

TD – Java Web Service - SOAP

Omar ABOU KHALED, Stefano CARRINO, Julien TSCHERRIG

1 Buts du travail

- Familiarisation avec Java EE
- Utilisation de NetBeans
- Mise en place d'un Web Service SOAP
- Consommation d'un Web Service SOAP

2 Composants nécessaires

Les composants suivants sont nécessaires à ce travail pratique :

Composants	Remarques
Java JDK 7	http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
NetBeans 8.0.1 - JavaEE	https://netbeans.org/downloads/

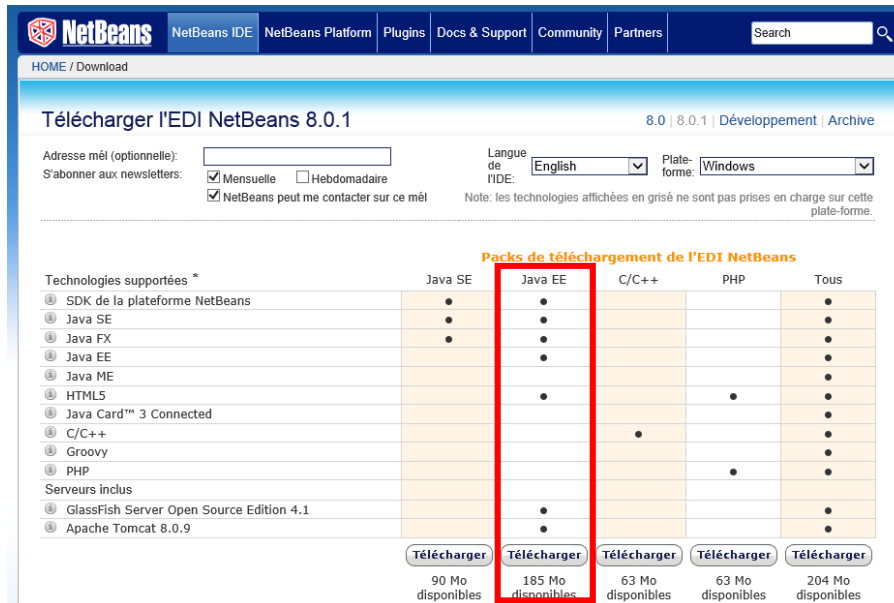
3 Travail à réaliser

1. Installation
2. Création d'un projet web
3. Création d'un Web Service SOAP
4. Utilisation du Web Service réalisé
5. Utilisation d'un Web Service tiers
6. Pour aller plus loin : Utilisation de SOAP avec JSF

4 Téléchargement et installation de NetBeans

Pour réaliser ce TP vous allez utiliser NetBeans pour le développement. NetBeans est un concurrent d'Eclipse appartenant à Oracle.

Si NetBeans n'est pas installé sur votre machine, télécharger et installer la version Java EE sur le site de NetBeans (<https://netbeans.org/downloads/>).



Télécharger l'EDI NetBeans 8.0.1

8.0 | 8.0.1 | Développement | Archive

Adresse mël (optionnelle):

S'abonner aux newsletters: ☒ Mensuelle ☐ Hebdomadaire

☒ NetBeans peut me contacter sur ce mël

Langue de l'IDE: Plate-forme:

Note: les technologies affichées en grisé ne sont pas prises en charge sur cette plate-forme.

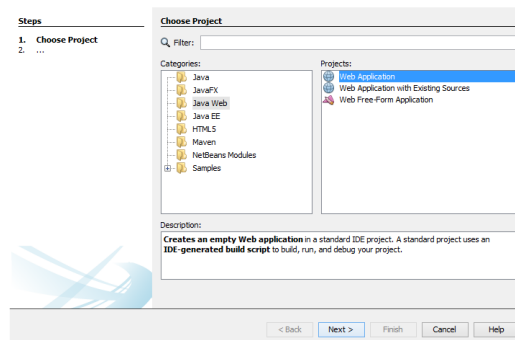
Packs de téléchargement de l'EDI NetBeans

Technologies supportées *	Java SE	Java EE	C/C++	PHP	Tous
SDK de la plateforme NetBeans	•	•			•
Java SE	•	•			•
Java FX	•	•			•
Java EE		•			•
Java ME		•			•
HTML5				•	•
Java Card™ 3 Connected				•	•
C/C++			•		•
Groovy					•
PHP				•	•
Serveurs inclus					
GlassFish Server Open Source Edition 4.1		•			•
Apache Tomcat 8.0.9		•			•
	Télécharger	Télécharger	Télécharger	Télécharger	Télécharger
	90 Mo disponibles	185 Mo disponibles	63 Mo disponibles	63 Mo disponibles	204 Mo disponibles

