V – Middleware et Internet

- Internet, le Web et systèmes distribués ouverts
- XML
- Architecture de service Web
  - SOAP
  - WSDL
  - UDDI
- Conclusion

1

Internet

- Révolution de l'informatique grand public
  - Système d'information à grande échelle
  - Système ouvert

2

Le Web

- A l'origine
  - Centré sur l'échange de documents
- Architecture du Web
  - Format HTML
  - Protocole HTTP
  - Caches hiérarchiques et coopératifs

3

Consortium W3C

- Standardisation
- www.w3.org
- Laboratoires de recherche
  - INRIA, MIT, Université de Keio
  - …
- 400 partenaires industriels
  - Compaq, IBM, Microsoft, Oracle, Xerox, …

4

Evolution des serveurs Web

- Serveurs d'applications
  - Couplage avec des SGBD
  - Qualité
    - Transactions
    - Sécurité
  - Technologies associées
    - Java/JavaScript, Cookies, …

5

Des serveurs Web aux services Web

- Nouvelles applications du Web
  - Commerce électronique
    - Services de réservation, météo, etc.
    - Bourse en ligne
    - B2B, B2C, …
  - Communautés Web
    - Napster, …

6

Le Web et systèmes distribués

- Spécificités de l'architecture Web
  - Echelle
  - Latence
  - Système ouvert
- Exigences
  - Intégration de données hétérogènes
  - Composition de systèmes autonomes

7

Un début de solution

- L'architecture de service Web
  - XML pour l'intégration de données hétérogènes
  - Noyau de middleware pour définir et accéder aux services Web

8

XML

- Motivation
- Principes
- Langage de marquage XML
- Typage des documents
- Fragmentation des documents
- Transformation de documents

9

Motivation

- Besoin d'intégrer des données hétérogènes
  - Bases de données :
    - relationnels,
    - objets
  - Données multimédias :
    - Texte
    - Image
    - Vidéo
    - Audio

10

Hétérogénéité des données

- Niveau structurel et sémantique
  - Structurel
    - "plate": postscript
    - Semi-structuré : SGML/HTML
  - Sémantique
    - Signification d'une entité (nom d'attribut) fonction de la source de donnée
      - Ex : nom

11

Principes de XML

- De HTML à XML
  - HTML
    - HyperText Markup Language
  - XHTML 1.0
    - Recommandation W3C
    - HTML devient une application de XML

12

Un document HTML

```
<HTML>
    <HEAD><TITLE> Officiel du spectacle </TITLE></HEAD>
    <BODY BGCOLOR="blue">
    <H1> Officiel du spectacle </H1>
    <H2> Cinemas </H2>
    <UL>
          <LI><B> Multiplex </B>, Porte d'Italie, Paris</LI>
    </UL>
    <H2> Films </H2>
    <UL>
          <LI> <A="URL"><B>Ocean's eleven</B></A>
    </UL>
    </BODY>
</HTML>
```

13

Structure, contenu et présentation

- Document HTML mélange contenu et présentation
  - Ex : balise <B> introduit nom de salle et présentation en caractères gras
- Document HTML ne structure pas l'information
  - Ex : pas de distinction entre noms de cinémas et noms de théatres

14

Distinguer structure, contenu et présentation

- Feuilles de style pour séparer le contenu de sa présentation
- XML pour définir des balises spécifiques aux applications

15

Langage de marquage XML (I)

- XML
  - eXtensible Markup Language
- Standard
  - Recommandation du W3C
- Objet
  - Structuration de documents
    - Héritage de SGML
  - Définition de documents Web
    - Généralisation de HTML

16

Langage de marquage XML (II)

- Pourquoi ?
  - Faciliter
    - Echange de données sur le Web
    - Intégration dápplications Web
    - Interrogation du Web
- Historique
  - 93 : Adaptation des techniques SGML (standard pour documents structurés) au Web
  - 96 : Working group du W3C
  - 98 : Recommandation V 1.0 du langage

17

Un modèle de documents structurés

- XML est compatible avec SGML
- Edition simple de documents XML
  - Editeur de texte standard suffisant
- Structure de document peut être prédéfinie par une grammaire et analysée par un parseur
  - DTD
- Séparation contenu et présentation
  - Feuille de style XSL

