CSI 3540

Structures, techniques et normes du Web

XML Schema, SOAP, WSDL

Objectif:

- Introduction à **XML Schema**
- Introduction aux vocabulaires **SOAP** et **WSDL**
- Comprendres les relations entre **XML Schema**, **SOAP** et **WSDL**

Lectures:

- Web Technologies (2007)
  Pages 502-525

Plan

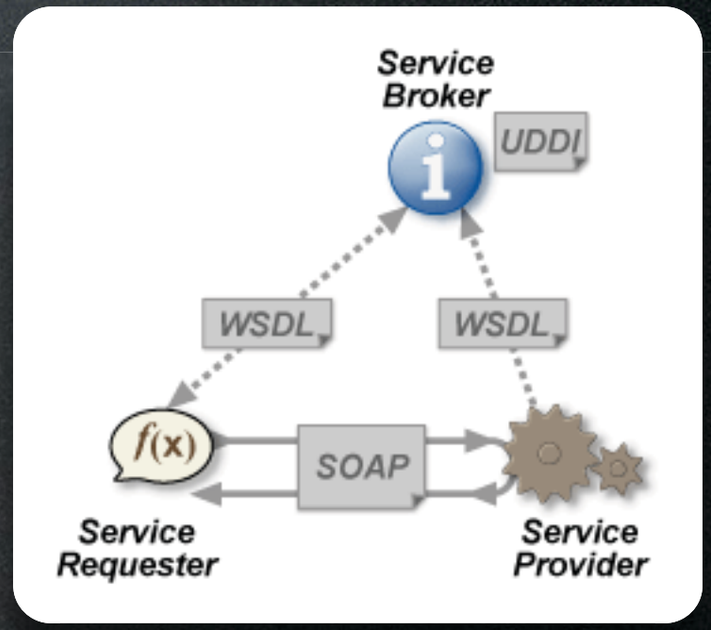
1. **XML Schema**
2. **WSDL**
3. **SOAP**

- Les **Services Web** supportent les interactions **machine-machine** à travers un réseau
- Un **Service Web** supporte les échanges de **messages XML** (**SOAP**) entre **clients** et **serveurs** à l'aide de **HTTP**
- Les (opérations) services sont décrits à l'aide d'un document XML (**WSDL**)
- **SOAP** et **WSDL** font appel à **XML Schema** pour définir la syntaxe des messages et définir les opérations, paramètres et valeurs de retour (particulièrement leurs types)

Service Web

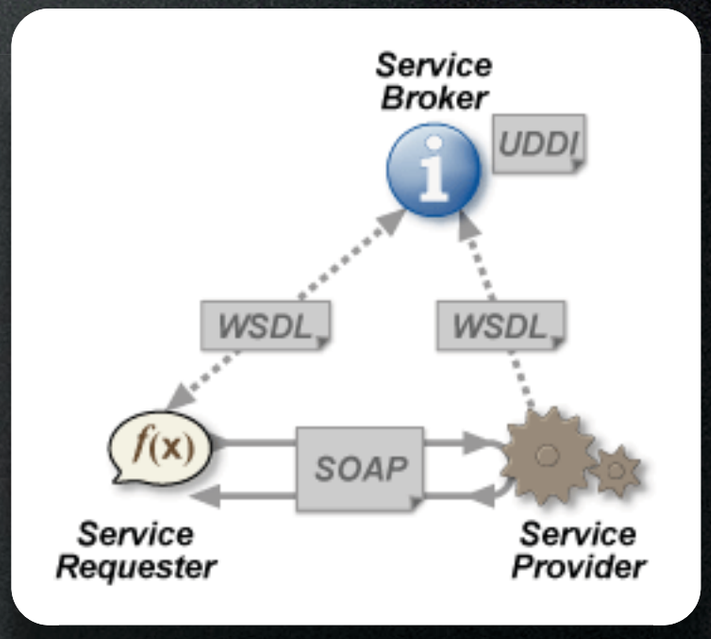
- “a software system designed to support interoperable Machine to Machine interaction over a network” (W3C)

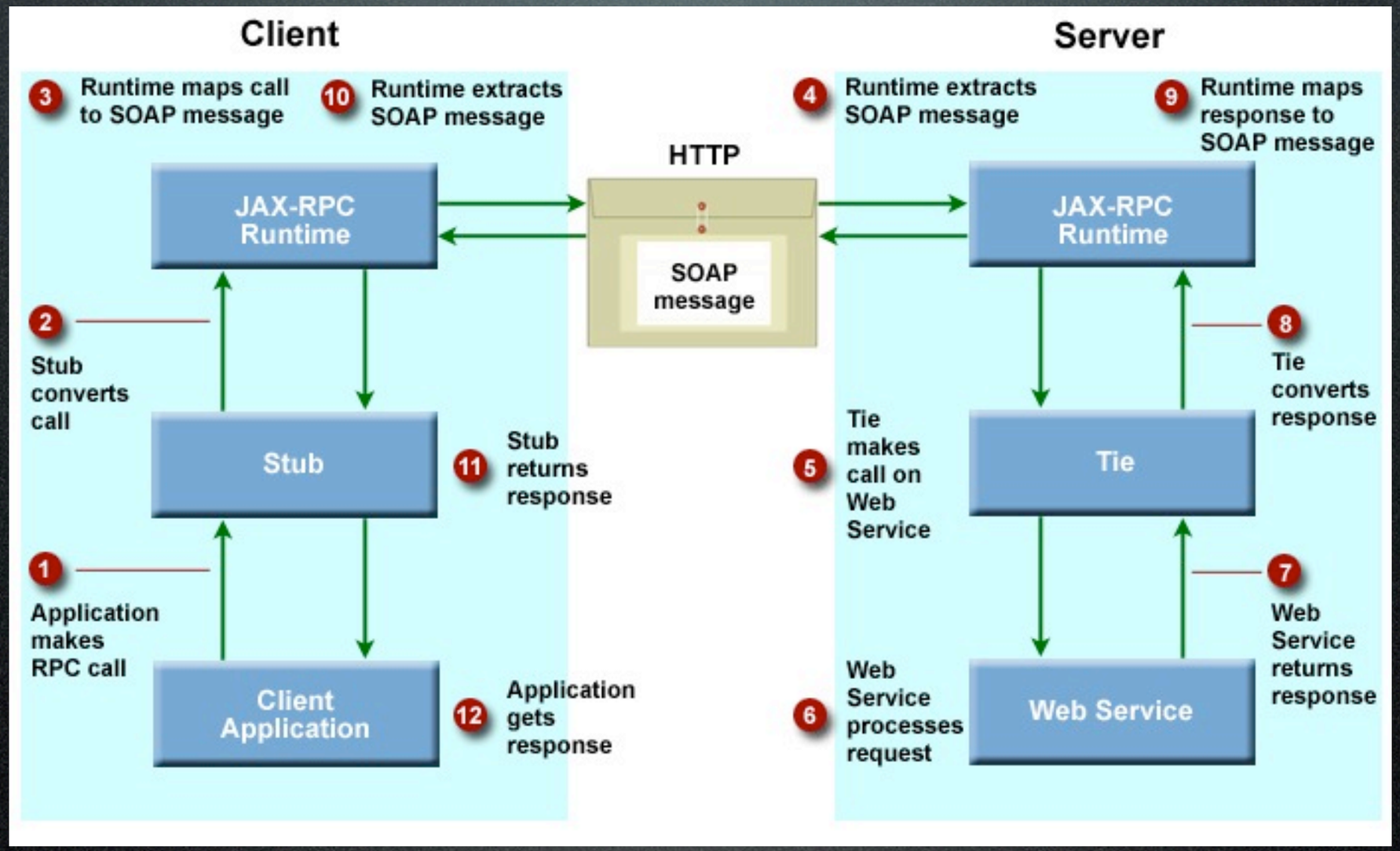
- **API Web**
- Messages **SOAP** via **HTTP**
- Descriptions **WSDL**
- Répertoires **UDDI**



