Architectures SOA et Services Web

Youssef BEN HALIMA
Youssef.benhalima@gmail.com

Problématique

- Concevoir et développer un système de gestion d'une nouvelle banque.
- Le système doit être :
  - Complet (toutes les fonctionnalités possibles).
  - Modulaire (décomposé en modules).
  - Architecture Centralisé (centraliser les données et les traitements) ne pas oublier la sauvegarde.
  - Système Évolutif (possibilité de maintenir le système indépendamment de la société qui l'a développé).
  - Multi langage (le système doit fonctionner avec des interfaces développés avec des technologies différentes).
  - Système Extensible en ligne(le système doit être étendu sans arrêter le fonctionnement).
  - Métier en constante évolution : changement dans les procédures métier et mise à jour instantanée sans arrêter le fonctionnement.

Nouveau métier en informatique

- En génie informatique, on appelle **architecte** la personne chargée de l'analyse technique nécessaire à la conception du *diagramme d'architecture*, c'est-à-dire le plan de construction d'un logiciel, d'un réseau, d'une base de données, etc. Un architecte peut travailler avec les développeurs du système actuel ou d'autres architectes.
- L'architecte logiciel est une spécialisation autour du logiciel.
- Salaire entre 60 K euros et 100 K euros par an

Métier d’architecte en informatique : Spécialisations

- Architecte d'entreprise : L'architecte d'entreprise est spécialisé dans la conception des sites informatiques des entreprises. Les flux d'informations entre les acteurs de l'entreprise (clients, fournisseurs, collaborateurs), et le transit des informations à travers le système informatique.
- Architecte applicatif : L'architecte applicatif est spécialisé dans la conception des logiciels applicatifs, les flux d'informations entre les pièces du logiciel, en particulier l'interface utilisateur, les pièces métier, et les pièces qui assurent la communication avec les autres logiciels.
- Architecte système : L'architecte système est spécialisé dans la conception des systèmes d'exploitation et leurs composants : les connexions entre les logiciels applicatifs, les différents composants du système d'exploitation, et les composants complémentaires (drivers).
- Architecte réseau : L'architecte réseau est spécialisé dans la conception de différents types de réseaux informatiques
- Architecte matériel : L'architecte matériel est spécialisé dans les composants matériels des systèmes d'informations: le dimensionnement et les interconnexions entre les composants matériels, ordinateurs, serveurs, appareils réseaux (routeurs, ...), imprimantes, disques durs, mémoire, processeur.

Penser aux SOA comme solution

- Comment orienter le projet → argumenter le choix de la solution
- Comment analyser le système → procéder par une analyse métier acteur par acteur
- Comment concevoir le système → concevoir une architecture orientée services
- Comment réaliser le système → choix du Framework de développement (entre technologie J2EE et .NET, que choisir)
  - Compétences équipe
  - Possibilités financières
  - Besoin de support externe ?

Plan

1. Architectures SOA
   - Définitions
   - Caractéristiques de SOA
   - Architectures XSOA
2. Services Web
   - Principes
   - XML (Rappel et Notions)
   - SOAP
   - WSDL
   - UDDI
   - Exemples
3. Cloud Computing
   - Grid computing
   - Différents types de Cloud
     - IAAS
     - PAAS
     - SAAS

PARTIE 1 :
Architectures orientées services SOA

Définitions

Définitions (Architecture)

- **Architecture**, nom féminin : Sens Art de concevoir et de construire des édifices
- Anglais architecture,
- Sens : Disposition, style d'un édifice
- Structure, organisation de quelque chose, d'un ensemble de choses

Définitions (Architecture)

- Une **architecture** est une description formelle d'un système, la définition de son objet, les fonctions, les propriétés visibles de l'extérieur, et des interfaces. Il inclut également la description des composants internes du système et leurs relations, ainsi que les principes régissant la conception, l'exploitation et l'évolution. (Srinivasan L., 2005)

Définitions (Service)

- **Service**, nom masculin :
- Sens 1 : Fonction de quelqu'un qui sert une cause ou qui aide une personne.
- Sens 2 : Partie d'une facture qui est destinée au personnel, pourboire.
- Sens 3 : Travail. Ex Commencer son service à 15h. Synonyme travail.
- Sens 4 : Département, branche d'activité d'une entreprise. Ex Le service marketing. Synonyme administration

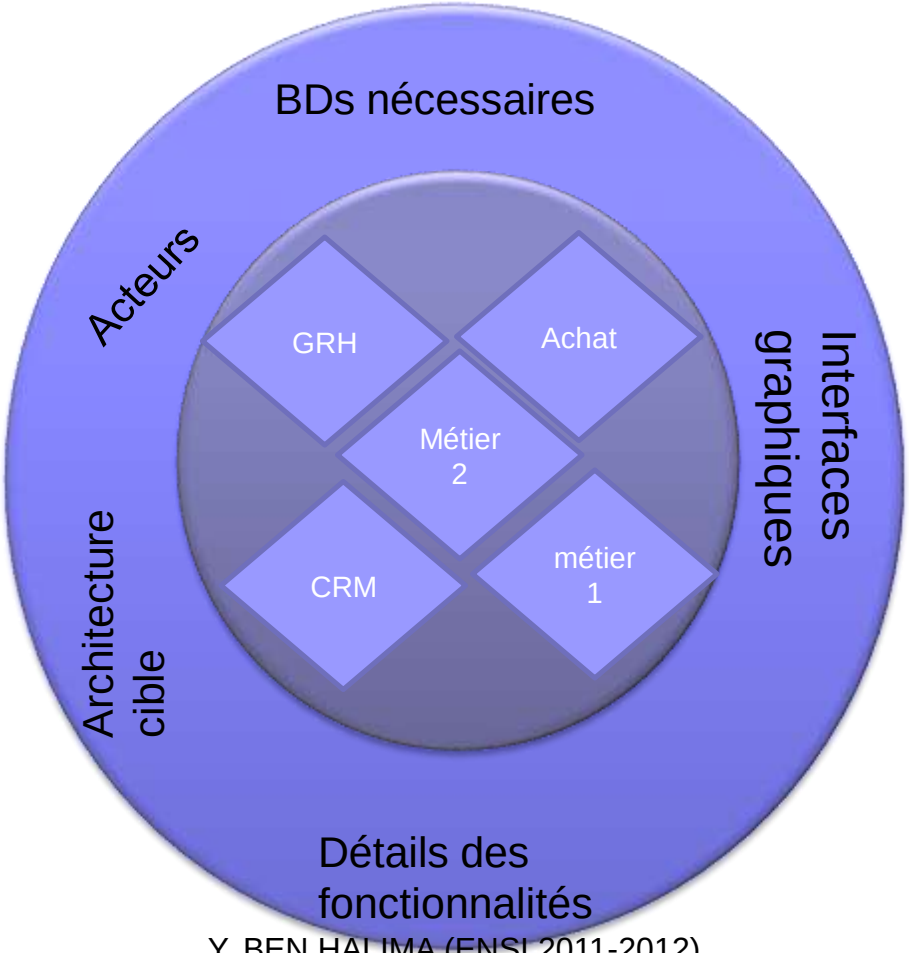
Définitions (Architecture SOA)

- Le terme **architecture orientée services** fait référence à un style de construction fiables des systèmes distribués qui offrent des fonctionnalités comme les services, en mettant l'accent sur la faiblesse de l’association entre les services qui interagissent (couplage lâche = association très faible)

Exemple :

- Dans la conception d'un Système informatique pour une entreprise, mettre en place une architecture orientée services revient à modéliser tout le système en se focalisant sur les services métier offerts par le système et non pas la base de données ou les interfaces (traités par la suite)

Exemple :


BDs nécessaires
Acteurs
GRH
Achat
Métier 2
CRM
métier 1
Interfaces graphiques
Architecture cible
Détails des fonctionnalités

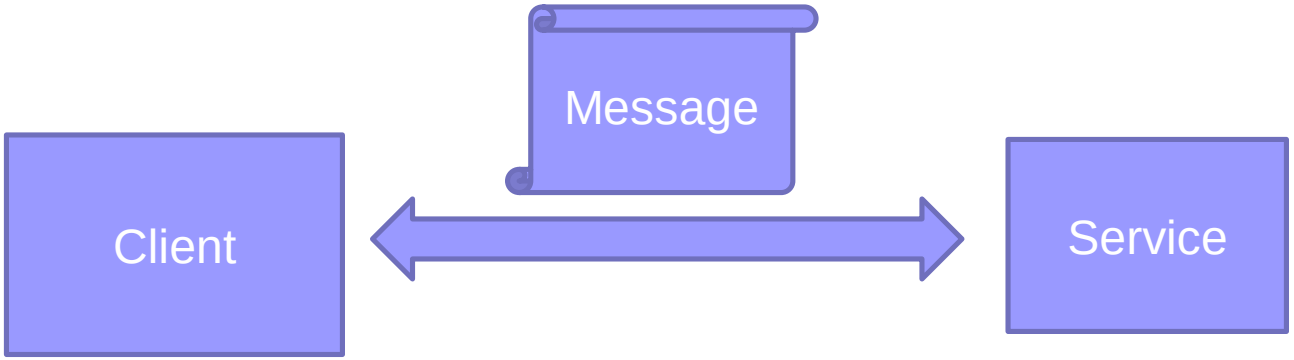
Architecture SOA

- Techniquement, le terme SOA se réfère à la conception d'un système basé sur l'offre de services, et non pas à sa mise en œuvre.
- Mais nous pouvons considérer que SOA est un style architectural qui met l'accent sur la mise en œuvre des composants comme des services modulaires qui peuvent être découverts et utilisées par les clients.

Caractéristiques des services et des SOA

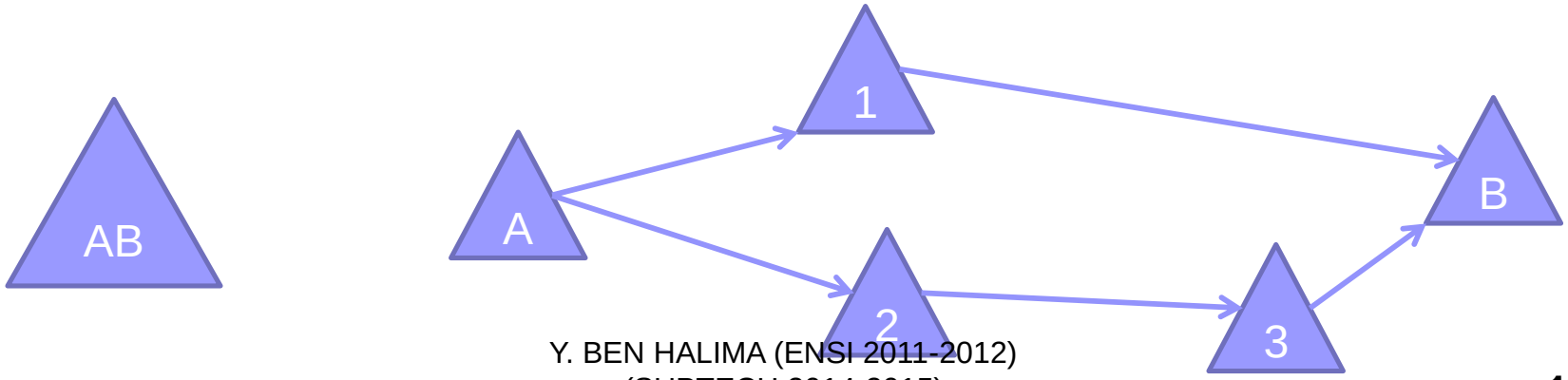
Caractéristiques des services
1- Echange de messages

- Comment communiquer ? : Services communiquent avec leurs clients en échangeant des messages: ils sont définis par les messages qu'ils peuvent accepter et les réponses qu'ils peuvent donner.



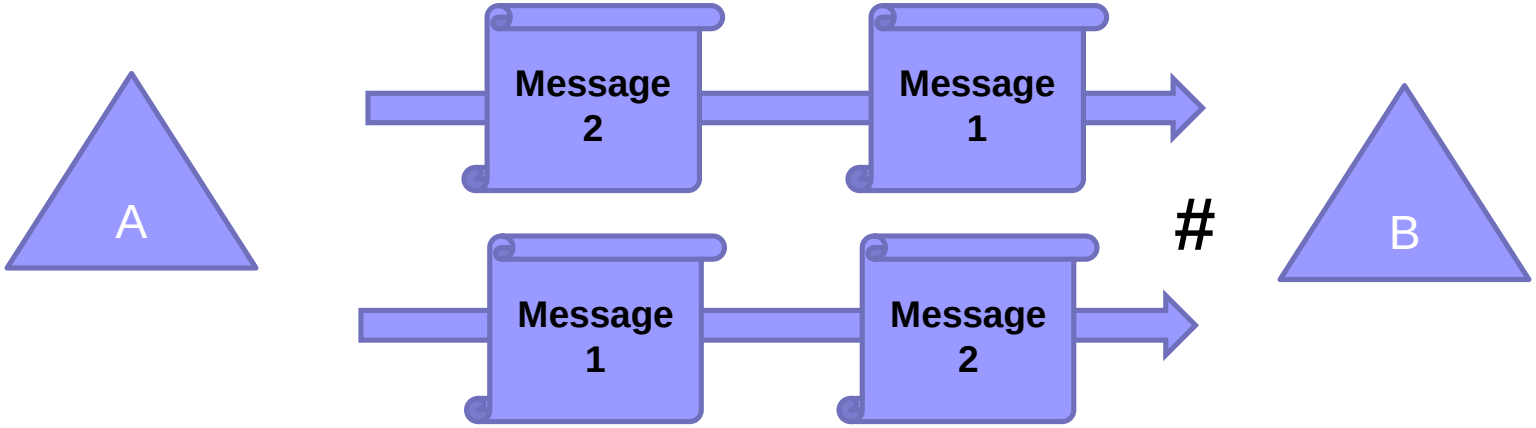
2- Composition de services

- Réutilisation par composition : Les services peuvent être individuellement utiles, ou ils peuvent être intégrés, composé pour fournir des services de haut niveau. Ce qui favorise la réutilisation des fonctionnalités existantes. Cette notion est définie comme « la composition de services »



3- chorégraphie de services

- Ordre d'appel des services : Les services peuvent participer à un workflow, où l'ordre dans lequel les messages sont envoyés et reçus affecte le résultat des opérations effectuées par un service. Cette notion est définie comme « la chorégraphie de services »



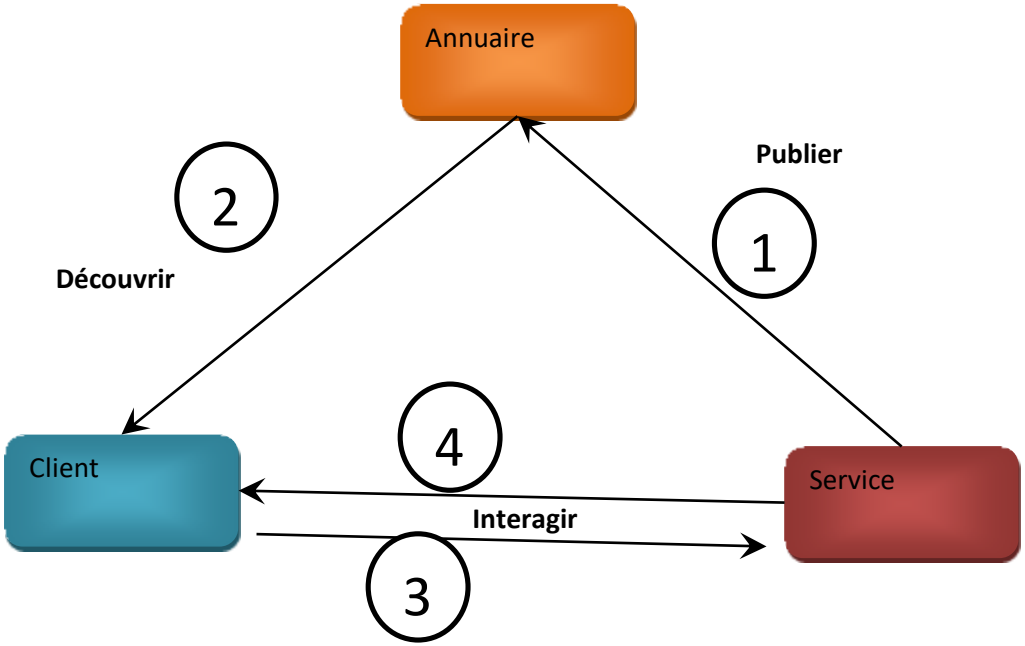
4- Dépendance

- Les services peuvent être totalement autonome, mais peuvent aussi dépendre de la disponibilité des autres services, ou sur l'existence d'une ressource comme une base de données par exemple.
- Exemples :
  - Service indépendant: Dans le cas le plus simple, un service peut effectuer un calcul comme le calcul de la racine cubique d'un nombre fourni sans avoir besoin de se référer à une ressource externe, ou il peut aussi charger à l'avance toutes les données dont il a besoin pour son exécution.
  - Service dépendant : Inversement, un service qui effectue la conversion des devises devrait accéder en temps réel aux informations des taux de change permettant de dégager des valeurs correctes.

5- Code de développement inaccessible

- Les Services publient des détails tels que leurs capacités, les interfaces, les politiques et protocoles pris en charge pour la communication.
- Les détails d'implémentation comme le langage de programmation et la plate-forme d'hébergement ne sont pas préoccupantes pour les clients, et ne sont pas révélées pour des raisons de sécurité.

Cycle d'intéraction des services



Description du cycle d’intéraction

- Un simple cycle d’interaction de service commence par la publication du service dans un annuaire de services bien connu (1).
- Un client potentiel, qui peut ou peut ne pas être un autre service, interroge le registre (2) à la recherche d'un service qui répond à ses besoins.
- Le registre renvoie une liste (éventuellement vide) des services appropriés, et le client choisit un et lui passe un message de demande, en utilisant un des protocoles mutuellement reconnus (3).
- Dans cet exemple, le service répond (4) ou avec le résultat de l'opération demandée ou avec un message d'erreur.

Caractéristiques de SOA
1-Couplage lâche

- Le couplage est une métrique indiquant le niveau d'interaction entre deux ou plusieurs composants logiciels (fonctions, modules, objets ou applications). *Deux composants sont dits couplés s'ils échangent de l'information*.
- On parle de couplage fort ou couplage serré si les composants échangent beaucoup d'information. On parle de couplage faible, couplage léger ou couplage lâche si les composants échangent peu d'information.

Avantages du couplage lâche

- **Flexibilité** : Un service peut être situé sur n'importe quel serveur, et déplacées au besoin. Tant qu'il maintient son entrée de registre, les clients potentiels seront en mesure de le trouver.
- **Evolutivité**: Les services peuvent être ajoutés et supprimés, car la demande des clients varie. (supprimer le service inutile)
- **Interchangeabilité** : la mise en ligne de nouvelles versions des services est transparente pour les utilisateurs. Les mises à jours des services se font sans interruption pour les utilisateurs.
- **Tolérance aux pannes**: Si un serveur, un composant logiciel, ou un segment de réseau tombe en panne, ou le service devient indisponible pour toute autre raison, les clients peuvent interroger le Registre pour les services de remplacement qui offrent les fonctionnalités requises, et de continuer à fonctionner sans interruption.

