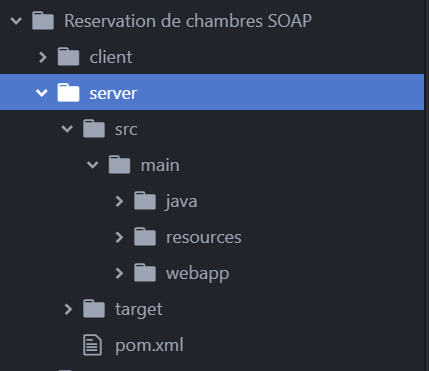
# TP Web Service sécurisés

# Partie I

### 1 – Mise en place de l’application SOAP

**1a**. Nous avons commencé par créer un répertoire « ***Reservation de chambres SOAP*** » pour la mise en place de la partie API SOAP. Ce répertoire est divisé en deux sous-répertoires « ***server*** » et « ***client*** ».  
A l’aide de Maven en ligne de commande dans le répertoire « ***server*** », nous avons généré l’arborescence et les fichiers de base du projet de type webapp.



Pour cette partie, nous avons décidé de créer notre programme Java « à l’ancienne » avec notre éditeur Atom et les commandes ***javac*** et ***java*** pour compiler et exécuter le code.   
Nous utiliserons un dépôt Github et un Git en ligne de commandes pour gérer le partage du code.

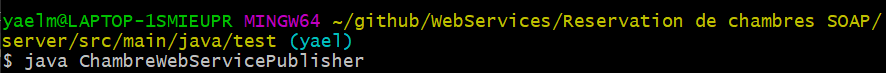
**1b**. Nous avons créé nos deux POJO représentant les objets principaux pour la gestion des réservations de chambres d’hôtel : ***chambre*.java** et ***reservation.java.***

L’interface ***ChambreWebService*** constitue la base de notre WebService. Nous l’avons implémentée dans la classe ***ChambreWebServiceImpl***. C’est dans cette classe que figureront les Web Methods utiles à l’API.

Nous avons ensuite codé les fonctions utiles au système de réservation.

**1c**. Afin de ne pas utiliser d’application tierce et tout centraliser dans notre code java, nous avons décidé de publier notre Web Service sur le réseau via l’objet Endpoint.   
Pour cela, nous avons créé la classe ***ChambreWebServicePublisher***, constituant le main de notre projet.

Le serveur est prêt. Pour le lancer, il suffit de se rendre dans le répertoire ***server/src/main/java/test*** et de lancer le programme java compilé à l’aide de ***java***:



Recompiler la classe à l’aide ***javac*** si nécessaire :



Le serveur tourne.

**1d**. Nous avons ensuite testé notre serveur en accédant au WSDL. Le fichier XML s’affiche bien dans notre navigateur.

Pour y accéder, saisissez dans votre navigateur favori cette URL : <http://localhost:10000/ReservationVoyage?wsdl>

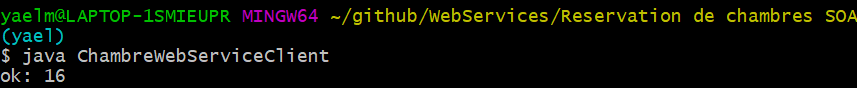
### 2 – Construire un client Java SOAP à partir du WSDL

2a. Nous restons avec notre simple éditeur Atom et notre java en ligne de commandes. Nous travaillons dans le répertoire « **Reservation de chambres SOAP/client** « .

2b. Avec Maven, nous générons les classes clientes dans un package « client » à partir du WSDL à l’aide de la commande suivante :

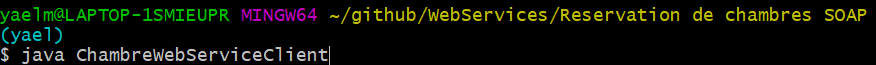


2c. On code notre classe cliente principale qui devra accéder au Web Service créé précédemment. Il s’agit de la classe ***ChambreWebServiceClient*** dans le package ***client***. On compile et exécute cette classe. Afin de tester simplement, nous avions créé une fonction très sommaire dans notre Web Service renvoyant « ok » suivant d’un numéro de chambre pour valider la connexion.

