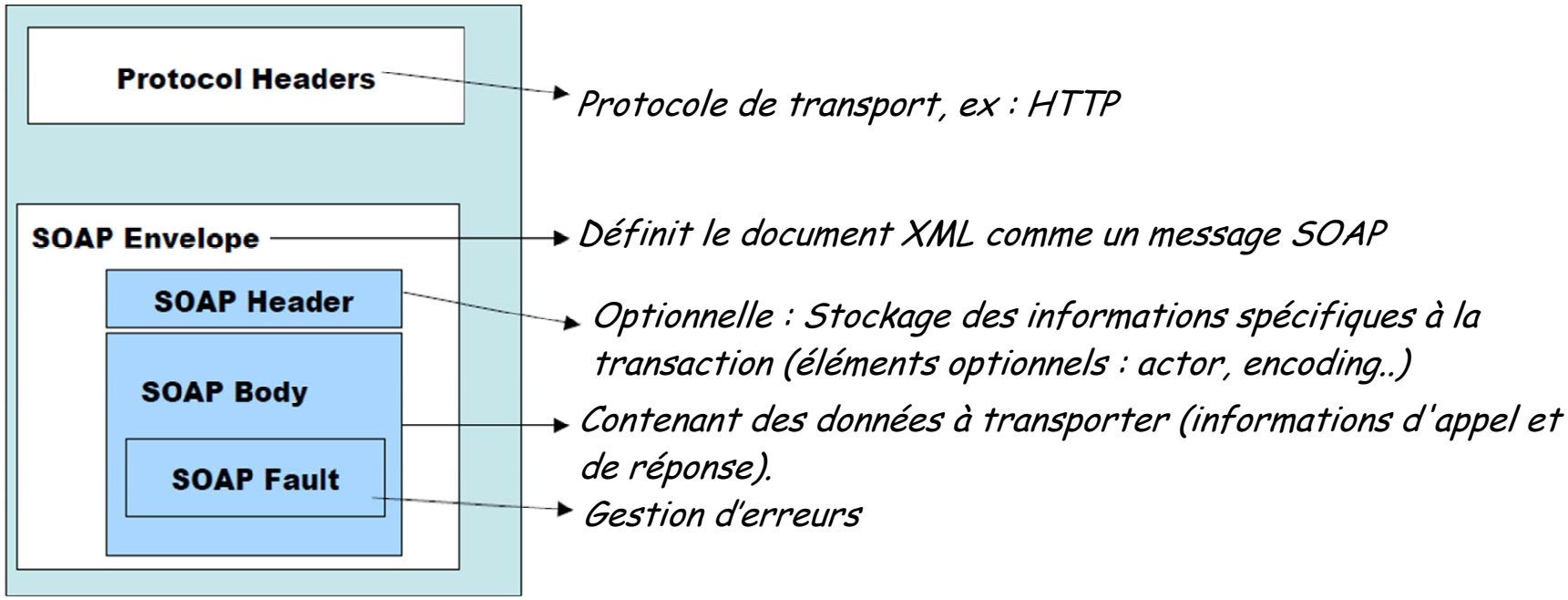
- Ouverture d'une connexion HTTP
- Requête SOAP: document XML décrivant :
  - La méthode à invoquer sur la machine distante
  - Les paramètres de la méthode

SOAP Côté Serveur

- Récupère la requête
- Exécution de la méthode avec les paramètres
- Renvoie une réponse SOAP (document XML) au client

Le message SOAP est destiné au fournisseur du Service après avoir contacté l'annuaire pour chercher le service correspondant au besoin du client, les informations obtenues permettent au client de connaitre la localisation du service pour pouvoir l'invoquer à l'aide de messages SOAP.

Structure d'un message SOAP



→ Un message SOAP est un document XML contenant les éléments suivants:

- ***Envelope:*** c'est lui qui contient le message et ses différents sous-blocs. Il s'agit du bloc racine XML. Il peut contenir un attribut *encodingStyle dont la valeur est une* URL vers un fichier de typage XML qui décrira les types applicables au message SOAP.
- ***Header:*** c'est un bloc optionnel qui contient des informations d'en-têtes sur le message. S'il est présent, ce bloc doit toujours se trouver avant le bloc **Body** à l'intérieur du bloc **Envelope**.
- ***Body:*** c'est le bloc qui contient le corps du message. Il doit absolument être présent de manière unique dans chaque message et être contenu dans le bloc **Envelope**. SOAP ne définit

pas comment est structuré le contenu de ce bloc. Cependant, il définit le bloc **Fault** qui peut s'y trouver.

— ***Fault:*** ce bloc est la seule structure définie par SOAP dans le bloc Body. Il sert à reporter des erreurs lors du traitement du message, ou lors de son transport. Il ne peut apparaître qu'une seule fois par message. Sa présence n'est pas obligatoire.

L'enveloppe SOAP

→L'enveloppe SOAP sert de conteneur aux autres éléments du message SOAP,

↳ définie par : la balise `<soap:Envelope>` et se termine par la balise `</soap:Envelope>`.

Exemple

```
<?xml version="1.0" encoding="utf-8"?>
 <soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
       soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding ">

    <soap:Header>
        <!-- en-tête -->
    </soap:Header>

    <soap:Body>
        <!-- corps -->
        <soap:Fault>
         <!-- erreurs  -->
        </soap: Fault >
    </soap:Body>
 </soap:Envelope>
```

→ Toutes les balises XML associées à SOAP ont le préfixe soap (ou "soap-env").

L'entête est `<soap:Header>` et le corps `<soap:Body>`.

→ SOAP repose entièrement sur les espaces de noms XML.

→ Les espaces de noms sont introduits à l'aide d'un mot-clé « `xmlns` » *XML namespace* qui signifie espace de noms XML.

