SOA – Services web étendus

WSDL : Décrire et configurer

Mickaël BARON – 2010 (Rev. Février 2019)
mailto:baron.mickael@gmail.com ou mailto:baron@ensma.fr

@mickaelbaron mickael-baron.fr

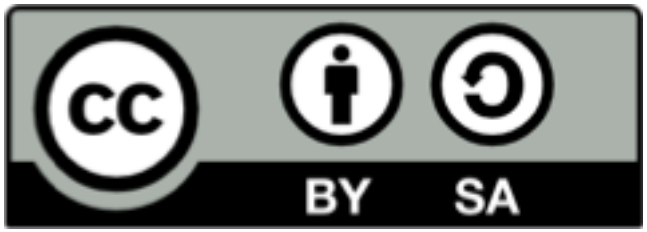
Licence

Creative Commons

Contrat Paternité

Partage des Conditions Initiales à l'Identique

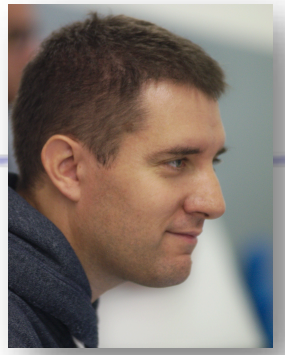
2.0 France



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

@mickaelbaron
mickael-baron.fr

À propos de l'auteur …



- **Mickaël BARON**
- Ingénieur de Recherche au LIAS
  - *https://www.lias-lab.fr*
  - Equipe : Ingénierie des Données et des Modèles
  - Responsable des plateformes logicielles, « coach » technique
- Responsable Rubriques Java de Developpez.com
  - Communauté Francophone dédiée au développement informatique
  - *https://java.developpez.com*
  - 4 millions de visiteurs uniques et 12 millions de pages vues par mois
  - 750 00 membres, 2 000 forums et jusqu'à 5 000 messages par jour

@mickaelbaron

mickael-baron.fr

@mickaelbaron
mickael-baron.fr

Plan du cours

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

@mickaelbaron
mickael-baron.fr

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

Ceci est une astuce

Ceci est une alerte 

mickael-baron.fr @mickaelbaron

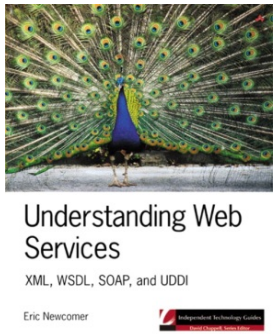
Ressources : liens sur le Web

- Articles
  - W3C *www.w3.org/TR/wsdl*
  - W3C *www.w3.org/TR/wsdl20/*
  - W3C *www.w3.org/2002/ws/desc/*
    - *www.ibm.com/developerworks/library/ws-intwsdl/*
    - *oreilly.com/catalog/webservess/chapter/ch06.html*
    - *www.ibm.com/developerworks/webservices/library/ws-whichwsdl/*
  - W3C *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*

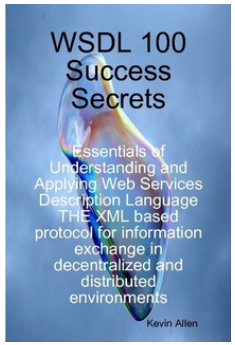
mickael-baron.fr @mickaelbaron

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…
  - 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

