**Les services web de type Soap**

Les services web de type Soap permettent l'appel d'une méthode d'un objet distant en utilisant un protocole web pour le transport (http en général) et XML pour formater les échanges. Les services web fonctionnent sur le principe du client serveur :

* un client appelle les services web
* le serveur traite la demande et renvoie le résultat au client
* le client utilise le résultat

Le service web permet d'utiliser des standards ouverts et reconnus notamment HTTP et XML. L'utilisation de ces standards permet d'écrire des services web dans plusieurs langages et de les utiliser sur des systèmes d'exploitation différents.

Les services web de type Soap utilisent des messages au format XML pour permettre l'appel de méthodes ou l'échange de messages.

Initialement, Sun a proposé un ensemble d'outils et d'API pour permettre le développement de services web avec Java. Cet ensemble se nomme JWSDP (Java Web Services Developer Pack) dont il existe plusieurs versions.

Depuis Sun a intégré la plupart de ces API permettant le développement de services web dans les spécifications de J2EE version 1.4.

La version 5 de Java EE et la version 6 de Java SE utilisent JAX-WS 2.0 pour faciliter le développement de services web en utilisant des annotations.

**La présentation des services web**

Les services web sont des composants distribués qui offrent des fonctionnalités aux applications au travers du réseau en utilisant des standards ouverts. Ils peuvent donc être utilisés par des applications écrites dans différents langages et exécutées dans différentes plateformes sur différents systèmes.

Les services Web utilisent une architecture distribuée composée de plusieurs ordinateurs et/ou systèmes différents qui communiquent sur le réseau. Ils mettent en œuvre un ensemble de normes et standards ouverts qui permettent aux développeurs d'implémenter des applications distribuées internes ou externes en utilisant des outils différents fournis par différents fournisseurs.

Un service web permet généralement de proposer une ou plusieurs fonctionnalités métiers qui seront invoquées par un ou plusieurs consommateurs.

Il existe deux grandes familles de services web :

* Les services web de type SOAP
* Les services web de type REST

**La définition d'un service web**

Il existe plusieurs définitions pour les services web mais la plus simple pourrait être "fonctionnalité utilisable au travers du réseau en mettant en œuvre un format standard, généralement utilisant XML".

Les services web ne sont donc qu'une nouvelle forme d'échanges de type RPC (Remote Procedure Call). Leur grand intérêt est de reposer sur des standards plutôt que sur des protocoles propriétaires. Par exemple, le transport repose généralement sur le protocol HTTP mais il est possible d'utiliser d'autres protocoles tels que JMS, FTP ou SMTP.

Les services web de type Soap font un usage intensif de XML, des namespaces XML et des schémas XML. Ces technologies font la force des services web pour permettre leur utilisation par des clients et des serveurs hétérogènes. XML est notamment utilisé pour stocker et organiser les informations de la requête et de la réponse mais aussi pour décrire le service web. L'utilisation de XML pour le format des messages rend les échanges indépendants du système d'exploitation, de la plate-forme et du langage.

Il est ainsi possible de développer des services web avec une plate-forme (par exemple Java) et d'utiliser ces services web avec une autre plate-forme (par exemple .Net ou PHP) : c'est une des grandes forces des services web même si cela reste parfois quelque peu théorique, essentiellement à cause des implémentations des moteurs utilisés pour mettre en œuvre les services web.

Un service web est donc une fonctionnalité accessible au travers du réseau grâce à des messages au format XML. Le format de ces messages est généralement SOAP bien que d'autres formats existent (REST, XML-RPC, ...).

L'appel à un service web de type SOAP suit plusieurs étapes :

1. Le client instancie une classe de type proxy encapsulant le service Web XML.
2. Le client invoque une méthode du proxy.
3. Le moteur SOAP sur le client créé le message à partir des paramètres utilisés pour invoquer la méthode
4. Le moteur SOAP envoie le message SOAP au serveur généralement en utilisant le protocol HTTP
5. Le moteur SOAP du serveur réceptionne et analyse le message SOAP
6. Le moteur fait appel à la méthode de l'objet correspondant à la requête SOAP
7. Le moteur SOAP sur le server créé le message réponse à partir de la valeur de retour
8. Le moteur SOAP envoie le message SOAP contenant la réponse au client généralement en utilisant le protocole http
9. Le moteur SOAP du client réceptionne et analyse le message SOAP
10. Le moteur SOAP du client instancie un objet à partir du message SOAP contenant la réponse

Un des intérêts des services web est de masquer aux développeurs la complexité de l'utilisation des standards sous-jacents. Ceci est réalisé grâce aux développements d'API et de moteurs pour la production et la consommation de services web.