Service Web

- Repose sur des standards ouverts (HTTP, XML)
- Independant de tout langage de programmation, environment, SE
- Modulaire





Source : http://java.sun.com/developer/technicalArticles/WebServices/WSPack2/jaxrpc.html

Pile de protocoles de communication

Couche	**Technologie**
Répertoire	UDDI
Description	**WSDL**
Message	**SOAP**
Transport	HTTP

Schéma XML

Méta-langage

Vocabulaire XML

- Un **vocabulaire** (application) **XML** est une spécification complète des éléments et attributs d'un type spécifique de documents **XML**

1. Langue naturelle (petits groupes)
2. Déclarations de type de document (**DTD**) (documents publiques)
3. **XML Schema**

Un autre formalisme ?

```
<!ENTITY % address "(street, city, province, postal-code)" >
   <!ELEMENT street (#PCDATA) >
   <!ELEMENT city (#PCDATA) >
   <!ELEMENT province (#PCDATA) >
   <!ELEMENT postal-code (#PCDATA) >
```

- Le **DTD** est **peu expressif**
- Le **DTD** contraint la structure du document (ordre et imbrication des éléments), mais pas le **contenu textuel**

Un autre formalisme ?

- Le **DTD** ne supporte pas les **espaces de nommage**
- Ainsi, si **a.dtd** importe le contenu de **b.dtd** (à l'aide d'une directive), il faut au préalable s'assurer que les noms d'éléments sont uniques
- ~~**a:name et b:name**~~

Schéma XML

- **Un schéma définit une classe de documents XML**
- Un document qui satisfait un schéma est une **instance** de ce schéma
- Un **Schéma XML** est un document **XML**!

Puisque c'est un document XML, les espaces de nommage sont supportés et l'on bénéficie de la panoplie d'outils XML

```
<?xml version="1.0" encoding="UTF-8"?>
<score xmlns="http://www.site.uottawa.ca/tennis"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.site.uottawa.ca/tennis tennis.xsd">10</score>
```

tennis.xsd :

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNameSpace="http://www.site.uottawa.ca/tennis">
  <xs:element name="score">
    <xs:simpleType>
      <xs:restriction base="xs:short">
        <xs:enumeration value="10">
        </xs:enumeration>
        <xs:enumeration value="15">
        </xs:enumeration>
        <xs:enumeration value="30">
        </xs:enumeration>
        <xs:enumeration value="40">
        </xs:enumeration>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<score>10</score>
```

```
...
<xs:complexType name="Wine">
   <xs:sequence>
      <xs:element name="quantity" type="xs:nonNegativeInteger" />
      <xs:element name="rating" minOccurs="0">
         <xs:complexType>
            <xs:attribute name="stars" type="xs:positiveInteger"/>
         </xs:complexType>
      </xs:element>
      <xs:element name="comment" type="cat:Comment" minOccurs="0" />
   </xs:sequence>
   <xs:attribute name="code" type="cat:SAQ-code" use="required"/>
</xs:complexType>
...
```

```
<?xml version="1.0" encoding="UTF-8"?>
<wine code="00518712" >
  <rating stars="2"/>
  <quantity>5</quantity>
</wine>
```

Schéma XML

- **Le formalisme permet la définition de types!**
- De sorte que lorsqu'on définit un élément, par exemple **code-postal**, on peut aussi lui associer un type, **CodePostal**, afin de spécifier l'ensemble des valeurs possibles, ici, on s'assure que le format est bien

[A-Z] [0-9] [A-Z] [0-9] [A-Z] [0-9]

Convention :
caligraphie semblable aux classes de Java.

code-postal élément
CodePostal type

Schéma XML

- Le document **WSDL** définit les opérations du service. Ce faisant, il doit aussi définir le type des paramètres et des valeurs de retour
- Les messages **SOAP** encodent les valeurs des paramètres et des valeurs de retour
- Des **définitions formelles** (rigoureuses) **du type** des valeurs sont absolument nécessaires

Schéma XML

- Puisque le schéma est un vocabulaire **XML**, il faudra **deux espaces de nommage** afin de distinguer le schéma du vocabulaire qu'il définit
- Usuellement un fichier **.xsd**
- **XML Schema** remplace le **DTD** comme formalisme pour la définition d'un vocabulaire

Schéma XML

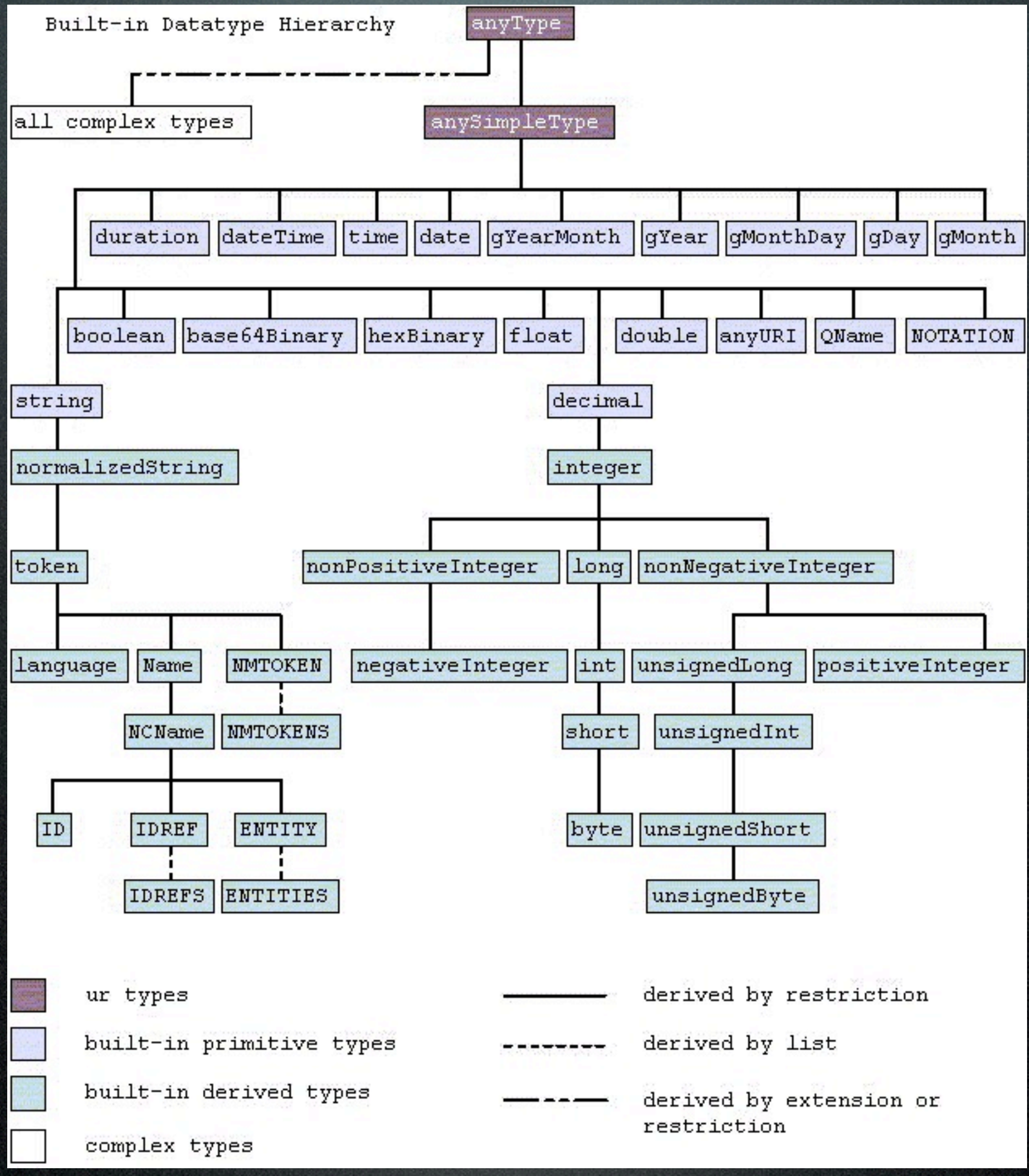
- Il y a deux catégories de types : **simples** et **complexes**
- Un **type simple** définit le contenu textuel d'un élément n'ayant aucun élément enfant
- Un **type complexe** définit le contenu d'un élément ayant des éléments enfants (données structurées)

Types

- **zip** sera associé à un type simple
- **address** sera associé à un type complexe

