INFO802/ETRS804

TD1 – Déploiement d'un Service Web

Ce tutoriel vous guide pas-à-pas pour développer, déployer et tester un Service Web très simple sur un serveur Glassfish en mode code-first avec l'API JAX-WS 2.2 et la pile SOAP Metro.

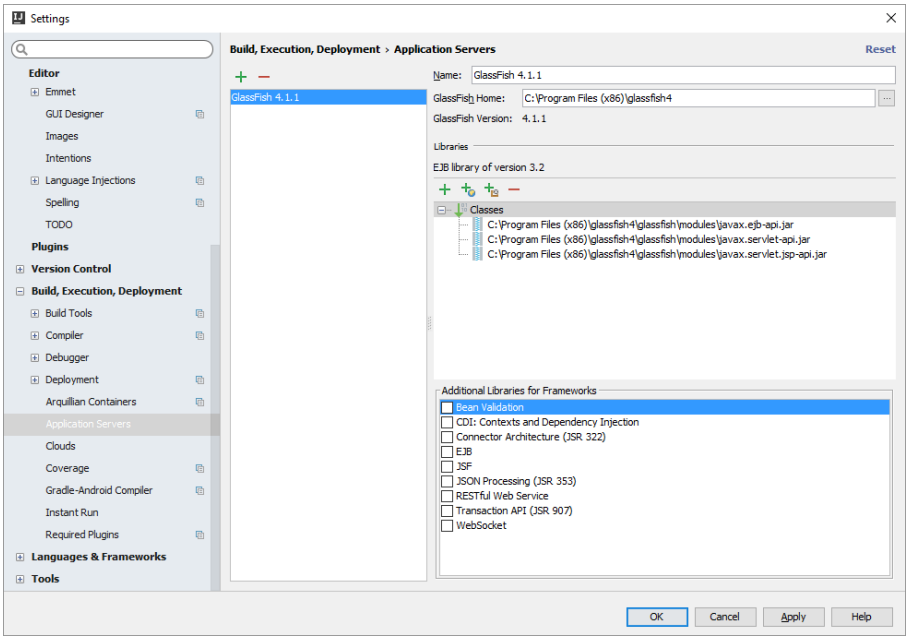
Cette API permet de générer automatiquement des classes Java à partir de fichiers descripteurs (WSDL) et de créer des points de terminaison (EndPoint) pour l'envoi et la réception de messages.

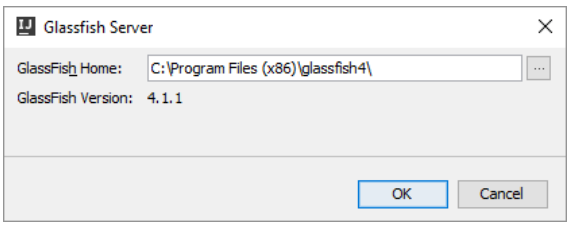
Après avoir terminé ce tutoriel, vous devez être en mesure de développer un service web SOAP simple à l'aide de l'environnement IntelliJ (ou tout autre Java IDE après avoir lu la documentation adéquate).

Etape 1 – Environnement de développement

1. IntelliJ IDEA (https://www.jetbrains.com/idea/download), vous devez enregistrer pour obtenir une licence étudiant : https://www.jetbrains.com/shop/eform/students
2. **JDK 1.8** (http://www.oracle.com/technetwork/java/javase/downloads)
3. Glassfish Server (https://javaee.github.io/glassfish/download)