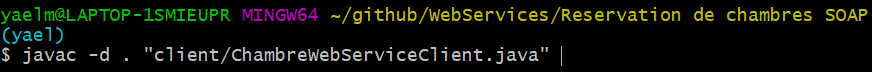


Le client affiche bien le résultat de cette fonction, tout semble fonctionner.

Afin de lancer le client, se placer dans le répertoire « ***Reservation de chambres SOAP*** » et exécuter la classe compilée :



Recompiler si nécessaire avec ***javac*** :



### 4 – Mise en place de l’application REST

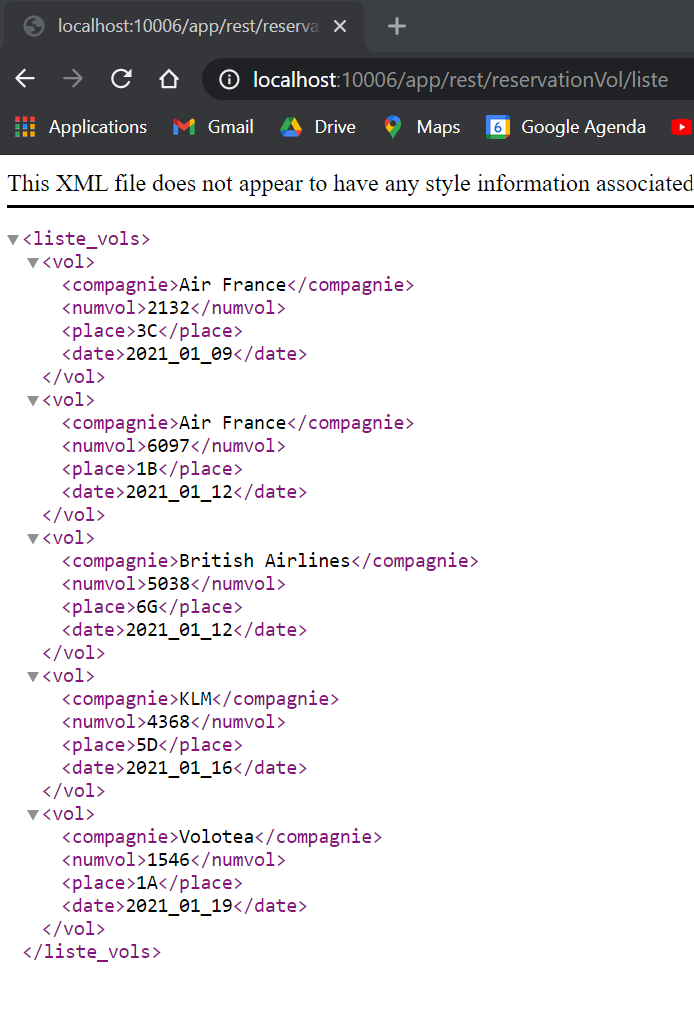
**4a**. Cette fois-ci, nous utilisons un IDE (Eclipse), ce qui nous facilitera la tâche pour crée notre application REST. Nous créons un nouveau projet Maven de type Webapp.

Le contenu de cette application figure dans le répertoire « ***Reservation de vols REST/Application REST*** » .

Dans le fichier ***pom.xml***, nous ajoutons les dépendances utiles à l’application : le servlet javax, Jersey, Json ainsi que le plugin Tomcat pour Maven, qui nous servira de serveur http.

**4b 4c.**

Le service reste est accessible à <http://localhost:10006/app/rest/reservationVol/liste>



Comment l’exécuter :

4d. Passage en JSON

4e. Code source du client SOAP fonctionne séparément mais pas dans ce projet Maven

5 -