```
<address>
  <street>800, King Edward</street>
  <city>Ottawa</city>
  <zip>K1N 6N5</zip>
</address>
```

- Pleusieurs types prédéfinis : **string**, **int**, **boolean**, mais aussi **date**, **URI**, etc.


Built-in Datatype Hierarchy
anyType
all complex types
anySimpleType
duration
dateTime
time
date
gYearMonth
gYear
gMonthDay
gDay
gMonth
boolean
base64Binary
hexBinary
float
double
anyURI
QName
NOTATION
string
decimal
normalizedString
integer
token
nonPositiveInteger
long
nonNegativeInteger
language
Name
NMTOKEN
negativeInteger
int
unsignedLong
positiveInteger
NCName
NMTOKENS
short
unsignedInt
ID
IDREF
ENTITY
byte
unsignedShort
IDREFS
ENTITIES
unsignedByte
ur types
built-in primitive types
built-in derived types
complex types
derived by restriction
derived by list
derived by extension or restriction

Types simples

- **XML Schema** spécifie l'**intervalle de valeurs** ainsi que la **représentation** sous forme de chaînes de caractères des types prédéfinis

Types simples

- Les types simples sont associés aux éléments sans enfants
- **<xs:simpleType>** est utilisé afin de définir un nouveau type simple
- Il faut spécifier un **type de base** ainsi des **restrictions**

Types simples

```
<xs:simpleType name="Pourcentage">
 <xs:restriction base="xs:number">
  <xs:fractionDigits value="1"/>
  <xs:minExclusive value="0.0"/>
  <xs:maxExclusive value="100.0"/>
 </xs:restriction>
</xs:simpleType>
```

```
<xs:simpleType name="Résultat" >
  <union memberTypes="Pourcentage Annotation" />
</xs:simpleType>
```

```
<xs:simpleType name="Annotation">
 <xs:restriction base="xs:string">
  <xs:enumeration value="INC"/>
  <xs:enumeration value="ABS"/>
 </xs:restriction>
</xs:simpleType>
```

Les éléments utilisés afin de contraindre l'espace des valeurs s'appellent facet

Types simples

```
<xs:simpleType name="Pourcentage">
 <xs:restriction base="xs:number">
  <xs:fractionDigits value="1"/>
  <xs:minExclusive value="0.0"/>
  <xs:maxExclusive value="100.0"/>
 </xs:restriction>
</xs:simpleType>
```

```
<xs:simpleType name="Annotation">
 <xs:restriction base="xs:string">
  <xs:enumeration value="INC"/>
  <xs:enumeration value="ABS"/>
 </xs:restriction>
</xs:simpleType>
```

```
<xs:simpleType name="Résultat" >
    <union memberTypes="Pourcentage Annotation" />
</xs:simpleType>
```

