

Série TD / TP 8

Création et utilisation des JAX-WS SOAP Message Handlers.

I. Buts

- ✓ Traiter à la volée les messages SOAP échangés coté client et/ou coté service web grâce aux **handlers** (intercepteurs).
- ✓ Personnaliser les éléments **Header** et **Fault** d'un message SOAP.

Les services web ainsi que leurs clients peuvent avoir besoin d'accéder aux requêtes et aux réponses SOAP échangées afin de procéder à un traitement supplémentaire des informations contenues dans ces dernières. JAX-WS offre à cet égard un mécanisme permettant de les intercepter et de les personnaliser. Ce mécanisme est appelé Message Handler¹ (i.e. Intercepteur de Message).

Un exemple simple illustrant l'utilisation des handlers consiste à accéder aux informations dans la partie Header d'un message soap. En effet, des informations spécifiques au service Web ou au client (comme le login/password, la clé de cryptage, l'adresse IP, l'adresse MAC, etc.) peuvent être injectées et stockées dans l'en-tête soap. Les handlers sont par conséquent utilisés pour les extraire et les manipuler.

Les handlers de messages SOAP peuvent également être utilisés pour améliorer les performances d'un service Web. Par exemple, une fois que le service Web ait été déployé pendant un certain temps, vous pouvez découvrir que de nombreux clients consommateurs l'invoquent avec les mêmes paramètres. Vous pouvez améliorer les performances du service Web en mettant en cache les résultats des appels récurrents du service Web (en supposant que les résultats sont statiques) et renvoie immédiatement ces résultats lorsque cela est approprié, sans jamais invoquer les composants qui implémentent le service Web. Vous implémentez cette amélioration de performance en utilisant des handlers, qui vérifient le message de requête soap et voient si elle contient les paramètres en question.

Il existe deux types de handlers de messages SOAP dans JAX WS :

- ✓ SOAP Protocol Handlers : Ce handler est utilisé pour accéder à l'ensemble du message SOAP, y compris les en-têtes.
- ✓ Logical Handlers : En utilisant ce type de handlers, uniquement le contenu (le payload) du message qui est accessible.

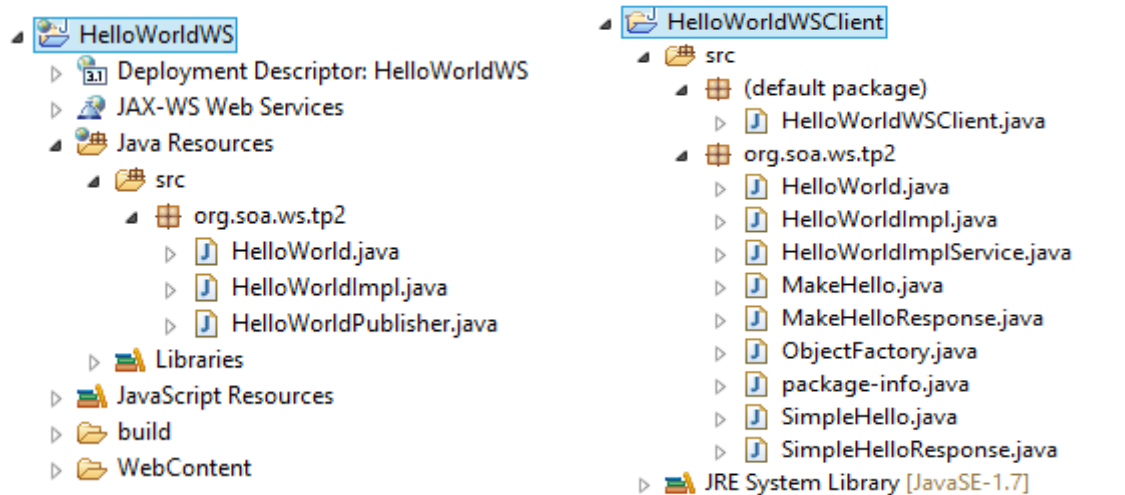
Les deux handlers étendent l'interface `javax.xml.ws.handler.Handler`. Les méthodes disponibles dans l'interface Handler sont: *handleMessage*, *handleFault*, et *Close*.

Un client ou un service web peuvent implémenter plusieurs handlers dans un même projet. Ces handlers peuvent être regroupés pour former une chaîne de handlers (handler chain).

