Web services

Pr Mostafa SAADI – ENSAK

L'Ecole Nationale des Sciences Appliquées à Khouribga

saadi_mo@yahoo.fr

11 décembre 2018

Plan du Cours

Context

Les technologies telles que **RMI**, **DCOM** et **CORBA** ont adopté le **SOA** mais ont généralement échoué :

- la diversité des plates-formes utilisées dans les organisations
- leur usage n'était pas adapté à Internet (problème de passage à travers des FireWalls, etc.)
- la lenteur, voire l'absence de réponses sur ce réseau.
- Les applications réparties fondées sur ces technologies offrent des solutions caractérisées par un couplage fort entre les objets.

Definition

On retrouve plusieurs définitions des services Web :

Citation : W3C

Un service Web est un composant **logiciel** identifié par une **URI**, dont les interfaces publiques sont définies et appelées en **XML**. Sa définition peut être découverte par d'autres systèmes logiciels. **Les services Web** peuvent interagir entre eux d'une manière prescrite par leurs définitions, en utilisant des messages XML portés par les protocoles Internet.

Definition

Citation : Dico du Net

Une technologie permettant à des applications de dialoguer **à distance via Internet indépendamment** des **plates-formes** et des **langages** sur lesquels elles reposent.

Citation : Wikipédia

Un service Web est un **programme informatique** permettant la communication et **l'échange de données** entre applications et systèmes hétérogènes dans des environnements distribués. Il s'agit donc d'un ensemble de fonctionnalités exposées sur internet ou sur un intranet, par et pour des applications ou machines, sans intervention humaine, et en temps réel.

Definition

En résumé

Un service Web est tout simplement un **programme** accessible au moyen d'Internet, qui utilise un système de messagerie standard XML, et n'est lié à aucun système d'exploitation ou langage de programmation.

Caractéristiques :

- **Réutilisable**
- **interopérabilité** entre services
- **Indépendamment** de
  - la plate-forme (UNIX, Windows,...)
  - l'implémentation (VB, C#, Java, ...)
  - l'architecture sous-jacente (.NET, J2EE, Axis ...)

L'intérêt d'un Service Web

Pour quoi faire

Les **services Web** fournissent un **lien entre applications**. Ainsi, des applications utilisant des technologies différentes peuvent envoyer et recevoir des données au travers de protocoles compréhensibles par tout le monde. ;)

- Les services Web permettent d'interconnecter :
  - Différentes entreprises
  - Différents matériels
  - Différentes applications
  - Différents clients
- Distribuer et intégrer des logiques métiers
- Web sémantique
- Les services Web sont faiblement couplés : le demandeur du service ne connaît pas forcément le fournisseur.

Avantages des Services Web

- Les services Web sont normalisés car ils utilisent les standards XML et HTTP pour transférer des données.
- Ils sont compatibles avec de nombreux autres environnements de développement.
- Ils sont indépendants des plates-formes (ils permettent aux entreprises d'offrir des applications accessibles à distance par d'autres entreprises).
- Les services Web n'imposent pas de modèles de programmation spécifiques. (ne sont pas concernés par la façon dont les messages sont produits ou consommés par des programmes).
- Les services Web représentent la façon la plus efficace de partager des méthodes et des fonctionnalités.
- Ils réduisent le temps de réalisation en permettant de tirer directement parti de services existants.

Objectifs

- Remplacer les protocoles actuels (RPC,DCOM,RMI) par une approche entièrement ouverte et interopérable, basée sur la généralisation des serveurs Web avec scripts CGI (Interface de passerelle commune : envoyer un HTML aux clients).
- Faire interagir des composants hétérogènes, distants, et indépendants avec un protocole standard (SOAP).
- Dédiés aux applications B2B (Business to Business), EAI (Enterprise Application Integration), P2P (Peer to Peer).
- La technologie des services Web est un moyen rapide de distribution de l'information entre clients, fournisseurs, partenaires commerciaux et leurs différentes plates-formes. Les services Web sont basés sur le modèle SOA .

Les caractéristiques d'un service Web

- Il est accessible via le réseau ;
- Une nouvelle technologie des objets distribués
  - Invocation distante des services Web : SOAP (IIOP)
  - Description des services Web : WSDL (IDL)
  - Enregistrement et découverte de services Web : UDDI (NameService)
- Basés sur des standards XML (la base de l'informatique distribuée sur Internet)
  - Standards du W3C : XML, SOAP, WSDL
  - Standards industriels : UDDI, ebXML
  - Propriétaires : DISCO, WSDD, WSFL, ASMX,...
- Implémentations actuelles :
  - Microsoft .Net
  - Java : J2EE + Web services (WSDP = JAXP,JAX-RPC,JAXM...)
  - Apache SOAP / Axis, IBM WSTK
  - Oracle, Bea, Iona, Enhydra...

Architecture d'un service Web

Les services Web reprennent la plupart des idées et des principes du Web (**HTTP**, **XML**), et les appliquent à des interactions entre machines. Comme pour le World Wide Web, les services Web communiquent via un ensemble de technologies fondamentales qui **partagent une architecture commune**. Ils ont été conçus pour être réalisés sur de nombreux systèmes développés et déployés de façon indépendante. Les technologies utilisées par les services Web sont **HTTP**, **WSDL**, **REST**, **XML-RPC**, **SOAP** et **UDDI**.

Architecture d'un service Web

REST

REST (Representational State Transfer) est une architecture de services Web. Élaborée en l'an 2000 par *Roy Fiedling*, l'un des créateurs du protocole HTTP, du serveur Apache HTTPd et d'autres travaux fondamentaux, **REST** est une manière de construire une application pour les systèmes distribués comme le World Wide Web.

Architecture d'un service Web

XML-RPC

XML-RPC est un protocole simple utilisant XML pour effectuer des messages RPC. Les requêtes sont écrites en XML et envoyées via HTTP POST. Les requêtes sont intégrées dans le corps de la réponse HTTP. XML-RPC est indépendant de la plate-forme, ce qui lui permet de communiquer avec diverses applications. Par exemple, un client Java peut parler de XML-RPC à un PerlServer.

Architecture d'un service Web

SOAP

SOAP (Simple object Access Protocol) est un protocole standard de communication. C'est l'épine dorsale du système d'interopérabilité. SOAP est un protocole **décrit en XML** et **standardisé** par le W3C. Il se présente comme une enveloppe pouvant être signée et pouvant contenir des données ou des pièces jointes. Il circule sur le protocole HTTP et permet d'effectuer des appels de méthodes à distance.

Architecture d'un service Web

WSDL

WSDL (Web Services Description Language) est un langage de description standard. C'est l'interface présentée aux utilisateurs. Il indique comment utiliser le service Web et comment interagir avec lui. **WSDL est basé sur XML** et permet de décrire de façon précise les détails concernant le service Web tels que les protocoles, les ports utilisés, les opérations pouvant être effectuées, les formats des messages d'entrée et de sortie et les exceptions pouvant être envoyées.

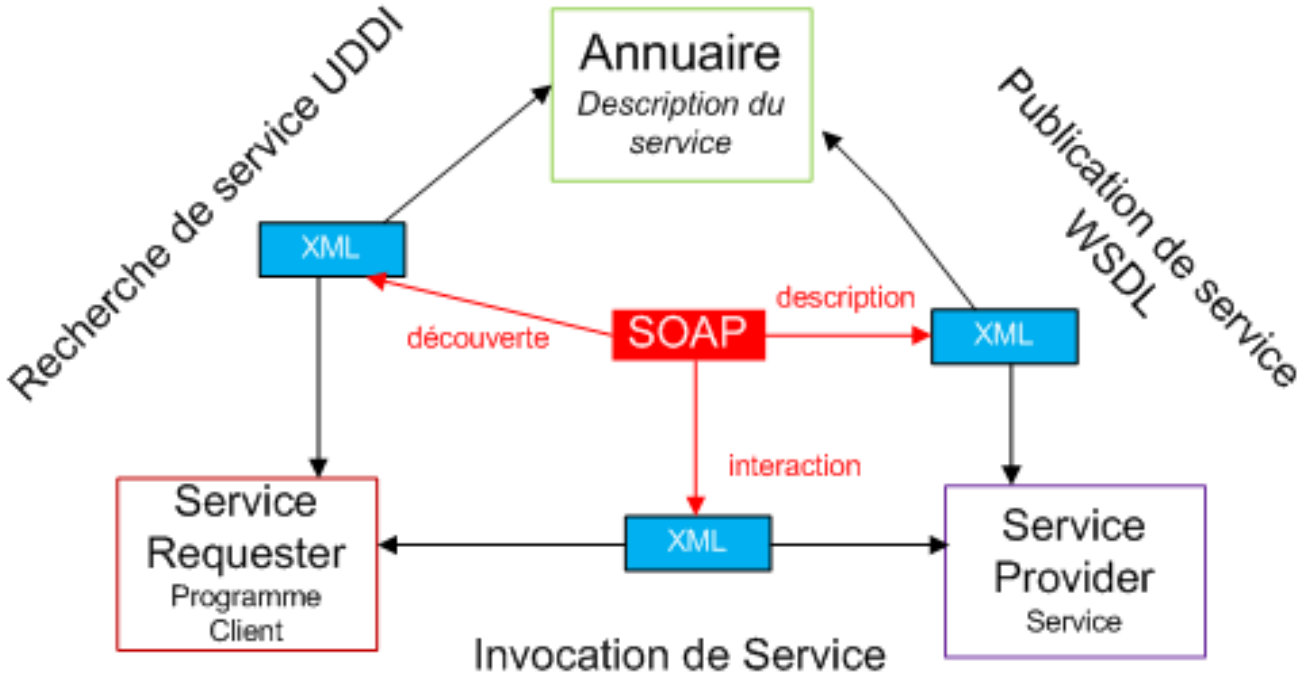
Architecture d'un service Web

UDDI

- **UDDI** (Universal Description, Discovery and Integration) est un annuaire de services. Il fournit l'infrastructure de base pour la publication et la découverte des services Web.
- UDDI permet aux fournisseurs de présenter leurs services Web aux clients.
- Les informations qu'il contient peuvent être séparées en trois types :
  - les pages blanches qui incluent l'adresse, le contact et les identifiants relatifs au service Web ;
  - les pages jaunes qui identifient les secteurs d'affaires relatifs au service Web ;
  - les pages vertes qui donnent les informations techniques.

Fonctionnement des services Web

Le fonctionnement des services Web s'articule autour de trois acteurs principaux illustrés par le schéma suivant :



Cycle de vie complet

1. Etape 1 : Déploiement du service Web(Dépendant de la plate-forme) (Apache : WSDD)
2. Etape 2 : Enregistrement du service Web
   - WSDL : description du service
   - Référentiels : DISCO (local), UDDI (global)
3. Etape 3 : Découverte du service Web
4. Etape 4 : Invocation du service Web par le client

Service Web : définition

- Un service Web est une application logicielle utilisant Internet pour interagir dynamiquement avec d'autres programme en s'appuyant sur d'autres standards sauvant du W3C (World Wide Web Consortium)[1] :

- *XML (Extensible Markp Language)[2] pour le formatage des données*
- *HTTP : pour le protocol de transport*
- *SOAP (Simple Object Access Protocol)[3] : pour le protocole de communication*

- Un service web est identifiée par un **URI**

- Mode de communication entre services : **messages**

- Un service web est accessible sur le Web : Les messages sont véhiculés par des protocoles Web

-Principe : invocation des méthodes sur des services distants ➔ utilisation du protocol SOAP (Simple Object Access Protocol)

-But : permettre à une application de communiquer sur Internet avec le service dont elle a besoin et d'échanger avec lui

[1] : http//www.w3c.org
[2] : http//www.w3c.org/XML/
[3] : http//www.w3c.org/TR/soap

Architecture

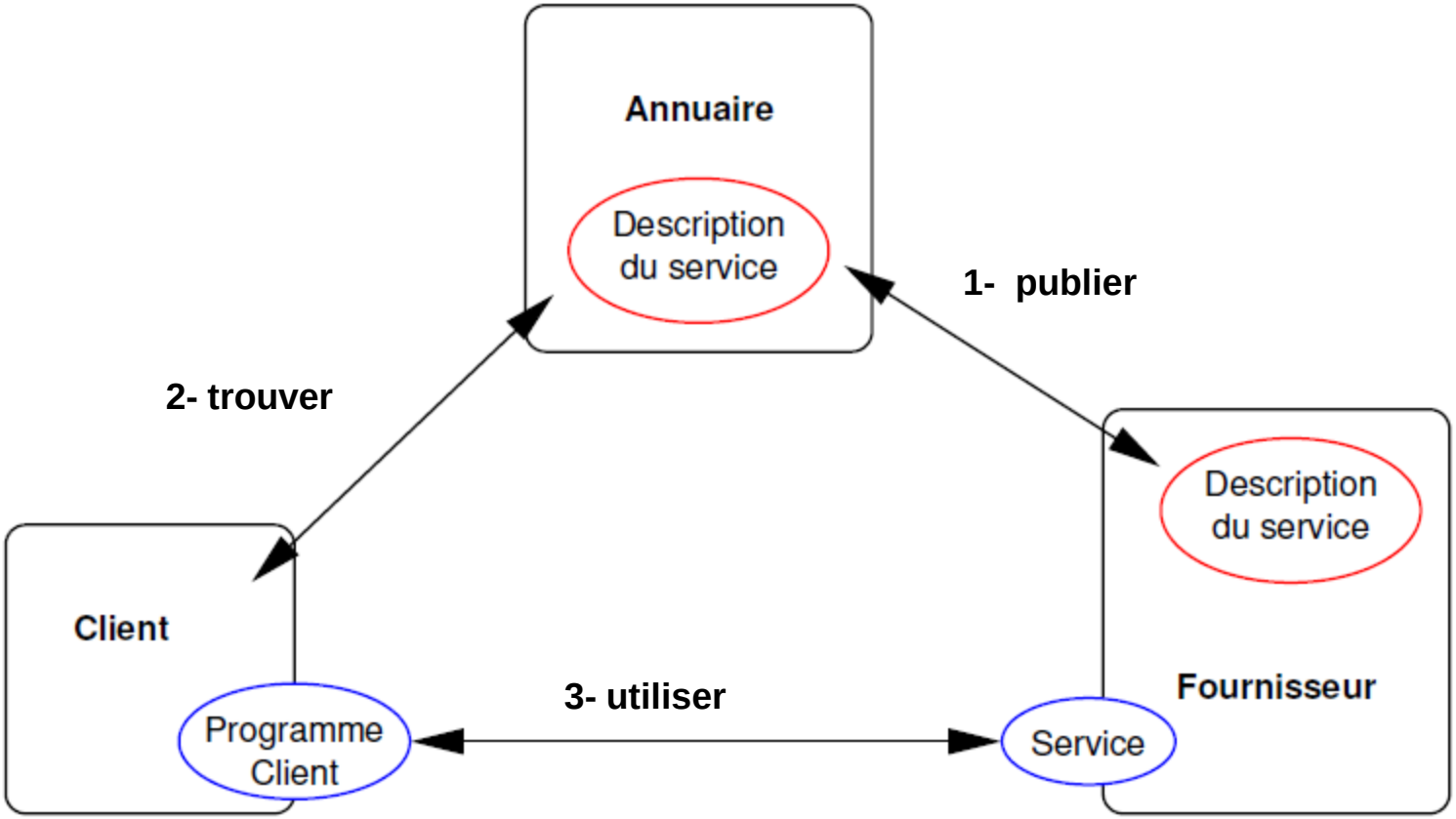
-Dans l'architecture CORBA on a l'ORB, l'IDL
-Dans les services web on retrouve trois composants équivalents :