→ L'espace de noms est utilisé pour identifier toutes les balises afin d'éviter les conflits.

→ La spécification (SOAP) impose que tous les attributs contenus dans l'enveloppe SOAP soient explicitement associés à un *namespace*, de manière à supprimer toute ambiguïté.

→ Par convention, la spécification SOAP définit deux *namespaces* fréquemment utilisés :

→ soap ou soap-env associé à l'URI «`http://www.w3.org/2003/05/soap-envelope/`» pour définir le *namespace* de l'enveloppe (ses éléments).

→ soap-enc:encodingStyle associé à l'URI «`http://www.w3.org/2003/05/soap-encoding`» pour la définition des formats de types de données.

→ Deux autres espaces de noms fortement utilisés dans SOAP sont « **xsd** » et « **xsi** ».

- **xsd namespace** précise que les balises proviennent de la définition de schéma XML.
- **xsi namespace** indique que les balises viennent d'une instance d'un schéma XML.

Le corps SOAP

→ Le corps SOAP est un élément obligatoire dans le message SOAP.

→ Il contient l'information destinée au receveur.

→ Le corps (*body*) doit fournir le nom de la méthode invoquée par une requête ainsi que les paramètres associés (c.à.d. appel de méthode à un ordinateur distant).

Exemple : la demande du solde d'un compte bancaire.

```
<soap:Body>
        <checkAccountBalance>
                <accountNumber xsi:type="xsd:int">1234567890</accountNumber>
        </checkAccountBalance>
</soap:Body>
```

un corps SOAP qui fait appel de procédure distante (RPC) à une méthode appelée `checkAccountBalance()` :

- L'élément `<checkAccountBalance>` fournit le nom de la méthode à appeler : `checkAccountBalance()`.
- L'élément `<accountNumber>` est un paramètre qui est passé dans la méthode `checkAccountBalance()`.
- Les attributs `xsi` et `xsd` définissent les espaces de noms qui vont être utilisés dans le corps du message.
  - La définition de `xsi` permet d'utiliser `xsi:type` dans le corps du message, le `xsd:int` signifie que cette valeur est de type entier.
- `1234567890` est la valeur donnée au paramètre.

L'ensemble de ces caractères représente un appel de méthode qui a la forme suivante en langage C :

```
int Balance = checkAccountBalance(1234567890);
```

L'en-tête SOAP

→ L'en-tête SOAP est un élément facultatif dans un message SOAP.

→ Le format de l'en-tête n'est pas défini, il est à la disposition des clients et des services pour leur propre usage.

→ Cet usage typique serait de communiquer des informations authentifiant l'émetteur ou bien encore le contexte d'une transaction dont le message SOAP doit passer par plusieurs intermédiaires SOAP pour arriver au destinataire final.

→ Un intermédiaire SOAP est toute entité capable de recevoir et transmettre des messages SOAP.

→ L'en-tête d'un message SOAP commence avec la balise `<soap:Header>` et se termine avec la balise `</soap:Header>`,

→ Trois attributs associés à l'en-tête SOAP peuvent être utilisés :

- ***soap:mustUnderstand*** : cet attribut prend la valeur 1 ou 0. La valeur 1 signale que le récepteur doit reconnaître l'information présente dans l'en-tête et que son traitement est obligatoire. La valeur 0 indique que l'en-tête peut être ignoré par le récepteur.
- ***soap:role*** : sert à indiquer le destinataire SOAP auquel un bloc d'en-tête SOAP particulier est destiné.
- ***soap:relay*** : est utilisé pour indiquer si un bloc d'en-tête SOAP ciblé sur un récepteur SOAP doit être réacheminé (relayé) s'il n'est pas traité.

Message d'erreur SOAP

→ La balise<soap:fault> est utilisée pour communiquer un problème qui a eu lieu dans la tentative de réalisation de la demande adressée au service Web.

→ L'élément d'erreur est facultatif et figure uniquement dans les messages de réponse, il ne peut y apparaître qu'une seule fois.

→ La balise <soap:fault> peut contenir quatre autres balises facultatives :

- **faultcode** : Il contient un code indiquant la nature du problème.
- **faultstring** : est la version lisible par l'homme de la balise *faultcode*. C'est la traduction en langage naturel du code d'erreur.
- **faultactor** : indique le service qui a généré l'erreur. Cela est important lorsqu'une chaîne de services a été utilisée pour traiter la demande.
- **detail** : cet élément doit contenir autant d'informations que possible sur l'état du serveur à l'instant de l'apparition de l'erreur. Il contient souvent des valeurs de variables au moment de l'échec.

→ Quatre types de codes d'erreur sont définis par la spécification :

- **soap:Server** : indique qu'une erreur s'est produite sur le serveur, mais pas avec le message lui-même.
- **soap:Client** : signifie que le message reçu contient une erreur.
- **soap:VersionMismatch** : signifie que les versions des protocoles SOAP utilisés par le client et le serveur sont différentes.
- **soap:MustUnderstand** : cette erreur est générée lorsqu'un élément dans l'en-tête ne peut pas traiter alors qu'il est marqué comme obligatoire.