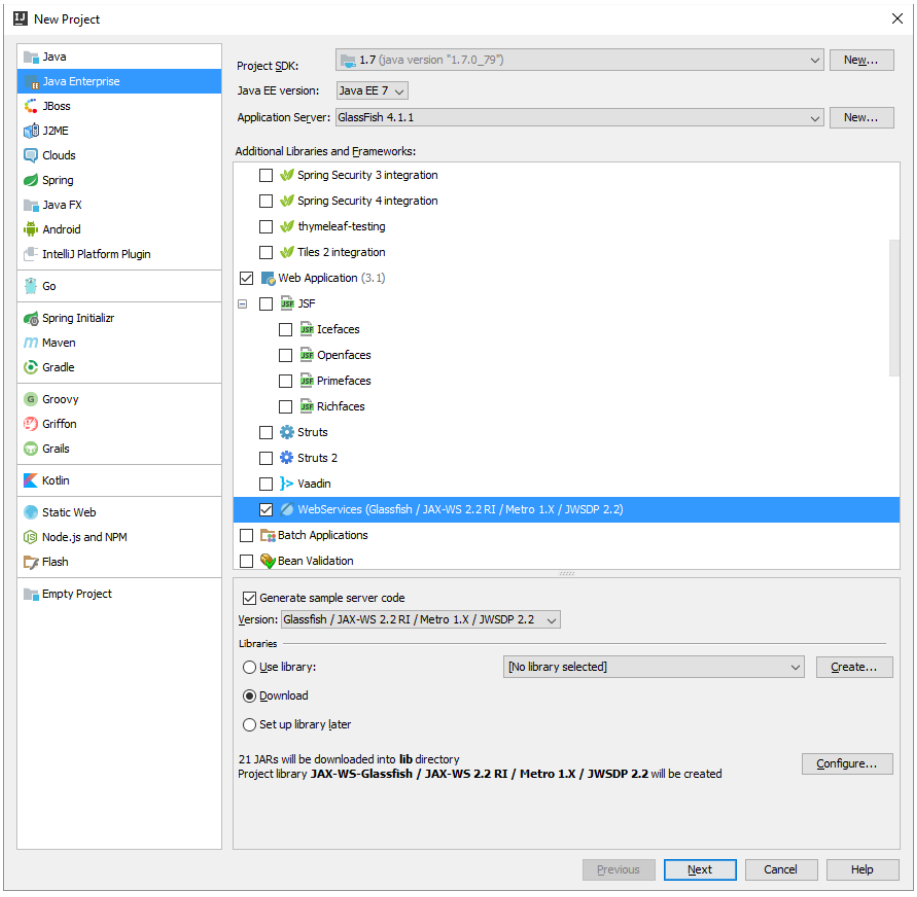
Après avoir téléchargé et désarchivé Glassfish (dans un répertoire avec autorisation d'écriture), vous devez l'intégrer dans l'IDE IntelliJ en ouvrant la boîte de dialogue *File | Settings | Build, Execution, Deployment | Application Servers*. Ajoutez un serveur Glassfish à l'aide du signe +



