



Web Services SOAP et RESTful

Mohamed Youssfi : med@youssfi.net
ENSET, Université Hassan II Mohammedia

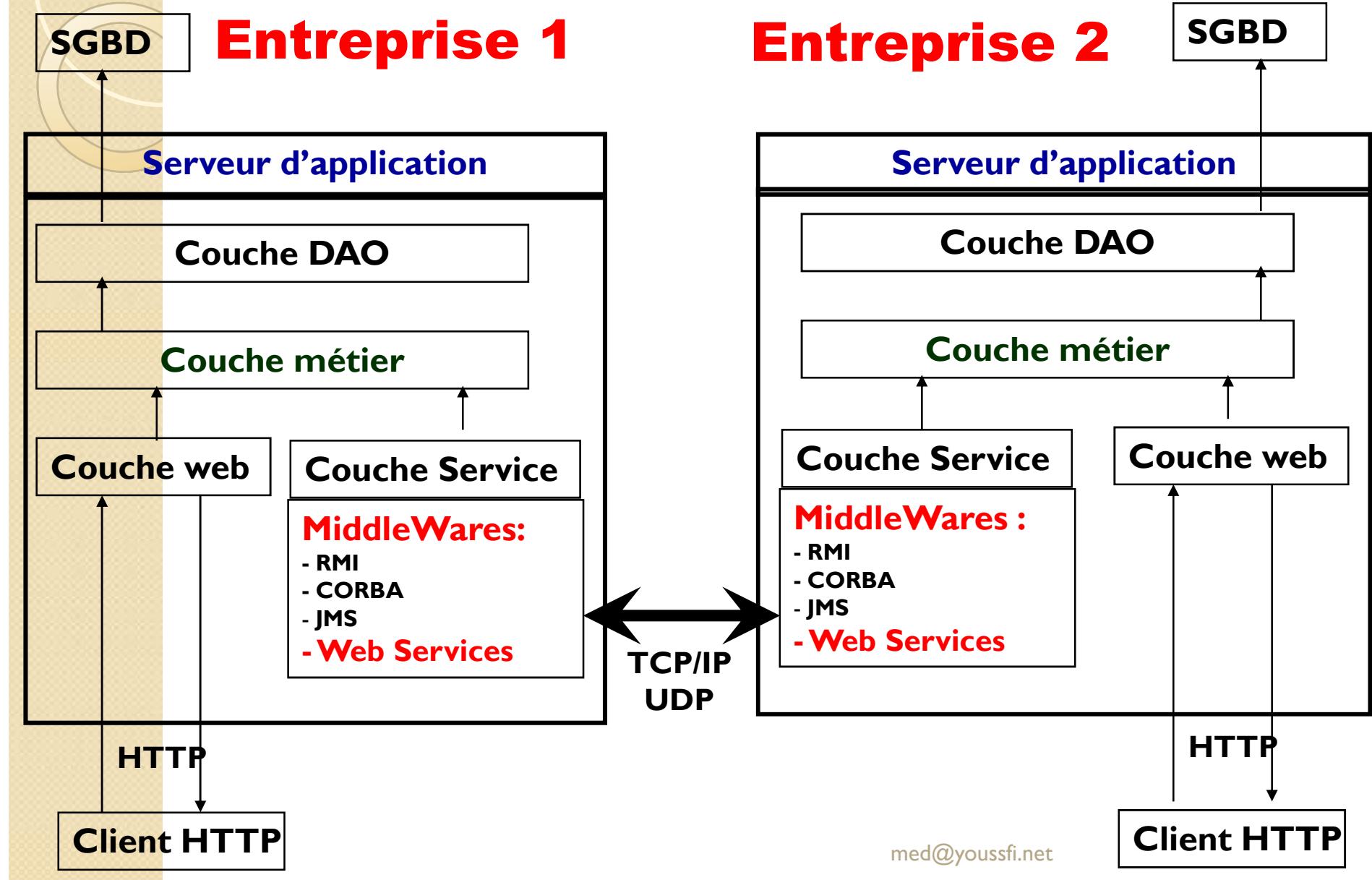
Exigences d'un projet informatique

- Exigences fonctionnelles:
 - Une application est créée pour répondre , tout d'abord, aux besoins fonctionnels des entreprises.
- Exigences Techniques :
 - Les performances:
 - Temps de réponse
 - Haute disponibilité et tolérance aux pannes
 - Eviter le problème de montée en charge
 - La maintenance:
 - Une application doit évoluer dans le temps.
 - Doit être fermée à la modification et ouverte à l'extension
 - Sécurité
 - Portabilité
 - Distribution
 - Capacité de communiquer avec d'autres applications distantes.
 - Capacité de fournir le service à différents type de clients (Desk TOP, Mobile, SMS, http...)
 -
 - Coût du logiciel