18

Une syntaxe pour le transfert de données

- Encodage des caractères avec ASCII et UNICODE/ISO
- Encodage des données avec XML
  - Représentation de données de structure irrégulière, implicite et partielle (données semi-structurées)

19

Balisage structurel

- Principe clé de SGML
- Séparer la structure logique de la présentation du document
- Le gain comparé à HTML
  - Indépendance des outils de navigation (browser) et des outils de gestion de données (SGBD)
  - Différentes présentations pour un même document
  - Indexation et interrogation structurelle
  - Nouvelles techniques de gestion de caches plus efficaces

20

Un document XML (I)

- **Fichier officiel.xml**

```
<officiel>
    <cinéma>
        <nom> Multiplex </nom>
        <adresse>
            <ville> Paris </ville>
            <rue> Place d'Italie </rue>
        </adresse>
        <séance heure='14:00' ref_film = '&1'/>
        <séance heure='20:00' ref_film = '&2'/>
    </cinema>
    <film film_id = '&1' résumé = '&50' />
        <titre> Ocean'seleven </tittre> <année> 2002 </année>
    </film>
</officiel>
```

21

Un document XML (II)

- Racine de document
- Définition structurée, imbriquée d'éléments
- Références à des éléments du document
  - Pas de typage des références
- Définition d'identifiants (film_id) pour les éléments référencés

22

DTD pour le typage

- Document Type Definition
- Déclaration de la structure d'un document XML
- Standardisation de DTDs

23

Déclaration du type de document

- **Fichier officiel.ltd**

```
<!ELEMENT Officiel (#PCDATA | cinéma | film)*>
<!ELEMENT cinéma (nom, adresse, (séance)*)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT adresse (ville, rue)>
<!ELEMENT séance EMPTY>
<!ATTLIST séance heure NMTOKEN #REQUIRED
          ref_film IDREF #REQUIRED>
<ELEMENT film (titre, année)>
<!ATTLIST film film_id ID #REQUIRED résumé IDREFS #IMPLIED>
<!ELEMENT titre (#PCDATA)>
….
```

24

Déclaration d'éléments

- **Syntaxe** : <!ELEMENT nom Modèle-de-Contenu>
  - MC : *expression régulière* sur alphabet des noms d'éléments N
  - MC *EMPTY* : élément vide
  - MC *ANY* : toute combinaison d'éléments
  - MC *#PCDATA* : texte

25

Expression régulière

- Chaque nom d'élément
- Cloture : (e)*
- Répétition : (e)+
- Option : (e)?
- Séquence : (e1, e2)
- Alternative : (e1|e2)

26

Déclaration des attributs (I)

- **Syntaxe** : <!ATTLIST élément nom *type mode* [défaut]>
- *type* (types d'attributs)
  - *CDATA* : chaîne de caractères
  - Enumerated : séquence de valeurs alternatives séparées par |
  - *ID, IDREF, IDREFS* : Identification et références
  - *ENTITY/ENTITIES* : nom d'une entité non analysée, déclarée ailleurs
  - *NMTOKEN/NMTOKENS* : chaîne de caractères sans espace
  - *NOTATION* : une ou plusieurs notations séparées par |

27

Déclaration des attributs (II)

- *mode* (modes d'attributs)
  - *#REQUIRED* : définition obligatoire de la valeur
  - *#IMPLIED* : définition optionnelle de la valeur
  - *#FIXED* : valeur constante

28

Exemple d'attributs

```
<!ATTLIST séance
   heure NMTOKEN #REQUIRED
   ref_film IDREF #REQUIRED>
<!ATTLIST film
   film_id ID #REQUIRED
   résumé IDREFS #IMPLIED
   langue (AN|FR|AL|GR|RU) #IMPLIED>
<!ATTLIST adresse
   ville CDATA #IMPLIED 'Paris'>
```

29

Utilisation de DTD

- Type du document donné dans la définition
  - Définition locale

```
<!DOCTYPE Officiel [
<!ELEMENT Officiel (#PCDATA | cinéma | film)*>
…]>
```