```
<xs:simpleType name="Résultats" >
    <list itemType="Résultat" />
</xs:simpleType>
```

Restrictions (facets)

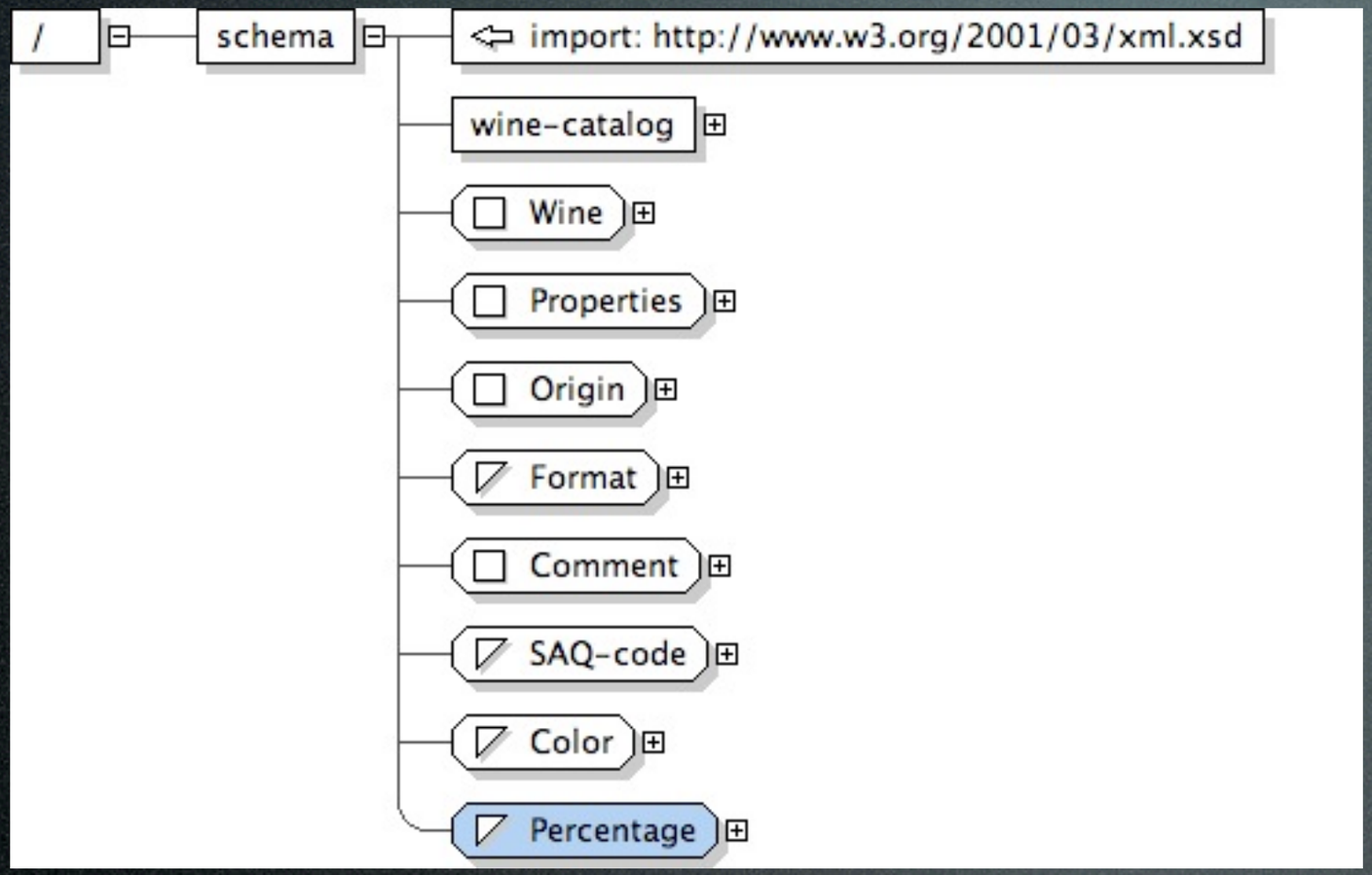
- En plus des restrictions présentées, on peut spécifier une taille minimale (**minLength**) ou maximale (**maxLength**) ainsi que plusieurs autres

Types complexes

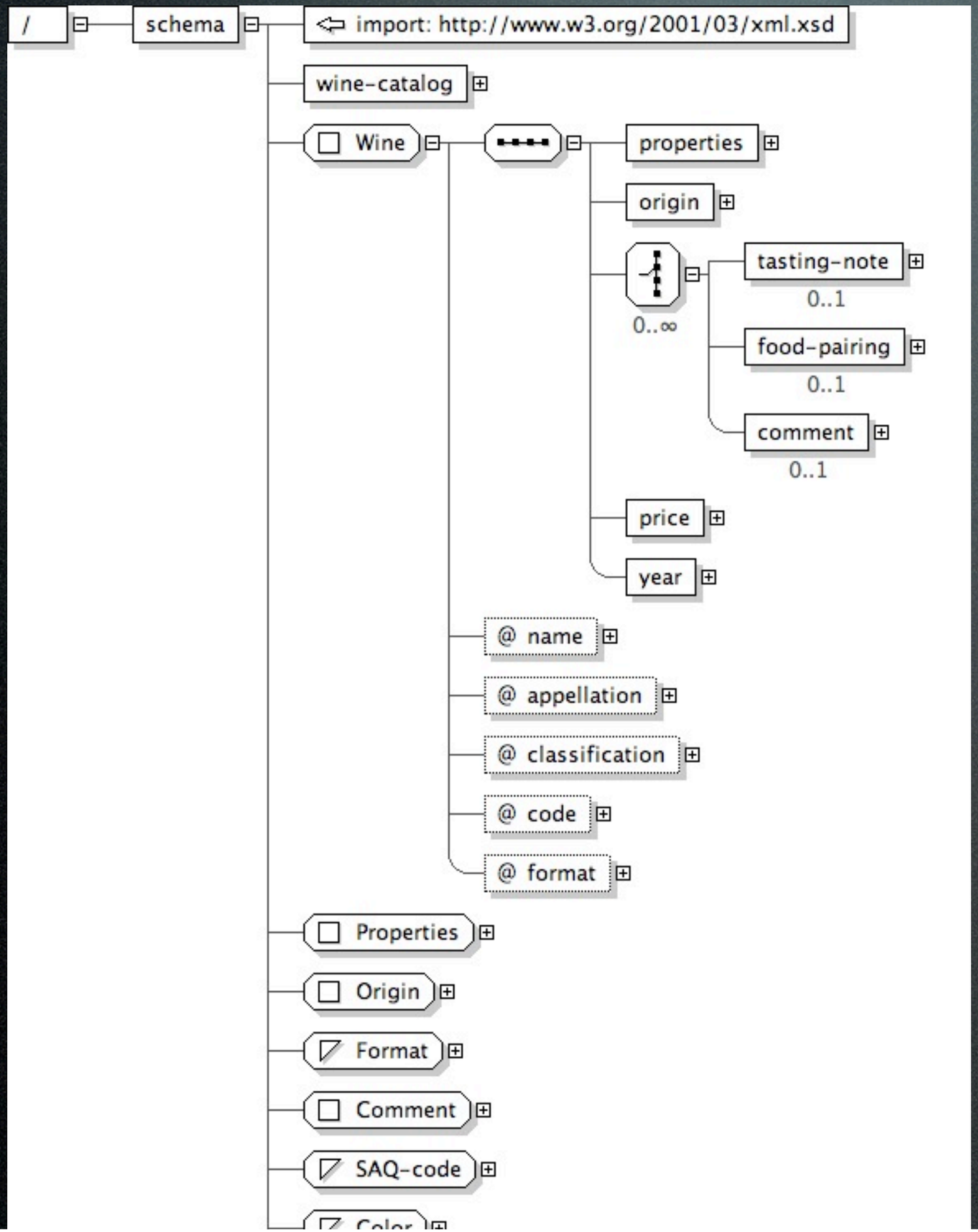
- Les types complexes sont associés aux éléments ayant des enfants (internes)
- **<xs:complexType>** est utilisé afin de définir un nouveau type complexe
- **xs:sequence**, **xs:choice**, **xs:all**

WineCatalog.xsd

- L'exemple qui suit servira à illustrer les différents concepts
- Notons que la balise **xs:element** sert à la déclaration d'un élément du vocabulaire **cible**
- Notons aussi qu'il y a plusieurs façons de structurer un schéma selon que l'on définit des types **anonymes** ou **nommés**



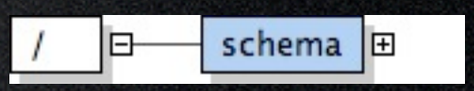
Organisation du schéma, et non l'organisation logique du vocabulaire cible



Le diagram illustre la structure du schéma et non la structure du vocabulaire

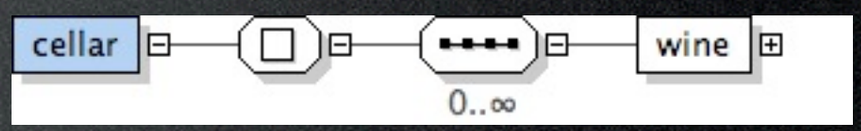
Élément racine : **xs:schema**

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:cat="http://www.iro.umontreal.ca/lapalme/wine-catalog"
           targetNamespace="http://www.iro.umontreal.ca/lapalme/wine-catalog">
...
</xs:schema>
```



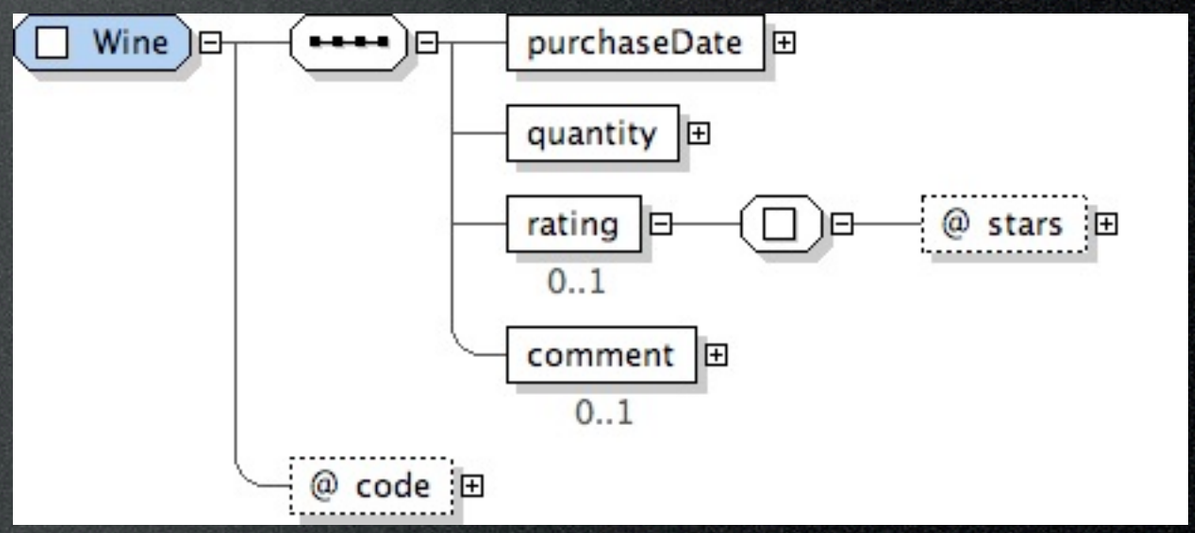
type anonyme

```
<xs:element name="cellar">
   <xs:complexType>
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
         <xs:element name="wine" type="Wine"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
...
```