Ces API sont dépendantes des plateformes utilisées (Java, .Net, PHP, Perl, ...) mais elles mettent toutes en œuvre avec plus ou moins de complétude les standards de l'industrie relatifs aux services web notamment SOAP et WSDL.

Ainsi les développeurs peuvent se concentrer sur l'écriture des traitements proposés par les services et par leur consommation sans se soucier de la tuyauterie sous-jacente. Un minimum de compréhension est cependant nécessaire pour bien appréhender les mécanismes mis en œuvre.

**Les différentes utilisations**

L'utilisation de services web peut avoir plusieurs intérêts :

* L'exposition de fonctionnalités au travers du réseau : les traitements des opérations des services web peuvent être invoqués via une requête HTTP, ce qui peut permettre à plusieurs applications de consommer ces services web
* La communication entre des applications et des systèmes hétérogènes : l'utilisation de standards ouverts permet la production et la consommation des services web par différentes technologies sur différents systèmes d'exploitation
* La mise en œuvre des protocoles standards de l'industrie au niveau des couches transport, messaging, description et recherche permet de choisir entre plusieurs implémentations proposées et ainsi de ne pas dépendre d'un seul fournisseur
* Les échanges se font en utilisant l'infrastructure existante puisque les services web sont généralement invoqués en utilisant le protocole HTTP. Ceci permet de facilement passer un firewall pour permettre une invocation depuis l'extérieur
* Les services web permettent un couplage faible entre les fonctionnalités exposées et les applications qui les utilisent à tel point que les consommateurs et les producteurs peuvent être écrits pour des plateformes ou des langages différents (Java, .Net, PHP, ...).
* Les services permettent de définir de nouvelles opportunités de business voire même de nouveaux modèles économiques en permettant de proposer des fonctionnalités à des partenaires par exemple

Les standards

L'intérêt des services web grandissant, des standards ont été développés pour assurer les besoins requis pour leur mise en œuvre.

L'architecture des services web est composée de quatre grandes couches mettant en oeuvre plusieurs technologies :

* Découverte : cette couche représente un annuaire dans lequel il est possible de publier des services et de les rechercher (UDDI est le standard)
* Description : cette couche normalise la description de l'interface publique d'un service web (WSDL (Web Service Description Language) est le standard)
* Communication : cette couche permet d'encoder les messages échangés (SOAP est le standard)
* Transport : cette couche assure le transport des messages : généralement HTTP est mis en oeuvre mais d'autres protocoles peuvent être utilisés (SMTP, FTP, ...)

La description d'un service web permet à son consommateur de connaître qu'elle est l'interface du service.

La communication permet de formaliser le format des messages échangés.

En plus de SOAP, WSDL et UDDI, il existe de nombreuses autres spécifications plus ou moins standards pour permettre la mise en œuvre de fonctionnalités manquantes dans ces standards comme la sécurité, la gestion des transactions, l'orchestration des services, ...

Ces spécifications sont en cours de développement ou d'évolution ce qui les rend généralement immatures. De plus, fréquemment, il existe plusieurs spécifications ayant trait à un même sujet qui sont donc concurrentes. La mise en œuvre de ces spécifications n'est pas requise pour des services web basiques mais elles peuvent être nécessaires pour des besoins plus spécifiques.

**SOAP**

SOAP (acronyme de Simple Object Acces Protocol jusqu'à sa version 1.1) est un standard du W3C qui permet l'échange formaté d'informations entre un client et un serveur. SOAP peut être utilisé pour la requête et la réponse de cet échange.