- Invocation des services web : SOAP
- Description des services web : WSDL, l'équivalent de l'IDL
- Enregistrement et découverte des services : UDDI
  (Uinversal Description Discovery and Integration) est l'équivalent de service de nom CORBA.
  UDDI est peut utilisé bien que normalisé depuis 2005 sous sa version 3 par l'OASIS (complexité …).
  L'URL du service est suffisant à son utilisation == > UDDI n'est pas indispensable

Les partenaires :

- le fournisseur de service (*service provider) :*
  - définit et crée le service
  - publie son interface ainsi que les informations d'accès au service (on utilise WSDL) dans un l'annuaire de service web.
  - implémente le service (les opérations) coté serveur
- l'annuaire :
  - reçoit et enregistre les descriptions de services publiés par les fournisseurs
  - reçoit et répond aux recherches de services lancées par les clients
- le client (*service requestor) :*
  - accède à l'annuaire de service pour effectuer une recherche afin de trouver les services désirés
  - obtient la description du service. Cette interface décrite en langage WSDL fournit l'ensemble des services disponibles. Elle offre la possibilité de vérifier que le contrat correspond bien aux besoins demandés.
  - utilise le service par simple prise en main de l'interface WSDL :
    le client doit maintenant passer les paramètres attendus par le service et assurer la communication avec le serveur. Il doit mettre en place un Proxy, c'est l'objet qui permet la communication (existence des outils pour la génération automatique à partir du fichier wsdl)

Architecture



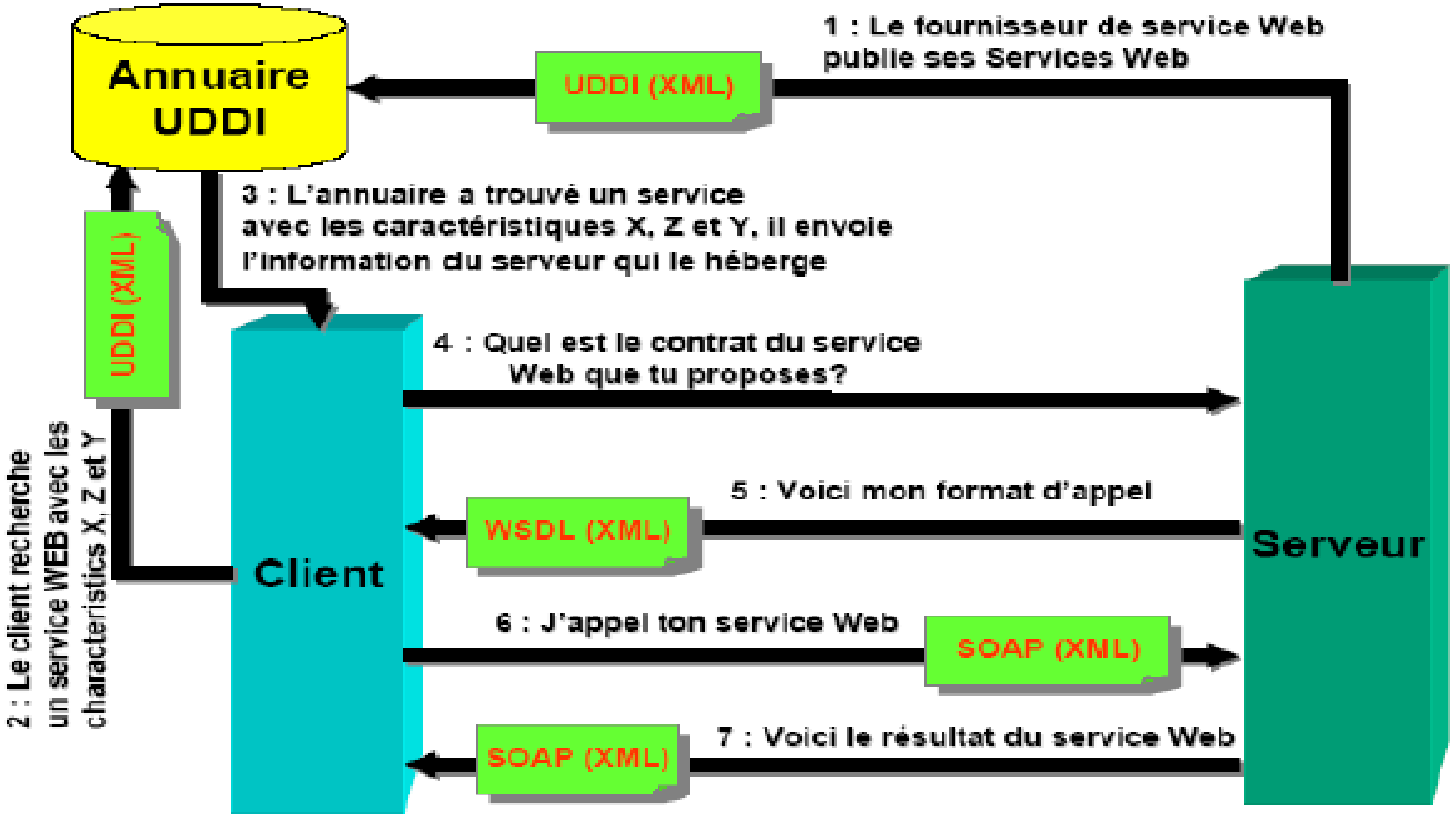
Les standards

-**SOAP (Simple Object Access Protocol) :**
- Achemine les messages : cadre général permettant l'échange de données structurées au format XML
- protocole de transport de ces données basé sur HTTP
- version 1.1 : mai 2000 (http://www.w3c.org/TR/SOAP/)
- version 1.2 : recommandation du W3C depuis juin 2003

-**WSDL (Web Services Description Language) :**
- permet tout comme IDL CORBA de décrire les interfaces de services web
- Il a été normalisé comme note du W3C sous la version 1.1 en mars 2001
- La version 2.0 est une recommandation du 26 juin 2007

- **norme UDDI** (Universal Data Description Interface):
- spécifie la structure des annuaires de services
- permet d'enregistrer et de rechercher des descriptions de services web

Architecture



Source : Samir Tata télécom Sud Paris (ex INT Paris)

Services Web

WSDL : Web Service description Langage

WSDL : un langage d'interface

Besoin : décrire les services disponibles sur un serveur
== > un langage de description de service a été mis en place : le WSDL (Web Service Description Langage).

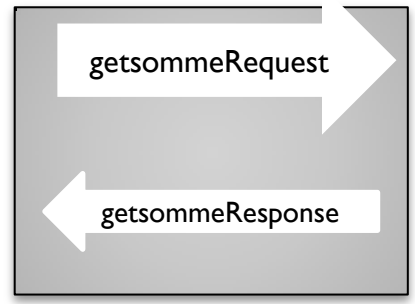
- WSDL est un langage qui permet :
  - de définir l'interface d'un service web (types de données, opérations, entrées, sorties)
  - et de décrire comment invoquer un service web
- Un fichier WSDL est un document au format XML qui décrit :
  - ce que fait le service : liste des opérations,
  - la localisation du service : le nom du service, et le point d'attache (endpoint) le plus souvent l'URI du serveur web
  - comment invoquer le service : les règles d'encodage des arguments, le protocole de transport utilisé

-Une fois muni de cette description, appelée aussi parfois « contrat », le client va pouvoir dialoguer avec les services de manière adéquate.
- On peut générer automatiquement à partir du fichier WSDL un client ou un proxy pour ce service.

Structure d'un document WSDL

1- Types de données :
- description des types de données
- permet de définir des structures de données spécifiques à l'application
- optionnelle

2- les messages
- sont unidirectionnels (sens entrant in, ou sortant out)
- une opération comporte donc en général deux messages : la requête et la réponse
- un message comporte plusieurs arguments : « parts » chaque argument a un nom et un type défini par un schéma



Opération getsomme

```
<wsdl:message name="getsommeRequest">
    <wsdl:part name="in0" type="xsd:int"/>
    <wsdl:part name="in1" type="xsd:int"/>
 </wsdl:message>
<wsdl:message name="getsommeResponse« >
      <wsdl:part name="getsommeReturn" type="xsd:int"/>
  </wsdl:message>
```

Structure d'un document WSDL

3- PortType (ou interface): définit l'interface abstraite du service comportant un ensemble d'opération. Il définit un ensemble d'opérations

- une méthode peut être représentée par une opération qui prend un message en entrée et renvoie un message en sortie

== > chaque opération (représentée par <operation></operation>) indique les flux de messages en entrée et en sortie correspondants en définissant les élément "input" et "output".

la collection de toutes les opérations d'un service est rassemblé dans un « portType »

```
<wsdl:portType name="calculette">
    <wsdl:operation name="getsomme" parameterOrder="in0 in1">
      <wsdl:input message="impl:getsommeRequest" name="getsommeRequest"/>
      <wsdl:output message="impl:getsommeResponse" name="getsommeResponse"/>
    </wsdl:operation>
</wsdl:portType>
```

Structure d'un document WSDL

4- Le binding (l'attache): définit un lien entre le « PortType » et un service réel associé à un protocole et un format

```
<wsdl:binding name="calculetteSoapBinding" type="impl:calculette" >
    <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="getsomme« >
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getsommeRequest " >
          <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                                                  namespace="http://calculette" use="encoded"/>
      </wsdl:input>
      <wsdl:output name="getsommeResponse« >
          <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                                                  namespace="http://calculette" use="encoded"/>
      </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
```

Structure d'un document WSDL : format des messages SOAP

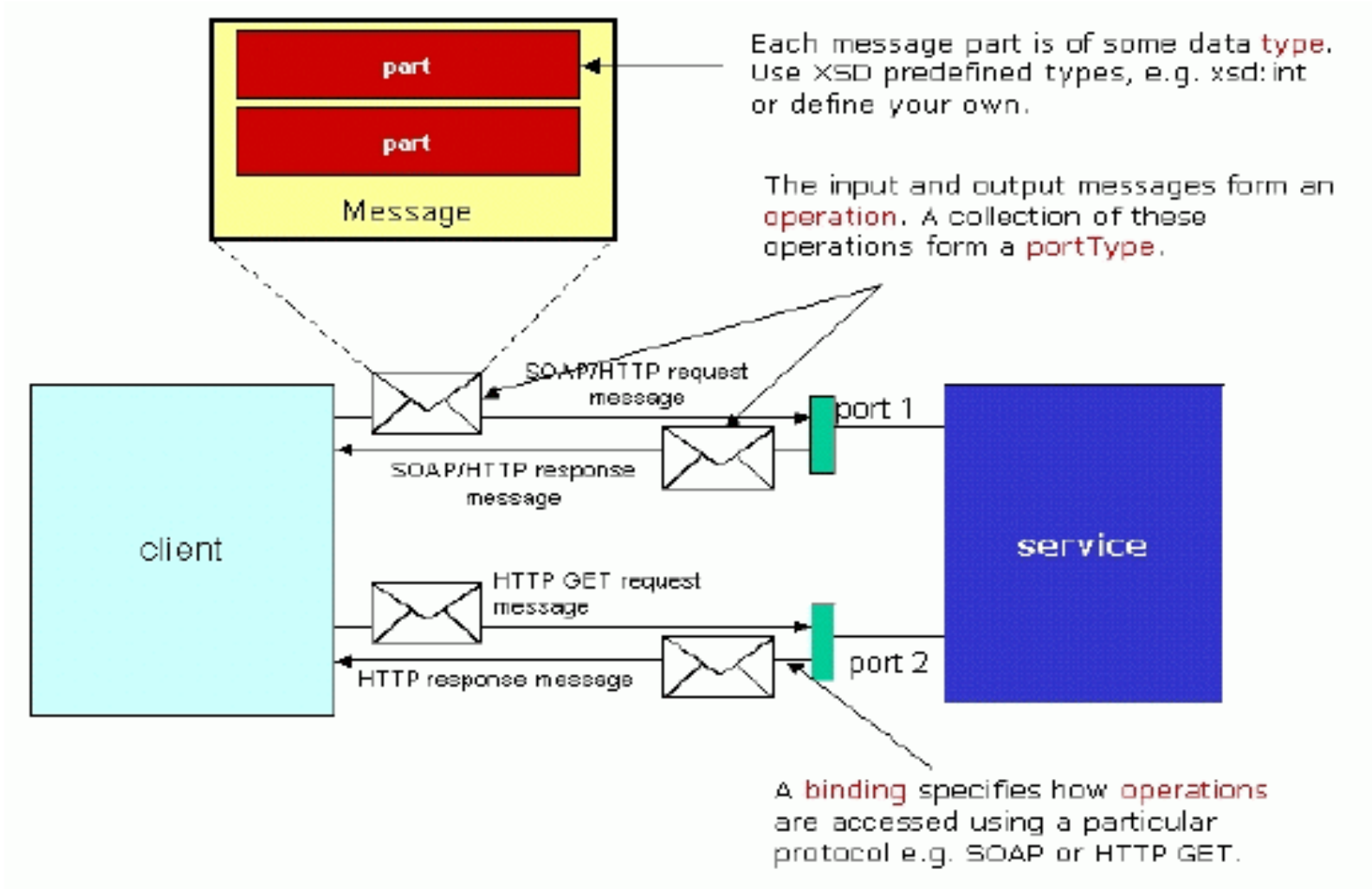
- Existence de plusieurs formats utilisables pour un message SOAP :
  - RPC Encoded :
    - le premier format historiquement proposé par SOAP
    - simple à mettre en oeuvre
    - nom recommandé par les spécifications WS-I Basic profile
  - RPC Literal :
  - Document Literal :
  - Document Literal Wrapped : recommandé
  - Document Encoded : pas supporté actuellement par les moteurs de services web

Structure d'un document WSDL

5- **Services** : c'est la mise à disposition d'une ou plusieurs méthodes.
La définition d'un service se fait par l'utilisation de <service>

Pour chaque service on va définir des ports par lesquels ce service sera disponible
Port : Un port définit la localisation concrète (endpoint) du service

```
<wsdl:service name="calculetteService">
    <wsdl:port binding="impl:calculetteSoapBinding" name="calculette">
        <wsdlsoap:address location="http://localhost:8080/axis/services/calculette"/>
    </wsdl:port>
</wsdl:service>
```



Différents éléments d'un fichier WSDL (source : LearnXmlws, http://www.learnxmlws.com/tutors/wsdl/wsdl.aspx)

Services Web

SOAP: Simple Object Access Protocol

SOAP

-SOAP (Simple Object Access Protocol) est un protocole d'invocation de méthodes sur des services distants. Il permet de coder l'appel de méthode distant dans une requête HTTP en utilisant XML. SOAP consiste en l'envoi de message à des objets distants avec du XML inclus dans des requêtes et réponse HTTP [1].