2-Service avec ou sans état

- Dans certains cas, le service est obligé de sauvegarder son dernier état à fin d'en disposer en cas de besoins, dans d'autres cas ce n'est pas nécessaire.
- Exemples :
  - Dans le cas simple d'une calculatrice ou d'un service de prix-enligne, il est facile de voir qu'une fois un client a demandé et reçu des informations, l'opération est terminée, et le client n'a pas besoin particulier de revoir le même service pour ses besoins futurs.

- Solution possible : Pour une transaction plus complexe qui nécessite plusieurs étapes, la conception du service pourrait être tel que le service conserve dans sa mémoire locale de l'information («état») au sujet de la première étape, s'attendant à en faire usage lorsque les clients les contactent pour la prochaine étape.
- Dans ce cas, le service est «avec état», et le client doit retourner au même service pour la prochaine étape. Cela pourrait entraîner un retard si de nombreux clients utilisent le même service ou un échec de la transaction si le nœud qui héberge le service échoue entre les étapes.

Et comment sauvegarder l’état?

1) Donnez un exemple de service qui nécessite la sauvegarde de l’état
2) Vos suggestions pour gérer la sauvegarde de l’état

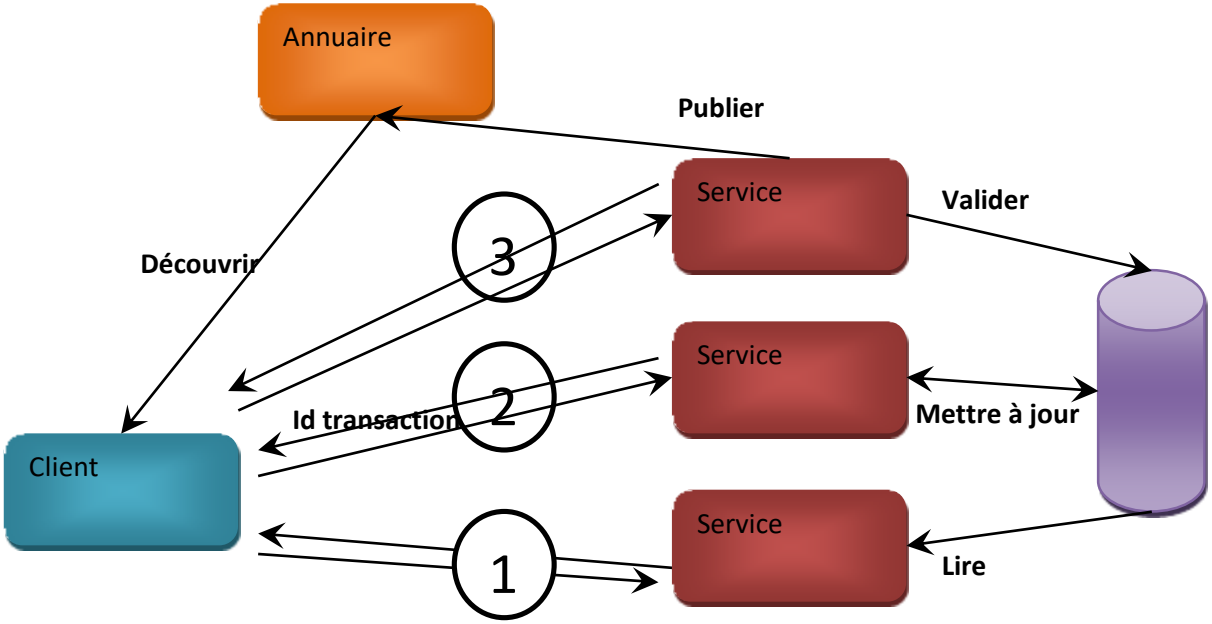
Exemple de services avec et sans état (à définir)

- Services de réservation en ligne d'un billet d'avion+payement. Composé de deux services qui s'exécutent en succession.
- Services de lancement par un client quelconque d'une commande d'articles auprès d'un fournisseur.
- Services de consultation et achat des actions en bourse en ligne

Sauvegarde de l’état chez le client

1) À la fin de chaque étape intermédiaire le service doit restituer au client les informations suffisantes sur l'état pour permettre à tout service qualifié d'identifier et de poursuivre la transaction.
2) Le client doit récupérer les informations d'état du service qu'il choisit pour traiter la prochaine étape de la transaction.
3) Le service choisi doit être en mesure d'accepter et de traiter les informations d'état fournies par le client, même si ce n’est pas le même service qui a traité les étapes précédentes.

Exemple :


Annuaire
Publier
Service
Valider
Découvrir
3
Service
2
Id transaction
Mettre à jour
Client
Service
1
Lire

3- temps d’exécution des services

- Calculé en fonction :
  - de l’horloge du processeur de la machine,
  - du nombre d’itération qui se trouvent dans un programme,
  - de l’emplacement du programme,
  - du langage de programmation,
  - …
- Un paramètre important intervenant dans le calcul de la durée d’exécution est l’interaction avec l’utilisateur. Plus l’utilisateur interagit avec l’application plus son exécution est considéré comme longue.

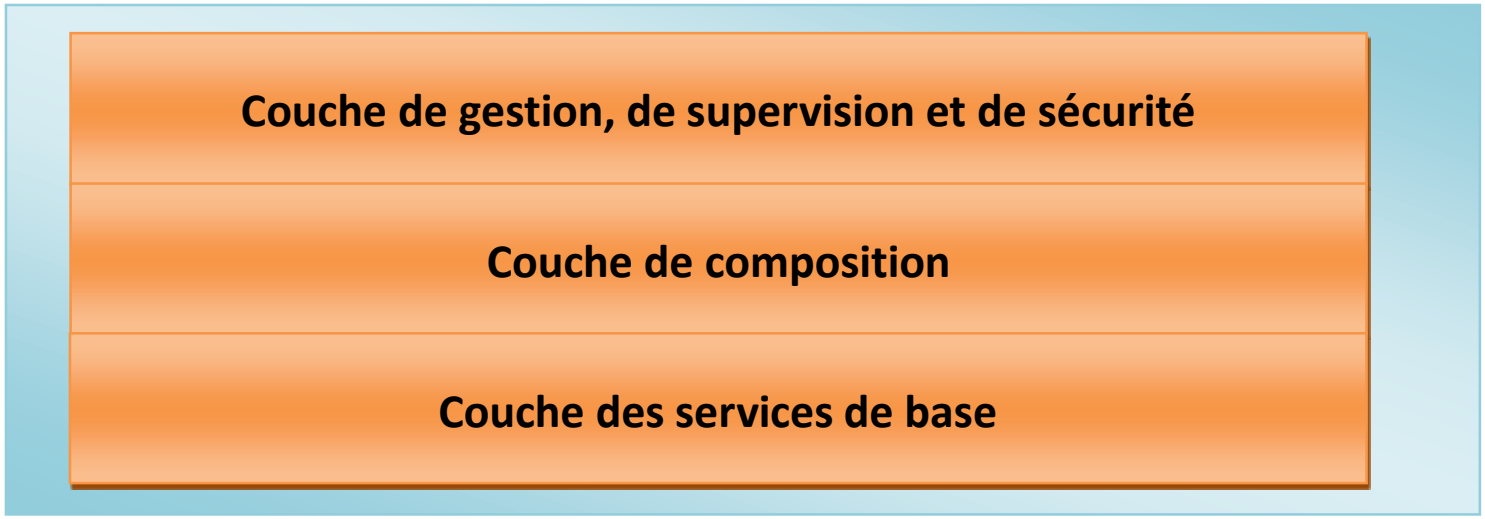
Architectures XSOA

Nécessité de XSOA

- il est parfois nécessaire de combiner ou de composer un ensemble de services en services plus complexes, qu’on appelle service composite, pour répondre à des exigences encore plus complexes

- cette architecture étendue expose un certain nombre de fonctionnalités complémentaires. Par exemple, un service doit être décrit et découvert non seulement en fonction de son interface mais aussi en considérant d'autres aspects à savoir, la sécurité, la disponibilité, les performances et la qualité de services (QoS – Quality of Service).

Structure en couches



- La première couche met en place les fondements des services de base en utilisant l'architecture SOA.
- La deuxième couche s'intéresse aux services composites en proposant les méthodes et outils nécessaires à leurs exécutions. De plus, elle comprend des fonctionnalités liées à la qualité des services composites.
- La troisième couche s'intéresse à la gestion, la supervision et la sécurisation des services en mettant en œuvre l'ensemble des outils et ressources nécessaires.

Exercice SOA

- Donnez une architecture orientée services du système informatique d'une banque
  - Définir les services a présenter
  - Définir les acteurs de ce système
  - Attribuer les services par acteur
  - Dessiner une architecture générale en couche

Eléments de solution

Sécurité par cryptage des communications, gestions des compositions et supervision du fonctionnement, remplacement des services en panne

Service de transfert d'argent d'un compte vers un autre, service de changement de l'adresse du client

Authentification, consultation compte, mise a jour compte, vérification profil, ….

Plan

1. Architectures SOA
   - Définitions
   - Caractéristiques de SOA
   - Architectures XSOA
2. Services Web
   - Définitions et Principes
   - XML (Rappel et Notions)
   - SOAP
   - WSDL
   - UDDI
   - Exemples
3. Cloud Computing
   - Grid computing
   - Différents types de Cloud
     - IAAS
     - PAAS
     - SAAS

PARTIE 2 :
Services Web : La technologie

Définition

Définition

- Les Web Services sont des services offerts via le web.
- "Un Web Service est une application logicielle identifiée par un URI (Uniform Resource Identifier), dont les interfaces et associations peuvent être définies, décrites et découvertes par des méthodes XML, et qui peut interagir directement avec d'autres applications en utilisant des messages XML via les protocoles Internet standards."

  « Source : W3C »
- "Logiciel conçu pour être utilisé par d'autres logiciels à travers les protocoles et standards Internet. "

  « Source : Forrester Research »

- Par exemple, un client demande le prix d'un article en envoyant un message sur le web. Ce message contient la référence de l'article. Le Web Service va recevoir la référence, effectuer le traitement du service et renvoyer le prix au client via un autre message.

Définitions …

- Un Service Web est une « unité logique applicative » accessible en utilisant les protocoles standard d'Internet.
- Un objet métier qui peut être déployé et combiné sur Internet avec une faible dépendance vis-à-vis des technologies et des protocoles.
- Combine les meilleurs aspects du développement à base de composants et du Web.
- Il s'agit donc d'un ensemble de fonctionnalités exposées sur internet ou sur un intranet, par et pour des applications ou machines, sans intervention humaine, et *en temps réel*

Principes

Pourquoi un nouveau middleware

Limitations des middleware

Passage à large échelle : Web

- Protocoles hétérogènes IIOP, RMI, DCOM
- Pas d'ouverture des services : Notion de moteur de recherche de service inexistante
- Trop de contraintes sur le client !
  - Doit posséder les souches
  - Difficulté de construire dynamiquement

Limitations des middleware

Inconvénients

- Complexité
  - CORBA : IDL, Mapping, …
  - EJB : Container, JNDI, …
- Pérennité (continuité, durabilité) : remise en question
  - CORBA, EJB, .Net, …
- Prix (mise en place, maintenance, …)
  - Plates-formes
  - Compétences

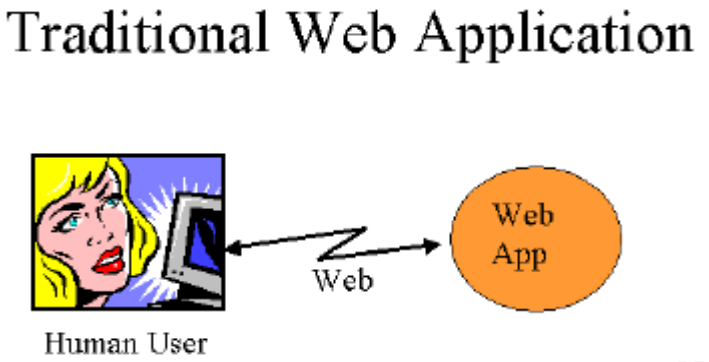
Solutions existantes

- Modification du Protocole (évolution d'un protocole existant)
- Passerelles (d'un protocole à un autre)
- Portage d'applications existantes difficile

→Solutions non standards

Le web traditionnel

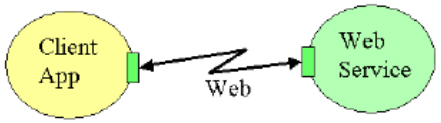
- Protocole: HTTP
- Documents: HTML
- Des millions de sites web indépendants, des milliards de pages
- Navigation et recherche par mots-clés
- Publication de bases de données, accessibles à travers des formulaires.



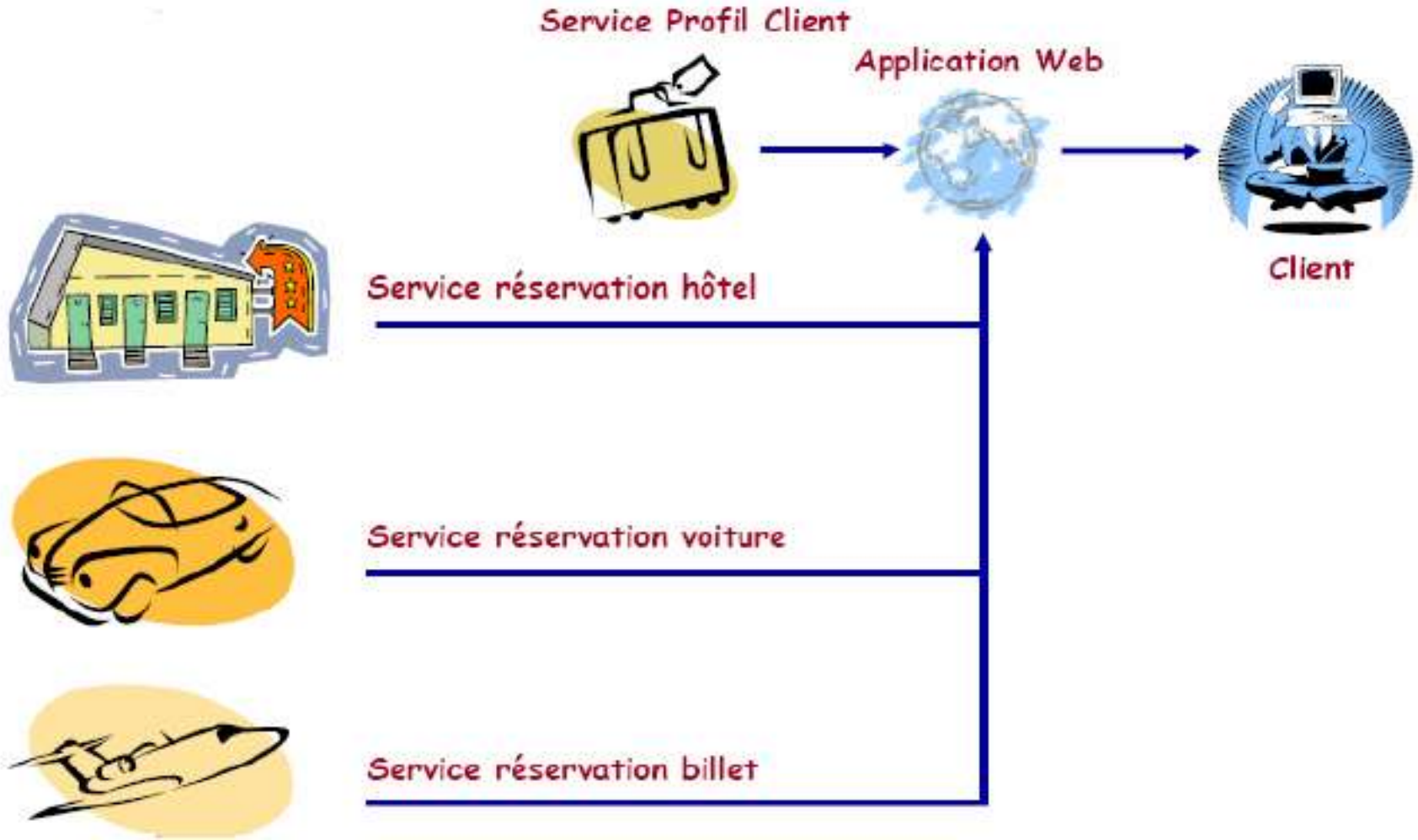
Les services web

- Possibilité d'invoquer une fonction sur un serveur web distant
- ***Fournit une infrastructure souple pour les systèmes distribués, basée sur XML***
- 2 applications principales
  - Commerce électronique
  - ***Accès à des bases de données distantes***

Web Service (WS)



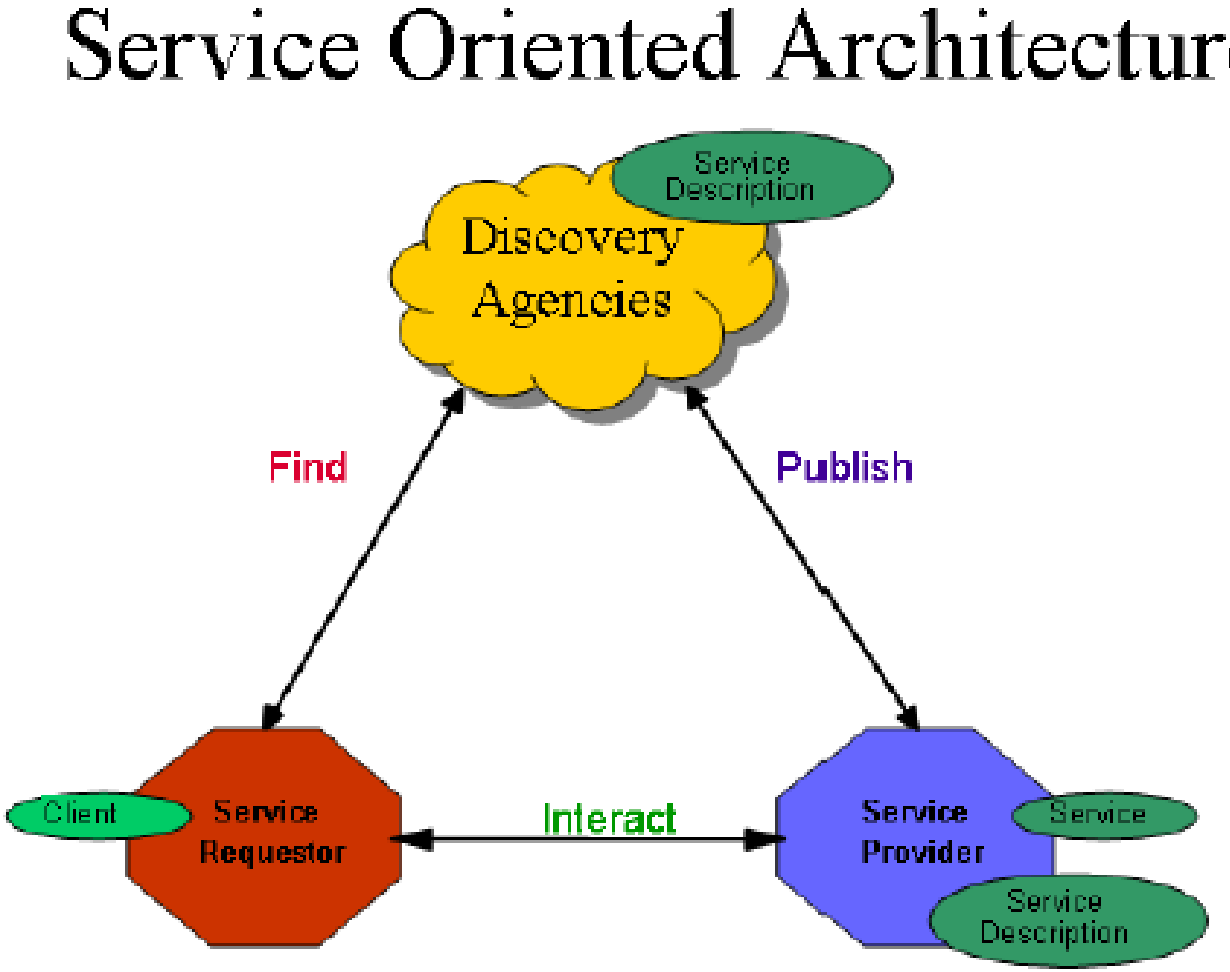
Exemple d'utilisation des services web



L’idée principale : trouver un Truc !