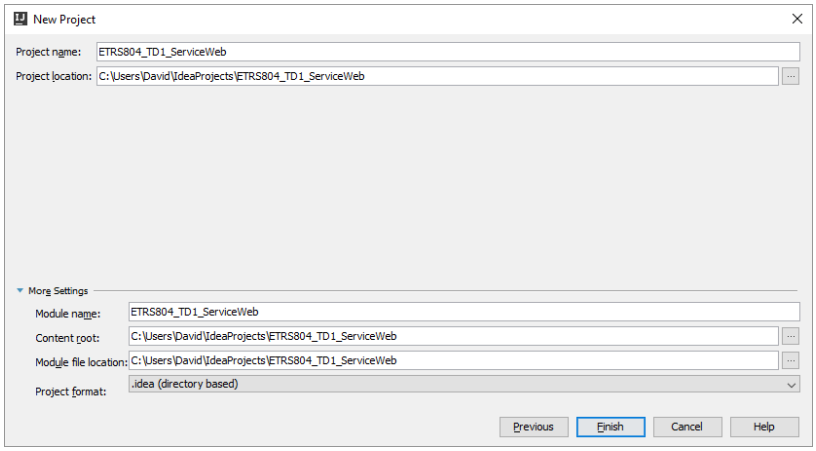


Etape 2 – Génération d'un service web

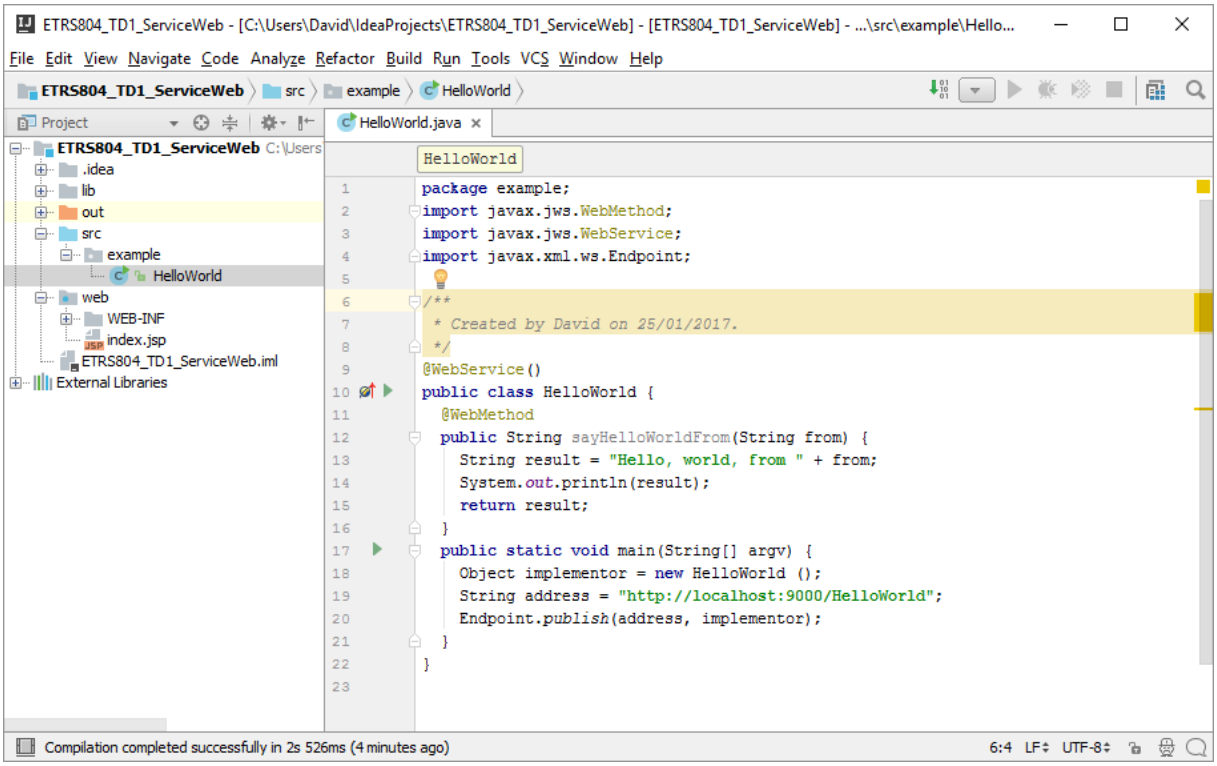
1. Créez un nouveau projet *Java Entreprise* et cochez la checkbox *WebServices*
2. En cochant *Generate Sample…,* IntelliJ va générer le code d'un exemple HelloWorld
3. Dans le cadre du TP, sélectionnez le déploiement sur Glassfish car les plugins sont installés par défaut mais vous pourriez en choisir un autre (Jboss, Axis, Websphere, …) si vous les aviez préalablement intégrés à votre IDE.
4. Cochez également *Download* afin que l'IDE télécharge les librairies adaptées à votre configuration



