

WSDL : Décrire et configurer

Mickaël BARON – 2010 (Rév. Janvier 2011)

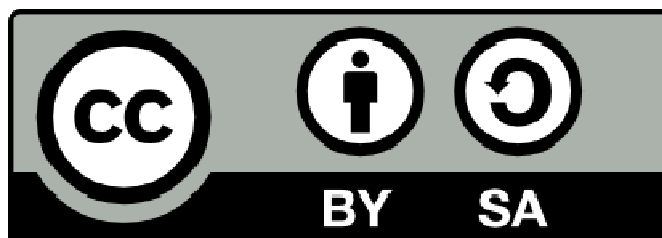
Licence

Creative Commons

Contrat Paternité

Partage des Conditions Initiales à l'Identique

2.0 France



<http://creativecommons.org/licenses/by-sa/2.0/fr>

Plan du cours

- Généralités WSDL
- WSDL par l'exemple : HelloWorld service
- Organisation d'un document WSDL
- Élément *Type*
- Élément *Message*
- Éléments *PortType* et *Operation*
- Élément *Binding*
- Binding SOAP
- Éléments *Service* et *Port*
- Binding HTTP GET & Post

Déroulement du cours

➤ Pédagogie du cours

- Des bulles d'aide tout au long du cours
- Survol des principaux concepts en évitant une présentation exhaustive

➤ Logiciels utilisés

- Navigateur Web 

➤ Pré-requis

- Ingénierie des données
- Schema XML

➤ Remerciements

- TODO



Ceci est une astuce



Ceci est une alerte

Ressources : Liens sur le Web

➤ Articles

 www.w3.org/TR/wsdl

 www.w3.org/TR/wsdl20/

 www.w3.org/2002/ws/desc/

➤ www.ibm.com/developerworks/library/ws-intwsdl/

➤ oreilly.com/catalog/websevrss/chapter/ch06.html

➤ www.ibm.com/developerworks/webservices/library/ws-whichwsdl/

 www.w3.org/XML/Schema

➤ www.relaxng.org/

➤ www.ibm.com/developerworks/java/library/j-jws20

➤ Cours

➤ piacoa.org/publications/teaching/webservices/WSDL.pdf

➤ www.javapassion.com/webservices/WSDLBasics.pdf

➤ www.javapassion.com/webservices/WSDLBinding.pdf

➤ www.w3schools.com/wsdl/default.asp

Ressources : Bibliothèque

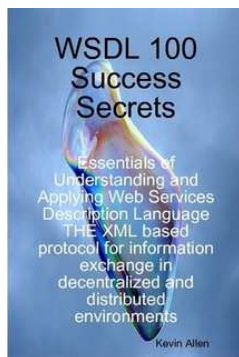


- Services Web avec SOAP, WSDL, UDDI, ebXML
 - Auteur : Jean-Marie Chauvet
 - Éditeur : Eyrolles
 - Edition : Mars 2003 - 524 pages - ISBN : 2212110472



Understanding Web Services
XML, WSDL, SOAP, and UDDI
Eric Newcomer

- Understanding Web Services : XML, WSDL, ...
 - Auteur : Eric Newcomer
 - Éditeur : Addison-Wesley
 - Edition : Mai 2002 - 368 pages - ISBN : 0201750813



- WSDL 100 Success Secrets Essentials of ...
 - Auteur : Kevin Allen
 - Éditeur : Emero Pty Ltd
 - Edition : Juillet 2008 - 144 pages - ISBN : 1921523220

Généralités WSDL

- WSDL est l'acronyme de **W**eb **S**ervice **D**escription **L**anguage
- Basé sur le langage XML et permet de décrire un service Web
- Fournit une description indépendante du langage et de la plate-forme
- Par comparaison WSDL est assez semblable au langage IDL défini par CORBA
- Spécification du W3C
 - WSDL 1.1 : *<http://www.w3.org/TR/wsdl>*
 - WSDL 2.0 : *<http://www.w3.org/TR/wsdl20/>*
- A partir d'un document WSDL il est possible
 - Générer un client pour appeler un Service Web
 - Générer le code pour implémenter un Service Web

Où trouver des documents WSDL

➤ Amazon Associates Web Service

- <https://affiliate-program.amazon.com>
- <http://webservices.amazon.com/AWSECommerceService/AWSECommerceService.wsdl>
- Nécessite la création d'un compte pour l'invocation

➤ ebaY

- <http://developer.ebay.com>
- <http://developer.ebay.com/webservices/finding/latest/FindingService.wsdl>
- Nécessite la création d'un compte pour l'invocation

