Composition d'applications réparties

Services Web

Sacha Krakowiak
Université Joseph Fourier
Projet Sardes (INRIA et IMAG-LSR)

http://sardes.inrialpes.fr/~krakowia

Services Web (1)

- **Motivations**
  - Intégration d'applications "à gros grain"
  - Unité d'intégration : le "service" (interface + contrat)
- **Contraintes**
  - Applications conçues indépendamment, sans avoir prévu une intégration future
  - Applications hétérogènes (modèles, plates-formes, langages)
- **Conséquences**
  - Pas de définition d'un modèle commun, seulement de protocoles et d'interfaces
  - Utilisation d'une base commune de niveau élémentaire
    - Pour les protocoles d'accès aux services
    - Pour la description des services
  - Outil de base : XML (car adaptable)

Services Web (2)

Objectifs des Services Web

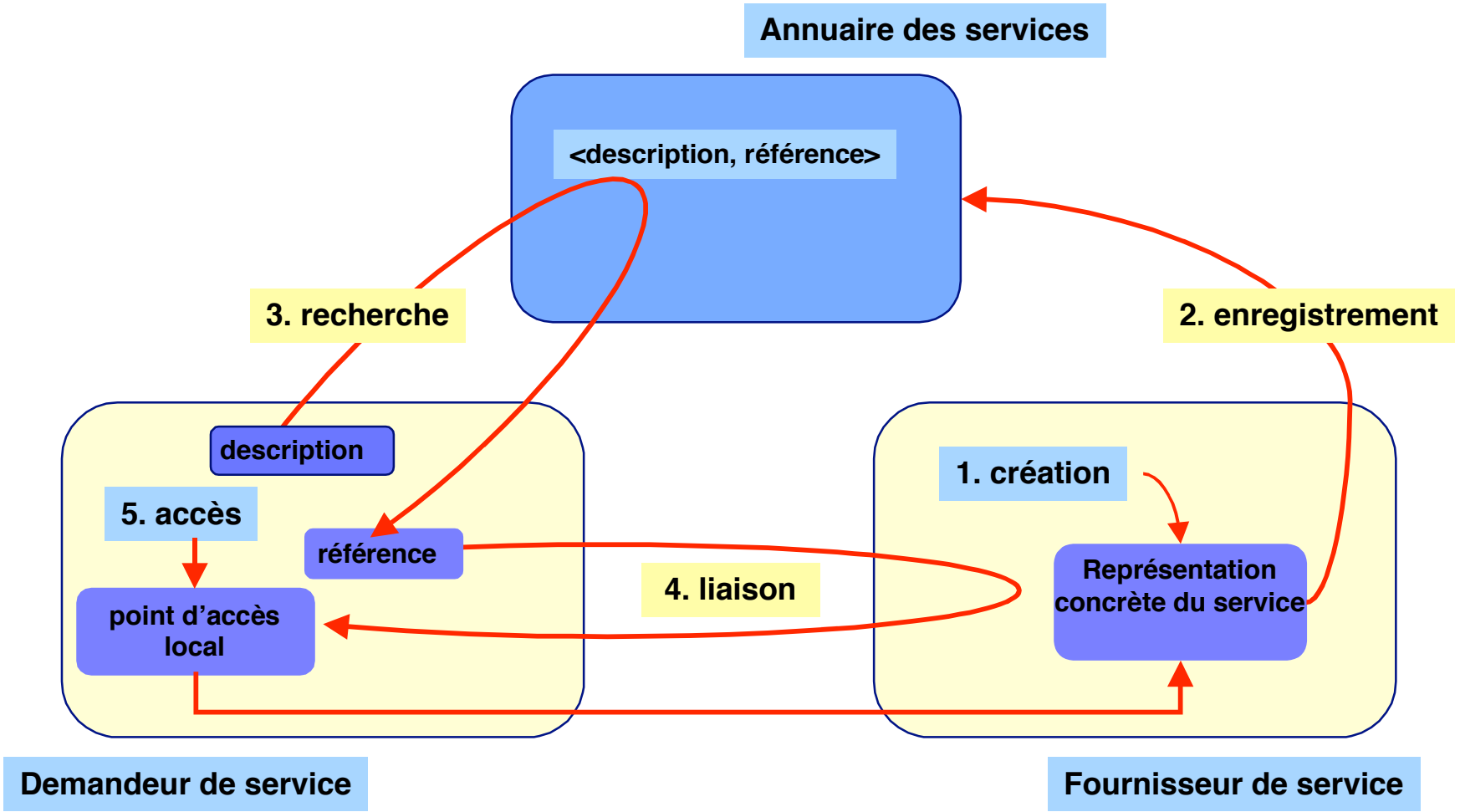
- Permettre l'interopérabilité des applications, indépendamment des plates-formes et des langages
- Permettre le couplage faible des applications (évolution indépendante) et leur coopération via des interfaces de haut niveau d'abstraction (services globaux)
- Permettre une coopération des applications avec un minmum d'intervention humaine