- 1. Interroger un annuaire : qui fournit des Trucs?
- 2. Négocier avec les fournisseurs potentiels
  - Nature exacte du service fournis
  - Qualité/coût/etc.
- 3.Interagir avec le service du fournisseur choisi
  - Connaître les modalités d’interaction
  - Introduire le service dans ma chaîne de traitements
- 4.Eventuellement composer des services
- 5.Eventuellement publier mes propres services

Principe général d'architecture



Principe général d'architecture

- L'architecture des services web repose essentiellement sur les technologies suivantes:

SOAP
W3C
Simple Object
Access Protocol

Transporte

WSDL
W3C
Web Services
Description Language

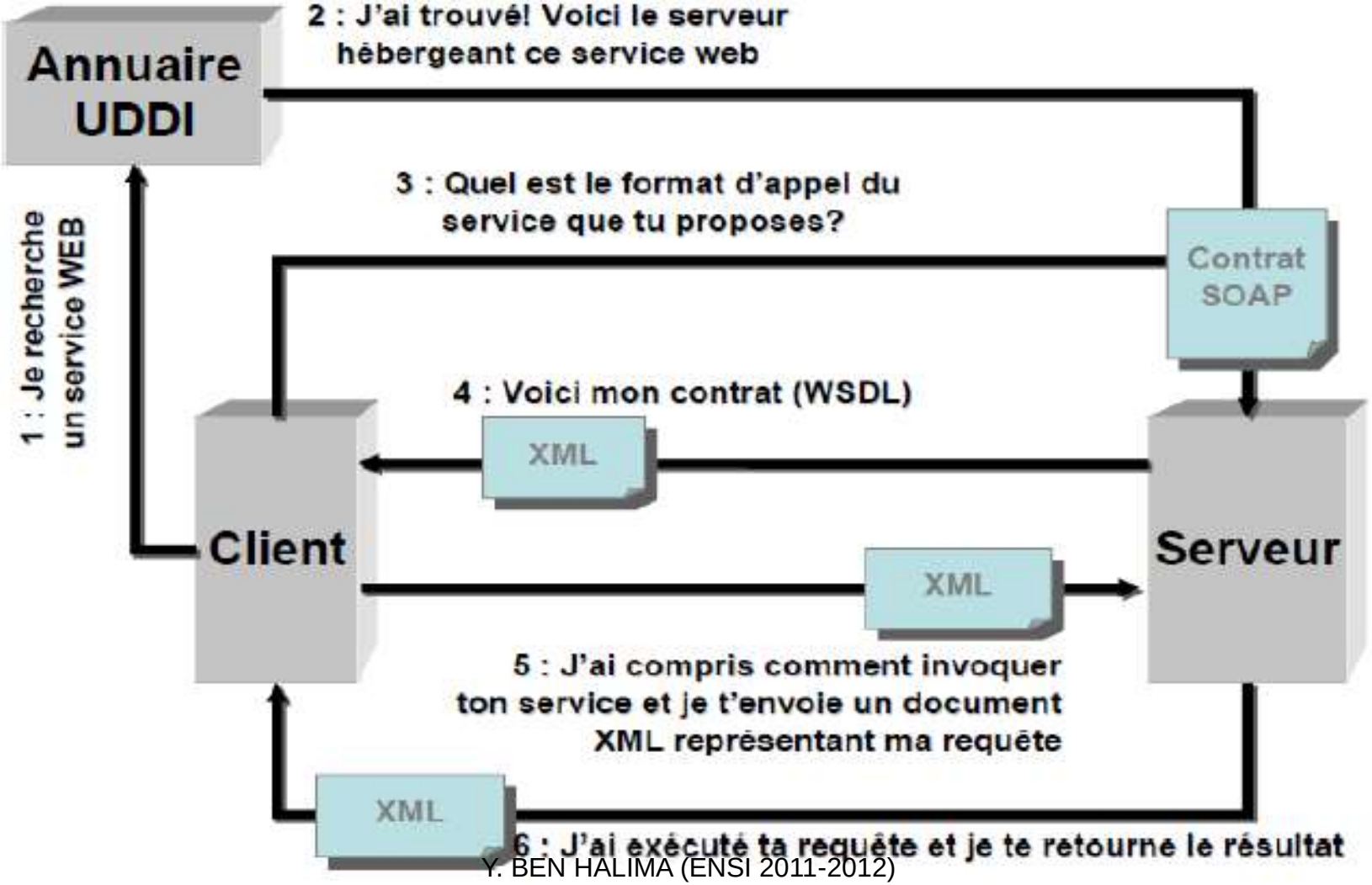
Décrit le contrat

UDDI
Microsoft,IBM, HP
Universal Description
Discovery and Integration

Stocke les descriptions de contrat

XML 1.0
eXtensible Markup Language

Un service web en action



Approche Envisagée

- Un nouveau Protocole de communication: SOAP
  - Basé sur XML
    - Portabilité, Hétérogénéité
  - Porté sur des protocoles large échelle existants
    - HTTP, SMTP, …
- Paradigme orienté service : WSDL
  - Définition de services offerts (en XML)
- Découverte automatique des services (dynamicité) : UDDI
  - Référentiel de Web Service (Pages Jaunes, Vertes, Blanches)

SOAP

- Protocole d'échange de messages (client / serveur)
- Basé entièrement sur XML
- Standard W3C (Initiative IBM et Microsoft)
  - Actuellement SOAP 1.2 (2007)
- Concepts
  - Message = Enveloppe (Header + Body)
- Extensibilité
  - Porté sur HTTP, SMTP, …

WSDL

- Langage de définition de Web Services
- Basé entièrement sur XML
- Standard W3C (Initiative IBM et Microsoft)
  - Actuellement WSDL 2.0 (2007)
- Définition de l'interface, de l'URL et du port du Web Service.
- Utilise le système de typage de XML Schéma

UDDI

- Référentiel de définitions Web Service
- Recommandation OASIS
- Référentiel défini lui-même en WSDL
- Référentiel Public / Privé
- UDDI 3.0 (2008)

XML

Extensible Markup Langage
Version 1.1 (2006)

Exemple de document XML

```
<livre>
        <titre> le super livre </titre>
        <chapitre>
                <numero> 1 </numero>
                <titre> titre du chapitre 1 </titre>
                <contenu> blabla blabla </contenu>
        </chapitre>
        <chapitre>
                …
        </chapitre>

</livre>
```

Principes

- Ensemble non fini de balises
  - L'utilisateur peut créer de nouvelles balises
- Définition de grammaires : XML est un Meta-Langage
  - MathML, NewsML, XMI, Doc, Slides, …
- Séparation de la forme et du fond
  - Un document XML peut être constitué de deux entités (le fond et la forme)

Grammaire

Deux façons de définir une grammaire XML :

- DTD
  - Langage de définition de grammaire XML
  - Largement utilisé
  - Expression faible (type, structure)
- XML Schéma
  - Langage XML de définition de grammaire XML
  - De + en + utilisé
  - Expression puissante (**type**, structure, héritage)

Un document XML est dit **valide** lorsqu'il est **conforme** à une grammaire,

bien formé lorsqu'il respecte la syntaxe d'un document XML (pas d'erreurs d'ecriture par exemple)

Espaces de noms

- Mécanismes permettant de partitionner les balises XML (permet d'avoir deux fois le même nom de balise)
- Un espace de nom est défini dans n'importe quelle balise par l'attribut xmlns et par une URI.
- Dans un document XML, un espace de noms est identifié par un nom logique, les balises appartenant à cet espace doivent alors être préfixée par ce nom logique.
- Ex :

```
<meta:body xmlns:meta="http://meta.ensi.rnu.tn/meta/"
```

XML est un succès !

- Standard W3C
- La syntaxe XML ne contient que peu de mot clef: Simplicité
- XML est indépendant des plates-formes: Portabilité de fichiers plats.
- XML est un méta-langage, il est possible de créer ses propres balises: Extensibilité
- Outils disponibles (et gratuits)

Largement utilisé pour les échanges inter-applications

A vous de jouer !

- Pouvez-vous définir un document XML qui représente un message Client/Serveur ?
- Quelles sont les informations que ce message doit contenir ?
- Comment échanger ce message ?
  - Quels sont les problèmes soulevés ?
- Pouvez vous définir un document XML qui décrit un service offert sur le web

XML

Introduction à XML

-Origines et Objectifs

-Introduction à XML

-XML pour quoi faire ?

XML, le langage

- Déclaration
- Racine
- Elément
- Attribut
- Entité
- Règles du développement XML
- Utilisation des Namespaces

DTD

CSS

XSL

XSD

Origine et objectifs

- XML est issu de la Gestion de Documents (GED)
- Séparation du fond et de la forme.
  - Forme = présentation à partir de la structure (style)
  - Fond = structure + données (contenu)
- Multiples précurseurs dont les plus connues :
  - HTML pour la présentation

Présentation et Structuration

Titre

Auteur

Section

Paragraphe

Paragraphe

Paragraphe

Section

XML: Des BD aux Services Web

Georges Gardarin

1. Introduction

Ces dernières années ont vu l'ouverture des systèmes d'information à l'Internet. Alors que depuis les années 1970, ces systèmes se développaient, le choc **Internet** …

Ainsi, on a vu apparaître une myriade de **technologies** nouvelles attrayantes mais peu structurantes voir perturbantes. Certaines n'ont guère survécues …

L'urbanisation passe avant tout par la standardisation des échanges : il faut s'appuyer sur des standards ouverts, solides, lisibles, sécurisés, capable d'assurer l'interopérabilité avec l'Internet et les systèmes d'information …

2. La société ProXML

Vue Balisée: Structure logique

```
<?xml version="1.0" encoding="iso-8859-1"?>
<Livre>
<Titre> XML : Des BD aux Services Web</Titre>
<Auteur>Georges Gardarin</Auteur>
<Section titre = "Introduction">
<Paragraphe>Ces dernières années ont vu l'ouverture des systèmes d'information à
    l'Internet. Alors que depuis les années 1970, ces systèmes se développaient, le choc
    Internet … </Paragraphe>
<Paragraphe>Ainsi, on a vu apparaître une myriade de technologies nouvelles
    attrayantes mais peu structurantes voir perturbantes. Certaines n'ont guère
    survécues … </Paragraphe>
<Paragraphe>L'urbanisation passe avant tout par la standardisation des échanges : il
    faut s'appuyer sur des standards ouverts, solides, lisibles, sécurisés, capable
    d'assurer l'interopérabilité avec l'Internet et les systèmes d'information …
    </Paragraphe>
</Section>
<Section titre= "La Société ProXML"> …
</Section>
</Livre>
```

Les balises (tags) peuvent porter plus ou moins de sémantique

Vue Balisée: Structure physique

```
<Livre>
  <Titre police="Times" taille="24" position="centré" format="gras"/>
  <Auteur police="Times" taille="20" position="centré" format="italique"/>
  <Section numero="1" police="Times" taille="18" position="centré" format=" gras "/>
  <Paragraphe police="Times" taille="18" position="justifié"/>
</Livre>
```

XML : objectifs

- XML= un nouveau langage d'échange basé sur le balisage
- XML= plus ouvert que HTML ➔ possibilité de définition de nouvelles balises
- XML = développé par XML Working Group dirigé par le W3C (depuis 1996)
- XML 1.0 = recommandation officielle du W3C depuis le 10 février 1998

les 10 objectifs de conception:

- XML doit pouvoir être utilisé sans difficulté sur Internet
- XML doit soutenir une grande variété d'applications
- XML doit être compatible avec HTML, XHTML, SGML, etc.
- Il doit être facile d'écrire des programmes traitant les documents XML
- Le nombre d'options dans XML doit être réduit au minimum, idéalement à aucune

les 10 objectifs de conception

- Les documents XML doivent être lisibles par l'homme et raisonnablement clairs
- La spécification de XML doit être disponible rapidement
- La conception de XML doit être formelle et concise
- Il doit être facile de créer des documents XML
- La concision dans le balisage de XML est peu importante

Forces de XML

- Séparation de la structure et de la présentation
- Moins confus que HTML
- Idéal pour l'échange de données semi-structurées
- Utilisable entre machines

EXEMPLE

De :youssef ben halima
A : SUPTECH
Sujet : SOA
XML + CSS + DTD + XSL

HTML	XML
<HTML><HEAD> <TITLE> Message </TITLE> <BODY><TABLE> <TR><TD> De : </TD> <TD> youssef ben halima</TD> </TR> <TR><TD> A :</TD> <TD> SUPTECH</TD> </TR> <TR><TD>Sujet :</TD> <TD>SOA</TD> </TR><TR><TD></TD> <TD>XML ….. </TR> </TABLE></BODY></HTML>	<?xml version= ”1.0 ”?> <message> <header> <from>> youssef ben halima</from> <to>SUPTECH</to> <subject>SOA</subject> </header> <body> <para> XML ……</para> </body> </message>

XML: définitions de base

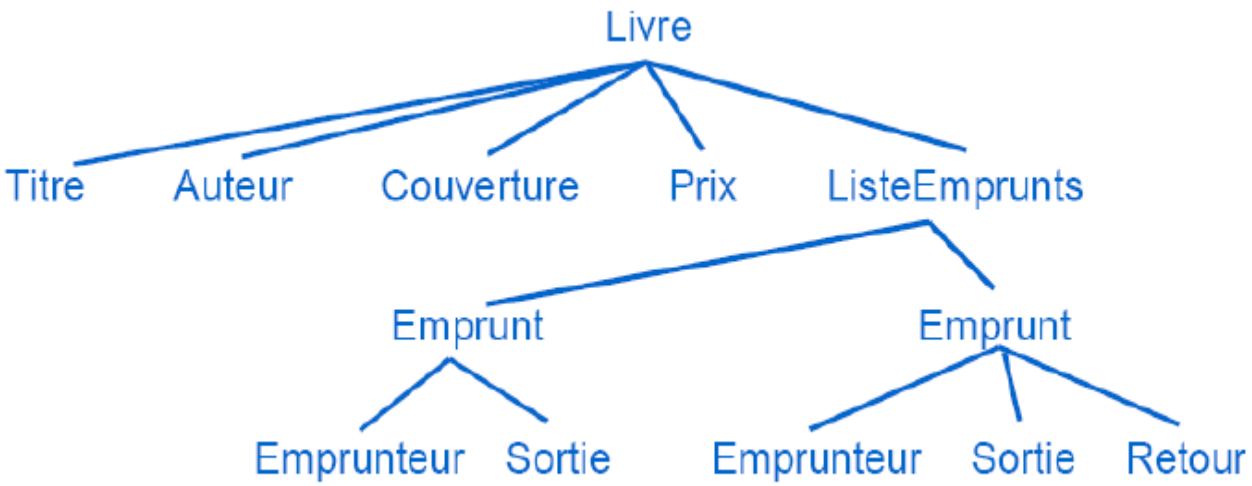
- XML est un méta-langage universel pour représenter les données échangées sur le Web qui permet au développeur de délivrer du contenu depuis les applications à d'autres applications ou aux navigateurs
- XML standardise la manière dont l'information est :
  - échangée
  - présentée
  - archivée
  - retrouvée
  - transformée
  - cryptée
  - ...

Exemple document XML

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<?xml-stylesheet href="bib.css" ?>
<?cocoon-process type="xslt" ?>
<!DOCTYPE Livre SYSTEM "Livre.dtd">

<Livre>
  <Titre>Les réseaux</Titre>
  <Auteur>A. Tanenbaum</Auteur>
  <Couverture imgsrc="/imgs/res-tan.jpg"/>
  <Prix devise="EUR">25.42</Prix>
  <ListeEmprunts>
    <Emprunt>
      <Emprunteur>François Duchemin</Emprunteur>
      <Sortie>25/09/2000</Sortie>
      <Retour>02/10/2000</Retour>
    </Emprunt>
    <Emprunt>
      <Emprunteur>Hervé Delarue</Emprunteur>
      <Sortie>05/10/2000</Sortie>
    </Emprunt>
  </ListeEmprunts>
</Livre>
```

Structure Arborescente



XML: contextes d’utilisation

- Un standard d'échange
  - Lisible : texte balisé avec marquage
  - Clair : séparation du fond et de la forme
  - Extensible : supporte les évolutions applicatives
  - Sécurisé : pare-feu, encryption, signature
- Développé par le W3C
  - Pour le Web (Internet, Intranet)
  - S'étend à l'entreprise et ses partenaires
- Supporté par les grands constructeurs
  - IBM, Microsoft .net, SUN, BEA, etc.
  - Des outils génériques et ouverts

Stockage de données

Fichiers de configuration (JBOSS)

```
<?xml version="1.0" encoding = "ISO-8859-1"?>
<!DOCTYPE ejb-jar PUBLIC '-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 2.0//EN'
'http://java.sun.com/dtd/ejb-jar_2_0.dtd'>

<ejb-jar>
   <enterprise-beans>
     <session>
            <ejb-name>LegoBlocksEJB</ejb-name>
            <home>legoBean.LegoBlocksSessionHome</home>
            <remote>legoBean.LegoBlocksSession</remote>
            <ejb-class>legoBean.LegoBlocksSessionBean</ejb-class>
            <session-type>Stateful</session-type>
            <transaction-type>Container</transaction-type>
     </session>
   </enterprise-beans>
</ejb-jar>
```

Forces et faiblesses de XML

- Une technologie structurante
- Clarifie tous les échanges
- Des standards internes et externes
- Transversale à l'entreprise
  - Échanges de données
  - Bureautique
  - GED
  - Sites Web
  - EDI
  - Bases de données
  - Intégration e-business
  - …

- Une syntaxe bavarde
- Un méta-langage, mais de nombreux langages
- Coûteux en CPU
  - Parsing
- Coûteux en mémoire
  - Instanciation
- Format compressé à l'étude

Exercice :1

- Le paragraphe suivant contient de l'information "en vrac".
- Réorganisez-la de manière à mettre en évidence sa structure logique,
-
- Une bouteille d'eau Cristaline de 150 cl contient par litre 71 mg d'ions positifs calcium, et 5,5 mg d'ions positifs magnésium. On y trouve également des ions négatifs comme des chlorures à 20 mg par litre et des nitrates avec 1 mg par litre. Elle est recueillie à **St-Cyr la Source**, dans le département du Loiret. Son code barre est 3274080005003 et son pH est de 7,45. Comme la bouteille est sale, quelques autres matériaux comme du fer s'y trouvent en suspension.
-
  Une seconde bouteille d'eau Cristaline a été, elle, recueillie à la source d'**Aurèle** dans les Alpes Maritimes. La concentration en ions calcium est de 98 mg/l, et en ions magnésium de 4 mg/l. Il y a 3,6 mg/l d'ions chlorure et 2 mg/l de nitrates, pour un pH de 7,4. Le code barre de cette bouteille de 50 cl est 3268840001008.
  Une bouteille de même contenance est de marque Volvic, et a été puisée à... **Volvic**, bien connu pour ses sources donnant un pH neutre de 7. Elle comprend 11,5 mg/l d'ions calcium, 8,0 mg/l d'ions magnésium, 13,5 mg/l d'ions chlorures et 6,3 mg/l d'ions nitrates. Elle contient également des particules de silice. Son code barre est 3057640117008.
-
  PS : Volvic est dans le Puy-de-Dôme...

Correction

