République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Abbess Laghrour Khenchela
Faculté Des Sciences Et De La Technologie
Département De Mathématiques Et Informatique

Cours Services Web

Dr. Hemam Sofiane Mounine

Novembre 2015

Table des matières

Introduction Générale

L'Architecture Orientée Services (AOS ou SOA – *Service-Oriented Architecture* en anglais) permet aux concepteurs de systèmes d'information d'organiser un ensemble de logiciels isolés en un ensemble de services interconnectés, accessibles par une interface et des protocoles standard. Les services web représentent la technologie la plus exploitée et la plus utilisée pour la fourniture de services, sous forme d'applications, sur Internet. L'intérêt des services web est de permettre à une entreprise d'exporter au travers du réseau internet ses compétences et son savoir-faire, d'interagir avec ses partenaires, de rechercher de nouveaux marchés et de nouveaux supports de vente. Contrairement aux interfaces de programmation (Application Programming Interface, API) « classiques », les services web sont conçus pour découvrir et invoquer d'autres services et tirent leur versatilité de leurs interfaces qui sont des abstractions n'imposant aucune contrainte en matière de mise en œuvre, e.g., langage de programmation, système d'exploitation, etc. Actuellement, les services web reposent principalement sur des standards XML, WSDL (Web Services Description Language) qui permet une description syntaxique des services en termes d'entrées, sorties et d'un répertoire de services UDDI (Universal Description, Discovery and Integration). Les services web s'appuient sur SOAP (Simple Object Access Protocol), un protocole d'échange de messages entre services qui est fondé sur HTTP.

Ce cours s'adresse aux étudiants de deuxième année Master Informatique spécialité « Sécurité et Technologie Web ». Il représente un pré-requis pour la matière Web Services et Sécurité du Troisième semestre, il est constitué de cinq chapitres qui peuvent être résumés comme suit :

Le premier chapitre sera consacré aux concepts généraux d'un Web Service, tout en évoquant les limites des Middlewares et les motivations d'utilisation des Web Services. Le langage XML avec des exemples sera présenté dans le deuxième chapitre, alors que le standard Langage de Description des Web Services :WSDL basé sur le XML sera détaillé dans le chapitre 3 suivi d'un exemple concret. Quant à l'annuaire de Web Service : UDDI, il sera présenté dans le quatrième chapitre de ce polycopié. Finalement, le cinquième chapitre sera dédié au protocole de communication et d'échange de messages SOAP.

I. Introduction

La généralisation de l'usage des réseaux Internet a amené le développement des interactions entre le système d'information d'une entreprise avec l'extérieur. Un des bénéfices immédiats est de permettre aux collaborateurs en déplacement d'accéder à certaines applications. Par exemple un représentant commercial va pouvoir interroger à distance les stocks, passer une commande et synchroniser ses contacts pour son assistant digital. Mais les bénéfices ne s'arrêtent pas là. Les transactions de type B2B et B2C se sont fortement développées. De manière très concrète cela passe par la mise à disposition d'applications par un serveur web. L'interface homme machine est alors constituée d'un simple navigateur web. Ainsi un partenaire peut effectuer à l'aide d'un accès privilégié un certain nombre d'actions qu'il aurait sinon effectuées par des moyens 'moins efficaces', tels l'appel téléphonique. De même on peut imaginer qu'un client va pouvoir consulter l'offre de l'entreprise et passer des commandes.

Les Web services sont un paradigme naissant qui vise à la transposition des architectures par composant dans le cadre du Web. Un Web service est un composant logiciel qui offre des services à travers une interface standardisée. La particularité des Web services réside dans le fait qu'elle utilise la technologie Internet comme infrastructure pour la communication entre les composants logiciels (les Web services) et ceci en mettant en place un cadre de travail basé sur un ensemble de standards.

II. Définitions

1. Middleware

Un middleware (Intergiciel) est un logiciel (ou un ensemble de logiciels ou de technologies informatiques) médiateur permettant à deux ou plusieurs applications réparties dans un réseau de communiquer entre eux. Il assure le dialogue entre différentes applications ou portions d'une même application réparties sur plusieurs postes, clients ou serveurs. Il existe différents types de middlewares : les moniteurs transactionnels (CICS, Tuxedo…), les messageries inter-applicatives asynchrones (MQSeries…), les ORB (VisiBroker, Orbix…), les appels de procédure à distance (RPC) et les **middlewares** d'accès aux données (ODBC, EDA/SQL…).

- Couche logicielle intermédiaire entre les applications et le réseau, permettant le dialogue entre des applications hétérogènes. Synonyme : Intergiciel.
- Logiciel standard qui permet à deux autres logiciels de communiquer en utilisant un protocole applicatif. Dans un Centre d'appels, prend en charge les communications entre les différents constituants : ACD, SVI, applicatif...
- Un middleware permet la communication entre des clients et des serveurs ayant des structures et une implémentation différentes.

Il permet l'échange d'informations dans tous les cas et pour toutes les architectures. Enfin, le middleware doit fournir un moyen aux clients de trouver leurs serveurs, aux serveurs de trouver leurs clients et en général de trouver n'importe quel objet atteignable.

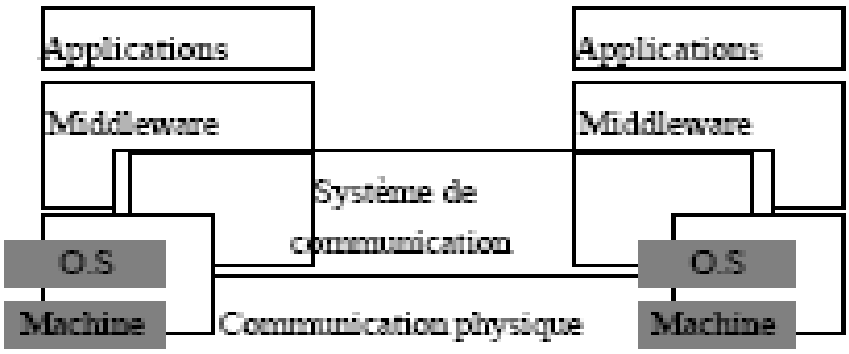


Figure 01: Les Middlewares

2. Les SOA ou les architectures orientées services

Les besoins de SOC pour la satisfaction des précédents cas d'usage peuvent être plus facilement satisfaits à travers une architecture répondant aux propriétés requises. Une telle architecture est appelée SOA ou architecture orientée services.

> ***Une architecture orientée services**, calque de l'anglais Service Oriented Architecture, ou SOA, est une architecture logicielle s'appuyant sur un ensemble de services simples. Son objectif est de décomposer une fonctionnalité en un ensemble de fonctions basiques (les services) fournies par des composants et de décrire le schéma d'interaction entre ces services.*

Dans une SOA, les services peuvent communiquer entre eux. Cette communication peut soit consister en un simple passage de données ou impliquer la coordination de deux ou plusieurs services pour l'accomplissement d'une activité. Il faut alors définir des moyens pour connecter les services entre eux.

Quelques remarques.

1. Beaucoup considèrent que la première SOA est apparue avec l'utilisation des DCOM (Distributed Component Object Model) ou ORB (Object Request Brokers) basés sur les spécifications de CORBA.
2. La principale différence entre les SOA et les autres architectures distribuées, telles que CORBA, est le faible couplage des services par l'expression des "interactions" entre les services. En effet, le point de départ des spécifications SOA a été le besoin croissant de sélectionner et d'intégrer, au fil de l'eau, des services hétérogènes et inter-organisationnels, aussi bien à travers le web qu'au sein d'environnements intelligents et pervasifs. Toutefois, là où CORBA se concentre sur les objets pour créer un environnement de programmation distribué, SOA se focalise sur les documents et l'interopératibilité des processus métiers entre partenaires et consommateurs à travers l'Internet moyennant des transactions à long terme (et pas seulement des transactions à court terme).

3. Service Web

Les Web Services sont des services offerts via le web. Par exemple, un client demande le prix d'un article en envoyant un message sur le web. Ce message contient la référence de l'article. Le Web Service va recevoir la référence, effectuer le traitement du service et renvoyer le prix au client via un autre message.

IBM donne la définition suivante des web services :

" Les web services sont la nouvelle vague des applications web. Ce sont des applications modulaires, auto contenues et auto descriptives qui peuvent être publiées, localisées et invoquées depuis le web. Les web services effectuent des actions allant de simples requêtes à des processus métiers complexes. Une fois qu'un web service est déployé, d'autres applications (y compris des web services) peuvent le découvrir et l'invoquer."

Les web services permettent d'intégrer des systèmes d'information hétérogènes en utilisant des protocoles et des formats de données standardisés, autorisant ainsi un faible couplage et une grande souplesse vis-à-vis des choix technologiques effectués. . Pour cela les normes de communication associées aux Web services sont standardisées et entièrement spécifiées en XML, langage permettant une grande interopérabilité entre les systèmes.

4. Caractéristiques des Services Web

Le consortium W3C définit un service Web comme étant une application ou un composant logiciel qui vérifie les propriétés suivante :

- Il est identifié par un URL: qui est une façon d'identifier un contenu web. L'"URI le plus reconnue est l'adresse d'une page.
- Ses interfaces et ses liens peuvent être décrits en XML.
- Capacité d'interagir avec d'autres éléments : Cela est permis grâce à l'utilisation du langage XML et les Protocol es standards d'Internet (HTTP, SMTP…etc.).
- Composante logicielle légèrement couplée à interaction dynamique.