SOAP assure la partie messaging dans l'architecture des services web : il est utilisé pour normaliser le format des messages échangés entre le consommateur et le fournisseur de services web. SOAP est donc un protocole qui est principalement utilisé pour dialoguer avec des objets distribués comme peut le proposer JRMP utilisé par RMI, DCOM ou IIOP.

Son grand intérêt est d'utiliser XML ce qui le rend ouvert contrairement aux autres protocoles qui sont propriétaires : cela permet la communication entre un client et un serveur utilisant des technologies différentes. SOAP fait un usage intensif des espaces de nommages (namespaces).

SOAP est défini pour être indépendant du protocole de transport utilisé pour véhiculer le message. Cependant, le protocole le plus utilisé avec SOAP est HTTP car c'est un des protocoles le plus répandu et utilisé du fait de sa simplicité. Son utilisation avec SOAP permet de rendre les services web plus interopérables. De plus, cela permet aux services web de facilement traverser les firewalls du côté producteur et consommateur notamment dans le cas d'échanges via internet.

D'autres protocoles peuvent être utilisés (par exemple SMTP ou FTP) mais leur configuration sera plus délicate car elle ne sera pas fournie en standard comme c'est le cas avec HTTP. En fait, tous les protocoles capables de véhiculer un flux d'octets peuvent être utilisés.

SOAP est aussi indépendant de tout système d'exploitation et de tout langage de programmation car il utilise XML. Ceci permet une exposition et une consommation de services web avec des outils et des OS différents.

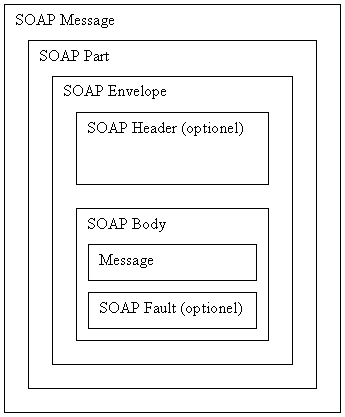
SOAP peut être utilisé pour :

* Appeler une méthode d'un service (SOAP RPC)
* Echanger un message avec un service (SOAP Messaging)
* Recevoir un message d'un service (selon la version de Soap)

Un message SOAP est contenu dans une enveloppe, ainsi le tag racine d'un document SOAP est le tag <Envelope>.

La structure d'une enveloppe SOAP se compose de plusieurs parties :

* Un entête optionnel composé d'un ou plusieurs headers : elle contient des informations sur le traitement du message
* Un corps (Body) : il contient les informations de la requête ou de la réponse
* Une gestion d'erreurs optionnelle (Fault) contenue dans le corps
* Des pièces jointes optionnelles (attachment) contenues dans le corps



L'entête contient des informations sur le traitement du message : ces informations sont contenues dans un tag <Header>. Pour des services web simples, cette partie peut être vide ou absente mais pour des services plus complexes elle peut contenir des informations concernant les transactions, la sécurité, le routage, etc ...

Le corps du message SOAP est contenu dans un tag <Body> obligatoire. Il contient les données échangées entre le client et le service sous la forme d'un fragment de document XML.

Tous ces éléments sont codés dans le message XML avec un tag particulier mettant en oeuvre un espace de nommage particulier défini dans les spécifications de SOAP.

Un message SOAP peut aussi contenir des pièces jointes contenues chacune dans une partie optionnelle nommée AttachmentPart. Ces parties appartiennent à la partie SOAP Part.

SOAP définit aussi l'encodage pour les différents types de données qui est basé sur la technologie schéma XML du W3C. Les données peuvent être de type simple (chaine, entier, flottant, ...) ou de type composé.

Les types simples peuvent être

* un type de base : string, int, float, ...
* une énumération
* un tableau d'octets (array of bytes)

Les types composés

* une structure (Struct)
* un tableau (Array)

La partie SOAP Fault permet d'indiquer qu'une erreur est survenue lors des traitements du service web. Cette partie peut être composée de 4 éléments :

* faultcode : indique le type de l'erreur (VersionMismatch en cas d'incompatibilité avec la version de SOAP utilisée, MustUnderstand en cas de problème dans le header du message, Client en cas de manque d'informations de la part du client, Server en cas de problème d'exécution des traitements par le serveur)
* faultstring : message décrivant l'erreur
* faultactor : URI de l'élément ayant déclenché l'erreur
* faultdetail

**WSDL**

WSDL (acronyme de Web Service Description Language) est utilisé pour fournir une description d'un service web afin de permettre son utilisation. C'est une recommandation du W3C.