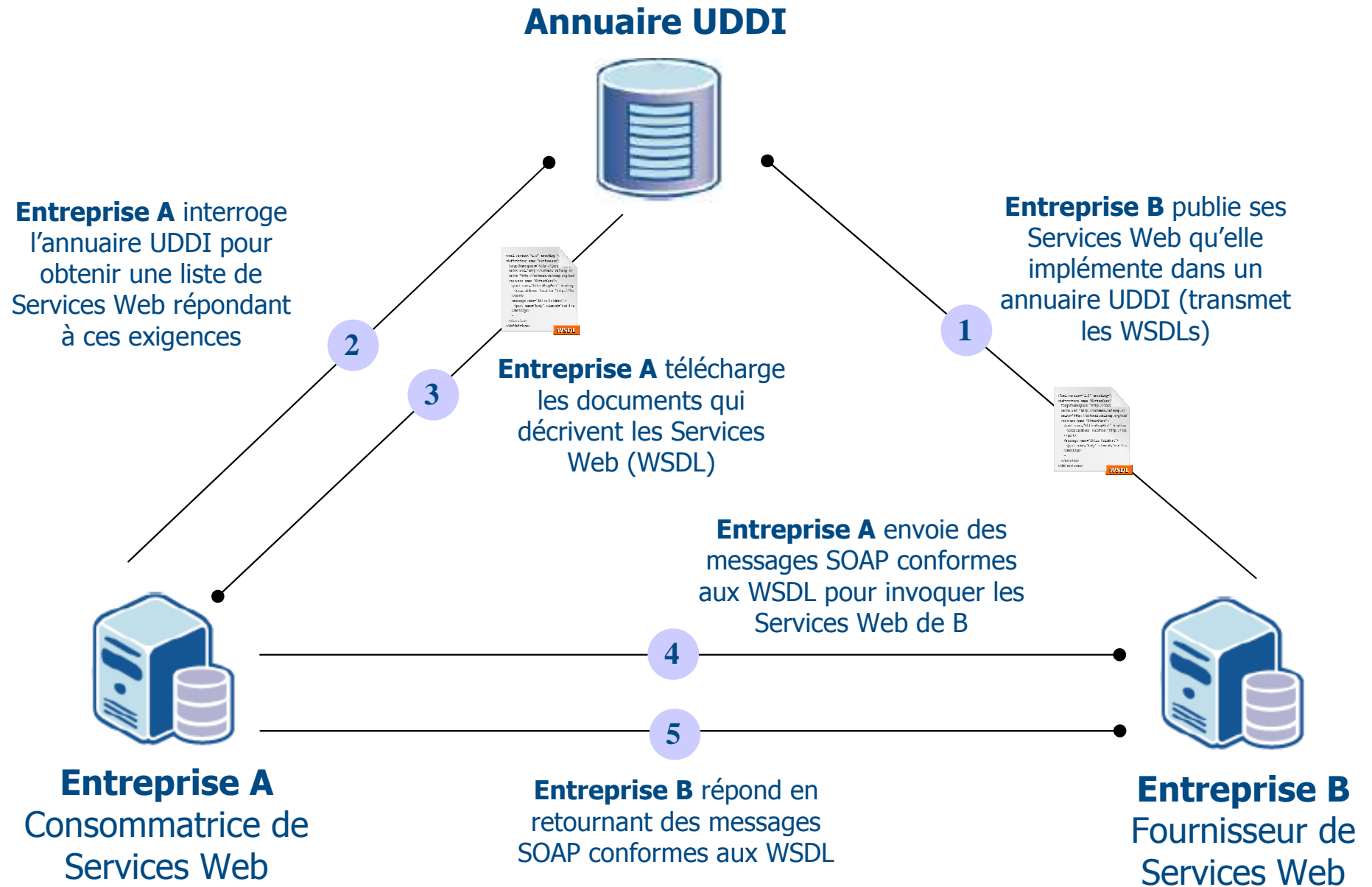
➤ National Oceanic and Atmospheric Administration

- <http://www.nws.noaa.gov/xml/>
- <http://www.weather.gov/forecasts/xml/DWMLgen/wsdl/ndfdXML.wsdl>

➤ WebserviceX.NET

- <http://www.websvcx.net>
- <http://www.websvcx.net/convertMetricWeight.aspx?wsdl>
- <http://www.websvcx.net/GenericNAICS.aspx?wsdl>

Où est utilisé WSDL ?