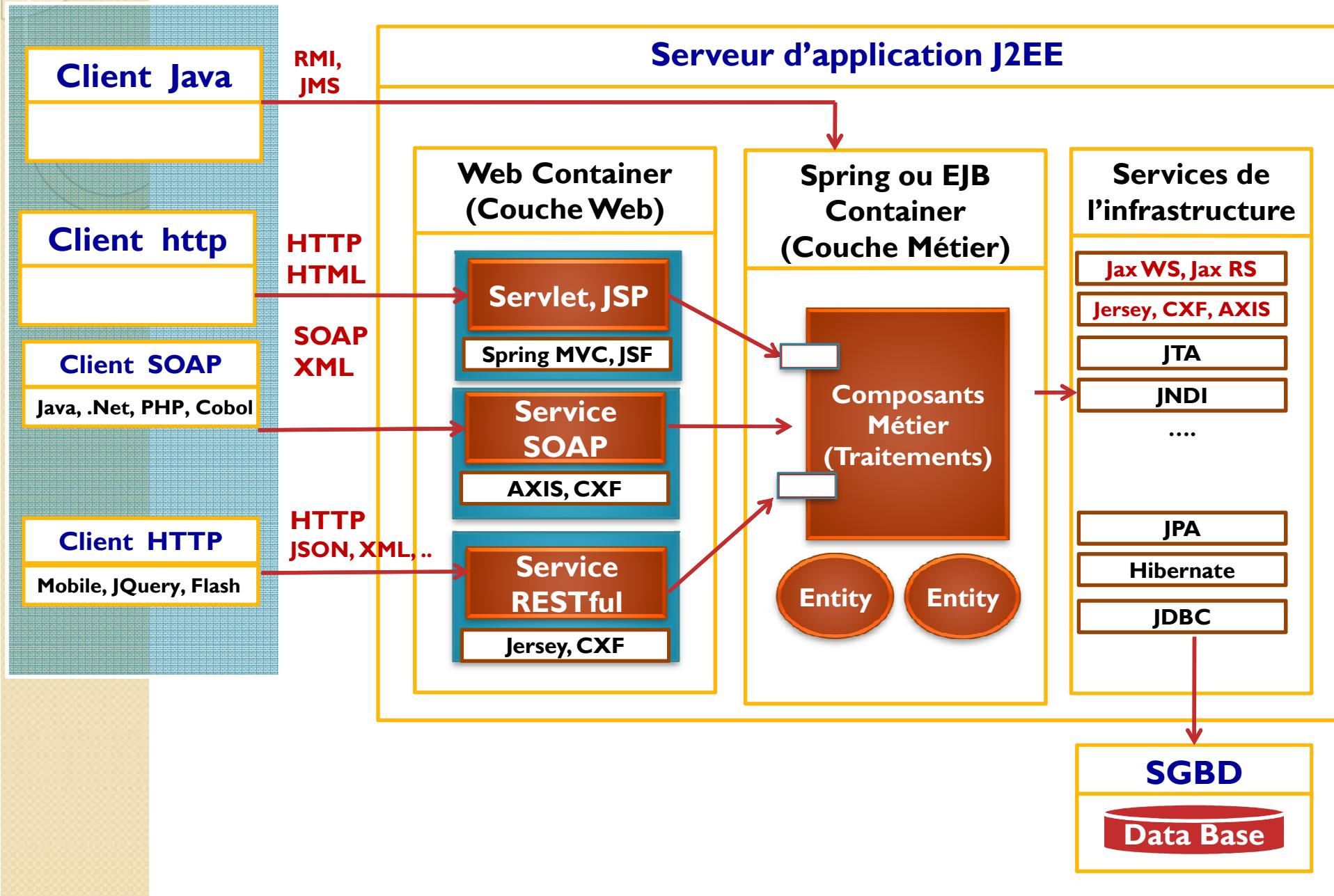
Constat

- Il est très difficile de développer un système logiciel qui respecte ces exigences sans **utiliser l'expérience des autres** :
 - Serveur d'application JEE:
 - JBOSS, Web Sphere,
 - GlassFish, Tomcat,
 - ...
 - Framework pour l'Inversion de contrôle:
 - Spring (Conteneur léger)
 - EJB (Conteneur lourd)
 - Frameworks :
 - Mapping objet relationnel (ORM) : JPA, Hibernate, Toplink, ...
 - Applications Web : Struts, JSF, SpringMVC
 -
 - Middlewares :
 - RMI, CORBA : Applications distribuées
 - **JAXWS pour Web services SOAP**
 - **JAXRS pour les Web services RESTful**
 - JMS : Communication asynchrone entre les application
 - ...