```
<cave>
<bouteille>
<marque>Cristaline</marque>
<composition>
<ion type="positif">calcium 71mg/l</ion>
<ion type="positif">magnésium 5,5mg/l</ion>
<ion type="negatif">chlorure 20mg/l</ion>
<ion type="negatif">nitrate 1mg/l</ion>
<autre type="metal">fer</autre>
</composition>
<source>
<ville>St-Cyr la Source</ville>
<departement>Loiret</departement>
</source>
<code_barre>3274080005003</code_barre>
<contenance unit="cl">150</contenance>
<ph>7,45</ph>
</bouteille>
<bouteille>
<marque>Cristaline</marque>
<composition>
<ion type="positif">calcium 98mg/l</ion>
<ion type="positif">magnésium 4mg/l</ion>
<ion type="negatif">chlorure 3,6mg/l</ion>
<ion type="negatif">nitrate 2mg/l</ion>
</composition>
<source>
<ville>Aurèle</ville>
<departement>Alpes Maritimes</departement>
</source>
<code_barre>3268840001008</code_barre>
<contenance unit="cl">50</contenance>
<ph>7,4</ph>
</bouteille>
<bouteille>
<marque>Volvic</marque>
<composition>
<ion type="positif">calcium 11,5mg/l</ion>
<ion type="positif">magnésium 8mg/l</ion>
<ion type="negatif">chlorure 13,5mg/l</ion>
<ion type="negatif">nitrate 6,3mg/l</ion>
</composition>
<source>
<ville>Volvic</ville>
<departement>Puy-de-Dôme</departement>
</source>
<code_barre>3057640117008</code_barre>
<contenance unit="cl">50</contenance>
<ph>7</ph>
</bouteille>
</cave>
```

Exercice 2 :

- Cet exercice est du même type que l'exercice précédent. Il s'agit de structurer le texte suivant :
- Il existe diverses variétés de nuages. La plupart de ceux dont nous allons parler ne produit aucun "hydrométéore", sauf le cumulonimbus, qui est accompagné d'averses (parfois sous la forme de neige, de grésil ou de grêle).
- L'altocumulus et le cirrocumulus partagent les mêmes "espèces" : *lenticularis*, *stratiformis*, *castellanus* et *flocus*.
- On retrouve ces deux espèces également chez le cirrus, ainsi que les espèces *spissatus*, *uncinus* et *fibratus*. Les espèces *stratiformis*, *lenticularis* et *castellanus* sont quant à elles partagées également avec les strato-cumulus.
- Ces derniers peuvent se traîner au ras du sol et monter à 2000m, mais certains nuages ont une altitude minimale à peine plus élevée, puisqu'elle n'est que de 200m pour les cumulus, et de 300m pour les cumulonimbus. Il est vrai que ces derniers compensent en montant jusqu'à une altitude maximale de 18000m, soit plus haut encore que les cirrus, qui plafonnent à 12000m. L'altitude minimale de ces derniers coïncide avec la fin de la présence possible des altocumulus, à 6000m. Et c'est autour de cette zone, entre 5000 et 7000m, que se trouvent les cirrocumulus. L'altitude minimale des altocumulus est de 2000m, soit quatre fois moins que l'altitude maximale des cumulus.
- Ces pauvres cumulus ne sont pas favorisés en nom d'espèces, puisqu'ils se trouvent affligés de noms tels que *fractus*, *mediocris*, *humilis* et *congestus*... alors que les cumulonimbus ont des espèces aux noms plus... capillaires tels que *calvus*, *capillatus*. Les très gros cumulonimbus sont appelés *mammatus*.

Correction

```
<nuages>
<nuage>
<nom>
altocumulus
<espece>lenticularis</espece>
<espece>stratiformis</espece>
<espece>castellanus</espece>
<espece>flocus</espece>
</nom>
<altitude max="6000" min="2000"/>
<hydrometeores>Aucun.</hydrometeores>
</nuage>
<nuage>
<nom>
cirrus
<espece>flocus</espece>
<espece>castellanus</espece>
<espece>spissatus</espece>
<espece>uncinus</espece>
<espece>fibratus</espece>
</nom>
<altitude max="12000" min="6000"/>
<hydrometeores>Aucun.</hydrometeores>
</nuage>
<nuage>
<nom>
cirrocumulus
<espece>lenticularis</espece>
<espece>stratiformis</espece>
<espece>flocus</espece>
<espece>castellanus</espece>
</nom>
<altitude max="7000" min="5000"/>
<hydrometeores>Aucun.</hydrometeores>
</nuage>
<nuage>
<nom>
cumulus
<espece>fractus</espece>
<espece>humilis</espece>
<espece>mediocris</espece>
<espece>congestus</espece>
</nom>
<altitude max="8000" min="200"/>
<hydrometeores>Aucun en général.
    </hydrometeores>
</nuage>
<nuage>
<nom>
strato-cumulus
<espece>stratiformis</espece>
<espece>lenticularis</espece>
<espece>castellanus</espece>
</nom>
<altitude max="2000" min="0"/>
<hydrometeores>Aucun.</hydrometeores>
</nuage>
<nuage>
<nom>
cumulonimbus
<espece>calvus</espece>
<espece>capillatus</espece>
<espece>mammatus</espece>
</nom>
<altitude max="18000" min="300"/>
<hydrometeores>Averses (parfois de neige, de grésil
    ou de grêle). </hydrometeores>
</nuage>
</nuages>
```

Définir la structure d’un fichier XML avec une DTD

Le DTD

- Le DTD ou **Document Type Declaration** ou encore **Document Type Definition** est l'ensemble des règles et des propriétés que doit suivre le document XML.

- Ces règles définissent généralement le nom et le contenu de chaque balise et le contexte dans lequel elles doivent exister.

- Cette formalisation des éléments est particulièrement utile lorsqu'on utilise de façon récurrente des balises dans un document XML.

DTD interne, DTD externe

- On peut inclure son propre DTD au code source du fichier XML (*.xml). On parlera alors d'un DTD interne.
- DTD externe :DTD dans un autre fichier (*.dtd).

- Par les DTDs externes, plusieurs concepteurs peuvent se mettre d'accord pour utiliser un DTD commun pour échanger leurs données.
- DTD interne : document indépendant (standalone)

Document Valide et bien formé

- Distinguer un document "bien formé" d'un document valide.
- **Valide :** un document qui respecte les règles spécifiques de son DTD.
- **bien formé**: un document qui respecte les règles générales de syntaxe du XML.

Le DTD interne

- Au début du fichier XML
- Le DTD interne suit la syntaxe suivante :

<!DOCTYPE élément-racine [

déclaration des éléments

]>

Exemple de DTD interne

```
<?xml version="1.0"
   standalone="yes"?>
<!DOCTYPE parent [
<!ELEMENT parent (garçon,fille)>
<!ELEMENT garçon (#PCDATA)>
<!ELEMENT fille (#PCDATA)>
]>
<parent>
   <garçon>Loic</garçon>
   <fille>Marine</fille>
</parent>
```

Début du DTD interne avec parent comme élément de racine.

Cet élément racine soit parent contiendra les sous-éléments garcon et fille.

#PCDATA : l'élément garcon contient des données exprimées en chiffres ou en lettres. Idem pour l'élément fille.

Fin du DTD

Racine du document XML.

Fin du document XML.

Le DTD externe

Le DTD externe suivra la syntaxe suivante :

<!DOCTYPE élément-racine SYSTEM "nom_du_fichier.dtd">

Le même exemple:

Le fichier XML (*.xml) : les données + le nom du fichier de DTD (*.dtd)

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE parent SYSTEM "parent.dtd">
<parent>
    <garçon>Loic</garçon>
    <fille>Marine</fille>
</parent>
```

Le fichier de DTD externe (ici dans le même répertoire) "parent.dtd" :

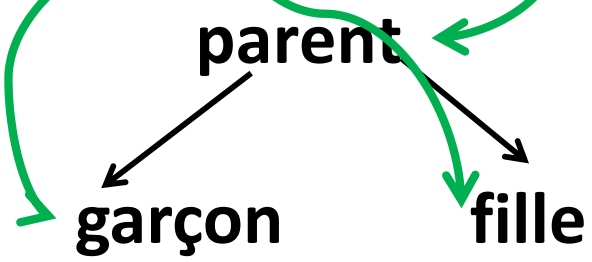
```
<!ELEMENT parent (garçon,fille)>
<!ELEMENT garçon (#PCDATA)>
<!ELEMENT fille (#PCDATA)>
```

Lien avec l’arborescence

<!ELEMENT parent (garçon,fille)>

<!ELEMENT garçon (#PCDATA)>

<!ELEMENT fille (#PCDATA)>



Déclarer un élément

- Pour pouvoir créer un document XML il est utile dans un premier temps de définir les éléments pouvant être utilisés dans le DTD.
- Ainsi pour définir un élément on utilisera la syntaxe suivante :

<! ELEMENT Nom_élément (type_ou_règle)>

Type prédéfini	Description
ANY	L'élément peut contenir des éléments fils (dans n'importe quel ordre d'une façon infinie), du texte ou peut être vide
EMPTY	L'élément doit obligatoirement être un élément vide
(#PCDATA)	L'élément doit contenir une chaîne de caractères (pas d'éléments fils)

Quelques exemples

- <!ELEMENT titre (#PCDATA)>
  - <titre> Le Rouge et le Noir </titre>

- <!ELEMENT titre EMPTY>
  - <titre …. />
- **<u>RQ</u>** Un élément vide peut posséder un ou plusieurs attributs:
  -

Règles sur les éléments

- Il est possible de définir des règles d'utilisation pour chaque élément, ex : les noms des éléments qu'un élément doit contenir, des règles sur la présence d'un élément

Opérateur	Signification	Exemple
+	L'élément doit être présent au minimum une fois (1 ou plusieurs fois)	**A+**
*****	L'élément peut être présent plusieurs fois (ou aucune) (0 ou plusieurs fois)	**A***
?	L'élément peut être optionnellement présent (0 ou 1 fois)	**A?**

Règles sur les éléments

- Autoriser un certain nombre de répétitions au niveau d'un élément.

Opérateur	Signification	Exemple
|	L'élément A ou l'élément B peuvent être présents (exclusivement)	**A|B**
Virgule ,	L'élément A doit être présent et suivi de l'élément B (dans l'ordre)	**A,B**
()	Les parenthèses permettent de regrouper des éléments afin de leur appliquer les autres opérateurs	**(A,B)+**

Quelques exemples

- <!ELEMENT livre (titre,auteur+)>
  - Un livre doit avoir un titre et un ou plusieurs auteurs.
- *<!ELEMENT livre((titre,auteur)|description)>*
  - Cette déclaration signifie qu'il est possible de saisir
    - soit la description d'un livre
    - soit son titre suivi de son auteur
- *<!ELEMENT citation (#PCDATA|auteur)*>*

<citation>

<auteur> Shakespeare </auteur> Etre ou ne pas être !

</citation>

Exemple de DTD

```
<!ELEMENT personne
  (nom,prenom,telephone,email?) >
<!ELEMENT nom (#PCDATA) >
<!ELEMENT prenom (#PCDATA) >
<!ELEMENT telephone (#PCDATA) >
<!ELEMENT email (#PCDATA) >
```

La racine du document XML est personne, elle contient obligatoirement les éléments nom, prenom, telephone qui sont tous des chaînes de caractères (#PCDATA) . La présence de l'élément email est optionnel.

XML correspondant

```
<personne>
    <nom>Pillou</nom>
    <prenom>Jean-Francois</prenom>
    <telephone>555-123456</telephone>
    <email>webmaster@commentcamarche.net </email>
</personne>
```

Ou bien

```
<personne>
    <nom>Pillou</nom>
    <prenom>Jeff</prenom>
    <telephone>555-542136</telephone>
</personne>
```

Déclarer des attributs

<! ATTLIST Elément Attribut Type >

Avec **Type** représente le type de données de l'attribut:

- **littéral**: il permet d'affecter une chaîne de caractères à un attribut. Pour déclarer un tel type il faut utiliser le mot clé ***CDATA***
- **l'énumération**: cela permet de définir une liste de valeurs possibles pour un attribut donné, afin de limiter le choix de l'utilisateur. La syntaxe de ce type d'attribut est :

<! ATTLIST Elément Attribut (Valeur1 | Valeur2 | ...) >

- Pour définir une valeur par défaut il suffit de faire suivre l'énumération par la valeur désirée entre guillemets :

<! ATTLIST Elément Attribut (Valeur1 | Valeur2) "valeur par défaut" >

- **atomique**: il permet de définir un identifiant unique pour chaque élément grâce au mot clé *ID*.

Exemples

- <!ATTLIST rectangle longueur CDATA "0">
- <!ATTLIST rectangle largeur CDATA "0">
- <!ATTLIST personne situation (célibataire | mariée | divorcée) "célibataire">

- <!ELEMENT img EMPTY>
- <!ATTLIST img format (jpg| png| gif) "png">

Nécessité de l'attribut

- Enfin chacun de ces types d'attributs peut être suivi d'un mot clé particulier permettant de spécifier le niveau de nécessité de l'attribut :
- **#IMPLIED** : l'attribut est optionnel, non obligatoire
- **#REQUIRED** : l'attribut est obligatoire
- **#FIXED "val":** l'attribut sera affecté d'une valeur par défaut s'il n'est pas défini. Il doit être immédiatement suivi de la valeur entre guillemets
- **"val":** la valeur par défaut de l'attribut

Exemple

- <!ATTLIST personne nom CDATA #REQUIRED>
- <!ATTLIST personne age CDATA #IMPLIED>
- <!ATTLIST date année CDATA #FIXED "2004">
- <!ATTLIST personne situation (célibataire|mariée|divorcée) #IMPLIED>
- <!ATTLIST machine état CDATA "neuf">

- <!ELEMENT elt (elt1,elt2, …)>
- <!ATTLIST elt

  attri1 CDATA #IMPLIED

  attri2 CDATA #REQUIRED

>

Exemple

<! ATTLIST disque IDdisk ID #REQUIRED type(K7|MiniDisc|Vinyl|CD)"CD" >

- L'élément ***disque*** a deux attributs ***IDdisk*** et ***type***.
- Le premier attribut ***IDdisk*** est de type atomique, il s'agit d'un identifiant unique *obligatoire*.
- L‘attribut ***type*** peut être soit *K7*, *MiniDisc*, *Vinyl* ou *CD*, sachant que ce dernier sera affecté par défaut...

Exercice :1

- Ecrire la DTD
-
- Une bouteille d'eau Cristaline de 150 cl contient par litre 71 mg d'ions positifs calcium, et 5,5 mg d'ions positifs magnésium. On y trouve également des ions négatifs comme des chlorures à 20 mg par litre et des nitrates avec 1 mg par litre. Elle est recueillie à **St-Cyr la Source**, dans le département du Loiret. Son code barre est 3274080005003 et son pH est de 7,45. Comme la bouteille est sale, quelques autres matériaux comme du fer s'y trouvent en suspension.
-
  Une seconde bouteille d'eau Cristaline a été, elle, recueillie à la source d'**Aurèle** dans les Alpes Maritimes. La concentration en ions calcium est de 98 mg/l, et en ions magnésium de 4 mg/l. Il y a 3,6 mg/l d'ions chlorure et 2 mg/l de nitrates, pour un pH de 7,4. Le code barre de cette bouteille de 50 cl est 3268840001008.
  Une bouteille de même contenance est de marque Volvic, et a été puisée à... **Volvic**, bien connu pour ses sources donnant un pH neutre de 7. Elle comprend 11,5 mg/l d'ions calcium, 8,0 mg/l d'ions magnésium, 13,5 mg/l d'ions chlorures et 6,3 mg/l d'ions nitrates. Elle contient également des particules de silice. Son code barre est 3057640117008.
-
  PS : Volvic est dans le Puy-de-Dôme...

En se basant sur la solution en XML

```
<cave>
<bouteille>
<marque>Cristaline</marque>
<composition>
<ion type="positif">calcium 71mg/l</ion>
<ion type="positif">magnésium 5,5mg/l</ion>
<ion type="negatif">chlorure 20mg/l</ion>
<ion type="negatif">nitrate 1mg/l</ion>
<autre type="metal">fer</autre>
</composition>
<source>
<ville>St-Cyr la Source</ville>
<departement>Loiret</departement>
</source>
<code_barre>3274080005003</code_barre>
<contenance unit="cl">150</contenance>
<ph>7,45</ph>
</bouteille>
<bouteille>
<marque>Cristaline</marque>
<composition>
<ion type="positif">calcium 98mg/l</ion>
<ion type="positif">magnésium 4mg/l</ion>
<ion type="negatif">chlorure 3,6mg/l</ion>
<ion type="negatif">nitrate 2mg/l</ion>
</composition>
<source>
<ville>Aurèle</ville>
<departement>Alpes Maritimes</departement>
</source>
<code_barre>3268840001008</code_barre>
<contenance unit="cl">50</contenance>
<ph>7,4</ph>
</bouteille>
<bouteille>
<marque>Volvic</marque>
<composition>
<ion type="positif">calcium 11,5mg/l</ion>
<ion type="positif">magnésium 8mg/l</ion>
<ion type="negatif">chlorure 13,5mg/l</ion>
<ion type="negatif">nitrate 6,3mg/l</ion>
</composition>
<source>
<ville>Volvic</ville>
<departement>Puy-de-Dôme</departement>
</source>
<code_barre>3057640117008</code_barre>
<contenance unit="cl">50</contenance>
<ph>7</ph>
</bouteille>
</cave>
```