mickael-baron.fr @mickaelbaron

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

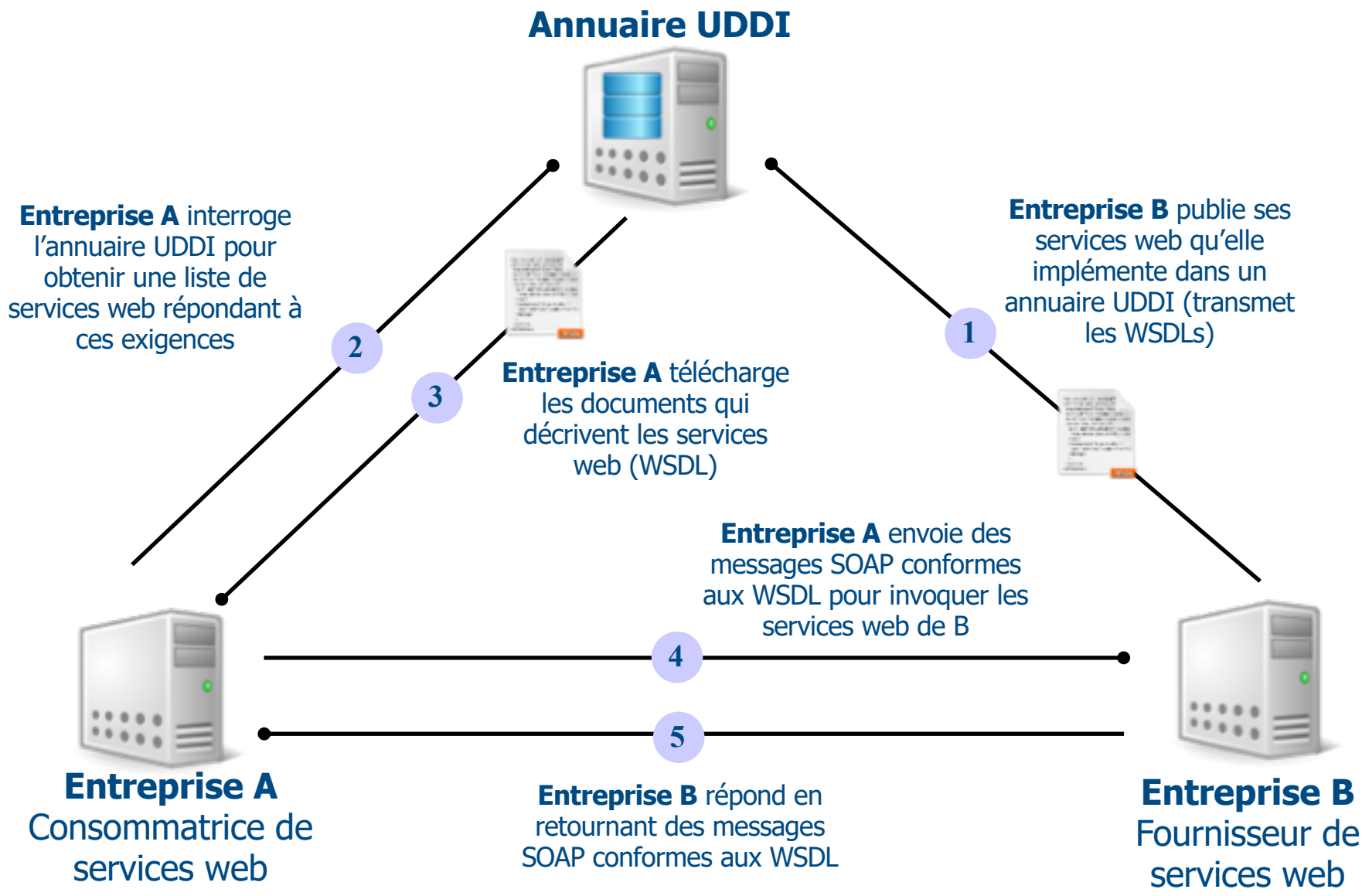
mickael-baron.fr @mickaelbaron

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.webservicex.net*
  - *http://www.webservicex.net/convertMetricWeight.asmx?wsdl*
  - *http://www.webservicex.net/GenericNAICS.asmx?wsdl*

mickael-baron.fr @mickaelbaron

Où est utilisé WSDL ?



