

SOAP - Travaux pratiques

Antoine Jedrezak

L'énoncé est à adapter que vous soyez sur Linux, Windows ou Mac.
Le TP contient des questions auxquelles il faut répondre, elles sont en lien avec les manipulations.

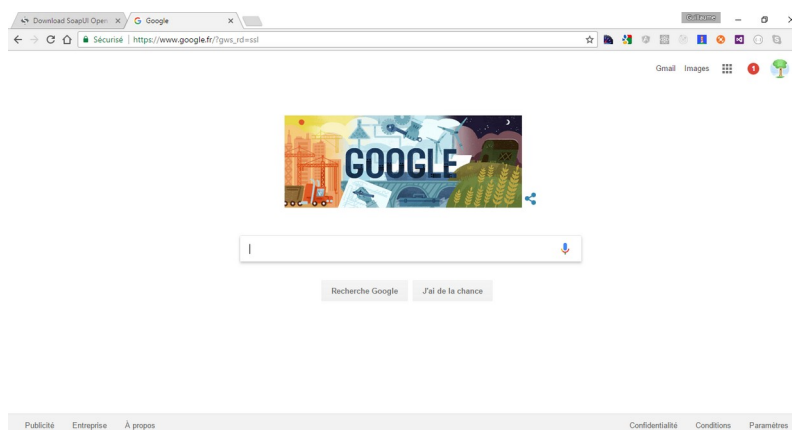
Le but de ce TP est de :

- Créer et modifier un serveur SOAP avec node.js
- Consommer un service SOAP à l'aide du logiciel SOAP UI.

1. Installation

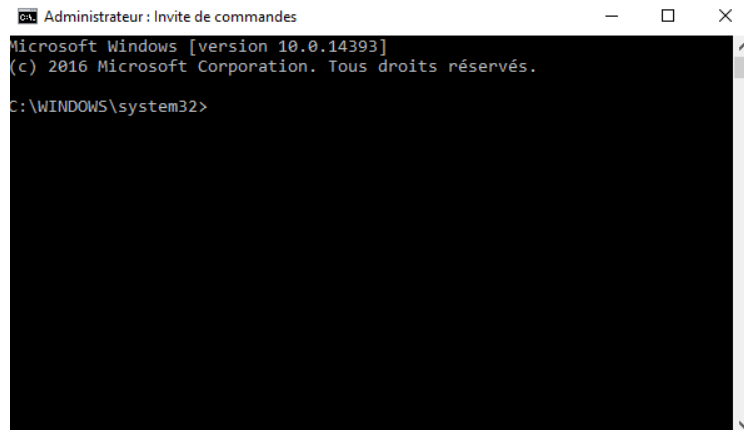
- Node.js dernière version LTS (Long Time Support)
- Client Git dernière version
- Visual Studio code dernière version
- SOAP UI version Open Source dernière version (gratuite), nécessite peut-être au préalable Java

Vous pouvez utiliser un moteur de recherche :



2. Récupération du projet sous git

1. Ouvrir un client de type « cmd » pour exécuter des lignes de commande



2. Organisez-vous dans un répertoire du type « C:/TP/WebServices/soap », pour mettre le code de votre TP
3. Dans votre « cmd » taper la commande :
 - a. `git clone --depth 1 --branch soap https://github.com/guillaume-chervet/course.rest.git`
 - b. Ceci récupérera un squelette de projet node.js
4. Placez-vous dans le nouveau répertoire « C:/TP/WebServices/soap/**course.rest** »
5. Dans votre « cmd » taper la commande :
 - a. `npm install`
 - b. Ceci va télécharger les dépendances node.js sur internet via le logiciel « npm ». Les dépendances sont décrites dans votre fichiers « package.json »
6. Ouvrez Visual Studio Code sur ce répertoire « C:/TP/WebServices/soap/**course.rest** ».

3. Exécution du serveur

1. Pour connaître la version de node.js sur votre poste, il faut exécuter la commande

- a. `node -v`

Question : Quelle version de « node.js » est installée sur votre poste ?

Réponse : _

La version installée est la v12.22.9

- b. `npm -v`

Question : Quelle version de « npm » est installé sur votre poste ?

