Université de Bouira
Département d'Informatique
Module : Les Architectures Orientées Services
Niveau M1-ISIL

Série TD / TP 4 - 5 (2 séances)

Styles de Binding/Encodage, Schema Types, et Annotations JAXB

I- Encodage des messages SOAP

Un message SOAP peut être encodé de plusieurs façons en fonction de son *style* et de son *type d'encodage*. Ces derniers permettent de définir comment les données seront sérialisées et désérialisées dans les requêtes et les réponses des messages SOAP. [1]

Il existe deux styles de communication utilisés pour traduire un WSDL Binding vers un corps de message SOAP : **RCP** et **Document**.
En plus du style, il existe deux types d'encodages : **Encoded** et **Literal**. Les règles d'encodage (Encoding rules) précisent les mécanismes de sérialization des données dans un message.

Le style RPC est parfaitement structuré alors que le type Document n'a pas de structure imposée mais son contenu peut être facilement validé grâce à un schéma XML ou traité puisque c'est un document XML. Avec le style document, il est donc possible de structurer librement le corps du message grâce au schéma XML.

Style	Description
RPC	Les messages contiennent le nom et les paramètres de l'opération ;
Document	Les messages ne contiennent pas le nom de l'opération ; Un document XML en entrée et en retour.

Dans le style RPC, les types de données échangées sont définis directement dans le fichier WSDL dans les éléments <message>, alors que dans le style Document, la définition se fait dans un document schéma XML plus détaillé, accessible depuis l'url de schemaLocation dans l'élément WSDL <types>.

Le type d'encodage Literal propose que le contenu du corps (body) soit conforme et validé par un schéma XML donné : chaque élément qui correspond à un paramètre ou à la valeur de retour est décrit dans un schéma XML.
Le type d'encodage Encoded utilise un ensemble de règles reposant sur les types de données des schémas XML pour encoder les données mais le message ne respecte pas de schéma particulier : chaque élément qui correspond à un paramètre ou à la valeur de retour contient la description de la donnée sous la forme d'attributs spécifiés dans la norme Soap.

Le style RPC/Encoded a largement été utilisé au début des services web : actuellement ce style est en cours d'abandon par l'industrie au profit du style Document/Literal.

Exemple – Convertisseur Web Service

Style RPC Literal

- Le Fichier WSDL

[1] http://www.jmdoudoux.fr/java/dej/chap-service-web.htm

```
<definitions xmlns:wsu=http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd … >
      <types/>
      <message name="getDinarFromEuro">
            <part name="arg0" type="xsd:double"/>
      </message>
      <message name="getDinarFromEuroResponse">
            <part name="return" type="xsd:double"/>
      </message>
      <message name="getEuroFromDinar">
            <part name="arg0" type="xsd:double"/>
      </message>
      <message name="getEuroFromDinarResponse">
            <part name="return" type="xsd:double"/>
      </message>
.
.
.
</definitions>
```

Style Document Literal

- Le Fichier WSDL

```
<definitions xmlns:wsu=http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd … >
      <types>
            <xsd:schema>
                  <xsd:import namespace="http://convertisseur.hedia.org/"
                   schemaLocation="http://user:8080/ConvertisseurWebSer
                   vice/ConvertisseurServiceImplService?xsd=1"/>
            </xsd:schema>
      </types>
      <message name="getDinarFromEuro">
            <part name="parameters" element="tns:getDinarFromEuro"/>
      </message>
      <message name="getDinarFromEuroResponse">
            <part name="parameters" element="tns:getDinarFromEuroRespons
            e"/>
      </message>
.
.
.
</definitions>
```