5. Infrastructures des Web Services

Bien que le Web soit constitué de plates-formes totalement hétérogènes où les intérêts des différents acteurs du marché s'entremêlent, ceci ne l'a pas empêché de se développer et d'être universel. Ce succès est dû essentiellement à l'édition d'un ensemble de standards ouverts, dont les plus connus sont le protocole http (HyperText Transfer Protocol) et le format MIME (Multipurpose Internet Mail Extension). Ensemble, ils offrent un mécanisme d'échange de données de toute sorte quelque soit la nature des plates-formes impliquées. Le développement du modèle services Web a suivi la même approche en mettant en place une compagne de standardisation qui a touché les aspects les plus importants du développement d'un modèle d'intégration d'applications hétérogènes.

La pile des web services est sujette à de nombreuses discussions. Les web services représentent un domaine de recherche jeune et nombre d'acteurs de l'industrie essaient d'imposer leur vision calquée sur les solutions qu'ils proposent. Il est donc assez compréhensible que la pile des web services ne trouve pas encore de consensus quand à sa définition standardisée. La figure 1 est une proposition de vue globale

simplifiée de la pile des web services. On trouve dans les couches les plus basses les protocoles de transport (HTTP, SMTP) ainsi que le moyen d'encoder les données (XML). Au dessus on trouve un moyen d'échanger des messages (SOAP) ainsi qu'un moyen pour définir l'interface statique d'un service (WSDL). Enfin on a un outil de découverte et localisation (UDDI) ainsi que des moyens de spécifier le comportement externe (dynamique) des services (WSCI, WSCL, BPEL4WS).

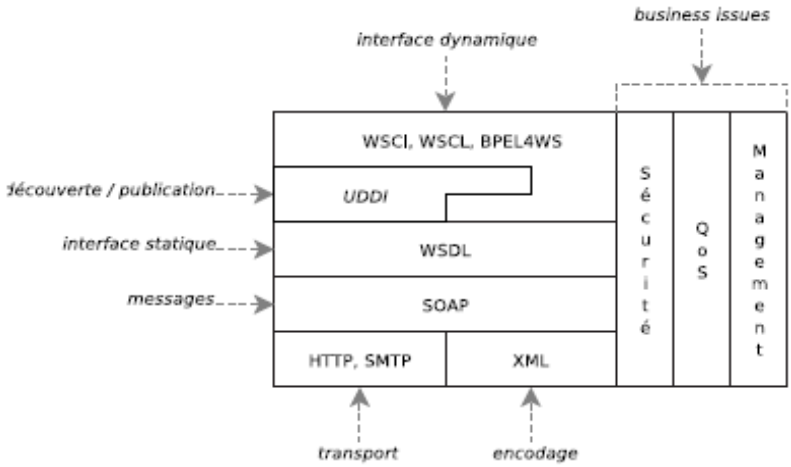


Figure 2 : La pile simplifiée des Web Service

5.1. 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 (Tableau II.1) :

- Le transport de messages XML-RPC ou SOAP est assuré par le standard HTTP.
- SOAP ou XML-RPC prévoit la couche de communication basée sur XML pour accéder à des services Web.
- La description d'un service Web se fait en utilisant le langage WSDL. WSDL expose l'interface du service.
- La publication et la découverte des services Web sont assurées par le biais du référentiel UDDI. Un référentiel UDDI est un catalogue de services Web.

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

Tableau I.1 : Différentes couches technologiques des services web

5.1.1. Couche transport

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

5.1.2. Couche communication

Cette couche est responsable du formatage des données échangées de sorte que les messages peuvent être compris à chaque extrémité. Actuellement, deux styles architecturaux totalement différents sont utilisés pour ces échanges de données.

Nous avons d'un côté l'architecture orientée opérations distribuées (protocoles RPC) basée sur XML et qui comprend XML-RPC et SOAP et de l'autre côté une architecture orientée ressources Web, REST (Representational State Transfer) qui se base uniquement sur le bon usage des principes du Web (en particulier, le protocole HTTP).

5.1.3. 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. Cette spécification définit une grammaire XML pour décrire les services Web comme des ensembles de points finaux de communication (ports) à travers lesquels on effectue l'échange de messages.

5.1.4. Couche publication 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, Discrovery, and Integration).

III. Principes des Web Services

1. Utilité d'un nouveau middleware

- Passage à large échelle : Web
- Protocoles hétérogènes
  * 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

2. Limitations des middlewares

Inconvénients Intrinsèques

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

Solutions existantes

- Modification du Protocole
  * RMI / IIOP
- Passerelles
  * CORBA vers DCOM
- Portage d'applications existantes difficile
- Solutions non standards

Approche Envisagée

- Un nouveau Protocole : SOAP
  * Basé sur XML
- Portabilité, Hétérogénéité
  * Porté sur des protocoles large échelle existants
- HTTP, SMTP, …
- Paradigme orienté service : WSDL
  * Définition de services offerts (en XML)
- Découverte automatique des services
(dynamicité) : UDDI
  * Référentiel de Web Service (Pages Jaunes, Vertes, Blanches)

➔ **Vers les WEB SERVICES**

3. Avantages et inconvénients des web Services

3.1. Avantages

La large implication des industriels dans l'élaboration des normes à tous les niveaux témoignent de l'ambition collective de mieux faire communiquer les systèmes informatiques par le biais du Web. Ainsi les web services permettent d'interfacer des systèmes d'information hétérogènes avec pour atouts :

- Un faible couplage avec les technologies employées en interne.
- Une grande flexibilité de mise à jour des systèmes employés de part et d'autre
- L'emploi de protocoles réseau simples, peu chers, répandus et bénéficiant d'implémentations dans toutes les technologies majeures.
- L'encodage des données sous forme de documents XML, facilement analysables par un composant logiciel et compréhensible pour un être humain.

3.2. Inconvénient

Les services web comporte des inconvénients dont :

- Les normes de services web dans certains domaines ne sont pas encore "mûres".
- Souffrent de performances faibles.

- L'utilisation du protocole HTTP augmente la possibilité pour les services web de contourner Les mesures de sécurité (firewalls).

Le langage XML
eXtensible Markup Language

I. Introduction

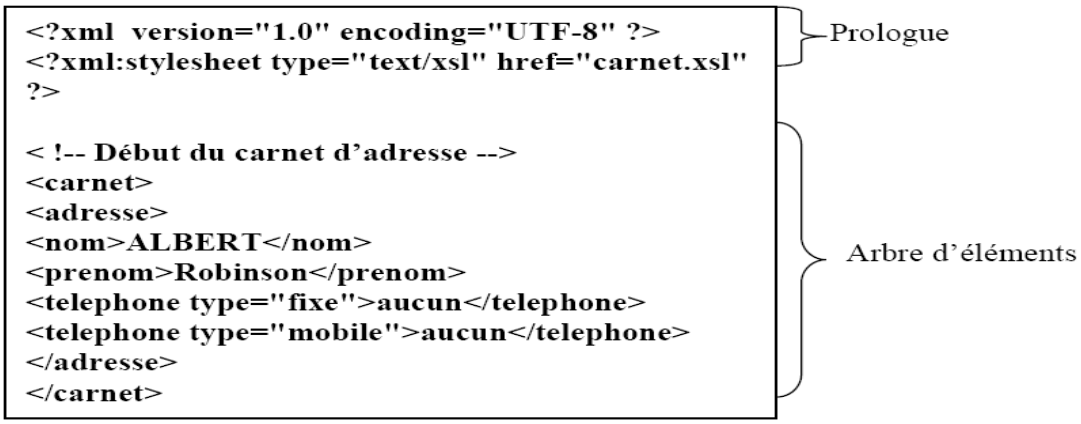
L'aspect le plus important des Web Services est qu'ils reposent sur plusieurs standards qui permettent la structuration des architectures. Cette collection de normes et de protocoles est appelée Web Services Protocol Stack (voir chapitre I). Elle contient entre autre XML et SOAP pour le formatage des données, WSDL pour la description des services Web et UDDI pour la recherche des services Web nécessaire au bon fonctionnement des applications. Donc, le XML est a la base de ces différents standards.

XML (pour *eXtensible Markup Language*) est un ensemble de règles pour la conception de fichiers textes permettant de structurer des données sous forme arborescente. XML facilite la réalisation de fichiers qui ne soient pas ambigus, et qui évitent les pièges courants, tels que la non extensibilité et la dépendance par rapport à certaines plates-formes. Pour cela XML utilise des balises analogues aux balises HTML; ces balises sont des mots encadrés par des < et des > et elles sont susceptibles de comporter des attributs sous la forme nom_attribut="valeur". Mais alors qu'en HTML ces balises sont prédéfinies et non extensibles, les balises XML sont là seulement pour délimiter les éléments de données. Une balise <p> peut aussi bien encadrer un paragraphe dans un fichier XML destiné à afficher une page Web qu'encadrer un prix dans un catalogue de vente par correspondances, au choix de son utilisateur.

***Objectif :* structurer l'information pour permettre son traitement par un ordinateur.**

II. Structure d'un document XML

Un document XML est constitué de deux parties : le ***prologue*** et ***l'arbre d'éléments***. On utilise cette terminologie car tout ensemble de tags XML peut être représenté sous forme ***arborescente***. Exemple :



1. Prologue d'un document XML

Le prologue n'est pas obligatoire, mais vivement recommandé (de part la recommandation XML 1.0). Il contient des informations utiles pour le traitement des données qui y sont contenues. Il est, de plus, subdivisé en plusieurs sous parties.

a) Version du XML et système d'encodage

- La première ligne sert, principalement, à deux choses. Premièrement, via l'attribut **version**, elle permet d'indiquer la version du langage XML utilisé. Pour l'heure, il n'y a pas réellement de difficulté à ce niveau, pour la simple et bonne raison qu'il n'existe qu'une seule et unique version de la recommandation : la version 1.0.