Réponse : _

La version installée est la v8.5.1

2. Pour exécuter le serveur, il faut réaliser la commande suivante depuis votre répertoire « C:/TP/WebServices/soap/**course.rest** » :
 - a. `node src/serverSoap`

Votre serveur est maintenant démarré. Il contient un service SOAP qui propose 2 méthodes.

Le module open source utilisé pour réaliser le serveur SOAP est :

<https://github.com/vpulim/node-soap>

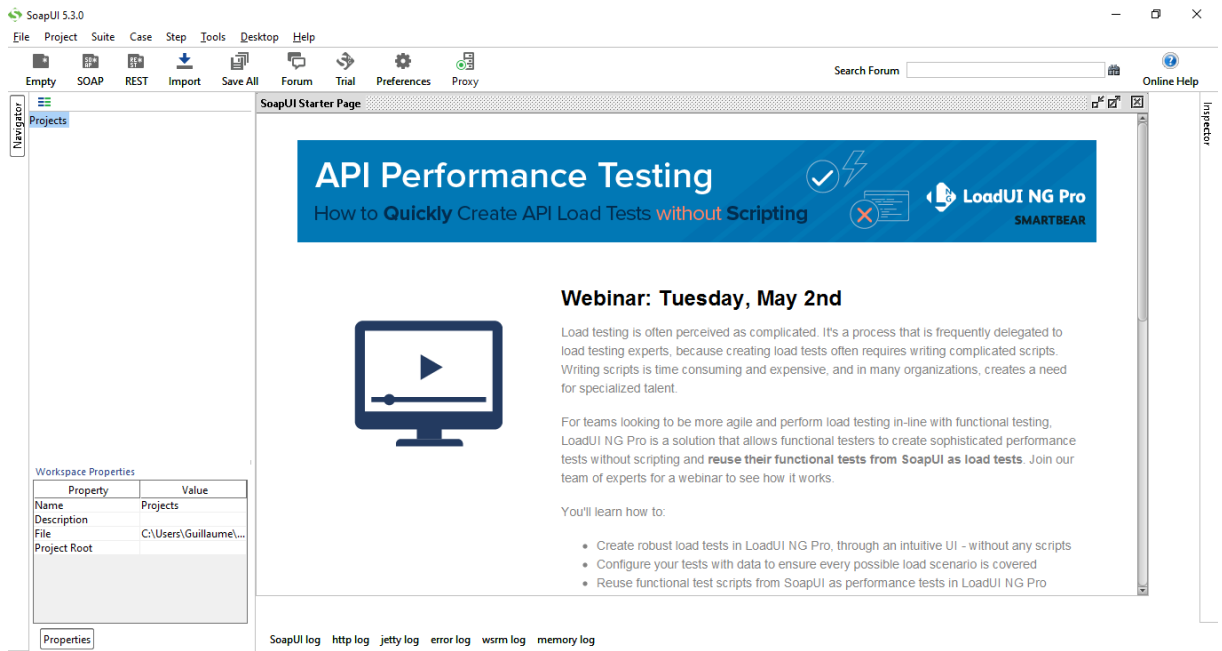
Question : Quel est l'URL du fichier WSDL ? Pour information, cette URL est accessible via un navigateur WEB. Une partie de la réponse est dans le code JavaScript que vous venez de télécharger.

Réponse :