Pour permettre à un client de consommer un service web, ce dernier a besoin d'une description détaillée du service avant de pouvoir interagir avec lui. Un WSDL fournit cette description dans un document XML. WSDL joue un rôle important dans l'architecture des services en assurant la partie description : il contient toutes les informations nécessaires à l'invocation du service qu'il décrit.

La description WSDL d'un service web comprend une définition du service, les types de données utilisées notamment dans le cas de types complexes, les opérations utilisables, le protocole utilisé pour le transport et l'adresse d'appel.

C'est un document XML qui décrit un service web de manière indépendante de tout langage. Il permet l'appel de ses opérations et l'exploitation des réponses (les paramètres, le format des messages, le protocole utilisé, ...).

WSDL est conçu pour être indépendant de tous protocoles. Ceci rend le standard WSDL flexible mais aussi plus complexe à comprendre. Comme SOAP et HTTP sont les deux protocoles les plus couramment utilisés pour implémenter les services web, le standard WSDL intègre un support de ces deux protocoles.

L'utilisation de XML permet à des outils de différents systèmes, plateformes et langages d'utiliser le contenu d'un WSDL pour générer du code permettant de consommer un service web. Les moteurs SOAP proposent en général un outil qui va lire le WSDL et générer les classes requises pour utiliser un service web avec la technologie du moteur SOAP. Le code généré utilise un moteur SOAP qui masque toute la tuyauterie du protocole utilisé et des messages échangés lors de la consommation de services web en agissant comme un proxy. Par exemple, Axis propose l'outil WSDL2Java pour la génération de ces classes à partir du WSDL.

Pour assurer une meilleure interopérabilité, WS-I Basic Profil 1.0 oblige à utiliser WSDL et les schémas XML pour la description des services web.

**Le format général d'un WSDL**

Un document WSDL définit plusieurs éléments :

* Type : la définition des types de données utilisés
* Message : la définition de la structure d'un message en lui attribuant un nom et en décrivant les éléments qui le composent avec un nom et un type
* PortType : la description de toutes les opérations proposées par le service web (interface du service) et identification de cet ensemble avec un nom
* Operation : la description d'une action proposée par le service web notamment en précisant les messages en entrée (input) et en sortie (output)
* Binding : la description du protocole de transport et d'encodage utilisé par un PortType afin de pouvoir invoquer un service web
* Port : référence un Binding (généralement cela correspond à l'url d'invocation du service web)
* Service : c'est un ensemble de ports

Un WSDL est un document XML dont le tag racine est <definitions> et qui utilise l'espace de nommage "http://schemas.xmlsoap.org/wsdl/".

Un WSDL est virtuellement composé de deux parties :

* des définitions abstraites : celles-ci concernent l'interface du service (types, message, portType). Ces informations sont exploitées dans le code du client
* des définitions concrètes : celles-ci concernent l'invocation du service (binding, service). Ces informations sont exploitées par le moteur SOAP.

Le contenu du WSDL d'un service nommé MonService est de la forme :

<!-Structure d'un WSDL -->

02.<definitions name="MonService"

03.targetNamespace="<http://com.jmdoudoux.test.ws.monservice/>"

04.xmlns="<http://schemas.xmlsoap.org/wsdl/>">

05.<!-- Définitions abstraites -->

06.<types> ... </types>

07.<message> ... </message>

08.<portType> ... </portType>

09.

10.<!-- Définitions concrètes -->

11.<binding> ... </binding>

12.<service> ... </service>

13.</definition>

L'ordre de définition des informations dans un WSDL facilite les traitements de ce document par une machine. Pour une exploitation par un humain, il est plus facile de lire le WSDL à l'envers (en commençant par la fin).