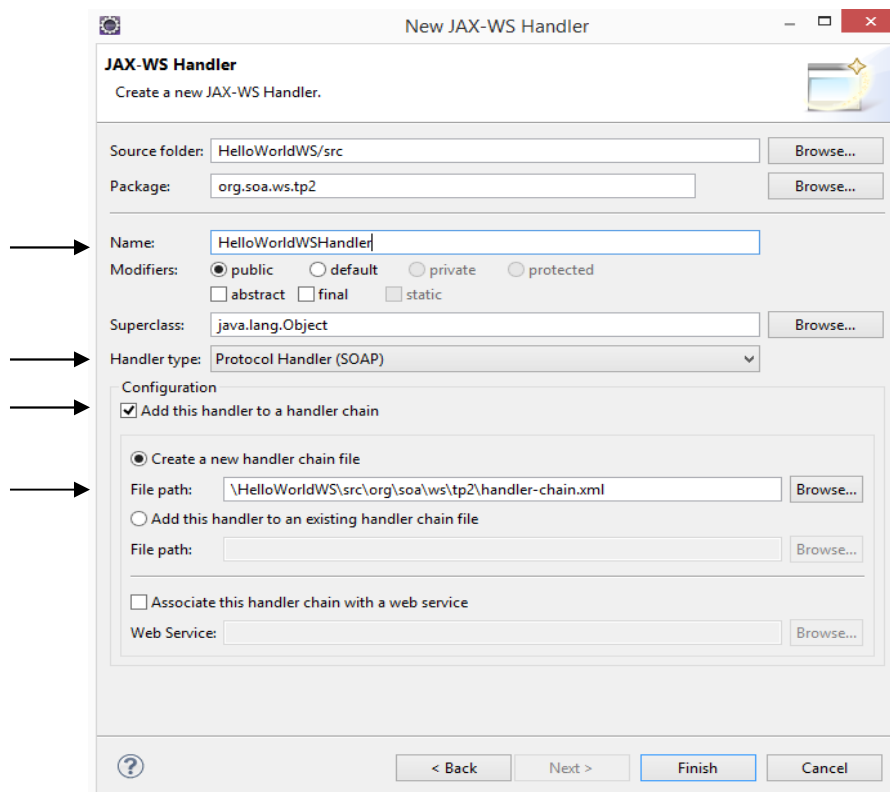
¹ https://docs.oracle.com/cd/E24329_01/web.1211/e24965/handlers.htm#WSADV161

II. Etapes générales d'implémentation d'un SOAP Handler - coté Web Service

Pour illustrer les étapes générales à suivre afin de créer et d'implémenter un SOAP Handler de messages SOAP, le service web HelloWorldWS avec les opérations *simpleHello* et *makeHello*, précédemment développé, est utilisé comme exemple.



1. Cliquer droit sur le package `org.soa.ws.tp2` puis choisir **New** → **Other** → **Rechercher JAX-WS Handler** dans *Wizards* et sélectionner le depuis la liste → *Next*.
2. Fournir et compléter les informations suivantes comme suit :
 - ✓ Name : représente le nom du handler. Ex : *HelloWorldWSHandler*.
 - ✓ Handler type: Choisir *Protocol Handler (SOAP)*.
 - ✓ Configuration: cocher *Add this handler to a handler chain*.
 - ✓ File path : Sélectionner le nom du projet pour créer **un nouveau fichier xml** de configuration de chaîne de handlers. Pour cet exemple, entrer : `\HelloWorldWS\src\org\soa\ws\tp2\handler-chain.xml`



3. Après avoir fourni tous les détails ci-dessus, cliquer sur le bouton *Finish*, une nouvelle classe de handler est créée et affichée. Cette classe implémente l'interface *SOAPHandler*.
 - En plus de la classe handler, le fichier xml *handler-chain* définissant le nom et la classe du handler est lui aussi généré et ajouté au package.

```
<?xml version="1.0" encoding="UTF-8"?>
<handler-chains xmlns="http://java.sun.com/xml/ns/javaee">
  <handler-chain>
    <handler>
      <handler-name>HelloWorldWSHandler</handler-name>
      <handler-class>org.soa.ws.tp2.HelloWorldWSHandler</handler-class>
    </handler>
  </handler-chain>
</handler-chains>
```

- L'implémentation par défaut de la classe Handler a les méthodes suivantes comme indiqué dans la figure ici-bas :
 - ✓ *handleMessage*: toutes les informations supplémentaires ajoutées au message soap, entrant ou sortant, sont traitées dans cette méthode. Cette méthode prend *SOAPMessageContext* comme paramètre.
 - ✓ *handleFault*: est appelée si des erreurs sont produites dans le traitement du message.
 - ✓ *getHeaders*: récupère les en-têtes avec un QName particulier à partir du message dans le contexte de message.
 - ✓ *close*: cette méthode termine le handler et libère les ressources utilisées.

```
import java.util.Set;

public class HelloWorldWSHandler implements SOAPHandler<SOAPMessageContext> {

    @Override
    public boolean handleMessage(SOAPMessageContext context) {
        // TODO Auto-generated method stub
        return false;
    }

    @Override
    public boolean handleFault(SOAPMessageContext context) {
        // TODO Auto-generated method stub
        return false;
    }

    @Override
    public void close(MessageContext context) {
        // TODO Auto-generated method stub
    }

    @Override
    public Set<QName> getHeaders() {
        // TODO Auto-generated method stub
        return null;
    }
}
```

