Slide 1

Cours No 10 - Conclusion et perspectives

Gestion de contenus Web

La gestion de contenus Web est possible grâce à des technologies complémentaires pour le traitement de données et de connaissances.

Slide 2

Deux piliers :

- XML = modèles de données : transformation, échange, stockage, interrogation, import/export
- RDF = modèle de connaissances : description, intégration, raisonnement

Services pour la Gestion de Contenus Web

La gestion de contenus Web nécessite le rapprochement de différentes technologies complémentaires :

Bases de données :

Slide 3

- stockage, gestion et interrogation de données XML et de métadonnées RDF
- intégration de données
- transactions, cohérence

Recherche d'information :

- indexation de texte
- moteurs de recherche

Intelligence artificielle :

- représentation de connaissances
- classification de données
- extraction de métadonnées/connaissances

Entrepôts de données :

- datamining
- OLAP

Slide 4

Traitement de langues naturelles :

- marquage sémantique de textes non-structurés (XMLisation)

Interface utilisateurs :

- espace d'information partagé
- gestion de communautés virtuels
- personnalisation (profils utilisateurs)

- communication

Authorisation et Sécurité :

- utilisateurs et groupes
- sessions
- privilèges
- transactions sécurisées

Slide 5

Slide 6

Un nouveau type de ressources : Les Services Web

Ressources Web “Actives”

Slide 7

Ressource “dynamique” = géneration dynamique de pages HTML

Ressource “active” = déclenchement de tâches

HTML+HTTP+scripts :

- format d’échange : HTML
- paramètres de type texte
- interface informel
- utilisation manuel

Services Web

Slide 8

Formalisation de la notion de ressource active.

XML+SOAP+code :

- format d’échange : XML
- paramètres de type XML Schéma
- interface formel : WSDL
- utilisation automatique

Propriétés des Services Web

Slide 9

Indépendance

- du protocole : SOAP/WSDL
- des données : XML
- du code : binding
- de la localisation : Web

Un service peut être appelé de partout sur le Web.

SOAP : Principes

SOAP (Simple Object Access Protocol) est un protocole pour l'échange de messages structurés dans un environnement distribué, décentralisé et hétérogène (comme par exemple le Web).

Slide 10

Principes :

- indépendance du protocole sous-jacent (HTTP POST/GET, SMTP, RPC, ...)
- messages autodescriptifs : chaque message décrit
  - l'encodage des données (XML, MIME)
  - les types des données XML (XML Schema)
  - des directives de traitement du message à effectuer
- règles de sérialisation de données structurées
- traitement d'erreurs standardisé

Traitement d'un message SOAP

Slide 11

Chaque message à un émetteur et suit un chemin à travers une séquence de noeuds.

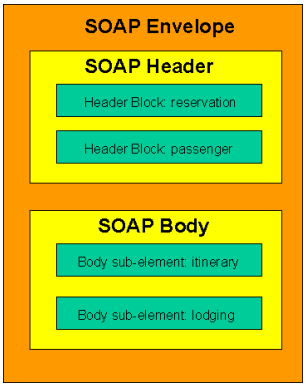
Chaque noeud sur le chemin doit traiter le message :

- identifier les fragments du message qui le concernent
- vérifier si toutes les fragments obligatoires sont supportés et les traiter
- si le noeud n'est pas le récepteur final, enlever les fragments du message qui le concernent et renvoyer le message au noeud suivant

Structure d'un message SOAP

Slide 12

- Head : fragments à traiter par les services intérmédiaires
- Body : fragment à traiter par le recepteur final



Message SOAP : Exemple

Slide 13

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2001/12/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
          env:actor="http://www.w3.org/2001/12/soap-envelope/actor/next"
           env:mustUnderstand="true">
    <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</reference>
    <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"
          env:actor="http://www.w3.org/2001/12/soap-envelope/actor/next"
           env:mustUnderstand="true">
   <n:name>John Q. Public</n:name>
  </n:passenger>
  <z:travelPolicy xmlns:z="http://mycompany.example.com/policies"
                  env:mustUnderstand="true">
   <z:class>economy</z:class>
   <z:fareBasis>non-refundable<z:fareBasis>
   <z:exceptions>none</z:exceptions>
  </z:travelPolicy>
 </env:Header>
```

Message SOAP : Exemple (bis)

Slide 14