Exemple de communication

```
<!--
       Protocole de transport ex. HTTP
      -->
      POST /stockquote.asmx HTTP/1.1
      Host: www.webservicex.net Content-Type: text/xml; charset=utf-8
      Content-Length: length
      SOAPAction: "http://www.webserviceX.NET/GetQuote"

<?xml version="1.0" encoding="utf-8"?>

       <!--
         Définit le document XML comme un message SOAP.
       -->
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

       <!--
      Contenant des données à transporter.
       -->
  <soap:Body>

    <GetQuote xmlns="http://www.webserviceX.NET/">
      <symbol>string</symbol>
    </GetQuote>

  </soap:Body>

</soap:Envelope>
```

Réponse

```
<StockQuotes>

    <Stock>
       <Symbol>Peugeot</Symbol>
       <Last>2.28</Last>
       <Date>20/11/2009</Date>
       <Time>4:00pm</Time>
       <Change>+0.20</Change>
       <Open>2.20</Open>
       <High>2.30</High>
       <Low>2.07</Low>
       <Volume>124718</Volume>
       <MktCap>18.0M</MktCap>
       <PreviousClose>2.08</PreviousClose>
       <PercentageChange>+9.62%</PercentageChange>
       <AnnRange>1.51 - 3.27</AnnRange>
       <Earns>-0.174</Earns>
       <P-E>N/A</P-E>
       <Name>Forward Industrie</Name>
    </Stock>

</StockQuotes>
```

4. WSDL (Web Services Description Language)

→ Standard du W3C :

- Version 1.1 en 2001
- Version 2.0 en 2007, encore peu supporté par les outils

→ ***Objectif :*** décrire l'interface publique d'un Web Service (contrat de service)

→ ***Grammaire :*** dérivée d'XML (WSDL est un document XML)

→ Web Service = ensemble de ports de connexions mettant à disposition des opérations qui reçoivent et envoient des messages.

→ Deux types d'informations :

- Fonctionnelles : interface du service (signature des méthodes, …)
- Techniques : URL, protocole, …

→ Fichier WSDL : utilisable par des outils de génération de code

→ Regroupe les informations nécessaires pour interagir avec le service (pour consommer le service)

↳ Les méthodes, les paramètres et valeurs retournées, le protocole de transport utilisé, la localisation du service

→ Est un document indispensable au déploiement de Services Web :

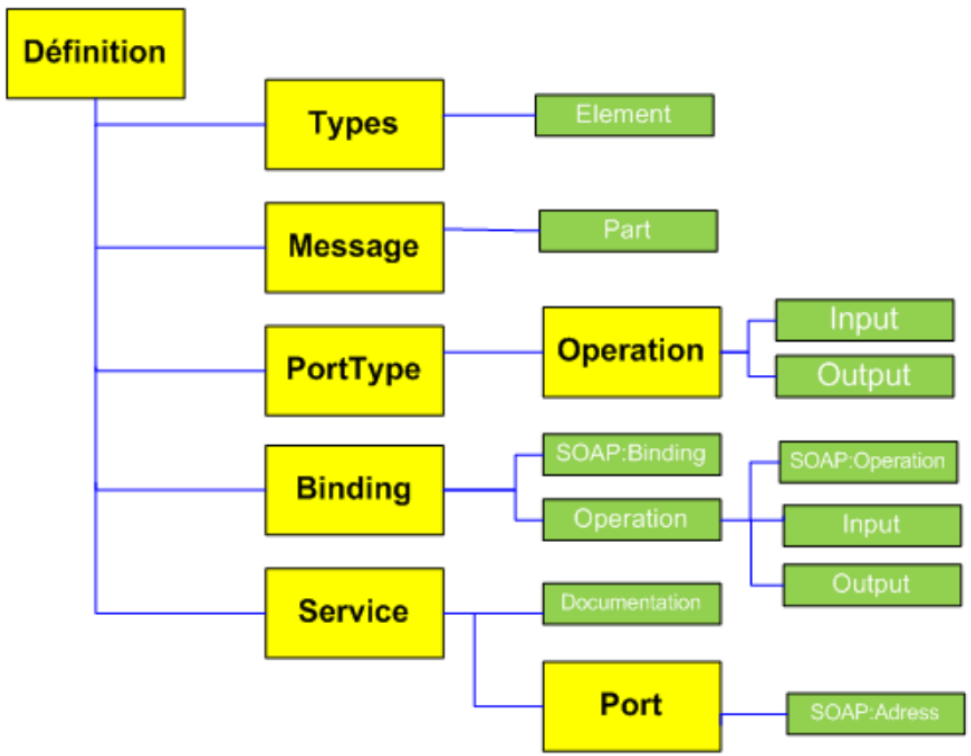
↳ Publication et recherche de services au sein de l'annuaire se font via les documents WSDL.

↳ Pour l'accès à un service particulier, un client se voit retourné l'URL du fichier WSDL décrivant l'implémentation du service.

Structure du langage WSDL

Un fichier WSDL contient donc sept éléments inclus dans la racine ***definitions*** .

- ***Types*** : fournit la définition de types de données utilisés pour décrire les messages échangés.
- ***Messages*** : représente une définition abstraire (noms et types) des données en cours de transmission.
- ***PortTypes*** : décrit un ensemble d'opérations.
- ***Binding*** : spécifie une liaison entre un <portType> et un protocole concret (SOAP, HTTP...). c.à.d. décrit la façon dont un ensemble d'opérations abstraites, appelé (type de port) est lié à un port selon un protocole réel.
- ***Service*** : indique les adresses de port de chaque liaison.



- ***Port*** : représente un point d'accès de services défini par une adresse réseau et une liaison.
- ***Opération*** : c'est la description d'une action exposée dans le port.

Structure d'un fichier WSDL

```
1:<definitions xmlns = "http://schemas.xmlsoap.org/wsdl/  "
2:              xmlns :xsd = http://www.w3.org/2001/XMLSchema ... >
3:    <types>                                  }
4:      […]                                    }
5:    </types>                                 }
6:    <message […]>                            }
7:        <part […]/>                          }
8:    </message >                              }  Interface du service (Fonctionnelle)
9:    <porttype […]>                           }
10:     <operation […]>                        }
11:             <input […] />                  }
12:             <output […] />                 }
13:     </operation>                           }
14:   </porttype>                              }
15:   <binding[…]>                             }
16:     […]                                    }
17:   </binding>                               }
18:   <service […]>                            }  Information techniques
19:     <port […]>                             }
20:       […]                                  }
21:     </port>                                }
22:   </service >                              }
23:</definitions >
```

Remarque :

Le document WSDL peut être divisé en deux parties :

1) Une partie pour les **définitions abstraites** : est composée des éléments qui sont orientés vers la description des capacités du service Web (`<wsdl:types>`, `<wsdl:message>`, `<wsdl:portType>` et `<wsdl:operation>`).
2) La deuxième contient les **descriptions concrètes** : Contient des éléments qui sont orientés vers le client pour le service physique (`<wsdl:service>`, `<wsdl:port>` et `<wsdl:binding>`).