Conséquences

- Un service doit être auto-descriptif
- Un service doit pouvoir être facilement trouvé (à partir d'une description)

Les Services Web sont définis par un ensemble de standards qui permettent de décrire des interfaces logicielles et d'accéder aux fonctions correspondantes sur un réseau via des messages exprimés en XML. Les constructions ainsi réalisées constituent une architecture à base de services (*Service Oriented Architecture*, ou SOA)

Accès à un service (rappel)



Services Web (3)

Apport conceptuel

- Pas de nouveaux concepts fondamentaux …
- … alors, quelle est la nouveauté ?

Apport concret

- Avant tout, solution en vue au problème de l'hétérogénéité
- Vers un schéma d'intégration et de coordination d'applications à grande échelle
- Forte implication des principaux acteurs du domaine
- Beaucoup de problèmes techniques restent encore à résoudre

Brefs rappels sur XML (1/5)

- **XML = *eXtensible Markup Language***
- **Motivation**
  - Fournir un formalisme pour la description de données structurées (essentiellement structure d'arbre)
- **Choix de base**
  - Fournir des éléments pour la construction de langages spécialisés plutôt qu'un langage universel. Avantages :
    - formalisme adaptable à des situations très diverses
    - formalisme extensible
- **Origines et historique**
  - Initialement inspiré par le problème de l'évolution d'HTML ; dérivé de SGML (*Standard Generalized Markup Language*), utilisé pour la description de documents structurés
  - Standard du W3C (depuis février 1998), voir http://www.w3.org/XML/

Brefs rappels sur XML (2/5)

Document XML

- Forme
  - une chaîne de caractères
- Structure
  - données séparées par des balises

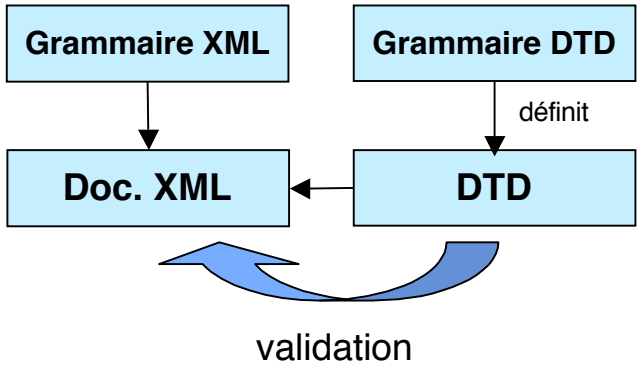
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<adresse>
   <lieu>
        <rue> avenue de l'Europe</rue>
        <numero>655</numero>
   </lieu>
    <ville>Montbonnot></ville>
    <code>38330</code>
</adresse>
```

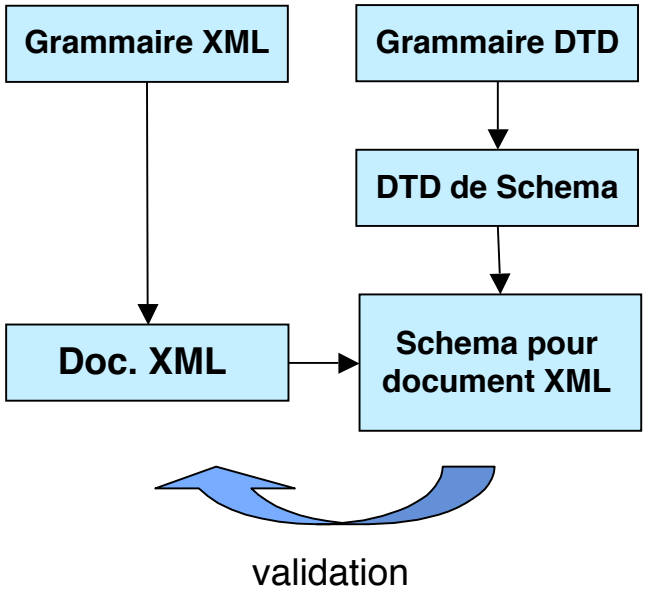
Propriétés

- Bonne forme (*well formedness*)
  - Conforme à des règles de syntaxe spécifiées ; concerne la forme du document. Exemple : jeu de caractères, parenthésage complet des balises, attributs entre " ", etc.
- Validité
  - Contraintes logiques (règles de validation) ; concerne la structure du document. Exemple : conformité à un modèle spécifié

Brefs rappels sur XML (3/5)

- **Validation des documents XML**
  - Deux modes possibles de spécification de structure (modèles)
- **DTD *(Document Type Definition)***
  - Exprimé dans un formalisme spécifique
- **XML *Schema***
  - Exprimé en XML





Brefs rappels sur XML (4/5)

DTD