type nommé

```
<xs:complexType name="Wine">
   <xs:sequence>
      <xs:element name="purchaseDate" type="xs:date"/>
      <xs:element name="quantity" type="xs:nonNegativeInteger"/>
      <xs:element name="rating" minOccurs="0">
         <xs:complexType>
            <xs:attribute name="stars" type="xs:positiveInteger"/>
         </xs:complexType>
      </xs:element>
      <xs:element name="comment" type="cat:Comment" minOccurs="0"/>
   </xs:sequence>
   <xs:attribute name="code" type="cat:SAQ-code" use="required"/>
</xs:complexType>
...
```



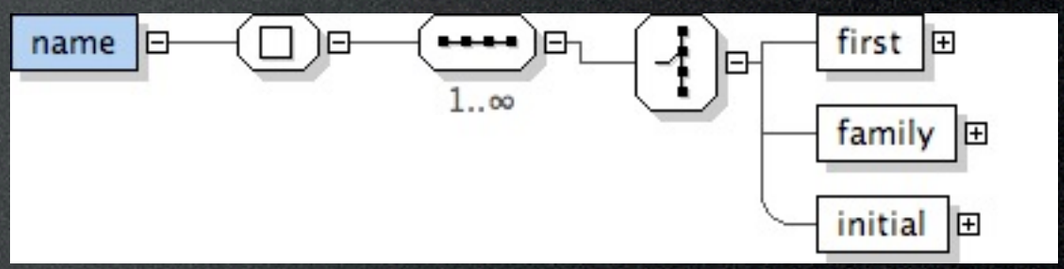
type nommé, utilisation de

```
...
<xs:element name="wine-catalog">
      <xs:complexType>
         <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:element name="wine" type="cat:Wine"/>
         </xs:sequence>
         <!-- needed because this schema will be imported...-->
         <xs:attribute ref="xml:base"/>
      </xs:complexType>
         <xs:key name="WineNumber">
            <!-- XPath expressions must be qualified -->
            <xs:selector xpath="cat:wine"/>
            <xs:field xpath="@code"/>
          </xs:key>
         <xs:unique name="WineName">
            <xs:selector xpath="cat:wine"/>
            <xs:field xpath="@name"/>
            <xs:field xpath="@appellation"/>
         </xs:unique>
   </xs:element>
...
```

xs:all

Les éléments apparaissent dans n'importe quel ordre

```
<xs:element name="name">
   <xs:complexType>
      <xs:all>
            <xs:element name="first" type="xs:string" minOccurs="0" />
            <xs:element name="family" type="xs:string" minOccurs="0"/>
            <xs:element name="initial" type="xs:string" minOccurs="0"/>
      </xs:all>
   </xs:complexType>
</xs:element>
...
```

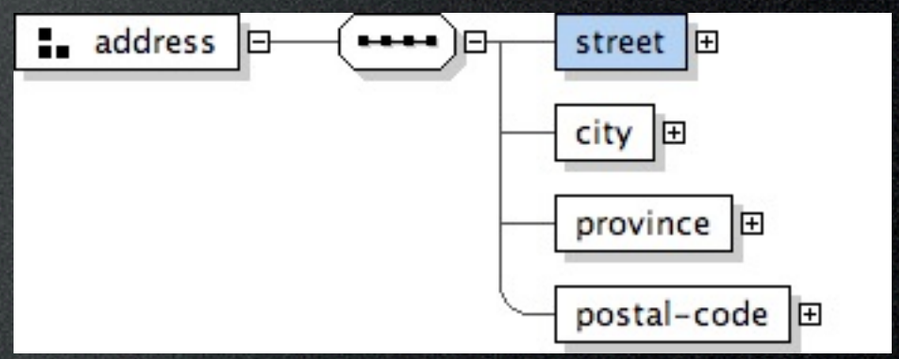


xs:group

Associe un nom à un groupe d'éléments qui servent ensuite de blocs de construction pour l'élaboration de types complexes

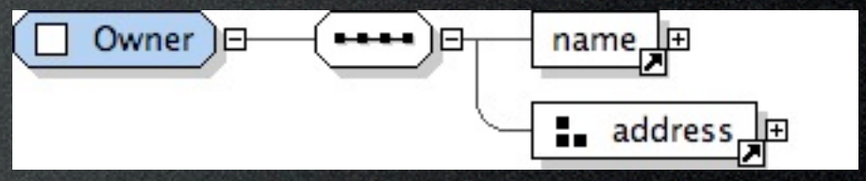
L'élément address n'existe pas

```
<xs:group name="address">
   <xs:sequence>
      <xs:element name="street" type="xs:string"/>
      <xs:element name="city" type="xs:string"/>
      <xs:element name="province" type="ProvinceCA"/>
      <xs:element name="postal-code" type="PostalCodeCA"/>
   </xs:sequence>
</xs:group>
...
```



