SOA
SOAP Web Services

Pré-requis

* Testé avec
  * JDK13
  * TOMEE+ 8.0.1
  * Netbeans 11.3
  * Springboot 2.2.5

* Également besoin de
  * Ant(?)
  * Mvn
  * Git
  * PHP
  * SoapUI

Web services Rest

* Simples, légers

* Appel de ressources via verbes HTTP
* Pas d'états (stateless WS)
  * (http: non connecté)
* Sérialisation aux soins du dev.
* Pas de normes de sécurité particulière

Services Web SOAP

* Répond aux besoins précédents
* Ajout du protocole SOAP (XML) au dessus d'HTTP
  * HTTP n'est plus utilisé, transparent
  * Définition des sérialisations, des accès (RPC encoded, document/literal, etc.)
  * Normes de sécurité (WS-Trust, WS-security)

* Plus complexe, plus lourds

Contenu du cours

* Présentation de WS SOAP
  * Description WSDL, sérialisation SOAP

* Introduction de Frameworks
* JAXWS via Netbeans
* Apache Axis2
  * (fait tout, mais en lcmd, rest en plus)
* Springboot

Description SOAP, WSDL,

Soap vs. Rest

	SOAP	REST
Meaning	Simple Object Access Protocol	Representational State Transfer
Design	Standardized protocol with pre-defined rules to follow.	Architectural style with loose guidelines and recommendations.
Statefulness	Stateless and stateful.	Stateless (no server-side sessions).
Caching	API calls cannot be cached.	API calls can be cached.
Security	WS-Security with SSL support. Built-in ACID compliance.	Supports HTTPS and SSL.
Performance	Requires more bandwidth and computing power.	Requires fewer resources.
Message format	Only XML.	Plain text, HTML, XML, JSON, YAML, and others.
Transfer protocol(s)	HTTP, SMTP, UDP, and others.	Only HTTP
Recommended for	Enterprise apps, high-security apps, distributed environment, financial services, payment gateways, telecommunication services.	Public APIs for web services, mobile services, social networks.
Advantages	High security, standardized, extensibility.	Scalability, better performance, browser-friendliness, flexibility.
Disadvantages	Poorer performance, more complexity	Less security, not suitable for distributed environments. No standard

Tiré en partie de https://raygun.com/blog/soap-vs-rest-vs-json/

web services SOAP

Utilisation d'HTTP:

- Protocole Internet
- Beaucoup d'entreprises possèdent un serveur web
- Protocole généralement autorisé au niveau de parefeu
- Protocole disponible sur toutes les plateformes
- Mode non connecté

Utilisation d'XML:

- Massivement utilisé et reconnu
- Permet de structurer l'information facilement

Pourquoi les web services ?

Implémentations actuelles (SOAP) :

Microsoft .Net

Sun Metro (JaxWS, etc.)

Apache Axis2 et CXF

IBM WSTK

Oracle, Bea, Iona, Enhydra …

Quelques normes

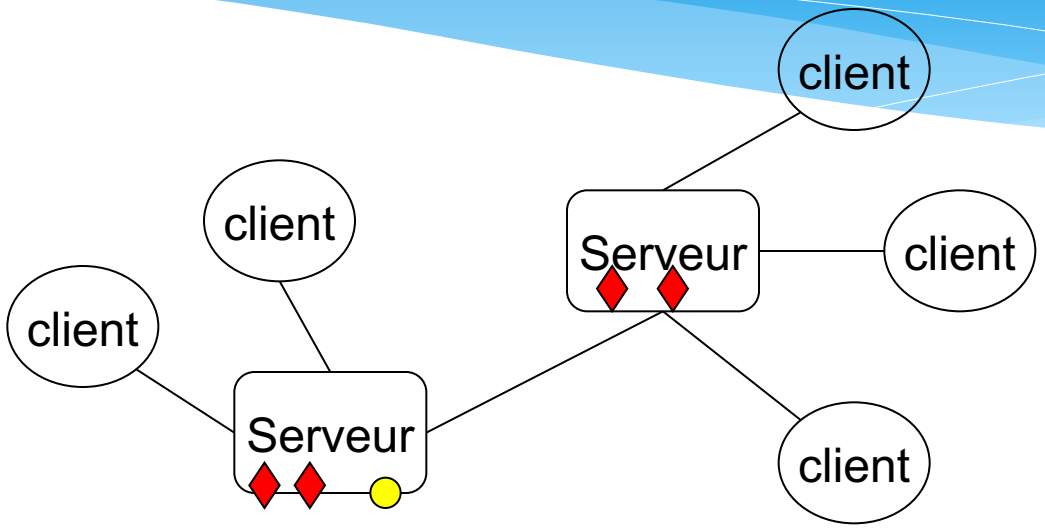
WS-I Organisation a pour but la normalisation de schémas (XML pour SOAP WSDL) interopérable