Vision globale d'une architectures Distribuées



Architecture J2EE





Web services

SOAP

WSDL



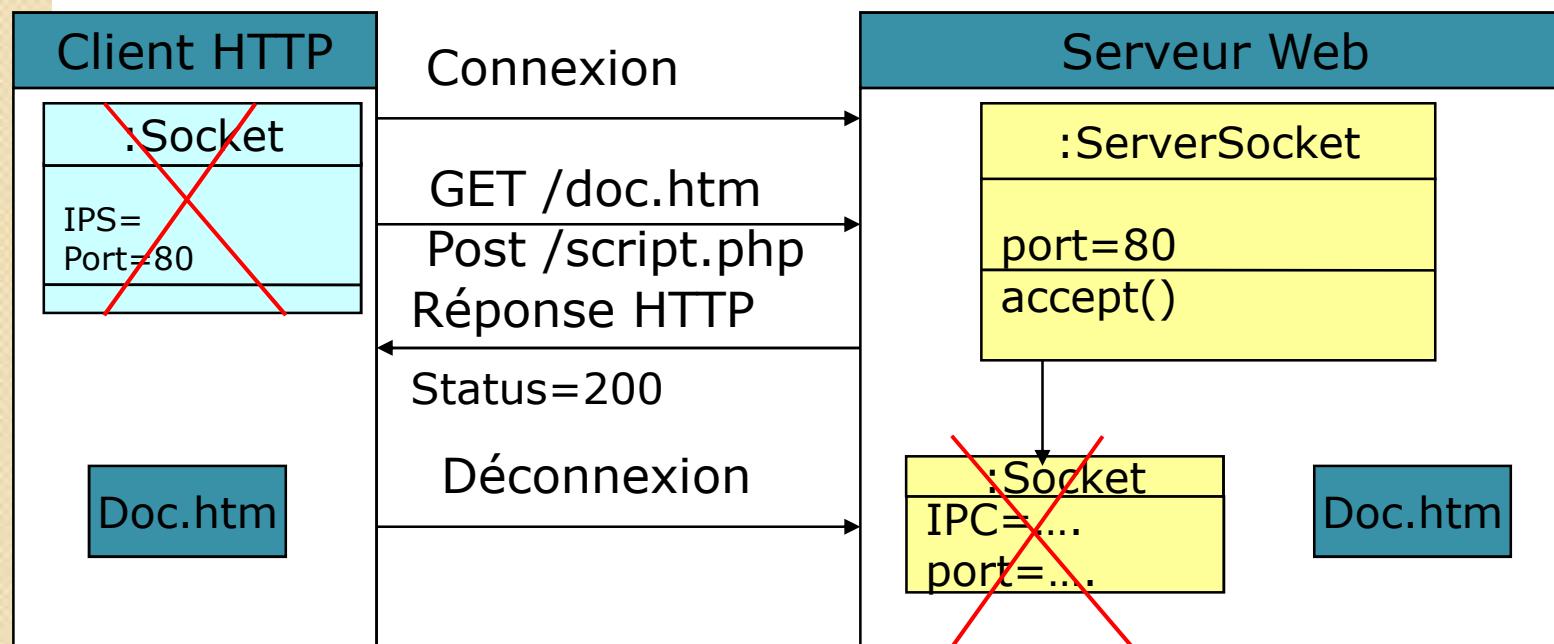
Introduction aux web services

- Les Web Services sont des composants web basés sur Internet (**HTTP**) qui exécutent des tâches précises et qui respectent un format spécifique (**XML**).
- Ils permettent aux applications de faire appel à des fonctionnalités à distance en simplifiant ainsi l'échange de données.
- Les Web Services permettent aux applications de dialoguer à travers le réseau, indépendamment de
 - leur plate-forme d'exécution
 - et de leur langage d'implémentation.
- Ils s'inscrivent dans la continuité d'initiatives telles que
 - CORBA (*Common Object Request Broker Architecture*, de l'OMG) en apportant toutefois une réponse plus simple, s'appuyant sur des technologies et standards reconnus et maintenant acceptés de tous.

LE PROTOCOLE HTTP

- **HTTP :HyperText Tranfert Protocol**
 - Protocole qui permet au client de récupérer des documents du serveur
 - Ces documents peuvent être statiques (contenu qui ne change pas : HTML, PDF, Image, etc..) ou dynamiques (Contenu généré dynamiquement au moment de la requête : PHP, JSP, ASP...)
 - Ce protocole permet également de soumissionner les formulaires
- **Fonctionnement (très simple en HTTP/1.0)**
 - Le client se connecte au serveur (Créer une socket)
 - Le client demande au serveur un document : Requête HTTP
 - Le serveur renvoi au client le document (status=200) ou d'une erreur (status=404 quand le document n'existe pas)
 - Déconnexion

Connexion



Méthodes du protocole HTTP

- Une requête HTTP peut être envoyée en utilisant les méthodes suivantes:
 - **GET** : Pour récupérer le contenu d'un document
 - **POST** : Pour soumissionner des formulaires (Envoyer, dans la requête, des données saisies par l'utilisateur)
 - **PUT** pour envoyer un fichier du client vers le serveur
 - **DELETE** permet de demander au serveur de supprimer un document.
 - **HEAD** permet de récupérer les informations sur un document (Type, Capacité, Date de dernière modification etc...)

Le client envoie la requête : Méthode POST

Entête de la requête

Post /Nom_Script HTTP/1.0

Accept: text/html

Accept-Language : fr

User-Agent : Mozilla/4.0

*** saut de ligne ***

login=Value1& pass=Value2

& Var3=Value3

corps de la requête

Le client envoie la requête : Méthode GET

Entête de la requête

GET /Nom_Script?login=val1&pass=val2&.... HTTP/1.0

Accept: text/html

Accept-Language : fr

User-Agent : Mozilla/4.0

corps de la requête est vide

Le Serveur retourne la réponse :

Entête de la réponse

HTTP/1.0 200 OK

Date : Wed, 05Feb02 15:02:01 GMT

Server : Apache/1.3.24

Last-Modified : Wed 02Oct01 24:05:01GMT

Content-Type : Text/html

Content-length : 4205

*** saut de ligne ***

<HTML><HEAD>

....