Concepts d'un document WSDL

- Une **donnée** : information typée
- Un **message** : regroupe un ensemble de données
- Une **opération** : action fournie par le Service Web (~ méthode au sens Java)
- Un **type de port** : ensemble d'action (~ interface au sens Java)
- Un **binding** : définit pour un type de port le protocole utilisé pour transmettre les informations et le format des données
- Un **port** : définit où est localisé le Service Web et le binding à utiliser
- Un **service** : un ensemble de ports

WSDL par l'exemple : service HelloWorld

- Pour introduire la présentation du langage WSDL nous définissons un **Service** *HelloWorld*
- Le service *HelloWorld* fournit deux **opérations**
 - Une **opération** *makeHello* qui prend en paramètre une chaîne de caractères et retourne une chaîne caractères
 - Une **opération** *simpleHello* sans paramètre en entrée et retourne une chaîne de caractères
- L'accès au service est réalisé par l'intermédiaire de messages **SOAP** (étudié en détail dans le prochain cours)
- Le **protocole** utilisé pour l'échange des messages SOAP est **HTTP**
- Le **style** utilisé est du **RPC**

WSDL par l'exemple : service HelloWorld

➤ Exemple : *HelloWorld* service

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<definitions name="HelloWorld"
  targetNamespace="http://helloworldwebservice.lisi.ensma.fr/"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:tns="http://helloworldwebservice.lisi.ensma.fr/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  <types/>
  <message name="makeHelloWorld">
    <part name="value" type="xsd:string"/>
  </message>
  <message name="makeHelloWorldResponse">
    <part name="helloWorldResult" type="xsd:string"/>
  </message>
  <message name="simpleHelloWorld"/>
  <message name="simpleHelloWorldResponse">
    <part name="helloWorldResult" type="xsd:string"/>
  </message>
  <portType name="HelloWorld">
    <operation name="makeHelloWorld">
      <input message="tns:makeHelloWorld"/>
      <output message="tns:makeHelloWorldResponse"/>
    </operation>
    <operation name="simpleHelloWorld">
      <input message="tns:simpleHelloWorld"/>
      <output message="tns:simpleHelloWorldResponse"/>
    </operation>
  </portType>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<definitions name="AktienKurs"
  targetNamespace="http://loca
  xmlns:xsd="http://schemas.xmlsoap.org
  xmlns="http://schemas.xmlsoap.org/wsd
  <service name="AktienKurs">
    <port name="AktienSoapPort" binding
      <soap:address location="http://loc
    </port>
    <message name="Aktie.HoleWert">
      <part name="body" element="xsd:Tra
    </message>
    ...
  </service>
</definitions>
```

WSDL

WSDL par l'exemple : service HelloWorld

► Exemple (suite) : *HelloWorld* service