But: normalisation des messages

- WS-I Basic Profile 1.1
  - Mode d'utilisation de SOAP, WSDL et UDDI
  - Support du DOCUMENT/Literal et RPC/Literal
  - Pas de prise en compte des attachements

- WS-I Attachments Profile 1.1
  - Couvre la gestion des attachements
  - Repose sur SOAP 1.1 With Attachments

les web services

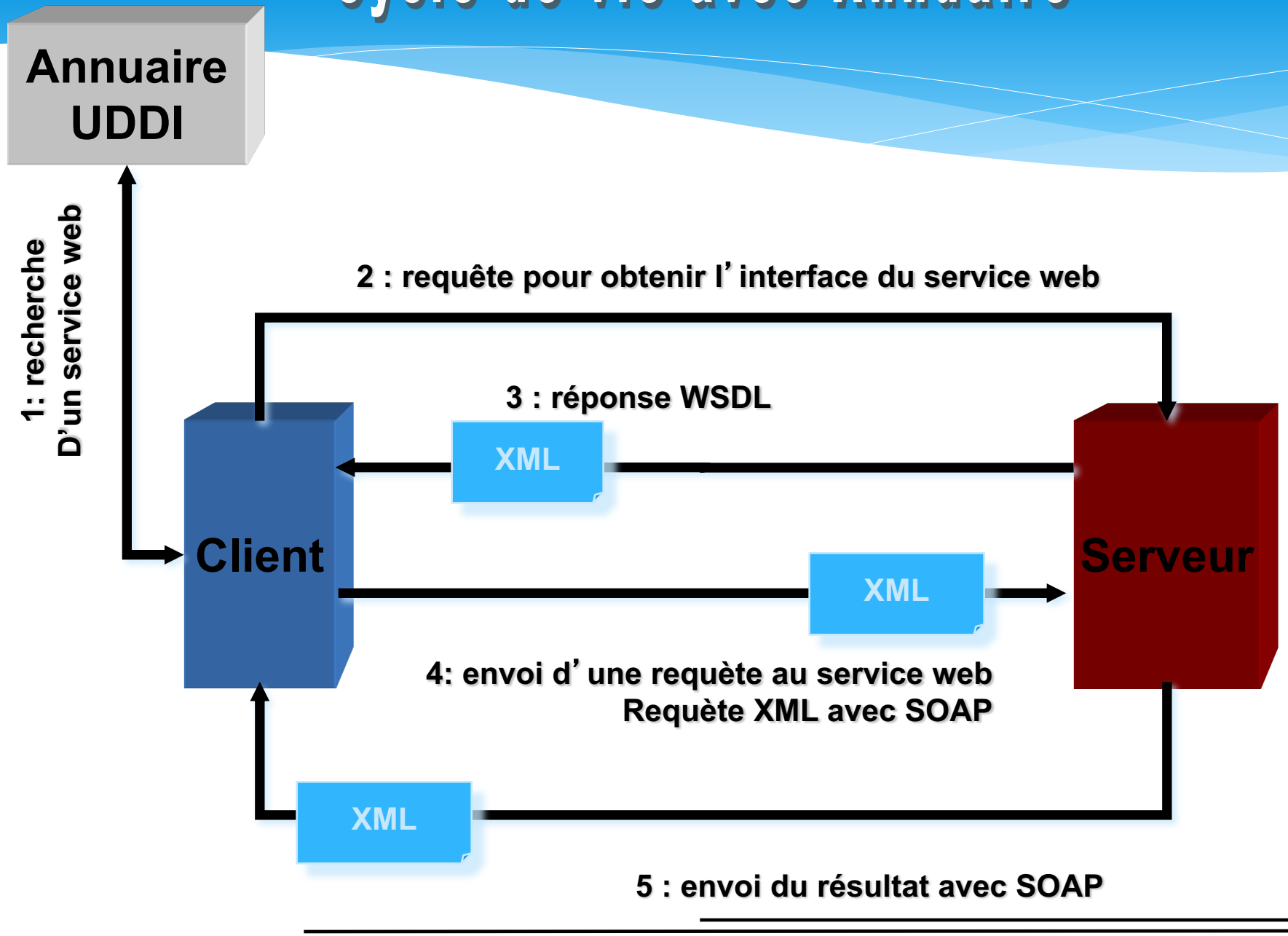


- Service web accessible via les protocoles HTTP+SOAP
- Page jaune de services SOAP avec le protocole UDDI