</BODY></HTML>

Ligne de Status

Date du serveur

Nom du Serveur

Dernière modification

Type de contenu

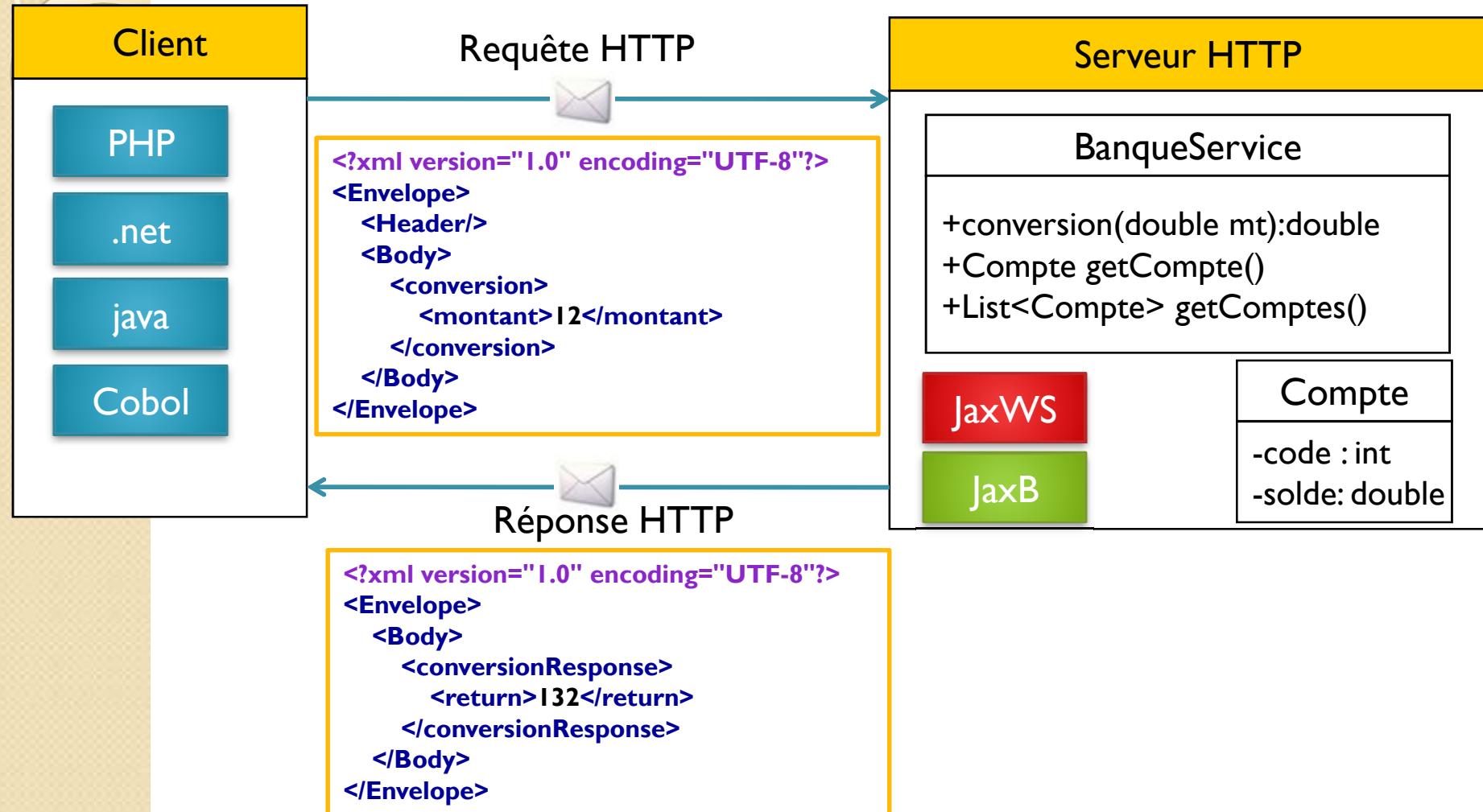
Sa taille

Le fichier que le client va afficher

corps de la réponse

net

L'idée des Web Services



Requête SOAP avec POST

Entête de la requête

Post /Nom_Script HTTP/1.0

Accept: application/xml

Accept-Language : fr

***** saut de ligne *****

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <conversion xmlns="http://bk/test">
      <montant xmlns="">12</montant>
    </conversion>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

corps de la requête SOAP

Réponse SOAP :