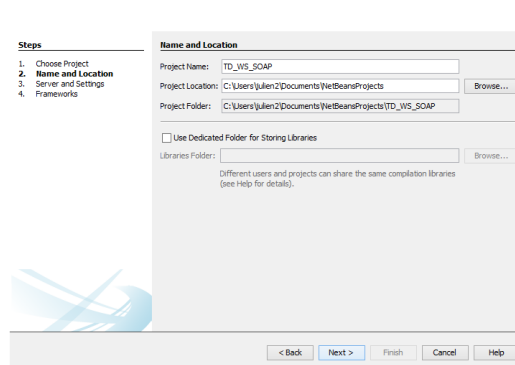
Effectuer l'installation par défaut, en faisant attention d'installer le serveur d'application GlassFish (normalement coché lors de l'installation).

5 Création d'une application web

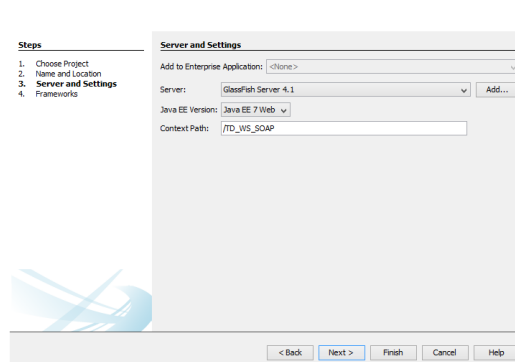
File -> New Project



Sélectionné « Java Web » puis « Web Application » et cliquer sur « Next »



Indiquer le nom du projet « TD_WS_SOAP » ainsi que l'emplacement désiré pour le projet puis cliquer sur « Next ».



Valider ensuite la création du projet à l'aide du bouton « Finish ». Le projet de type Web Application est maintenant terminé.

Remarque : La structure d'un projet avec NetBeans est semblable à celle vue avec Eclipse.

6 Création d'un web service SOAP (simple)

Maintenant que nous avons créé une application de type Web nous allons pouvoir commencer la création de notre service web (WS – web service).

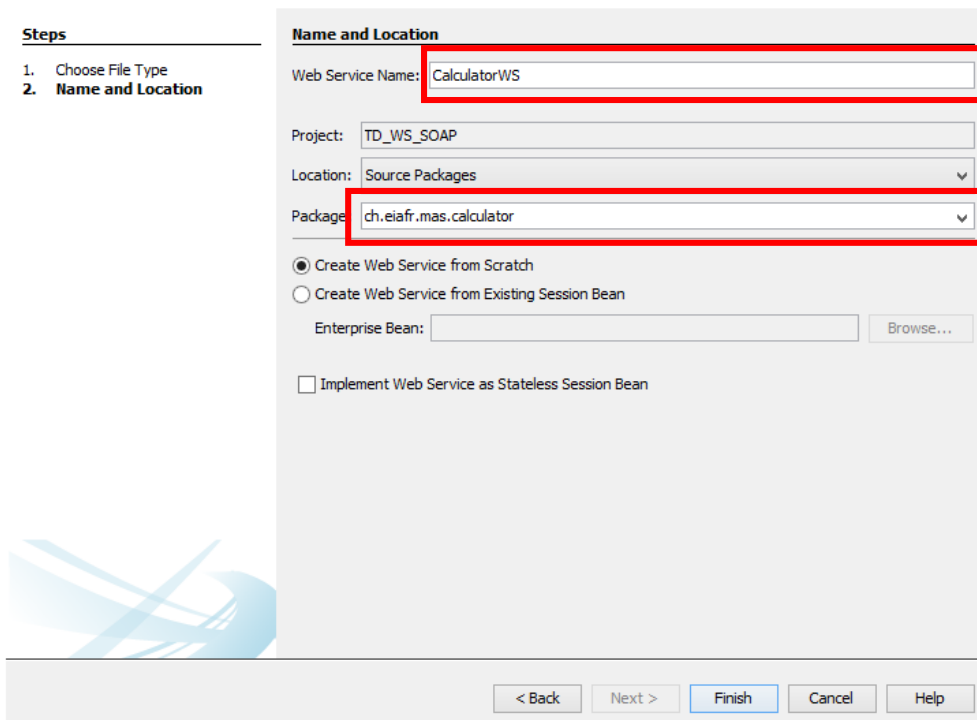
- L'objectif de cette première partie est de réaliser un WS de base
- Il s'agira d'une méthode d'addition de 2 nombres entiers
- Réaliser d'autres opérations mathématiques

6.1 Création de la classe du WS

Faire un clic droit sur le projet (TD_WS_SOAP) et choisir « News » puis « Web Service ».