- Sa seconde grosse utilité est de pouvoir spécifier quel est le système d'encodage ayant servit à générer le fichier. Exemple : ISO-8859-1 pour prendre en considérations les caractères accentués.

b) Liaison à une feuille de style

L'une des finalités de notre document XML est certainement de se présenter dans un navigateur. Pour ce faire, il faut le lier à une feuille de style. Pour le faire, deux langages peuvent être utilisés : CSS (Cascading StyleSheet) ou XSL (eXtensible Stylesheet Language).

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<?xml:stylesheet type="text/css" href="Personnes.css" ?>
<PERSONNE>
   <NOM>Durand</NOM>
   <PRENOM>Gérard</PRENOM>
</PERSONNE>
```

c) Liaison à une DTD

Cette liaison se réalise aussi au niveau du prologue, mais comparé aux lignes précédemment étudiées, la syntaxe diffère quelque peu. Il est aussi à noter qu'une DTD peut directement être embarquée dans le fichier de données. Dans ce cas, la DTD ne sert que pour cet unique fichier.

L'exemple qui suit nous montre comment la DTD est liée au fichier de données. Une remarque importante est à faire : **le nom associé à la DTD doit être exactement identique au nom du tag racine, sans quoi une erreur nous sera retournée**.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<?xml:stylesheet type="text/css" href="Personnes.css" ?>
<!DOCTYPE PERSONNE SYSTEM "Personnes.dtd" [] >
<PERSONNE>
   <NOM>Durand</NOM>
   <PRENOM>Gérard</PRENOM>
</PERSONNE>
```

2. L’arbre d’éléments d’un document XML

- L’élément fondamental dans un document XML est la **balise** : **<balise>**contenu**</balise>.**
- Forme courte : **<balise />** pour les balises vides**.**
- Les **attributs** servent à donner des informations supplémentaires sur une balise ou son contenu. <balise **attribut="valeur"**>contenu</balise>

Exemple :

<total devise="euros">11,08</total>
<total devise="FRF">72,68</total>

- Les **entités** servent à encoder des caractères spéciaux :

- **< = <**
- **>= >**
- **& = &**
- **" = "**
- **’ = '**

- Les **commentaires** servent à clarifier le code: < !-- Texte de commentaire -->

a) Règles du langage XML

1. Le document doit contenir un unique élément racine.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<lettre>
        <date>17/10/2000</date>
        <objet>Réclamation</objet>
        <politesse>Madame</politesse>
        <para>En réponse à votre lettre du <date>15/12/1998</date>
          dans laquelle Monsieur <name>Ybroc</name> ...
        </para>
        <salutation/>
</lettre>
```

2. Les attributs doivent avoir une valeur (éventuellement vide) et celle-ci doit être entre guillemets simples ou doubles. ***Exemples*** :

<total **devise="euros"**>11,08</total>
<total **devise=’FRF’**>72,68</total>
<total **devise=""**>8215,30</total>

Contre-exemples :

<total **devise=euros**>11,08</total>

<total **devise=**>72,68</total>

3. Les balises ne doivent pas se recouvrir, *i.e.* toute balise B ouverte après une balise A doit être fermée avant cette balise.

4. Les caractères de marquage (<, > et &) ne doivent pas être utilisés dans le contenu des balises mais être remplacés par l'entité correspondante.

Contre-exemples :

<condition> x **>** y </condition>

À remplacer par :

<condition> x **>** y </condition>

Remarque :

Un fichier XML qui respecte les règles du langage est dit *bien formé*. Ceci signifie qu'il pourra être lu par tout analyseur XML (*XML parser*) conforme aux standards.

III. DTD (Document Type Définition)

Une DTD va décrire quelle doit être la racine d'un fichier XML et pour chaque balise ou élément quels sont les attributs autorisés et/ou indispensables, quels sont les fils autorisés et/ou indispensables avec leur ordre, ainsi que les types de données qu'ils peuvent éventuellement contenir. Avec une DTD :

– Chaque document peut inclure une description de sa structure.

– Plusieurs utilisateurs peuvent se mettre d'accord sur un format de fichier leur permettant d'échanger des données.

– Une application ou un utilisateur peut vérifier que des données sont valides.

1. Déclaration de la DTD

Il existe deux possibilités pour définir une DTD. Soit on met notre DTD au sein d'un fichier de données (locale), soit on peut la définir dans un fichier externe (importée).

a) Définition d'une DTD embarquée dans un fichier de données XML

Si on embarque notre DTD dans un fichier de données, elle sera alors localisée dans le prologue, au niveau du tag *<!DOCTYPE ...>*. L'extrait de code suivant nous montre

qu'elle est la syntaxe à utiliser. A noter que dans ce cas, le nom du tag racine du fichier XML doit obligatoirement être <exemple>.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE exemple [
        <!-- début de la DTD -->
        . . .
        <!-- fin de la DTD -->
]>
<exemple>
        <!-- Suite du document XML -->
</exemple>
```

b) Définition d'une DTD dans un fichier externe

La seconde possibilité permet de définir un fichier de validation externe que l'on pourra par la suite associé à plusieurs fichiers de données. Pour lier un fichier de données à une DTD, il nous faut aussi utiliser le tag <!DOCTYPE ... >. L'exemple qui suit nous montre un exemple de liaison. Le mot clé SYSTEM est utilisé pour indiquer ou se trouve la DTD : c'est une URL qui est attendue. Nous pourrons donc lier nos fichiers à une quelconque DTD présente sur le Web.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE exemple SYSTEM "fichier.dtd" []>
        <exemple>
          <!-- Suite du document XML -->
        </exemple>
```

2. Définition des règles d'utilisation des tags

Une DTD définie plusieurs types de choses et notamment des tags et leurs règles d'imbrication, des listes d'attributs et des entités. On va travailler sur l'exemple suivant :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<!DOCTYPE TABLEAU SYSTEM "table.dtd">
<TABLEAU>
    <TITRE>Titre du tableau</TITRE>
    <LIGNE>
        <CELL-E>\</CELL-E>
        <CELL-E>Statistique 1</CELL-E>
        <CELL-E>Statistique 2</CELL-E>
        <CELL-E>Statistique 3</CELL-E>
    </LIGNE>
    <LIGNE>
        <CELL-E>Expérience 1</CELL-E>
        <CELL-D>25</CELL-D>
        <CELL-D>34</CELL-D>
        <CELL-D>75</CELL-D>
    </LIGNE>
    <LIGNE>
        <CELL-E>Expérience 2</CELL-E>
        <CELL-D>80</CELL-D>
        <CELL-D>56</CELL-D>
        <CELL-D>61</CELL-D>
    </LIGNE>
</TABLEAU>
```

a) Définition de tags contenant des données

On peut noter principalement deux types de tags : ceux qui contiennent des **données textuelles** et ceux qui contiennent des **sous tags**. Le premier cas est clairement le plus simple à définir. Les règles suivantes définissent les tags de titres, de cellules d'entête et de cellules de données : en effet, chacun de ces tags ne contient que des données.

```
<!ELEMENT TITRE (#PCDATA)>
<!ELEMENT CELL-E (#PCDATA)>
<!ELEMENT CELL-D (#PCDATA)>
```

Le mot clé *ELEMENT* permet d'indiquer que l'on cherche à définir un tag. Le mot clé

#PCDATA indique, quant à lui, que ce tag contiendra des données : *#PCDATA* signifiant Parsed Character DATA.

b) Définition de tags contenant des sous-tags

Plusieurs caractères vous sont proposés afin de pouvoir jouer sur l'occurrence d'un tag ou bien pour exprimer un choix.

`?` **Occurrence** : peut apparaître 0 ou 1 fois.

`+` **Occurrence** : peut apparaître 1 ou plusieurs fois.

`*` **Occurrence** : peut apparaître 0 ou plusieurs fois.

`,` **Séquence** : Les deux parties doivent obligatoirement apparaître et dans cet ordre.

`|` **Choix** : permet de choisir entre deux alternatives.

`ANY` **Choix** : le tag peut contenir n'importe quoi.

`EMPTY` Le tag ne peut strictement rien contenir.

Exemple :

```
<!ELEMENT agenda (personne*) >
<!ELEMENT personne (nom, prenom?, date)>
<!ELEMENT hr (EMPTY) >
```

Selon notre exemple précédent :

```
<!ELEMENT LIGNE (CELL-E | CELL-D)* >
```

=> Donc, notre grammaire de définition est :

```
<!ELEMENT TABLEAU (LIGNE*, TITRE?, LIGNE*) >
<!ELEMENT LIGNE (CELL-E | CELL-D)* >
<!ELEMENT TITRE (#PCDATA)>
<!ELEMENT CELL-E (#PCDATA)>
<!ELEMENT CELL-D (#PCDATA)>
```

22

Exemple :

<!ELEMENT agenda (personne*) >

<!ELEMENT personne (nom, prenom?, date)>

<!ELEMENT hr (EMPTY) >

Selon notre exemple précédent :

<!ELEMENT LIGNE (CELL-E | CELL-D)* >

=> Donc, notre grammaire de définition est :

<!ELEMENT TABLEAU (LIGNE*, TITRE?, LIGNE*) >

<!ELEMENT LIGNE (CELL-E | CELL-D)* >

<!ELEMENT TITRE (#PCDATA)>

<!ELEMENT CELL-E (#PCDATA)>

<!ELEMENT CELL-D (#PCDATA)>

c) Définition de listes d'attributs

Un tag ne se limite pas uniquement à contenir des données ou des sous-tags. Il peut aussi contenir des attributs. Si on cherche à valider un fichier de données contenant des attributs de tags, il faudra impérativement qu'ils soient définis dans la grammaire (et donc dans notre DTD).