```
<!ELEMENT rue (#PCDATA)>
<!ELEMENT numero (#PCDATA)>
<!ELEMENT ville (#PCDATA)>
<!ELEMENT code (#PCDATA)>
<!ELEMENT lieu(rue, numero)>
<!ELEMENT adresse(lieu, ville, code)>
```

PCDATA=Parsed Character Data

Document XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<adresse>
   <lieu>
        <rue> avenue de l'Europe</rue>
        <numero>655</numero>
  </lieu>
   <ville>Montbonnot></ville>
   <code>38330</code>
</adresse>
```

Schema

```
<element name="rue" type="string"/>
<element name="numero" type="string"/>
<element name="ville" type="string"/>
<element name="code" type="string"/>
<element name="lieu">
    <complexType>
      <sequence>
          <element ref="rue"/>
          <element ref="numero"/>
      </sequence>
    </complexType>
</element>
<element name="adresse">
    <complexType>
      <sequence>
          <element ref="lieu"/>
          <element ref="ville"/>
          <element ref="code"/>
      </sequence>
    </complexType>
</element>
```

Brefs rappels sur XML (5/5)

Comment exploiter un document XML ?

- 2 APIs couramment utilisées par les applications
  - DOM : *Document Object Model*
    - APIs pour des objets de type document définis par une structure XML ; permet de naviguer dans le document (traversées d'arbre), d'accèder à des parties, etc;
  - SAX : *Simple API for XML*
    - Interface plus simple, même usage

- Ces interfaces sont créées pour une application, à partir d'un document XML, par des outils spécialisés

Usages de XML dans le *middleware*

- **Base pour la construction d'ADLs**
  - **Exemples : ADL Fractal, propositions de standards en cours (ACME)**
- **Base pour les Services Web**
  - **Description de services**
  - **Description des protocoles**
- **Outils pour la manipulation de données XML**
  - **Exemples : outils en CORBA**
    - **Emballage-déballage de données XML dans *strings* IDL**

Forme simple de Service Web : RPC-XML

Description en XML d'un appel de procédure à distance
Le type des paramètres est spécifié dans le schéma XML