  - Définition externe

```
<!DOCTYPE Officiel SYSTEM "officiel.dtd">
```

30

Entités générales et paramètres

- Entité générale
  - Déclaration dans DTD et utilisation dans DTD et document
- Entité paramètre
  - Déclaration et utilisation dans DTD

```
<!DOCTYPE Officiel [
<!ELEMENT officiel(année)>
<!ENTITY % text 'PCDATA'>
<!ELEMENT année (%text;)> ]>
<officiel> <année> 2002 </année></officiel>
```

31

Entités externes

- Définition du document par un ensemble de sous-documents
- Réutilisation de DTDs et de déclarations
- Références à des données non-XML (*NOTATION*)

32

Adressage

- URL

  <!ENTITY % autre SYSTEM 'http://site.fr/ext.xml'>

33

Entités non-XML

- Déclaration du format pour entités non-XML
- Référence à une entité de type notation comme valeur d'attribut

```
<!DOCTYPE exemple [
<!NOTATION gif SYSTEM '/usr/local/bin/xv'>
<!ENTITY photo SYSTEM './photo.gif'NDATA gif >
<!ELEMENT person EMPTY>
<!ATTLIST person photoId NOTATION (gif) #IMPLIED>
]>
<person photoId = 'photo'>
```

34

Nommage

- Un espace de noms XML est une collection de noms d'éléments ou de noms d'attributs (identifiés par URI)
- Permet d'éviter les conflits
  - Ex : cas d'utilisation conjointe de plusieurs DTD

35

Définition d'un espace de noms

```
<?xml version='1.0'?>
<film xmlns:fi = 'http://www.sitefilms.fr/films.dtd>
    <acteur>
        <nom> Brad Pitt </nom>
    </acteur>
```

- Nom qualifié : fi:film, fi:acteur, fi:nom

36

Documents XML bien formés et valides

- Document XML bien formé
  - Pas de DTD
  - Structure imbriquée (arborescence)
- Document XML valide
  - Existence de DTD
  - Respect du DTD
    - Grammaire, racine, attributs
  - Respect de l'intégrité référentielle
    - Valeurs des attributs ID distinctes
    - Validité de toutes les références

37

Intérêt de la validation des documents XML

- DTD définit une interface
  - Contrat entre le producteur et le consommateur
  - Qualité des données produites du point de vue du producteur
  - Séparation de la vérification syntaxique et de la logique de l'application du point de vue du consommateur

38

Fragments XML

- Xpath pour la sélection de fragments XML
  - DOM : Représentation arborescente du document XML
  - Référencer les noeuds d'un document XML
  - Utilisation
    - Création de clés et références dans un schéma XML
    - Création de lien entre documents/fragments par Xlink
    - Sélection de règles de transformation par XSL
  - Définition de règles pour parcours arborescence

39

Exemples

- Racine du document Officiel.xml :
  - Officiel.xml/
- Tous les fils de type film de la racine :
  - Officiel.xml/child::film
  - Officiel/film (sucre syntaxique)
- Tous les éléments de type film
  - Officiel.xml/descendants::film
  - Officiel.xml//film

40

XSL pour la transformation de documents

- Working draft W3C (Avril 99)
- Feuille de style XSL définie au moyen de deux langages
  - XSLT : transformation de la structure du document
  - Vocabulaire pour la spécification de la présentation

41

Utilisations

- Transformation document XML en document HTML
- Transformation document XML en un autre document XML
- Interrogation document XML
- …

42

Noyau middleware pour les services Web

- SOAP pour l'accès aux services
  - SOAP version 1.2
  - W3C Working draft, Déc. 01
- WSDL pour la déclaration des services
  - WSDL 1.1
  - W3C note, March 01
- UDDI pour découvrir les services
  - UDDI version 2.0
  - UDDI Open draft specification, June 01
  - Uddi.org

43

SOAP : Simple Object Access Protocol (I)

- http://www.w3.org/TR/2001/WD-soap12-part{0,1,2}-20011217/
- Document XML pour échange d'information typée et structurée dans un environnement distribué
- Echange de message unidirectionnel, sans état
- Possibilité de créer schémas d'interactions complexes au moyen protocole de transport sous-jacent (ex : HTTP) et/ou information de l'applicatif
  - Req-rep, req-muli-rep

44

SOAP : Simple Object Access Protocol (II)

- Ne spécifie pas
  - Sémantique des données de l'application
  - Routage des messages
  - Fiabilité de l'échange de données
  - Transfert par *firewall*