HTTP/1.0 200 OK

Date :Wed, 05Feb02 15:02:01 GMT

Server :Apache/1.3.24

Mime-Version 1.0

Last-Modified :Wed 02Oct01 24:05:01GMT

Content-Type :Text/xml

Content-length : 4205

***** saut de ligne *****

Entête de la réponse

```
<?xml version="1.0" ?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:ns1="http://bk/test">
  <soapenv:Body>
    <ns1:conversionResponse>
      <return>132.0</return>
    </ns1:conversionResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

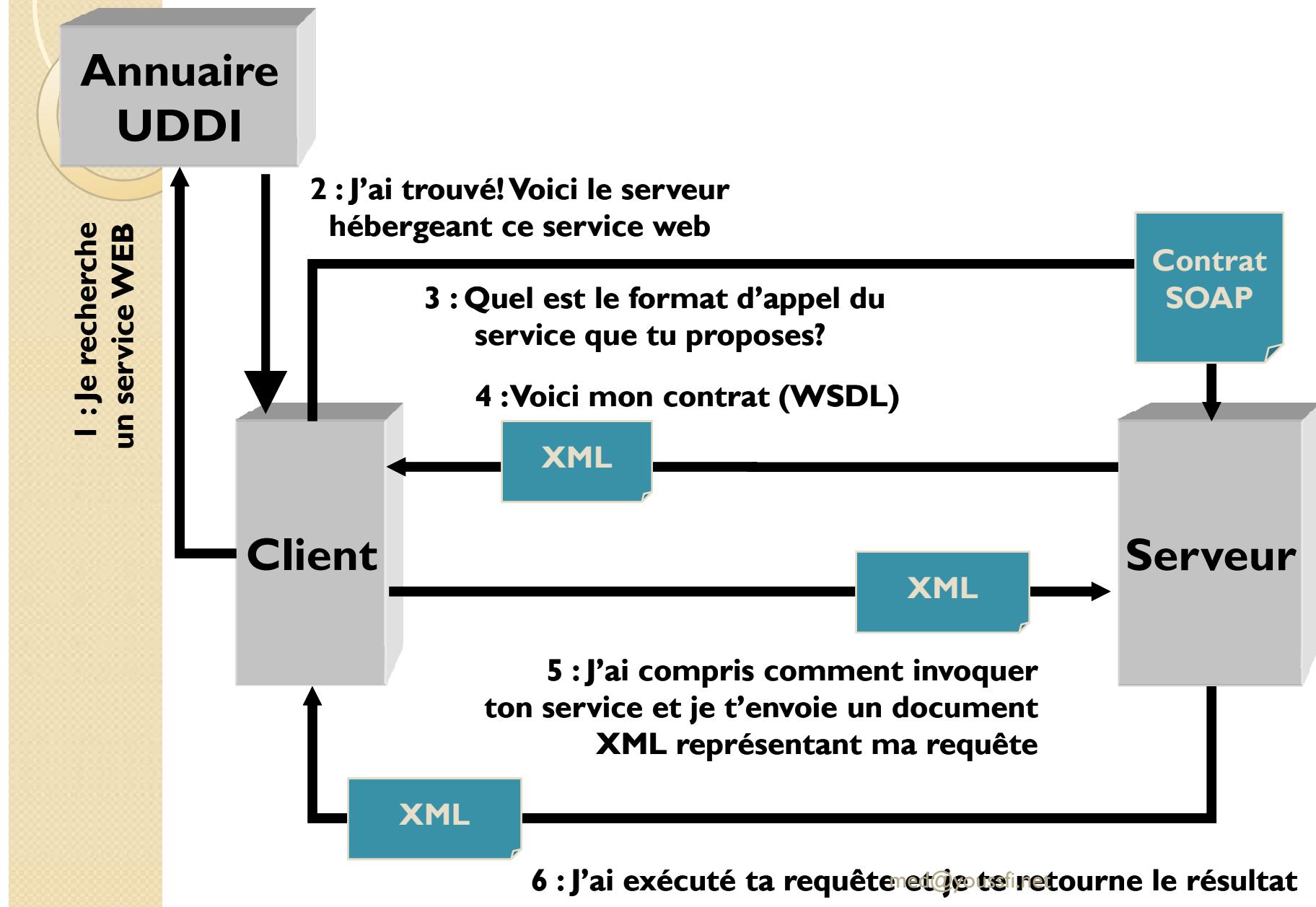
corps de la réponse



Concepts des web services

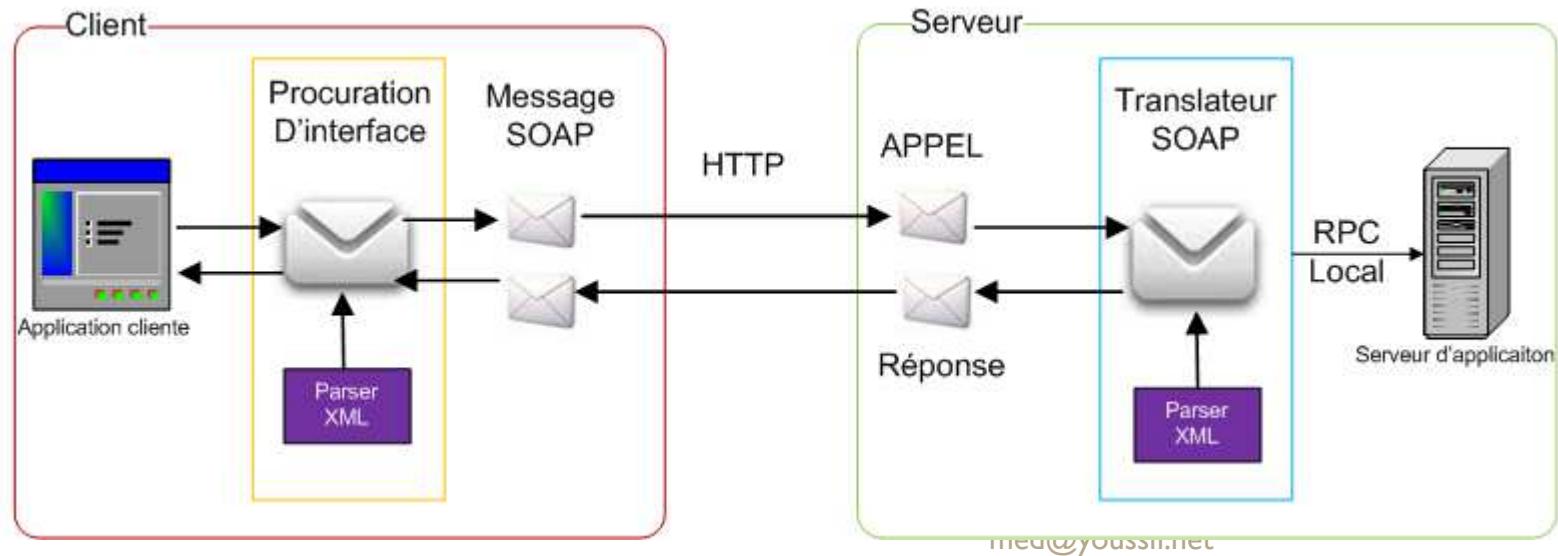
- Le concept des Web Services s'articule actuellement autour des trois concepts suivants :
 - SOAP (*Simple Object Access Protocol*)
 - est un protocole d'échange inter-applications indépendant de toute plate-forme, basé sur le langage XML.
 - Un appel de service SOAP est un flux ASCII encadré dans des balises XML et transporté dans le protocole HTTP.
 - WSDL (*Web Services Description Language*)
 - donne la description au format XML des Web Services en précisant les méthodes pouvant être invoquées, leurs signatures et le point d'accès (URL, port, etc..).
 - C'est, en quelque sorte, l'équivalent du langage IDL pour la programmation distribuée CORBA.
 - UDDI (*Universal Description, Discovery and Integration*)
 - normalise une solution d'annuaire distribué de Web Services, permettant à la fois la publication et l'exploration (recherche) de Web Services.
 - UDDI se comporte lui-même comme un Web service dont les méthodes sont appelées via le protocole SOAP.

Cycle de vie d'utilisation

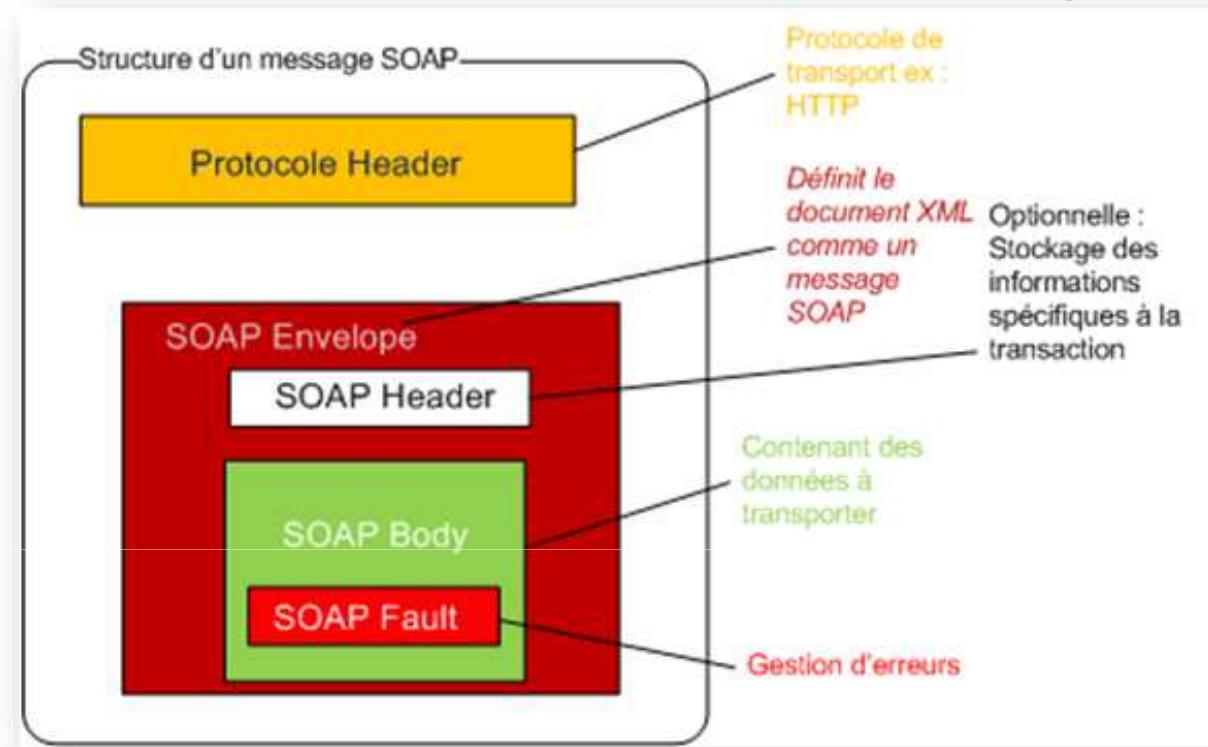


SOAP

- SOAP est un protocole d'invocation de méthodes sur des services distants. **Basé sur XML**,
- SOAP a pour principal objectif **d'assurer la communication entre machines**.
- Le protocole permet d'appeler une méthode RPC et d'envoyer des messages aux machines distantes via un protocole de transport (HTTP).
- Ce protocole est très bien adapté à l'utilisation des services Web, car il permet de fournir au client une grande quantité d'informations récupérées sur un réseau de serveurs tiers, voyez :



Structure d'un message SOAP