@mickaelbaron
mickael-baron.fr

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'actions (~ 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

mickael-baron.fr @mickaelbaron

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**

mickael-baron.fr @mickaelbaron

WSDL par l'exemple : service HelloWorld

Exemple : *HelloWorld* service

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<definitions name="HelloWorldService"
     targetNamespace="http://helloworldwebservice.mickaelbaron.fr/"
     xmlns="http://schemas.xmlsoap.org/wsdl/"
     xmlns:tns="http://helloworldwebservice.mickaelbaron.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="HelloWorldService">
    <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>
```

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.mickaelbaron.fr/"/>
     </input>
     <output>
       <soap:body use="literal" namespace="http://helloworldwebservice.mickaelbaron.fr/"/>
     </output>
   </operation>
   <operation name="simpleHelloWorld">
     <soap:operation soapAction=""/>
     <input>
       <soap:body use="literal" namespace="http://helloworldwebservice.mickaelbaron.fr/"/>
     </input>
     <output>
       <soap:body use="literal" namespace="http://helloworldwebservice.mickaelbaron.fr/"/>
     </output>
   </operation>
 </binding>
 <service name="HelloWorldService">
   <port name="HelloWorldPort" binding="tns:HelloWorldPortBinding">
     <soap:address location="TODO"/>
   </port>
 </service>
</definitions>
```


mickael-baron.fr @mickaelbaron

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)