Cycle de vie:

1. Accès à un annuaire UDDI pour localiser un service
2. Récupération de l'interface du service web
3. Appel au service (requète, réponse)

Cycle de vie avec Annuaire



les web services

•Le concept de service web s'articule autour de 3 protocoles:

•SOAP (Simple Object Access Protocol) est un protocole permettant l'échange de données quelquesoit la plateforme. Un appel de service SOAP coorespond à un flux de données XML sur HTTP

•WSDL (Web Service Description Language) permet de décrire au format XML les fonctionnalités du service web (méthodes, URLs, signature, port,…)

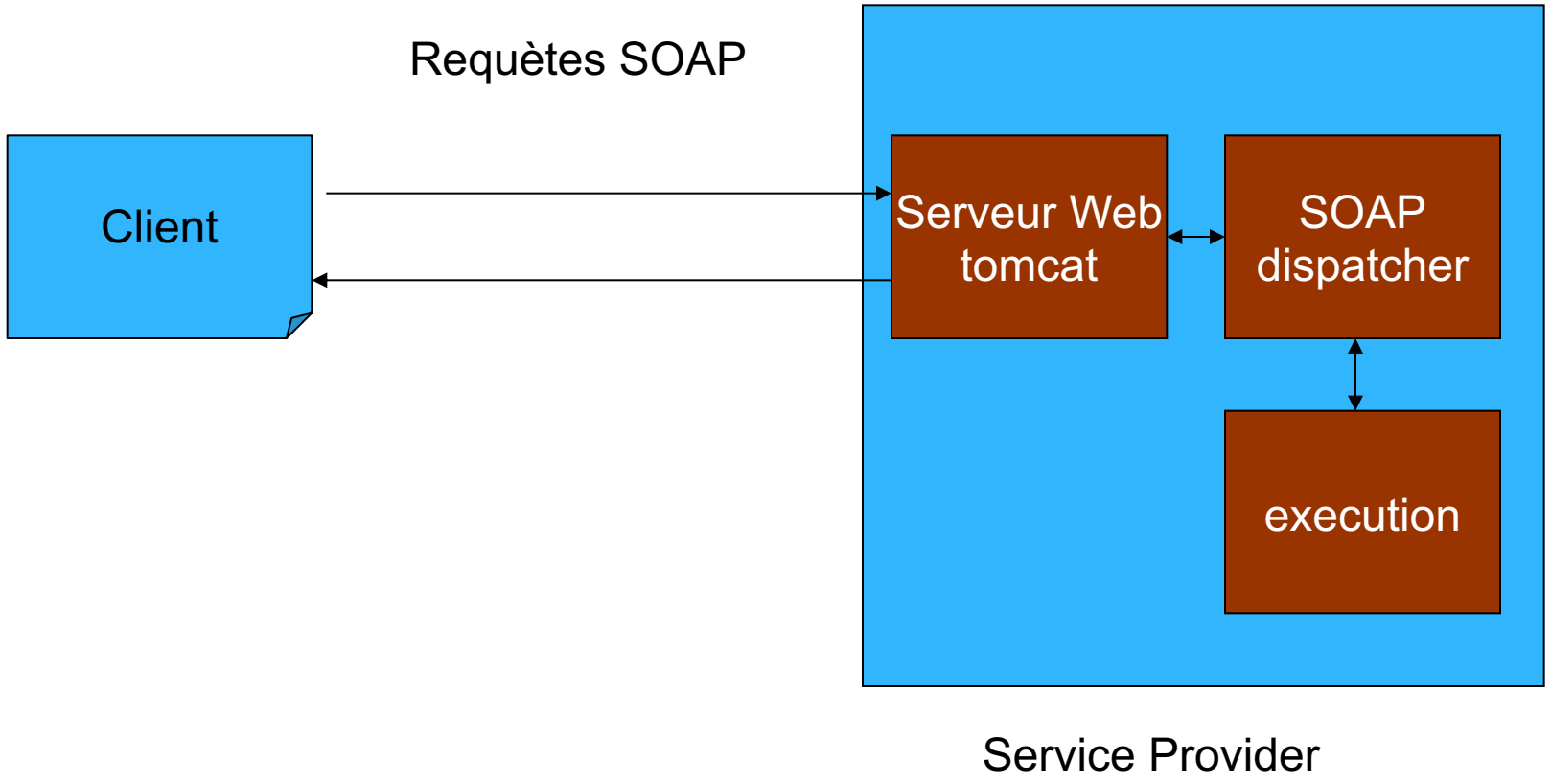
•UDDI (Universal Description Discovery and Integration) normalise un solution d'annuaire des service web