Attention : Bien choisir « Web Service » et non pas « Web Service Client »

Indiquer un nom pour le WS ainsi que le chemin du package. Le nom du WS indiqué sera également le nom de la classe Java qui sera générée. Cliquer ensuite sur « Finish ».



The screenshot shows the 'Name and Location' dialog box in the Eclipse IDE. On the left, a 'Steps' pane lists: 1. Choose File Type, 2. Name and Location. The main dialog has the following fields and options:

- Web Service Name:** CalculatorWS (highlighted with a red box)
- Project:** TD_WS_SOAP
- Location:** Source Packages
- Package:** ch.eiafr.mas.calculator (highlighted with a red box)
- Options:**
 - ☒ Create Web Service from Scratch
 - ☐ Create Web Service from Existing Session Bean
 - Enterprise Bean: (empty field) [Browse...]
 - ☐ Implement Web Service as Stateless Session Bean

At the bottom, there are buttons: < Back, Next >, Finish (highlighted in blue), Cancel, and Help.

Le code Java généré pour le WS est le suivant

```
package ch.eiafr.mas.calculator;

import javax.ws.WebService;
import javax.ws.WebMethod;
import javax.ws.WebParam;

@WebService(serviceName = "CalculatorWS")
public class CalculatorWS {

    /**
     * This is a sample web service operation
     */
    @WebMethod(operationName = "hello")
    public String hello(@WebParam(name = "name") String txt) {
        return "Hello " + txt + " !";
    }
}
```

Il s'agit ici d'une méthode (hello) appartenant un à WS qui prend en entrée un String « txt » et qui en retour affiche « Hello [txt] ».

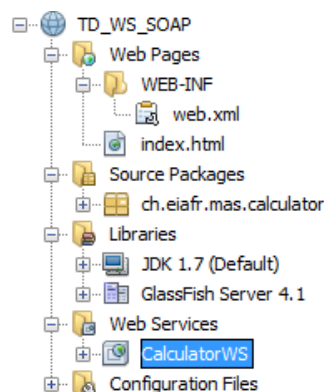
6.2 Déploiement du WS

Avant de pouvoir exécuter le web service, il est nécessaire de le déployer sur le serveur.

Faire un clic droit sur le projet (TD_WS_SOAP) et choisir « Deploy ». Le WS est maintenant chargé sur GlassFish. Il est possible de le tester.

6.3 Accéder au WS

Il est possible de tester son fonctionnement.



Dans le dossier « Web Services », faite un clic droit sur le web service que vous venez de créer (CalculatorWS) puis choisissez « Test Web Service ». Une fenêtre s'ouvre et présente toute les points d'entrée accessibles pour ce WS. Actuellement seul le point d'accès « hello » est disponible.

6.3.1 Tester le WS

Test le bon fonctionnement de votre WS en saisissant une valeur, puis en cliquant sur le bouton « hello ». Sur l'écran ci-dessous, il est également possible de consulter le WSDL. Il s'agit de la présentation du contrat du WS qui indique sous forme XML les possibilités proposées par le WS.

CalculatorWS Web Service Tester

This form will allow you to test your web service implementation ([WSDL File](#))

To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

Methods :

public abstract java.lang.String ch.eiafr.mas.calculator.CalculatorWS.hello(java.lang.String)

hello

Remarque : Il est indiqué que le paramètre du point d'accès « hello » est un String.

6.3.2 SOAP Request

Les informations indiquées sous SOAP Request correspondent à ce qui est transmis du Client au WS.