- Définit un ensemble de règles pour structurer des messages principalement pour exécuter des dialogues requête/ réponse.

SOAP

- SOAP s'appuie:
  - Sur XML pour structurer les messages
  - et sur des protocoles internet (principalement HTTP) pour la transmission des messages

  ==> SOAP permet de coder l'appel de méthode distant dans une requête HTTP en utilisant XML.

- Le client et le serveur peuvent tourner sur n'importe quelle plateforme et écrire avec n'importe quelle langage il faut juste qu'ils puissent formuler et comprendre les messages SOAP

Eléments d'un message

- Envelope
  - Élément pouvant contenir des déclarations d'espaces de noms ou des sous-éléments expliquant comment la requête doit être traitée et présentant les éléments contenus dans le message.

- Header
  - Élément optionnel fils de Envelope
  - Permet des extensions telles que authentification, session, etc.
- Body
  - Élément obligatoire fils de Envelope
  - Définit la méthode appelée, contient les paramètres
  - Peut contenir un élément Fault en cas d'erreur

- SOAP utilise les protocoles HTTP et XML. HTTP comme mécanisme d'invocation de méthodes en utilisant un des balises spécifiques pour indiquer la présence de SOAP comme <soapenv:>

Exemple d'une réponse à une requête SOAP :

```
<?xml version="1.0" encoding="UTF-8" ?>
-<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
-            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
-            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
-       <soapenv:Body>
-            <getsommeResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
                  <getsommeReturn xsi:type="xsd:int">5</getsommeReturn>
             </getsommeResponse>
       </soapenv:Body>
 </soapenv:Envelope>
```

Souvenez-vous dans le fichier wsdl on a :

```
…
<wsdl:message name="getsommeResponse« >
       <wsdl:part name="getsommeReturn" type="xsd:int"/>
  </wsdl:message>
```

Mise en œuvre avec Axis

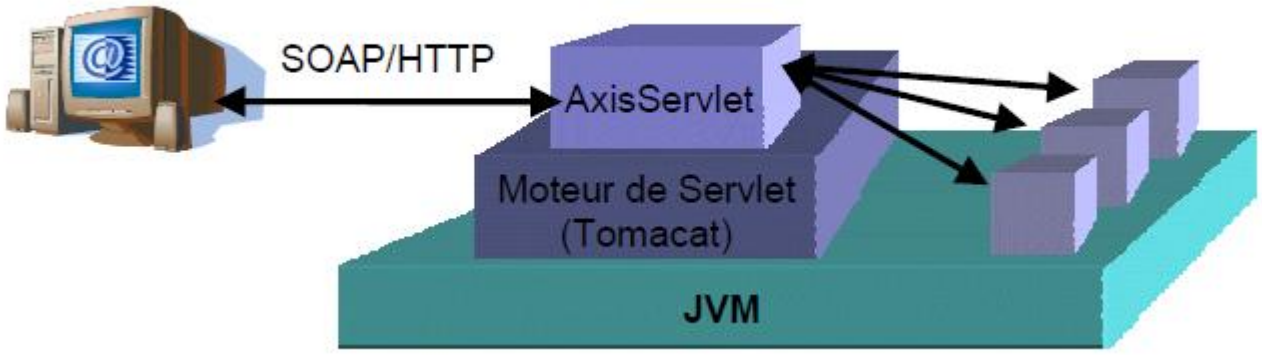
- Axis (Apache eXtensible Interaction System)
- c'est un projet open-source du groupe Apache (diffusé sous la licence Apache 2.0)
- propose une implémentation d'un moteur de service web qui implémente le protocole SOAP : il permet de créer, déployer et de consommer des services web.

- Support coté serveur
  - Servlet qui reçoit et envoie des messages SOAP HTTP
- Support coté client
  - API pour envoyer des messages SOAP sur HTTP

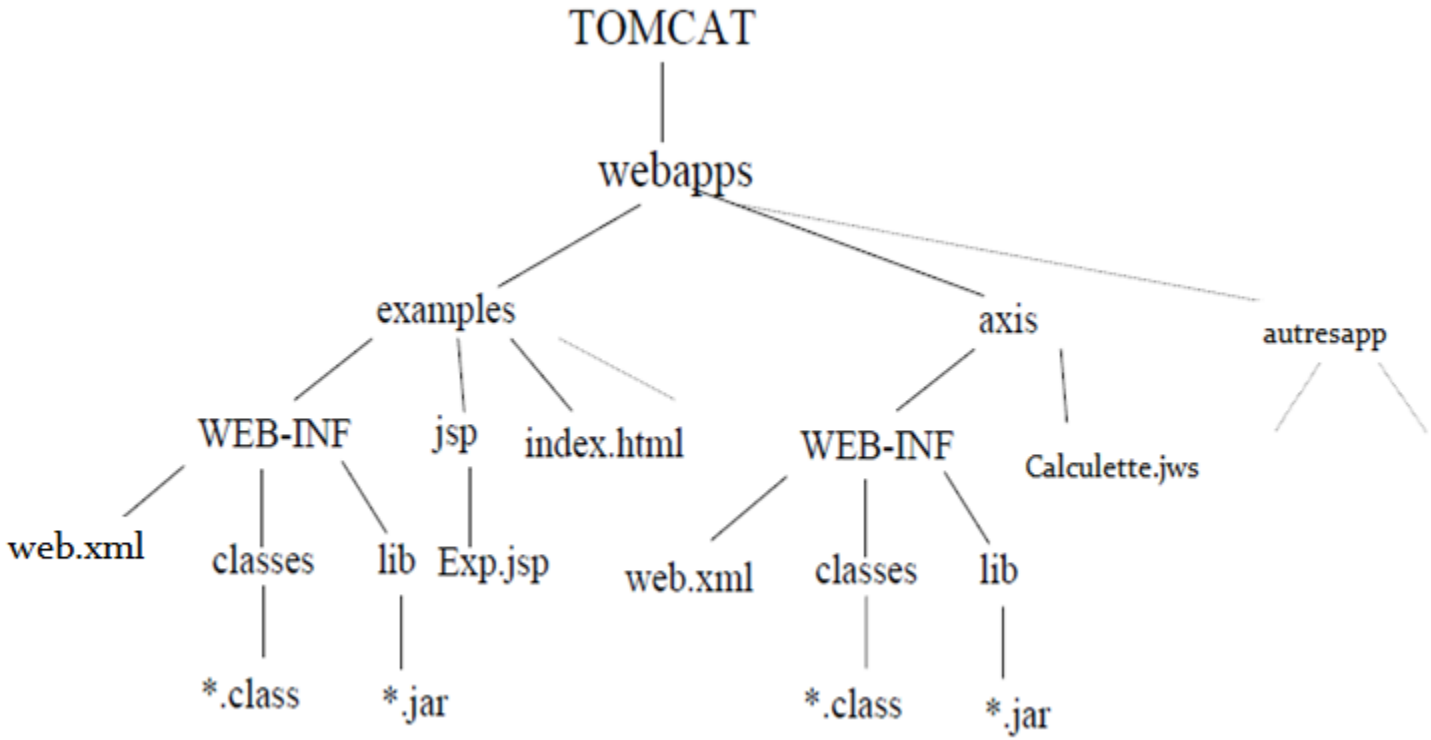
- Axis propose l'outil WSDL2Java pour la génération des classes stubs à partir du WSDL afin de mettre en place le client consommateur du service

Architecture Serveur

- Le client envoie des messages SOAP/HTTP
- La Servlet AxisServlet reçoit et renvoie les messages SOAP et les transmet aux objets Java correspondant
- Les Objets Java effectuent les services : sont des objets Java classiques.



Tomcat file system



== > C'est le déploiement instantanée par fichier JWS : méthode simple (voir TP1)

1- Développer une classe Java

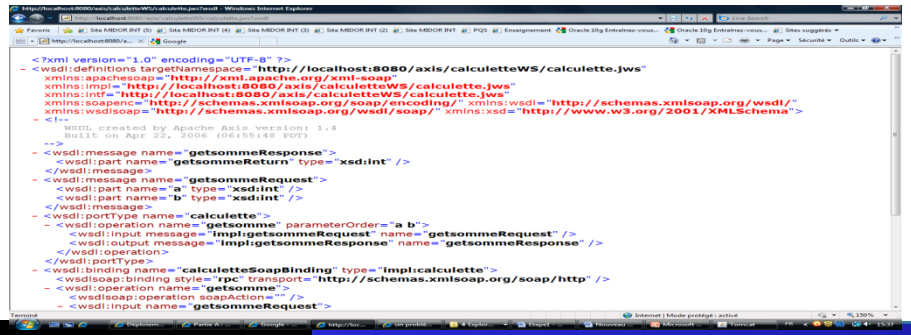
```
public class Calculette{
   public int getsomme(inti1, inti2) {
        return i1+i2;
    }
 }
```

2- Sauvegarder la classe dans Calculette.jws

3- Copier Calculette.jws dans le répertoire axis (vaut mieux dans un sous répertoire axis « calculetteWS » par exemple qu'on crées

4-Lancer le serveur tomcat

5-Obtention de la description WSDL : http://hoste:8080/axis/ calculetteWS/Calculette.jws?WSDL



6- invocation des méthodes :

=> visualisation résultats sous format message SOAP :

http://localhost:8080/axis/calculetteWS/calculette.jws?method=getsomme&a=10&b=13

Voir TP (§ II- Méthode de déploiement 2)

1- écriture de la classe
2- compilation de la classe est dépôt dans le répertoire WEB-INF/classe de axis
3- écriture de descripteur de déploiement (le fichier WSDD)
4- ne pas oublier de lancer le serveur tomcat
5- déploiement : java org.apache.axis.client.AdminClient <nomfichier>.wsdd
6- teste de l'accès au service :
http://localhost:8080/axis/services/<nom service>
7- pour visualiser WSDL : http://localhost:8080/axis/services/<nom service>?wsdl

Ps : axis offre la possibilité de générer le wsdl à partir d'une classe :

```
java org.apache.axis.wsdl.Java2WSDL -o calcul.wsdl   -l "http://localhost:8080/axis/services/calculSweb"   calcul.calcul
```

Le développement d'un client

Méthode 1 : Développement d'un client à partir du WSDL

- Génération d'un ensemble de classes (classes proxy) facilitant l'envoi de messages SOAP :

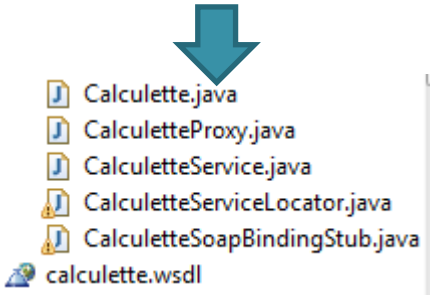
➔ java org.apache.axis.wsdl.WSDL2Java calculette.wsdl ← Nom du fichier wsdl

Ou en utilisant l'url du service :

java org.apache.axis.wsdl.WSDL2Java "http://localhost:8080/axis/services/<nom service>?wsdl"

Exemple : java org.apache.axis.wsdl.WSDL2Java http://localhost:8080/axis/calculetteWS/calculette.jws?WSDL

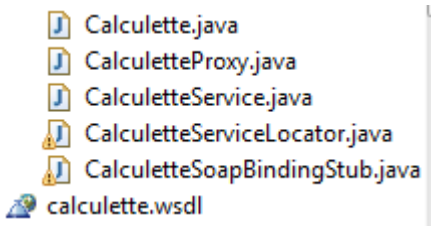
- Classes générées:



- Construction du client :
  - Instancier un Service
  - Obtenir un Port à partir du Service
  - Utiliser les méthodes du Port

Exemple :

```
Public class Client {
   public static void main(String[] args) {
    try{
                    // Instancier un Service
                  CalculetteService service = new CalculetteServiceLocator();

                  // Obtenir un Port à partir du Service
                     Calculette port = service.getcalculette();

                 // Utiliser les méthodes du Port
                   int result = port.getsommer(2,3);
                 System.out.println("La somme est: "+result);

         } catch(Exception ex) {
                 ex.printStackTrace();
        }
    }
}
```



1- création de l'accès vers le service
2- création d'un appel vers le service
3- récupération de l'adresse URL du service
4- récupération de la méthode à appliquer
5- Mise en place des éventuels valeurs des paramètres de la méthode
6- invocation du service
7- récupération du résultat

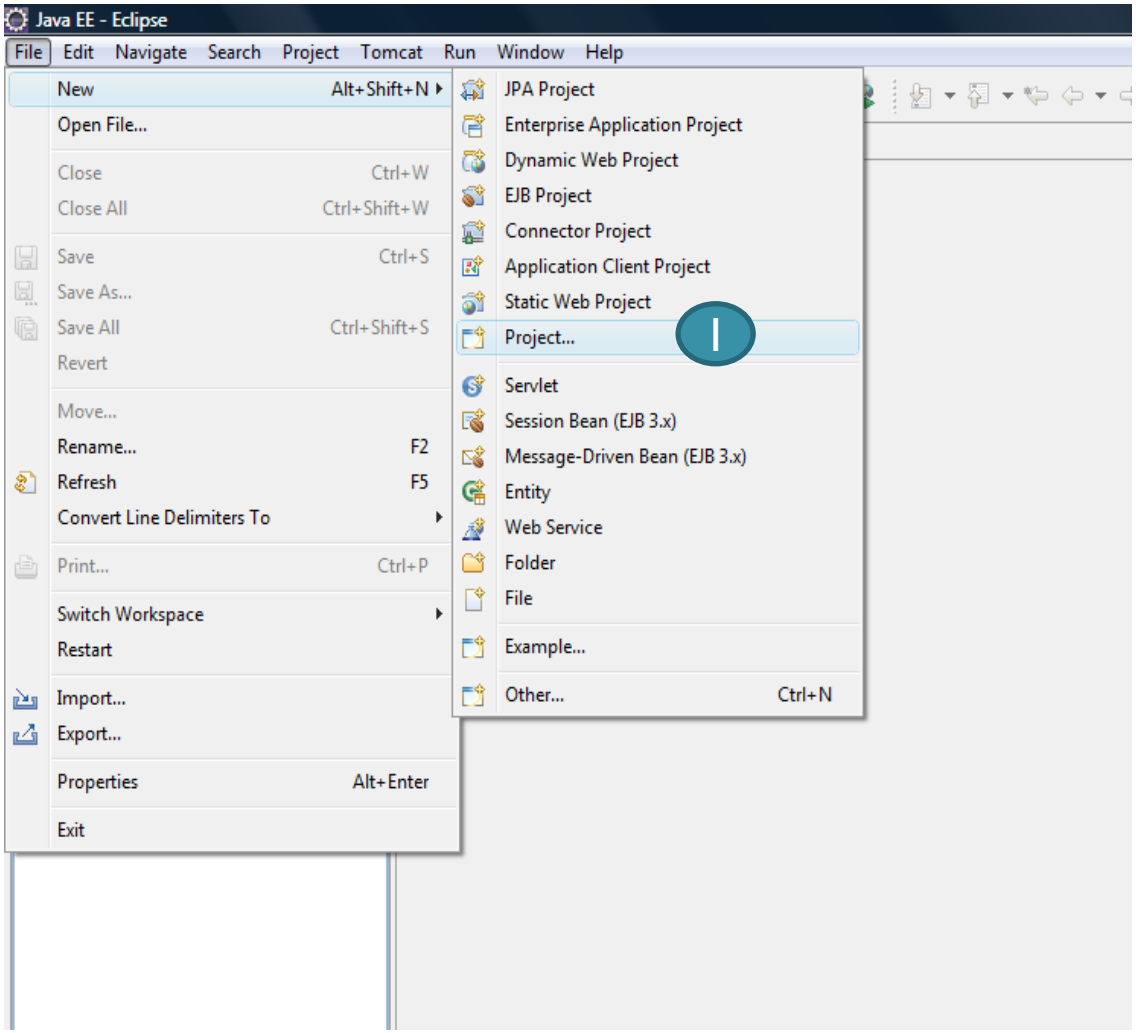
Exemple de client générique :

```
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import org.apache.axis.encoding.XMLType;
import org.apache.axis.utils.Options;
import javax.xml.rpc.ParameterMode;

public class client {
        public static void main(String [] args) throws Exception {

                Integer a = new Integer(Integer.parseInt(args[0]));
                Integer b = new Integer(Integer.parseInt(args[1]));
                //l'adresse URL du service
               String endpoint ="http://localhost:8080/axis/calculetteWS/calculette.jws";
              //  String endpoint = "http://localhost:8080/WStest/services/calculette";  //ex. d'url  pour SW mis en place sous eclipse
                                                                                          //déploiement sous tomcat (par
            exemple)
              // String endpoint ="http://localhost:8080/axis/services/calculette";  // déploiement sous axis

                //création de l'accès vers le service
                Service  service = new Service();
                //création d'un appel vers le service
                   Call     call    = (Call) service.createCall();
                   //récupération de l'adresse URL du service
                   call.setTargetEndpointAddress(new java.net.URL(endpoint));
                   //récupération de la méthode à appliquer
                   call.setOperationName( "getsomme" );
                   //Mise en place des  valeurs des paramètres de la méthode
                   call.addParameter("op1", XMLType.XSD_INT, ParameterMode.IN);
                   call.addParameter("op2", XMLType.XSD_INT, ParameterMode.IN);
                   call.setReturnType(XMLType.XSD_INT);
                   //invocation du service et récupération du résultat
                   Integer res = (Integer) call.invoke( new Object [] { a, b });

                   System.out.println("resultat : "+ res);
        }
}
```