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

<wsdl:definitions targetNamespace="urn:WS" xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:impl="urn:WS" xmlns:intf="urn:WS" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<!--WSDL created by Apache Axis version: 1.3

Built on Oct 05, 2005 (05:23:37 EDT)-->

<wsdl:message name="addResponse">

<wsdl:part name="addReturn" type="xsd:int">

</wsdl:part>

</wsdl:message>

<wsdl:message name="sayHelloRequest">

</wsdl:message>

<wsdl:message name="authenticateRequest">

<wsdl:part name="usercode" type="xsd:string">

</wsdl:part>

<wsdl:part name="password" type="xsd:string">

</wsdl:part>

</wsdl:message>

<wsdl:message name="sayHelloResponse">

<wsdl:part name="sayHelloReturn" type="xsd:string">

</wsdl:part>

</wsdl:message>

<wsdl:message name="authenticateResponse">

<wsdl:part name="authenticateReturn" type="xsd:string">

</wsdl:part>

</wsdl:message>

<wsdl:message name="addRequest">

<wsdl:part name="n1" type="xsd:int">

</wsdl:part>

<wsdl:part name="n2" type="xsd:int">

</wsdl:part>

</wsdl:message>

<wsdl:portType name="WS">

<wsdl:operation name="add" parameterOrder="n1 n2">

<wsdl:input message="impl:addRequest" name="addRequest">

</wsdl:input>

<wsdl:output message="impl:addResponse" name="addResponse">

</wsdl:output>

</wsdl:operation>

<wsdl:operation name="authenticate" parameterOrder="usercode password">

<wsdl:input message="impl:authenticateRequest" name="authenticateRequest">

</wsdl:input>

<wsdl:output message="impl:authenticateResponse" name="authenticateResponse">

</wsdl:output>

</wsdl:operation>

<wsdl:operation name="sayHello">

<wsdl:input message="impl:sayHelloRequest" name="sayHelloRequest">

</wsdl:input>

<wsdl:output message="impl:sayHelloResponse" name="sayHelloResponse">

</wsdl:output>

</wsdl:operation>

</wsdl:portType>

<wsdl:binding name="WSSoapBinding" type="impl:WS">

<wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>

<wsdl:operation name="add">

<wsdlsoap:operation soapAction=""/>

<wsdl:input name="addRequest">

<wsdlsoap:body namespace="urn:WS" use="literal"/>

</wsdl:input>

<wsdl:output name="addResponse">

<wsdlsoap:body namespace="urn:WS" use="literal"/>

</wsdl:output>

</wsdl:operation>

<wsdl:operation name="authenticate">

<wsdlsoap:operation soapAction=""/>

<wsdl:input name="authenticateRequest">

<wsdlsoap:body namespace="urn:WS" use="literal"/>

</wsdl:input>

<wsdl:output name="authenticateResponse">

<wsdlsoap:body namespace="urn:WS" use="literal"/>

</wsdl:output>

</wsdl:operation>

<wsdl:operation name="sayHello">

<wsdlsoap:operation soapAction=""/>

<wsdl:input name="sayHelloRequest">

<wsdlsoap:body namespace="urn:WS" use="literal"/>

</wsdl:input>

<wsdl:output name="sayHelloResponse">

<wsdlsoap:body namespace="urn:WS" use="literal"/>

</wsdl:output>

</wsdl:operation>

</wsdl:binding>

<wsdl:service name="WSService">

<wsdl:port binding="impl:WSSoapBinding" name="WS">

<wsdlsoap:address location="http://localhost:8080/WSDEMO/services/WS"/>

</wsdl:port>

</wsdl:service>

</wsdl:definitions>

Un document WSDL est un document XML dont le tag racine est <definitions>. Généralement, ce tag contient la définition des différents espaces de nommage qui seront utilisés dans le document XML.

L'espace de nommage du document WSDL est défini.

Exemple :

xmlns=<http://schemas.xmlsoap.org/wsdl/> (déclaration de l'espace de nommage par défaut)

xmlns:wsdl=<http://schemas.xmlsoap.org/wsdl/>

L'espace de nommage de la norme Schema XML est défini.