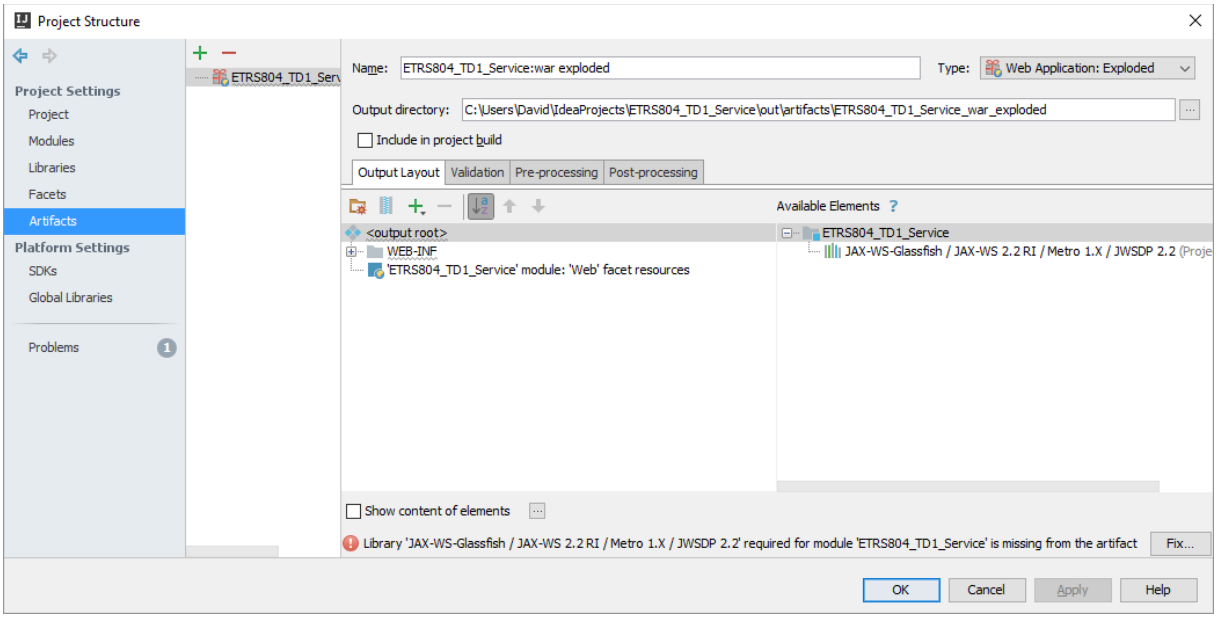
5. Nommez votre projet et validez



6. Analysez le code ainsi généré par l'IDE :
    a. A quoi correspondent les annotations @WebService et @WebMethod ?
    b. Quel est, dans ce contexte, le rôle de la méthode main ? Est-il indispensable de la conserver ?
    c. A quel adresse votre service est-il invocable ?



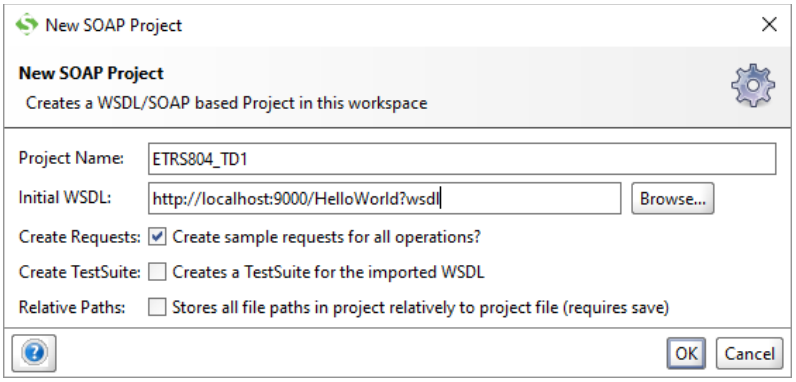
7. Avant de déployer votre service, vérifiez que votre projet soit bien configuré pour publier. (*File | Project Structure*). L'IDE peut vous proposer de « fixer » un éventuel problème.