```
<binding name="HelloWorldPortBinding" type="tns:HelloWorld">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
  <operation name="makeHelloWorld">
    <soap:operation soapAction=""/>
    <input>
      <soap:body use="literal" namespace="http://helloworldwebservice.lisi.ensma.fr/">
    </input>
    <output>
      <soap:body use="literal" namespace="http://helloworldwebservice.lisi.ensma.fr/">
    </output>
  </operation>
  <operation name="simpleHelloWorld">
    <soap:operation soapAction=""/>
    <input>
      <soap:body use="literal" namespace="http://helloworldwebservice.lisi.ensma.fr/">
    </input>
    <output>
      <soap:body use="literal" namespace="http://helloworldwebservice.lisi.ensma.fr/">
    </output>
  </operation>
</binding>
<service name="HelloWorld">
  <port name="HelloWorldPort" binding="tns:HelloWorldPortBinding">
    <soap:address location="TODO"/>
  </port>
</service>
</definitions>
```

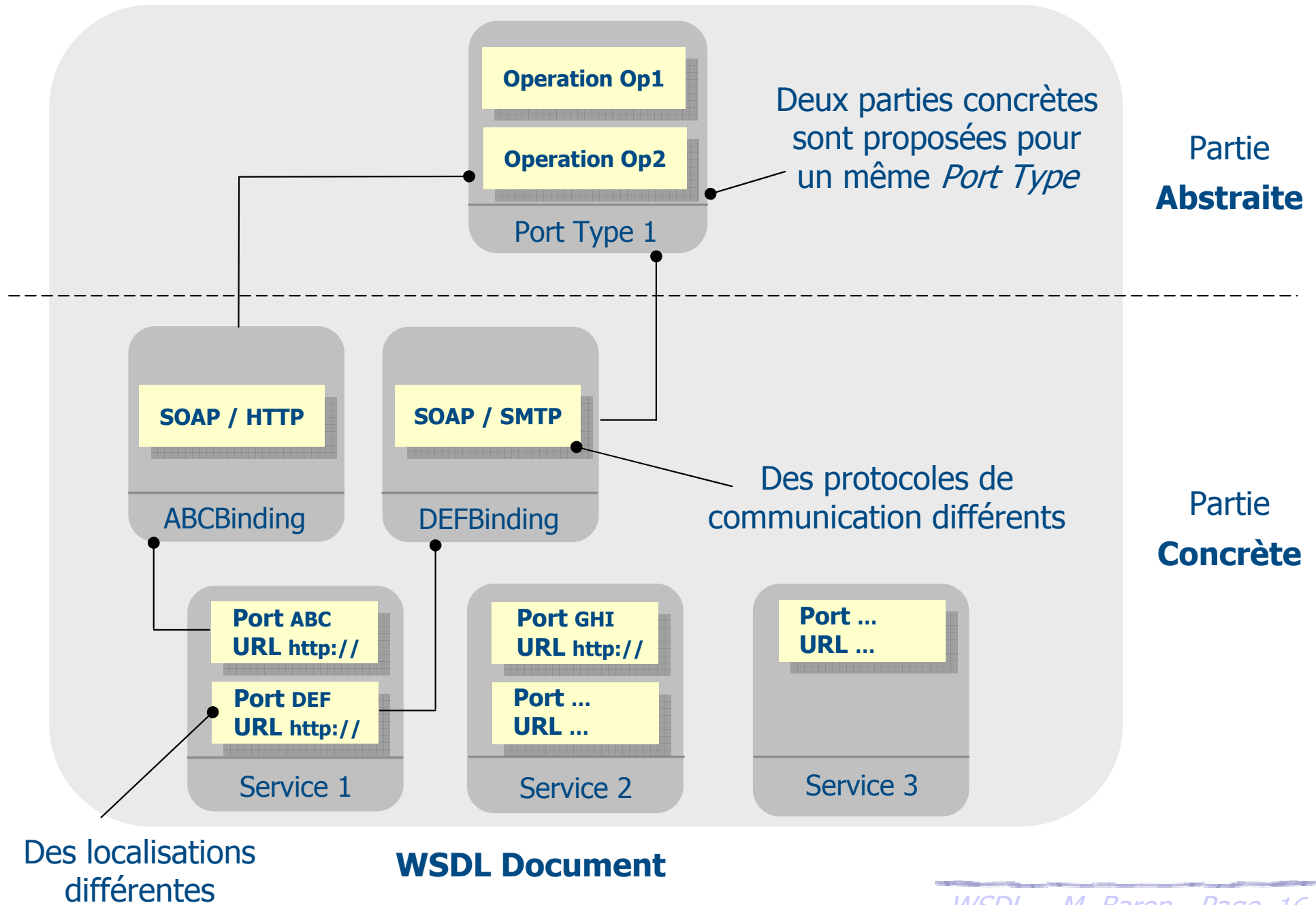
Organisation d'un document WSDL

- **<definitions>**
 - Racine d'un document WSDL
- **<types>** (optionnel et un seul autorisé)
 - Contient la définition des types des données exprimée sous forme de XML Schema
- **<message>** (plusieurs autorisés)
 - Décrit des messages à transmettre (paramètre d'une opération, valeur de retour, exception, ...)
- **<portType>** (plusieurs autorisés)
 - Décrit un ensemble d'opérations où chacune à 0 ou plusieurs messages en entrée, 0 ou plusieurs messages de sortie ou de fautes
- **<binding>** (plusieurs autorisés)
 - Spécifie une liaison entre un *portType* à un protocole (SOAP, HTTP)
- **<service>** (plusieurs autorisés)
 - Regroupe l'ensemble des ports (relation entre *binding* et URL)

Organisation d'un document WSDL

- Un document WSDL est décomposé en deux parties
- **Partie abstraite** qui décrit les messages et les opérations disponibles
 - Types (*<types>*)
 - Messages (*<message>*)
 - Types de port (*<portType>*)
- **Partie concrète** qui décrit le protocole à utiliser et le type d'encodage à utiliser pour les messages
 - Bindings (*<binding>*)
 - Services (*<service>*)
- Plusieurs parties concrètes peuvent être proposées pour la partie abstraite
- Motivation de cette séparation ? Réutilisabilité de la partie abstraite

Organisation d'un document WSDL



WSDL par l'exemple : Carnet d'adresse

- Le service *Notebook* fournit trois opérations
 - Une opération *addPerson* qui prend en paramètre un objet *Person* et retourne un booléen pour indiquer l'état de création
 - Une opération *addPerson* qui prend en paramètre trois chaînes de caractères (*name*, *address* et *birthyear*) sans retour
 - Une opération *getPersonByName* qui prend en paramètre une chaîne de caractère et retourne un objet *Person*
 - Une opération *getPersons* sans paramètre en entrée et qui retourne un tableau d'objets *Person*
- L'accès au service est réalisé par l'intermédiaire de messages SOAP (étudié en détail dans le prochain cours)
- Le protocole utilisé pour l'échange des messages SOAP est HTTP et le style utilisé est du RPC

Elément *Types*

- L'élément *<types>* contient la définition des types utilisés pour décrire la structure des messages échangés par le Web Service
- Le système de typage est généralement un Schema XSD mais d'autres systèmes sont autorisés (RELAX NG par exemple)
- Cet élément peut être facultatif si les types utilisés par les messages sont des types de bases (*Integer, Boolean, ...*)
- Dans le cas de structures complexes (*Person* par exemple) un Schema XML est alors employé
- Un rappel sur le langage Schema XML est disponible
 - Le cours d'Eric Sardet (sardet@ensma.fr)
 - <http://mbaron.developpez.com/divers/schemaxml/>

Elément *Types*

➤ Exemple : Définition des types pour *Notebook* service