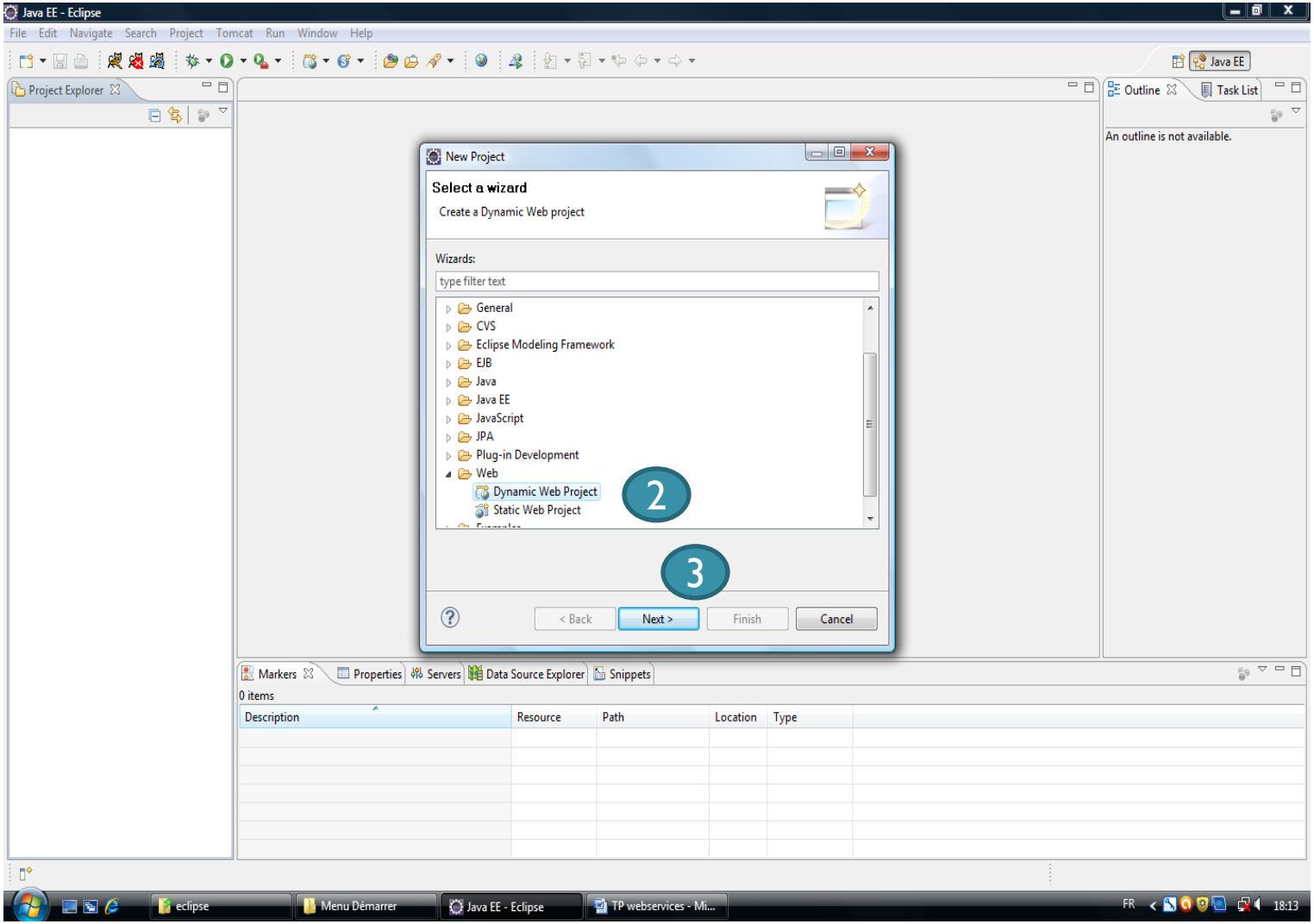
Application sur eclipse

Mise en place d'un service web avec eclipse

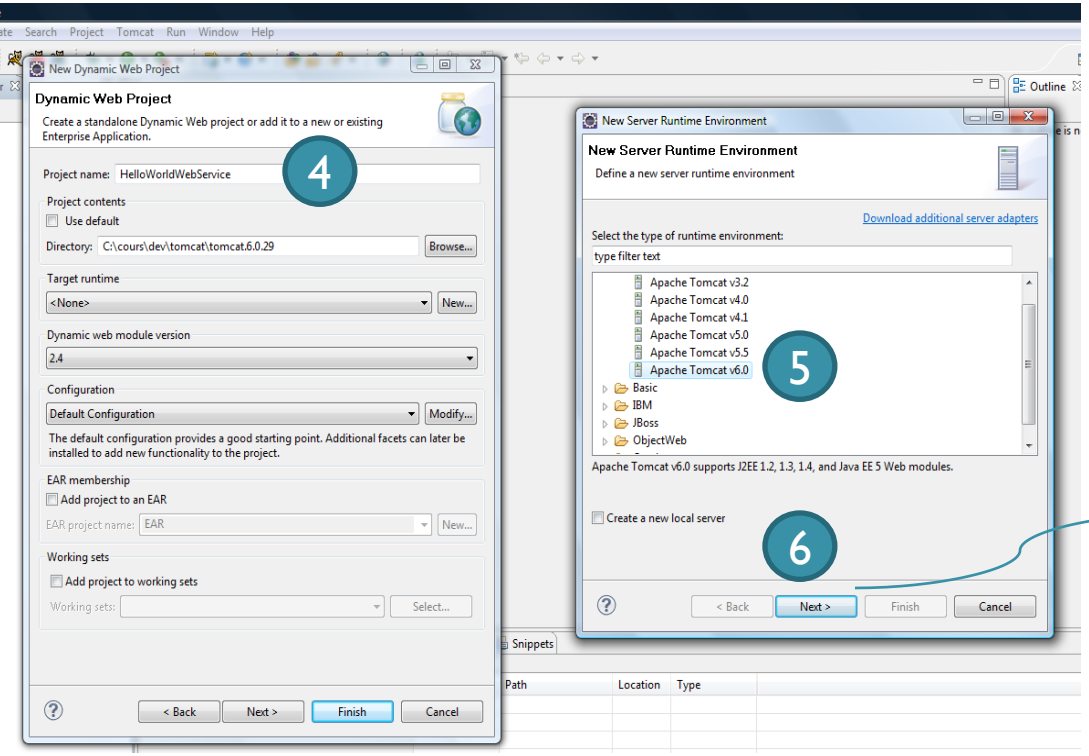
I- ajouter un nouveau projet

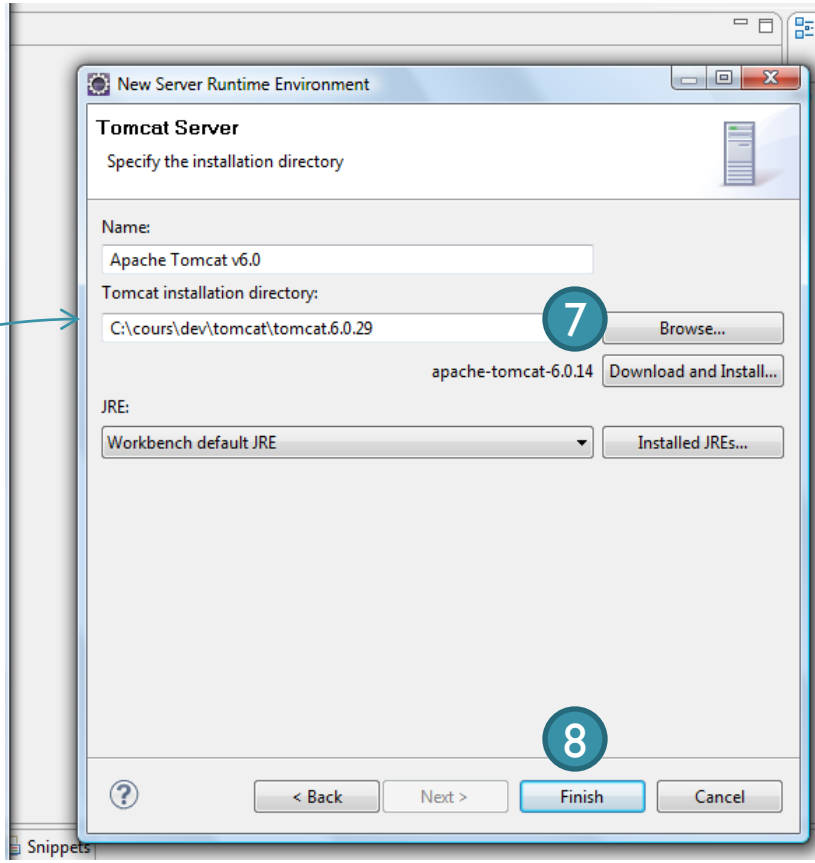


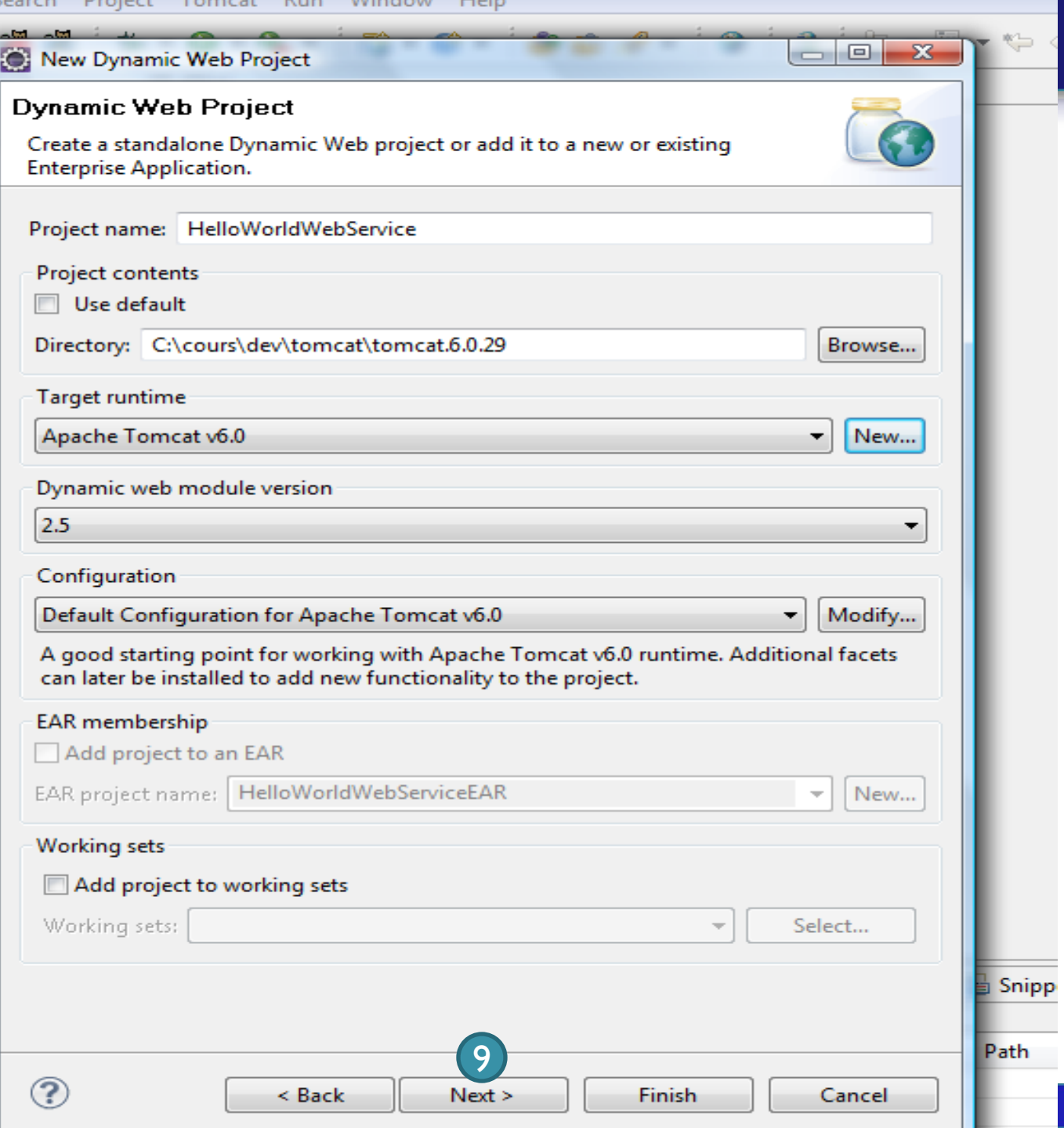
2) Selectionner "Dynamic Web Project" et cliquer sur next