- Le Schéma XML :

```
<xs:schema xmlns:tns="http://convertisseur.hedia.org/" xmlns:xs="http://
www.w3.org/2001/XMLSchema" version="1.0" targetNamespace="http://convert
isseur.hedia.org/">
      <xs:element name="getDinarFromEuro" type="tns:getDinarFromEuro"/>
      <xs:element name="getDinarFromEuroResponse" type="tns:getDinarFrom
      EuroResponse"/>
      <xs:element name="getEuroFromDinar" type="tns:getEuroFromDinar"/>
      <xs:element name="getEuroFromDinarResponse" type="tns:getEuroFromD
      inarResponse"/>
```

```
      <xs:complexType name="getDinarFromEuro">
            <xs:sequence>
                  <xs:element name="arg0" type="xs:double"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name="getDinarFromEuroResponse">
            <xs:sequence>
                  <xs:element name="return" type="xs:double"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name="getEuroFromDinar">
            <xs:sequence>
                  <xs:element name="arg0" type="xs:double"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name="getEuroFromDinarResponse">
            <xs:sequence>
                  <xs:element name="return" type="xs:double"/>
            </xs:sequence>
      </xs:complexType>
</xs:schema>
```

II- Configurer le Style de Binding

Pour déterminer le style et le type d'encodage des messages et réponses SOAP, l'annotation **SOAPBinding** de javax.jws.soap est utilisée comme suit :

Attribut	Rôle
Style style	Définir le style d'encodage du message. Style est une énumération qui contient DOCUMENT et RPC. Par défaut: DOCUMENT
Use use	Définir le format du message. Use est une énumération qui contient ENCODED et LITERAL. Par défaut: LITERAL
ParameterStyle parameterStyle	Définir si les paramètres forment le contenu du message ou s'ils sont encapsulés par un tag du nom de l'opération à invoquer. ParameterStyle est une énumération qui contient BARE et WRAPPED. BARE ne peut être utilisé qu'avec le style DOCUMENT Par défaut: WRAPPED

Exemple d'utilisation

```
@WebService
@SOAPBinding(style=Style.DOCUMENT, use=Use.LITERAL,
       parameterStyle=ParameterStyle.BARE)
public class MonService {
      @WebMethod
      public void MonOperation() {  }
}
```

- La différence entre le style **RPC** et le style **Document** se voit notamment dans le corps des messages de réponses SOAP.

Exemple - Person Web Service

- **Style RPC**

```
@WebService
@SOAPBinding(style=Style.RPC)
public interface PersonService {
      public String getPersonId(String personName);
      public ArrayList getPersonIdList(ArrayList personNameList);
}
```

La réponse SOAP pour la seconde opération aura un résultat vide et ressemblera à :

```
<S:Enveloppe xmlns:S=”http://schemas.xmlsoap.org.soap/envelope/”>
      <S :Body>
            <ns2:getPersonIdListResponse
                              xmlns:ns2="http://tp4.ws.soa.org/">
                  <return/>
            </ns2:getPersonIdListResponse>
      </S:Body>
</S:Envelope>
```

- **Style Document**

La réponse SOAP à la même opération avec le style Document retournera l’ArrayList :

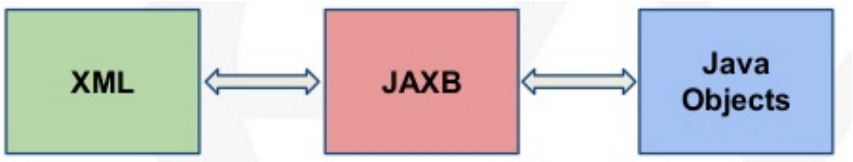
```
<S:Enveloppe xmlns:S=”http://schemas.xmlsoap.org.soap/envelope/”>
      <S :Body>
            <ns2:getPersonIdListResponse
                              xmlns:ns2="http://tp4.ws.soa.org/">
                  <return>Valeur1<return/>
                  <return>Valeur2<return/>
                  <return>Valeur3<return/>
            </ns2:getPersonIdListResponse>
      </S:Body>
</S:Envelope>
```

Exercice Pratique

- Ecrire et déployer le service web Convertisseur en style RPC puis en Document.
- Remarquer et noter les différences dans le fichier WSDL.
- Ecrire et déployer le service web PersonService de l’exemple ci-dessus.
- Tester les deux styles de binding RPC et Document en utilisant SoapUI et remarquer les différences dans les body des messages de réponse SOAP.

III- Annotation JAXB

JAXB (Java Architecture for XML Binding) est une API permettant de manipuler de l'XML en Java. JAXB est donc capable de sérialiser des objets Java en un fichier XML et de les désérialiser (i.e. Marshalling/Unmarshalling), et de générer un schéma XML à partir d'une classe Java et vice versa. Il offre notamment des fonctionnalités de validation.