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE cave [
<!ELEMENT cave (bouteille)*>
<!ELEMENT bouteille (marque,composition,source,code_barre,contenance,ph)>
<!ELEMENT marque (#PCDATA)>
<!ELEMENT composition (ion*,autre?)>
<!ELEMENT source (ville,departement)>
<!ELEMENT code_barre (#PCDATA)>
<!ELEMENT contenance (#PCDATA)>
<!ELEMENT ph (#PCDATA)>
<!ELEMENT ion (#PCDATA)>
<!ATTLIST ion type (positif|negatif) "positif">
<!ELEMENT autre (#PCDATA)>
<!ATTLIST autre type (CDATA) "metal">
<!ELEMENT ville (#PCDATA)>
<!ELEMENT departement (#PCDATA)>
]>
```

Exercice 2 :

- Ecrire la DTD :
- Il existe diverses variétés de nuages. La plupart de ceux dont nous allons parler ne produit aucun "hydrométéore", sauf le cumulonimbus, qui est accompagné d'averses (parfois sous la forme de neige, de grésil ou de grêle).
- L'altocumulus et le cirrocumulus partagent les mêmes "espèces" : *lenticularis*, *stratiformis*, *castellanus* et *flocus*.
- On retrouve ces deux espèces également chez le cirrus, ainsi que les espèces *spissatus*, *uncinus* et *fibratus*. Les espèces *stratiformis*, *lenticularis* et *castellanus* sont quant à elles partagées également avec les strato-cumulus.
- Ces derniers peuvent se traîner au ras du sol et monter à 2000m, mais certains nuages ont une altitude minimale à peine plus élevée, puisqu'elle n'est que de 200m pour les cumulus, et de 300m pour les cumulonimbus. Il est vrai que ces derniers compensent en montant jusqu'à une altitude maximale de 18000m, soit plus haut encore que les cirrus, qui plafonnent à 12000m. L'altitude minimale de ces derniers coïncide avec la fin de la présence possible des altocumulus, à 6000m. Et c'est autour de cette zone, entre 5000 et 7000m, que se trouvent les cirrocumulus. L'altitude minimale des altocumulus est de 2000m, soit quatre fois moins que l'altitude maximale des cumulus.
- Ces pauvres cumulus ne sont pas favorisés en nom d'espèces, puisqu'ils se trouvent affligés de noms tels que *fractus*, *mediocris*, *humilis* et *congestus*... alors que les cumulonimbus ont des espèces aux noms plus... capillaires tels que *calvus*, *capillatus*. Les très gros cumulonimbus sont appelés *mammatus*.

En se basant sur la solution en XML

```
<nuages>
<nuage>
<nom>
altocumulus
<espece>lenticularis</espece>
<espece>stratiformis</espece>
<espece>castellanus</espece>
<espece>flocus</espece>
</nom>
<altitude max="6000" min="2000"/>
<hydrometeores>Aucun.</hydrometeores>
</nuage>
<nuage>
<nom>
cirrus
<espece>flocus</espece>
<espece>castellanus</espece>
<espece>spissatus</espece>
<espece>uncinus</espece>
<espece>fibratus</espece>
</nom>
<altitude max="12000" min="6000"/>
<hydrometeores>Aucun.</hydrometeores>
</nuage>
<nuage>
<nom>
cirrocumulus
<espece>lenticularis</espece>
<espece>stratiformis</espece>
<espece>flocus</espece>
<espece>castellanus</espece>
</nom>
<altitude max="7000" min="5000"/>
<hydrometeores>Aucun.</hydrometeores>
</nuage>
<nuage>
<nom>
cumulus
<espece>fractus</espece>
<espece>humilis</espece>
<espece>mediocris</espece>
<espece>congestus</espece>
</nom>
<altitude max="8000" min="200"/>
<hydrometeores>Aucun en général.
    </hydrometeores>
</nuage>
<nuage>
<nom>
strato-cumulus
<espece>stratiformis</espece>
<espece>lenticularis</espece>
<espece>castellanus</espece>
</nom>
<altitude max="2000" min="0"/>
<hydrometeores>Aucun.</hydrometeores>
</nuage>
<nuage>
<nom>
cumulonimbus
<espece>calvus</espece>
<espece>capillatus</espece>
<espece>mammatus</espece>
</nom>
<altitude max="18000" min="300"/>
<hydrometeores>Averses (parfois de neige, de grésil
    ou de grêle). </hydrometeores>
</nuage>
</nuages>
```

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE nuages [
<!ELEMENT nuages (nuage)*>
<!ELEMENT nuage (nom,espece+,altitude,hydrometheore)>
<!ELEMENT nom (#PCDATA)>
<!ELEMENT espece (#PCDATA)>
<!ELEMENT altitude (EMPTY)>
<!ELEMENT hydrometheore (#PCDATA)>
<!ATTLIST altitude max (CDATA) "0"
                                                    min (CDATA) "0">
]>
```

Exercice :3

- Rédiger une DTD pour une bibliographie. Cette bibliographie :
- contient des livres et des articles ;
- les informations nécessaires pour un livre sont :
  - son titre général ;
  - les noms des auteurs ;
  - ses tomes et pour chaque tome, leur nombre de pages ;
  - des informations générales sur son édition comme par exemple le nom de l'éditeur, le lieu d'édition, le lieu d'impression, son numéro ISBN ;
- les informations nécessaires pour un article sont :
  - son titre ;
  - les noms des auteurs ;
  - ses références de publication : nom du journal, numéro des pages, année de publication et numéro du journal
- on réservera aussi un champ optionnel pour un avis personnel.

Correction

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
 <!ELEMENT biblio (livre|article)*>
<!ELEMENT livre (titre, auteur+, tome*, edition, avis?)>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT auteur (#PCDATA)>
<!ELEMENT tome (nb_pages)>
<!ELEMENT nb_pages (#PCDATA)>
<!ELEMENT edition (editeur, lieu_edition, lieu_impression, isbn)>
                    <!ELEMENT editeur (#PCDATA)>
                    <!ELEMENT lieu_edition (#PCDATA)>
                    <!ELEMENT lieu_impression (#PCDATA)>
                    <!ELEMENT isbn (#PCDATA)>
                    <!ELEMENT avis (#PCDATA)>
<!ELEMENT article (titre, auteur+, journal)>
<!ELEMENT journal (nom_journal, page, num_journal, annee)>
                    <!ELEMENT nom_journal (#PCDATA)>
                    <!ELEMENT page (#PCDATA)>
                    <!ELEMENT num_journal (#PCDATA)>
                    <!ELEMENT annee (#PCDATA)>
```

Exercice 4 :

- Modifier la DTD précédente...
- ... en ajoutant un attribut optionnel soustitre à l'élément titre ;
- ... en faisant de l'élément tome un élément vide et en lui ajoutant un attribut requis nb_pages et un attribut optionnel soustitre ;
- ... en faisant de l'élément nom_journal un attribut de l'élément journal et en lui donnant comme valeur par défaut Feuille de Chou ;
- ... en faisant de l'élément annee un attribut de type énuméré, prenant comme valeurs possibles 2000, 2001, 2002, "avant_2000" et "inconnue" et proposant comme valeur par défaut inconnue.
- Utiliser cette DTD pour créer un fichier XML valide.

Correction

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
 <!ELEMENT biblio (livre|article)*>
<!ELEMENT livre (titre, auteur+, tome*, edition, avis?)>
<!ATTLIST titre soustitre CDATA #IMPLIED >
<!ELEMENT titre (#PCDATA)>
<!ELEMENT auteur (#PCDATA)>
<!ELEMENT tome EMPTY>
<!ATTLIST tome nb_pages CDATA #REQUIRED soustitre CDATA #IMPLIED          >

<!ELEMENT edition (editeur, lieu_edition, lieu_impression, isbn)>

<!ELEMENT editeur (#PCDATA)>
<!ELEMENT lieu_edition (#PCDATA)>
<!ELEMENT lieu_impression (#PCDATA)>
<!ELEMENT isbn (#PCDATA)>
<!ELEMENT avis (#PCDATA)>
<!ELEMENT article (titre, auteur+, journal)>
<!ELEMENT journal (page, num_journal)>
<!ATTLIST journal nom_journal CDATA "Feuille de Chou "  annee (2000 | 2001 | 2002 | avant_2000 | inconnue) "inconnue">
<!ELEMENT page (#PCDATA)>
<!ELEMENT num_journal (#PCDATA)>
<!ELEMENT annee (#PCDATA)>
```

Afficher le XML avec CSS

Le CSS

- Pour afficher les balises XML, on peut faire appel aux bonnes vieilles feuilles de style (CSS), maintenant classiques dans le paysage Html.
- A chaque balise "inventée" dans le fichier XML, on va définir un élément de style que le navigateur pourra alors afficher.

Un exemple de XML

Voici notre document XML de départ :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<racine>
<enfant>
<nom>Loïc</nom>
<lien>garçon</lien>
<date>07/11/83</date>
<data>Le petit qui me dépasse d'une tête.</data>
</enfant>
<enfant>
<nom>Marine</nom>
<lien>fille</lien>
<date>20/12/85</date>
<data>La petite fille chérie à son papa.</data>
</enfant>
</racine>
```

Résultat sans CSS

A:\simple.xml - Microsoft Internet Explorer

Fichier Edition Affichage Favoris Outils ?

```
  <?xml version="1.0" encoding="ISO-8859-1" ?>
- <racine>
  - <enfant>
      <nom>Loïc</nom>
      <lien>garçon</lien>
      <date>07/11/83</date>
      <data>Le petit qui me dépasse d'une tête.</data>
    </enfant>
  - <enfant>
      <nom>Marine</nom>
      <lien>fille</lien>
      <date>20/12/85</date>
      <data>La petite fille chérie à son papa.</data>
    </enfant>
  </racine>
```

Fichier CSS

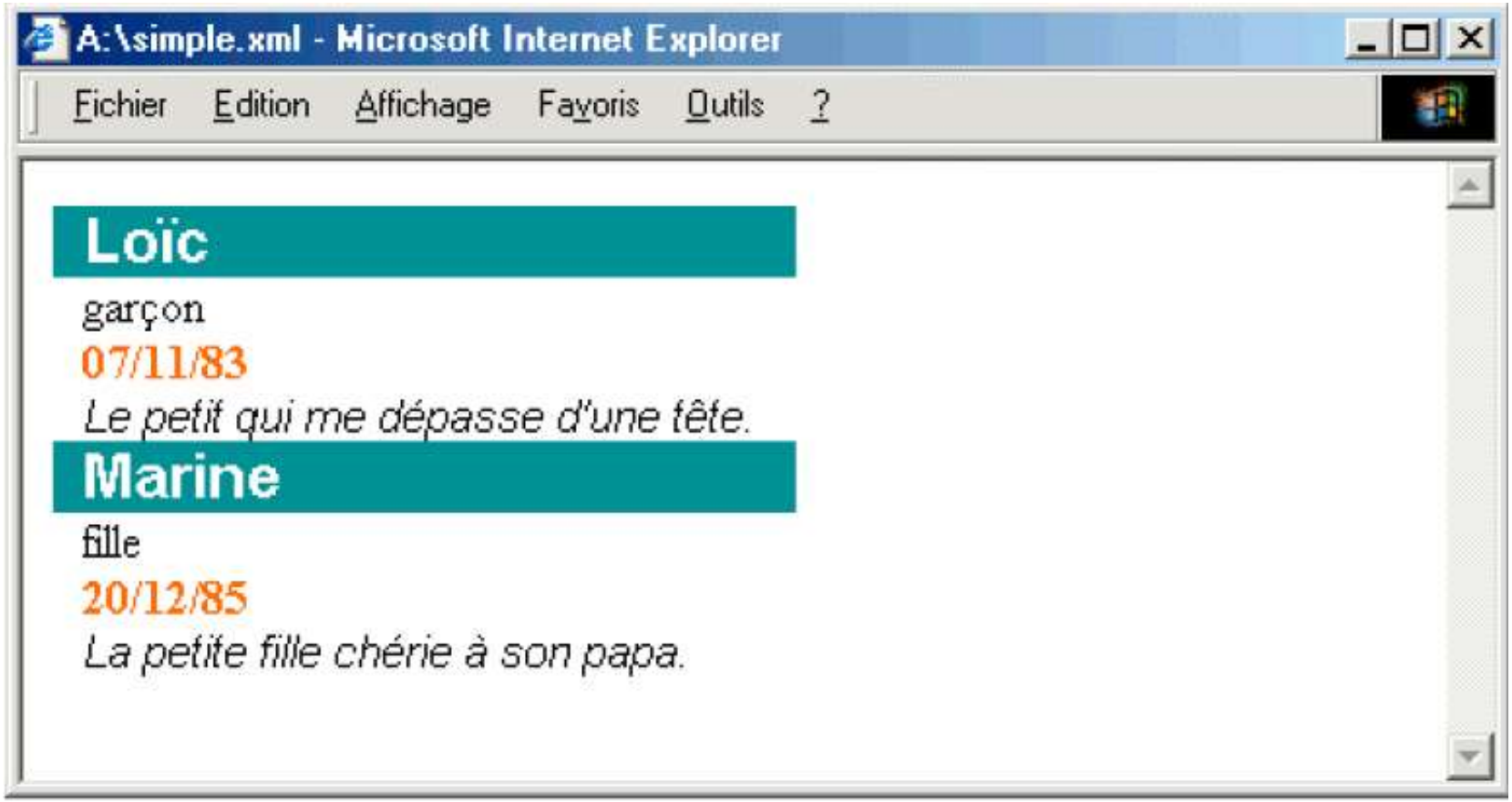
```
<style type="text/css">
racine , enfant {}
nom {
display: block;
width: 250px;
font-size: 16pt ;
font-family: arial ;
font-weight: bold;
background-color: teal;
color: white;
padding-left: 10px;
} lien {
display: block;
font-size: 12pt;
padding-left: 10px;
}
date {
display: block;
font-size: 12pt;
color: red ;
font-weight: bold;
padding-left: 10px;
}
data {
display: block;
font-size: 11pt ;
font-style: italic;
font-family: arial ;
padding-left: 10px;
}
</style>
```

Lien entre le fichier CSS et le XML

- Dans l’entête du fichier xml, Il faut ajouter un lien vers le fichier css :

```
<?xml-stylesheet href="css.css"
type="text/css"?>
```

Résultat avec CSS



Afficher le XML avec XSL

Le XSL - Les feuilles de style du XML

- Comme le XML n'utilise pas des balises prédéfinies (car on peut inventer ses propres balises), le navigateur ne "comprend" pas les balises du XML et ne sais pas trop comment afficher un document XML.
- Néanmoins, pour afficher des documents XML, il est nécessaire d'avoir un mécanisme pour décrire comment le document pourrait être affiché. Un de ces mécanismes est les feuilles de style classiques du Html (CSS),
- **XSL pour eXtensible Stylesheet Language est de loin un langage de feuille de style plus adapté au XML et donc plus performant.**

Exemple XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<racine>
<enfant>
<nom>Loïc</nom>
<lien>garçon</lien>
<date>07/11/83</date>
<data>Le petit qui me dépasse d'une
tête.</data>
</enfant>
<enfant>
<nom>Marine</nom>
<lien>fille</lien>
<date>20/12/85</date>
<data>La petite fille chérie à son papa.</data>
</enfant>
</racine>
```

Fichier XSL

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
    <head>
      <title>Exemple de sortie HTML</title>
      <meta http-equiv="Content-Type" content="text/html;
    charset=ISO-8859-1" />
    </head>
<body> (voir la diapo suivante!!) </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

```
<body style="font-family:Arial; font-size:12pt;">
<xsl:for-each select="racine/enfant">
<div style="background-color:teal; color:white;">
<span style="font-weight:bold; color:white; padding:4px">
<xsl:value-of select="nom"/>
</span>
</div ><div style="margin-left:20px; font-size:10pt">
<xsl:value-of select="lien"/> <br/>
<span style="color: red ; " > <xsl:value-of select="date"/>
</span > <br/>
<span style="font-style:italic"> - <xsl:value-of select="data"/>
</span > </div >
</xsl:for-each>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Lien entre le fichier XSL et le XML

- Dans l’entête du fichier xml , Il faut ajouter un lien vers le fichier xsl:

```
<?xml-stylesheet type="text/xsl"
  href="test.xsl"?>
```

Résultat avec XSL

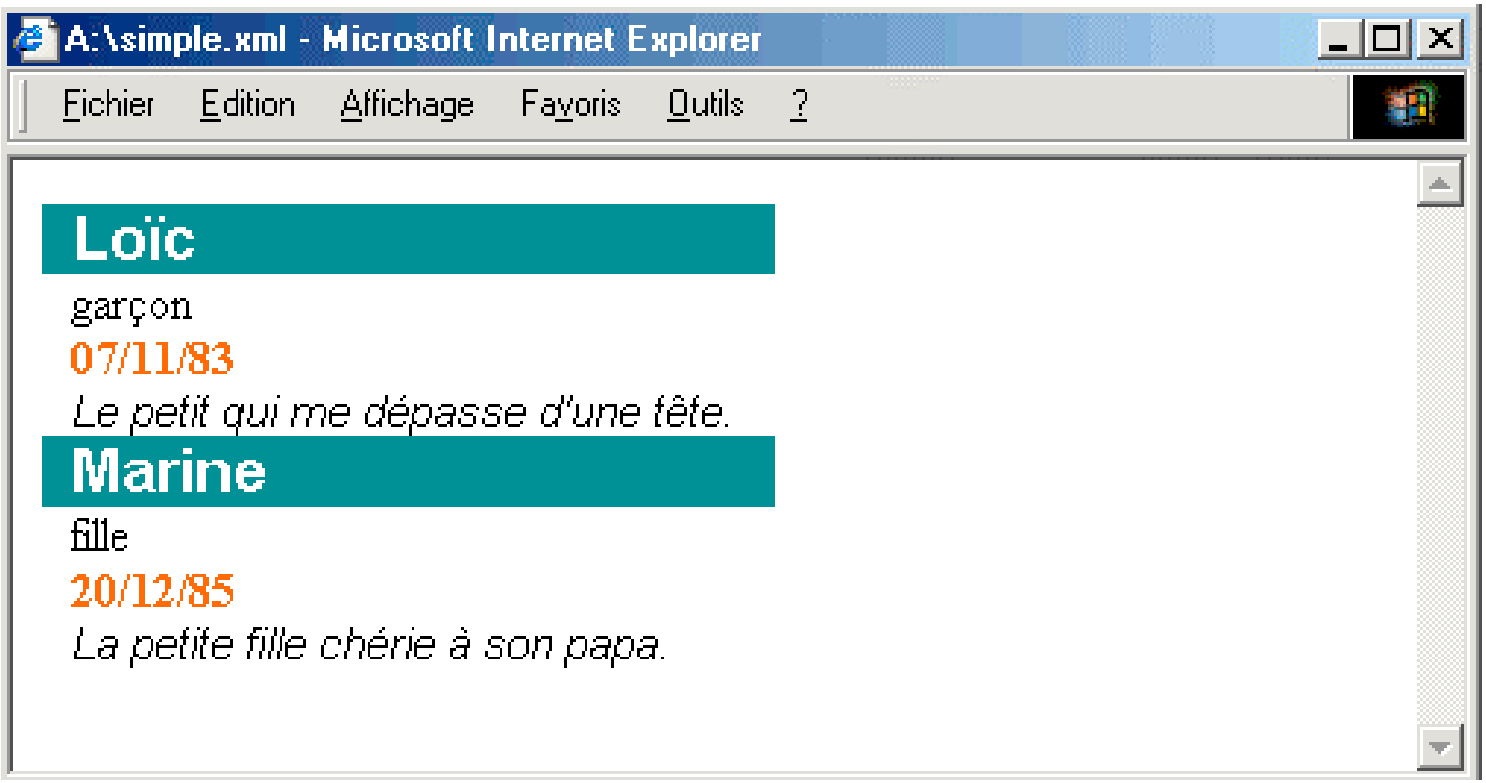


Schéma XML - XSD

Limitations des DTD

- Syntaxe particulière
  - les DTD ne sont pas au format XML.
- Type de données limités : PCDATA
  - Comment imposer qu'un élément soit un entier entre 0 et 100 ?
- Pas de définition précise du nombre d'occurrence
  - Constructeurs (, et |) et indicateurs (*, + et ?) ne suffisent pas
- Pas de support des "*espaces de nom". En pratique, cela implique qu'il n'est pas possible, dans un fichier XML défini par une DTD, d'importer des définitions de balises définies par ailleurs.*

Définition

- XSD : XML Schema Definition
- Définition d'une classe de définition de documents XML.
  - Éléments et leurs attributs,
  - Imbrication des éléments,
  - Ordre d'apparition des éléments,
  - Et plus
- Un document XSD est un document XML.
- Le vocabulaire de XML Schema est composé d'environ 30 éléments et attributs

Définition

- Un schéma d'un document définit:
  - les éléments possibles dans le document
  - les attributs associés à ces éléments
  - la structure du document et les types de données
- Le schéma est spécifié en XML
  - pas de nouveau langage
  - balisage de déclaration
  - utilise un espace de nom xsd: (ou xs:)
- Présente de nombreux avantages
  - types de données personnalisés
  - extensibilité par héritage et ouverture
  - analysable par un parseur XML standard

Objectifs des schémas

- Reprendre les acquis des DTD
  - Plus riche et complet que les DTD
- Permettre de typer les données
  - Eléments simples et complexes
  - Attributs simples
- Permettre de définir plus de contraintes
  - Existence obligatoire ou optionnelle
  - Domaines de valeurs, cardinalités, références
  - Patterns, ...

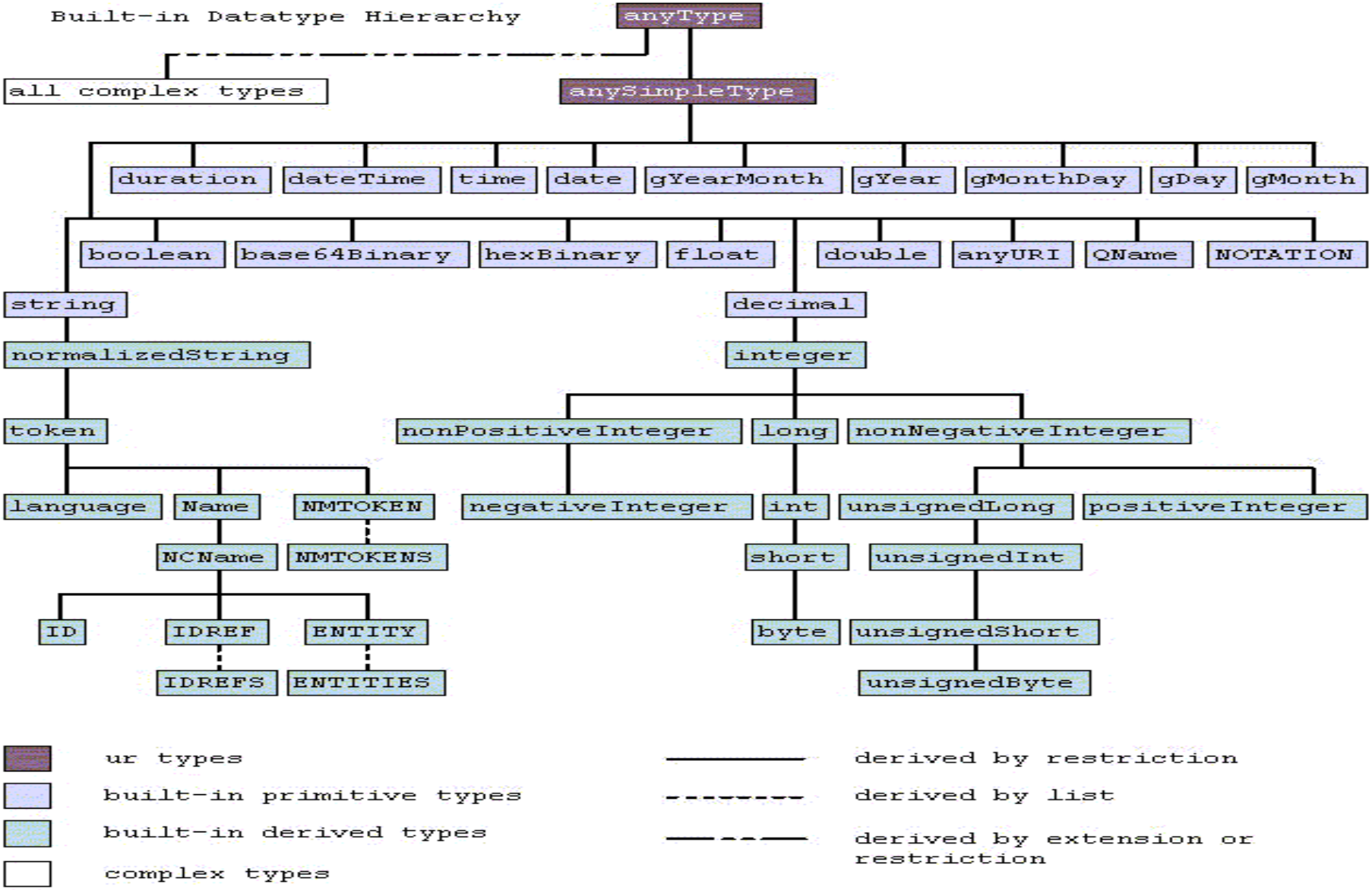
Les types XML

- La base d'un schéma XML: l'élément

<xsd:element name="…" type="…"/>

- nom: permet de nommer l'élément
- type: donne le type de l'élément, il peut être :
  - Simple si sa valeur a un type prédéfini en XML-SCHEMA (xsd:string, xsd:int, xsd:decimal, xsd:double…) ou une extension de ces types
  - Complexe s'il contient des sous éléments ou s'il comporte un attribut
    - xsd:all tous les éléments doivent exister (peu importe l'ordre)
    - xsd:choice un des éléments doit exister
    - xsd:sequence tous les éléments doivent exister dans l'ordre spécifié

Les types XML



Les types simples

- string

Chaine de Caractères

- byte

-1 - >126

- integer

-126789 – >126789

- positiveInteger

1 -> 126789

- negativeInteger

-126789 -> -1

- hexBinary

0FB7

- int

-1 -> 126789675

- unsignedInt

0 - > 1267896754

- boolean

true, false 1, 0

- date

1999-05-31

- dateTime

1999-05-31T13:20:00.000-05:00

- ID

"A212"

- IDREF

"A212«

- IDREFS

"A212" "B213"

- anyURI

http://www.example.com/e1.html#5

- language

en-GB, en-US, fr

- Et beaucoup d'autres

Short, long, float

Déclaration d'élément

- Un élément, dans un schéma, se déclare avec la balise <xsd:element>. Par exemple

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
<xsd:element name="contacts" type="typeContacts"> </xsd:element>
<xsd:element name="remarque" type="xsd:string"></xsd:element>
</xsd:schema>
```

- **contacts** est du type **typeContacts**, type complexe
- **Remarque** du type **String**, type simple
- Chaque élément déclaré est associé à un type de données via l'attribut **type**.
- Les éléments pouvant contenir des élément-enfants ou possède des attributs sont dits de type *complexe*,

Les attributs 1/3

- Les éléments à contenu simple avec attributs

```
<contacts maj="25-11-2007">Ali …</contacts>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
<xsd:element name="contacts" type="typeContacts" />
  <xsd:element name="remarque" type="xsd:string">
    <xsd:complexType name="typeContacts">
    <!-- déclarations du modèle de contenu ici -->
      <xsd:attribute name="maj" type="xsd:date" />
    </xsd:complexType>
</xsd:element>
</xsd:schema>
```

Les attributs 2/3

- Les éléments à contenu complexe avec attributs

```
<traduction langue="allemand" dateTraduction="2003-12-01">
        <traducteur>Michael</traducteur>
</traduction>
```

```
<xsd:element name="traduction" minOccurs="0" maxOccurs="unbounded">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="traducteur" type="xsd:string" minOccurs="1"
maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="langue" use="required" type="xsd:string"/>
<xsd:attribute name="dateTraduction" use="optional" type="xsd:date"/>
</xsd:complexType>
</xsd:element>
```

Les attributs 3/3

- L'élément attribute d'un Schema XML peut avoir trois attributs optionnels : **use**, **default** et **fixed**.
- Par exemple, la ligne suivante permet de rendre l'attribut **maj** optionnel, avec une valeur par défaut au 11 octobre 2003

```
<xsd:attribute name="maj" type="xsd:date" use="optional" default="2003-10-11" />
```

DTD	Attribut use	Attribut default	Commentaire
#REQUIRED	required	-	
"tt" #REQUIRED	required	tt	
#IMPLIED	optional	-	
"tt" #IMPLIED	optional	tt	
-	prohibited	-	Cet attribut ne doit pas apparaître

Les occurrences

- Une bibliothèque contient au moins un livre

```
<xsd:element name="biblio">
<xsd:complexType>
<xsd:sequence>
<xsd:element ref="livre" minOccurs="1" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
```

Les types complexes

- Définition d'objets complexes
  - <xsd:sequence> : collection ordonnée d'éléments typés
  - <xsd:all> : collection non ordonnée d'éléments typés
  - <xsd:choice>: choix entre éléments typés
- Exemple

```
<xsd:complexType name="Addresse">
<xsd:sequence>
<xsd:element name="nom" type="xsd:string"/>
<xsd:element name="rue" type="xsd:string"/>
<xsd:element name="ville" type="xsd:string"/>
<xsd:element name="codep" type="xsd:decimal"/>
</xsd:sequence>
<xsd:attribute name="pays" type="xsd:string" fixed="FR"/>
</xsd:complexType>
```

Séquences d'éléments

- Dans une DTD, nous pouvons déclarer un élément comme pouvant contenir une suite de sous-éléments dans un ordre déterminé
- On utilise l'élément xsd:sequence, qui reproduit l'opérateur « **,** » du langage DTD

```
<xsd:complexType>
  <xsd:sequence>
    <xsd:element name="nom" type="xsd:string" />
    <xsd:element name="prénom" type="xsd:string" />
    <xsd:element name="dateDeNaissance" type="xsd:date" />
    <xsd:element name="adresse" type="xsd:string" />
    <xsd:element name="adresseElectronique" type="xsd:string" />
    <xsd:element name="téléphone" type="numDeTéléphone" />
  </xsd:sequence>
</xsd:complexType>
```

Choix d'élément

- Si on veut indiquer soit l'adresse d'une personne, soit son adresse électronique. il suffit d'utiliser un élément xsd:choice
- Ce connecteur a donc les mêmes effets que l'opérateur « **|** » dans une DTD.

```
<xsd:complexType name="typePersonne">
  <xsd:sequence>
    <xsd:element name="nom" type="xsd:string" />
    <xsd:element name="prénom" type="xsd:string" />
    <xsd:element name="dateDeNaissance" type="xsd:date" />
    <xsd:choice>
      <xsd:element name="adresse" type="xsd:string" />
      <xsd:element name="adresseElectronique" type="xsd:string" />
    </xsd:choice>
  </xsd:sequence>
  <xsd:element name="téléphone" type="numDeTéléphone" />
</xsd:complexType>
```

L'élément all

- Cet élément est une **nouveauté par rapport aux DTD**. Il indique que les éléments enfants doivent apparaître une fois (ou pas du tout), et dans n'importe quel ordre.

```
<xsd:complexType>
  <xsd:all>
    <xsd:element name="nom" type="xsd:string" />
    <xsd:element name="prénom" type="xsd:string" />
    <xsd:element name="dateDeNaissance" type="xsd:date" />
    <xsd:element name="adresse" type="xsd:string" />
    <xsd:element name="adresseElectronique" type="xsd:string" />
    <xsd:element name="téléphone" type="numDeTéléphone" />
  </xsd:all>
</xsd:complexType>
```

Héritage de types

- Définition de sous-types par héritage
  - Par extension : ajout d'informations
  - Par restriction : ajout de contraintes
- Possibilité de contraindre la dérivation
- Exemple **d'extension** :

```
<xsd:complexType name="AdressePays">
<xsd:complexContent>
<xsd:extension base="Adresse">
<xsd:sequence>
<xsd:element name="pays"  type="xsd:string"/>
</xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
```

Création de type complexe à partir de types simples

- Il est possible également de créer un type complexe à partir d'un type simple.
- On peut avoir besoin de définir un élément contenant une valeur simple, et possédant un attribut.

<poids unite="kg">67</poids>

- Un tel élément ne peut pas être déclaré de type simple, car il contient un attribut. Il faut *dériver* un type complexe à partir du type simple positiveInteger

```
<xsd:complexType name="typePoids">
  <xsd:simpleContent>
    <xsd:extension base="xsd:positiveInteger">
      <xsd:attribute name="unite" type="xsd:string" />
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

Les patterns

- Contraintes sur type simple prédéfini
- Utilisation d'expression régulières
  - Similaires à celles de Perl
- Exemple de restriction

```
<xsd:simpleType name="num5">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="\d{5}"/>
    </xsd:restriction>
</xsd:simpleType>
```

- Autres facettes de restriction

xsd:mininclusive, xsd:maxexclusive, xsd:enumeration, xsd:length…

Les facettes de restriction

- lenght : la longueur d'une donnée.
- minLenght: la longueur minimum.
- maxLenght: la longueur maximum.
- pattern: défini par une expression régulière.
- enumeration: un ensemble discret de valeurs.
- maxInclusive: une valeur max comprise.
- maxExclusive: une valeur max exclue.
- minInclusive: une valeur min comprise.
- minExclusive: une valeur min exclue.
- totalDigits: le nombre total de chiffres.
- fractionDigits: le nombre de chiffres dans la partie fractionnaire.
- …

Réutilisation de types

- Type simple avec extension

```
<xsd:simpleType name="num5">
        <xsd:restriction base="xsd:string">
                <xsd:pattern value="\d{5}"/>
        </xsd:restriction>
</xsd:simpleType>
```

- Type complexe (séquence)

```
<xsd:element name="livre">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="Titre" type="xsd:string"/>
<xsd:element name="Auteur" type="xsd:string"/>
<xsd:element name="ISBN" type="num5"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
```

Restriction de type

```
<xsd:simpleType name="monEntier">
  <xsd:restriction base="nonNegativeInteger">
    <xsd:maxExclusive value="100" />
  </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:simpleType name= "ChiffresOctaux">
            <xsd:restriction base="xsd:integer">
                        <xsd:minInclusive value="0" />
                        <xsd:maxInclusive value= 7" />
            </xsd:restriction>
</xsd:simpleType>
```

Restriction de type

```
<xsd:attribute name="jour" type="typeJourSemaine"
use="required" />
<xsd:simpleType name="typeJourSemaine">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="lundi" />
    <xsd:enumeration value="mardi" />
    <xsd:enumeration value="mercredi" />
    <xsd:enumeration value="jeudi" />
    <xsd:enumeration value="vendredi" />
    <xsd:enumeration value="samedi" />
    <xsd:enumeration value="dimanche" />
  </xsd:restriction>
</xsd:simpleType>
```

Restriction de type

```
<xsd:simpleType name="typeMotLangueFrancaise">
  <xsd:restriction base="xsd:string">
          <xsd:length value="21" />
  </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:simpleType name="typeAdresseElectronique">
  <xsd:restriction base="xsd:string">
          <xsd:pattern value="(.)+@(.)+" />
  </xsd:restriction>
</xsd:simpleType>
```

Groupage d'éléments

```
<xsd:group name="TitreAuteurISBN">
    <xsd:sequence>
        <xsd:element name="Titre" type="xsd:string"/>
        <xsd:element name="Auteur" type="xsd:string"/>
        <xsd:element name="ISBN" type="num5"/>
    </xsd:sequence>
</xsd:group>
<xsd:element name="livre">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:group ref="TitreAuteurISBN"/>
            …
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

```
<xsd:complexType name="Items">                                  <!– Explicit complexType →
   <xsd:sequence>
      <xsd:element name="item" minOccurs="0" maxOccurs="unbounded">
         <xsd:complexType>                                      <!-- Implicit complexType -->
            <xsd:sequence>
               <xsd:element name="productName" type="xsd:string"/>
               <xsd:element name="quantity">
                  <xsd:simpleType>                              <!-- Implicit simpleType -->
                     <xsd:restriction base="xsd:positiveInteger">
                        <xsd:maxExclusive value="100"/>
                     </xsd:restriction>
                  </xsd:simpleType>
               </xsd:element>
               <xsd:element name="USPrice" type="xsd:decimal"/>
               <xsd:element ref="comment" minOccurs="0"/>
               <xsd:element name="shipDate" type="xsd:date" minOccurs="0"/>
            </xsd:sequence>
            <xsd:attribute name="partNum" type="SKU"/>
         </xsd:complexType>
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>
```

Exercice :1

- Rédiger un Schema XML pour une bibliographie. Cette bibliographie :
- contient des livres et des articles ;
- les informations nécessaires pour un livre (élément livre) sont :
  - son titre général (élément titre) ;
  - les noms des auteurs (éléments auteur) ;
  - ses tomes (élément tomes) et pour chaque tome (éléments tome), leur nombre de pages (élément pages) ;
  - des informations générales sur son édition (élément infosEdition)comme par exemple le nom de l'éditeur (élément editeur), le lieu d'édition (élément lieuEdition), le lieu d'impression (élément lieuImpression), son numéro ISBN (élément ISBN) ;
- les informations nécessaires pour un article (élément article) sont :
  - son titre (élément titre) ;
  - les noms des auteurs (éléments auteur) ;
  - ses références de publication (élément infosPublication) : nom du journal (élément nomJournal), numéro des pages (élément pages), année de publication (élément anneePublication) et numéro du journal (élément numéroJournal)
- on réservera aussi un champ optionnel, pour chaque livre et ahcque article, pour un avis (élément avis) personnel.