```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"  
    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding">  
    <soap:Header>  
        <!-- en-tête -->  
    </soap:Header>  
    <soap:Body>  
        <!-- corps -->  
    </soap:Body>  
</soap:Envelope>
```



Structure d'un message SOAP

- **SOAP envelope** (enveloppe)
 - Est l'élément de base du message SOAP.
 - L'enveloppe contient la spécification des espaces de désignation (namespace) et du codage de données.
- **SOAP header**
 - (entête) est une partie facultative qui permet d'ajouter des fonctionnalités à un message SOAP de manière décentralisée sans agrément entre les parties qui communiquent.
 - L'entête est utile surtout, quand le message doit être traité par plusieurs intermédiaires.
- **SOAP body** (corps) est un *container* pour les informations mandataires à l'intention du récepteur du message, il contient les méthodes et les paramètres qui seront exécutés par le destinataire final.
- **SOAP fault** (erreur) est un élément facultatif défini dans le corps SOAP et qui est utilisé pour reporter les erreurs.

Mise en œuvre des web services avec JAX-WS

med@youssf.net

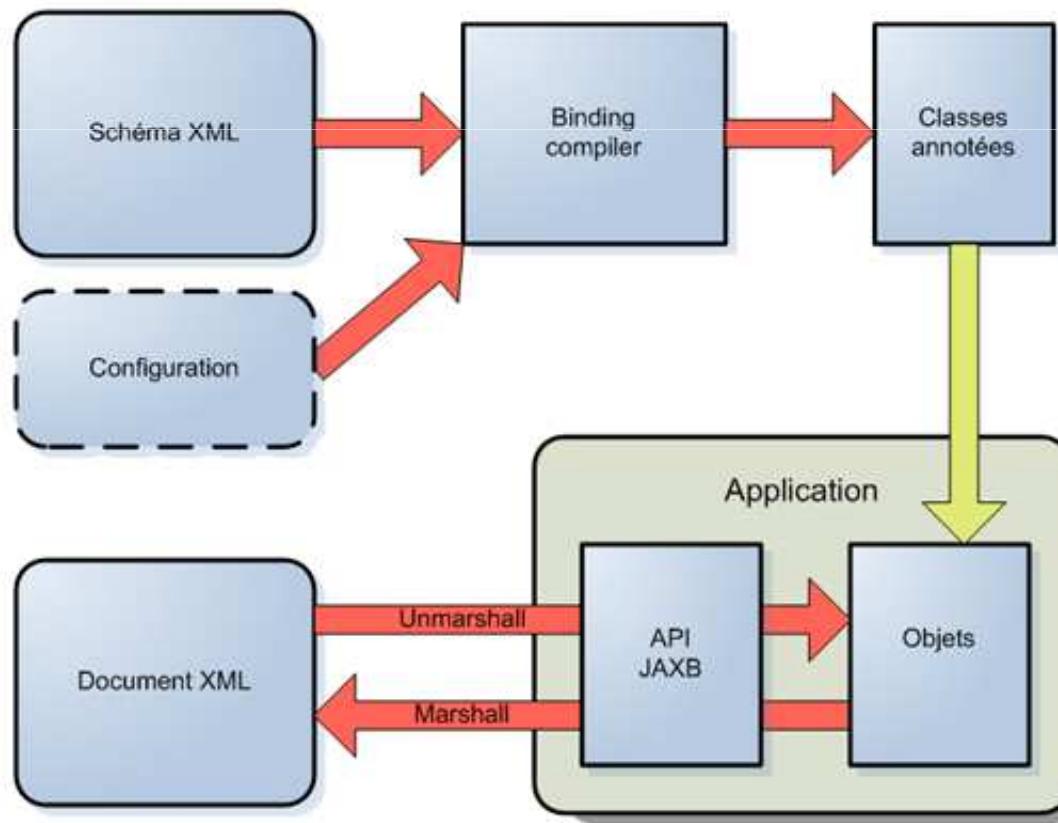
JAX-WS

- JAX-WS est la nouvelle appellation de JAX-RPC (Java API for XML Based RPC) qui permet de développer très simplement des services web en Java.
- JAX-WS fournit un ensemble d'annotations pour mapper la correspondance Java-WSDL. Il suffit pour cela d'annoter directement les classes Java qui vont représenter le service web.
- Dans l'exemple ci-dessous, une classe Java utilise des annotations JAX-WS qui vont permettre par la suite de générer le document WSDL. Le document WSDL est auto-généré par le serveur d'application au moment du déploiement :