4. Associer le handler au service web dans le code de la classe d'implémentation en utilisant l'annotation **@HandlerChain** comme suit :

```

import javax.ws.HandlerChain;

@WebService(endpointInterface="org.soa.ws.tp2.HelloWorld")
@HandlerChain(file="handler-chain.xml")
public class HelloWorldImpl implements HelloWorld{

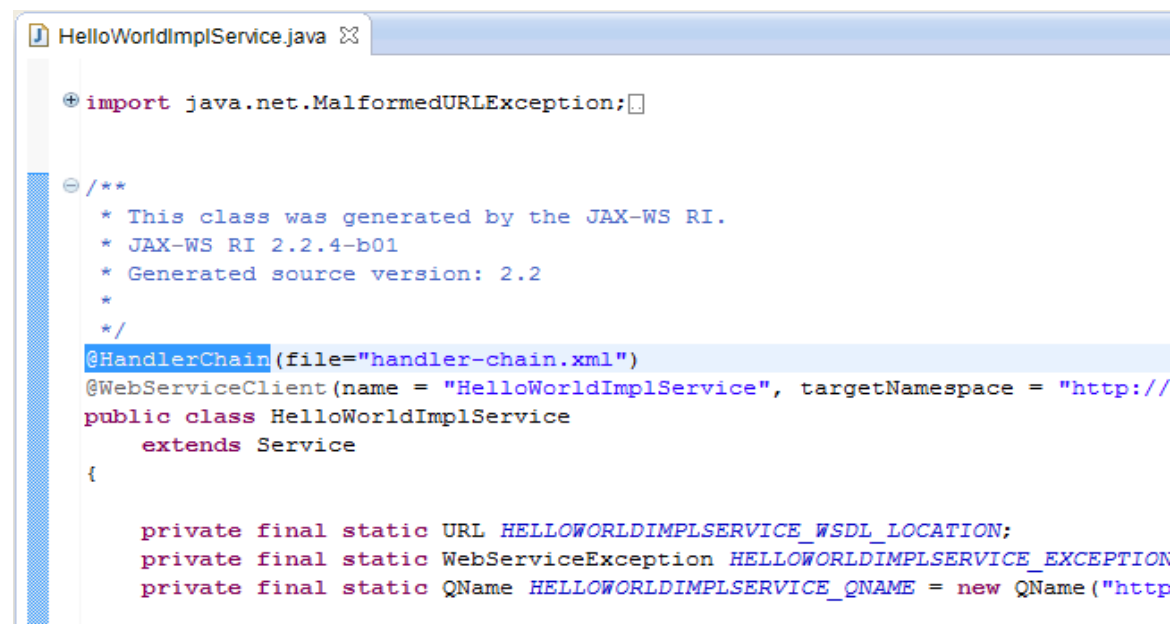
    @Override
    @WebMethod
    public String simpleHello() {
        return "Hello World!";
    }
}

```

5. Modifier la méthode handleMessage selon le besoin et Republier le service web.

III. Etapes générales d'implémentation d'un SOAP Handler - coté Client

Il est tout à fait possible d'implémenter des handlers coté client. Cela se fait en suivant les mêmes étapes ci-dessus, à une différence près. En effet, les étapes 1, 2, 3, et 5 restent les mêmes ; Seule l'étape 4 change, où l'annotation permettant d'associer le(s) handler(s) créé(s) au client est ajoutée dans le code la classe d'implémentation <ServiceName> du service web générée via wsimport comme suit :



```

HelloWorldImplService.java
import java.net.MalformedURLException;

/**
 * This class was generated by the JAX-WS RI.
 * JAX-WS RI 2.2.4-b01
 * Generated source version: 2.2
 */
@HandlerChain(file="handler-chain.xml")
@WebServiceClient(name = "HelloWorldImplService", targetNamespace = "http://")
public class HelloWorldImplService
    extends Service
{

    private final static URL HELLOWORLDDIMPLSERVICE_WSDL_LOCATION;
    private final static WebServiceException HELLOWORLDDIMPLSERVICE_EXCEPTION
    private final static QName HELLOWORLDDIMPLSERVICE_QNAME = new QName("http

```

IV. SOAP Fault Handling

Si une erreur se produit pendant le traitement, la réponse à un message-requête SOAP est un élément SOAP Fault dans le corps (body) du message-réponse et l'erreur est renvoyée à l'expéditeur de la requête.