Dans les Java-based web service Soap, c'est l'API JAXB qui se charge de générer les documents XML décrivant le service web tels que le fichier WSDL, les schémas XML et les messages Soap.

Afin de pouvoir les personnaliser, JAXB offre un ensemble d'**annotations**. Ces annotations définies par JAXB se posent sur les différents éléments d'une classe : son nom, ses champs, ses getters , ou le nom de son package. [2]

Quelques annotations posées sur une classe

@XmlRootElement : associe la classe annotée avec un nœud racine d'un document XML. Cette annotation est suffisante pour générer un document XML à partir d'une instance de cette classe.
@XmlType : permet plusieurs choses. Le point principal est son attribut *propOrder*, qui permet de fixer l'ordre dans lequel les champs de cette classe doivent être enregistrés dans le document XML.

Quelques annotations posées sur un champs ou getter

@XmlElement : permet d'associer un champ ou un *getter* à un nœud d'un document XML. Il permet de spécifier le nom de cet élément, son espace de nom, sa valeur par défaut, etc.

@XmlTransient : Cette annotation posée sur un champ ou un *getter* le retire des éléments pris en compte pour la création des schémas et des documents XML.

Exemple d'utilisation – ShopCatalaog Web Service

En **annotant** comme ci-dessous la **classe** Product du service web ShoCatalog implémenté dans la série 3 comme, des changements dans le **schéma XML** et les messages SOAP seront observés :

- **La Classe Product annotée**

[2] http://blog.paumard.org/cours/jaxb-rest/chap02-jaxb-annotations.html

```
@XmlRootElement(name=”Produit”)
@XmlType(propOrder = {prix, id, categorie})
Public class Product {
      String id ;
      String categorie ;
      double prix ;
      double remise ;

      @XmlElement(name=”identif”)
      public String getId( ) {
            return id;
      }

      @XmlTransient
      public double getRemise( ) {
            return remise;
      }
}
```

- **Le schéma XML généré – Opération Binding**

```
<xs:schema xmlns:tns="http://tp3.ws.soa.org/" xmlns:xs="http://www.w3.or
g/2001/XMLSchema" version="1.0" targetNamespace="http://tp3.ws.soa.org/"
>
      <xs:element name="Produit" type="tns:product"/>
      <xs:element name="getProductById" type="tns:getProductById"/>
      <xs:element name="getProductByIdResponse" type="tns:getProductById
      Response"/>
  ...
      <xs:complexType name="getProductById">
            <xs:sequence>
                  <xs:element name="produitID" type="xs:string" minOccurs
                  ="0"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name="getProductByIdResponse">
            <xs:sequence>
                  <xs:element name="produitRetourne" type="tns:product" m
                  inOccurs="0"/>
            </xs:sequence>
      </xs:complexType>
  ...
      <xs:complexType name="product">
            <xs:sequence>
                  <xs:element name="prix" type="xs:double"/>
                  <xs:element name="identif" type="xs:string" minOccurs="
                  0"/>
                  <xs:element name="categorie" type="xs:string" minOccurs
                  ="0"/>
            </xs:sequence>
      </xs:complexType>

      .
      .
      .
</xs:schema>
```

- **Le message SOAP response sérialisé – Opération sérialisation**

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:getProductByIdResponse xmlns:ns2="http://tp3.ws.soa.org/">
         <produitRetourne>
            <prix>500.0</prix>
            <identif>prodV4</identif>
            <categorie>Disque Vinyle</categorie>
         </produitRetourne>
      </ns2:getProductByIdResponse>
   </S:Body>
</S:Envelope>
```

IV- Exercice

- Ecrire et déployer le service web Soap décrit par ce schéma XML.