```
@WebService(serviceName="BanqueWS")
public class BanqueService {
    @WebMethod(operationName="ConversionEuroToDh")
    public double conversion(@WebParam(name="montant")double mt){
        return mt*11;
    }
    @WebMethod
    public String test(){    return "Test";    }
    @WebMethod
    public Compte getCompte(){    return new Compte (1,7000);    }
    @WebMethod
    public List<Compte> getComptes(){
        List<Compte> cptes=new ArrayList<Compte>();
        cptes.add (new Compte (1,7000)); cptes.add (new Compte (2,9000));
        return cptes;
    }
}
```

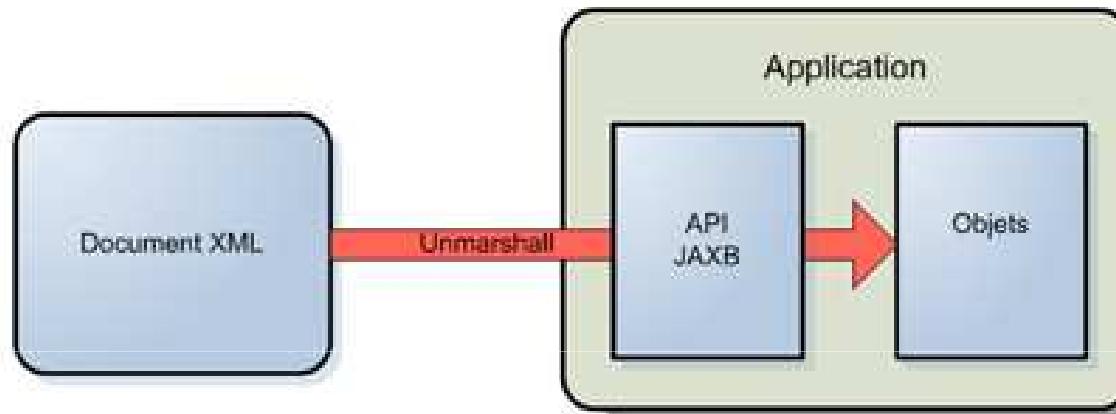
JAX-WS / JAXB

- JAX-WS s'appuie sur l'API JAXB 2.0 pour tout ce qui concerne la correspondance entre document XML et objets Java.
- JAXB 2.0 permet de mapper des objets Java dans un document XML et vice versa.
- Il permet aussi de générer des classes Java à partir un schéma XML et vice et versa.

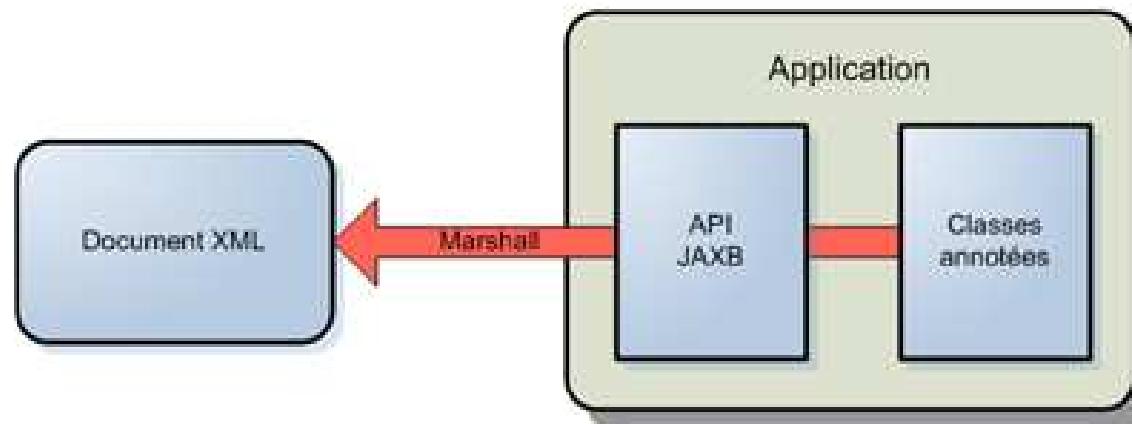


Principe de JAXB

- Le mapping d'un document XML à des objets (unmarshal)



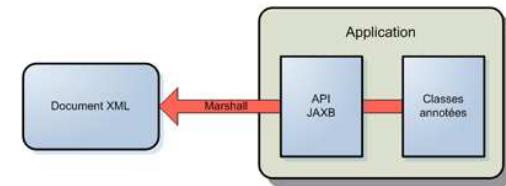
- La création d'un document XML à partir d'objets (marshal)



Générer XML à partir des objet java avec JAXB

```
package ws;
import java.io.File; import java.util.Date;
import javax.xml.bind.*;
public class Banque {
    public static void main(String[] args) throws Exception {
        JAXBContext context=JAXBContext.newInstance(Compte.class);
        Marshaller marshaller=context.createMarshaller();
        marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,true);
        Compte cp=new Compte(1,8000,new Date());
        marshaller.marshal(cp,new File("comptes.xml"));
    }
}
```