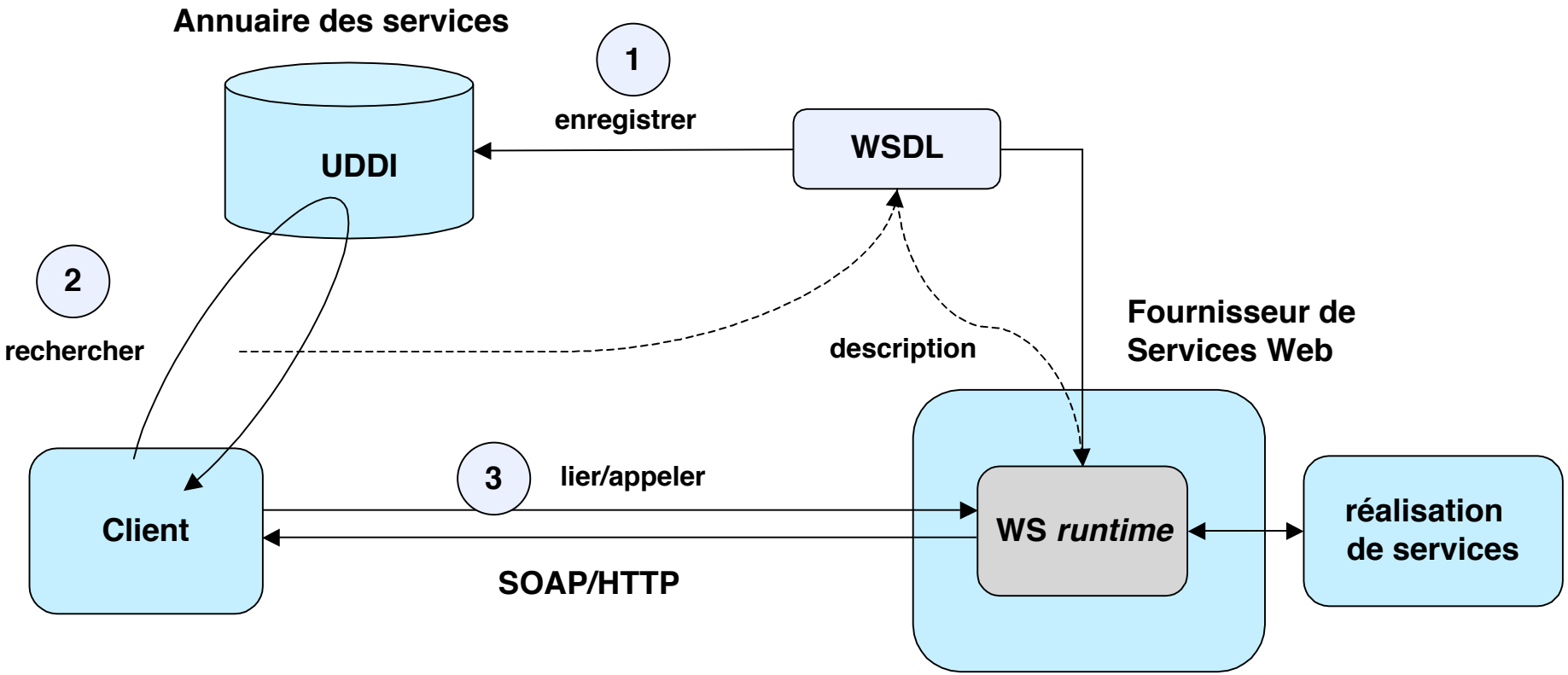
```
<methodCall>
      <methodName>meteo.temperature</methodName>
      <params>
            <param>
                  <value>38330</value>
            </param>
      </params>
</methodCall>
```

Description en XML du retour des résultats

```
<methodResponse>
      <params>
            <param>
                  <value>12</value>
            </param>
      </params>
</methodResponse>
```

Intérêt : indépendance par rapport aux plates-formes et par rapport au support de communication

Mise en œuvre d'un Service Web



Éléments des Services Web

- **Description d'un service**
  - WSDL : *Web Services Description Language*
  - Notation standard pour la description de l'interface d'un service
- **Accès à un service**
  - SOAP : *Simple Object Access Protocol*
  - Protocole Internet permettant la communication entre applications pour l'accès aux services Web
- **Annuaire des services**
  - UDDI : *Universal Description, Discovery and Integration*
  - Protocole pour l'enregistrement et la découverte de services