```
<xs:schema xmlns:tns="http://tp4.ws.soa/" xmlns:xs="http://www.w3.org/2001/
XMLSchema" version="1.0" targetNamespace="http://tp4.ws.soa/">
      <xs:element name="Personne" type="tns:personne"/>
      <xs:element name="getAddresByName" type="tns:getAddresByName"/>
      <xs:element name="getAddresByNameResponse" type="tns:getAddresByNameR
      esponse"/>
      <xs:element name="getPersons" type="tns:getPersons"/>
      <xs:element name="getPersonsResponse" type="tns:getPersonsResponse"/>
      <xs:complexType name="getAddresByName">
            <xs:sequence>
                  <xs:element name="nom" type="xs:string" minOccurs="0"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name="getAddresByNameResponse">
            <xs:sequence>
                  <xs:element name="adresseRetournee" type="tns:adresse" mi
                  nOccurs="0"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name="adresse">
            <xs:sequence>
                  <xs:element name="pays" type="xs:string" minOccurs="0"/>
                  <xs:element name="email" type="xs:string" minOccurs="0"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name="getPersons">
            <xs:sequence/>
      </xs:complexType>
      <xs:complexType name="getPersonsResponse">
            <xs:sequence>
                  <xs:element name="listePersonnes" type="tns:personne" min
                  Occurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name="personne">
            <xs:sequence>
                  <xs:element name="NomPersonne" type="xs:string" minOccurs
                  ="0"/>
                  <xs:element name="AdressePersonne" type="tns:adresse" min
                  Occurs="0"/>
            </xs:sequence>
      </xs:complexType>
</xs:schema>
```

- Ecrire et déployer le service web Saop décrit par ce fichier WSDL, ainsi que le message Soap en réponse à une de ses opérations.

```
<definitions  ...  name="BookShelfService">
	<types>
		<xsd:schema>
			<xsd:import namespace="http://tp5.ws.soa.org/" schemaLocation="http:/
			/user:8080/BookShelfWebService/BookShelfService?xsd=1"/>
		</xsd:schema>
	</types>
	<message name="getBook">
		<part name="title" type="xsd:string"/>
	</message>
	<message name="getBookResponse">
		<part name="ReturnedBook" type="tns:bookVO"/>
	</message>
	<message name="isBookAvailable">
		<part name="title" type="xsd:string"/>
	</message>
	<message name="isBookAvailableResponse">
		<part name="return" type="xsd:boolean"/>
	</message>
	<portType name="BookShelf">
		<operation name="getBook">
			<input wsam:Action="http://tp5.ws.soa.org/BookShelf/getBookRequest" m
			essage="tns:getBook"/>
			<output wsam:Action="http://tp5.ws.soa.org/BookShelf/getBookResponse"
			 message="tns:getBookResponse"/>
		</operation>
		<operation name="isBookAvailable">
			<input wsam:Action="http://tp5.ws.soa.org/BookShelf/isBookAvailableRe
			quest" message="tns:isBookAvailable"/>
			<output wsam:Action="http://tp5.ws.soa.org/BookShelf/isBookAvailableR
			esponse" message="tns:isBookAvailableResponse"/>
		</operation>
	</portType>
	<binding name="BookShelfServicePortBinding" type="tns:BookShelf">
		<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
		<operation name="getBook">
			<soap:operation soapAction=""/>
			<input>
				<soap:body use="literal" namespace="http://tp5.ws.soa.org/"/>
			</input>
			<output>
				<soap:body use="literal" namespace="http://tp5.ws.soa.org/"/>
			</output>
		</operation>
		<operation name="isBookAvailable">
			<soap:operation soapAction=""/>
			<input>
				<soap:body use="literal" namespace="http://tp5.ws.soa.org/"/>
			</input>
			<output>
				<soap:body use="literal" namespace="http://tp5.ws.soa.org/"/>
			</output>
		</operation>
	</binding>
	<service name="BookShelfService">
		<port name="BookShelfServicePort" binding="tns:BookShelfServicePortBinding">
			<soap:address location="http://hedia:8080/BookShelfWebService/BookShe
			lfService"/>
		</port>
	</service>
</definitions>
```

```
<S:Body>
   <ns2:getBookResponse>
      <ReturnedBook>
         <bookId>bookD</bookId>
         <author>Spinoza</author>
         <bookName>TTP</bookName>
       </ReturnedBook>
    </ns2:getBookResponse>
</S:Body>
```