Pour introduire une liste d'attributs, il faut utiliser le mot clé ***ATTLIST***. Il est suivi du nom du tag et des données descriptives de chaque attribut. Il est a préciser que la syntaxe utilisée n’est pas réellement triviale : soyez donc consciencieux et séparez la définition de chaque attribut pas un retour à la ligne.

Exemple :

```
<!ELEMENT TABLEAU (LIGNE*, TITRE?, LIGNE*) >
<!ATTLIST TABLEAU
        largeur CDATA "80%"
        bordure CDATA "1px"
>
<!ELEMENT LIGNE (CELL-E | CELL-D)* >
<!ELEMENT TITRE (#PCDATA)>
<!ATTLIST TITRE
```

alignement (top | bottom) "bottom"

>

<!ELEMENT CELL-E (#PCDATA)>

<!ELEMENT CELL-D (#PCDATA)>

IV. XML Schema

XML Schema est un **langage de description de format de document XML** permettant de définir la structure d'un document XML. Un schéma XML est lui-même un fichier XML. La connaissance de la structure d'un document XML permet notamment de vérifier la validité de ce document. Un fichier de description de structure (*XML Schema Description* ou fichier *XSD*) est donc lui-même un document XML.

Une instance d'un *XML Schema* est un peu l'équivalent d'une ***DTD***. XML Schema amène cependant plusieurs différences avec les DTD : il permet par exemple de définir des domaines de validité pour la valeur d'un champ, alors que cela n'est pas possible dans une DTD

Un exemple de fichier XSD :

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
        xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="personne">
    <xs:complexType>
      <xs:sequence>
       <xs:element name="nom" type="xs:string"/>
       <xs:element name="prenom" type="xs:string"/>
       <xs:element name="date_naissance"
          type="xs:date"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
</xs:schema>
```

Exemple d'un fichier xml valide :

```
<?xml version="1.0" encoding="UTF-8"?>
  <personne
  xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance"
  xsi:noNamespaceSchemaLocation="personne.xsd">
       <nom>De Latour</nom>
       <prenom>Jean</prenom>
       <date_naissance>1967-08-13</date_naissance>
  </personne>
```

V. Liens et chemins (XLink, XPath, XPointer)

1. XLink (XML Linking Language)

- Etendre les liens hypertextes de HTML.
- Associer de la sémantique aux liens.
- Tout élément XML peut « être » un lien hypertexte.

Exemple :

```
<a xmlns:xlink='http://wwww.w3.org/namespace/xlink/1999/'
     xlink:type='simple'
     xlink:href='students.xml'
     xlink:title='Student List'
     xlink:show='new'
     xlink:actuate='onRequest'>
Current List of Students
</a>
```

a) Attributs de comportement

- Affichage du document cible

```
xlink:show
```

```
(new | -- nouvelle fenêtre ou frame
replace | -- charger la ressource dans la même
fenêtre
embed | -- prend la place du lien dans le document
source
other | -- d'autres attributs peuvent donner des
indications
none) -- pas d'indications
```

- Parcours du lien

```
xlink:actuate

(onLoad | -- au chargement du document
onRequest | -- à la demande de l'utilisateur (click,
etc.)
other | -- indications possibles dans d'autres
attributs
none) -- pas d'indication
```

2. XPath

- Description de chemin pour :
  - Accéder, désigner une portion de document.
  - Filtrer des éléments sur la base d'un *pattern* XPath (Modèle XPath).
  - Exprimer des portions de requête.
- Exploite la structure du document dans la navigation.
- Langage de chemins.
- Axes de parcours.

Exemple1:

XPath: /talk/slide/title <title>XPath Exemple</title>

Document:

```
<talk>
 <slide>
  <title>XPath Exemple</title>
  ...
 </slide>
</talk>
```

Exemple2:

XPath: /talk/slide[2]/title <title>XLink</title>

Document:

```
<talk>
 <slide>
  <title>XPath Exemple</title>
   ...
 </slide>
 <slide>
  <title>XLink</title>
  ...
 </slide>
</talk>
```

3. XPointer

- Extension de XPath.
- Peut être utilisé dans un URI pour désigner un élément de document (sans y insérer d'ancre).
- Pour désigner des portions de documents XML :

  - qui peuvent commencer (ou finir) au milieu d'un élément XML.
  - Avec un début et une fin.

VI. Processeurs (XSLT, DOM)

1. XSL (eXtensible Stylesheet Language)

- Feuilles de style.
- Extensible Stylesheet Language, inspiré de DSSSL (formatage de SGML).

Spécifier la présentation d'un document XML.

2. XSLT (XSL Transformations)

- Langage de transformation/formatage.
- Transforme un document XML en un autre document XML.
- Peut engendrer du HTML.
- Langage à base de règles.
- Une règle s'applique à un type d'élément XML.
- Extraire des informations.
- Restructurer :
  - Engendrer une table des matières.
  - Engendrer un tableau à partir de listes d'éléments.
- Combiner des informations de provenances différentes.

Exemple de transformation :

```
<chapter>
  <title>XML</title> --> <h1>XML</h1>
<section>
  <title>XSLT</title> --> <h2>XSLT</h2>
```

Règles de transformation:

```
<xsl:template match='chapter'>
 <h1><xsl:value-of select='title'/></h1>
</xsl:template>
<xsl:template match='section'>
 <h2><xsl:value-of select='title'/></h2>
</xsl:template>
```

3. DOM (Document Object Model)

Une interface de programmation d'application (Application Programming Interface,en abrégé API) est un moyen standardisé d'accéder à certaines fonctionnalités, indépendant du langage ou de la fa¸con dont les structures de données sont implémentées. Les analyseurs XML sont définis par le W3C (World Wide Web Consortium) sous forme d'API.

Le programmeur doit donc éviter d'accéder directement aux structures de données (que ce soit les fichiers XML ou les arbres XML) et n'utiliser que des *méthodes* standardisées de manipulation des *objets* relatifs à la technologie XML. DOM est un analyseur syntaxique XML de type arborescent. Le rôle de cette API sera donc de permettre au programmeur de lire un fichier XML, de construire en mémoire l'arbre syntaxique de ce fichier, d'accéder de manière standardisée à la structure de l'arbre et à son contenu (indépendamment de son implémentation), de manipuler cet arbre et éventuellement de réécrire l'arbre modifié dans un fichier XML.

Le type d'objet standard de cette API est le noeud, sous forme de la classe ***Node***. Celle-ci dispose d'un grand nombre de méthodes permettant d'avoir accès à la structure de l'arbre syntaxique et éventuellement de la modifier.

WSDL
Web Services Description Language

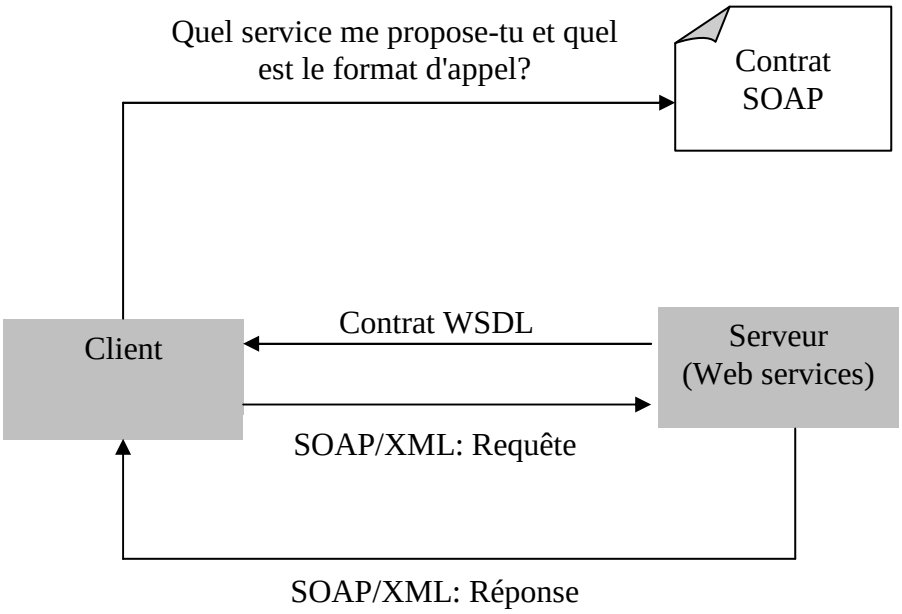
I. Introduction

Il faut savoir avant toute chose que chaque Web service possède un fichier de spécification: c'est ce qu'on appelle le WSDL (Web Services Description Language). C'est à partir de ce fichier que le développeur pourra savoir comment interroger le Web service, quelles sont ses différentes WebMethods, quels types ces WebMethods retournent-elles,…

Le WSDL décrit le fonctionnement d'un web Service et il est donc indispensable pour pouvoir utiliser ce dernier. Ce fichier est structuré en XML et il est assez complexe quand le développeur ne connaît pas XML.

Le WSDL décrit les aspects techniques d'implantation d'un service web (quel est le protocole utilisé, quelle est l'adresse du service). En fait, cette description sert à se connecter concrètement à un service web.

Enfin, pour que notre application puisse utiliser un Web Service, il faut qu'elle dispose d'une classe Proxy appelée Proxy Web qui sera créée à partir du contrat du Web Service, c'est-à-dire le fichier de description WSDL. Le Proxy va aider l'utilisateur à savoir où trouver le Web Service. Il contiendra également les détails des communications avec le Web Service.



II. Balises WSDL

Une description WSDL est un document XML qui commence par la balise **definitions** et contient les balises suivantes :

- **types**: cette balise décrit les types utilisés par le web service.
- **message**: cette balise décrit la structure d'un message échangé
- **portType**: cette balise décrit un ensemble d'opérations (interface d'un service web)
- **operation**: cette balise décrit une opération réalisée par le service web. Une opération reçoit des messages et envois des messages.
- **binding**: cette balise décrit le lien entre un protocole (http) et un portType.
- **service**: cette balise décrit un service comme un ensemble de ports.
- **port**: cette balise décrit un port au travers duquel il est possible d'accéder à un ensemble d'opérations. Un port référence un Binding.

1. Types

- Description en utilisant XML Schema.

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

2. Message

- Les messages sont envoyés entre deux interlocuteurs (ex: une opération reçoit des messages et envoie des messages.
- Un message est composé de plusieurs **part**
- Il existe deux façons pour définir des part:
  - Soit une part est un élément de type simple
  - Soit une part est un élément XML dont le type est défini dans un XML Schema

- Part de type simple