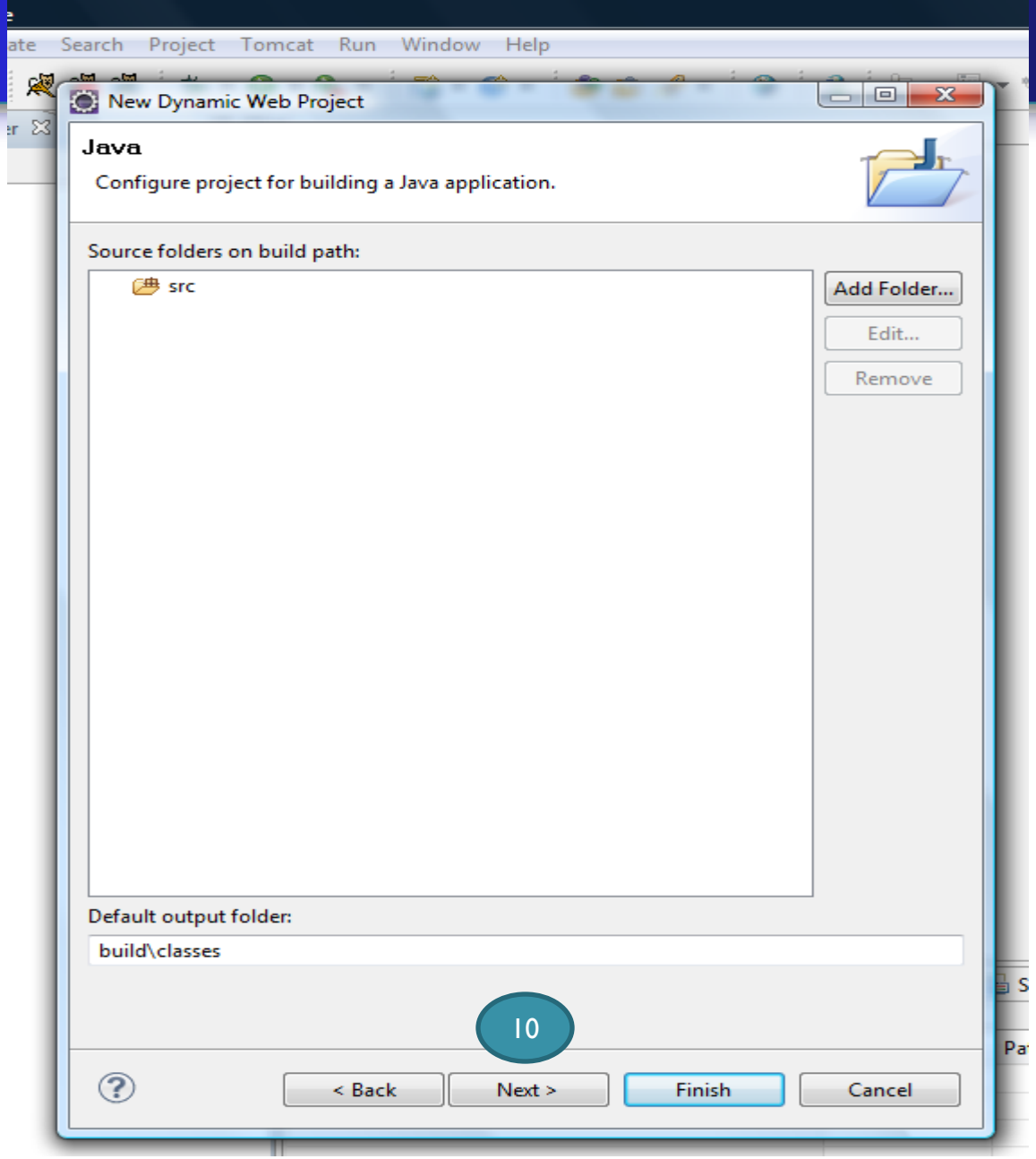


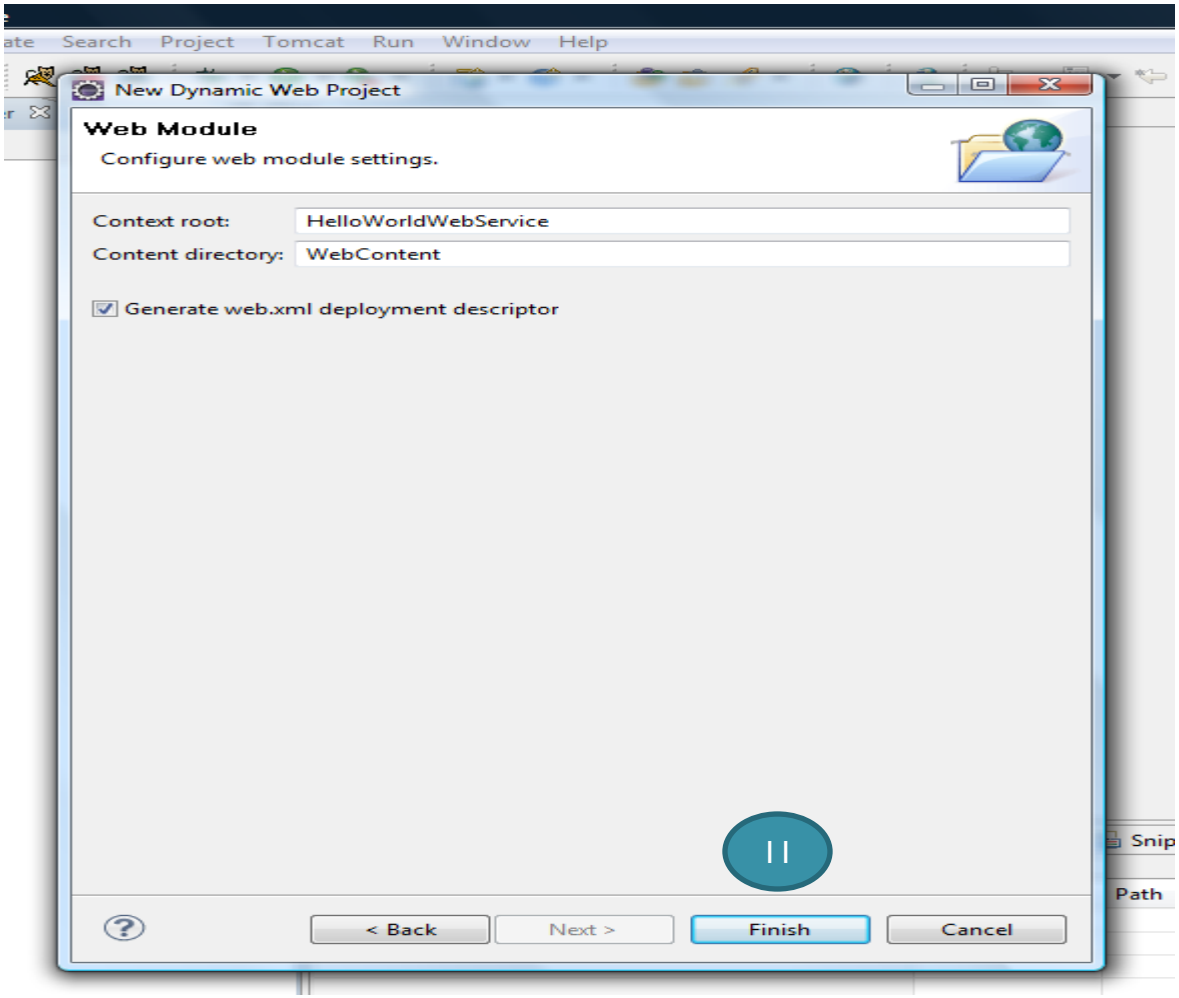
3) Saisir le nom du projet (4) et sélectionner le serveur web (5 et 6)



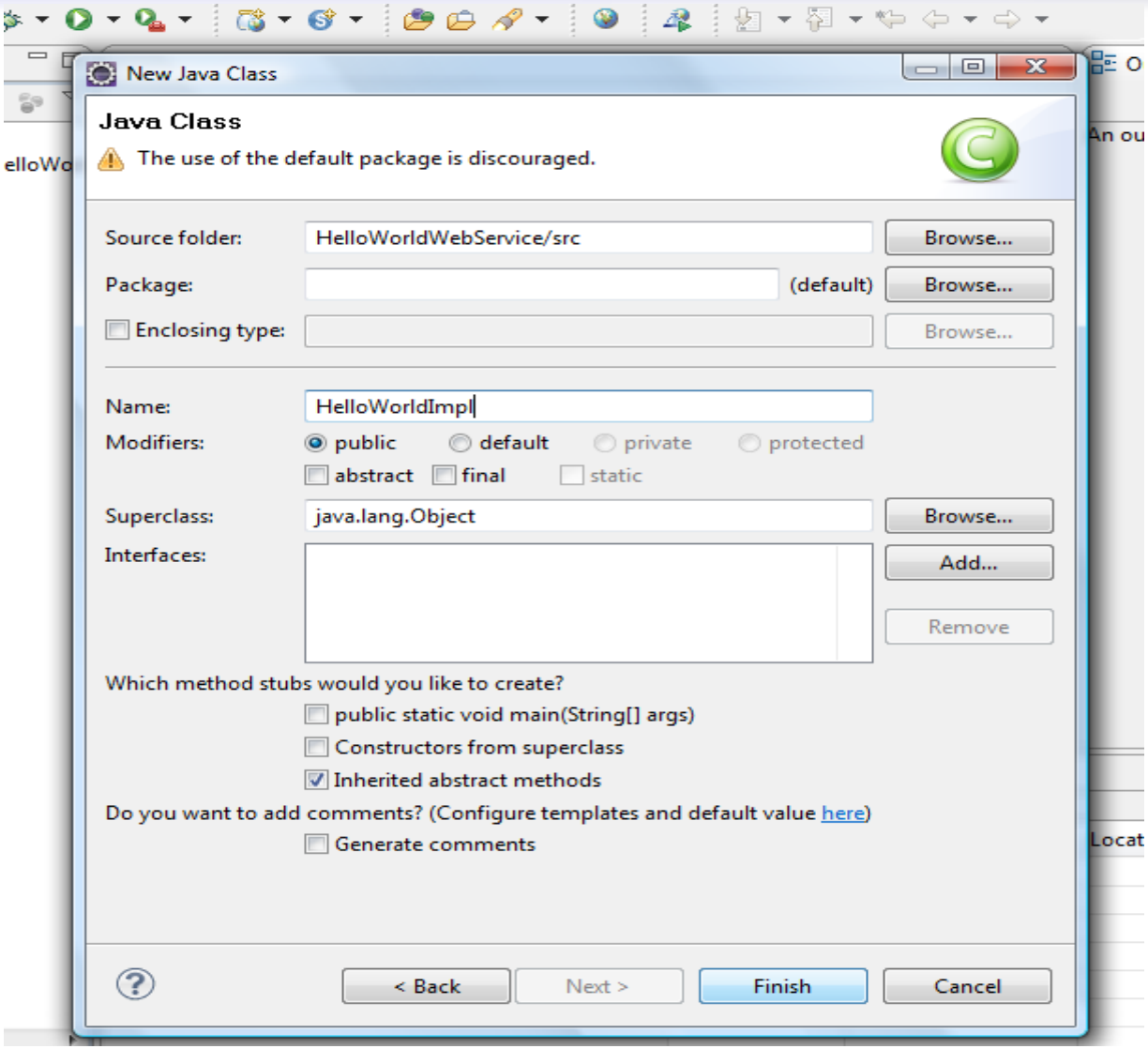


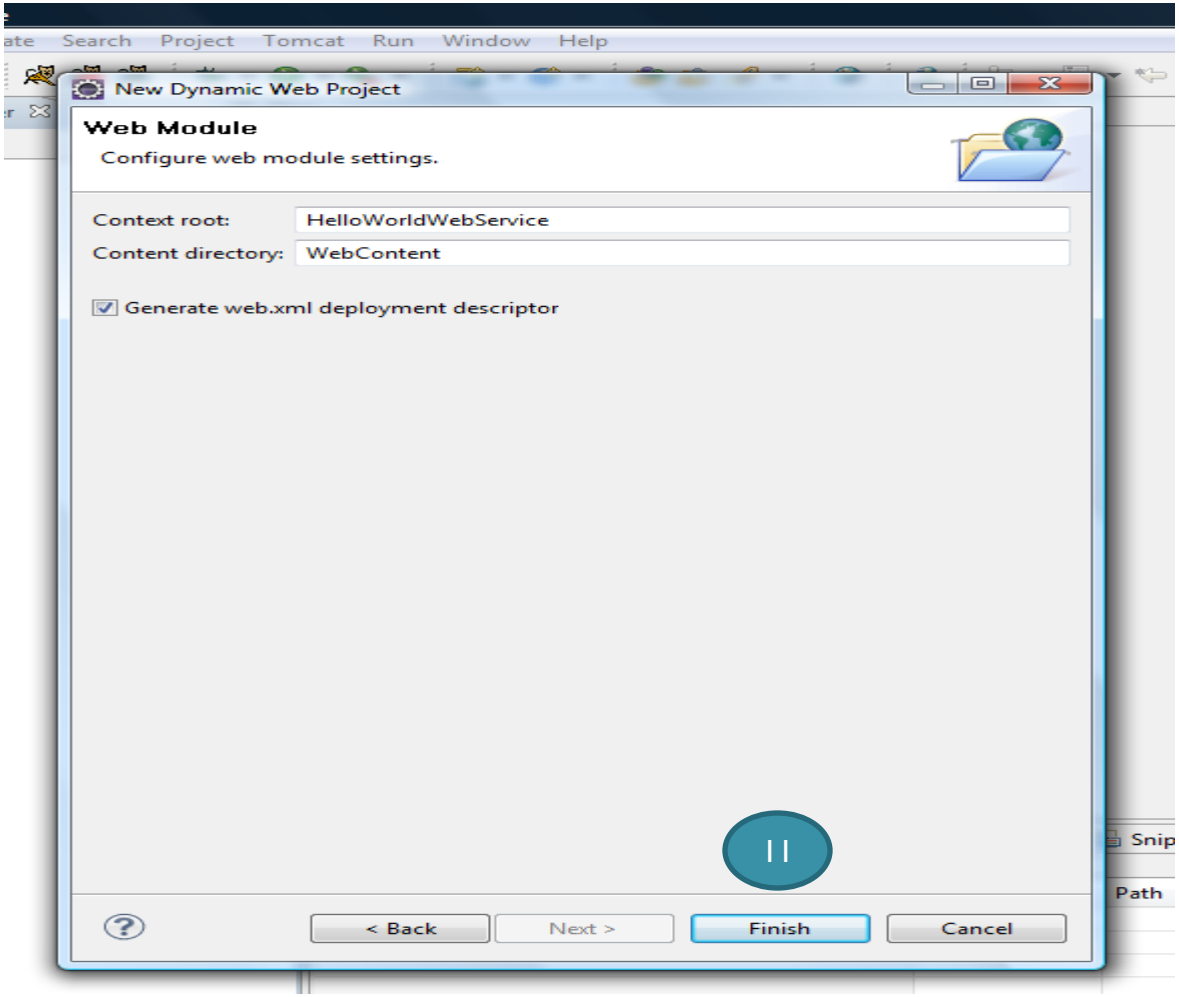

New Dynamic Web Project
Dynamic Web Project
Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.
Project name: HelloWorldWebService
Project contents
Use default
Directory: C:\cours\dev\tomcat\tomcat.6.0.29
Browse...
Target runtime
Apache Tomcat v6.0
New...
Dynamic web module version
2.5
Configuration
Default Configuration for Apache Tomcat v6.0
Modify...
A good starting point for working with Apache Tomcat v6.0 runtime. Additional facets can later be installed to add new functionality to the project.
EAR membership
Add project to an EAR
EAR project name: HelloWorldWebServiceEAR
New...
Working sets
Add project to working sets
Working sets:
Select...
9
< Back
Next >
Finish
Cancel


ate Search Project Tomcat Run Window Help
New Dynamic Web Project
Java
Configure project for building a Java application.
Source folders on build path:
src
Add Folder...
Edit...
Remove
Default output folder:
build\classes
10
< Back
Next >
Finish
Cancel