•Il existe aussi des protocoles pour XML spécifiques à la sécurité

Les protocoles, WSDL, SOAP, UDDI

Le protocole SOAP

Protocole permettant de faire des appels de méthodes de services Web décrits en XML et encapsulés sur HTTP(mode synchrone) ou sur SMTP,JMS(java message service) (mode asynchrone)



Le protocole SOAP

- Service synchrones
  - sur HTTP
  - appels bloquants
  - Pas de garanties de retour
  - principalement utilisé

- Service asynchrones
  - **sur SMTP**, JMS
  - non bloquants
  - garantie de service (1 retour)

Le protocole SOAP

Un message SOAP doit utiliser les balises SOAP et doit avoir la forme suivante:

```
<soap:Envelope

xmlns:soap=http://www.w3.org/2001/12/soap-envelope
soap:encodingStyle=http://www.w3.org/2001/12/soap-encoding>

        <soap:Header>
        …
        </soap:Header>

        <soap:Body>
        …
        <soap:Fault>
        …
        </soap:Fault>

        </soap:Body>

</soap:Envelope>
```

Les messages SOAP

•La balise <SOAP-ENV:Envelope… spécifie le type d'encodage et contient des définitions de namespace

•Les entêtes SOAP (optionnelles) permettent d'échanger des informations d'authentification ou de gestion de session

•Les balises <SOAP-ENV:Body> … </SOAP-ENV:Body> contiennent le corps du message = description des paramètres, retours des méthodes

•La balise de la méthode contient son namespace (MySOAPSERVICE) (nom unique de la méthode sur Internet) puis les paramètres (et surtout leurs types)

•Les paramètres peuvent être de plusieurs types et peuvent être plus ou moins complexes (chaîne, tableaux,…)

Le protocole SOAP

Rappel sur les namespaces

espace de nommage XML = collection de noms (RFC 2396)

utilisés comme type d'éléments et noms d'attribut

```
<x
xmlns:edi='http://ecommerce.org/schema>
</x>
```

Edi est lié à http://...

Pour l'élément x et son contenu

Namespace préfixes:

Soap « http://schemas.xmlsoap.org/soap/envelope/ »

Wsdl « http://schemas.xmlsoap.org/wsdl/ »

…

Le protocole SOAP

Un exemple de message SOAP en RPC literal

```
package com.jwsbook.soap;
import java.rmi.RemoteException;

public interface BookQuote extends java.rmi.Remote {
        // Get the wholesale price of a book
        public float getBookPrice(String ISBN) throws RemoteException, InvalidISBNException;
}
```

requête

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:mh="http://www.Monson-Haefel.com/jwsbook/BookQuote">
<soap:Body>
        <mh:getBookPrice>
                <isbn>0321146182</isbn>
        </mh:getBookPrice>
</soap:Body>
</soap:Envelope>
```

Le protocole SOAP

SOAP propose différents types d'encodage:

- 2 types de messages:
  - RPC: appel d'une méthode d'un objet distant avec passage de paramètres.
  - Document: plusieurs parties pour l'appel, pour la méthode, pour les paramètres, pas de formatage

- Pour les applications qui sérialisent/désérialisent les données (paramètres ou retours), il existe aussi 2 types d'encodage:
  - SOAP encoded: les données sont sérialisées via des codages SOAP (généralement SOAP RPC)
  - Literal: données sérialisées par schemas XML. Pas de règles définies. Document/literal choisi par microsoft

Le protocole SOAP

Exemples (trouvé sur ibm.com):

On part de cette méthode: public void myMethod(int x, float y);

Voici les différents messages (enveloppes) SOAP et descriptions WSDL possibles:

La description WSDL contient la structure des appels (RPC) ou le schéma (structure des messages SOAP) XML (Document)

Le protocole SOAP

RPC encoded:

WSDL:

```
<message name="myMethodRequest">
        <part name="x" type="xsd:int"/>
        <part name="y" type="xsd:float"/>
</message>
<message name="empty"/>
<portType name="PT">
<operation name="myMethod">
        <input message="myMethodRequest"/>
        <output message="empty"/>
</operation>
</portType>
<binding .../>
```

SOAP:

```
<soap:envelope>
<soap:body>
        <myMethod>
        <x xsi:type="xsd:int">5</x>
        <y xsi:type="xsd:float">5.0</y>
        </myMethod>
</soap:body>
</soap:envelope>
```

deprecated

⇒Non-conforme à WS-I basic profile, peu performant (à cause des types)

⇒Pb d'interopérabilité !

Le protocole SOAP

RPC literal:

WSDL:

```
<message name="myMethodRequest">
        <part name="x" type="xsd:int"/>
        <part name="y" type="xsd:float"/%gt;
</message>
<message name="empty"/>
<portType name="PT">
        <operation name="myMethod">
        <input message="myMethodRequest"/>
        <output message="empty"/>
</operation>
</portType>
<binding .../>
```

SOAP:

```
<soap:envelope>
<soap:body>
        <myMethod>
        <x>5</x>
        <y>5.0</y>
        </myMethod>
</soap:body>
</soap:envelope>
```

deprecated

⇒Conforme WS-I basic profile

⇒Encodage XML, mais difficile de valider le message (myMethod ?)

Document encoded:

=> Pas utilisé, non-conforme WS-I

Le protocole SOAP

Document literal:

WSDL:

```
<types>
<schema>
        <element name="xElement" type="xsd:int"/>
        <element name="yElement" type="xsd:float"/>
</schema>
</types>
        <message name="myMethodRequest">
                <part name="x" element="xElement"/>
                <part name="y" element="yElement"/>
        </message>
        <message name="empty"/>
        <portType name="PT">
        <operation name="myMethod">
                <input message="myMethodRequest"/>
                <output message="empty"/>
        </operation>
</portType>
<binding .../>
```

SOAP:

```
<soap:envelope>
<soap:body>
        <xElement>5</xElement>
        <yElement>5.0</yElement>
</soap:body>
</soap:envelope>
```

Le protocole SOAP

Document literal:

•Conforme WS-I basic profile si une rubrique dans soap:body
(ici on en a 2 !)
=> interopérable

•Plus flexible

•pas d'encodage déterminé, n'importe quel XML validator peut valider ce message

•Perte de l'opération, méthode mappée par rapport a ses paramètres

pas possible dans un même service web d'avoir deux méthodes avec la même liste de paramètres

Le protocole SOAP

Document/literal wrapped

WSDL:

```
<types> <schema>
        <element name="myMethod">
        <complexType>
        <sequence>
                <element name="x" type="xsd:int"/>
                <element name="y" type="xsd:float"/>
        </sequence>
        </complexType>
        </element>
        <element name="myMethodResponse">
        <complexType/>
        </element>
</schema> </types>
<message name="myMethodRequest">
        <part name="parameters" element="myMethod"/>
</message>
<message name="empty">
        <part name="parameters" element="myMethodResponse"/>
</message>
<portType name="PT"> <operation name="myMethod">
        <input message="myMethodRequest"/>
        <output message="empty"/>
</operation> </portType>
<binding .../>
```

SOAP:

```
<soap:envelope>
<soap:body>
        <myMethod>
        <x>5</x>
        <y>5.0</y>
        </myMethod>
        </soap:body>
</soap:envelope>
```

Le protocole SOAP

Document/literal wrapped:

- L'opération est encapsulé dans un élément XML portant le nom de l'opération

  ⇒meilleure interopérabilité ?

- le meilleur de RPC et document literal ?

- Proposé par microsoft

- Respecte le WS-I basic profile si une rubrique par soap:body
  (ici c'est bien le cas)

- Messages plus complexes => performance ?

- Semble la meilleure solution (?)

Le protocole SOAP

•Mais que fau-il utiliser ?

1. Ne pas utiliser Document/literal wrapped avec des méthodes surchargées
   (=> 2 éléments avec le même nom)
   utiliser document literal

2. Si méthodes surchargées et renvoi de méthodes => utiliser rpc literal
   selon quelques tests, RPC literal semble être le plus performant ?

deprecated

3. Le type RPC encoded peut être interressant pour l'utilisation de données spécifiques
   (arbres,…)
   RPC encoded permet de créer un lien entre données (href…)

Le protocole SOAP

Utilisation des attachments (par email !)

•Utilisation pour un gros volume de données en entrée ou sortie du service web

•Souvent utilisé pour des services web asynchrones (traitement lourds)

•Pas de sérialisation/deserialisation automatique, A faire !

générallement en XML=> découpage en fragments et déserialisation

(dans des threads si bq de données)

Les messages SOAP

Les messages d'erreurs:

Messages d'erreurs générés par les destinataires d'un message

Message d'erreur: une seul élément erreur dans le Body SOAP

Plusieurs codes d'erreurs: client, server, VersionMismatch, MustUnderstand
Ex: Client.Authentication

Messages d'erreurs supplémentaires: messages HTTP !!!! (400 bad request, 405 method not allowed, 500 internal server error…)

Les messages SOAP

Un message d'erreur est composé de 4 éléments:

- Faultcode (obligatoire)
  Code d'erreur utilisé par le logiciel (utilisation de la classe SOAPException)

- Faultstring (obligatoire)
  Explication lisible d'un humain

- faultactor (optionnel)
  Erreur en cours de cheminement du message (firewall, proxy, MOM)

- Detail
  Détail de l'erreur non lié au Body du message

- Autres
  D'autres éléments qualifiés par un namespace peuvent être ajoutés

Les messages SOAP

Un message d'erreur

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:mh="http://www.Monson-Haefel.com/jwsbook/BookQuote" > *
<soap:Body>
 <soap:Fault>
          <faultcode>soap:Client</faultcode>
          <faultstring> The ISBN value contains invalid characters </faultstring>
          <faultactor>http://www.xyzcorp.com</faultactor>

          <detail> <mh:InvalidIsbnFaultDetail>
          <offending-value>19318224-D</offending-value>
          <conformance-rules>
          The first nine characters must be digits.
          The last character may be a digit or the letter 'X'. Case is not important.
          </conformance-rules>
          </mh:InvalidIsbnFaultDetail>
          </detail>
</soap:Fault>
</soap:Body>
</soap:Envelope>
```