45

Définition

- Messages
- Modèle de traitement
- Echange de messages
- Traitement des défaillances
- Liaisons avec différents protocoles de transport

46

Message SOAP (I)

- Elément englobant: **Envelope**
- Deux sous-éléments:
  - Contenu spécifique à l'applicatif, indépendant de SOAP
  - **Header**
    - Collection de Header Blocks
    - Définition optionnelle,
    - Enrichit information sur le message
    - Utilisation par différents services, autre que le service cible (ex : services intermédiaires)
  - **Body**
    - Définition obligatoire
    - Contient principale information du message nécessaire au service cible pour traitement

47

Structure des messages

SOAP Envelope

SOAP header (optionnel) – Ex *Travel* :
- Header block: reservation data
- Header Block: passenger data

Utilisation possible par services intermédiaires et service final

SOAP Body (obligatoire) / – Ex *Travel* :
- Body sub-element: itinerary req. data
- Body sub-element: lodging req. data

48

Exemple (I)

```
<?xml version= ‘1.0’?>
<env: Enveloppe xmlns:env=“http://www.w3.org/2001/12/soap-enveloppe”>
<env: Header>
    <m: reservation xmlns’=http://travelcompany.example.org/rservation”
                    env:actor=“http://www.w3.org/2001/12/soap-enveloppe/actor/next
                    env:mustunderstand= “true”>
        <m:reference>uuid:…</reference>
        <m:dateAndTime> 2002-03-15T20:00-05:00</m:dateAndTime>
    </m:reservation>
    <n:passenger xmlns:p=http://travelcompany.example.org/employees
                    env:actor= “http://www.w3.org/2001/12/soap-enveloppe/actor/next
                    env:mustunderstand= “true”>
        <n:name> John Public</n:name>
    </n:passenger>
</env: Header>
```

49

Exemple (II)

```
<env: Body>
    <p: itinerary xmlns’=http://travelcompany.example.org/reservation/travel”>
        <p:departure>
                <p:departing>New York</p:departing>
                <p:arriving> Los Angeles</p:arriving>
                <p:departureDate>2002-04-15</p:departureDate>
                <p:departureTime>mid morning</p:departureTime>
                <p:seatPreference/>
        </p:departure>
        <p:return>…</p:return>
    </p:itinerary>
    <q:lodging xmlns:p=http://travelcompany.example.org/reservation/hotels”>
        <q:preference> none</q:preference>
    </q:lodging>
</env: Body>
</env:Enveloppe>
```

50

Modèle de traitement

- Décrit action d’un processus SOAP à la réception d’un message SOAP
  - Analyse des parties spécifiques à SOAP
    - C-à-d. Eléments de l’espace de noms “env” de SOAP
    - *Envelope*
      - *Header*
      - *Body*

51

Traitement du contenu du *Header* (I)

```
<?xml version= ‘1.0’?>
<env: Enveloppe xmlns:env=“http://www.w3.org/2001/12/soap-enveloppe”>
<env: Header>
    <p:oneBlock xmlns:p=http://example.com env:actor=“http://example.com/Log”>
    ….
    </p:oneBlock>
    <q:anotherBlock xmlns:q=http://example.com
            env:actor=“http://www.w3.org/2001/12/soap-enveloppe/actor/next>
    ….
    </q:anotherBlock>
    <r:aThirdBlock xmlns:p=http://example.com>
    ….
    </r:aThirdBlock>
</env: Header>
<env: Body> … </env:Body>
</env: Enveloppe>
```

52

Traitement du contenu du *Header* (II)

- Rôle du processus pour le traitement des blocs du *header*, défini par attribut *actor* (anyURI)
- Trois rôles standardisés
  - *none* : contenu ne doit pas être traité
  - *next* : contenu doit être traité par tout processus qui reçoit le message
  - *anonymous* : contenu traité par processus cible du message, rôle par défaut dans message

53

Traitement du contenu du *Body*

- Doit être traité par le processus cible/destinataire

54

Echange de messages

- SOAP définit messages sous la forme de documents XML
- Possibilité de composer un ensemble de messages pour réaliser des schémas d'interactions complexes
  - Messages de requête et de réponse
  - Utilisation protocole de transport sous-jacent pour réalisation effective de la synchronisation