Exemple : sommer.wsdl

```
1  <?xml version="1.0"?>
2  <definitions targetNamespace="http://localhost:8080/axix/sommer.jws"
3               xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
4               xmlns="http://schemas.xmlsoap.org/wsdl/">
5      <!-- definition des type de données la balise types -->
6      <wsdl:message name="getsommeRequest">
7          < wsdl:part name="a" type="xsd:int"/>
8          < wsdl:part name="b" type="xsd:int"/>
9      </wsdl:message>
10     <wsdl:message name="getsommeResponse">
11         < wsdl:part name="getsommereturn" type="xsd:int"/>
12     </wsdl:message>
13     <wsdl:portType name="sommer">
14         <wsdl:operation name="getsomme"  parameterOrder = "a b">
15             <wsdl:input message="impl:getsommeRequest" name ="getsommeRequest "/>
16             <wsdl:output message="impl:getsommeResponse" name ="getsommeResponse"/>
17         </wsdl:operation>
18     </wsdl:portType>
19     <wsdl:binding name="sommerSoapBinding" type="impl:sommer">
20         <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
21         <wsdl:operation name="getsomme">
22             <wsdlsoap:operation soapAction=""/>
23             <wsdl:input name= "getsommeRequest">
24                 <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
25                 namespace="http://DefaultNamespaceexample.com/stockquote.xsd"
26                 use="encoded" />
27             </wsdl:input>
28             <wsdl:output name= "getsommeResponse">
29                 <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
30                 namespace="http://localhost:8080/axis/sommer.jws"
31                 use="encoded" />
32             </wsdl:output>
33         </wsdl:operation>
34     </wsdl:binding>
35     <wsdl:service name="sommerService">
36         <wsdl:documentation> My first service</ wsdl:documentation>
37         <wsdl:port binding="impl:sommerSoapBinding" name="sommer">
38             <wsdlsoap:address location="http://localhost:8080/axis/sommer"/>
39         </wsdl:port>
40     </wsdl:service>
41 </definitions>
```

- Racine du document
- Définition des types de données (facultatif)
- Message : définition des messages échangeables
- PortType : définition des ensembles d'opérations
- Liaison
- Service : localisation des services web

L'élément <definition … >

→ L'élément racine dans un document WSDL est `<wsdl:definition>`.

→ Il contient un attribut `targetNamespace` qui définit un certain nombre d'espaces de noms `namespace` auquel tous les noms déclarés dans un élément du document WSDL appartiennent, ce qui permet d'éviter les conflits de nommage.

```
<definitions targetNamespace="http://localhost:8080/axix/sommer.jws"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns="http://schemas.xmlsoap.org/wsdl/">
```

Dans cet exemple, l'attribut `targetNamespace` a pour valeur l'URL : «`http://localhost:8080/axix/sommer.jws`». Cela signifie que tous les noms déclarés dans ce document WSDL appartiennent à cet espace de noms.

Rappelle

— Un espace de noms est identifié par une URI (Uniform Resource Identifier) qui permet de l'identifier de manière unique.

— La déclaration d'un espace de noms par défaut se fait dans le premier élément qui utilise le vocabulaire, grâce au mot clef `xmlns` comme XML namespace. Exemple : `xmlns="mon_uri"`

L'élément <type … >

→ L'élément <types> décrit la structure de données transmises dans un message.

→ Contient les définitions de types en utilisant un système de typage par défaut XML Schema (XSD).

→ Pouvant contenir des types simples et complexes

L'élément <message … >

→ Décrit les données associées à une opération (1 requête et 1 réponse HTTP par opération, 1 message d'erreur "fault" optionnel).

→ Un document WSDL peut contenir zéro ou plusieurs messages.

→ Chaque message peut être composé de plusieurs parties.

```
06: <wsdl:message name="getsommeRequest">
07:      < wsdl:part name="a" type="xsd:int"/>
08:      < wsdl:part name="b" type="xsd:int"/>
09: </wsdl:message>
10: <wsdl:message name="getsommeResponse">
11:      < wsdl:part name="getsommereturn" type="xsd:int"/>
12: </wsdl:message>
```

L'élément <portType … >

→ Un document WSDL peut contenir 0 à plusieurs `portType`

→ L'élément `portType` contient un seul attribut `name`.

✍ La convention de nommage `nameOfWebService` PortType.

→ Composé d'un ensemble d'opérations abstraites (i.e. signature de la méthode).

→ Une opération est composée d'un message pour l'appel (Input) et un pour le retour (Output).

```
13: < wsdl:portType name="sommer">
14:    <wsdl:operation name="getsomme"  parameterOrder = "a b">
15:        <wsdl:input message="impl:getsommeRequest" name ="getsommeRequest "/>
16:        <wsdl:output message="impl:getsommeResponse" name ="getsommeResponse"/>
17:    </wsdl:operation>
18: </wsdl:portType>
```

Remarque

> Définition d'un seul type de port, avec les opérations abstraites, correspondant aux déclarations de méthodes dans l'interface Java.