Search
Project
Tomcat
Run
Window
Help
New Dynamic Web Project
Web Module
Configure web module settings.
Context root:
HelloWorldWebService
Content directory:
WebContent
Generate web.xml deployment descriptor
< Back
Next >
Finish
Cancel
Path

- Créer une simple class



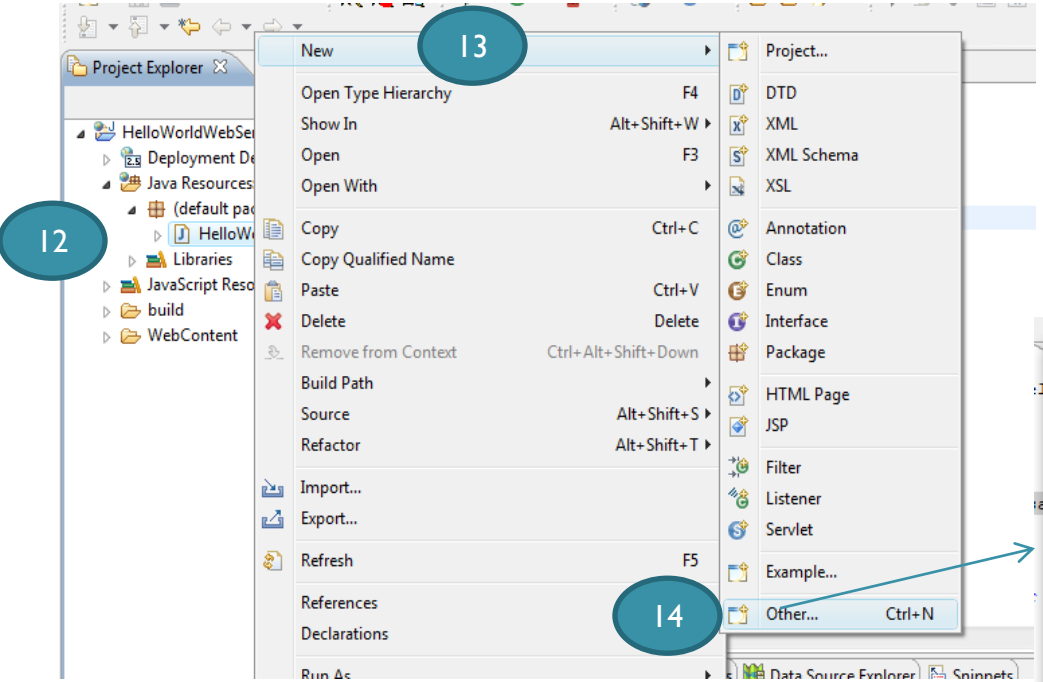

Search
Project
Tomcat
Run
Window
Help
New Dynamic Web Project
Web Module
Configure web module settings.
Context root:
HelloWorldWebService
Content directory:
WebContent
Generate web.xml deployment descriptor
< Back
Next >
Finish
Cancel
Path

- Saisir le contenu de la class

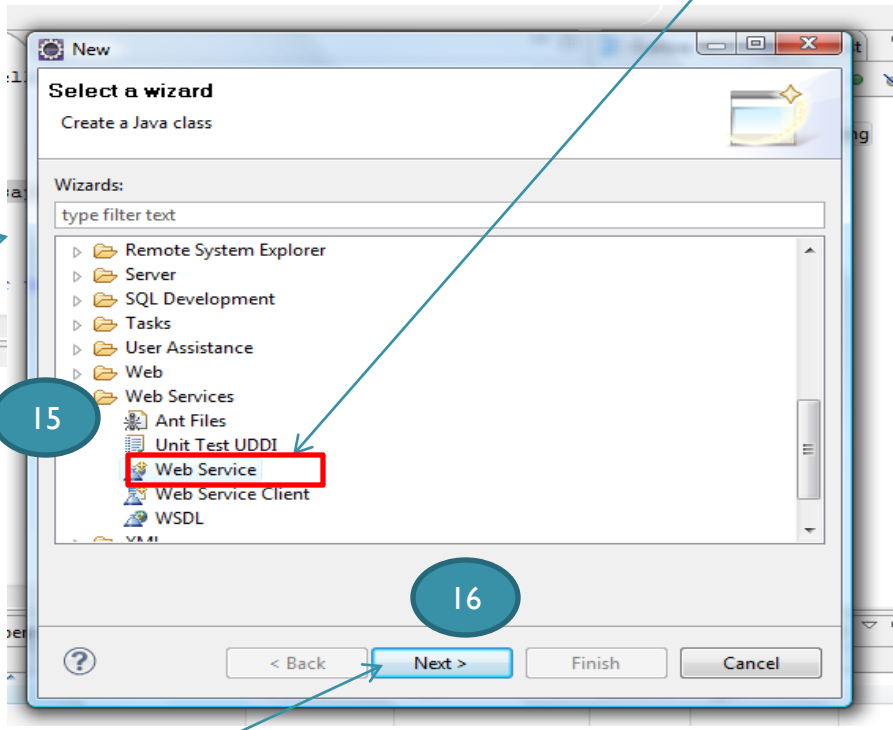
12

```
public class HelloImpl {
        public String sayHello(){

                return "Hello World ";

        }

}
```

- Clique droit sur la classe >> New >> Other

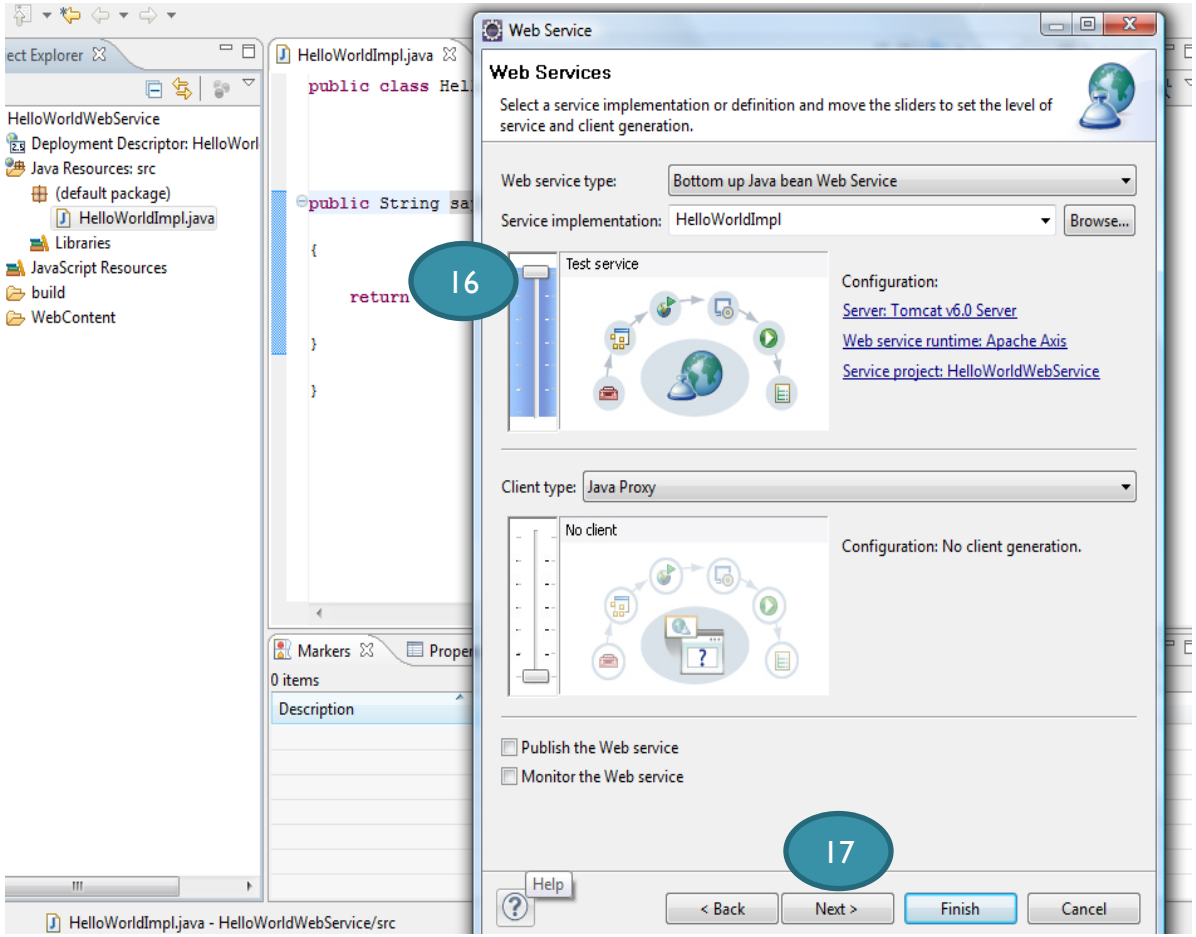


Puis sélectionner Web Services >> Web Service

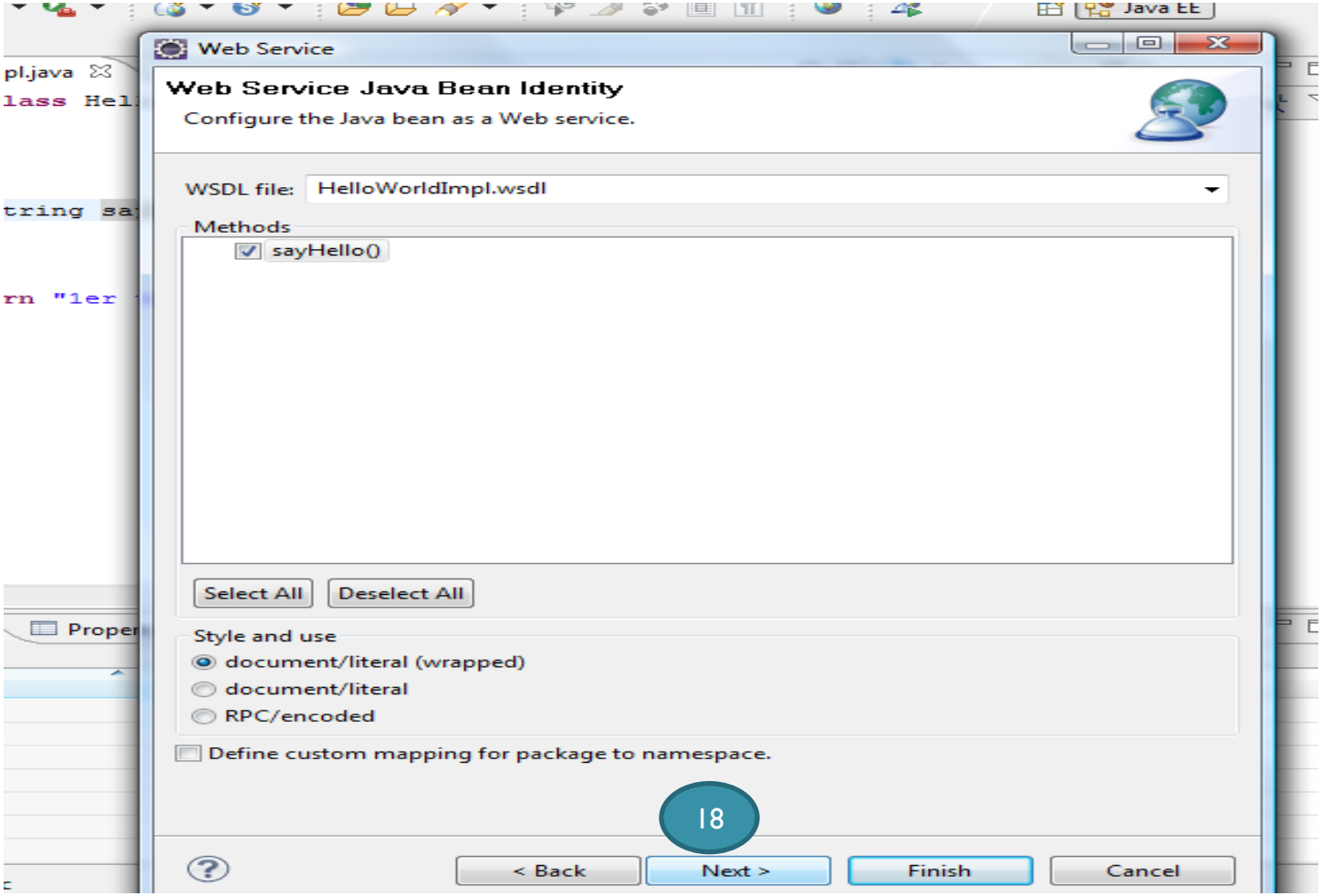


- A la fin cliquer sur next (16)