```
<definitions
  name="Notebook"
  targetNamespace="http://notebookwebservice.lisi.ensma.fr/"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:tns="http://notebookwebservice.lisi.ensma.fr/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  <types>
    <xsd:schema targetNamespace="http://notebookwebservice.lisi.ensma.fr/">
      <xsd:complexType name="person">
        <xsd:sequence>
          <xsd:element name="address" type="xs:string" minOccurs="0"/>
          <xsd:element name="birthyear" type="xs:string" minOccurs="0"/>
          <xsd:element name="name" type="xs:string" minOccurs="0"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="personArray" final="#all">
        <xsd:sequence>
          <xsd:element name="item" type="tns:person" minOccurs="0" maxOccurs="unbounded" nillable="true"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:schema>
  </types>
  ...
</definitions>
```

Une personne est définie par une *adresse*, une *année de naissance* et un *nom*

Définition d'un type tableau de personne

Elément *Types*

- La définition des types peut également être importée à partir d'un fichier Schema XML
- Le fichier XML est accessible au même titre que le document WSDL
- L'adresse de l'hôte du Schema XML n'est pas forcément la même que celle du document WSDL
- Cette séparation permet
 - de réutiliser des types dans plusieurs WSDL différents
 - d'éviter d'alourdir le document WSDL
- Par la suite nous privilégierons la séparation des types du document WSDL

Elément *Types*

➤ Exemple : Définition des types pour *Notebook* service (bis)

```
<definitions
  targetNamespace="http://notebookwebservice.lisi.ensma.fr/"
  name="Notebook"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  ...>
  <types>
    <xsd:schema>
      <xsd:import namespace="http://notebookwebservice.lisi.ensma.fr/" schemaLocation="Notebook_schema1.xsd"/>
    </xsd:schema>
  </types>
  ...
</definitions>
```

Import le fichier XSD

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0"
  targetNamespace="http://notebookwebservice.lisi.ensma.fr/"
  ...>
  <xs:complexType name="person">
    <xs:sequence>
      <xs:element name="address" type="xs:string" minOccurs="0"/>
      <xs:element name="birthyear" type="xs:string" minOccurs="0"/>
      <xs:element name="name" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="personArray" final="#all">
    <xs:sequence>
      <xs:element name="item" type="tns:person" minOccurs="0" maxOccurs="unbounded" nillable="true"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Élément *Messages*

- L'élément *<message>* permet de décrire les messages échangés par les services
 - Paramètres d'entrées des opérations
 - Paramètres de sorties
 - Exception
- Chaque *<message>* est identifié par un nom (attribut *name*) et est constitué d'un ensemble d'éléments *<part>*
- En quelque sorte un élément *<part>* correspond à un paramètre d'une opération
- Si une opération est décrit par plusieurs paramètres, plusieurs éléments *<part>* seront à définir
- L'élément *<part>* est défini par
 - un nom (attribut *name*)
 - un type (attribut *type*)

Elément *Messages*

➤ Exemple : Définition des messages pour Notebook service