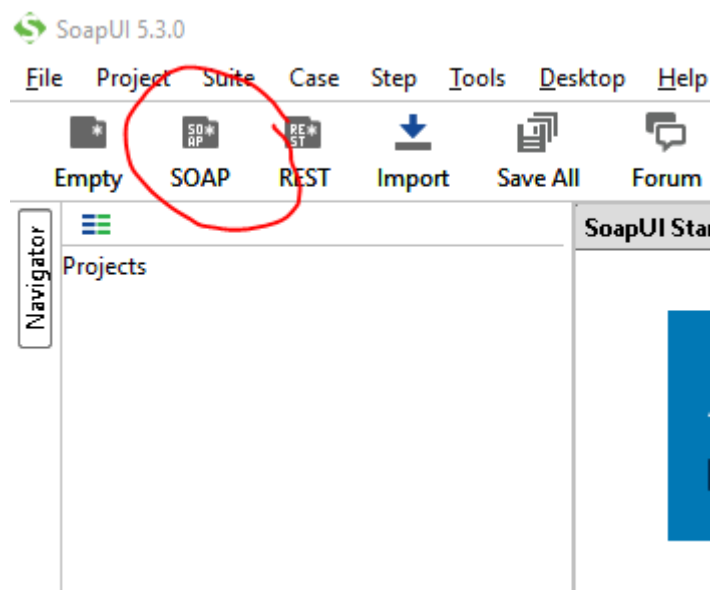


3. Consommer le service SOAP

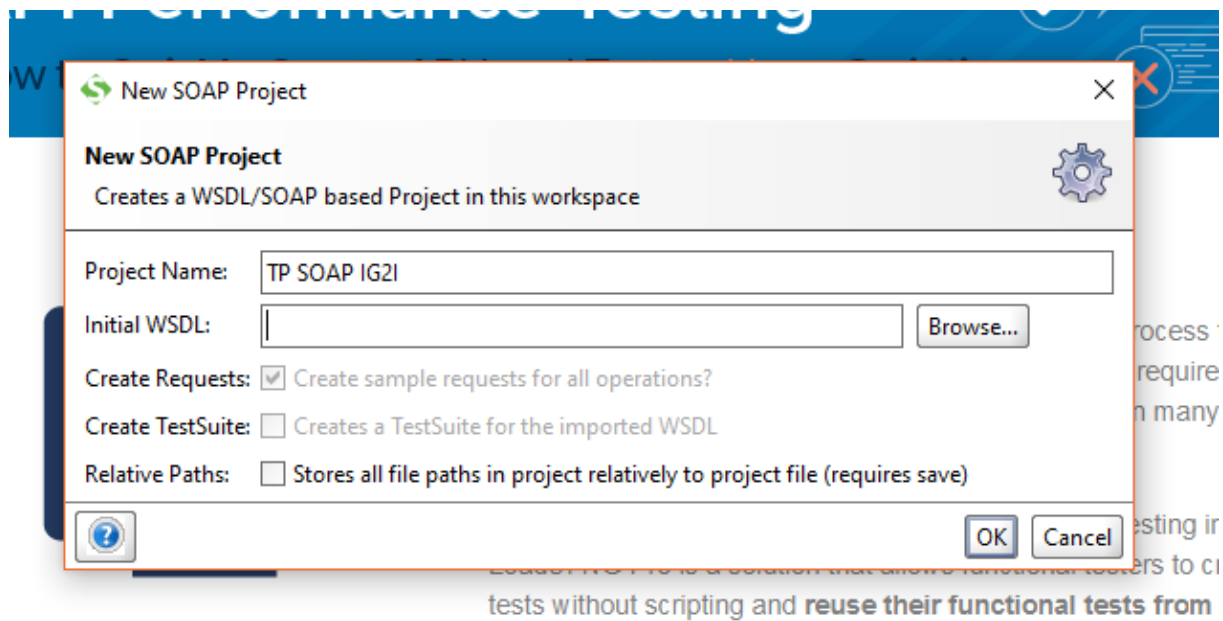
1. Exécutez le logiciel SOAP UI



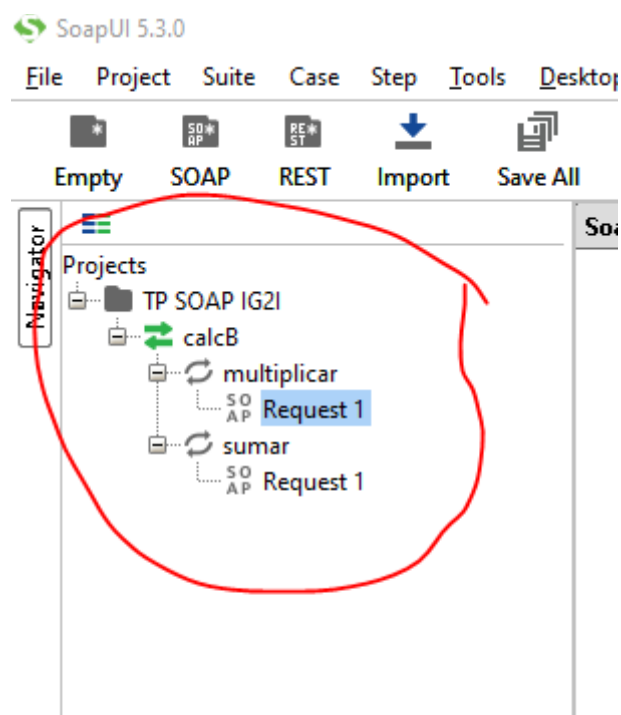
2. Ajoutez une référence vers votre Service SOAP.