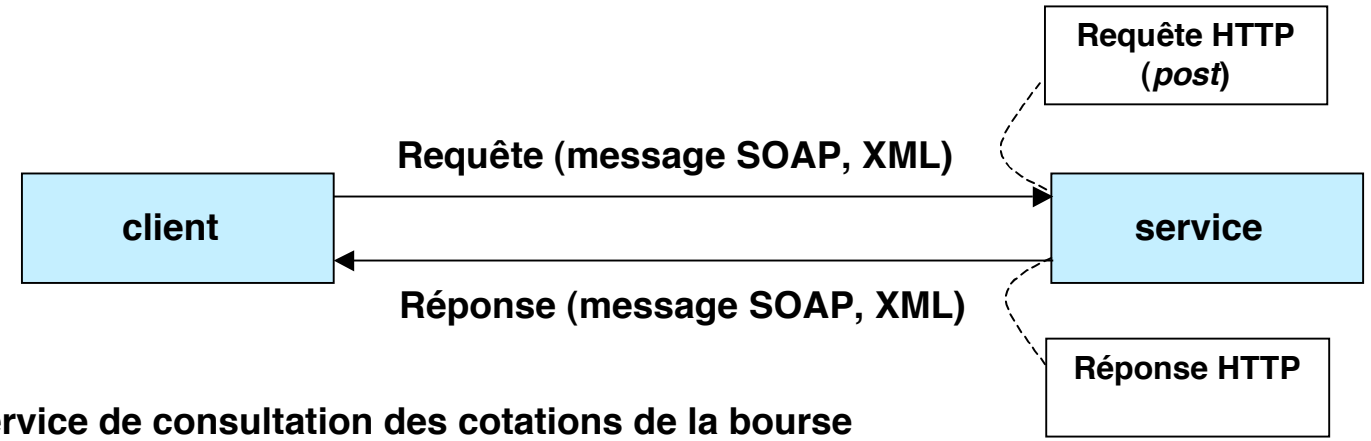
SOAP

Fonction

- Un mécanisme simple pour échanger des informations structurées et typées décrites en XML

Propriétés

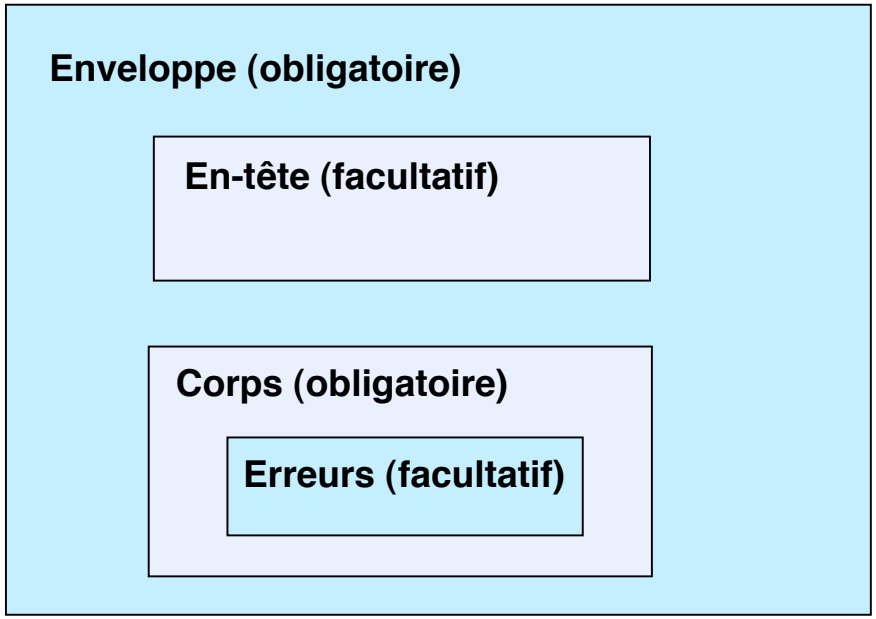
- Messages unidirectionnels (un couple de messages peut être utilisé pour réaliser un échange requête-réponse)
- SOAP peut être réalisé au-dessus d'un protocole quelconque au niveau application (HTTP, FTP, SMTP)



Exemple : service de consultation des cotations de la bourse

Éléments de SOAP

Un message SOAP



Enveloppe (*envelope*) : spécification des espaces de désignation, du codage des données, etc.

En-tête (*header*) : signature (sécurité), informations pour facturation, etc.

Corps (*body*) : élément principal du service : méthodes, paramètres

Erreurs(*fault*) : réactions en cas d'erreur (selon différentes causes)

Complément sur XML : espaces de noms

Les espaces de noms pemettent de définir des “domaines” séparés pour la définition des balises (les marques situées entre < et >)

Ainsi on peut réutiliser les mêmes balises si elles sont dans des espaces de noms différents.