Remarque : Dans notre cas, la valeur transmise est « Julien »

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:hello xmlns:ns2="http://calculator.mas.eiafr.ch/">
      <name>Julien</name>
    </ns2:hello>
  </S:Body>
</S:Envelope>
```

6.3.3 SOAP Response

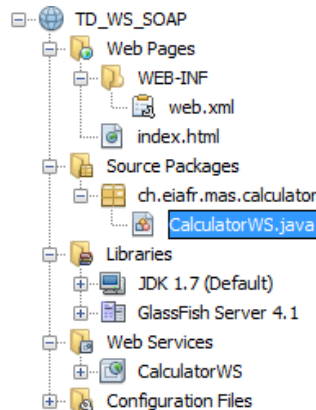
Les informations indiquées sous SOAP Response correspondent à ce qui est du WS au Client.

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:helloResponse xmlns:ns2="http://calculator.mas.eiafr.ch/">
      <return>Hello Julien !</return>
    </ns2:helloResponse>
  </S:Body>
</S:Envelope>
```

Remarque : Dans notre cas, la valeur reçue est « Hello Julien ! »

7 Création d'une nouvelle action

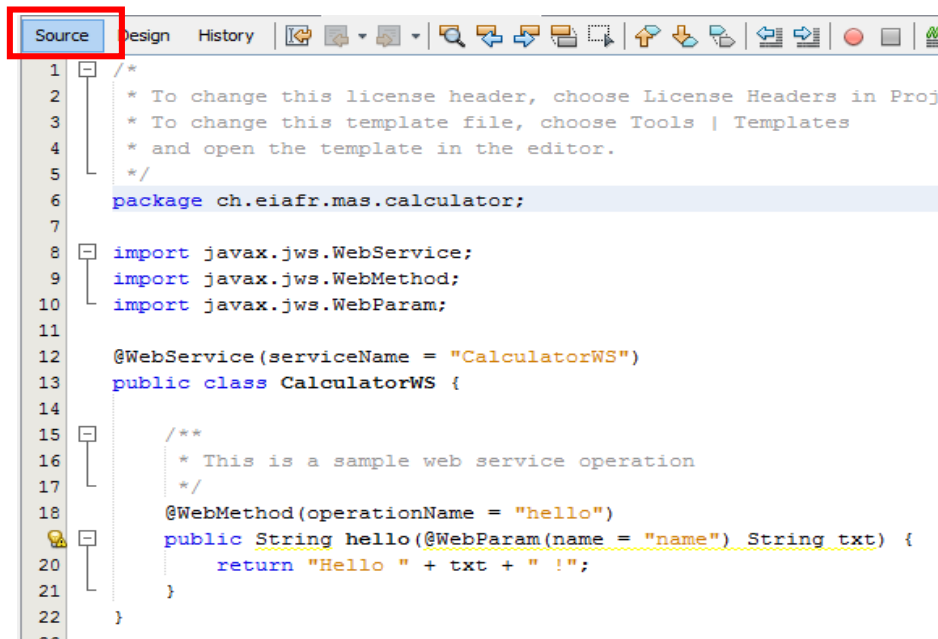
Nous avons pu tester et comprendre l'exemple de base. Pour rajouter d'autres opérations accessibles via le WS, NetBeans offre la possibilité de les créer graphiquement.



Pour cela double cliquer sur CalculatoWS.java se trouvant dans le dossier « Sources Packages ». Deux vue différentes sont alors possible. Elles présentent les mêmes informations. La vue « Source » permet de visualiser et de modifier le contenu d'une méthode.

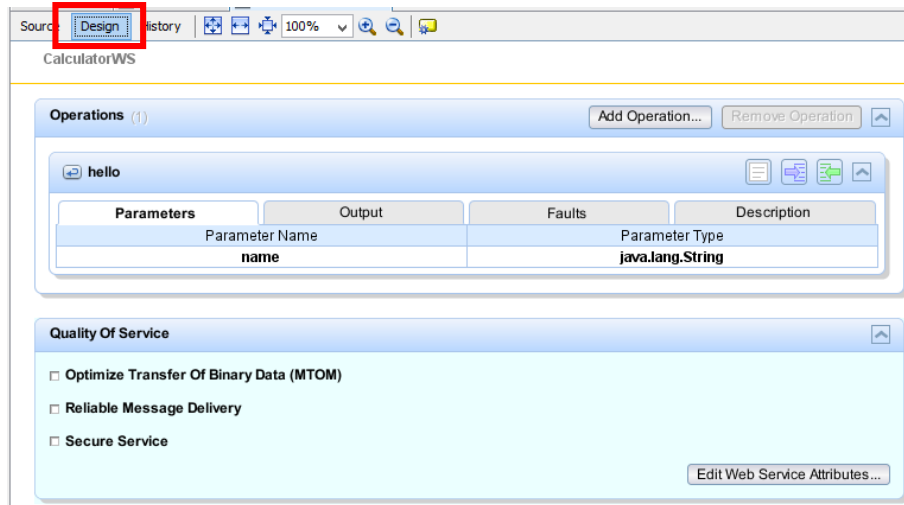
7.1 Vue Code

En sélectionnant la vue « Source », nous pouvons voir le code Java de la classe « CalculatorWS.java ». Elle est composée d'une seule méthode (hello), prenant un paramètre String.