```
<definitions
  targetNamespace="http://notebookwebservice.lisi.ensma.fr/"
  name="Notebook"
  ...>
  <types>
    ...
  </types>
  <message name="addPersonWithComplexType">
    <part name="newPerson" type="tns:person"/>
  </message>
  <message name="addPersonWithComplexTypeResponse">
    <part name="addPersonWithComplexTypeResult" type="xsd:boolean"/>
  </message>
  <message name="addPersonWithSimpleType">
    <part name="name" type="xsd:string"/>
    <part name="address" type="xsd:string"/>
    <part name="birthyear" type="xsd:string"/>
  </message>
  <message name="getPerson">
    <part name="personName" type="xsd:string"/>
  </message>
  <message name="getPersonResponse">
    <part name="getPersonResult" type="tns:person"/>
  </message>
  <message name="getPersons"/>
  <message name="getPersonsResponse">
    <part name="getPersonsResult" type="tns:personArray"/>
  </message>
</definitions>
```

Message utilisé pour l'appel
d'une opération avec une
seule partie

Message utilisé pour le
résultat d'une opération avec
une seule partie

Message utilisé pour l'appel
d'une opération avec trois
parties

Une partie qui pointe sur un type
défini par l'élément *<types>*

Elément *portType* et sous élément *Operation*

- Un élément *<portType>* est un regroupement d'opérations et peut comparé à une interface Java
- Caractéristique d'un élément *<portType>*
 - Identifiable par un nom (attribut *name*)
 - Composé de sous élément *<operation>*
- Une opération est comparable une méthode Java
 - Identifiable par un nom (attribut *name*)
 - La description des paramètres est obtenue par une liste de messages

Elément *portType* et sous élément *Operation*

- Une opération exploite les messages via les sous éléments
 - *<input>* : message transmis au service
 - *<output>* : message produit par le service
 - *<fault>* : message d'erreur (très proche des exceptions)
- Chaque sous élément possède les attributs suivants
 - *name* : nom explicite donné au message (optionnel)
 - *message* : référence à un message (défini précédemment)
- La surcharge d'opération est autorisée sous condition
 - Messages *<input>* et/ou *<output>* soient différents

Elément *portType* et sous élément *Operation*

➤ Exemple : Définition des Ports pour Notebook service

```
<definitions
  targetNamespace="http://notebookwebservice.lisi.ensma.fr/"
  name="Notebook"
  ...>
  <types>
    ...
  </types>
  <message>
    ...
  </message>
  <portType name="Notebook">
    <operation name="addPerson">
      <input message="tns:addPersonWithComplexType"/>
      <output message="tns:addPersonWithComplexTypeResponse"/>
    </operation>
    <operation name="addPerson" parameterOrder="name address birthyear">
      <input message="tns:addPersonWithSimpleType"/>
    </operation>
    <operation name="getPerson">
      <input message="tns:getPerson"/>
      <output message="tns:getPersonResponse"/>
    </operation>
    <operation name="getPersons">
      <input message="tns:getPersons"/>
      <output message="tns:getPersonsResponse"/>
    </operation>
  </portType>
</definitions>
```

L'opération *addPerson*
est surchargée

Possibilité de fixer
l'ordre des paramètres
définis par cette
opération

Elément *portType* et sous élément *Operation*

- Possibilité de définir une opération suivant quatre modèles
- **One-way** : envoi de messages
 - Le client du service envoie un message à l'opération et n'attend pas de réponse
 - Uniquement un seul message utilisé *<input>*

```
<operation name="addPerson" parameterOrder="name address birthyear">  
  <input message="tns:addPersonWithSimpleType"/>  
</operation>
```

- **Request/Response** : question – réponse
 - Le client du service envoie un message à l'opération et un message est retournée au client
 - Un message *<input>*, un message *<output>* et un message *<fault>*

```
<operation name="addPerson">  
  <input message="tns:addPersonWithComplexType"/>  
  <output message="tns:addPersonWithComplexTypeResponse"/>  
</operation>
```

Elément *portType* et sous élément *Operation*

➤ **Notification** : notification

- Le service envoie un message au client
- Uniquement un seul message utilisé *<output>*

```
<operation name="personStatus">  
  <output message="trackingInformation"/>  
</operation>
```

➤ **Solicit - response** : sollicitation - réponse

- Le client reçoit un message du service et répond au service
- Un message *<output>*, un message *<input>* et un message *<fault>*

```
<operation name="clientQuery">  
  <output message="bandWithRequest"/>  
  <input message="bandwidthInfo" />  
  <fault message="faultMessage" />  
</operation>
```

Elément *Binding*

- Un élément *<binding>* permet de réaliser la partie concrète d'un élément *<portType>*
 - un nom (attribut *name*)
 - un *portType* (attribut *type*)
- Il décrit précisément le protocole à utiliser pour manipuler un élément *<portType>*
 - SOAP 1.1 et 1.2
 - HTTP GET & Post (pour le transfert d'images par exemple)
 - MIME
- Plusieurs éléments *<binding>* peuvent être définis de sorte qu'un élément *portType* peut être appelé de différentes manières
- La structure de l'élément *<binding>* dépend du protocole utilisé

Elément *Binding*

- Structure générale de l'élément *<binding>* sans précision sur le protocole employé