Utilisation de SOAP coté client

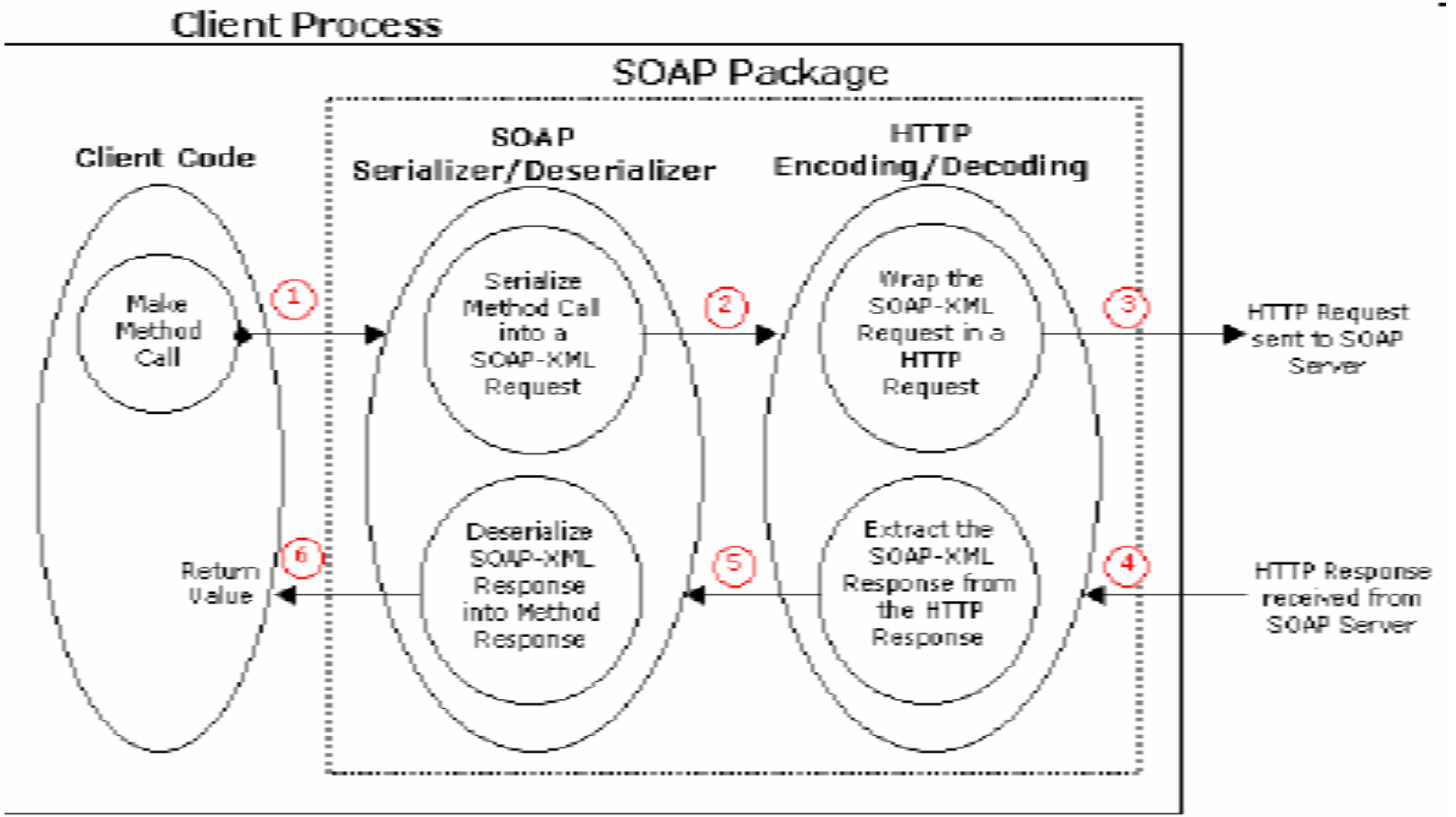
Les messages envoyés au serveur seront de la forme SOAP XML de même que les réponses du serveur

⇒Besoin de sérialiser les messages. Utilisation simple d'un package SOAP (par héritage) qui fait le travail pour vous

Invocation du service en appelant une de ses méthodes grâce au package SOAP en spécifiant l'url du service, les paramètres

Le package SOAP sérialise l'invocation, encode le message dans une requète HTTP puis effectue une requête au serveur web à l'url du service web. A la réponse du service web, le package SOAP effectue l'inverse

Utilisation de SOAP coté client



Utilisation de SOAP coté serveur

•Le serveur repose principalement sur un package SOAP identique au client

•Il possède en plus un listener (processus d'écoute) en attente de connexion de clients. Ce listener est implanté par une servlet avec Axis

•La servlet SOAP extrait le message SOAP-XML, la de-serialise (récupération du nom de la méthode, des paramètres), puis invoque la méthode concernée du service. Le résultat et sérialisé puis renvoyé au client

Le protocle WSDL

Langage XML de description de services web

description normalisée qui permet une utilisation inter-plateforme

cache le code du web service

Regroupe :

- Les méthodes du service web
- Les paramètres et les valeurs de retour
- Le protocole de transfert (SOAP ou autre)
- La localisation

Les documents WSDL peuvent être écrits à la main (oupss) mais sont surtout générés

Le protocle WSDL

Un document WSDL est constitué de plusieurs parties (7) qui sont principalement:

- L'url du service
- Le protocole associé à l'accès au service (SOAP, RPC,…)
- Les méthodes du service
- Les paramètres d'entrées, sorties et leurs encodages

Le protocle WSDL

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

- **types**: cette balise décrit les types utilisés
- **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

Le message WSDL

Partie 1: description des types utilisés par le service (tableau, vecteur,…)

Exemple:

```
xsd:complexType name="phone">
<xsd:all>
        <xsd:element name="areaCode" type="xsd:int"/>
        <xsd:element name="exchange" type="xsd:string"/>
        <xsd:element name="number" type="xsd:string"/>
</xsd:all>
</xsd:complexType>
```