55

Un exemple de msg de réponse (I)

```
<?xml version= '1.0'?>
<env: Enveloppe xmlns:env="http://www.w3.org/2001/12/soap-enveloppe">
<env: Header>
    <m: reservation xmlns'=http://travelcompany.example.org/reservation"
                    env:actor="http://www.w3.org/2001/12/soap-enveloppe/actor/next
                    env:mustunderstand= "true">
        <m:reference>uuid:…</reference>
        <m:dateAndTime> 2002-03-15T20:00-05:00</m:dateAndTime>
    </m:reservation>
    <n:passenger xmlns:p=http://travelcompany.example.org/employees
                    env:actor= "http://www.w3.org/2001/12/soap-enveloppe/actor/next
                    env:mustunderstand= "true">
        <n:name> John Public</n:name>
    </n:passenger>
</env: Header>
```

56

Un exemple de msg de réponse (II)

```
<env: Body>
    <p: itinerary xmlns'=http://travelcompany.example.org/reservation/travel">
        <p:airportChoices>
            LAX ORANGE
        </p:airportChoices>
    </p:itinerary>
</env: Body>
</env:Enveloppe>
```

57

Réalisation d'une conversation Message de choix de l'utilisateur

```
<?xml version= '1.0'?>
<env: Enveloppe xmlns:env="http://www.w3.org/2001/12/soap-enveloppe">
    <env: Header>
        Identique
    </env: Header>

<env: Body>
    <p: itinerary xmlns'=http://travelcompany.example.org/reservation/travel">
        <p:departure>
            JFK
        </p:departure>
        <p: return>
            LAX
        </p: return>
    </p:itinerary>
</env: Body>
</env:Enveloppe>
```

58

De SOAP au RPC

- Définition d'une représentation uniforme pour appel et retour RPC
- Information d'un RPC
  - Serveur
  - Nom procédure/méthode
  - Paramètres

59

Exemple – Requête (I)

```
<?xml version= '1.0'?>
<env: Enveloppe xmlns:env="http://www.w3.org/2001/12/soap-enveloppe">
<env: Header>
INFORMATION SUR LE TRAITEMENT (INFO D'ENVIRONNEMENT)
    <m: reservation xmlns'=http://travelcompany.example.org/rservation"
            env:actor="http://www.w3.org/2001/12/soap-enveloppe/actor/next
            env:mustunderstand= "true">
        <m:reference>uuid:…</reference>
        <m:dateAndTime> 2002-03-15T20:00-05:00</m:dateAndTime>
    </m:reservation>
    <t: transaction xmlns:p=http://thirdparty.example.org/transaction"
            env:encodingstyle= "http://example.com/encoding"
            env:mustunderstand= "true">
                5
    </t:transaction>
</env: Header>
```

60

Exemple – Requête (II)

```
<env: Body>
    PROCEDURE (nom, codage des params, params effectifs)
    <m: reserveAndCharge
                env: encodingStyle: "http://www.w3.org/2001/12/soap-encoding"
                        xmlns'=http://travelcompany.example.org/">
        PARAMETRES EFFECTIFS
        <n:name xmlns:n=http://travelcompany.example.org/employee/">
                John Public
        </n:name>
        <o: creditCard xmlns:o="http:// …">
                <o: number> … </o:number>
                <o: expiration> … </o: expiration>
        </o: creditCard>
    </m: reserveAndCharge>
</env: Body>
</env:Enveloppe>
```


Exemple – Résultat

```
<?xml version= '1.0'?>
<env: Enveloppe xmlns:env="http://www.w3.org/2001/12/soap-enveloppe">
<env: Header>
    <t: transaction xmlns:p=http://thirdparty.example.org/transaction"
                env:encodingstyle= "http://example.com/encoding"
                env:mustunderstand= "true">
                        5
    </t:transaction>
</env: Header>
<env: Body>
    <m: reserveAndChargeResponse
                env: encodingStyle: "http://www.w3.org/2001/12/soap-encoding"
                        xmlns'=http://travelcompany.example.org/">
        <m: confirmation>
                ….
        </m: confirmation>
    </m: reserveAndChargeResponse>
</env: Body>
</env:Enveloppe>
```