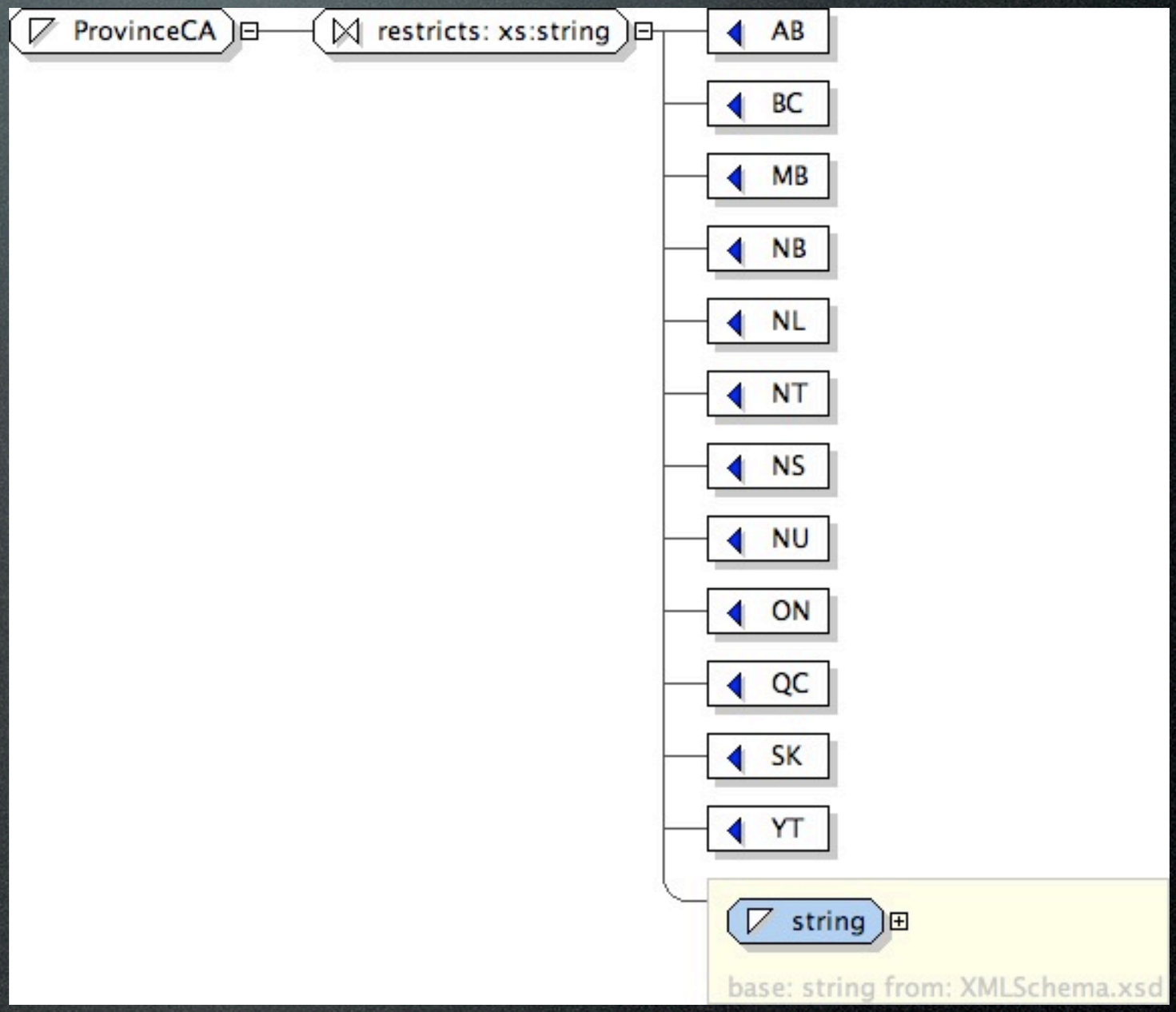
Fait appel au groupe **address**

```
<xs:complexType name="Owner">
   <xs:sequence>
      <xs:element ref="name"/>
      <xs:group ref="address"/>
   </xs:sequence>
</xs:complexType>
...
```



xs:restriction/xs:enumeration

```
<xs:simpleType name="ProvinceCA">
  <xs:restriction base="xs:string">
    <xs:enumeration value="AB"/>
    <xs:enumeration value="BC"/>
    <xs:enumeration value="MB"/>
    <xs:enumeration value="NB"/>
    <xs:enumeration value="NL"/>
    <xs:enumeration value="NT"/>
    <xs:enumeration value="NS"/>
    <xs:enumeration value="NU"/>
    <xs:enumeration value="ON"/>
    <xs:enumeration value="QC"/>
    <xs:enumeration value="SK"/>
    <xs:enumeration value="YT"/>
  </xs:restriction>
</xs:simpleType>
...
```



xs:restriction/xs:pattern

```
<xs:simpleType name="PostalCodeCA">
   <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z][0-9][A-Z] [0-9][A-Z][0-9]"/>
   </xs:restriction>
</xs:simpleType>
...
```

Principaux éléments

- xs:schema
- xs:element
- xs:attribute
- xs:simpleType
- xs:complexType
- xs:group
- xs:sequence
- xs:choice
- xs:restriction

Remarques

Remplace le DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="WineCatalog.xsl"?>

<wine-catalog xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.iro.umontreal.ca/lapalme/wine-catalog WineCatalog.xsd"
   xmlns="http://www.iro.umontreal.ca/lapalme/wine-catalog">
   <wine name="Domaine de l'île Margaux" appellation="Bordeaux supérieur"
      classification="a.c." code="C00043125" format="750ml">
      <properties>
         <color>red</color>
         <alcoholic-strength>12.5</alcoholic-strength>
         <nature>still</nature>
      </properties>
      ...
</wine-catalog>
```

Schéma comme sous partie d'un document XML

```
...
  <types>
    <schema targetNamespace="http://tempuri.org/types"
            xmlns:tns="http://tempuri.org/types"
            xmlns:soap11-enc="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
            xmlns="http://www.w3.org/2001/XMLSchema">
      <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
      <complexType name="ExchangeValues">
        <sequence>
          <element name="dollars" type="double"/>
          <element name="euros" type="double"/>
          <element name="yen" type="double"/>
        </sequence>
      </complexType>
    </schema>
  </types>
...
```

Partie d'un document WSDL comportant des éléments XML Schema

Ce document WSDL définit le type ExchangeValues

Le document SOAP de la page qui suit l'utilise!

Schéma comme sous partie d'un document XML

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
              xmlns:ns0="http://tempuri.org/types"
              env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <env:Body>
    <ans1:fromDollarsResponse xmlns:ans1 ="http://tempuri.org/wsdl">
      <result href="#ID1"/>
    </ans1:fromDollarsResponse>
    <ns0:ExchangeValues id="ID1" xsi:type="ns0:ExchangeValues">
      <dollars xsi:type="xsd:double">1.0</dollars>
      <euros xsi:type="xsd:double">0.746826</euros>
      <yen xsi:type="xsd:double">102.56</yen>
    </ns0:ExchangeValues>
  </env:Body>
</env:Envelope>
```

Message SOAP comportant certains éléments XML Schema

Alternatives

- **RELAX NG** (**REgular LAnguage for XML, New Generation**) ; notation plus compacte et plus puissante
- **Schematron** (un vocabulaire XML) ajoute un formalisme pour définir des contraintes sémantique («**est-que la valeur de la date de fin est plus grande que celle de la date du début ?**»)

Services Web

SOAP et DOM

Pile de protocoles de communication

Couche	**Technologie**
Répertoire	UDDI
Description	**WSDL**
Message	**SOAP**
Transport	HTTP

WSDL

Web Services Description Language

WSDL

- **« Web Services Description Language »**
- **Contrat** entre un **service** et les **consommateurs** de ce service
  - « service endpoint »
    - opérations
    - types de données
    - description des messages

WSDL

- Vocabulaire XML pour la description de services Web
- Recommandation W3C
  Version 2.0 ; 26 juin 2007
- http://www.w3.org/TR/wsdl20/
- **wscompile** produit des documents `1.1`

Interface du service

```
package myCurCon;

public interface CurCon {
    public ExchangeValues fromDollars( double dollars );
    public ExchangeValues fromEuros( double euros );
    public ExchangeValues fromYen( double yen );
}
```

```
public class ExchangeValues {
    public double dollars;
    public double euros;
    public double yen;
}
```

Exemple du livre du cours, pages 491-502.