Le message WSDL

Partie 2: les messages (ensemble des données transmises) description de la requète HTTP et de la réponse HTTP avec les paramètres mis en jeu

```
<message name="getTermRequest">
        <part name="term" type="xs:string"/>
</message>
<message name="getTermResponse">
        <part name="value" type="xs:string"/>
</message>
```

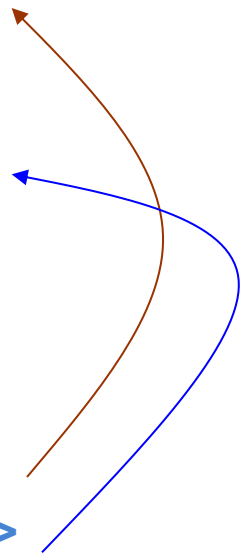
Le message WSDL

Partie 3: les types de port (desc la plus importante) définissent les points de connections au service web (peut être comparé à une classe)

Opération type requête réponse (il y en a d'autres):

```
<message name="getTermRequest">
        <part name="term" type="xs:string"/>
</message>
<message name="getTermResponse">
        <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
<operation name="getTerm">
        <input message="getTermRequest"/>
        <output message="getTermResponse"/>
</operation>
</portType>
```



Le message WSDL

WSDL définit 4 types d'opération :

•One-Way : lorsque les opérations reçoivent des messages mais n'en n'envoient pas

•Request-response : lorsque les opérations reçoivent des messages puis renvoient des Messages

•Solicit-response : lorsque les opérations envoient des messages puis en reçoivent

•Notification : lorsque les opérations envoient des messages mais n'en reçoivent pas

•Une opération :

Reçoit des messages : <wsdl:input …>
Envoie des messages : <wsdl:output …> ou <wsdl:fault …>

Le message WSDL

Partie 4: lier une description abstraite (portType) à un protocole.

(à la suite du code XML précèdent, on trouve…)

```
<binding type="glossaryTerms" name="b1">
<soap:binding style="document" transport=
"http://schemas.xmlsoap.org/soap/http" />

<operation>
<soap:operation soapAction="http://example.com/getTerm"/>
        <input>
        <soap:body use="literal"/>
        </input>

        <output>
        <soap:body use="literal"/>
        </output>
</operation>

</binding>
```

Type port, nom binding

Style: document ou rpc

Protocole (http)

Opération du port

Style d'encodage

Le message WSDL

Partie 5: les ports (différents de type de port) définissent l'url chez le fournisseur, effectue la relation entre lien et url

Partie 6: partie service qui rassemble l'ensemble des ports

Nom du lien

```
<wsdl:service name=« nomservice" >
        <wsdl:port name=" nomport" binding=" impl: glossaryTerms ">
        <wsdlsoap: address location=" http://serveur:8080/axis/nomport"  />
        </wsdl:port>
</wsdl:service>
```

Le message WSDL

Exemples

```
−<wsdl:definitions name="NewWebService" targetNamespace="http://ws/">
  −<wsdl:types>
    −<xs:schema elementFormDefault="unqualified" targetNamespace="http://ws/" version="1.0">
        <xs:element name="hello" type="tns:hello"/>
        <xs:element name="helloResponse" type="tns:helloResponse"/>
      −<xs:complexType name="hello">
        −<xs:sequence>
            <xs:element minOccurs="0" name="name" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      −<xs:complexType name="helloResponse">
        −<xs:sequence>
            <xs:element minOccurs="0" name="return" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
    </xs:schema>
  </wsdl:types>
−<wsdl:message name="helloResponse">
    <wsdl:part element="tns:helloResponse" name="parameters"> </wsdl:part>
  </wsdl:message>
−<wsdl:message name="hello">
    <wsdl:part element="tns:hello" name="parameters"> </wsdl:part>
  </wsdl:message>
−<wsdl:portType name="NewWebService">
  −<wsdl:operation name="hello">
      <wsdl:input message="tns:hello" name="hello"> </wsdl:input>
      <wsdl:output message="tns:helloResponse" name="helloResponse"> </wsdl:output>
    </wsdl:operation>
  </wsdl:portType>
−<wsdl:binding name="NewWebServiceSoapBinding" type="tns:NewWebService">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  −<wsdl:operation name="hello">
      <soap:operation soapAction="" style="document"/>
```

Schema fonctionnel