7.2 Vue Design

La vue « Design » reprend les mêmes informations que la vue « Code »

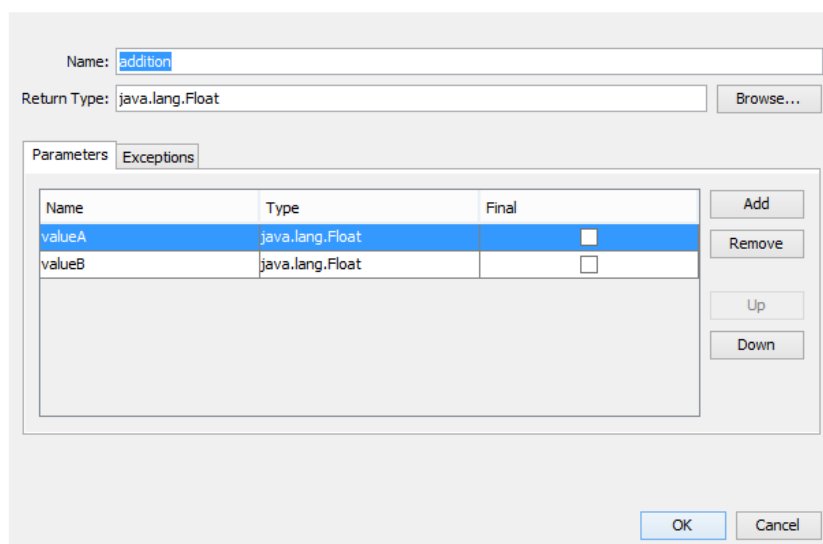


Depuis la vue « Design », cliquer sur le bouton « Add Operation... »

Comme indiqué sur l'image ci-dessous, nommez votre méthode (addition dans notre cas). Indiquer également le type de retour (java.lang.Float dans notre cas).

Notre méthode d'addition prend 2 paramètres, valueA et valueB de type java.lang.Float.

Puis cliquer sur OK



La vue « Design » possède maintenant une nouvelle opération nommée addition

addition			
Parameters	Output	Faults	Description
Parameter Name		Parameter Type	
valueA		java.lang.Float	
valueB		java.lang.Float	

La vue « Source » possède également une opération nommée addition

```

6  package ch.eiafr.mas.calculator;
7
8  import javax.xml.ws.WebService;
9  import javax.xml.ws.WebMethod;
10 import javax.xml.ws.WebParam;
11
12 @WebService(serviceName = "CalculatorWS")
13 public class CalculatorWS {
14
15     /**
16      * This is a sample web service operation
17      */
18     @WebMethod(operationName = "hello")
19     public String hello(@WebParam(name = "name") String txt) {
20         return "Hello " + txt + " !";
21     }
22
23     /**
24      * Web service operation
25      */
26     @WebMethod(operationName = "addition")
27     public Float addition(@WebParam(name = "valueA") Float valueA, @WebParam(name = "valueB") Float valueB) {
28         //TODO write your implementation code here:
29         return null;
30     }
31 }

```

Pour le moment la fonction implémentée retourne « null » (par défaut). Il est nécessaire de procéder à l'implémentation de la fonction d'addition.

Saisissez alors le code suivant :

```

/**
 * Web service operation
 */
@WebMethod(operationName = "addition")
public Float addition(
    @WebParam(name = "valueA") Float valueA,
    @WebParam(name = "valueB") Float valueB) {

    Float total = valueA + valueB;
    return total;
}

```

7.3 Test du web service

Vérifier le bon fonctionnement de votre code en déployant puis en testant le web service (voir 6.2 Déploiement du WS et 6.3 Accéder au WS).