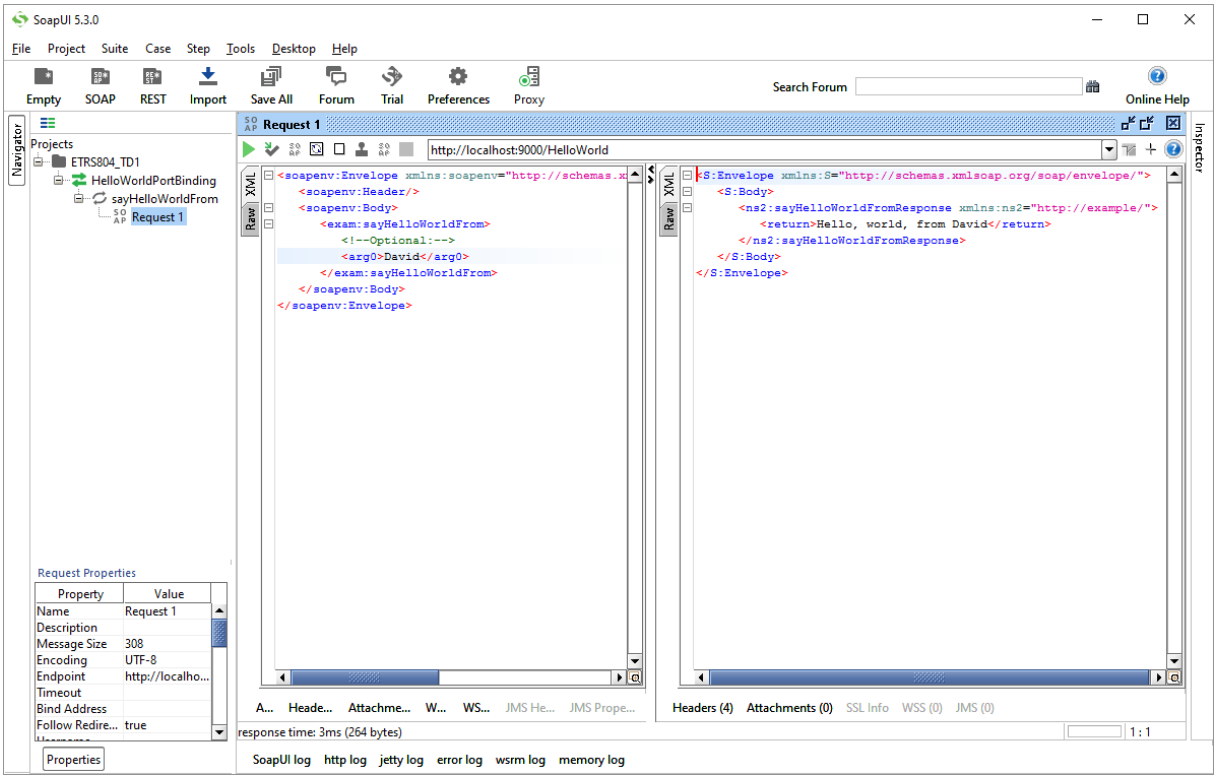
8. Depuis un navigateur, testez l'accès au Service Web et observez le fichier WSDL généré.
    a. A quoi sert ce fichier ?
    b. Où se trouve la définition des types de données qui vont transiter ?
    c. Quel est le protocole à utiliser pour invoquer votre service ?
    d. Comment faites-vous le lien entre les messages décrits dans le WSDL et l'implantation du service ?

Etape 3 - Testez votre Service Web

1. Installez la version opensource de SoapUI (www.soapui.org). Cet outil permet d'invoquer pour inspecter le fonctionnement de services web.
2. Créez un nouveau projet SOAP et entrez l'URL de localisation du fichier WSDL



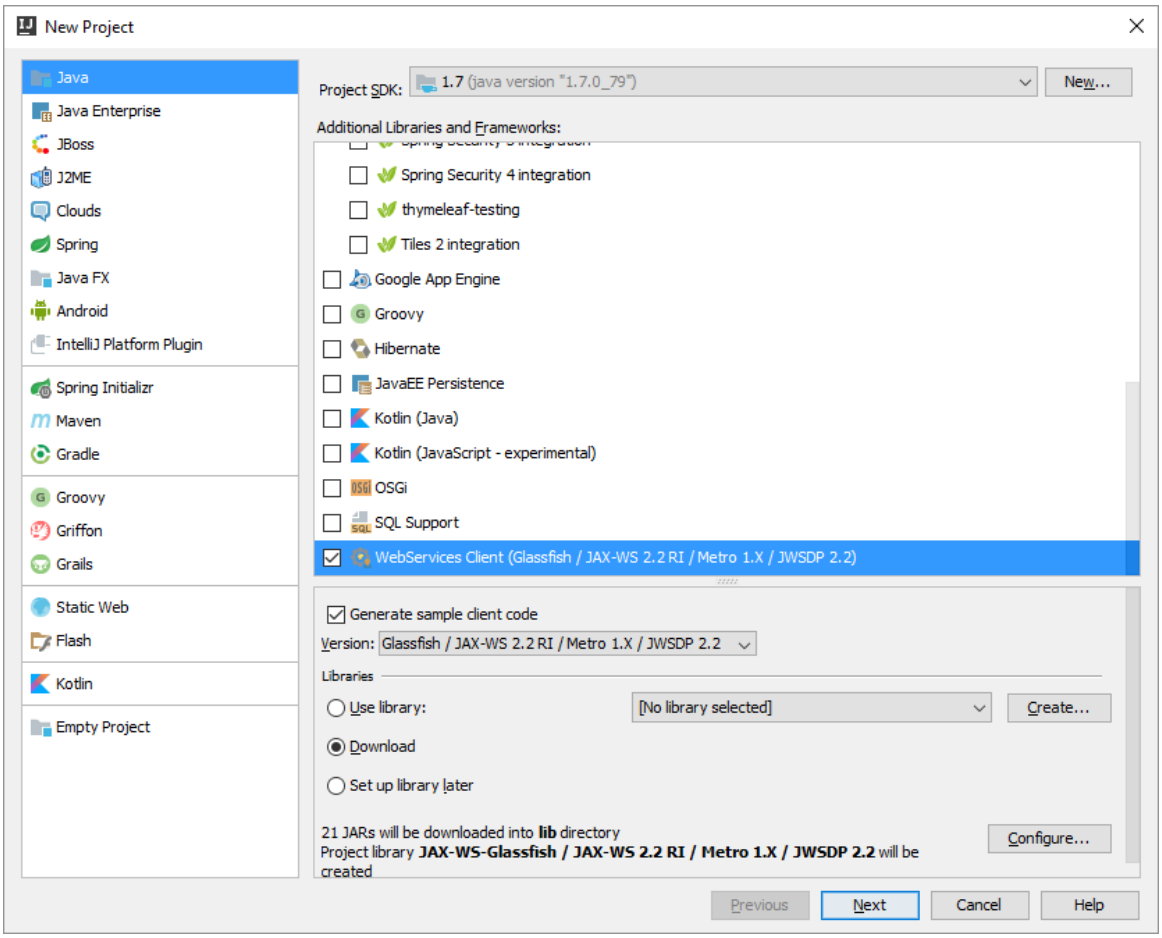
3. Entrez les arguments attendus par le Service Web et exécutez-le.