```
 <env:Body>
  <p:itinerary xmlns:p="http://travelcompany.example.org/reservation/travel">
   <p:departure>
     <p:departing>New York</p:departing>
     <p:arriving>Los Angeles</p:arriving>
     <p:departureDate>2001-12-14</p:departureDate>
     <p:departureTime>late afternoon</p:departureTime>
     <p:seatPreference>aisle</p:seatPreference>
   </p:departure>
   <p:return>
     <p:departing>Los Angeles</p:departing>
     <p:arriving>New York</p:arriving>
     <p:departureDate>2001-12-20</p:departureDate>
     <p:departureTime>mid morning</p:departureTime>
     <p:seatPreference/>
   </p:return>
  </p:itinerary>
  <q:lodging xmlns:q="http://travelcompany.example.org/reservation/hotels">
   <q:preference>none</q:preference>
  </q:lodging>
 </env:Body>
</env:Envelope>
```

Composants d'une description WSDL

SOAP est un protocole pour l'échange de messages, mais ne décrit pas les services Web qui les traitent.

WSDL : Langage pour la description de services Web.

Slide 15

Description de l'interface du service :

- Types : description des données (de paramètres) utilisées dans les messages
- Messages : description structurée des données échangées
- Opérations : description abstraite des actions supportées par un service.
- Type de port : ensemble d'opérations abstraites supportées par un ou plusieurs services Web

- Binding : un protocole concret et une spécification de format de données pour un type de port particulier.

Description de l'implantation du service :

- Port : un point d'accès unique défini sous forme d'un binding et d'une adresse réseau
- Service Web : un ensemble de ports

Slide 16

Types d'opérations

Slide 17

Une opération est composée d'un message de type `input` et d'un message de type `output`. Les deux types de messages sont optionnelles et leur ordre désigne le type de l'opération :

- One-way : `input` sans `output`
- Request-response : `input` suivi de `output`
- Solicit-response : `output` suivi de `input`
- Notification : `output` sans `input`

Services = Ressource Web

Slide 18

Un service est une ressource Web :

- "N'importe qui n'importe où" peut publier un service.
- On a les mêmes besoins que pour les ressources Web passives : On veut *chercher, composer, décrire, ...* des services.

Exemple : Découverte et composition de Services

Je veux aller de Paris à Marseille le 1 Juillet et rester 3 jours dans un hôtel.

Slide 19

- Quels sont les services disponibles ?
- Comment choisir les services ?
- Comment les composer ?
- Est-ce qu'il est possible de les composer automatiquement/dynamiquement?

Standards et Recherche

Il existe un certain nombre de standards pour

Slide 20

- définir un service : WSDL
- appeler un service : SOAP
- décrire un service : UDDI

Mais beaucoup de questions restent encore ouvertes : workflow, transactions distribués, qualité de service, ...

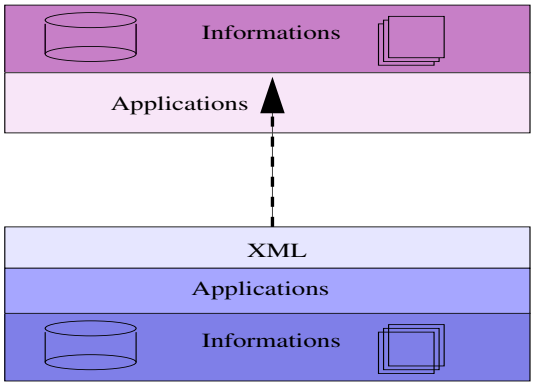
Slide 21

Services et Données

Le Web Passif

XML facilite l'échange de donnée entre applications, mais les ressources restent passives!

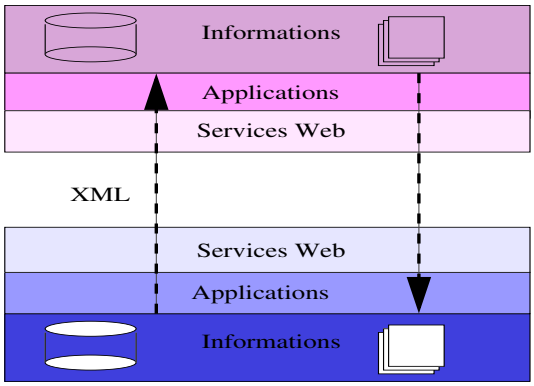
Slide 22



Le Web Actif

Slide 23

Les services Web donnent la possibilité de manipuler des données gérées localement par des applications indépendantes.



Services et Données

Slide 24

Les services Web permettent une autre conception de la gestion d'informations (de données) sur le Web.

Architecture pair-à-pair (P2P) : plus de distinction entre clients et serveurs de données

Avantages :

- Performance : pas de serveur centralisé
- Autonomie : chaque pair a le contrôle sur ses données
- Passage à l'échelle : distribution de la charge; replication des données
- Dynamicité : système ouvert (gestion dynamique des pairs)

Inconvénients :

- Coût de communication
- Cohérence des données
- Qualité des données difficile à maîtriser

Slide 25

Slide 26

ActiveXML

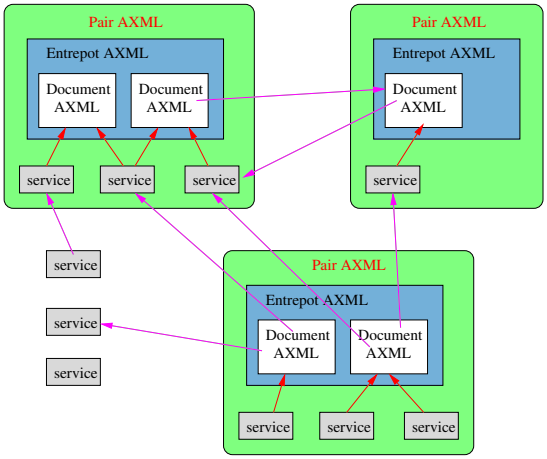
ActiveXML

Slide 27

- Système distribué pour le partage de données et de services Web.
- En cours de développement au projet Gémo, INRIA-Futurs
- Principe : Document XML intensionnel

Environnement ActiveXML

Slide 28



Documents Intensionnel

- Service Web : *foot.com*
- Opération : *getMatch*
- Paramètres : code de l'équipe, code du tournoi

Slide 29