Vous devriez obtenir une page ressemblant à celle-ci-dessous. On peut voir que la méthode « addition » que vous venez de créer existe. Elle nécessite deux « java.lang.Float » en paramètre. Vous pouvez ensuite tester son bon fonctionnement en insérant deux valeurs numériques puis cliquer sur « addition »

CalculatorWS Web Service Tester

This form will allow you to test your web service implementation ([WSDL File](#))

To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

Methods :

public abstract java.lang.String ch.eiafr.mas.calculator.CalculatorWS.hello(java.lang.String)

hello ()

public abstract java.lang.Float ch.eiafr.mas.calculator.CalculatorWS.addition(java.lang.Float,java.lang.Float)

addition (2 3)

7.4 Suite

Pour continuer, en vous basant sur ce que vous avez précédemment réalisé, implémenter les fonctionnalités suivantes :

- Soustraction
- Multiplication
- Division

Tester ensuite leur fonctionnement.

8 Exercice Blog (Back-end)

On vous demande d'implémenter la partie back-end d'un blog. Il s'agit de fournir via un WS SOAP, les informations utiles au bon fonctionnement d'un blog.

8.1 Données

On vous donne les informations suivantes :

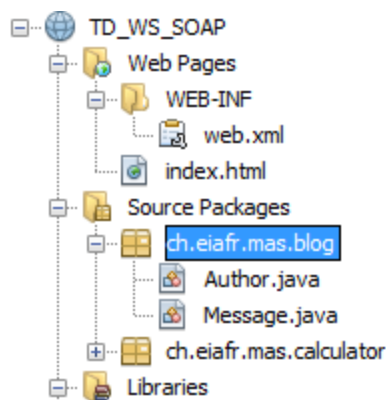
- Les auteurs du blog sont représentés par un nom, un prénom et un pseudo. Les auteurs contiennent une liste de messages
- Les billets publiés sur le blog sont représentés par un auteur, un titre, un message et une date (`java.util.Date`).

Les données seront initialisées de manière statique (au lancement du WS).

8.2 Création des modèles

Créer les deux classes (par exemple, auteur/Author et billet/Message) dans le package `ch.eiafr.mas.blog` comme illustré ci-dessous.

Implémenter les deux classes que vous venez de créer avec les indications données au point 8.1.



8.3 Création d'un nouveau WS

En vous basant sur le point 6.1 créer un WS se nommant « BlogWS » dans le package « `ch.eiafr.mas.blog` »

8.3.1 Ajout de l'accès aux données statiques

Les données statiques vont simuler ici l'interaction avec une base de données. Nous allons travailler ici directement avec un modèle statique.

Pour cela, il est nécessaire de déclarer les deux listes (auteurs et messages) de la manière suivante :

```
@WebService(serviceName = "BlogWS")
public class BlogWS {

    private static ArrayList<Author> AUTHORS = new ArrayList<>();
    private static ArrayList<Message> MESSAGES = new ArrayList<>();
}
```

8.3.2 Implémentation des méthodes

En utilisant les deux listes (AUTHORS et MESSAGES), rajouter l'implémentation pour les signatures (en-tête) de méthodes suivantes selon votre logique.

- Permet de lister les auteurs

```
@WebMethod(operationName = "listAuthors")
public ArrayList<Author> listAuthors()
```

- Permet d'ajouter un auteur

```
@WebMethod(operationName = "addAuthor")
public boolean addAuthor(
    @WebParam(name = "pseudo") String pseudo,
    @WebParam(name = "firstName") String firstName,
    @WebParam(name = "lastName") String lastName)
```

- Permet de supprimer un auteur

```
@WebMethod(operationName = "removeAuthor")
public boolean removeAuthor(@WebParam(name = "author") String pseudo)
```

- Permet de lister les messages

```
@WebMethod(operationName = "listMessages")
public ArrayList<Message> listMessages() {
    return MESSAGES;
}
```

- Permet d'ajouter un message

```
@WebMethod(operationName = "addMessage")
public boolean addMessage(
    @WebParam(name = "author") String pseudo,
    @WebParam(name = "title") String title,
    @WebParam(name = "content") String content)
```

- Permet de supprimer un message

```
@WebMethod(operationName = "removeMessage")
public boolean removeMessage(
    @WebParam(name = "message") String title)
```

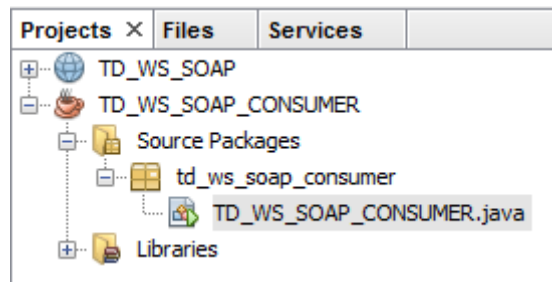
9 Exercice Blog (Consommation des données du WS)

Ce dernier point va vous permettre de tester le bon fonctionnement de votre WS précédemment réalisé. Il va également permettre de comprendre récupérer les informations mises à disposition via un WS implémentant SOAP.