L'élément <binding … >

→ Une liaison (ou `binding`) décrit la façon dont un `portType` (en d'autres termes l'abstraction du service, i.e. ses opérations abstraites) est mis en œuvre pour un protocole particulier (HTTP par exemple) et un mode d'invocation (RPC par exemple).

→ Pour un `portType`, on peut avoir plusieurs liaisons, pour différencier les modes d'invocation (RPC ou autres) ou de transport (HTTP ou autre) des différentes opérations.

Mode d'invocation

Protocole

```
19: < wsdl:binding name="sommerSoapBinding" type="impl:sommer">
20: < wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
21: < wsdl:operation name="getsomme">
22:    < wsdlsoap:operation soapAction=""/>
23:    < wsdl:input name= "getsommeRequest">
24:        < wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                           namespace="http://DefaultNamespaceexample.com/stockquote.xsd"
                           use="encoded" />
27:    </wsdl:input>
28:    < wsdl:output name= "getsommeResponse">
29:        < wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                           namespace="http://localhost:8080/axis/sommer.jws"
                           use="encoded" />
32:    </wsdl:output>
33: </ wsdl:operation>
34: </ wsdl:binding>
```

Nom de l'opération dans le type de port

La représentation du message request

La représentation du message response

→ Une liaison WSDL décrit comment le service est lié à un protocole, généralement le protocole de SOAP.

→ Une liaison WSDL SOAP peut être une liaison de style Remote Procedure (RPC) ou une liaison de style document.

→ Le style RPC spécifie que le <soap: body> contient un élément avec le nom de la méthode Web appelée. Cet élément contient à son tour une entrée pour chaque paramètre et la valeur de retour de cette méthode.

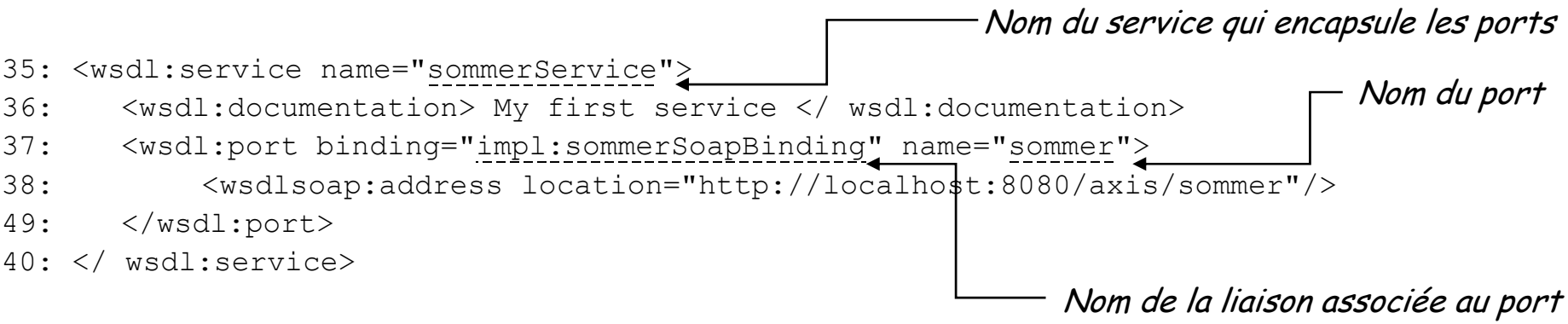
→ Avec le style de document, les parties de message apparaissent directement sous l'élément <soap: body>.

L'élément < service … >

→ Un service est décrit comme un ensemble de points finaux du réseau appelés « ports »

→ Un port spécifie une URL qui correspond à l'implémentation du service par un fournisseur.

→ Le port est associé à un « binding » définissant ainsi un simple point de terminaison (endpoint:@ où se situe le WS)



5. UDDI (Universal Description, Discovery and Integration)

→ UDDI est une spécification définissant la manière de publier et de découvrir des informations sur les services Web.

→ La description du service est faite en XML selon la spécification UDDI.

→ spécification UDDI est une initiative lancée par ARIBA, Microsoft et IBM.

→ Versions :

- Version 1: les bases d'un annuaire de services.
- Version 2: adaptation à SOAP et WSDL.
- Version 3: redéfinition du rôle UDDI, accent sur les implémentations privées, sur l'interaction entre annuaires privés et publics.

→ L'annuaire UDDI permet de :

- Décrire et Publier,
- Découvrir et Intégrer

} des informations sur une entreprise et ses services

→ L'inscription sur UDDI permet à une entreprise de se présenter ainsi que ses services.

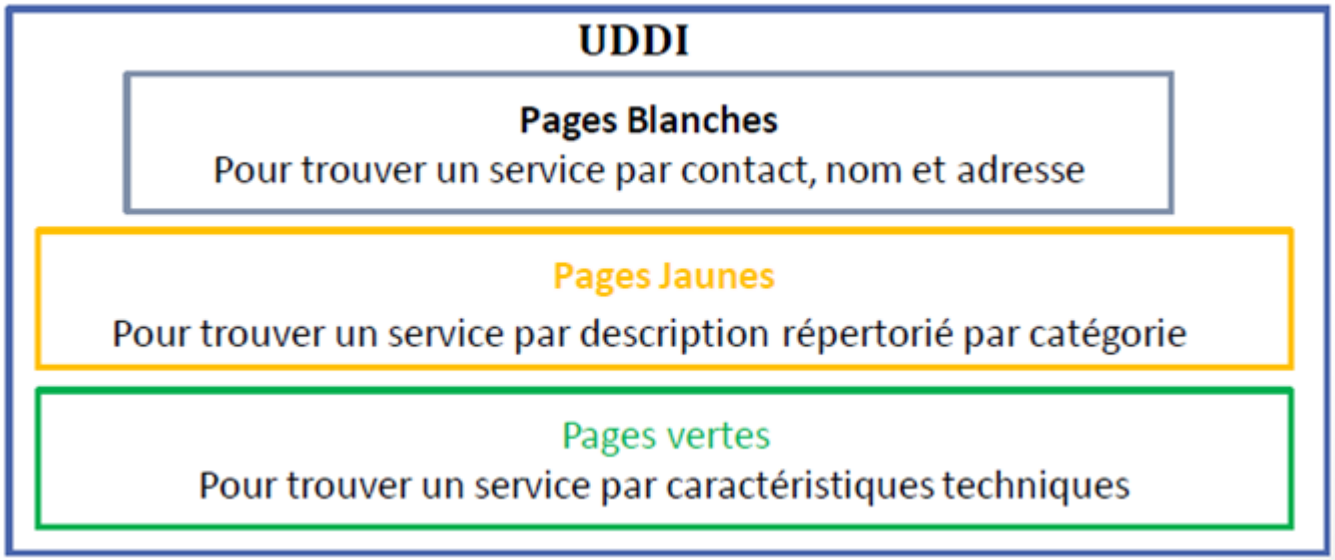
→ L'enregistrement des services dans un annuaire s'effectue auprès d'un opérateur (Microsoft ou IBM actuellement) à travers son site mais on peut créer ses propres registres UDDI