Le mécanisme SOAP Fault renvoie des informations spécifiques sur l'erreur, y compris un code prédéfini, une description et l'adresse du processeur SOAP qui a généré l'erreur. Ces informations représentent le Fault Block du message SOAP.

Le code suivant est un exemple d'un message SOAP Fault :

```

<?xml version='1.0' encoding='UTF-8'?>

<SOAP-ENV:Envelope xmlns:SOAP-
    ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema" >

    <SOAP-ENV:Body>

        <SOAP-ENV:Fault>
            <faultcode xsi:type="xsd:string">
                SOAP-ENV:Client
            </faultcode>

            <faultstring xsi:type="xsd:string">
                - Description de l'erreur -
            </faultstring>
        </SOAP-ENV:Fault>

    </SOAP-ENV:Body>

</SOAP-ENV:Envelope>

```

L'élément SOAP Fault dans le body du message peut comporter les sous-éléments suivants :

Sous-élément	Description
<faultcode>	<p>Il s'agit d'un code texte utilisé pour indiquer une classe d'erreurs.</p> <p><i>SOAP-ENV: VersionMismatch:</i> un espace de noms non valide dans l'enveloppe a été trouvé.</p> <p><i>SOAP-ENV: Client:</i> le message a été mal formé ou contenait des informations incorrectes.</p> <p><i>SOAP-ENV: Server:</i> Il y a eu un problème avec le serveur, donc le message n'a pas pu être traité.</p> <p>Etc.</p>
<faultstring>	C'est un message texte expliquant l'erreur.
<faultactor>	Il s'agit d'un texte string indiquant qui a causé l'erreur. Il est utile si le message SOAP parcourt plusieurs nœuds et que le client doit savoir quel nœud a causé l'erreur.
<detail>	Il s'agit d'un élément utilisé pour transmettre des messages d'erreur spécifiques à l'application.

V. Exercice

Compléter le service web Hello World et son client consommateur pour que ce dernier ne reçoit une réponse à sa requête que si son adresse IP est autorisée. Le cas contraire, le client se verra refuser l'accès au service web invoqué.

Pour ce faire, attacher un SOAP handler coté client afin d'injecter l'adresse IP du client dans le block Header de chaque requête SOAP sortante. Puis, attacher un deuxième SOAP handler, coté service web cette fois, permettant d'extraire l'adresse IP du client depuis n'importe quel requête entrante et de vérifier sa validité.

A noter :

- Un SOAP handler est appelé automatiquement à chaque envoi **et** à chaque réception d'un message soap (i.e. une fois en cas de requête et une deuxième fois en cas de réponse). La méthode *getHeaders* s'exécute en premier, puis, elle sera suivie par *handleMessage*. En cas d'erreur dans le traitement de la requête/réponse, *handleFault* est invoquée.
- Quand le traitement (méthode *handleMessage*) se termine avec succès, la méthode *close* est appelée. Ne pas oublier de retourner *true* à la fin de *handleMessage*.
- Il est possible de vérifier si le message SOAP traité par un handler est un message entrant ou un message sortant en utilisant la méthode *get* de *SOAPMessageContext* et la propriété `MESSAGE_OUTBOUND_PROPERTY` comme suit :

```
Boolean isRequest = (Boolean) context.get(MessageContext.MESSAGE_OUTBOUND_PROPERTY);
```

- La handler coté web service doit gérer les erreurs (faults) et renvoyer une description claire au client dans le fault block du body soap.
 - ✓ Cas partie SOAP Header inexistante ;
 - ✓ Cas partie SOAP Header existante mais vide ;
 - ✓ Cas adresse IP introuvable dans le SOAP Header ;
 - ✓ Cas adresse IP non autorisée – accès refusé.
- Utiliser le TCP/IP Monitor pour tracker et vérifier les messages SOAP échangés.

Exemple de messages soap échangés – Cas adresse IP non autorisée :

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Header>
    <ipAddress xmlns="http://tp2.ws.soa.org/">127.1.1.0</ipAddress>
  </S:Header>
  <S:Body>
    <ns2:makeHello xmlns:ns2="http://tp8.ws.soa.org/">
      <arg0 xmlns="">JAX-WS</arg0>
    </ns2:makeHello>
  </S:Body>
</S:Envelope>
```

Requête Client

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Header/>
  <S:Body>
    <S:Fault xmlns:ns4="http://www.w3.org/2003/05/soap-envelope">
      <faultcode>S:Server</faultcode>
      <faultstring>
        Invalid IP Address - Access is denied!
      </faultstring>
    </S:Fault>
  </S:Body>
</S:Envelope>
```

Réponse SW