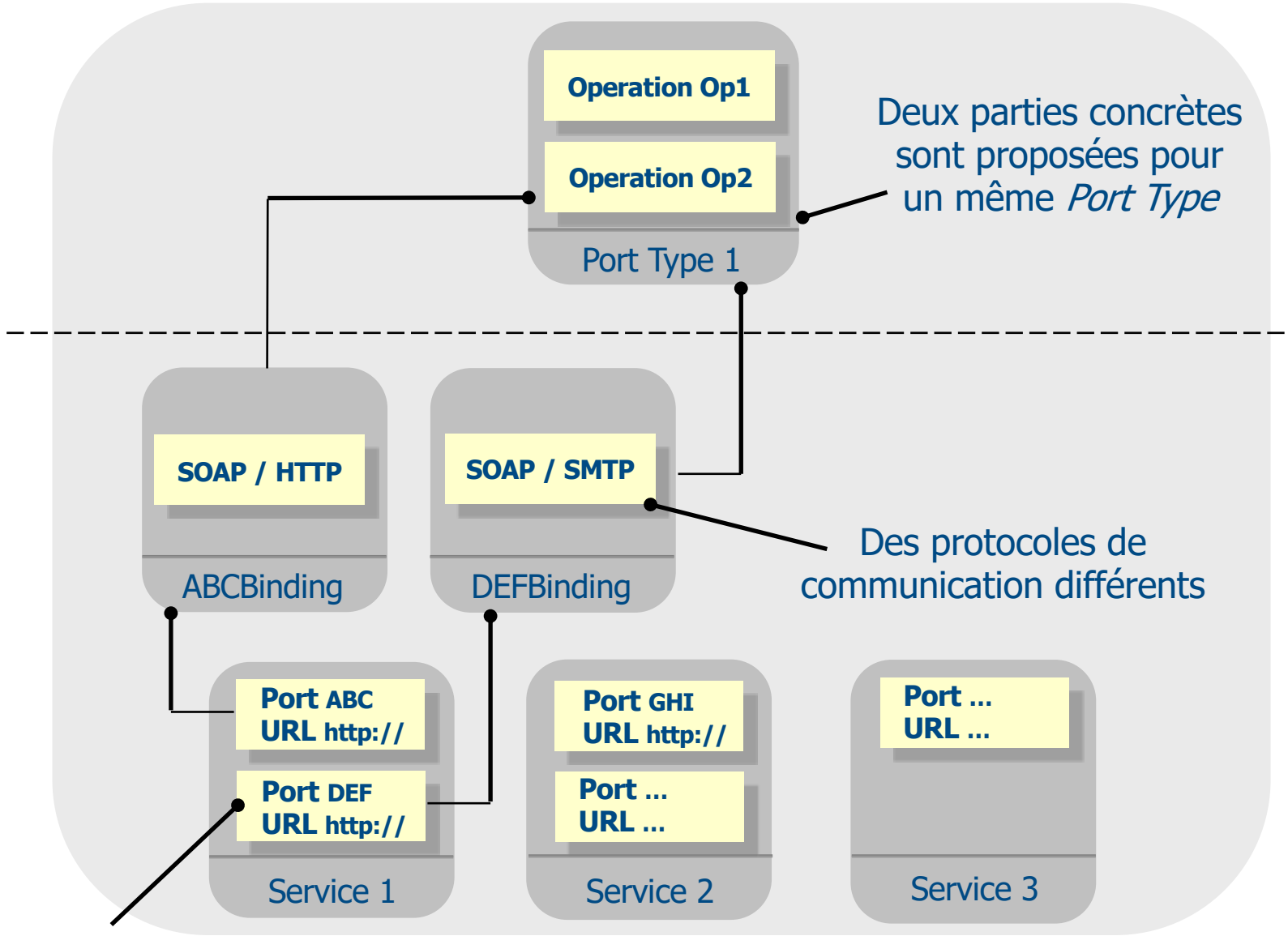
@mickaelbaron
mickael-baron.fr

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

mickael-baron.fr @mickaelbaron

Organisation d'un document WSDL



@mickaelbaron
mickael-baron.fr

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 chaines 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

mickael-baron.fr @mickaelbaron

Elément *Types*

- L'élément *<types>* contient la définition des types utilisés pour décrire la structure des messages échangés par le service web
- 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 de Yassine OUHAMMOU
  - *http://mbaron.developpez.com/divers/schemaxml*

mickael-baron.fr @mickaelbaron

Elément *Types*

- Exemple : définition des types pour *Notebook* service

```
<definitions
      xmlns:tns="http://notebookwebservice.mickaelbaron.fr/"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      name="Notebook"
      targetNamespace="http://notebookwebservice.mickaelbaron.fr/"
      xmlns="http://schemas.xmlsoap.org/wsdl/"
      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
      <types>
            <xsd:schema targetNamespace="http://notebookwebservice.mickaelbaron.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

mickael-baron.fr @mickaelbaron

Elément *Types*

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

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

Elément *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*)

@mickaelbaron
mickael-baron.fr

Elément *Messages*

- Exemple : définition des messages pour Notebook service

```
<definitions
      targetNamespace="http://notebookwebservice.mickaelbaron.fr/" name="NotebookService" ...>
      <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

mickael-baron.fr @mickaelbaron

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 *<ouput>* soient différents

@mickaelbaron
mickael-baron.fr

Elément *portType* et sous élément *Operation*

- Exemple : définition des Ports pour Notebook service

```
<definitions
      targetNamespace="http://notebookwebservice.mickaelbaron.fr/"
      name="Notebook"
      ...>
      <types>
            ...
      </types>
      <message>
            ...
      </message>
      <portType name="NotebookService">
            <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** : envoie 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 *<ouput>*, un message *<input>* et un message *<fault>*

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


mickael-baron.fr @mickaelbaron

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

@mickaelbaron
mickael-baron.fr

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

mickael-baron.fr @mickaelbaron

Elément *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*)

mickael-baron.fr @mickaelbaron

Elément *Service* et *Port*

Exemple : définition d'un service

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

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

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

@mickaelbaron
mickael-baron.fr

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

mickael-baron.fr @mickaelbaron

Outils

- Des outils pour construire un document WSDL
  - VS Code, Notepad++ (éditeur de texte puisqu'il s'agit d'XML)
  - Eclipse 
  - Netbeans 
  - IntelliJ 
  - Visual Studio 
  - … (tous les environnements de développement qui manipulent les services web)
- Des outils pour valider un document WDSL
  - *coveloping.com/tools/wsdl-validator*
- Des outils pour manipuler un WSDL
  - SoapUI

mickael-baron.fr @mickaelbaron