9.1 Réutilisation de notre WS

Tout en conservant le projet TD_WS_SOAP déployé, créer un nouveau projet du type « Java Application » (voir 5 Création d'une application web) nommée « TD_WS_SOAP_CONSUMER ».

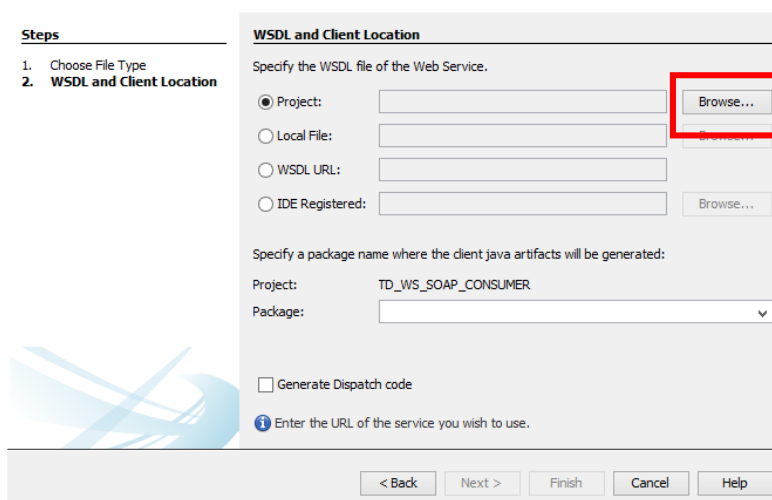
Vous devriez ensuite avoir la structure suivante dans votre liste de projets NetBeans



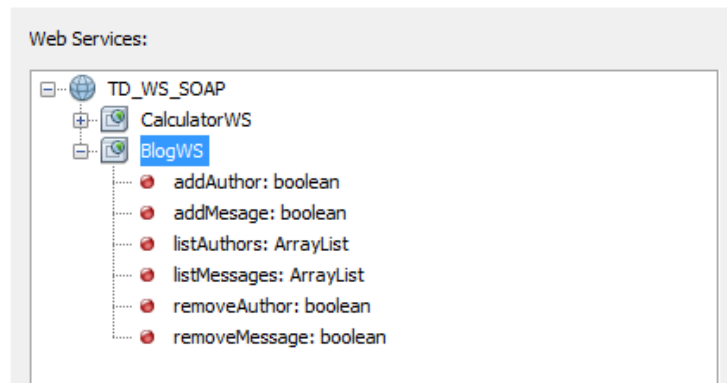
Nous allons maintenant nous connecter à notre WS SOAP crée au point 8.

Faites un clic droit sur le projet « TD_WS_SOAP_CONSUMER » → New → Web Service Client.

Cliquer sur « Browse... »