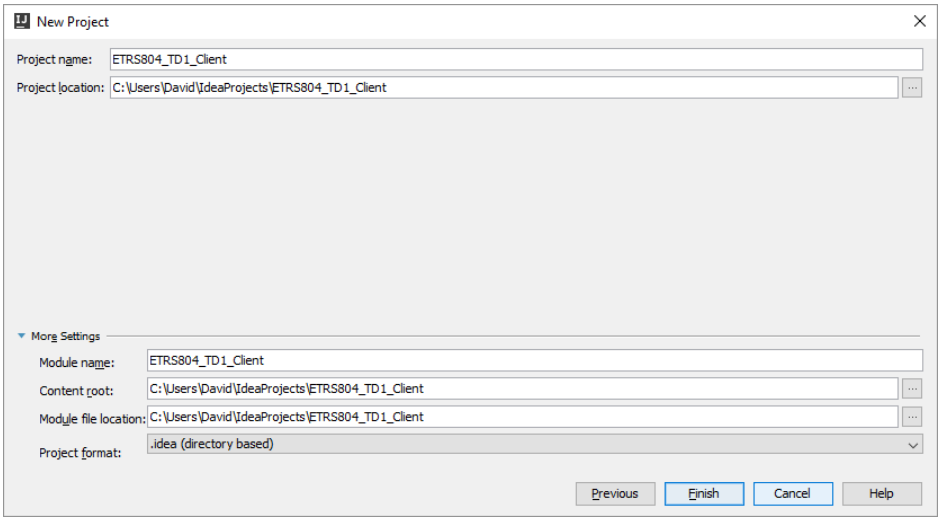
4. Testez le service web d'un de vos camarades

Etape 4 – Génération d'un client

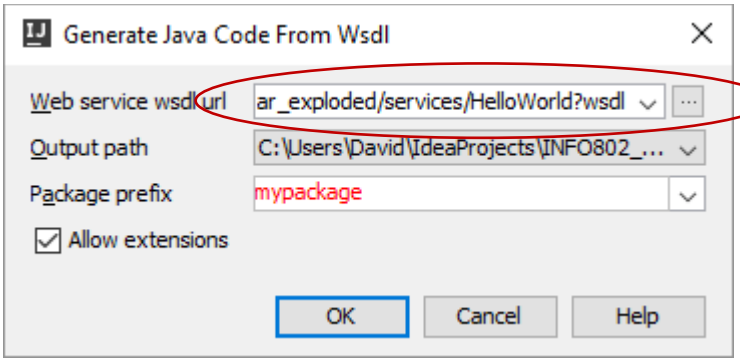
1. Pour générer un client, avec IntelliJ, créez un nouveau projet **Java** et cochez la checkbox *WebServices Client*