```
<wsdl:message name="personneMsg">
  <wsdl:part name="nom" type="xsd:string" />
  <wsdl:part name="prenom" type="xsd:string" />
</wsdl:message>
```

- Part qui utilise un XML Schema

```
<wsdl:message name="personneMsg">
  <wsdl:part name="personne" element="exemple:personne" />
</wsdl:message>
```

Définit dans un XML schema

3. portType

- Un portType permet d'identifier (nommer) de manière abstraite un ensemble d'opérations.

```
<wsdl:portType name="descriptionPersonnes" >
  <wsdl:operation name="getPersonne" >
    …
  </wsdl:operation>
  <wsdl:operation name="setPersonne" >
    …
  </wsdl:operation>
</wsdl:portType>
```

4. Opération

- WSDL définit 4 types d'opération :
  - One-Way : lorsque les opérations reçoivent des messages mais n'en n'envoient pas. (**input** sans **output**)
  - Request-response : lorsque les opérations reçoivent des messages puis renvoient des messages. (**input** suivi de **output**)
  - Solicit-response : lorsque les opérations envoient des messages puis en reçoivent. (**output** suivi de **input**)
  - Notification : lorsque les opérations envoient des messages mais n'en reçoivent pas. (**output** sans **input**)
- Quelque soit le type d'opération la définition est sensiblement la même :
  - Une opération :
    - Reçoit des messages : <wsdl:input …>
    - Envoie des messages : <wsdl:output …> ou<wsdl:fault …>
- La présence et l'ordre des input/outputs/fault dépendent du type de l'opération.

```
<wsdl:operation name="operation_name">
   <wsdl:input name="nom_optionel" message="nom_message" />
</wsdl:operation>
<wsdl:operation name="operation_name">
    <wsdl:output name="nom_optionel" message="nom_message" />
    <wsdl:input name="nom_optionel" message="nom_message" />
    <wsdl:fault name="nom_optionel" message="nom_message" />*
</wsdl:operation>
<wsdl:operation name="operation_name">
    <wsdl:input name="nom_optionel" message="nom_message" />
    <wsdl:output name="nom_optionel" message="nom_message" />
    <wsdl:fault name="nom_optionel" message="nom_message" />*
</wsdl:operation>
```

5. Binding

- WSDL permet de lier une description abstraite (portType) à un protocole.
- Chacune des opérations d'un portType pourra être liée de manière différente.
- Le protocole SOAP est un des protocoles qui peut être utilisé.
- D'autres binding sont standardisés par WSDL : HTTP et MIME.

- Un Binding :
  - peut être identifié par un nom : **name**
  - identifie le portType : **type**

<wsdl:binding **name**="*binding_name*"
type="*nom du portType*" >
…
</wsdl:binding>

6. Binding SOAP

- Pour préciser que le binding est de type SOAP, il faut inclure la balise suivante :

<soap:binding transport="*uri*" style="*soap_style*" />

- Transport définit le type de transport
  - http://schemas.xmlsoap.org/soap/http

pour utiliser SOAP/HTTP

- Style définit la façon dont sont créer les messages SOAP de toutes les opérations
  - rpc : Encodage RPC défini par SOAP RPC
  - document : Encodage sous forme d'élément XML
- Pour chaque opération du portType :
  - il faut préciser l'URI de l'opération : soapAction.
  - Il est aussi possible de repréciser la façon dont sont créés les messages SOAP : style.
- Pour chaque message de chaque opération, il faut définir comment sera créé le message SOAP.

Exemple:

```
<wsdl:binding type="descriptionPersonnes" >
    <soap:binding
        transport="http://schemas.xmlsoap.org/soap/http"
        style="rpc" />
    <wsdl:operation name="getPersonne">
    <soap:operation soapAction="http://www.exemple.fr/getPersonne" />
      <wsdl:input>
            <soap:body use="encoded"
             encodingStyle="schemas.xmlsoap.org/soap/encoding"/>
```

```
        </wsdl:input>
        <wsdl:output>
              <soap:body use="encoded"
              encodingStyle="schemas.xmlsoap.org/soap/encoding"/>
         </wsdl:output>
     </wsdl:operation>
   </wsdl:binding>
```

7. Service

- Un service est un ensemble de ports
- Un port a un portType
- Dans le cadre de SOAP, un port à une adresse (qui correspond à l'adresse http)

```
<wsdl:service name="MonService">
   <wsdl:port binding="intf:MonServiceSoapBinding">
      <soap:address
    location="http://mda.lip6.fr:8080/axis/services/MonService"/>
    </wsdl:port>
</wsdl:service>
```

Exemple d'un Web Service

```
<?xml version="1.0"?>
 <definitions name="StockQuote"
    targetNamespace="http://example.com/stockquote.wsdl"
    xmlns:tns="http://example.com/stockquote.wsdl"
    xmlns:xsd1="http://example.com/stockquote.xsd"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns="http://schemas.xmlsoap.org/wsdl/">
    <message name="SubscribeToQuotes">
      <part name="body" element="xsd1:SubscribeToQuotes"/>
      <part name="subscribeheader"
            element="xsd1:SubscriptionHeader"/>
    </message>
    <portType name="StockQuotePortType">
      <operation name="SubscribeToQuotes">
         <input message="tns:SubscribeToQuotes"/>
       </operation>
    </portType>
    <binding name="StockQuoteSoap"
              type="tns:StockQuotePortType">
             <soap:binding style="document"
```