Exemple :

xmlns:xsd=<http://www.w3.org/2001/XMLSchema>

Plusieurs autres espaces de nommages standards sont généralement définis selon les protocoles utilisés.

Exemple :

xmlns:soapenc=<http://schemas.xmlsoap.org/soap/encoding/>

xmlns:wsdlsoap=<http://schemas.xmlsoap.org/wsdl/soap/>

Il est possible de trouver la définition d'espaces de nommage propres à l'implémentation.

Exemple :

xmlns:apachesoap=<http://xml.apache.org/xml-soap>

Un ou plusieurs espaces de nommage sont définis pour le service web lui-même.

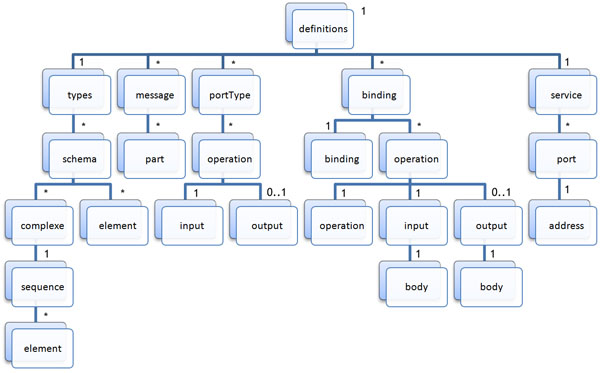
Remarque : les préfixes utilisés pour ces espaces de nommages peuvent être différents selon l'implémentation des services web mise en œuvre

Dans un document WSDL, les différentes entités font référence entre elles grâce à leur nom complet (espace de nommage et nom).

L'élément racine d'un WDSL est le tag <definitions>.

Le tag <definitions> peut contenir plusieurs tags fils :

* <types> : description des types de données utilisés
* <message> : description des messages qui peuvent être composés de plusieurs types
* <portType> : description des opérations de l’endpoint sous la forme d'échanges de messages. Ceci correspond à l'interface du service
* <binding> : description du protocole et spécification du format des données pour un portType
* <service> : description des endpoints du service (binding et uri)



**mise en oeuvre côté serveur**

**package** wsdemo.cls.ws;

**import** java.rmi.Remote;

**public** **class** WS **implements** Remote {

**public** String authenticate(String usercode, String password) {

**return** "client called this webservice with usercode=" + usercode

+ " and password=" + password + ".\n\nHello " + usercode + "!";

}

**public** **int** add(**int** n1, **int** n2)

{

**return** n1 + n2;

}

**public** String sayHello()

{

**return** "Hello";

}

}

Apres avoir compilé cette classe et mettre le fichier .class dans le répertoire WEB-INF/classes de la webapps axis, Exécuter axis.bat, un nouveau fichier est créé deploy.wsdd:

<!-- Use this file to deploy some handlers/chains and services -->

<!-- Two ways to do this: -->

<!-- java org.apache.axis.client.AdminClient deploy.wsdd -->

<!-- after the axis server is running -->

<!-- or -->

<!-- java org.apache.axis.utils.Admin client|server deploy.wsdd -->

<!-- from the same directory that the Axis engine runs -->

<deployment

xmlns="http://xml.apache.org/axis/wsdd/"

xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">

<!-- Services from WSService WSDL service -->

<service name="WS" provider="java:RPC" style="rpc" use="literal">

<parameter name="wsdlTargetNamespace" value="urn:WS"/>

<parameter name="wsdlServiceElement" value="WSService"/>

<parameter name="wsdlServicePort" value="WS"/>

<parameter name="className" value="wsdemo.cls.ws.WSSoapBindingImpl"/>

<parameter name="wsdlPortType" value="WS"/>

<parameter name="typeMappingVersion" value="1.2"/>

<operation name="add" qname="operNS:add" xmlns:operNS="urn:WS" returnQName="addReturn" returnType="rtns:int" xmlns:rtns="http://www.w3.org/2001/XMLSchema" soapAction="" >

<parameter qname="n1" type="tns:int" xmlns:tns="http://www.w3.org/2001/XMLSchema"/>