Traitement des défaillances

- Définition de l'élément *fault* pour rapporter occurrence de fautes

Exemple

```
<?xml version= '1.0'?>
<env: Enveloppe xmlns:env=http://www.w3.org/2001/12/soap-enveloppe
        xmlns:f="http://www.w3.org/2001/12/soap-faults">
<env: Body>
    <env: fault>
        <faultcode> env: receiver </faultcode>
        <faultstring> Processing Error </faultstring>
        <detail>
            <e: myfaultdetails xmlns:e = "…">
                <message> Name does not match card number </message>
                <errorcode> 999 </errorcode>
            </e: myfaultdetails>
        </detail>
    </env: fault>
</env: Body>
</env:Enveloppe>
```

64

Liaison avec différents protocoles de transports

- SOAP ne prescrit pas protocole de transport sous-jacent
- Spécification de liaisons pour différents protocoles
- Exemples
  - HTTP (cas *standardisé*)
    - Utilisation msgs req/rep
    - Méthode HTTP POST pour véhiculer messages SOAP dans msgs de req.
  - Email

65

WSDL : Web Services Description Language

- http://www.w3.org/TR/2001/NOTE-wsdl-20010315
- Format de document XML pour décrire services
- N'est pas lié à format de messages, ni protocole réseau particuliers
- Liaisons définies pour
  - SOAP 1.1
  - HTTP GET/POST
  - MIME

66

Exemple (1)

```
<?xml version="1.0"?>
        <definitions name="StockQuote"

        targetNamespace="http://example.com/stockquote.wsdl"
                xmlns:tns="http://example.com/stockquote.wsdl"
                xmlns:xsd1="http://example.com/stockquote.xsd"
                xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                xmlns="http://schemas.xmlsoap.org/wsdl/">
```

67

Exemple (2)

```
<types>
   <schema targetNamespace="http://example.com/stockquote.xsd"
         xmlns="http://www.w3.org/2000/10/XMLSchema">
      <element name="TradePriceRequest">
        <complexType>
          <all>
            <element name="tickerSymbol" type="string"/>
          </all>
        </complexType>
      </element>
      <element name="TradePrice">
        <complexType>
          <all>
            <element name="price" type="float"/>
          </all>
        </complexType>
      </element>
   </schema>
</types>
```

68

Exemple (3)

```
<message name="GetLastTradePriceInput">
   <part name="body" element="xsd1:TradePriceRequest"/>
</message>

<message name="GetLastTradePriceOutput">
   <part name="body" element="xsd1:TradePrice"/>
</message>

<portType name="StockQuotePortType">
   <operation name="GetLastTradePrice">
     <input message="tns:GetLastTradePriceInput"/>
     <output message="tns:GetLastTradePriceOutput"/>
   </operation>
</portType>
```

69

Exemple (4)

```
<binding name="StockQuoteSoapBinding" type="tns: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>
```

70

Exemple (5)

```
 <service name="StockQuoteService">
    <documentation>My first service</documentation>
    <port name="StockQuotePort" binding="tns:StockQuoteBinding">
      <soap:address location="http://example.com/stockquote"/>
    </port>
  </service>

</definitions>
```

71

SOAP : un langage de définition d'interfaces

- Document XML pour la spécification des services

72

Structure de document WSDL

- Types
  - Définition types utilisés pour décrire échange de messages
- Message
  - Définition abstraite des données transmises
- portType
  - Ensemble d'opérations abstraites (msgs req/rep)
- Binding
  - Spécifie protocole concret et format de données pour opérations et messages
- Port
  - Spécifie adresse de liaison pour communication
- Service
  - Combinaison d'un ensemble de ports

73

Grammaire (1)

```
<wsdl: definitions name = "nmtoken"? targetNameSpace= "uri"?>
<wsdl: documentation … />?
<wsdl: types> ?
    <wsdl: documentation … />?
    <xsd: schema … />*
    < --  extensibility elements -- > *
</wsdl: types>
<wsdl: message name="nmtoken">*
    <wsdl: documentation … />?
    <part name = "nmtoken"element = "qname"? Type = "qname"?/> *
</wsdl:message>
```