→ Exemple : Un annuaire à l'aide d'un browser en ligne: http://soapclient.com/UDDIAdv.html

Consultation de l'annuaire

L'accès à l'annuaire des services UDDI s'effectue de différentes manières :

- *Les pages blanches :* comprennent la liste des entreprises ainsi que des informations associées à ces dernières (coordonnées, description de l'entreprise, identifiants...).
- *Les pages jaunes :* recensent les services Web de chacune des entreprises.
- *Les pages vertes :* fournissent des informations techniques précises sur les services fournis.



Structure de données UDDI

Un registre UDDI se compose de quatre types de structures de données :

- **businessEntity** : organisation qui offre le service.
- **businessService** : liste des services web offerts par l'organisation.
- **bindingTemplate** : informations sur les liaisons (aspects techniques du service offerts)
- **tModel** : informations décrivant les spécifications relatives aux services (élément générique pour info supplémentaire sur le service)

Cette répartition par type fournit des partitions simples pour faciliter la localisation rapide et la compréhension des différentes informations qui constituent un enregistrement d'un service Web .

BusinessEntity

→ Les « businessEntities » sont en quelque sorte les pages blanches d'un annuaire UDDI.

→ Elles décrivent les organisations ayant publié des services dans le répertoire :

- Tel que : le nom de l'organisation, ses adresses (physiques et Web), … .

BusinessService

→ Les « businessServices » sont en quelque sorte les pages jaunes d'un annuaire UDDI.

→ Elles décrivent de manière non technique les services proposés par les différentes organisations :

- Tel que : le nom et la description textuelle des services, … .

BindingTemplate

→ Ce sont les pages vertes de l'annuaire UDDI.

→ La structure bindingTemplate détient des informations techniques servant à déterminer le point d'entrée (i.e. adresse du fournisseur) et les spécifications de construction pour l'appel d'un service Web.

→ Elle fournit les descriptions de service Web utiles aux développeurs d'applications souhaitant rechercher et appeler un service Web.

→ La structure bindingTemplate pointe vers des descriptions d'implémentation d'un service (*tModels*), par le biais d'une adresse URL, par exemple.