<parameter qname="n2" type="tns:int" xmlns:tns="http://www.w3.org/2001/XMLSchema"/>

</operation>

<operation name="authenticate" qname="operNS:authenticate" xmlns:operNS="urn:WS" returnQName="authenticateReturn" returnType="rtns:string" xmlns:rtns="http://www.w3.org/2001/XMLSchema" soapAction="" >

<parameter qname="usercode" type="tns:string" xmlns:tns="http://www.w3.org/2001/XMLSchema"/>

<parameter qname="password" type="tns:string" xmlns:tns="http://www.w3.org/2001/XMLSchema"/>

</operation>

<operation name="sayHello" qname="operNS:sayHello" xmlns:operNS="urn:WS" returnQName="sayHelloReturn" returnType="rtns:string" xmlns:rtns="http://www.w3.org/2001/XMLSchema" soapAction="" >

</operation>

<parameter name="allowedMethods" value="authenticate add sayHello"/>

</service>

</deployment>

L'extension wsdd signifie WebService Deployment Descriptor.

C'est un document xml dont le tag racine est deployment.

Les informations relatives au service web sont définies dans le tag service qui possède plusieurs attributs notamment :

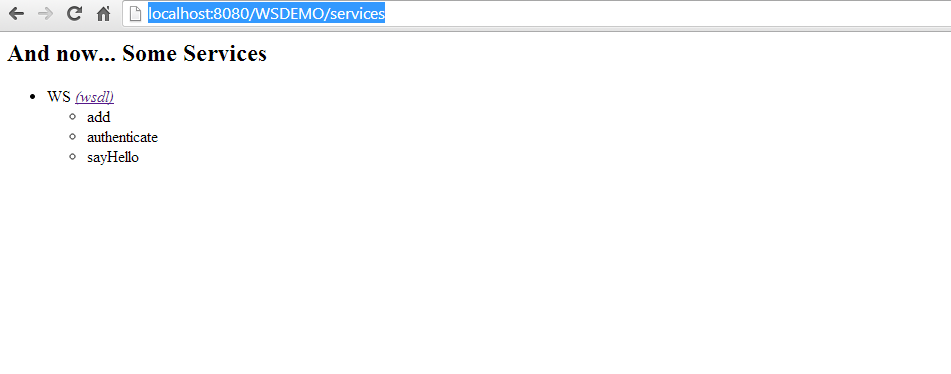
* name :
* provider :

Plusieurs informations doivent être fournies avec un tag parameter qui possède les attributes name et value :

* className : le nom pleinement qualifié de la classe d'implémentation du service web
* allowedMethods : précise les méthodes qui sont exposées. Le caractère étoile permet d'indiquer toutes les méthodes

**Mise en œuvre côté client**

Dans Visual Studio, créer un nouveau projet (ici j’ai créé un projet Console application C#) et j’ai appelé le service web que j’ai créé en java exemple : <http://localhost:8080/WSDEMO/services>



Code C# ou j’ai appelé les fonction crées du service web :

using System;

using System.Collections.Generic;

using System.Text;

namespace wsclienttest

{

class Program

{

static void Main(string[] args)

{

int n1=0, n2=0;

Console.Write("testing webservice from c# console application\nplease provide usercode:");

String usercode = Console.ReadLine();

Console.Write("enter password:");

String password = Console.ReadLine();

wsdemo.WSService webservice = new wsdemo.WSService();

String ret = webservice.authenticate(usercode, password);

Console.WriteLine("service returned: " + ret);

Console.WriteLine("-----------------");

Console.WriteLine("enter a number n1: ");

try

{

n1 = Int32.Parse(Console.ReadLine());

}

catch (Exception)

{

Console.WriteLine("Please enter a valid number!");

}

Console.WriteLine("enter another number n2: ");

try

{

n2 = Int32.Parse(Console.ReadLine());

}

catch (Exception)

{

Console.WriteLine("Please enter a valid number!");

}

if (n1 > 0 && n2 > 0)

{

Console.WriteLine("n1 + n2 = "+ webservice.add(n1, n2));

}

else

Console.WriteLine("Numbers should be different than 0!");

Console.Read();

}

}

}