Pour définir un espace de noms appelé *toto*, définir l’attribut *xmlns* :

xmlns:toto=http:// ... / Une URL où est défini cet espace

Pour utiliser les balises de cet espace dans le document XML :

```
<toto:truc>
   ...
</toto:truc>
```

où *truc* est une balise définie dans l'espace de noms *toto*
Le même nom *truc* peut être défini dans un espace de noms différent (avec une autre signification).

Un message SOAP sur HTTP (requête)

Prélude pour transport sur HTTP (spécifique à HTTP)

```
POST /StockQuote HTTP/1.1
Host:www.stockquoteserver.com
Content-Type: text/xml: charset="utf-8"
Content-Length: nnnn
SOAPAction: "SomeURI"
```

```
<SOAP-ENV:Envelope
      xmlns:SOAP-ENV= "http://schemas.xmlsoap.org/soap/envelope/"
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

      <SOAP-ENV:Body>
         <m:getLastTradePrice xmlns:m="SomeURI">
            <symbol>SomeCompany </symbol>
         </m:getLastTradePrice>
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Message proprement dit

Un message SOAP sur HTTP (réponse)

Prélude pour transport sur HTTP (spécifique à HTTP)

```
HTTP/1.1 200 OK
Content-Type: text/xml: charset="utf-8"
Content-Length: nnnn
```

```
<SOAP-ENV:Envelope
      xmlns:SOAP-ENV= "http://schemas.xmlsoap.org/soap/envelope/"
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

      <SOAP-ENV:Body>
         <m:getLastrTradePriceResponse xmlns:m="SomeURI">
            < price>34.5 </price>
         </m:getLastTradePriceResponse>
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Message proprement dit

Messages SOAP

- **Un message SOAP est un document XML**
  - Dans l'exemple, on a trouvé
    - Un espace de noms SOAP (SOAP-ENV)
    - Un espace de noms propre à l'application (m)
    - Des balises propres à chaque espace
- **Conséquences**
  - Auto-descriptif (modèle en DTD ou Schema XML)
    - Exemple

      Schema XML : <element name="price" type="float">
      Dans message : <price>6.3</price>
  - Directement lisible (éventuellement utile pour mise au point)
  - L'analyse des messages peut être complexe donc coûteuse

SOAP : encodage/décodage des données

Un message SOAP contient des données, dont il faut pouvoir déterminer la valeur

Encodage : donnée vers représentation XML

Décodage : représentation XML vers donnée

Encodage et décodage sont définis de diverses façons (en général via schéma XML)

Schéma	<element name="price" type="float/>
Valeur	<price>23.5</price>

Schéma	<element name="color"> <simpleType base="xsd:string"> <enumeration value ="Green"/> <enumeration value ="Red"/> <enumeration value ="Blue"/> </simpleType> </element>
Valeur	<color>Red</color>

WSDL

- **Définition**
  - Formalisme pour la description d'un service donnée par le fournisseur
    - Format des messages échangés (SOAP)
    - Relations entre les messages (requête-réponse)
- **Propriétés**
  - Expression en XML
  - Définition d'interface (cf IDL)
    - Types, messages, opérations, portes
  - Définition des points d'entrée (*endpoints*)
    - Liaisons (protocoles), port, service

Éléments de WSDL

<definitions>	**fonctions disponibles, espaces de désignation utilisés dans le reste de la description**
<types>	**définition des types de données utilisés**
<message>	**description des paramètres d'appel et de retour**
<portType>	**description des opérations (interface du service) (utilise les *messages*)**
<binding>	**description du mode de transport des *messages* lien entre protocole et *portType***
<service>	**localisation du service fourni (souvent une URL) ensemble de *ports* qui référencent des *bindings***

Exemple de définition en WSDL (1)

Définition de schéma (par ex. dans http://example.com/stockquote/stockquote.xsd)

```
<?xml version="1.0"?>
<schema
   targetNameSpace="http://example.com/stockquote/schemas"
      xmlns=http://www.w3.org/2000/10/XMLSChema>
   <element name="TradePriceRequest">
      <complexType>
         <all><element name="symbol" type="string"/></all>
      </complexType>
   </element>
   <element name="TradePrice">
       <complexType>
         <all><element name="price" type="float"/></all>
      </complexType>
    </element>
</schema>
```