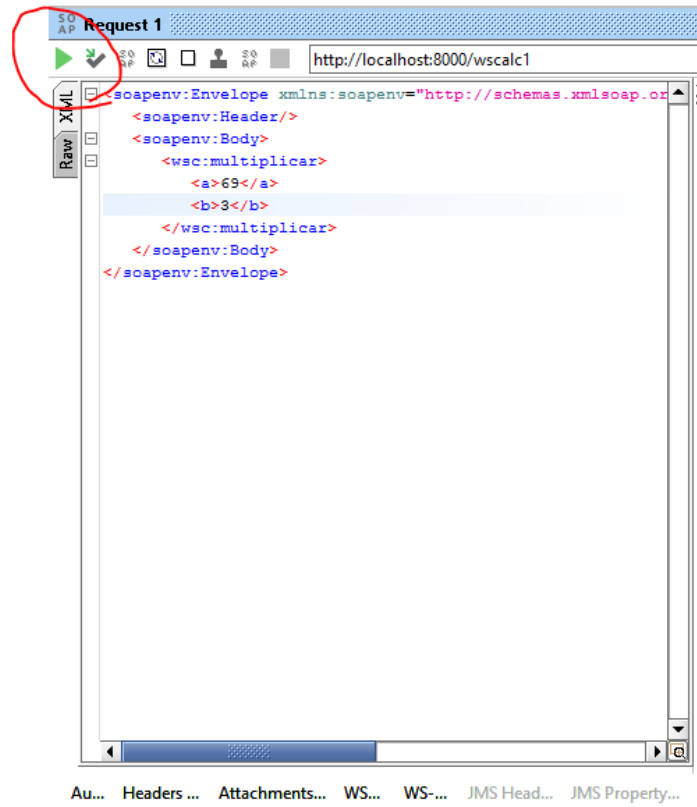
Cliquer sur le bouton encadré en rouge



Renseigner l'url du fichier wsdl



Résultat une fois le fichier importé



Bouton pour appeler un service

- Appeler la méthode « multiplicar » du service SOAP « wscal1 » avec les paramètres a=112 et b=99

Question : Quelle est la réponse de la méthode « multiplicar » du service SOAP « wscal1 » appelée avec le paramètre a=112 et b=99 ?

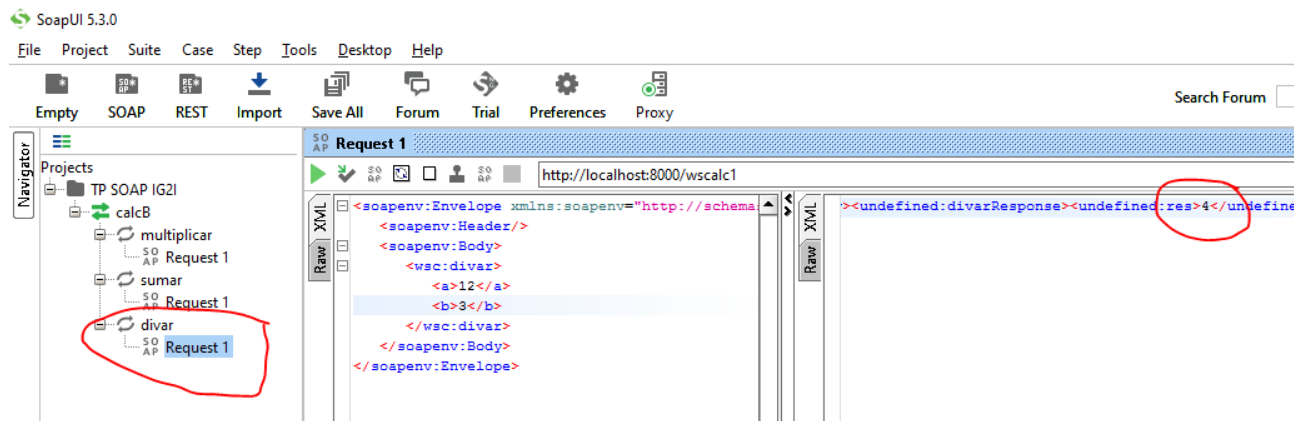
Réponse :

```
<?xml version="1.0" encoding="utf-8"?><soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
><soap:Body><undefined:multiplicarResponse><undefined:res>11088
</undefined:res></undefined:multiplicarResponse></soap:Body></
soap:Envelope>
```