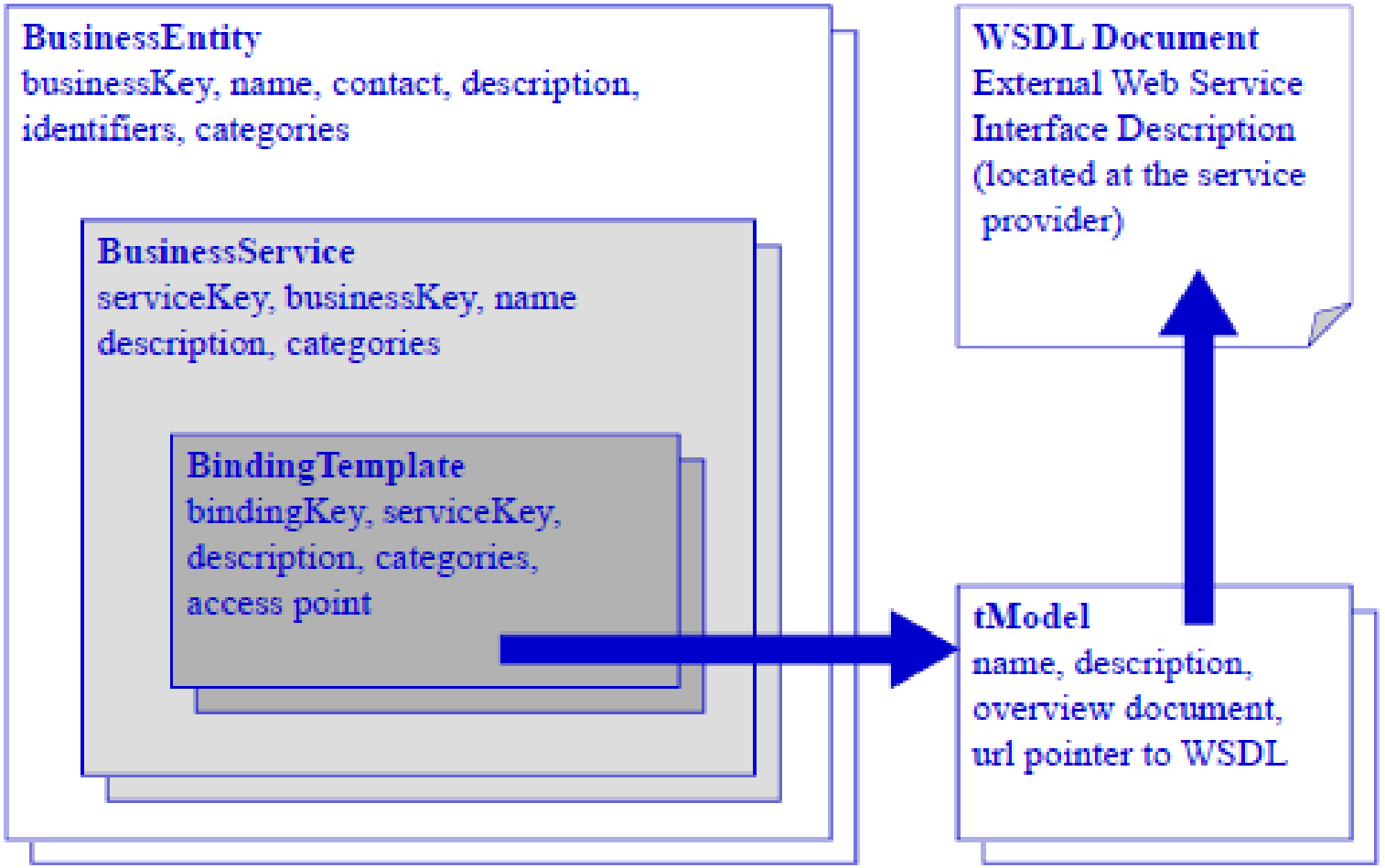
tModel

→ Les « tModels » sont les descriptions techniques des services.

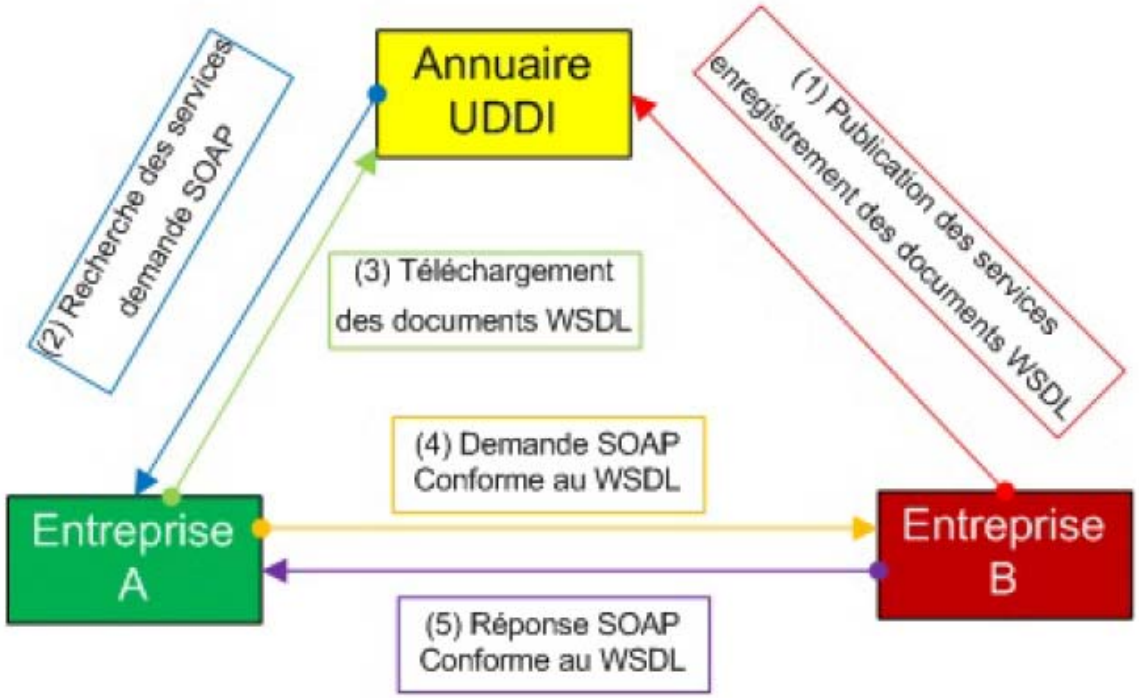
→ UDDI n'impose aucun format pour ces descriptions qui peuvent être publiées sous n'importe quelle forme et notamment sous forme de documents textuels.

→ C'est à ce niveau que WSDL intervient comme le vocabulaire de choix pour publier des descriptions techniques de services.

Structure de données UDDI, vue d'ensemble



Le scénario classique d'utilisation de UDDI : Revenons à l'architecture



1) Les entreprises publient les descriptions de leurs services Web en UDDI, sous la forme de fichiers WSDL.
2) les clients peuvent plus facilement rechercher les services Web dont ils ont besoin en interrogeant le registre UDDI.
3) Lorsqu'un client trouve une description de service Web qui lui convient, il télécharge son fichier WSDL depuis le registre UDDI.
4) Ensuite, à partir des informations inscrites dans le fichier WSDL, notamment la référence vers le service Web, le client peut invoquer le service Web et lui demande d'exécuter certaines de ses fonctionnalités.