Structure d'un WSDL

- **descriptions** (élément racine)
  - **types**
  - **messages**
  - **portType**
  - **binding**
  - **service**

Document WSDL

```
<?xml version="1.0" encoding="UTF-8"?>

<definitions name="HistoricCurrencyConverter"
             targetNamespace="http://tempuri.org/wsdl"
             xmlns:tns="http://tempuri.org/wsdl"
             xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:ns2="http://tempuri.org/types"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
...
```

«definitions» est l'élément racine

Ses attributs sont les déclarations des espaces de nommages utilisés par les diverses parties du document

Document WSDL

```
...
  <types>
    <schema targetNamespace="http://tempuri.org/types"
            xmlns:tns="http://tempuri.org/types"
            xmlns:soapl 1-enc="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
            xmlns="http://www.w3.org/2001/XMLSchema">
      <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
      <complexType name="ExchangeValues">
        <sequence>
          <element name="dollars" type="double"/>
          <element name="euros" type="double"/>
          <element name="yen" type="double"/>
        </sequence>
      </complexType>
    </schema>
  </types>
...
```

Définition de types à l'aide de XML Schema

Ici, le type ExchangeValues est définit et associé à l'espace de nommage : http://tempuri.org/types

Globalement, via l'élément definitions, le prefixe ns2 avait été associé à l'espace de nommage http://tempuri.org/types

ns2:ExchangeValues a donc été définit

Remarque : l'espace de nomme par défaut est celui d'XML Schema

Document WSDL

```
...
  <message name="CurCon_fromDollars">
    <part name="double_1" type="xsd:double"/>
  </message>
  <message name="CurCon_fromDollarsResponse">
    <part name="result" type="ns2:ExchangeValues"/>
  </message>
  <message name="CurCon_fromEuros">
    <part name="double_1" type="xsd:double"/>
  </message>
  <message name="CurCon_fromEurosResponse">
    <part name="result" type="ns2:ExchangeValues"/>
  </message>
  <message name="CurCon_fromYen">
    <part name="double_1" type="xsd:double"/>
  </message>
  <message name="CurCon_fromYenResponse">
    <part name="result" type="ns2:ExchangeValues"/>
  </message>
...
```

Pour chaque opération, définis (nom + type) la liste des paramètres et la valeur de retour

Il y aura un élément « part » par paramètre

Les noms sont uniques (seulement) à l'intérieur d'un message

Les types sont soient les types prédéfinis de XML Schema ou ceux définis dans par l'élément type plus haut

Document WSDL

```
...
  <portType name="CurCon">
    <operation name="fromDollars" parameterOrder="double_1">
      <input message="tns:CurCon_fromDollars"/>
      <output message="tns:CurCon_fromDollarsResponse"/>
    </operation>
    <operation name="fromEuros" parameterOrder="double_1">
      <input message="tns:CurCon_fromEuros"/>
      <output message="tns:CurCon_fromEurosResponse"/>
    </operation>
    <operation name="fromYen" parameterOrder="double_1">
      <input message="tns:CurCon_fromYen"/>
      <output message="tns:CurCon_fromYenResponse"/>
    </operation>
  </portType>
...
```

Le nom des méthodes est déterminé par les annotations @WebMethod

Définis les opérations de ce service Web

Les entrées et sorties sont associées aux messages définis ci-haut (précédemment)

Document WSDL

```
...
  <binding name="CurConBinding" type="tns:CurCon">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
    <operation name="fromDollars">
      <soap:operation soapAction=""/>
      <input>
        <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                   use="encoded" namespace="http://tempuri.org/wsdl"/>
      </input>
      <output>
        <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                   use="encoded" namespace="http://tempuri.org/wsdl"/>
      </output>
    </operation>
    <operation name="fromEuros">
      ...
    </operation>
    <operation name="fromYen">
      ...
    </operation>
  </binding>
...
```

application-layer protocol

rpc ou **document**

Si portType spécifie l'interface, binding est l'implémentation

SOAP over HTTP

Spécifie l'encodage des messages, ici SOAP, et le protocol de transport, ici HTTP.

Il pourrait y avoir plusieurs associations (bindings) par service (portType, tns:CurCon)

Document WSDL

```
...
  <service name="HistoricCurrencyConverter">
    <port name="CurConPort" binding="tns:CurConBinding">
      <soap:address location="REPLACE_WITH_ACTUAL_URL"/>
    </port>
  </service>
</definitions>
```

Utilisations concrètes de WSDL

- http://webservices.amazon.com/AWSECommerceService/AWSECommerceService.wsdl?
- http://developer.ebay.com/webservices/latest/ebaySvc.wsdl
- http://www.weather.gov/forecasts/xml/DWMLgen/schema/DWML.xsd
- http://api.google.com/GoogleSearch.wsdl

Web Inspector — http://api.google.com/GoogleSearch.wsdl

Search

DOCUMENTS

GoogleSearch.wsdl

```
<!-- WSDL description of the Google Web APIs.
     The Google Web APIs are in beta release. All interfaces are subject to
     change as we refine and extend our APIs. Please see the terms of use
     for more information. -->

<!-- Revision 2002-08-16 -->

<definitions name="GoogleSearch"
             targetNamespace="urn:GoogleSearch"
             xmlns:typens="urn:GoogleSearch"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
             xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
             xmlns="http://schemas.xmlsoap.org/wsdl/">

  <!-- Types for search - result elements, directory categories -->

  <types>
    <xsd:schema xmlns="http://www.w3.org/2001/XMLSchema"
                targetNamespace="urn:GoogleSearch">

      <xsd:complexType name="GoogleSearchResult">
        <xsd:all>
          <xsd:element name="documentFiltering"           type="xsd:boolean"/>
          <xsd:element name="searchComments"              type="xsd:string"/>
          <xsd:element name="estimatedTotalResultsCount"  type="xsd:int"/>
          <xsd:element name="estimateIsExact"             type="xsd:boolean"/>
          <xsd:element name="resultElements"              type="typens:ResultElementArray"/>
          <xsd:element name="searchQuery"                 type="xsd:string"/>
          <xsd:element name="startIndex"                  type="xsd:int"/>
          <xsd:element name="endIndex"                    type="xsd:int"/>
          <xsd:element name="searchTips"                  type="xsd:string"/>
          <xsd:element name="directoryCategories"         type="typens:DirectoryCategoryArray"/>
          <xsd:element name="searchTime"                  type="xsd:double"/>
        </xsd:all>
      </xsd:complexType>

      <xsd:complexType name="ResultElement">
        <xsd:all>
          <xsd:element name="summary" type="xsd:string"/>
          <xsd:element name="URL" type="xsd:string"/>
          <xsd:element name="snippet" type="xsd:string"/>
          <xsd:element name="title" type="xsd:string"/>
          <xsd:element name="cachedSize" type="xsd:string"/>
          <xsd:element name="relatedInformationPresent" type="xsd:boolean"/>
          <xsd:element name="hostName" type="xsd:string"/>
          <xsd:element name="directoryCategory" type="typens:DirectoryCategory"/>
          <xsd:element name="directoryTitle" type="xsd:string"/>
        </xsd:all>
      </xsd:complexType>
```

Console

Network

Friday, April 9, 2010

Web Inspector — http://api.google.com/GoogleSearch.wsdl

Search

DOCUMENTS

GoogleSearch.wsdl

```
<!-- Messages for Google Web APIs - cached page, search, spelling. -->

<message name="doGetCachedPage">
  <part name="key"             type="xsd:string"/>
  <part name="url"             type="xsd:string"/>
</message>

<message name="doGetCachedPageResponse">
  <part name="return"          type="xsd:base64Binary"/>
</message>

<message name="doSpellingSuggestion">
  <part name="key"             type="xsd:string"/>
  <part name="phrase"          type="xsd:string"/>
</message>

<message name="doSpellingSuggestionResponse">
  <part name="return"          type="xsd:string"/>
</message>

<!-- note, ie and oe are ignored by server; all traffic is UTF-8. -->

<message name="doGoogleSearch">
  <part name="key"             type="xsd:string"/>
  <part name="q"               type="xsd:string"/>
  <part name="start"           type="xsd:int"/>
  <part name="maxResults"      type="xsd:int"/>
  <part name="filter"          type="xsd:boolean"/>
  <part name="restrict"        type="xsd:string"/>
  <part name="safeSearch"      type="xsd:boolean"/>
  <part name="lr"              type="xsd:string"/>
  <part name="ie"              type="xsd:string"/>
  <part name="oe"              type="xsd:string"/>
</message>

<message name="doGoogleSearchResponse">
  <part name="return"          type="typens:GoogleSearchResult"/>
</message>

<!-- Port for Google Web APIs, "GoogleSearch" -->

<portType name="GoogleSearchPort">

  <operation name="doGetCachedPage">
    <input message="typens:doGetCachedPage"/>
    <output message="typens:doGetCachedPageResponse"/>
  </operation>

  <operation name="doSpellingSuggestion">
    <input message="typens:doSpellingSuggestion"/>
    <output message="typens:doSpellingSuggestionResponse"/>
  </operation>
```