74

Grammaire (2)

```
<wsdl: portType name = "nmtoken">*
    <wsdl: documentation … />?
    <wsdl: operation name="nmtoken"> *
        <wsdl: documentation … />?
        <wsdl: input name="nmtoken"? message= "qname"> ?
            <wsdl: documentation … />?
        </wsdl: input>
        <wsdl: output name="nmtoken"? message= "qname"> ?
            <wsdl: documentation … />?
        </wsdl: output>
        <wsdl: fault name="nmtoken"? message= "qname"> ?
            <wsdl: documentation … />?
        </wsdl: fault>
    </wsdl: operation>
</wsdl: portType>
```

75

Grammaire (3)

```
<wsdl: binding name = "nmtoken" type="qname">*
    <wsdl: documentation … />?
    < -- extensibility elements -- > *
    <wsdl: operation name="nmtoken"> *
        <wsdl: documentation … />?
        <wsdl: input>?
            <wsdl: documentation … />?
            < -- extensibility elements -- > *
        </wsdl: input>
        <wsdl: output>?
            <wsdl: documentation … />?
            < -- extensibility elements -- > *
        </wsdl: output>
        <wsdl: fault name="nmtoken"? > ?
            <wsdl: documentation … />?
            < -- extensibility elements -- > *
        </wsdl: fault>
    </wsdl: operation>
</wsdl: binding>
```

76

Grammaire (4)

```
<wsdl: service name = "nmtoken">*
    <wsdl: documentation … />?
    <wsdl: port name="nmtoken" binding = "qname"> *
        <wsdl: documentation … />?
        < -- extensibility elements -- > *
    </wsdl: port>
</wsdl: service>
< -- extensibility elements -- > *
</wsdl: definitions>
```

77

Liaison avec SOAP

- Spécification d'information spécifique
  - Liaison avec protocole SOAP 1.1
  - Adresse processus SOAP cible
  - Définition de headers et body

78

Exemple – RPC sur HTTP (1)

```
<?xml version="1.0"?>
        <definitions name="StockQuote"
                targetNamespace="http://example.com/stockquote.wsdl"
                xmlns:tns="http://example.com/stockquote.wsdl"
                xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
                xmlns:xsd1="http://example.com/stockquote.xsd"
                xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                xmlns="http://schemas.xmlsoap.org/wsdl/">

          <message name="GetTradePriceInput">
            <part name="tickerSymbol" element="xsd:string"/>
            <part name="time" element="xsd:timeInstant"/>
          </message>

          <message name="GetTradePriceOutput">
            <part name="result" type="xsd:float"/>
          </message>
```

79

Exemple (2)

```
<portType name="StockQuotePortType">
              <operation name="GetTradePrice">
                <input message="tns:GetTradePriceInput"/>
                <output message="tns:GetTradePriceOutput"/>
              </operation>
            </portType>

            <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
              <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
              <operation name="GetTradePrice">
                <soap:operation soapAction="http://example.com/GetTradePrice"/>
                <input>
                  <soap:body use="encoded" namespace="http://example.com/stockquote"
                          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                </input>
                <output>
                  <soap:body use="encoded" namespace="http://example.com/stockquote"
                          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                </output>
              </operation>>
            </binding>
```

80

Exemple (3)

```
<service name="StockQuoteService">
              <documentation>My first service</documentation>
              <port name="StockQuotePort" binding="tns:StockQuoteBinding">
                <soap:address location="http://example.com/stockquote"/>
              </port>
            </service>
          </definitions>
```

81

UDDI : Universal Description, Discovery and Integration

- Répertoires Web d'information sur organisation ou toute autre entité et leurs interfaces techniques
  - Trader pour le Web
  - Utilisation pour la publication et découverte de services Web

82

Conclusion : vers une architecture Web de systèmes distribués (I)

- Architecture de service Web doit évoluer
  - Composition de services
    - Spécification et réalisation
      - Comment composer et garantir la correction
      - Système de workflow
      - XLANG, WFSL
    - Sûreté de fonctionnement
      - Evolution des systèmes transactionnels

83

Conclusion : vers une architecture Web de systèmes distribués (II)

- Spécificités du Web requièrent évolution des services de base pour le développement de systèmes distribués
  - Echelle
  - Autonomie des services/sites
  - Ouverture
- Modèle client-serveur et P2P

84