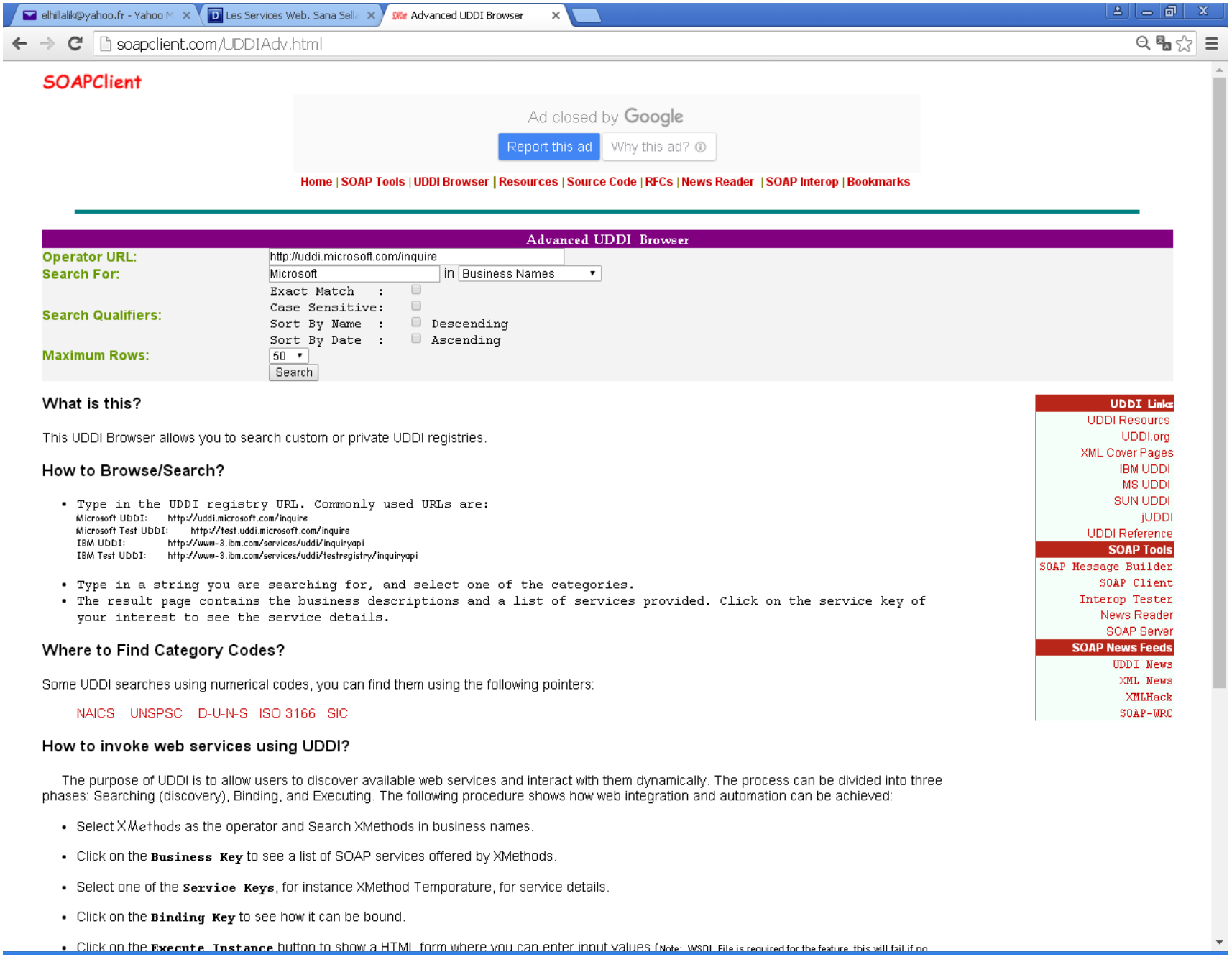
Remarque

Un registre UDDI peut être accédé en SOAP

Publication de service Web

→ La publication d'un service Web requiert que l'entreprise s'authentifie auprès de l'opérateur UDDI

→ Il faut fournir les données nécessaires à l'exploitation du service Web (IP, Nom de domaine, modalités d'utilisation, … .)


elhillalik@yahoo.fr - Yahoo M
Les Services Web. Sana Sell
Advanced UDDI Browser
soapclient.com/UDDIAdv.html
SOAPClient
Ad closed by Google
Report this ad
Why this ad?
Home | SOAP Tools | UDDI Browser | Resources | Source Code | RFCs | News Reader | SOAP Interop | Bookmarks
Advanced UDDI Browser
Operator URL:
http://uddi.microsoft.com/inquire
Search For:
Microsoft
in
Business Names
Exact Match :
Case Sensitive:
Search Qualifiers:
Sort By Name :
Descending
Sort By Date :
Ascending
Maximum Rows:
50
Search
What is this?
This UDDI Browser allows you to search custom or private UDDI registries.
How to Browse/Search?
Type in the UDDI registry URL. Commonly used URLs are:
Microsoft UDDI: http://uddi.microsoft.com/inquire
Microsoft Test UDDI: http://test.uddi.microsoft.com/inquire
IBM UDDI: http://www-3.ibm.com/services/uddi/inquiryapi
IBM Test UDDI: http://www-3.ibm.com/services/uddi/testregistry/inquiryapi
Type in a string you are searching for, and select one of the categories.
The result page contains the business descriptions and a list of services provided. Click on the service key of your interest to see the service details.
Where to Find Category Codes?
Some UDDI searches using numerical codes, you can find them using the following pointers:
NAICS UNSPSC D-U-N-S ISO 3166 SIC
How to invoke web services using UDDI?
The purpose of UDDI is to allow users to discover available web services and interact with them dynamically. The process can be divided into three phases: Searching (discovery), Binding, and Executing. The following procedure shows how web integration and automation can be achieved:
Select XMethods as the operator and Search XMethods in business names.
Click on the Business Key to see a list of SOAP services offered by XMethods.
Select one of the Service Keys, for instance XMethod Temporature, for service details.
Click on the Binding Key to see how it can be bound.
UDDI Links
UDDI Resources
UDDI.org
XML Cover Pages
IBM UDDI
MS UDDI
SUN UDDI
jUDDI
UDDI Reference
SOAP Tools
SOAP Message Builder
SOAP Client
Interop Tester
News Reader
SOAP Server
SOAP News Feeds
UDDI News
XML News
XMLHack
SOAP-WRC