```
            transport="http://example.com/smtp"/>
          <operation name="SubscribeToQuotes">
            <input message="tns:SubscribeToQuotes">
                  <soap:body parts="body" use="literal"/>
                  <soap:header message="tns:SubscribeToQuotes"
                        part="subscribeheader" use="literal"/>
            </input>
          </operation>
    </binding>
    <service name="StockQuoteService">
      <port name="StockQuotePort" binding="tns:StockQuoteSoap">
        <soap:address location="mailto:subscribe@example.com"/>
      </port>
    </service>
</definitions>
```

UDDI
PUBLICATION, RECHERCHE ET SÉLECTION DE **SERVICES WEB**

I. Introduction

Une fois le service Web décrit par son fournisseur, pour qu'il puisse être utilisé (et réutilisé) par le plus grand nombre de clients, il doit être publié dans un registre public. Lors de la conception d'une application, ses concepteurs (futurs clients de services) doivent rechercher et sélectionner à partir de registres existants les services Web de leur choix.

II. UDDI : vers des spécifications standard de publication et recherche de services Web

UDDI a été conçu en 2000 à l'initiative d'un ensemble d'industriels (Ariba, IBM, Microsoft), en vue de devenir le registre standard de la technologie des services Web. Pour convenir à la technologie des services Web, les services référencés dans UDDI sont accessibles par l'intermédiaire du protocole de communication SOAP, et la publication des informations concernant les fournisseurs et les services doit être spécifiée en XML afin que la recherche et l'utilisation soient faites de manière dynamique et automatique. UDDI constitue un méta-service possédant des fonctions de publication et de recherche. Nous étudions tout d'abord comment les spécifications UDDI rendent possible la publication des services Web. Ensuite, les moyens mis en œuvre pour rechercher et sélectionner des services Web sont décrits.

1. Publication de services Web avec UDDI

Les différents composants de la publication faite par UDDI sont des documents XML manipulant de l'information à propos du fournisseur (Business Entity), le service lui-même (Business Service), les accès au service (Binding Template), le type de service (tModel) et des relations entre deux parties (Publisher Assertion).

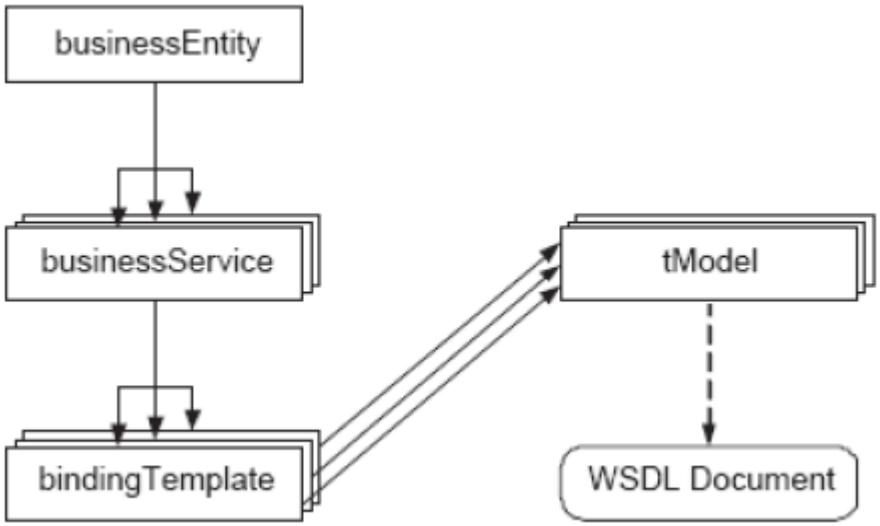


Figure 1. Entités composant un annuaire UDDI

Le fournisseur (Business Entity). Les informations concernant l'organisation hébergeant le service, le fournisseur et celles nécessaires à l'identification de l'entreprise sont répertoriées dans ce document XML. Ce document contient de l'information descriptive sur l'entreprise ou le fournisseur et sur les services proposés (lien vers l'entité Business Service – cf. Figure 1).

Le service (Business Service). Ce composant représente les services proposés par l'organisation. La description des services contenue dans l'entité Business Service est de haut niveau (aucune information technique n'est décrite ici). Les informations à propos du nom du service et de son objectif sont représentées dans ce composant. Le fournisseur peut rassembler dans cette entité un ensemble de services répondant aux mêmes objectifs dans une même catégorie. Par exemple, une catégorie tourisme peut contenir un service météorologique et un service localisant les sites touristiques. Les catégories de service (contenant un ou plusieurs services) sont liées (selon le nombre de services) à un ou plusieurs points d'accès (Binding Template – cf. Figure 1).

Les accès au service (Binding Template). Ce module décrit les points d'accès aux services Web (URL) et le moyen d'y accéder (les différents protocoles à utiliser) afin d'invoquer les services.

Le type de service (tModel). Le tModel permet d'associer un service à sa description WSDL. Le client potentiel peut ainsi avoir connaissance des conventions d'utilisation du service. La liaison entre les entités Binding Template et tModel est nécessaire pour l'invocation du service (cf. Figure 1).

2. Recherche de services Web avec UDDI

La recherche et la sélection dans UDDI reposent sur la publication préalablement décrite du service et de son fournisseur (*cf.* sous-section précédente). Le futur client peut connaître par l'intermédiaire d'UDDI : les fournisseurs d'un service, les services proposés par un fournisseur donné, les moyens d'invoquer un service. Pour apporter aux clients la réponse à ces questions, UDDI organise l'ensemble des informations qu'il contient en trois parties, spécifiées en XML. Chacune d'elles peut être utilisée pour faire une recherche via UDDI. Ces parties sont les suivantes :

Les pages blanches (*White Paper*). Ce composant permet de connaître les informations à propos de l'organisation proposant le service. Cette description contient toutes les informations jugées pertinentes pour identifier l'organisation (telles que son nom, son adresse physique). Le futur client du service retrouve dans les pages blanches les informations que le fournisseur a renseignées dans l'élément *Business Entity* lors de la publication.

Les pages jaunes (*Yellow Paper*). Les pages jaunes d'UDDI détaillent la description de l'organisation faite dans les pages blanches en répertoriant les services proposés. Dans cette section, sont décrits : la catégorie de l'entreprise, le secteur d'activité dans lequel exerce l'entreprise, les services offerts par cette organisation, le type de services et les conventions d'utilisation – prix, qualité de service, etc. Cette description repose sur la classification standard de l'industrie nord américaine (NAICS et UNSPSC2). La description des services contenue dans les pages jaunes est non technique et est renseignée par les fournisseurs eux mêmes.

Les pages vertes (*Green Paper*). Les pages vertes comportent les informations techniques liées aux services Web et basées sur leur description WSDL.

À l'origine, il existait des registres UDDI dits publics (tels que ceux de Microsoft ou IBM) pour lesquels n'importe qui pouvait devenir, soit fournisseur, soit client de services Web. L'universalité de ces registres devait amener UDDI à devenir le standard de publication des services Web. En 2006, le nombre de services Web publiés a atteint le nombre de 50000. Malgré ce nombre, UDDI n'a jamais atteint son

but : devenir le registre standard des services Web. Par conséquent, la maintenance des registres publics UDDI (tels que ceux de Microsoft et d'IBM) a été suspendue. Le réel succès d'UDDI se situe au niveau des registres privés. En effet, de nombreuses organisations utilisent les spécifications de UDDI afin d'implémenter leur propre registre de services Web. Cependant, les spécifications UDDI souffrent de certaines limitations : il n'existe pas de plate-forme d'édition ; les API de UDDI sont insuffisantes pour développer efficacement des méthodes de publication et de recherche ; le modèle de recherche de UDDI est pauvre (recherche portant sur l'identifiant, le nom du service, ou sur les éléments du document WSDL).

III. Registres de services Web classiques accessibles sur le Web

Les registres publics UDDI n'étant plus maintenus, différentes solutions de registres publics ont été créées et sont disponibles sur le Web. Ces registres ont vu le jour afin de proposer une interface entre les fournisseurs et les clients de services Web. L'analyse que nous menons sur les registres existants s'intéresse à trois aspects : la représentation des services Web proposée par les registres, les méthodes de recherche, et les critères de sélection proposés aux clients. Parmi le panel de registres accessibles via le Web (tels que *Web Service List*), nous avons choisi d'en étudier deux (*XMethods* et *RemoteMethods*) issus des propositions d'industriels, et un travail académique (*QWS Dataset*). Notre choix est orienté vers trois champs d'observation (la représentation de services, les méthodes de recherche, et les critères de sélection).

1. XMethods

Le site *XMethods*31, proposé par l'organisation du même nom, est un registre de services Web hébergé sur Internet. Il contient environ 500 Web services34. Ce site permet de publier des services Web et de faire la recherche de services préalablement publiés.

La publication des services sur le site *XMethods* repose sur une publication UDDI. Le fournisseur doit tout d'abord être inscrit auprès du site *XMethods*. Ensuite, le fournisseur doit décrire un document *tModel* et un document *Business Service* selon les spécifications UDDI et une extension spécifique au site *XMethods*.

Le document tmodel. Ce document contient une liste de services Web que le fournisseur souhaite publier sur *XMethods*. L'ensemble des services est trié par le biais de leur description WSDL et de la localisation du document WSDL (identifiée par un URL). *XMethods* propose aux fournisseurs de décrire en langage naturel le service fourni (par le biais d'une description courte – une phrase, et d'une description détaillée – environ un paragraphe) et des notes d'utilisation à destination des futurs clients. Ces descriptions textuelles sont à ajouter dans le document *tModel* par le biais des éléments <short description/>, <detailed description/> et <usage notes/>, qui étendent les spécifications UDDI.

Le document Business Service. En plus des informations prises en compte dans la spécification UDDI, *XMethods* propose d'ajouter dans le document *Business Service* des informations concernant l'e-mail du fournisseur (élément <contact email/>), l'outil de développement (tel que .NET, AXIS) avec lequel le fournisseur a implémenté le service (élément <implementation/>). Les ports d'accès au service (décrits par le document *Binding Template* associé au *Business Service* et au *tModel*) doivent être renseignés par le fournisseur.

Pour rechercher un service publié sur *XMethods*, le client a deux listes à sa disposition. La première contient la liste des services Web récemment enregistrés et la seconde, dite complète, recense l'ensemble des services Web disponibles sur ce site Aucune méthode de recherche n'est disponible, les clients de services Web doivent chercher un service convenant à leurs besoins en explorant l'ensemble des services publiés. Les services Web apparaissant sur les listes sont ordonnés selon la date de mise en ligne, ce qui ne facilite pas le processus de recherche. Les critères discriminants les services sont le fournisseur (*Publisher*), le nom du service (*Service Name*), la description du service (*Description*) et l'outil avec lequel le fournisseur a implémenté le service (*Implementation*).

Toutefois, afin d'aider la sélection des services Web contenus dans la liste, le client peut disposer d'informations supplémentaires sur le service telles que la localisation de la description WSDL, l'adresse de son site Web, et une description détaillée des tâches proposées par le service. Ces informations sont celles renseignées par le fournisseur lors du processus de publication. Elles sont visibles sur l'interface du site une fois que le client sélectionne le lien hypertexte du service.

Publisher	Style		Service Name	Description	Implementation
StrikeIron	DOC	Try It	Wall Street Horizon Real-Time Company Earnings	Information to analyze and evaluate investments and future earning potential	
SOATrader	RPC	Try It	Captcha Web Service	This Web service will create captcha requests.	
StrikeIron	DOC	Try It	StrikeIron Reverse Phone Business Intel	Get detailed information about a company based on a phone number	
StrikeIron	DOC	Try It	StrikeIron Reverse Phone Residential Intel	Get detailed information about a residence based on a phone number	
melmasry	DOC	Try It	Jobs in British Columbia and Alberta	Jobs in Western Canada	MS .NET
TQFTD2007	DOC	Try It	Mighty Maxims	Hand-picked selections from the The Quote For Today!	MS .NET
TQFTD2007	DOC	Try It	Mighty Meals	A refreshing Cookpedia.com recipe every day!	MS .NET
kyngas	DOC	Try It	Yellow Pages Lookup	search for companies by name	
StrikeIron	DOC	Try It	MapQuest Basic Mapping	Add mapping to your Web site or application	
cocoma		Try It	Cocoma Google Search Web Services	Check your search engine placement and page ranking.	
cocoma		Try It	Cocoma Video Web Services	Get video list from Google, MSN, Yahoo, AOL.	

Figure 2.Extrait de la liste des services Web disponibles sur *XMethods*.

Ce registre de services Web permet aux fournisseurs de rendre visibles leurs services Web via la publication sur un site Web. Cependant, du point de vue d'un client de service, *XMethods* ne propose aucune fonction de recherche et les catégories d'information décrivant le service reste basiques.

2. RemoteMethods

Le site *RemoteMethods* est un registre de services Web accessible sur le Web créé en 2002. Il propose environ 400 services Web. Ce site propose aux fournisseurs de publier leurs services et facilite les étapes de recherche et de sélection de services Web pour les clients. *RemoteMethods* propose deux moyens de publier un service sur le site : le formulaire de publication ou la recommandation de service.

Le formulaire de publication. Le formulaire propose aux propriétaires de services de les rendre visibles sur *RemoteMethods*. La publication à l'aide du formulaire s'effectue en quatre étapes :

- L'inscription ou la validation du **profil de l'organisation** qui fournit le service.

- La publication de la **description détaillée du service** qui comporte des informations

telles que le nom du service, le nom de la compagnie qui a développé le service, l'URL du site du fournisseur, la description WSDL, une description textuelle du service.

- La publication de la **description détaillée de l'évaluation du service** qui comporte des informations telles que le prix du service, la date de mise à jour du service, la durée de la période d'évaluation, si elle existe.
- Une fois le processus de description complété, la représentation du service est soumise à l'évaluation des administrateurs de RemoteMethods pour approbation de la publication.

La recommandation de services. Cette forme de publication permet à des clients de services non propriétaires de ces derniers de les recommander au site *RemoteMethods*. Cette recommandation repose sur un simple formulaire dans lequel les informations suivantes sont à renseigner :

- le nom du service à recommander,
- la page d'accueil du site du fournisseur,
- l'URL de la localisation de la description WSDL,
- la catégorie dans laquelle se situe le service (optionnelle),
- un commentaire textuel,
- l'adresse email de la personne qui recommande le service.

À la suite de la recommandation, les administrateurs de *RemoteMethods* décident de publier ou non le service recommandé et se chargent de décrire le service et son évaluation.

La recherche de services Web sur *RemoteMethods* repose sur un ensemble de huit catégories et sous-catégories de services Web pré-déterminées (*cf.* Figure 3). Le client peut alors naviguer par le biais de liens hypertexte entre les catégories et sous-catégories proposées par *RemoteMethods*.

Lorsqu'il n'existe plus de sous-catégories, les services Web relatifs à la (aux) catégorie(s) choisie(s) sont listés.

Figure 3. Catégories de services pour la recherche de services sur le site *RemoteMethods*.

Une fois que le client a trouvé la catégorie correspondant à ses besoins, il doit sélectionner le service qui lui convient le mieux. En vue d'aider les clients dans cette tâche, le site *RemoteMethods* met à la disposition du client un ensemble d'informations à propos du fournisseur, du service Web (renseigné, soit par le fournisseur, soit par le site lui-même), et un outil de tri de services.

Les informations de description du fournisseur. La description du fournisseur est composée d'informations sur la sécurité garantie par le fournisseur lors de l'invocation du service (avec les méthodes de sécurité utilisées) et d'informations sur les moyens mis en oeuvre par le fournisseur pour garantir la maintenance de ses services Web (avec les horaires du support téléphonique, le temps de réponse à un mail, etc.).

Les informations de description du service renseignées par le fournisseur. Lors de la publication des services sur *RemoteMethods*, le fournisseur doit renseigner un ensemble d'informations qui permettent de faciliter le processus de sélection de son service. Ces informations sont le prix du service et une description textuelle de l'objectif du service. La date de mise à jour du service par le fournisseur est contenue dans la description du service.

Les informations de description du service renseignées par le site RemoteMethods. L'originalité de RemoteMethods est qu'il propose aux clients des services Web d'annoter les services Web qu'ils ont utilisés. Cette annotation est constituée d'une note globale (de une à cinq étoiles), d'un rapport d'erreurs (rapport textuel) et d'un compte-rendu. Le compte-rendu est composé des forces et faiblesses du service, d'un paragraphe de description libre, et de la durée d'utilisation du service sur laquelle se base l'évaluation du service.

Le tri de service. La liste de services Web peut être triée selon un ensemble de critères : par nom, par prix, par note, par date de mise à jour.

L'originalité de RemoteMethods repose sur l'aide à la navigation des clients dans l'ensemble des services Web disponibles et la possibilité, pour ces mêmes clients, de recommander des services Web. Bien que RemoteMethods offre une représentation riche des services publiés, la méthode de recherche proposée ne permet pas aux clients d'établir une requête portant sur l'ensemble des informations composant cette description.

Le Protocole SOAP

I. Introduction

SOAP est un protocole adopté par le Consortium W3C. Le Consortium W3C crée des standards pour le Web : son but est donc de créer des standards pour favoriser l'échange d'information. Un standard veut tout simplement dire qu'il peut être accessible à tout le monde, et donc qu'il n'est pas propriétaire. Ce qui a pour conséquence qu'un protocole standard contrairement à un protocole propriétaire pourra être utilisé sous n'importe quelle plateforme.

SOAP veut dire : Simple Object Access Protocol. En français, Protocole Simple d'Accès aux Objets. En effet, le protocole SOAP consiste à faire circuler du XML via du http sur le port 80. Cela facilite grandement les communications, car le XML est un langage standard et le port utilisé est le port 80, qui ne pose donc pas de problèmes pour les firewalls de l'entreprise, contrairement à d'autres protocoles.

Tout comme la technologie des Web Services, le protocole SOAP est très jeune. Le protocole SOAP a été crée en Septembre 98, avec la version 0.9, par trois grandes entreprises : Microsoft, UserLand et DevelopMentor. IBM n'a participé au protocole SOAP qu'à partir de la version 1.1 en Avril 2000.
Depuis Septembre 2000, SOAP 1.1 est en refonte complète pour donner jour à la version 1.2 avec un groupe de travail de plus de 40 entreprises§ Parmi ces 40 entreprises, on retrouve bien sûr Microsoft, IBM mais aussi HP, Sun, Intel, …

II. Les autres protocoles

Jusqu'à la création du protocole SOAP, trois grands protocole étaient utilisés :

1. COM et DCOM

Les protocoles COM (Component Object Model) et DCOM (Distributed Component Object Model) ont été écrits par Microsoft et permettaient de faciliter la communication entre les composants Windows. Il y a eu un portage de COM sous UNIX, mais ce protocole n'a été utilisé que par des plateformes Windows et pour l'Intranet.
Les protocoles COM et DCOM n'étaient utilisés la plupart du temps que pour l'Intranet, car le port d'écoute des communications était statique : c'est-à-dire qu'on

ne pouvait pas changer ce port et cela posait de gros problèmes de sécurité pour les entreprises qui voulaient utiliser ce protocole pour communiquer entre elle.

2. CORBA

CORBA (Common Object Request Broker Architecture) a été créé par l'OMG (Object Management Group) pour faciliter la communication sous n'importe quelle plateforme. Ceci a été réalisé via un langage neutre de définition d'interface appelé IDL (Interface Definition Language) et un protocole commun de transport des données.

Malheureusement, les spécifications de ce protocole sont très denses et l'architecture est donc au final très lourde à déployer.

3. RMI

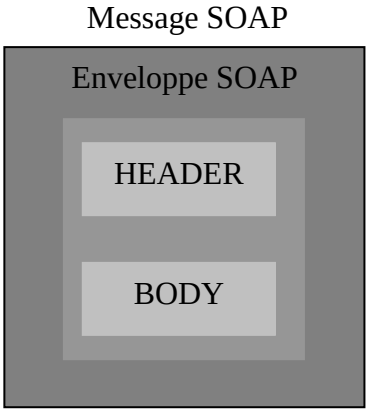
RMI (Remote Method Invocation) était un protocole très simple à utiliser et très efficace mais le problème est que ce protocole ne fonctionnait que sous environnement JAVA.

III. Messages SOAP

1. Structure d'un message SOAP

Un message SOAP est un document XML qui doit avoir la forme suivante :

- Une déclaration XML qui n'est pas obligatoire.
- Une enveloppe SOAP qui est composée de :
    - Un en-tête SOAP (HEADER).
    - Un corps SOAP (BODY).



2. Message SOAP dans l'exemple

Voyons à présent un exemple concret d'un message SOAP de requête et le message de réponse. Dans notre exemple, nous allons prendre une méthode qui retourne le nombre entré en paramètre. Ce type de méthode n'a bien sûr aucun intérêt dans la pratique, mais plus la méthode est simple, mieux nous comprendrons la structure d'un message SOAP.

Voici la signature de notre méthode:

```
Int GetNombre (int Nombre);
```

La structure du message de requête sera du type:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<SOAP-ENV: envelope
SOAP-ENV: encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV=http:// schemas.xmlsoap.org/soap/envelope/
xmlns:SOAP-ENC=http:// schemas.xmlsoap.org/soap/encoding/
xmlns:xsi=http://www.w3.org/1999/XMLSchema-instance
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV: Body
        <nsl:GetNombre
          xmlns:nsl="urn:MySoapServices">
          <paraml xsi:type="xsd:int">10</paraml>
        </nsl:GetNombre>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

La première ligne du message SOAP contient une déclaration XML qui n'est pas obligatoire. L'enveloppe SOAP est ensuite déclarée via la balise <SOAP-ENV:Envelope>. Cette enveloppe est composée d'un corps (<SOAP-ENV:BODY>).

Dans le corps de ce message, nous pouvons très bien voir notre méthode "GetNombre" et son paramètre qui est égale à 10 dans notre exemple.

Voyons à présent le message de réponse:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<SOAP-ENV: envelope
```

```
xmlns:SOAP-ENV=http:// schemas.xmlsoap.org/soap/envelope/
xmlns:xsi=http://www.w3.org/1999/XMLSchema-instance
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV: Body
        <nsl:GetNombreResponse
          xmlns:nsl="urn:MySoapServices">
          SOAP-ENV: encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
                    <return xsi:type="xsd:int">10</return> >
        </nsl:GetNombreResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Le message de réponse a la même structure que celle du message envoyé. Ceci veut dire qu'il contient une déclaration XML, ainsi qu'une enveloppe SOAP composée d'un corps.

Dans le corps du message de réponse, nous pouvons toujours voir notre méthode "GetNombre": le mot "Response" a été rajouté à la fin de la méthode pour bien préciser qu'il s'agit du retour d'une requête sur la méthode.

La valeur renvoyée par la méthode "GetNombre" est égale à 10.

Dans notre exemple, l'enveloppe SOAP n'était pas composée d'un en-tête: en effet, l'en-tête d'un message SOAP est optionnelle et est utilisée pour transmettre des données d'authentification ou de gestion de sessions.

IV. Sécurité et SOAP

La sécurité est un des grands défauts du protocole SOAP. En effet, lors de la création du protocole SOAP, des groupes ont voulu faire pression pour insérer dans le protocole SOAP des mécanismes de sécurité. Mais le projet a été abandonné, car le protocole SOAP devait rester facile à mettre en œuvre. De plus, comme nous avons pu le voir dans la section précédente, les messages SOAP peuvent être lus sans problèmes et il suffit donc qu'une personne lise les messages SOAP émis et reçus par le Web Service pour avoir l'intégralité des informations sans aucune difficulté.

Le cryptage devient donc nécessaire pour des services transmettant des données sensibles. Le protocole HTTPS, qui propose un chiffrement jusqu'à 128 bits, pourra donc être utilisé pour mettre en place un système de communication sécurisé.

V. Encodage

- Un message SOAP contient des données typées. Il faut donc définir un moyen d'encoder ces données.

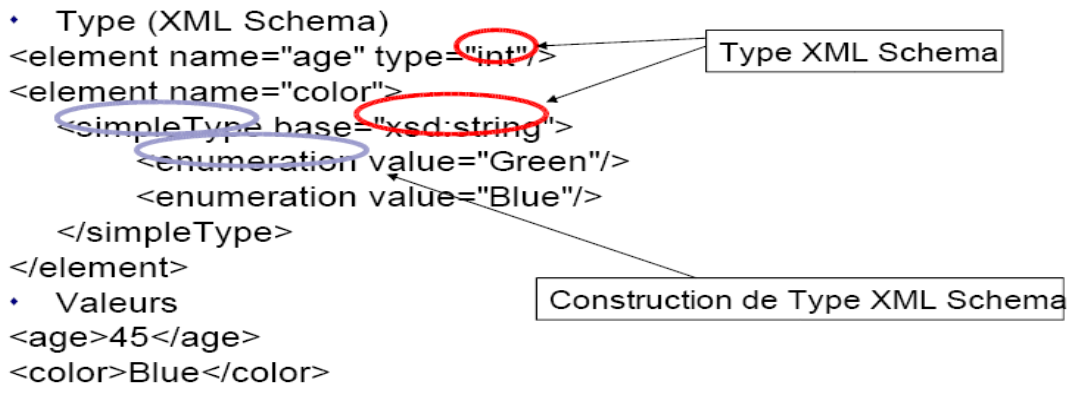
✓ Vocabulaire SOAP :
- Value (valeur d'une donnée)
  - Simple value (string, integers,etc)
  - Compound value (array, struct, …)
- Type (d'une value)
  - Simple Type
  - Compound Type

- L'encodage c'est la représentation de valeurs sous forme XML.
- Le décodage c'est la construction de valeurs à partir d'XML
- L'XML qui représente les valeurs a une structure qui dépend du type des valeurs
- Il faut donc définir le type:
  - Soit un mécanisme définit par l'utilisateur
  - Soit l'utilisation d'XML Schéma (préconisé)

1. Simple Type

- La définition d'un XML Schéma pour tout type peut être fastidieux
- SOAP a défini deux façons de préciser le type d'une valeur sans définir le XML Schéma:
  - <SOAP-ENC:int>45</SOAP-ENC:int>
  - <cost xsi:type="xsd:float">29.5</cost>

Exemple:



2. Compound Types

Une structure est un type composé dans lequel les membres sont accessibles uniquement grâce à des noms différents.

- Un tableau est un type composé dans lequel les membres sont accessibles uniquement grâce à leur position.

Exemple

- Type (XML Schéma)