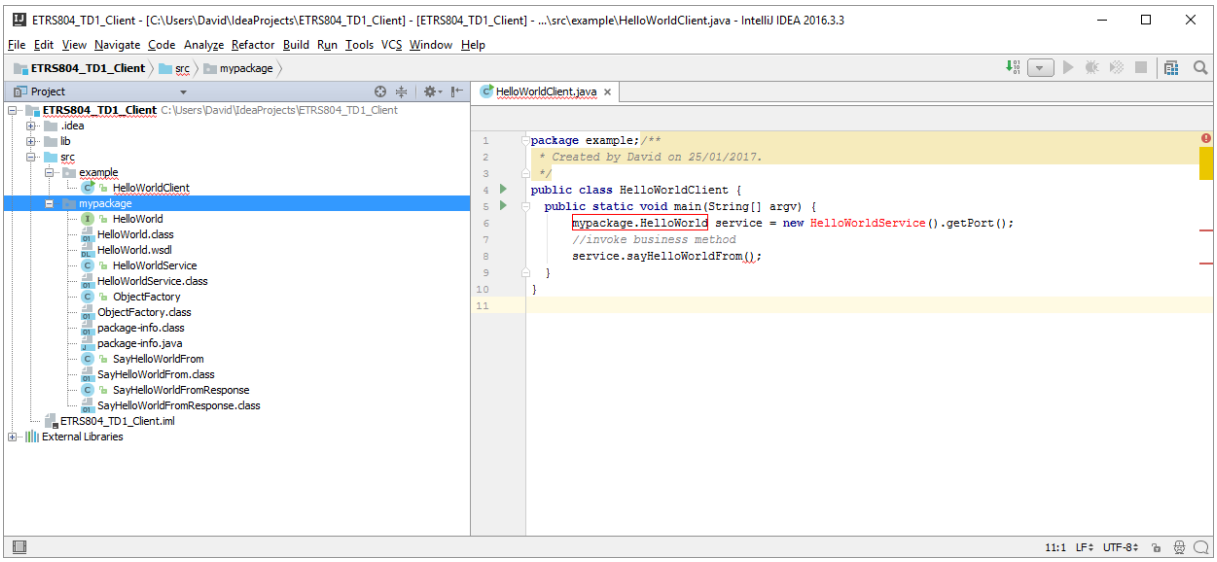
2. Nommez votre projet et validez



3. Attention, pour générer le client, il faut préciser l'URL du fichier WSDL



4. Si vous rencontrez une erreur d'écriture AccessExternalSchema
    a. Créez un fichier nommé : jaxp.properties
    b. Ecrivez dans ce fichier une ligne « javax.xml.accessExternalSchema = all »
    c. Placez ce fichier dans le répertoire ../jdk/JRE/lib
5. Par défaut, l'IDE a généré une classe *HelloWorldClient* avec un *main*
    a. A quoi correspond la classe *HelloWorldService* ?
    b. Est-ce que la méthode *getPort*() existe ? Quelle méthode devez-vous invoquez ?
    c. A quoi correspond la méthode *sayHelloWorldFrom* ?
    d. Quels sont les arguments attendus par cette méthode ?
    e. Affichez le résultat en console.