Console

Network

Friday, April 9, 2010

Web Inspector — http://api.google.com/GoogleSearch.wsdl

Search

DOCUMENTS

GoogleSearch.wsdl

```
<!-- Binding for Google Web APIs - RPC, SOAP over HTTP -->

<binding name="GoogleSearchBinding" type="typens:GoogleSearchPort">
  <soap:binding style="rpc"
                transport="http://schemas.xmlsoap.org/soap/http"/>

  <operation name="doGetCachedPage">
    <soap:operation soapAction="urn:GoogleSearchAction"/>
    <input>
      <soap:body use="encoded"
                 namespace="urn:GoogleSearch"
                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </input>
    <output>
      <soap:body use="encoded"
                 namespace="urn:GoogleSearch"
                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </output>
  </operation>

  <operation name="doSpellingSuggestion">
    <soap:operation soapAction="urn:GoogleSearchAction"/>
    <input>
      <soap:body use="encoded"
                 namespace="urn:GoogleSearch"
                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </input>
    <output>
      <soap:body use="encoded"
                 namespace="urn:GoogleSearch"
                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </output>
  </operation>

  <operation name="doGoogleSearch">
    <soap:operation soapAction="urn:GoogleSearchAction"/>
    <input>
      <soap:body use="encoded"
                 namespace="urn:GoogleSearch"
                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </input>
    <output>
      <soap:body use="encoded"
                 namespace="urn:GoogleSearch"
                 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </output>
  </operation>
</binding>

<!-- Endpoint for Google Web APIs -->
<service name="GoogleSearchService">
  <port name="GoogleSearchPort" binding="typens:GoogleSearchBinding">
```

Console

Network

Friday, April 9, 2010

SOAP

Simple Object Access Protocol

SOAP

- **Simple Object Access Protocol**
- Recommandation **W3C**
  Version 1.2 ; 27 avril 2007
- «(...) a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment»
- C’est un vocabulaire XML

SOAP

- On utilise **SOAP** à deux fins dans le contexte des services Web :
  - Appels de méthodes distantes (SOAP for **RPC** ; JAX-**RPC**)
  - Passage de messages (SOAP for **messaging**)

«Remote procedure call»

- **RPC** («Remote Procedure Call») est une technologie permettant l'appel de méthodes sur un ordinateur distant
- Les diverses implémentations, en général, **cachent les détails liés aux communications** de sorte que l'appel ressemble à un appel local
- **RFC 707** de 14-JAN-**1976** ; **SUN RPC** est l'une des implémentations les mieux connues

«Remote procedure call»

- Ce qui rend les implémentations basées sur **SOAP** particulièrement intéressantes, ce sont les standards ouverts utilisés (**XML**, **HTTP**...) qui permettent donc la cohabitation de langages de programmation différents, systèmes d'exploitation différents...

des protocoles texte, bien supportés...

SOAP

- Un **document** est un message ; tout échange d'information entre un service et son client se fait par l'envoi de messages
- Le **message** réside dans une enveloppe
- Une **enveloppe** est constituée
  - d'un **en-tête** (optionnelle)
  - et obligatoirement d'un **corps**

SOAP

- L'élément racine est **Envelope**
- L'**espace de nommage** des documents SOAP est :

http://schemas.xmlaoap.org/soap/envelope

SOAP

- L'attribut **env:encodingStyle** spécifie le schéma d'encodage (types des données, méthode utilisée pour leur sérialisation)
- Depuis 1.2 : le schéma d'encodage ne peut apparaître dans l'élément racine, il n'apparaît que dans certains éléments spécifiques

Message requête SOAP

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
              xmlns:ns0="http://tempuri.org/wsdl"
              xmlns:ns1="http://tempuri.org/types"
              env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <env:Body>
    <ns0:fromDollars>
      <double_1 xsi:type="xsd:double">1.0</double_1>
    </ns0:fromDollars>
  </env:Body>
</env:Envelope>
```

accessor

struct

SOAP ne se conforme pas au mécanismes standards liés aux espaces de nommages

Techniquement, doubl_1 n'appartient à aucun espace de nommage...

Message réponse SOAP

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
              xmlns:ns0="http://tempuri.org/types"
              env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <env:Body>
    <ans1:fromDollarsResponse xmlns:ans1 ="http://tempuri.org/wsdl">
      <result href="#ID1"/>
    </ans1:fromDollarsResponse>
    <ns0:ExchangeValues id="ID1" xsi:type="ns0:ExchangeValues">
      <dollars xsi:type="xsd:double">1.0</dollars>
      <euros xsi:type="xsd:double">0.746826</euros>
      <yen xsi:type="xsd:double">102.56</yen>
    </ns0:ExchangeValues>
  </env:Body>
</env:Envelope>
```

struct accessor

Services Web

- Les **Services Web** supportent les appels de méthodes sur des ordinateurs distants
- Pour ce faire, ils reposent sur un ensemble de technologies XML ouvertes

Couche	**Technologie**
Répertoire	UDDI
Description	**WSDL**
Message	**SOAP**
Transport	HTTP

Standards Ouverts

- **Web Services Interoperability Organization (WS-I)** : IBM, Microsoft, BEA Systems, SAP, Oracle, Fujitsu, Hewlett-Packard, et Intel ; ainsi que deux membres élus Sun Microsystems et webMethods

- www.sw-i.org

Épilogue

Concepts

- Prologue
- Assises du Web
- **HTTP**
- Langage de balisage XHTML
- DTD XHTML
- CSS
- **JavaScript**
- **DOM**
- **Événements du DOM**
- CGI
- **Servlet**
- **XML, Processeur SAX, Processeur DOM, Transformations, XPath, XSL**
- **JSP**, EL, JSTL
- **Ajax**
- **WebServices** : **SOAP**, **WSDL**
- Épilogue

Sujets connexes

1. Connexions aux bases de données
2. SIP (Session Initiation Protocol) (Voice-over-IP (VoIP) phone service, instant messaging, presence and buddy list management and web conferencing) SailFin/ GlassFish
3. JavaFaces Server
4. Web sémantique (Ontologies, sémantique et services)
5. Friends of Friends (FOF)...
6. Sécurité (authentication, encryption...)
7. Vie privée

Discussion

- **Au sujet du cours** :
  - Qu'est-ce qui a fonctionné ?
  - Qu'est-ce qui doit être changé ?
  - Qu'est-ce qui manque ?
    - En plus des laboratoires, il pourrait y avoir des tutoriels, quand pensez-vous ?

Discussion

- **Au sujet du cours** :
  - Vous est-il arrivé (depuis le début du cours) d'associer certaines technologies vues en classe à certains sites Web ?

Merci!

Bon examens finaux! Bonne continuation!