Correction

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <xsd:schema
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
- <!--
 Déclarations des éléments de type simple   -->
  <xsd:element name="titre" type="xsd:string" />
  <xsd:element name="auteur" type="xsd:string" />
  <xsd:element name="pages" type="xsd:string" />
  <xsd:element name="editeur" type="xsd:string" />
  <xsd:element name="lieuEdition" type="xsd:string" />
  <xsd:element name="lieuImpression" type="xsd:string" />
  <xsd:element name="ISBN" type="xsd:string" />
  <xsd:element name="nomJournal" type="xsd:string" />
  <xsd:element name="anneePublication" type="xsd:string" />
  <xsd:element name="numéroJournal" type="xsd:string" />
  <xsd:element name="avis" type="xsd:string" />
- <!--
 Déclarations des éléments de type complexe   -->
- <xsd:element name="infosEdition">
- <xsd:complexType>
- <xsd:sequence>
  <xsd:element ref="editeur" />
  <xsd:element ref="lieuEdition" />
  <xsd:element ref="lieuImpression" />
  <xsd:element ref="ISBN" />
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
- <xsd:element name="tome">
- <xsd:complexType>
- <xsd:sequence>
  <xsd:element ref="pages" />
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
- <xsd:element name="tomes">
- <xsd:complexType>
- <xsd:sequence>
  <xsd:element ref="tome" minOccurs="1"
      maxOccurs="unbounded" />
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
- <xsd:element name="livre">
- <xsd:complexType>
- <xsd:sequence>
  <xsd:element ref="titre" />
  <xsd:sequence id="auteur" maxOccurs="unbounded" />
  <xsd:element ref="tomes" />
  <xsd:element ref="infosEdition" />
  <xsd:element ref="avis" minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
- <xsd:element name="infosPublication">
- <xsd:complexType>
- <xsd:sequence>
  <xsd:element ref="nomJournal" />
  <xsd:element ref="pages" />
  <xsd:element ref="anneePublication" />
  <xsd:element ref="numéroJournal" />
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
```

```
-
- <xsd:element name="article">
- <xsd:complexType>
- <xsd:sequence>
  <xsd:element ref="titre" />
  <xsd:element ref="auteur" minOccurs="1"
        maxOccurs="unbounded" />
  <xsd:element ref="infosPublication" />
  <xsd:element ref="avis" minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
- <xsd:element name="biblio">
- <xsd:complexType>
- <xsd:choice minOccurs="0" maxOccurs="unbounded">
  <xsd:element ref="livre" />
  <xsd:element ref="article" />
  </xsd:choice>
  </xsd:complexType>
  </xsd:element>
  </xsd:schema>
```

Exercice 2 :

Modifier le Schéma précédent... On ne déclarera, pour le moment, que des types de chaînes de caractères.

- ... en ajoutant un attribut optionnel soustitre à l'élément titre ;
- ... en faisant de l'élément tome un élément vide et en lui ajoutant un attribut requis nbPages et un attribut optionnel sousTitre ;
- ... en faisant de l'élément nomJournal un attribut de l'élément infosPublication et en lui donnant comme valeur par défaut Feuille de Chou ;
- Utiliser ce Schéma pour créer un fichier XML valide.

Correction

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
- <!--
 Déclarations des éléments de type simple   -->
  <xsd:element name="auteur" type="xsd:string" />
  <xsd:element name="pages" type="xsd:string" />
  <xsd:element name="editeur" type="xsd:string" />
  <xsd:element name="lieuEdition" type="xsd:string" />
  <xsd:element name="lieuImpression" type="xsd:string" />
  <xsd:element name="ISBN" type="xsd:string" />
  <xsd:element name="anneePublication" type="xsd:string" />
  <xsd:element name="numéroJournal" type="xsd:string" />
  <xsd:element name="avis" type="xsd:string" />
- <!--
 Déclarations des éléments de type complexe   -->
- <xsd:element name="titre">
- <xsd:complexType mixed="true">
  <xsd:attribute name="sousTitre" type="xsd:string" use="optional" />
  </xsd:complexType>
  </xsd:element>
- <xsd:element name="infosEdition">
- <xsd:complexType>
- <xsd:sequence>
  <xsd:element ref="editeur" />
  <xsd:element ref="lieuEdition" />
  <xsd:element ref="lieuImpression" />
  <xsd:element ref="ISBN" />
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
- <xsd:element name="tome">
- <xsd:complexType>
  <xsd:attribute name="nbPages" type="xsd:string" use="required" />
  <xsd:attribute name="sousTitre" type="xsd:string" use="optional" />
  </xsd:complexType>
  </xsd:element>
- <xsd:element name="tomes">
- <xsd:complexType>
- <xsd:sequence>
  <xsd:element ref="tome" minOccurs="1" maxOccurs="unbounded" />
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
- <xsd:element name="livre">
- <xsd:complexType>
- <xsd:sequence>
  <xsd:element ref="titre" />
  <xsd:sequence id="auteur" maxOccurs="unbounded" />
  <xsd:element ref="tomes" />
  <xsd:element ref="infosEdition" />
  <xsd:element ref="avis" minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
- <xsd:element name="infosPublication">
- <xsd:complexType>
- <xsd:sequence>
  <xsd:element ref="pages" />
  <xsd:element ref="anneePublication" />
  <xsd:element ref="numéroJournal" />
  </xsd:sequence>
  <xsd:attribute name="nomJournal" type="xsd:string" use="optional"
      default="Feuille de Chou" />
  </xsd:complexType>
  </xsd:element>
```

```
- <xsd:element name="article">
- <xsd:complexType>
- <xsd:sequence>
  <xsd:element ref="titre" />
  <xsd:element ref="auteur" minOccurs="1" maxOccurs="unbounded" />
  <xsd:element ref="infosPublication" />
  <xsd:element ref="avis" minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
  </xsd:complexType>
  </xsd:element>
- <xsd:element name="biblio">
- <xsd:complexType>
- <xsd:choice minOccurs="0" maxOccurs="unbounded">
  <xsd:element ref="livre" />
  <xsd:element ref="article" />
  </xsd:choice>
  </xsd:complexType>
  </xsd:element>
  </xsd:schema>
```

SOAP

Simple Object Access Protocol *Version 1.2*

Les points forts de SOAP

- SOAP est un protocole de transmission de messages.
- Indépendant de la plateforme (windows, unix, mac, …)
- Définit un ensemble de règles pour structurer des messages principalement pour exécuter des dialogues requête-réponse de type RPC (Remote Procedural Call)
- SOAP n'est pas lié à un protocole de transport donné, mais il est souvent porté sur HTTP.
- Un message SOAP est un document XML ayant la forme suivante:
  - Une déclaration XML (optionnelle)
  - Une enveloppe SOAP (élément racine) composée de:
    - Une entête SOAP (Header) .
    - Un corps SOAP (body).Basé sur XML et les namespaces.

Principes de SOAP

- Permet d'envoyer des messages XML entre deux machines.
- Les messages sont «emballés» dans une enveloppe SOAP
  - L'enveloppe SOAP utilise un XML schéma prédéfini
  - Le schéma du message dépend de l'application

Exemple : requête SOAP

```
POST /InStock HTTP/1.1
Host: www.stock.org
Content-Type: application/soap; charset=utf-8

<?xml version="1.0">
<soap:Envelope xmlns:soap=http://www.w3.org/2001/12/soap-envelope
   soap:encodingStyle=http://www.w3.org/2001/12/soap-encoding>
   <soap:Body xmlns:m=http://www.stock.org/stock>
         <m:GetStockPrice>
                  <m:StockName>IBM</m:StockName>
         </m:GetStockPrice>
   </soap:Body>
</soap:Envelope>
```

Propre au portage sur HTTP

Exemple : réponse SOAP

```
HTTP/1.1 200 OK
Connection: close
Content-Type: application/soap; charset=utf-8
Date: Mon, 28 Sep 2002 10:05:04 GMT
```

Propre au portage sur HTTP

```
<?xml version="1.0">
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-encoding"
    soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
    <soap:Body xmlns:m="http://www.stock.org/stock">
            <m:GetStockPricesResponse>
                    <m:Price>34.5</m:price>
            </m:GetStockPricesResponse>
    </soap:Body>
</soap:Envelope>
```

Analyse de l’exemple

- Des Balises Utilisateur
  - GetStockResponse
  - StockName
  - Price
- Un Namespace Utilisateur
  - xmlns:m="Some-URI"
- Des Balises SOAP
  - Enveloppe
  - Body
- Un Namespace SOAP
  - xmlns:SOAP="http://w3.org/2001/12/soap-envelope"
- Des informations dans la partie HTTP

Namespace SOAP

- Le namespace des balises SOAP
  - http://w3.org/2001/12/soap-envelope
- Le namespace de l’encodage SOAP
  - http://w3.org/2001/12/soap-encoding

Structure d’un message SOAP

- Un message SOAP est contenu dans une balise **Envelope**
  - Une **Envelope** peut contenir une balise **Header**
    - Une Header peut contenir n’importe quel ensemble de balises. Ces balises doivent appartenir à des namespaces.
  - Une **Envelope** doit contenir une balise **Body**
    - Un Body peut contenir n’importe quelle ensemble de balises. Ces balises peuvent appartenir à des namespaces.
    - Un Body peut contenir des balises **Fault** qui permettent d’identifier des erreurs.

SOAP Header

Mécanisme d’extension du protocol SOAP

- La balise Header est optionnelle
- Si la balise Header est présente, elle doit être le premier fils de la balise Envelope
- La balise Header contient des *entrées*
- Une *entrée* est n’importe quelle balise incluse dans un namespace

SOAP Header Example

```
<SOAP:Header>
  <t:Transaction xmlns:t="some-URI">
      5
  </t:Transaction>
</SOAP:Header>
```

SOAP Body

Le Body contient le message à échanger

- La balise Body est obligatoire
- La balise Body doit être le premier fils de la balise Envelope (ou le deuxième si il existe une balise Header)
- La balise Body contient des *entrées*
- Une *entrée* est n'importe quelle balise incluse optionnellement dans un namespace
- Une *entrée* peut être une Fault.

SOAP Fault

Balise permettant de signaler des cas d'erreur.

- La balise Fault contient les balises suivantes
  - faultcode : un code permettant d'identifier le type d'erreur
    - Client, Server, VersionMismatch, MustUnderstand
  - faultstring : une explication en langage naturel
  - faultactor : une information pour identifier l'initiateur de l'erreur
  - detail : Définition précise de l'erreur.

Encodage

- Un message SOAP contient des données typées. Il faut donc définir un moyen d'encoder ces données.
- Vocabulaire SOAP :
  - Value (valeur d'une donnée)
    - Simple value (string, integers,etc)
    - Compound value (array, struct, …)
  - Type (d'une value)
    - Simple Type
    - Compound Type

Encodage

- L'encodage c'est la représentation de valeurs sous forme XML.
- Le décodage c'est la construction de valeurs à partir d'XML
- L'XML qui représente les valeurs a une structure qui dépend du type des valeurs
- Il faut donc définir le type
  - Soit mécanisme définit par l'utilisateur
  - Soit utilisation d'XML Schéma (préconisé)

Simple Types

- Type (XML Schema)

```
<element name="age" type="int"/>
<element name="color">
    <simpleType base="xsd:string">
        <enumeration value="Green"/>
        <enumeration value="Blue"/>
    </simpleType>
</element>
```

Type XML Schema

Construction de Type XML Schema

- Valeurs

```
<age>45</age>
<color>Blue</color>
```

Simple Types

- La définition d’un XML Schéma pour tout type peut être fastidieux
- SOAP a défini deux façons de préciser le type d’une valeur sans définir le Schéma XML:
  - <SOAP-ENC:int>45</SOAP-ENC:int>
  - <cost xsi:type="xsd:float">29.5</cost>

Compound Types

- Une structure est un type composé dans lequel les membres sont accessibles uniquement grâce à des noms différents.
- Un tableau est un type composé dans lequel les membres sont accessibles uniquement grâce à leur position.

Struct

- Type (XML Schéma)

```
<element name="Person">
<complexType>
    <element name="name" type="xsd:string"/>
    <element name="age" type="xsd:int"/>
</complexType>
<element>
```

- Valeur

```
<Person>
    <name>Xavier</name>
    <age>28</age>
</Person>
```

Array

- Le type est directement précisé grâce aux balises SOAP:

```
<myFavoriteNumbers SOAP-ENC:arrayType="xsd:int[2] ">
  <SOAP-ENC:int>3</SOAP-ENC:int>
  <SOAP-ENC:int>4</SOAP-ENC:int>
</myFavoriteNumbers>
```

SOAP avec HTTP

- SOAP peut être facilement porté sur Http.
  - Convient au mode Request/Response de Http
  - Le message SOAP est mis dans une requête POST avec un content-type text/xml

SOAP avec HTTP

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

<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="Some-URI">
            <symbol>DIS</symbol>
        </m:GetLastTradePrice>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP avec HTTP

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<SOAP-ENV:Envelope
      xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"/>
    <SOAP-ENV:Body>
        <SOAP-ENV:Fault>
            <faultcode>SOAP-ENV:Server</faultcode>
            <faultstring>Server Error</faultstring>
        </SOAP-ENV:Fault>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

WSDL

Web Services Description Language

Version 2.0

Présentation

Une description WSDL :

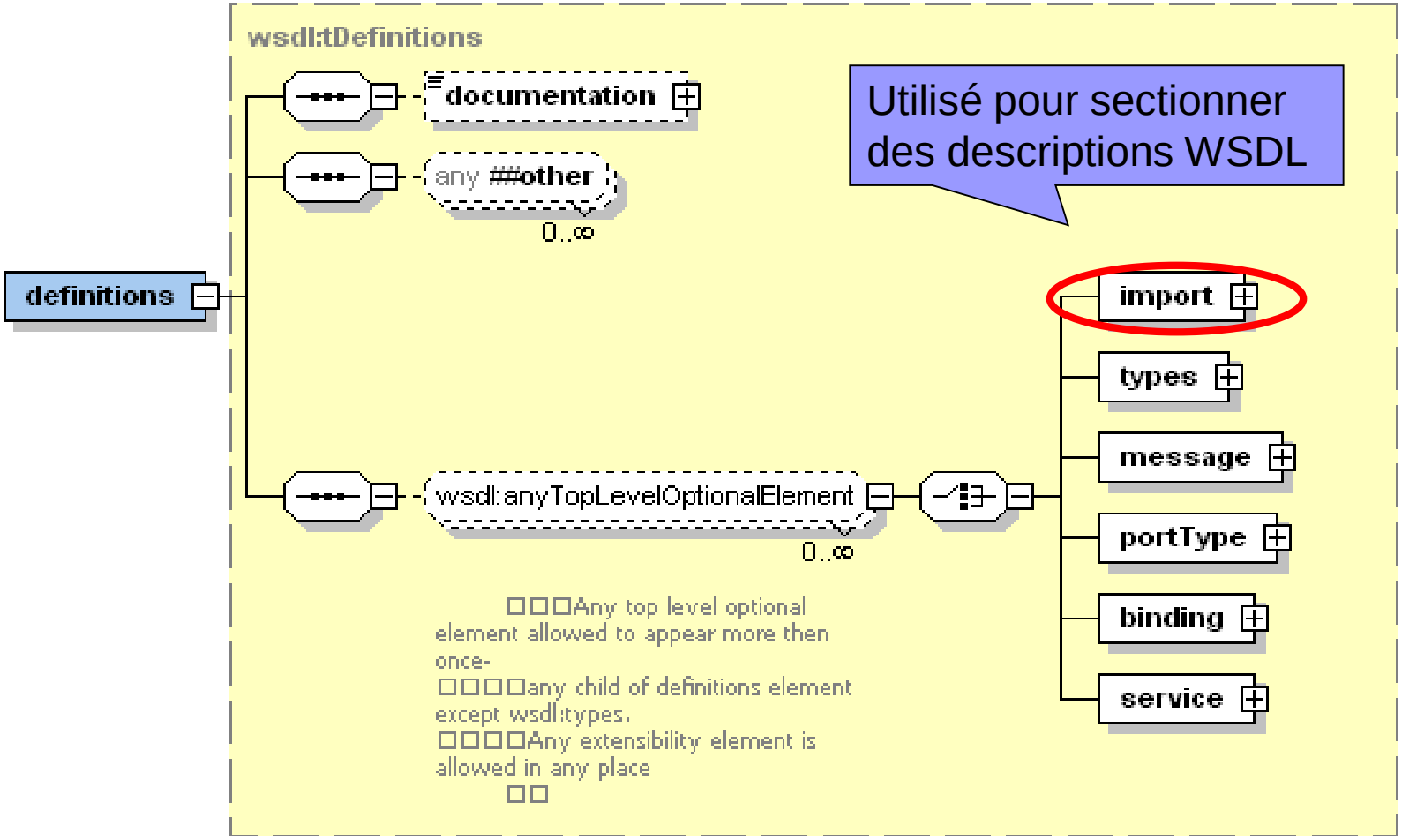
1. Décrit le type d'un service web (méthodes, types des paramètres)

2. Décrit les aspects techniques d'implantation d'un service web (quel est le protocole utilisé, quel est l'adresse du service)
   Cette description sert à se connecter concrètement à un service web.

Balises

- 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

Balises (avec XML Spy)



types

- Cette balise décrit les types utilisés
- Description en utilisant XML Schema.

```
<wsdl:types>
        <xs:schema
                targetNameSpace="http://www.exemple.fr/personne.xsd"
                xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:element name="personne">
                <xs:complexType>
                  <xs:sequence>
                        <xs:element name="nom" type="xs:string" />
                        <xs:element name="prenom" type="xs:string" />
                  </xs:sequence>
                </xs:complexType>
          </xs:element>
        </xs:schema>
</wsdl:types>
```

message

- Les messages sont envoyés entre deux interlocuteurs (ex: une opération reçoit des message et envoie des messages.
- Un message est composé de plusieurs **part**
- Deux façons de définir des part
  - Soit un **part** est un élément de type simple
  - Soit un **part** est un élément XML dont le type est défini dans un XML Schema

message

- Part de type simple

```
<wsdl:message name="personneMsg">
    <wsdl:part name="nom" type="xsd:string" />
    <wsdl:part name="prenom" type="xsd:string" />
</wsdl:message>
```

- Part qui utilise un XML Schema

```
<wsdl:message name="personneMsg">
    <wsdl:part name="personne" element="exemple:personne" />
</wsdl:message>
```

Défini dans un XML Schema

portType

- Un portType permet d’identifier (nommer) de manière abstraite un ensemble d’opérations.

```
<wsdl:portType name="descriptionPersonnes" >
        <wsdl:operation name="getPersonne" >
                …
        </wsdl:operation>
        <wsdl:operation name=“setPersonne" >
                …
        </wsdl:operation>
</wsdl:portType>
```

operation

- WSDL définit 4 types d'opérations :
  - 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

operation

- Quelque soit le type d'opération la définition est sensiblement la même :
- Une opération :
  - Reçoit des messages : <wsdl:input …>
  - Envoie des messages : <wsdl:output …> ou <wsdl:fault …>
- La présence et l'ordre des input/outputs/fault dépendent du type de l'opération.

operation

```
<wsdl:operation name="operation_name">
    <wsdl:input name="nom_optionel" message="nom_message" />
</wsdl:operation>

<wsdl:operation name="operation_name">
    <wsdl:output name="nom_optionel" message="nom_message" />
    <wsdl:input name="nom_optionel" message="nom_message" />
    <wsdl:fault name="nom_optionel" message="nom_message" />*
</wsdl:operation>

<wsdl:operation name="operation_name">
    <wsdl:input name="nom_optionel" message="nom_message" />
    <wsdl:output name="nom_optionel" message="nom_message" />
    <wsdl:fault name="nom_optionel" message="nom_message" />*
</wsdl:operation>
```

binding

- WSDL permet de lier une description abstraite (portType) à un protocole.
- Chacune des opérations d'un portType pourra être liée de manière différente.
- Le protocole SOAP est un des protocole qui peut être utilisé.
- D'autres binding sont standardisés par WSDL : HTTP et MIME.

binding

- Un Binding :
  - peut être identifié par un nom : **name**
  - identifie le portType : **type**

<wsdl:binding **name**="*binding_name*"
type="*nom du portType*" >
…
</wsdl:binding>

binding SOAP

- Pour préciser que le binding est de type SOAP, il faut inclure la balise suivante :

<soap:binding transport="*uri*" style="*soap_style*" />

- Transport définit le type de transport
  - http://schemas.xmlsoap.org/soap/http pour utiliser SOAP/HTTP
- Style définit la façon dont sont crée les messages SOAP de toutes les opérations
  - rpc : Encodage RPC défini par SOAP RPC
  - document : Encodage sous forme d'élément XML

binding SOAP

- Pour chaque opération du portType :
  - il faut préciser l'URI de l'opération : soapAction
  - Il est aussi possible de repréciser la façon dont sont créés les messages SOAP : style
- Pour chaque message de chaque opération, il faut définir comment sera créé le message SOAP

Exemple

```
<wsdl:binding type="descriptionPersonnes" >
    <soap:binding
            transport="http://schemas.xmlsoap.org/soap/http"
            style="rpc" />
    <wsdl:operation name="getPersonne">
    <soap:operation  soapAction="http://www.exemple.fr/getPersonne" />
            <wsdl:input>
                        <soap:body use="encoded"
                          encodingStyle="schemas.xmlsoap.org/soap/encoding"/>
             </wsdl:input>
             <wsdl:output>
                        <soap:body use="encoded"
                          encodingStyle="schemas.xmlsoap.org/soap/encoding"/>
             </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
```

service

- Un service est un ensemble de ports
- Un port a un portType
- Dans le cadre de SOAP, un port à une adresse (qui correspond à l’adresse http)

```
<wsdl:service name=“MonService">
    <wsdl:port    binding="intf:MonServiceSoapBinding">
        <soap:address
        location="http://mda.lip6.fr:8080/axis/services/MonService"/>
    </wsdl:port>
</wsdl:service>
```

UDDI

Universal Description Discovery and Integration

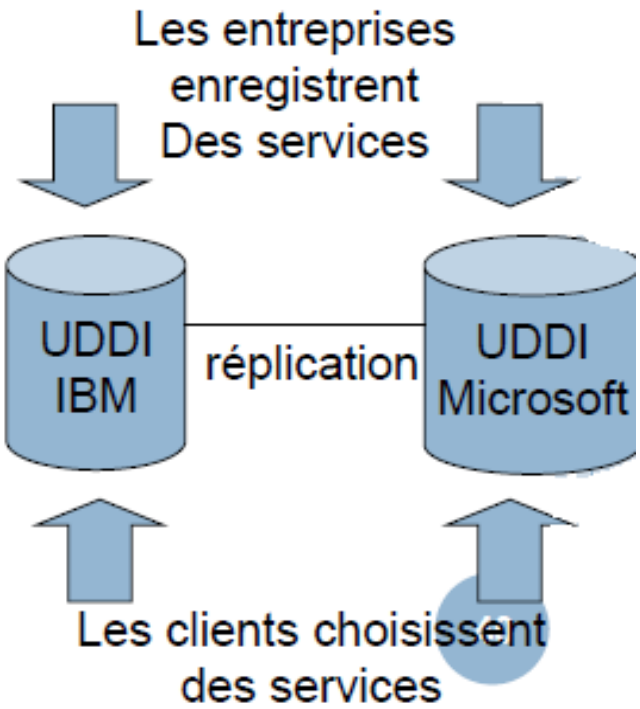
Version 3.0

UDDI : Universal Description, Discovery and Integration (of services)

- Où trouver le service dont j'ai besoin ?
  - Quels sont les fournisseurs potentiels ?
  - Lequel est le meilleur pour moi ?
- Noyau: annuaires –les pages jaunes
  - Liste d'entreprises + comment les contacter
  - Classification en catégories
  - Informations en +: protocole, coût, qualité, contrat…
- Question cruciale : Qui contrôle l'annuaire ?
  - Par exemple: qui contrôle les catégories ? Qui peut s'enregistrer dans l'annuaire ?