Exemple de définition en WSDL (2)

Définition en WDSL (par ex. dans http://example.com/stockquote/stockquote.wsdl)

```
<?xml version="1.0"?>
<definitions name = … targetNameSpace=… >
   <import namespace= … location= … />
      <message name ="GetLastTradePriceInput">
         <part name="body" element=xsdl:TradePriceRequest"/>
      < /message>
      <message name ="GetLastTradepriceOutput">
         <part name="body" element= xsdl:TradePrice"/>
      < /message>
      <portType name="StockQuotePortType">
        <operation name="GetLastTradePrice">
          <input message="tns:GetLastTradePriceInput"/>
          <output message="tns:GetLastTradePriceOutput"/>
        </operation>
      </portType>
…
```

Exemple de définition en WSDL (3)

Définition en WDSL (par ex. dans http://example.com/stockquote/stockquote.wsdl) - suite

```
...
<binding name = "StockQuoteSoapBinding" type="defs:StockQuotePortType">
        <soap:binding style ="document"
    transport=http://schemas.xmlsoap.org/soap/http/>
          <operation name="GetLastTradePrice">
            <soap:operation
                soapAction=http://example.com/GetLastTradePrice/>
            <input>
                soap:body use="literal"/>
            </input>
            <output>
                soap:body use="literal"/>
            </output>
          </operation
</binding
...
```

"literal" par opposition à "encoded"

Exemple de définition en WSDL (4)

Définition en WDSL (par ex. dans http://example.com/stockquote/stockquote.wsdl) - fin

```
...
  <service name= "StockQuoteService">
      <documentation>First example of simple service </documentation>
      <port
        name="StockQuotePort"
        binding="tns:StockQuoteSoapBinding">
          <soap:address location="http://example.com/stockquote"/>
      </port>
  </service>
</definitions>
```

UDDI (Universal Description, Discovery and Integration)

Objectif : fournir un annuaire (ou référentiel) pour la description de services web.

Fonctions

Pages blanches : informations sur les fournisseurs de services (par nom)

Pages jaunes : informations sur les fournisseurs de services (par catégorie)

Pages vertes (spécificité de UDDI) : définition des services fournis, en WSDL

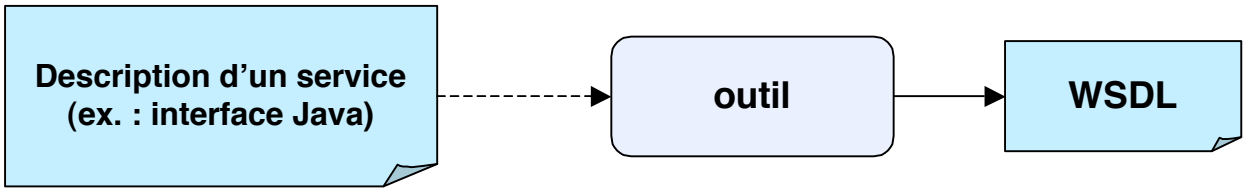
Les services du référentiel sont eux-mêmes des services web

Outils pour Services Web

Des outils pour les services Web commencent à être disponibles

Génération automatique des appels du client

Génération automatique de descriptions WSDL



Conclusion sur Services Web

- **Point fort : interopérabilité d'applications hétérogènes**
  - Mais beaucoup de lacunes fonctionnelles par rapport au *middleware* à composants
    - Pas de modèle de déploiement
      - Donc difficile de diffuser des composants
    - Pas d'environnement standard d'exécution
      - Donc pas de portabilité
    - Pas de projection sur langages de programmation
    - Aspects non-fonctionnels encore peu développés (travaux en cours)
      - Sécurité
      - Qualité de service
      - Transactions
- **Place des Services Web**
  - Le *middleware* "traditionnel" restera pour les applications fortement intégrées (environnement bien maîtrisé)
  - Les services Web serviront à assembler entre elles un ensemble de telles applications
  - C'est une technologie encore jeune, qui doit être consolidée