```
<worldcup year="2002">
    <sc>foot.com/getMatch("FRA","MONDIAL02")</sc>
</worldcup>
```

Evaluation

Slide 30

```
<worldcup year="2002">
 <sc>foot.com/getMatch("FRA", "MONDIAL02")</sc>
 <match id="1" location="Séoul" date="31 May">
  <equipe id="FRA" score="0"/>
  <equipe id="SEN" score="1"/>
 </match>
 <match id="18" location="Busan" date="06 Jun">
  <equipe id="FRA" score="0"/>
  <equipe id="URU" score="0"/>
 </match>
</worldcup>
```

Service ActiveXML

Slide 31

Requête XML (XOQL) sur des documents ActiveXML :

```
let Resultats($pays) be
   for $match in document("worldcup.xml")/worldcup/match,
       $equipe1 in $match/equipe,
       $equipe2 in $match/equipe
 where $equipe1/@id=$pays
   and not($equipe2/@id=$pays)
return if ($equipe1/@score < $equipe2/@score)
       then concat("Perdu contre ", string($equipe2/@id))
       else if ($equipe1/@score > $equipe2/@score)
            then concat("Gagné contre ", string($equipe2/@id))
            else concat("Match nul contre ", string($equipe2/@id))
```

Appel de service :

```
http://www.foot.org/Resultats("FRA")
```

Résultat :

```
''Perdu contre SEN'', ''Match nul contre URU''
```

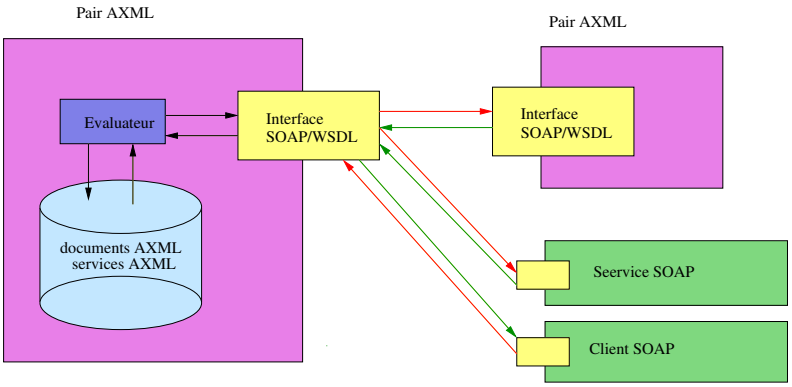
Paramètres d'un appel de service

Slide 32

- Durée de validité d'un résultat
- Fréquence de rafraichissement d'un appel de service
- Mode d'évaluation :
  - paresseux
  - immédiat
- Permet de s'adapter à différents style d'intégration de données
  - Mediateur (mode=lazy, valid=0)
  - Mediateur avec cache (mode=lazy, valid > 0)
  - Entrepôt (mode=immediate, valid > frequency)

Comment ça marche?

Slide 33



Typage de documents Active XML

Slide 34

Types de noeuds d'un document Active XML:

- éléments, attributs, ...
- appels de service:
  - nom + types de paramètres et type du résultat

Extension de XML Schéma avec un nouveau type *ServiceCall* : XMLSchema$_{int}$

Validation

Slide 35

Validation :

- Les paramètres d'un service peuvent être validés avant l'appel du service.
- Il est possible de contrôler la présence des appels de services dans un document.

Validation et sécurité

Slide 36

La validation permet de restreindre les appels de services pouvant être utilisés dans un document.

- Attaque client → serveur

  ```
  <sc>qod.com/QuoteOfDay(<sc>buy.com/BuyCar("BMW Z3")</sc>)</sc>
  ```

- Attaque serveur → client (cheval de Troie)

  ```
  <sc>buy.com/BuyCar("BMW Z3")</sc>
  ```

Avec validation :

- Appels de services autorisés dans un paramètre d'un appel de service (pas d'attaque client vers serveur)
- Vérification du typage du résultat d'un appel de service (pas d'attaque serveur vers client)

Adaptation par matérialisation

On peut adapter automatiquement le degré de matérialisation d'un document :

Slide 37

- Etant donnée un document et un schéma, chercher une séquence de matérialisations (remplacements d'un appel de service par le type de son résultat) telle que le document est une instance du schéma.

Applications :

- Sécurité : le pair récepteur ne veut pas effectuer certains appels de service
- Performance : diminuer la taille de données échangées (taille(appel) $<$ taille(résultat))

Slide 38

Merci pour votre attention!