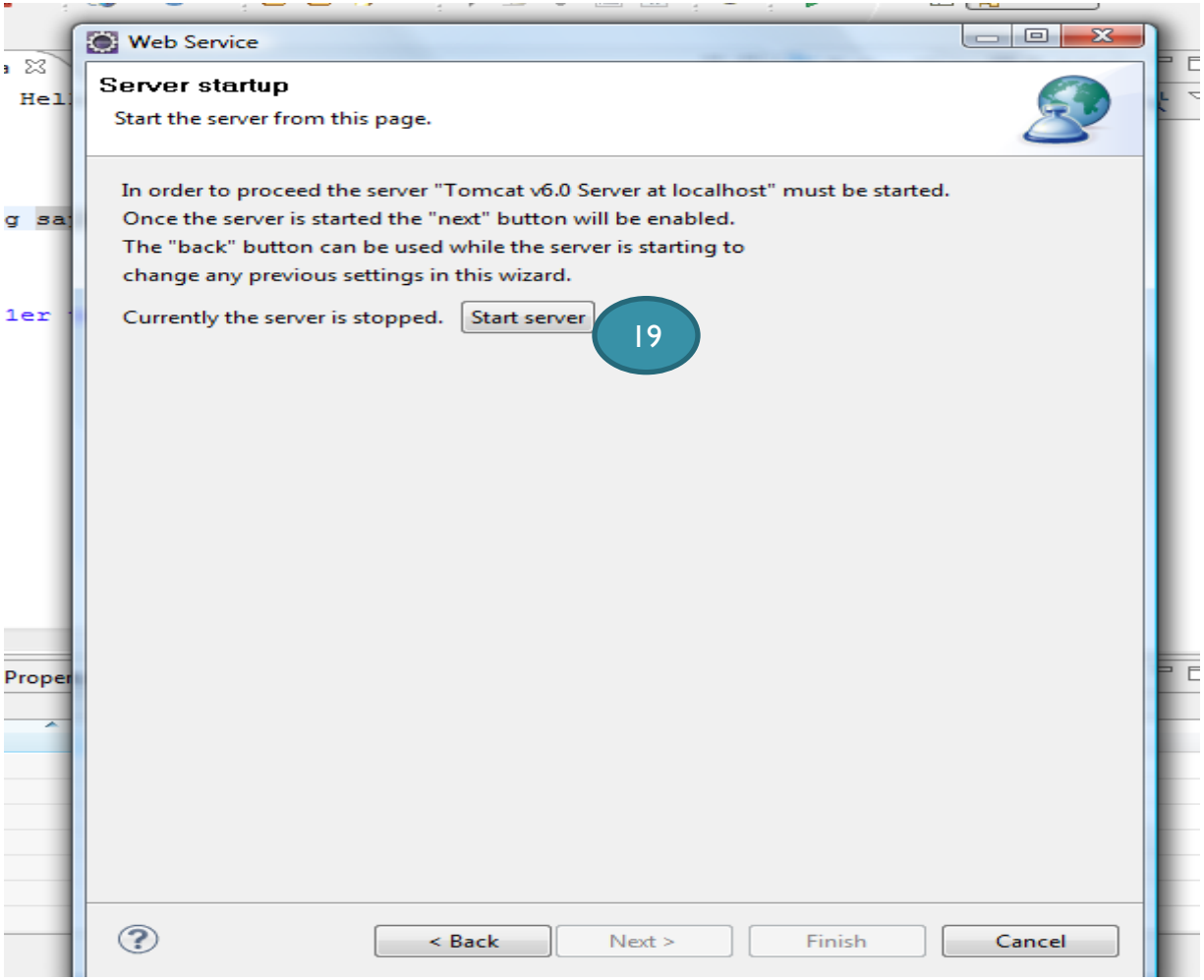
- On aura la copie d'écran ci-desous. En suite, on mit le curseur (16) au niveau du « test service » puis cliquer sur Next

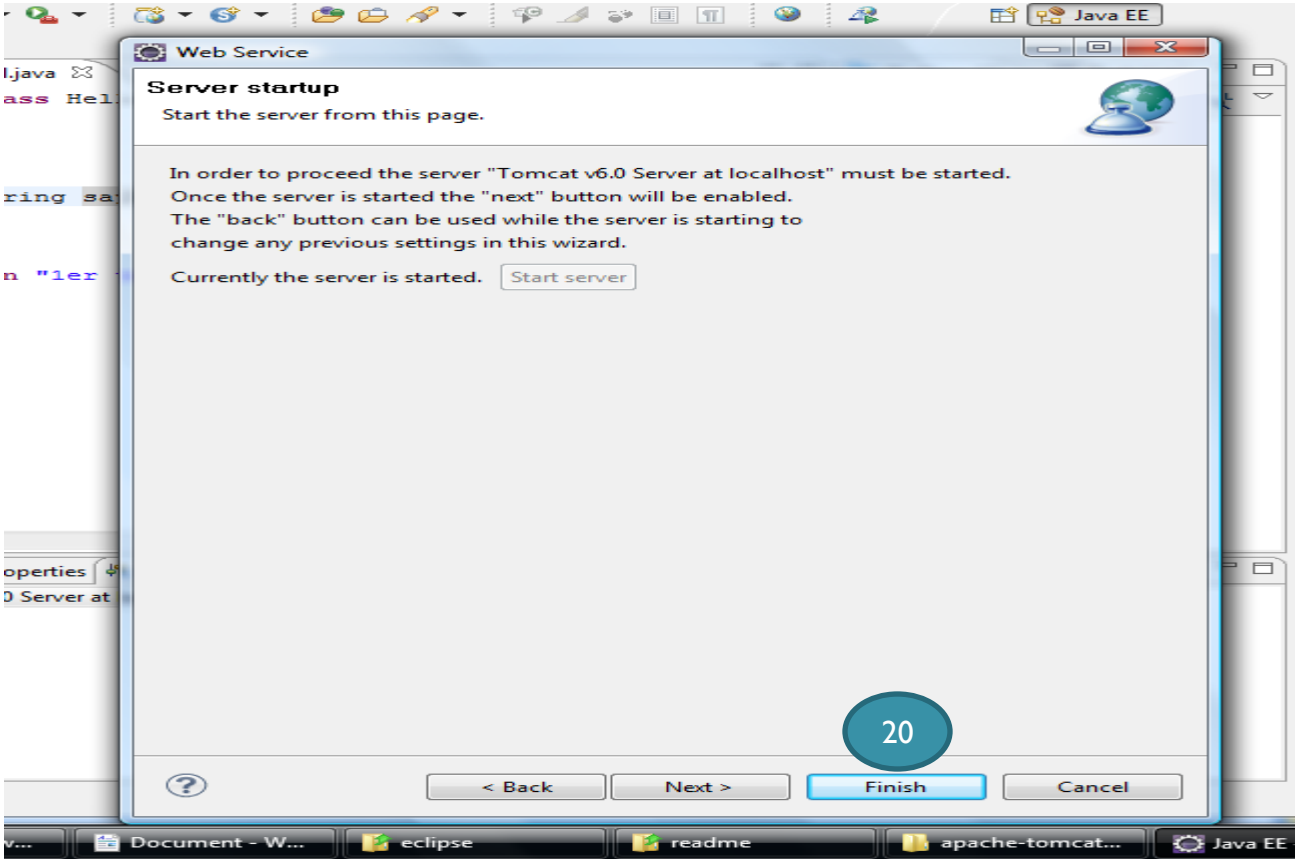


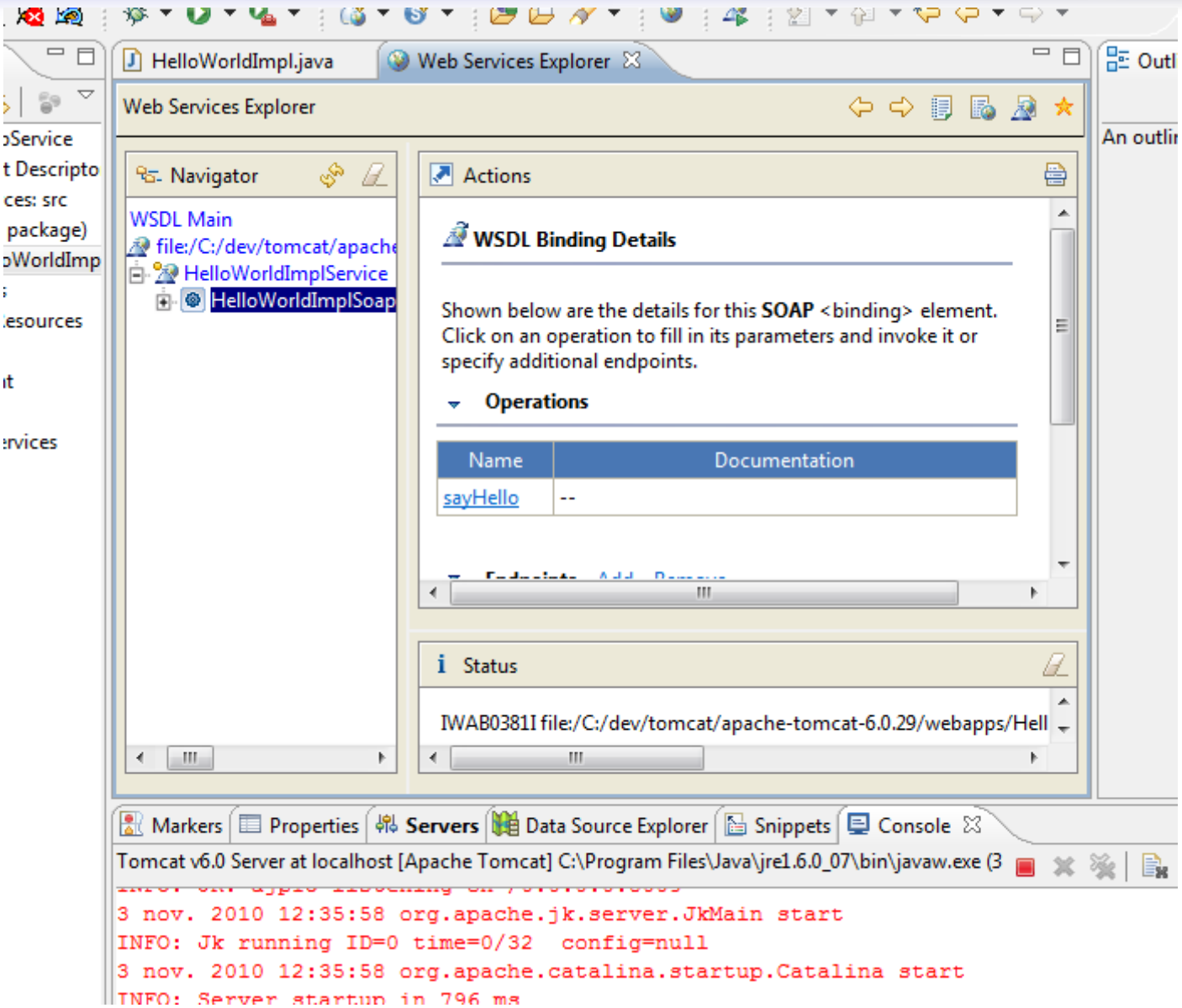
- Cliquez sur le bouton « next » (17) pour afficher la page « Identité du JavaBean du service web » de l'assistant



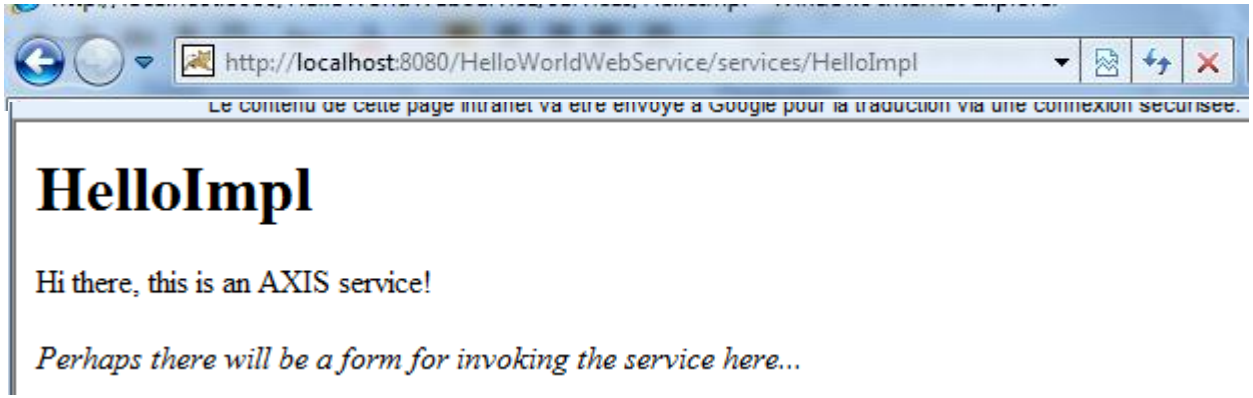
- Cliquez sur le bouton « start server » pour démarrer le service dans le serveur.




Java EE
Web Service
Server startup
Start the server from this page.
In order to proceed the server "Tomcat v6.0 Server at localhost" must be started.
Once the server is started the "next" button will be enabled.
The "back" button can be used while the server is starting to
change any previous settings in this wizard.
Currently the server is started.
Start server
20
< Back
Next >
Finish
Cancel
Document - W...
eclipse
readme
apache-tomcat...
Java EE

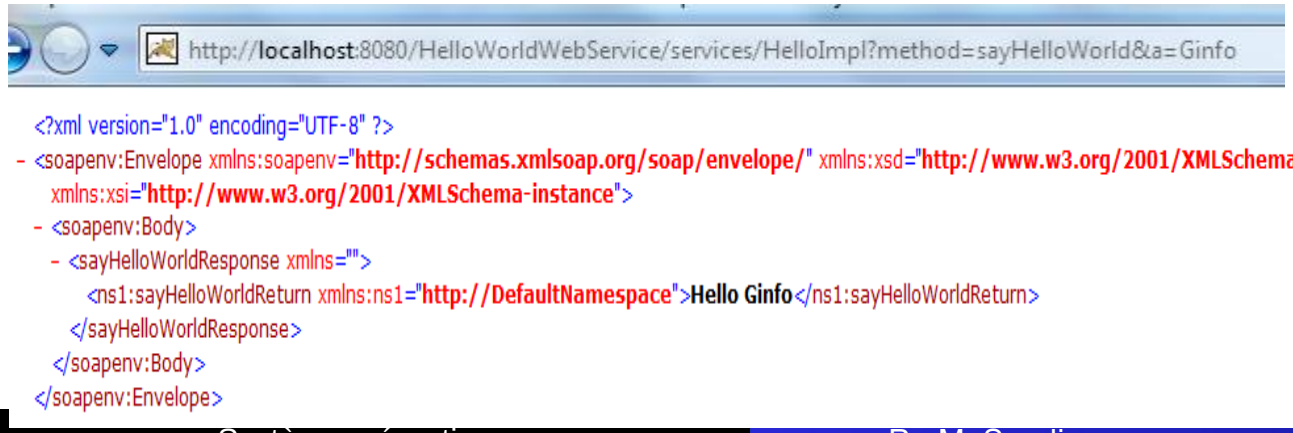

HelloWorldImpl.java
Web Services Explorer
Web Services Explorer
Navigator
WSDL Main
file:/C:/dev/tomcat/apache
HelloWorldImplService
HelloWorldImplSoap
Actions
WSDL Binding Details
Shown below are the details for this SOAP <binding> element.
Click on an operation to fill in its parameters and invoke it or
specify additional endpoints.
Operations
Name
Documentation
sayHello
--
Status
IWAB0381I file:/C:/dev/tomcat/apache-tomcat-6.0.29/webapps/Hell
Markers
Properties
Servers
Data Source Explorer
Snippets
Console
Tomcat v6.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre1.6.0_07\bin\javaw.exe (3
3 nov. 2010 12:35:58 org.apache.jk.server.JkMain start
INFO: Jk running ID=0 time=0/32 config=null
3 nov. 2010 12:35:58 org.apache.catalina.startup.Catalina start
INFO: Server startup in 796 ms

Le service est désormais accessible à partir de :
http://localhost:8080/HelloWorldWebService/services/HelloImpl



Paramètre d'entrée

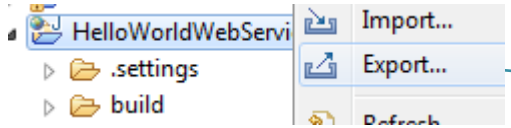
- Testez le web service avec cet url:
http://localhost:8080/HelloWorldWebService/services/HelloImpl?method=sayHelloWorld&a=Ginfo