On retrouve bien le résultat de la multiplication 11088 malgré le fait qu'il ait de nombreux « undefined »

4. Création d'une méthode SOAP

- Ajoutez à votre webservice une méthode « divar » qui va permettre de réaliser une fraction avec en entrée 2 paramètres ; « a » le numérateur et « b » le dénominateur tel que a/b. « a » et « b » étant tous les deux des entiers supérieurs à 0.



Question : Quelles sont toutes les manipulations que vous avez dû réaliser afin d'ajouter cette méthode et de la tester ?

Réponse :

1/ Ajouter la fonction divar dans le fichier serverSoap.js

```
divar: function (args) {
    console.log('divar called');
    console.log(args);
    var n = parseInt(args.a) / parseInt(args.b);
    return {
        res: n
    };
}
```

2/ Modifier le fichier wascalc1.wsdl

Il faut ajouter les bindings, les messages et les portTypes
Pour la construction, je me base sur multiplier déjà existant.

3/ Relancer le serveur SOAP

Test de la méthode : a =5 et b=5

```
<?xml version="1.0" encoding="utf-8"?><soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
><soap:Body><undefined:divarResponse><undefined:res>1</undefin
ed:res></undefined:divarResponse></soap:Body></soap:Envelope>
```

On a bien le résultat

5. Code Erreur

1. Dans le cas où le dénominateur « b » est égal à 0, votre professeur souhaite que votre service retourne un code erreur SOAP normalisé.
2. Réalisez l'implémentation à l'aide de la documentation : <https://github.com/vpulim/node-soap> et les valeurs ci-dessous :
 - a. **Code** : 500
 - b. **Reason** : b cannot be 0
3. Réalisez un appel via SOAP UI avec les valeurs ci-dessous :
 - a. a=2
 - b. b=0

Question : Quel est le nom complet de la balise « parente » contenant les informations sur l'erreur ?

Réponse :

Pour réaliser cette amélioration on modifie la fonction `divar` dans le fichier `serverSoap.js` en ajoutant le code suivant :

```
if (parseInt(args.b) === 0) {  
  throw {  
    Fault: {  
      Code: {  
        Value: 'soap:Sender',  
        Subcode: { value: 'rpc:BadArguments' }  
      },  
      Reason: { Text: 'b cannot be nul' },  
      statusCode: 500  
    }  
  };  
}
```

On obtient le résultat suivant :

```
<?xml version="1.0" encoding="utf-8"?><soap:Envelope  
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"  
><soap:Body><soap:Fault><soap:Code><soap:Value>soap:Sender</s  
oap:Value><soap:Subcode><soap:value>rpc:BadArguments</  
soap:value></soap:Subcode></  
soap:Code><soap:Reason><soap:Text>b cannot be  
nul</soap:Text></soap:Reason></soap:Fault></soap:Body></soap:En  
velope>
```

Le nom de la balise parente est soap:Fault

6. Questions

Question : Est-ce possible d'écrire un client en node.js (afin de remplacer SOAP UI dans le TP) ?

Réponse :

Oui il semblerait qu'il existe des librairies qui de remplacer SOAP UI par un client node.js.
<https://github.com/mhzed/tinysoap>

Question : Quel est le protocole utilisé pour la communication client/serveur dans ce TP ?

Réponse :

On utilise http
`const http = require('http');`

Question : Quel est le format de message utilisés pour l'échange d'information client/serveur ?

Réponse :

On utilise SOAP
`const soap = require('soap');`

Question : A quoi sert le fichier WSDL ?

Réponse :

Il permet d'interroger l'API SOAP à l'aide de messages. Le fichier permet que tous les clients savent comment interagir avec le serveur