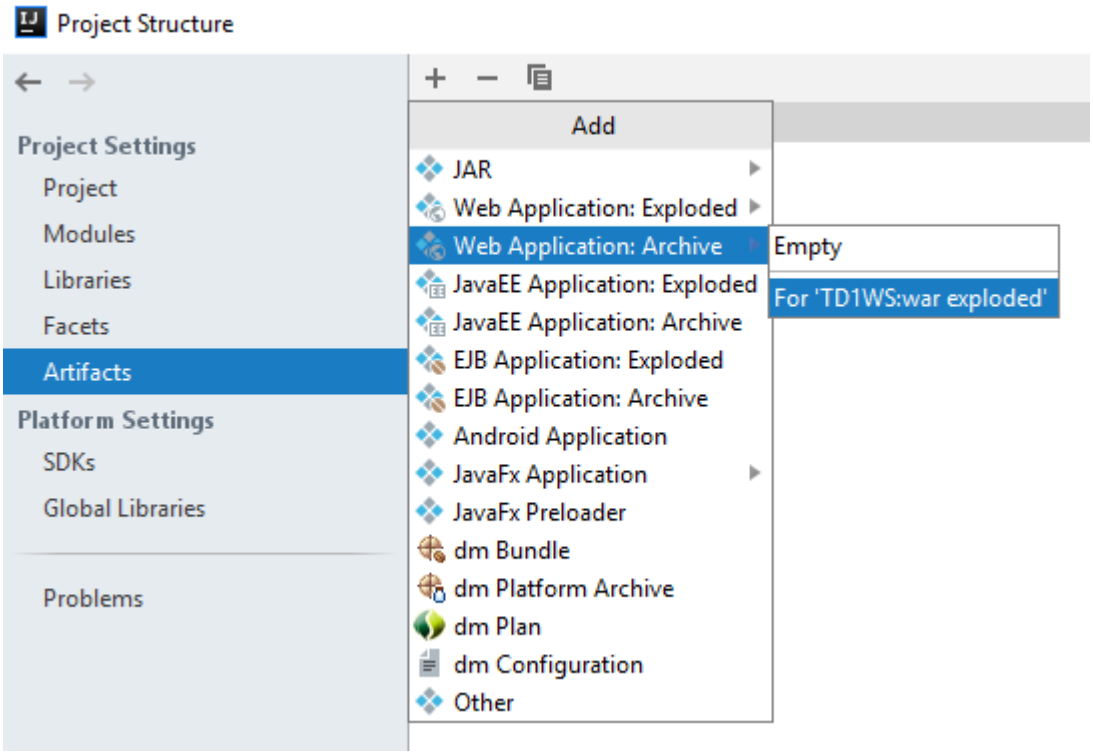
6. Testez le Service Web d'un camarade et modifiez les arguments

Etape 5 - Modifiez votre Service Web

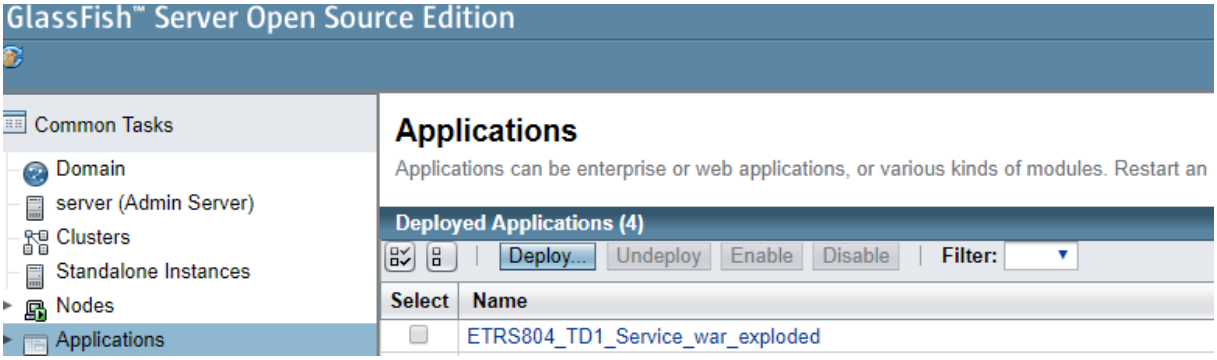
7. Dans votre service web, ajoutez une nouvelle méthode *: int addition (int a, int b)* qui retourne le résultat de l'addition
8. Régénérez votre projet : Build → Rebuild project
9. Redéployiez votre projet sans **redémarrer** le serveur
10. Vérifiez que votre fichier WSDL a bien été mis à jour
11. Dans votre client, régénérez les classes soap : clc droit sur le projet → Web service → Generate Java Code from WSDL
12. Modifiez votre client pour appeler la méthode *addition*

Etape 6 – Générez un war pour exporter sur le serveur Glassfish

13. Créez une archive de votre projet



14. Générez le war dans le répertoire **out** de votre projet : Build → build artifacts
15. Ouvrez la console d'administration de votre serveur Glassfish : localhost : 4848



16. Déployez votre fichier .war via l'interface d'administration et testez votre service web

17. Vous pouvez désormais démarrer votre serveur en ligne de commande :
`asadmin start-domain domain1`