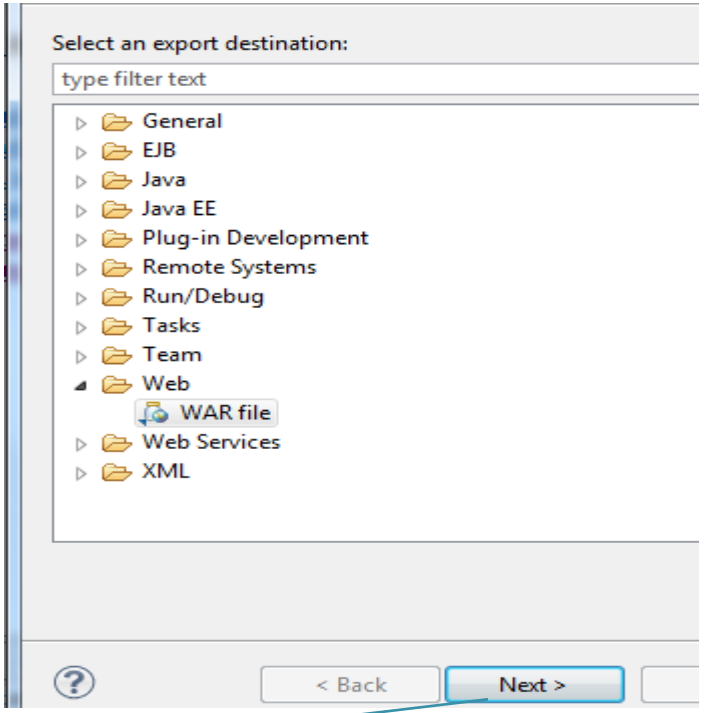


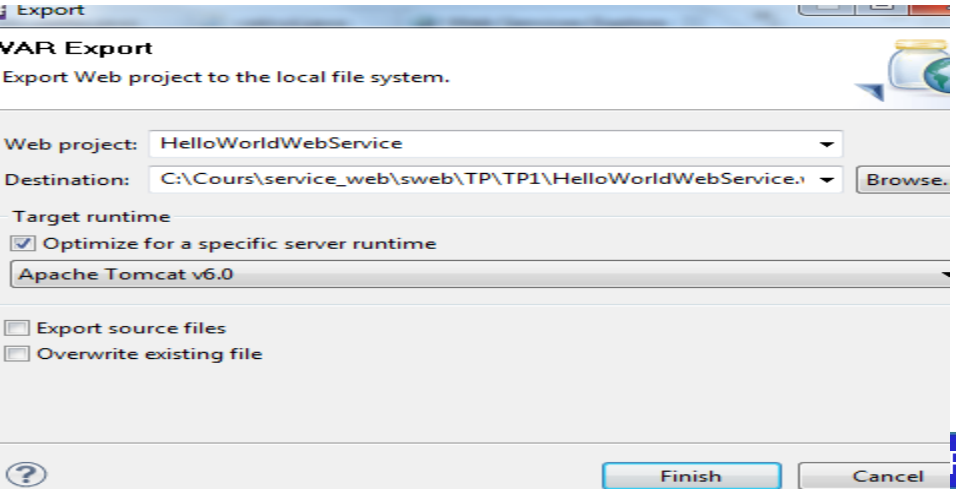
Méthode invoquée

Le service a été lancé dans le serveur Tomcat lancé par Eclipse. Cette étape passée, l'objectif est maintenant d'exporter le service afin qu'il puisse être hébergé au sein d'un serveur Tomcat indépendant :

1- Cliquer du bouton droit sur le projet pour exporter le service dans une **archive war.**

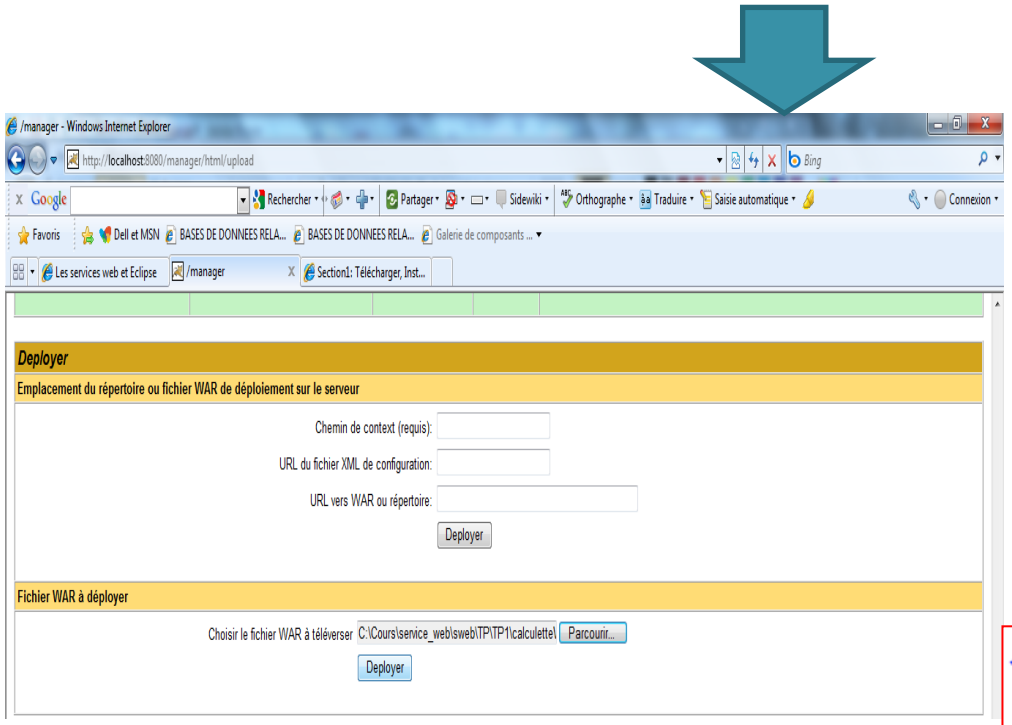






génération de HelloWorldWebService.war
Ce war généré peut être maintenant déployer dans tomcat

-Déployer le .war généré dans tomcat ➔ on utilise tomcat manager



Rappel :

NB : il est nécessaire de configurer les droits du serveur Tomcat pour accéder à la page d'administration ➔*fichier* ***tomcatusers.xml du répertoire conf de Tomcat)***

➔ ***ajouter :***

```
<tomcat-users>
  <role rolename="manager"/>
  <user username="tomcat" password="tomcat" roles="manager"/>
</tomcat-users>
```

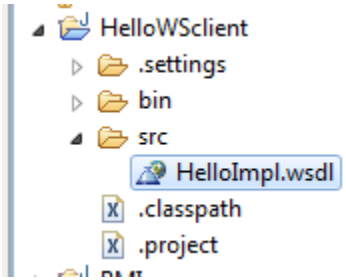
PS : normalement sous tomcat un arrêt/relance de tomcat suffit pour déployer le .war

- test : Lancez cet url : http://localhost:8080/HelloWorldWebService/services
- invoquez une méthode

1- Créer un projet java

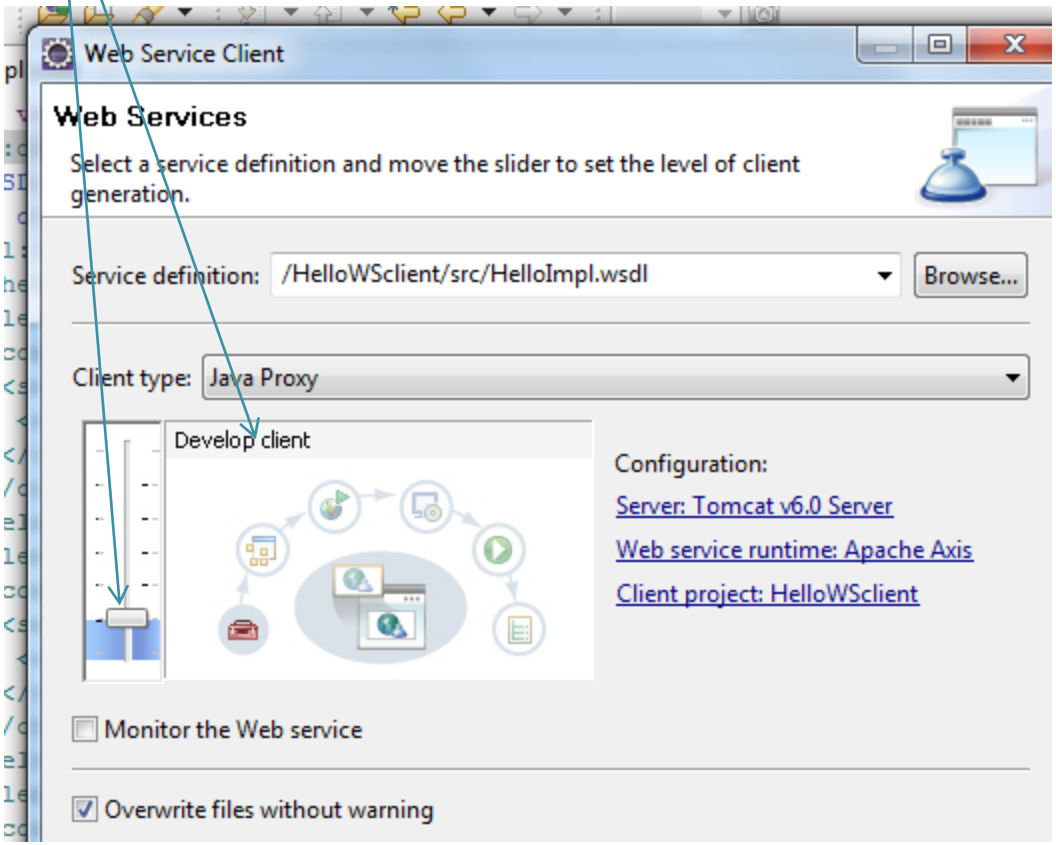
Première méthode :

2- dans le répertoire « src » copier le fichier wsdl généré lors de la mise en place du webservice



3- cliquez droit sur le fichier wsdl >> New >> Other >> Web Service Client >> « Next » >>

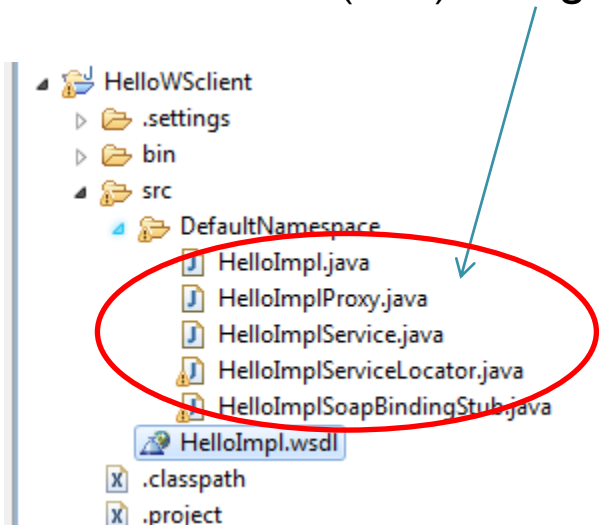
5- choisir « Develop client »



6- cliquez sur « Next » jusqu'à la fin puis « finish »

- Un répertoire portant le nom spécifié pour le targetNamespac dans le fichier wsdl
Dans notre cas on avait targetNamespace="*http://DefaultNamespace*"

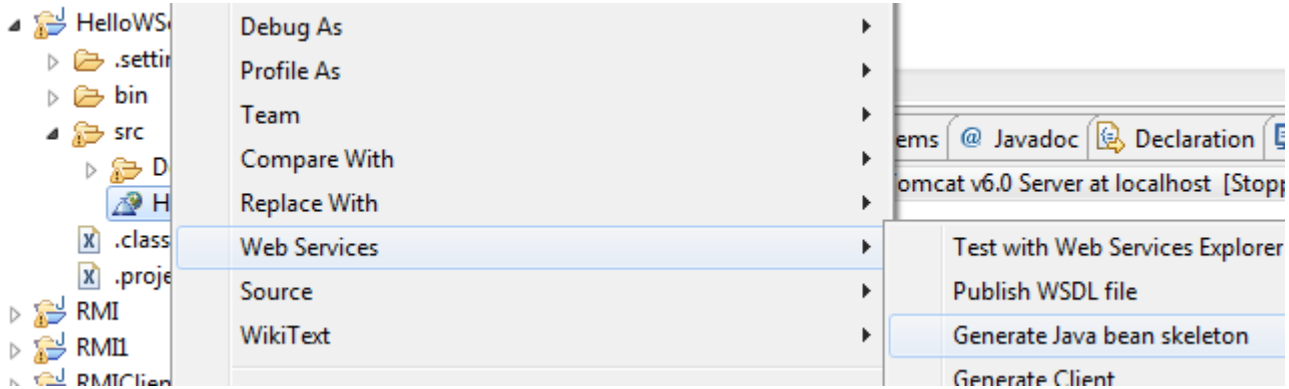
- Dans le répertoire créé, un ensemble de classes (stub) sont généré.



7- créer un package client est écrire une classe qui consomme le service (voir TP et cours . adapter la classe en fct des classe générés automatiquement)

Deuxième méthode :

clic droit sur le fichier wsdl >> Web Services >> Generate Java Been skeleton



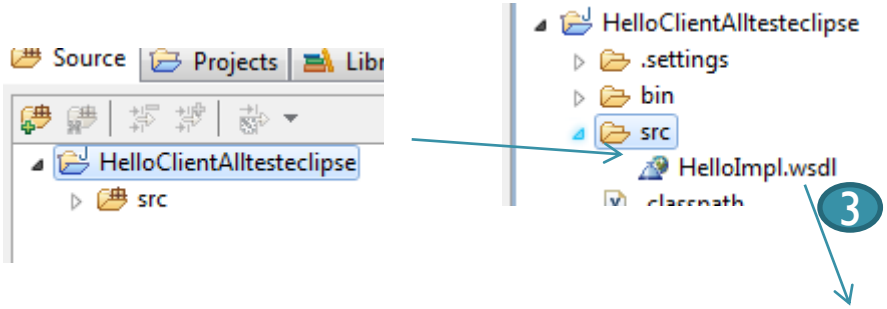
- Reste à écrire le programme client : étape 7 du transparent précédent

Création du client test du service automatiquement

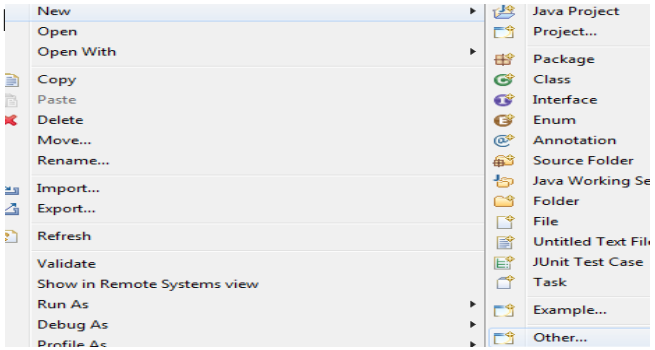
- remarque : on peut générer automatiquement un client qui test le web service via eclipse

1- créer un projet java

2- dans src copier le .wsdl gnéré lors de la mise en place du web service



3- clic droit sur le.wsdl >> New >> Othe



4- sélectionner web service client puis cliquer sur « Next »

5- choisir le niveau « test client »