```
<definitions>
  ...
  <binding name="NamePortBinding" type="tns:portType">
    <!-- Décrit le protocole à utiliser -->
    <operation name="operation1">
      <!-- Action du protocole sur l'opération -->
      <input>
        <!-- Action du protocole sur les messages d'entrés (input) -->
      </input>
      <output>
        <!-- Action du protocole sur les messages de sorties (ouput) -->
      </output>
      <fault>
        <!-- Action du protocole sur les messages d'erreurs (fault) -->
      </fault>
    </operation>
  </binding>
  ...
</definitions>
```

Ces informations sont spécifiques au protocole utilisé

- Le schema XML de WSDL ne décrit pas les sous éléments de *binding*, *operation*, *input*, *ouput* et *fault*
- Ces éléments sont spécifiques aux protocoles utilisés

Binding SOAP

- Le binding SOAP est défini par l'espace de noms suivant
 - *<http://schemas.xmlsoap.org/wsdl/soap/>*
 - Préfixe utilisé est généralement *soap* (de la forme *<soap:binding>*)
- Différentes versions peuvent être utilisées : 1.1 et 1.2 (dans la suite nous utiliserons la version 1.1)
- Les principales balises à exploiter dans le binding sont
 - *<soap:binding>*
 - *<soap:operation>*
 - *<soap:body>*, *<soap:header>*, *<soap:headerfault>*
 - *<soap:fault>*
- A noter que nous détaillerons certains aspects dans le cours consacré au protocole SOAP

Binding SOAP

➤ Exemple : Définition d'un Binding SOAP 1.1

```
<definitions ...>
  <!-- Définition de la partie Abstraite du WSDL -->
  <binding name="NoteBookPortBinding" type="tns:Notebook">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
    <operation name="addPersonWithComplexType">
      <soap:operation soapAction=""/>
      <input>
        <soap:body use="literal" namespace="http://notebookwebservice.lisi.ensma.fr/">
      </input>
      <output>
        <soap:body use="literal" namespace="http://notebookwebservice.lisi.ensma.fr/">
      </output>
    </operation>
    <operation name="addPersonWithSimpleType">
      <soap:operation soapAction=""/>
      <input>
        <soap:body use="literal" namespace="http://notebookwebservice.lisi.ensma.fr/">
      </input>
    </operation>
    <operation name="getPerson">
      <soap:operation soapAction=""/>
      <input>
        <soap:body use="literal" namespace="http://notebookwebservice.lisi.ensma.fr/">
      </input>
      <output>
        <soap:body use="literal" namespace="http://notebookwebservice.lisi.ensma.fr/">
      </output>
    </operation>
    ...
  </binding>
</definitions>
```

Éléments spécifiques au
protocole SOAP 1.1

Binding SOAP : Élément *soap:binding*

- L'élément *<soap:binding>* doit être présent lors de la définition d'un binding à base de messages SOAP

```
<definitions ...>
  <!-- Définition de la partie Abstraite du WSDL -->
  <binding ...>
    <soap:binding transport="uri"? style="rpc|document"/>
  </binding>
</definitions>
```

- L'attribut *style* permet d'indiquer la façon dont sont créés les messages SOAP pour l'ensemble des opérations
 - *rpc* : encodage défini par SOAP RPC
 - *document* : encodage sous forme d'élément XML
- L'attribut *transport* permet de préciser le protocole à utiliser pour le transport des messages SOAP
 - HTTP : *http://schemas.xmlsoap.org/soap/http*
 - SMTP :
 - FTP, ...

Binding SOAP : Élément *soap:operation*

- L'élément *<soap:operation>* doit être présent pour chaque opération définie dans la partie abstraite du document

```
<definitions ...>
  <!-- Définition de la partie Abstraite du WSDL -->
  <binding ...>
    <operation ...>
      <soap:operation soapAction="uri"? style="rpc|document"?>
    </operation>
  </binding>
</definitions>
```

- L'attribut *soapAction* permet de préciser la valeur de l'en-tête HTTP (dans notre cas la valeur sera vide)
- L'attribut *style* permet de préciser la façon dont sont créés les messages SOAP de l'opération en question (RPC ou document)

Binding SOAP : Élément *soap:body*

- L'élément `<soap:body>` précise le format des messages échangés par une opération
- Il y a autant d'élément `<soap:body>` qu'il y a de messages définis par une opération (`<input>`, `<output>` et `<fault>`)