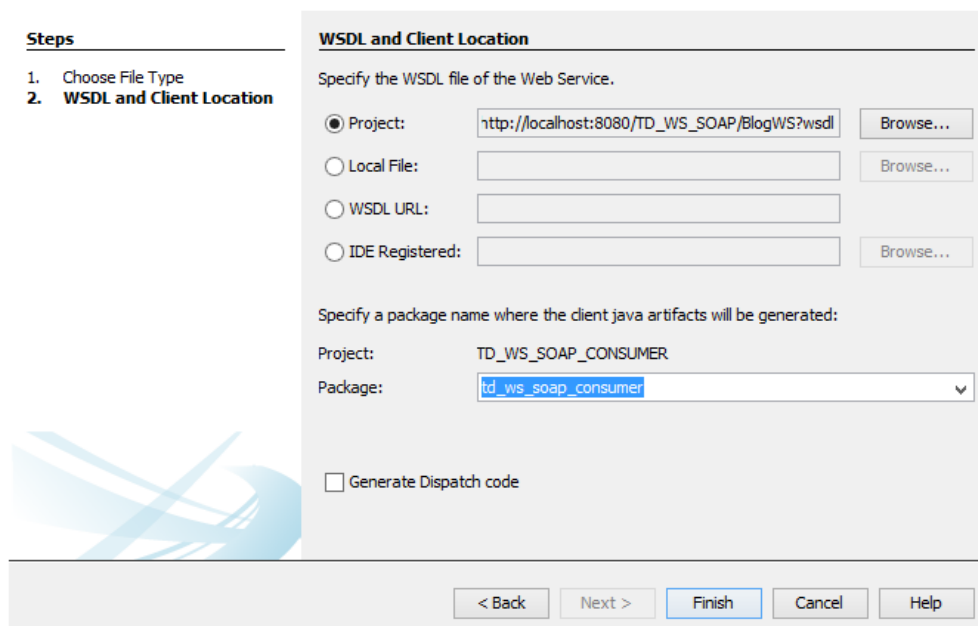


Et choisissez le WS Blog réalisé (BlogWS) et cliquez sur « OK »

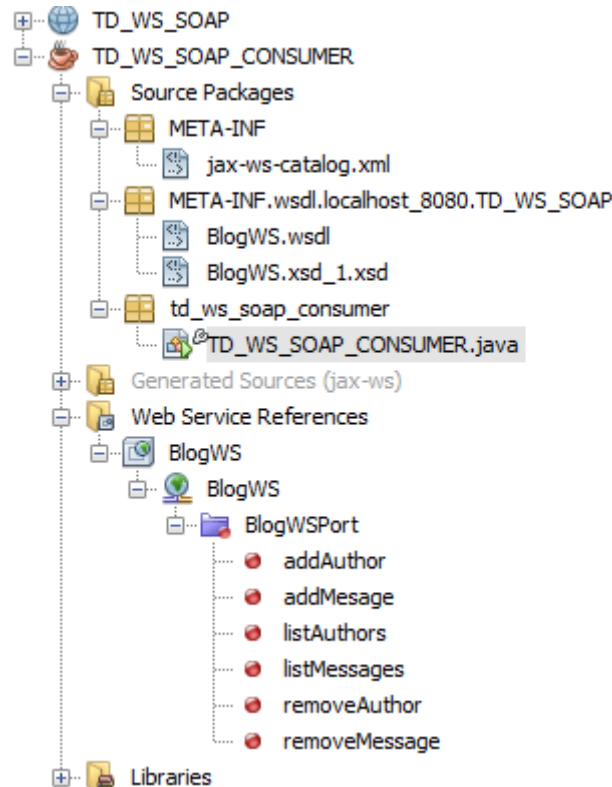


On peut voir que le WSDL (http://localhost:8080/TD_WS_SOAP/BlogWS?wsdl) est maintenant affiché.

Sélectionner ensuite le package par défaut dans le menu déroulant



La structure du projet se présente maintenant de cette façon. Intéressons-nous de plus près aux informations se trouvant sous le répertoire « Web Service References ».



On peut voir que notre WS SOAP BlogWS est maintenant présent, ainsi que la liste des méthodes que nous avons créées précédemment.

Pour consommer le Webservice, il suffit de faire un drag&drop des méthodes dans un code Java. Par exemple, rendez-vous sur le fichier TD_WS_SOAP_CONSUMER.java et effectuez le drag&drop des méthodes addAuthor et listAuthors.

Deux méthodes sont alors générées :

```
private static boolean addAuthor(java.lang.String pseudo, java.lang.String firstName, java.lang.String lastName) {
    td_ws_soap_consumer.BlogWS_Service service = new td_ws_soap_consumer.BlogWS_Service();
    td_ws_soap_consumer.BlogWS port = service.getBlogWSPort();
    return port.addAuthor(pseudo, firstName, lastName);
}

private static java.util.List<td_ws_soap_consumer.Author> listAuthors() {
    td_ws_soap_consumer.BlogWS_Service service = new td_ws_soap_consumer.BlogWS_Service();
    td_ws_soap_consumer.BlogWS port = service.getBlogWSPort();
    return port.listAuthors();
}
```

Il est maintenant possible d'utiliser ces méthodes directement dans le main :

```
public static void main(String[] args) {  
    addAuthor("tsc", "julien", "tscherrig");  
    for (Author author : listAuthors())  
        System.out.println(author.getPseudo());  
}
```

Effectuer les différentes méthodes de votre WS en reproduisant ce qui est proposé ci-dessus.

9.2 Pour finir

Essayer de vous connecter via ce WSDL

- <http://www.webservicex.com/globalweather.asmx?WSDL>

9.3 Pour aller plus loin

Essayer d'intégrer la consommation du WS SOAP avec JSF

- <http://www.javabeat.net/accessing-web-services-from-jsf-applications/>