```
<element name="Person">
<complexType>
<element name="name" type="xsd:string"/>
<element name="age" type="xsd:int"/>
</complexType>
<element>
```

- Valeur

```
<Person>
<name>Xavier</name>
<age>28</age>
</Person>
```

Les tableaux

Le type est directement précisé grâce aux balises SOAP:

Exemple:

```
<myFavoriteNumbers SOAPENC: arrayType="xsd:int[2] ">
<SOAP-ENC:int>3</SOAP-ENC:int>
```

<SOAP-ENC:int>4</SOAP-ENC:int>

</myFavoriteNumbers>

VI. SOAP avec HTTP

SOAP peut être facilement porté sur http, parce que:

- Il Convient au mode Request/Response de Http
- Le message SOAP est mis dans une requête POST avec un content-type text/xml
- On peut définir un header http : SOAPAction

Exemple 1:

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAPENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<m:GetLastTradePrice xmlns:m="Some-URI">
<symbol>DIS</symbol>
</m:GetLastTradePrice>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Exemple 2:

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
<SOAP-ENV:Envelope
xmlns:SOAPENV="
http://schemas.xmlsoap.org/soap/envelope/"/>
<SOAP-ENV:Body>
<SOAP-ENV:Fault>
<faultcode>SOAP-ENV:Server</faultcode>
<faultstring>Server Error</faultstring>
</SOAP-ENV:Fault></SOAP-ENV:Body></SOAP-ENV:Envelope>
```

Références

Alex Ferrata et Matthew MacDonald: Programming .NET Web Services, O'Reilly, September 2002. ISBN: 0-596-00250-5

David Chappell et Tyler Jewell: Java Web Services, O'Reilly, March 2002. ISBN: 0-596-00269-6

Ethan Cerami: Web Services Essentials – Distributed Applications with XML-RPC, SOAP, UDDI and WSDL, O'Reilly, February 2002. ISBN: 0-596-00224-6

Fabrice Rossi : Services Web–SOAP, Université Paris-IX Dauphine

George Reese: Database Programming with JDBC and Java, 2nd Edition 2000, O'Reilly, 2000. ISBN: 1-56592-616-1

James Snell, Doug Tidwell et Pavel Kulchenko: Programming Web Services with SOAP– Building Distributed Applications, O'Reilly, January 2002. ISBN: 0-596-00095-2

Mickaël Baron : SOA – Services Web Etendus, WSDL : Décrire et configurer, 2010 (Rév. Janvier 2011)

Simon St. Laurent, Joe Johnston, et al.: Programming Web Services with XML-RPC, O'Reilly, June 2001. ISBN: 0-596-00119-3

UDDI Project, UDDI Technical White Paper, September 2000.

(http://java.sun.com/webservices) -> JWSDP last version 65

(http://java.sun.com/webservices/archive.html) -> JWSDP version 1.1

Common Object Request Broker Architecture (CORBA), OMG (http://www.omg.org)

dotNET, Microsoft (http://www.microsoft.com/net) (http://www.dotnet-fr.org)

HyperTest Markup Language (HTML), W3C (http://www.w3.org/MarkUp)

Java RMI, Sun Microsystems (http://java.sun.com/products/jdk/rmi)

Java Web Service Developer Pack 1.1 (JWSDP), Sun Microsystems

JDBC, Sun Microsystems (http://java.sun.com/products/jdbc)

Oracle 8i, Oracle (http://www.oracle.com)

Servlet, Sun Microsystems (http://java.sun.com/products/servlet)

Simple Object Access Protocol (SOAP) 1.1, W3C Submission, May, 2001. (http://www.w3.org/TR/SOAP/)

SQL, (http://www.sql.org)

Web Services Description Language (WSDL) 1.1, W3C Note, March 2001. (http://www.w3.org/TR/wsdl)