```
<definitions ...>
  <!-- Définition de la partie Abstraite du WSDL -->
  <binding ...>
    <operation ...>
      <input>
        <soap:body parts="nmtokens"? use="literal|encoded"? encodingStyle="uri-list"? namespace="uri"?>
      </input>
      <output>
        <soap:body parts="nmtokens"? use="literal|encoded"? encodingStyle="uri-list"? namespace="uri"?>
      </output>
    </operation>
  </binding>
</definitions>
```

- L'attribut *use* caractérise la forme des parties des messages
 - *encoded* : transformation suivant un mécanisme défini par l'attribut *encodingStyle*
 - *literal* : pas de transformation des parties des messages, elles apparaissent directement

Binding SOAP : debriefing

- Beaucoup de chose très abstraites qui seront illustrées dans le support de cours sur SOAP
- Rapide mise au point ...
 - Le style *document* est largement utilisé par la plateforme .NET
 - Le choix entre *RPC* et *Document* n'a pas d'importance puisque le résultat peut être identique (historique du standard SOAP)
 - L'utilisation de la valeur *encoded* (forme des parties des messages) n'est pratiquement plus supportée par les boîtes à outils Web Services
 - Le protocole HTTP est massivement utilisé pour le transport des messages SOAP

Élément *Service* et *Port*

- Un élément service définit l'ensemble des points d'entrée du Service Web, en regroupant des éléments *<port>*
- L'élément *<port>* permet de spécifier une adresse pour un binding donné
- Un port est défini par deux attributs
 - *name* : nom du port
 - *binding* : nom du binding (défini précédemment)
- Le corps de l'élément *<port>* est spécifique au protocole utilisé pour définir le binding
- Dans le cas d'un binding de type SOAP, un élément *<soap:address>* précise l'URI du port
- Il est par conséquent possible d'appeler un service à des endroits différents (plusieurs éléments *port*)

Élément *Service* et *Port*

➤ Exemple : Définition d'un service

```
<definitions ...>
  <!-- Définition de la partie Abstraite du WSDL -->

  <binding ...>
  </binding>
  <service name="Notebook">
    <port name="NoteBookPort" binding="tns:NoteBookPortBinding">
      <soap:address location="http://localhost:8080/NotebookWebService/notebook"/>
    </port>
  </service>
</definitions>
```

Le Port Type *Notebook* est accessible
en SOAP/HTTP via cette URL

Autre chose que du SOAP comme transport

➤ Exemple : WSDL qui retourne un GIF ou JPG

```
<definitions ....>
  <message name="m1">
    <part name="part1" type="xsd:string"/>
    <part name="part2" type="xsd:int"/>
    <part name="part3" type="xsd:string"/>
  </message>
  <message name="m2">
    <part name="image" type="xsd:binary"/>
  </message>

  <portType name="pt1">
    <operation name="o1">
      <input message="tns:m1"/>
      <output message="tns:m2"/>
    </operation>
  </portType>

  <service name="service1">
    <port name="port1" binding="tns:b1">
      <http:address location="http://example.com/">
    </port>
    <port name="port2" binding="tns:b2">
      <http:address location="http://example.com/">
    </port>
    <port name="port3" binding="tns:b3">
      <http:address location="http://example.com/">
    </port>
  </service>
```

Exemple situé sur

http://www.w3.org/TR/wsdl#_http-e

Autre chose que du SOAP comme transport

► Exemple (suite) : WSDL qui retourne un GIF ou JPG

```
<binding name="b1" type="pt1">
  <http:binding verb="GET"/>
  <operation name="o1">
    <http:operation location="o1/A(part1)B(part2)/(part3)"/>
    <input><http:urlReplacement/></input>
    <output>
      <mime:content type="image/gif"/>
      <mime:content type="image/jpeg"/>
    </output>
  </operation>
</binding>
<binding name="b2" type="pt1">
  <http:binding verb="GET"/>
  <operation name="o1">
    <http:operation location="o1"/>
    <input><http:urlEncoded/></input>
    <output>
      <mime:content type="image/gif"/>
      <mime:content type="image/jpeg"/>
    </output>
  </operation>
</binding>
<binding name="b3" type="pt1">
  <http:binding verb="POST"/>
  <operation name="o1">
    <http:operation location="o1"/>
    <input><mime:content type="application/x-www-form-urlencoded"/></input>
    <output>
      <mime:content type="image/gif"/>
      <mime:content type="image/jpeg"/>
    </output>
  </operation>
</binding>
</definitions>
```

Exemple situé sur
http://www.w3.org/TR/wsdl#_http-e

Outils

- Des outils pour construire un document WSDL
 - Notepad++ (éditeur de texte puisqu'il s'agit d'XML)
 - Eclipse JavaEE
 - Netbeans
 - Visual Studio
 - ... (tous les environnements de développement qui manipulent les Services Web)
- Des outils pour valider un document WSDL
 - *www.validwsdl.com*
- Des outils pour manipuler un WSDL
 - SOAPUI