UDDI (2)

- Consortium industriel (IBM, Microsoft,…)
- + de 200 entreprises.
- Moyens de publier et de rechercher
- Beaucoup de bruit
- Limité pour l'instant
  - Peu de services
  - Langage d'interrogation primitif
  - Informations très limitées



Rôles

- Un référentiel UDDI joue 3 rôles :
  - Pages blanches : le référentiel comporte des informations sur les fournisseurs de services.
  - Pages Jaunes : le référentiel comporte des critères de catégorisation de services.
  - Pages vertes : le référentiel comporte des informations techniques (WSDL).
- Les services d'un référentiel UDDI sont des Web Services !

Exemple

- Le référentiel UDDI de microsoft est accessible à http://uddi.microsoft.com
- Il est possible de parcourir ce référentiel à l'aide d'un navigateur pour :
  - Rechercher un service.
  - Ajouter un service au référentiel.

Exemple : search

Façons de rechercher un service.

Browse by Category | Services | Providers | tModels

Type one or more initial characters in the name of the provider you want to find. If you do not know the exact name, use % as a wildcard character. You can further refine your search by specifying categories, identifiers, or tModels.

Provider Name:

Amazon

Search

Nous allons rechercher les Web Services de la société Amazon.

	Actions
	Add Category

Identifiers	Actions
	Add Identifier

tModels	Actions
	Add tModel

Exemple : search

Référentiels

- Type
  - Public :
    - Microsoft : uddi.microsoft.com
    - IBM : www.ibm.com/services/uddi
    - HP : uddi.hp.com
    - SAP : udditest.sap.com
  - Privé ou d’entreprise
- Accès
  - Défini en WSDL
  - JAXR définit une API pour naviguer dans un référentiel UDDI

A vous de jouer

- Quel est la place d'UDDI dans les Web Services ?
- Comparer les référentiels UDDI avec les moteurs de recherche style Yahoo et Google ?
- Quel est l'intérêt des référentiels UDDI d'entreprise ?

Sécurité des services web

- Principales fonctions :
  - **Confidentialité :** s’assurer que seules les personnes autorisées aient accès aux ressources échangées
  - **Authentification :** s’assurer que seules les personnes autorisées aient accès aux ressources
  - **Intégrité des messages :** déterminer si les données n'ont pas été altérées (modifiées) durant la communication
  - **Non-répudiation :** garantir qu'aucun des correspondants ne pourra nier la transaction
  - **Disponibilité :** maintenir le bon fonctionnement du système d'information

Sécurité des services web

- HTTP fournit un mécanisme d'authentification très simple
- SSL: Secure Socket Layer; un protocole pour transmettre des données encryptées
- HTTPS = HTTP over SSL: très utilisé
- XML digital signature non-répudiation
- XML encryption
- SSL encrypte le message en entier; problème des intermédiaires.
- XML encryption permet d'encrypter de manière sélective

Exemple en .NET avec Visual Studio 2010

Exemple en java avec NETBEANS

Exemple client en VB.NET avec service développé en java

Exemple client WEB en ASP.NET et VB.NET avec service en VB.NET + BD MYSQL

Exemple client développé avec java pour Symbian

Exemple de client développé avec C# pour Windows mobile 7.1

Mini-Projet (par groupes de 3)

- Conception orientée services (3 ing.) (1 jour)
- Conception de la base de données (3 ing.) (1 jour)
- Un développeur pour les services (3 jours)
- Un développeur pour les clients (web) (3 jours)
- Un développeur pour les clients (interfaces) (3 jours)
- Intégration (3 ing.) (1 jours)
- Développement d'un exemple de service composite (3 ing.) (1 jours)

Référence [http://tahe.developpez.com/dotnet/exercices/]

Description du projet

- Nous allons développer une application de gestion des RDV dans un cabinet médical. Nous appellerons [RdvMedecins] l'application. Nous présentons ci-dessous des copies d'écran de ce que pourrait être son fonctionnement avec un client web par exemple.

- La page d'accueil de l'application est la suivante :

Cabinet médical - LES MEDECINS ASSOCIES -

Prise de rendez-vous :

Médecin: Mme Marie PELISSIER

Jour (JJ/MM/AAAA): 23/08/2006

Valider | Effacer

L'utilisateur a sélectionné un médecin et a saisi un jour de RV

Cabinet médical - LES MEDECINS ASSOCIES -

Accueil

Rendez-vous de Mme Marie PELISSIER le 23/08/2006

Créneau horaire	Client	Action
8h0-8h20		Réserver
8h20-8h40		Réserver
8h40-9h0		Réserver
9h0-9h20		Réserver
9h20-9h40		Réserver
9h40-10h0		Réserver
10h0-10h20	Mr Jules JACQUARD	Supprimer
10h20-10h40		Réserver
10h40-11h0		Réserver
11h0-11h20		Réserver
11h20-11h40	Mme Christine GERMAN	Supprimer

On obtient la liste (vue partielle ici) des RV du médecin sélectionné pour le jour indiqué.

- A partir de cette première page, l'utilisateur (Secrétariat, Médecin) va engager un certain nombre d'actions. Nous les présentons ci-dessous La vue de gauche présente la vue à partir de laquelle l'utilisateur fait une **demande, la vue de droite la réponse envoyée** par le serveur.

Accueil

Rendez-vous de Mme Marie PELISSIER le 23/08/2006

Créneau horaire	Client	Action
8h0-8h20		Réserver
8h20-8h40		Réserver
8h40-9h0		Réserver

On réserve

Cabinet médical - LES MEDECINS ASSOCIES -

Rendez-vous : 8h0-8h20, Jour : 2006:08:23, Médecin : Mme Marie PELISSIER

Client Mr Jules MARTIN

Valider Annuler

On obtient une page à renseigner

Cabinet médical - LES MEDECINS ASSOCIES -

Rendez-vous : 8h0-8h20, Jour : 2006:08:23, Médecin : Mme Marie PELISSIER

Client Melle Brigitte BISTROU

Valider Annuler

On la renseigne

Cabinet médical - LES MEDECINS ASSOCIES -

Accueil

Rendez-vous de Mme Marie PELISSIER le 23/08/2006

Créneau horaire	Client	Action
8h0-8h20	Melle Brigitte BISTROU	Supprimer
8h20-8h40		Réserver
8h40-9h0		Réserver

Le nouveau RV apparaît dans la liste

Notation par groupe

- Quelle est la technologie utilisée pour les services (1 pts)
- Quelle est la technologie utilisée pour les interfaces (3 pts)
- Quelle conception pour l'application (2 pts)
- Quelle sécurité pour l'accès aux services (3 pts)
- Tests des méthodes du service (2 pts)
- Fonctionnement de l'application (2 pts)
- Déploiement des services sous un serveur (2 pts)
- Répartition des tâches dans le groupe (1 pts)
- Rapport de conception 10 pages (2 pts)
- Rapport d'utilisation 10 pages (2 pts)

Plan

1. Architectures SOA
   - Définitions
   - Caractéristiques de SOA
   - Architectures XSOA
2. Services Web
   - Définitions et Principes
   - XML (Rappel et Notions)
   - SOAP
   - WSDL
   - UDDI
   - Exemples
3. Cloud Computing
   - Grid computing
   - Différents types de Cloud
     - IAAS
     - PAAS
     - SAAS

PARTIE 3 :
Cloud Computing

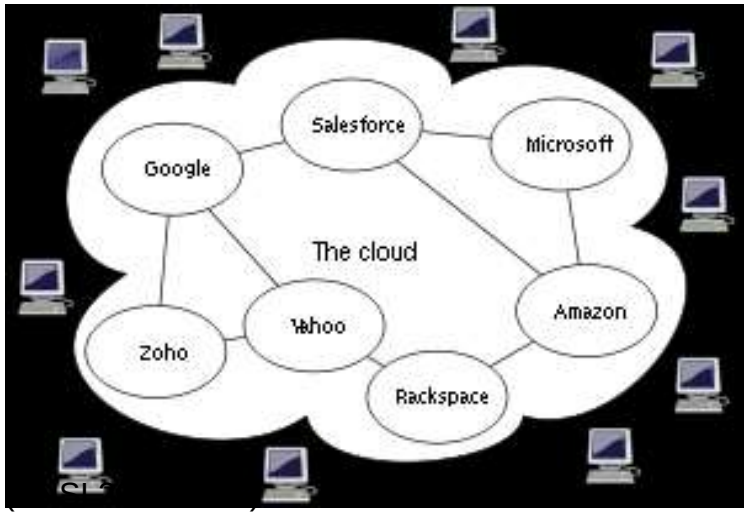
Grid Computing

Types de Cloud : IaaS - Infrastructure as a Service

Types de Cloud : PaaS - Platform as a Service

Pour aller plus loin !!

- SaaS est un modèle de « cloud computing », parmi d'autres
- Le **Cloud computing** est un concept de déportation sur des serveurs distants des traitements informatiques traditionnellement localisés sur le poste utilisateur.
- Ce concept est aussi désigné par « informatique dans le nuage », « informatique en nuage », « informatique dématérialisée », ou encore « infonuagique ».



Types de Cloud : SaaS - Software as a Service

Définition

- Un concept consistant à proposer un abonnement à un logiciel plutôt que l'achat d'une licence.
- Les clients ne paient pas pour posséder le logiciel en lui-même mais plutôt pour l’utiliser.
- Avec le développement des TIC (Technologies de l'information et de la communication), de plus en plus d'offres **SaaS** se font au travers du web

Définition

- La plupart des experts s'accorde à définir **SaaS** comme un modèle économique qui consisterait à commercialiser un **logiciel** *non pas sous la forme d'un produit* (en licence définitive), que le client installerait en interne sur ses serveurs, mais en tant qu'application accessible à distance comme un **service**, par le biais d'**Internet** et du **Web**.

Principe

- SaaS est donc la livraison conjointe de moyens, de services et d'expertise qui permettent aux entreprises de :
  - externaliser intégralement un aspect de leur système d'information (messagerie, sécurité...)
  - l'assimiler à un coût de fonctionnement plutôt qu'à un investissement.
  - mettre en œuvre des niveaux de qualité de service (SLA) selon des contrats
  - limiter leur exposition à une technologie puisque la résiliation du contrat leur permet de se désengager d'une technologie plus simplement.

Avantages

- L'utilisation de solutions SaaS en entreprise permet un meilleur contrôle des charges techniques.
- La réduction des dépenses auparavant impliquées par le coût des licences des logiciels, de la maintenance et de l'infrastructure matérielle requise
- La rapidité de déploiement. Les solutions SaaS étant déjà pré-existantes, le temps de déploiement est extrêmement faible.
- La réduction de la consommation électrique en permettant la mutualisation des ressources sur des serveurs partagés par plusieurs entreprises ainsi que l'usage d'un ordinateur à faible consommation muni d'un simple Navigateur Web sans autres licences associés.

Inconvénients

- Lors de la mise en place de solutions SaaS, les données relatives à l'entreprise cliente sont, généralement, stockées sur les serveurs du prestataire fournissant la solution ➔ des soucis de sécurité et de confidentialité.
- Les migrations informatiques peuvent être compliquées puisqu'il faut basculer les données de la plateforme d'un fournisseur vers celle d'un autre, avec divers problèmes associés (compatibilité, apparence pour le client, etc.) ➔ problème de déploiement
- Le même service nécessite le fonctionnement d'au moins trois ordinateurs : client, le fournisseur de service (prestataire) et le fournisseur d'accès internet (FAI) qui assure la communication entre le client et le prestataire.

Exemples de fournisseurs

- Les premiers fournisseurs de solutions SAAS ont été des startups de l'internet, "jeunes" entreprises innovantes offrant des services en B2C (Business To Consumer)comme :
  - Yahoo & Google (Webmail)
  - Amazon.
- D’autres entreprises ont adapté ce type d'offre au monde professionnel en B2B (Business To Business) comme :
  - SalesForce : CRM (Client Relationship Management) à la demande
  - Google : Google Apps
  - Microsoft : Exchange, SharePoint, Live Meeting,…

Qualité des services

Définition

- Aptitude d'un produit ou d'un service à satisfaire les besoins des utilisateurs en termes de fonctionnalités, délais, coûts.
- La notion de qualité est subjective, mais peut se quantifier dans l'entreprise par une mise en conformité avec des standards ou des normes
- Une **norme** est document établi par consensus et approuvé par un organisme reconnu, qui fournit, pour des usages communs et répétés, des règles, des lignes directrices ou des caractéristiques, pour des activités ou leurs résultats.
- Un **standard** est un référentiel publié par une entité privée autre qu'un organisme de normalisation national ou international ou non approuvé par un de ces organismes.
- Non qualité :
  - Les défauts apparaissent lors de l'exploitation du logiciel
  - coût de correction élevé

Définition

- La qualité d'un logiciel n'a pas de mesure objective, ni de définition formelle
- En génie logiciel, 75% des efforts sont consacrés à la maintenance
- Perceptions différentes, par exemple, en fonction de la position dans l'organisation de l'entreprise
- Quelques facteurs de qualité :
  - Pour un **Produit/Logiciel** : conformité, portabilité, maintenabilité, flexibilité
  - **Pour un Service** : efficacité, disponibilité, sécurité, fiabilité

Définition de qques critères de qualité

- **Adaptabilité :** minimiser l'effort nécessaire pour le modifier par suite d'évolution des spécifications
- **Conformité :** contenir un minimum d'erreurs, à satisfaire aux spécifications et à remplir ses missions dans les situations opérationnelles définies.
- **Efficacité :** se limiter à l'utilisation des ressources strictement nécessaires à l'accomplissement de ses fonctions.
- **Maintenabilité :** minimiser l'effort pour localiser et corriger les fautes.
- **Testabilité :** faciliter les procédures de test permettant de s'assurer de l'adéquation des fonctionnalités
- **Interopérabilité :** s'interconnecter à d'autres systèmes.
- **Portabilité :** minimiser l'effort pour se faire transporter dans un autre environnement matériel et/ou logiciel.

Critères de qualité sur les réseaux

- La caractérisation de la qualité des services sur Internet est généralement exprimée par les critères suivants:
  - **Délai:** temps écoulé entre l'envoi d'un paquet par un émetteur et sa réception par le destinataire. Le délai tient compte du délai de propagation le long du chemin et du délai de transmission induit par la mise en file d'attente des paquets dans les systèmes intermédiaires.
  - **Gigue :** variation du délai de bout en bout
  - **Bande passante ou débit maximum:** taux de transfert maximum pouvant être maintenu entre deux points terminaux
  - **Disponibilité** : taux moyen d'erreurs d'une liaison

La série des normes ISO 9000

- **ISO 9000** désigne un ensemble de **normes** relatives à la **gestion de la qualité** publiées par l'Organisation internationale de normalisation (**ISO**).
- Un comité national accorde l'accréditation à divers organisme qui reçoivent ainsi le droit de certifier la conformité des entreprises qui en font la demande à telle ou telle norme ISO.
- En Tunisie, l'**INNORPI** (Institut National De La Normalisation Et De La Propriété Industrielle), mis en place par le gouvernement, est chargé de définir des règles à suivre en matière d'audit et de certification pour les entreprises tunisiennes.

CMMI : Capability Maturity Model + Integration

- CMMI est un modèle de référence, un ensemble structuré de bonnes pratiques, destiné à appréhender, évaluer et améliorer les activités des entreprises d'ingénierie.
- Initialement, CMMI a été développé par le *Software Engineering Institute* pour **appréhender** et **mesurer** la **qualité des services** rendus par les fournisseurs de logiciels informatiques du Département de la Défense US (DoD).
- Il est maintenant largement employé par les entreprises d'ingénierie informatique pour évaluer et améliorer leurs propres services et produits développés.

SLA – Service Level Agreement

- Le SLA est un document qui définit la **qualité de service** requise entre un fournisseur de service et un client.
- SLA = « *accord de niveau de service* » ou « *contrat de niveau de service* » ➔ un contrat dans lequel on formalise la qualité du service en question.
- Dans la pratique, le *SLA* est quelquefois utilisé en référence au temps de délivrance et/ou à la performance (du service) tel que défini dans le contrat.
- SLA se base sur plusieurs mesures de performance tels que le temps de réponse, le pourcentage de satisfaction des invocations/appels, …

Conclusion

Web Service :
Un nouveau Buzz Word ?

Conclusion

- Avantages :
  - Des standards simples (SOAP, WSDL, UDDI)
  - Multi Protocole / Multi OS / Multi Langage
  - Paradigme de Service
  - Des outils (éditeurs et moteurs)
- Inconvénients :
  - Typage (pas de consensus)
  - Performance
  - Jeunesse (Sécurité, Transaction,…)

Références

- SOAP : http://www.w3.org/TR/SOAP/
- WSDL : http://www.w3.org/TR/2001/NOTE-wsdl-20010315
- UDDI : http://www.uddi.org/
- Apache SOAP : http://xml.apache.org/soap/
- Ma thèse