```
package ws;
import java.util.Date;
import javax.xml.bind.annotation.*;
@XmlRootElement
public class Compte {
    private int code;
    private float solde;
    private Date dateCreation;
    // Constructeur sans paramètre
    // Constructeur avec paramètres
    // Getters et Setters
}
```



Fichier XML Généré : comptes.xml

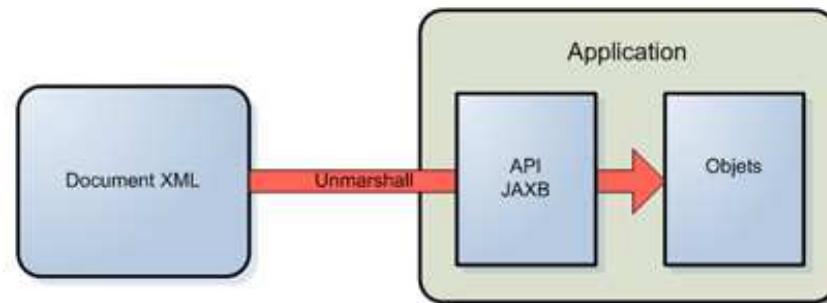
```
<?xml version="1.0" encoding="UTF-8"?>
<compte>
    <code>1</code>
    <dateCreation>
        2014-01-16T12:33:16.960Z
    </dateCreation>
    <solde>8000.0</solde>
</compte>
```

Générer des objets java à partir des données XML

```
package ws;
import java.io.*;
import javax.xml.bind.*;
public class Banque2 {
    public static void main(String[] args) throws Exception {
        JAXBContext jc=JAXBContext.newInstance(Compte.class);
        Unmarshaller unmarshaller=jc.createUnmarshaller();
        Compte cp=(Compte) unmarshaller.unmarshal(new File("comptes.xml"));
        System.out.println(cp.getCode()+" - "+cp.getSolde()+
                           "- "+cp.getDateCreation());
    }
}
```

Fichier XML Source : comptes.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<compte>
    <code>1</code>
    <dateCreation>
        2014-01-16T12:33:16.960Z
    </dateCreation>
    <solde>8000.0</solde>
</compte>
```



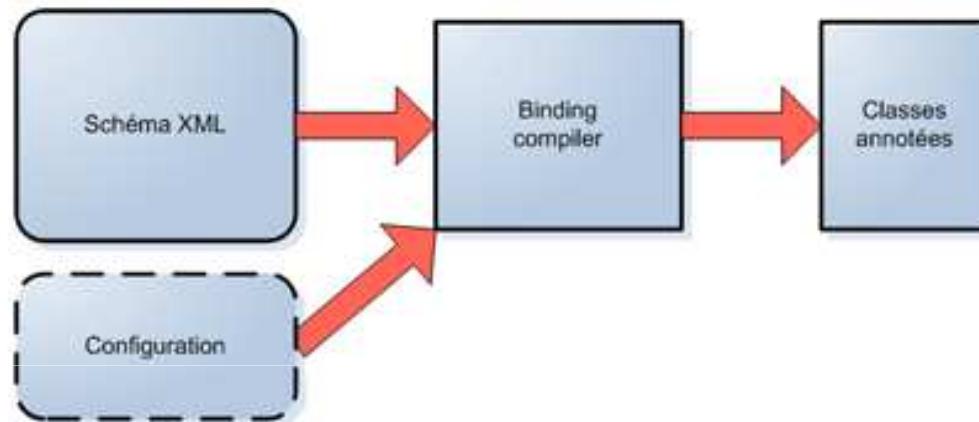
Générer un schéma XML à partir d'une classe avec JaxB

```
package ws;
import java.io.*;
import javax.xml.bind.*;import javax.xml.transform.Result;
import javax.xml.transform.stream.StreamResult;
public class Banque {
    public static void main(String[] args) throws Exception {
        JAXBContext context=JAXBContext.newInstance(Compte.class);
        context.generateSchema(new SchemaOutputResolver() {
            @Override
            public Result createOutput(String namespaceUri, String suggestedFileName)
throws IOException {
                File f=new File("compte.xsd");
                StreamResult result=new StreamResult(f);
                result.setSystemId(f.getName());
                return result;
            }
        });
    }
});}}
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xsschema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xselement name="compte" type="compte"/>
    <xsccomplexType name="compte">
        <xsssequence>
            <xselement name="code" type="xs:int"/>
            <xselement name="dateCreation" type="xs:dateTime" minOccurs="0"/>
            <xselement name="solde" type="xs:float"/>
        </xsssequence>
    </xsccomplexType>
</xsschema>
```

Génération des classes à partir d'un schéma XML

- Pour permettre l'utilisation et la manipulation d'un document XML, JAXB propose de générer un ensemble de classes à partir du schéma XML du document.



- L'implémentation de référence fournit l'outil **xjc** pour générer les classes à partir d'un schéma XML.
- L'utilisation la plus simple de l'outil **xjc** est de lui fournir simplement le fichier qui contient le schéma XML du document à utiliser.

```
C:\Windows\system32\cmd.exe
C:\Users\youssfi\w\JB2\src>xjc compte.xsd
parsing a schema...
compiling a schema...
generated\Compte.java
generated\ObjectFactory.java
C:\Users\youssfi\w\JB2\src>
```

Classes générées par XJC

- L'outil XJC génère deux classes :
 - Compte.java qui correspond au type complexe Compte dans le schéma xml.
 - ObjectFactory.java : une fabrique qui permet de créer des objets de type Compte.



Quelques annotations JAXB

Annotation	Description
@XmlRootElement	Associer une classe ou une énumération à un élément XML
@XmlSchema	Associer un espace de nommage à un package
@XmlTransient	Marquer une entité pour ne pas être mappée dans le document XML
@XmlAttribute	Convertir une propriété en un attribut dans le document XML
@XmlElement	Convertir une propriété en un élément dans le document XML
@XmlAccessorType	Préciser comment un champ ou une propriété est sérialisé
@XmlNs	Associer un préfixe d'un espace de nommage à un URI

Comment développer un web service JAW-WS

1. Créer le service Web

- Développer le web service
- Déployer le web service
 - Un serveur HTTP
 - Un Conteneur WS (JaxWS, AXIS, CXF, etc...)

2. Tester le web service avec un analyseur SOAP

- SoapUI,
- Oxygen, etc...

3. Créer les clients :

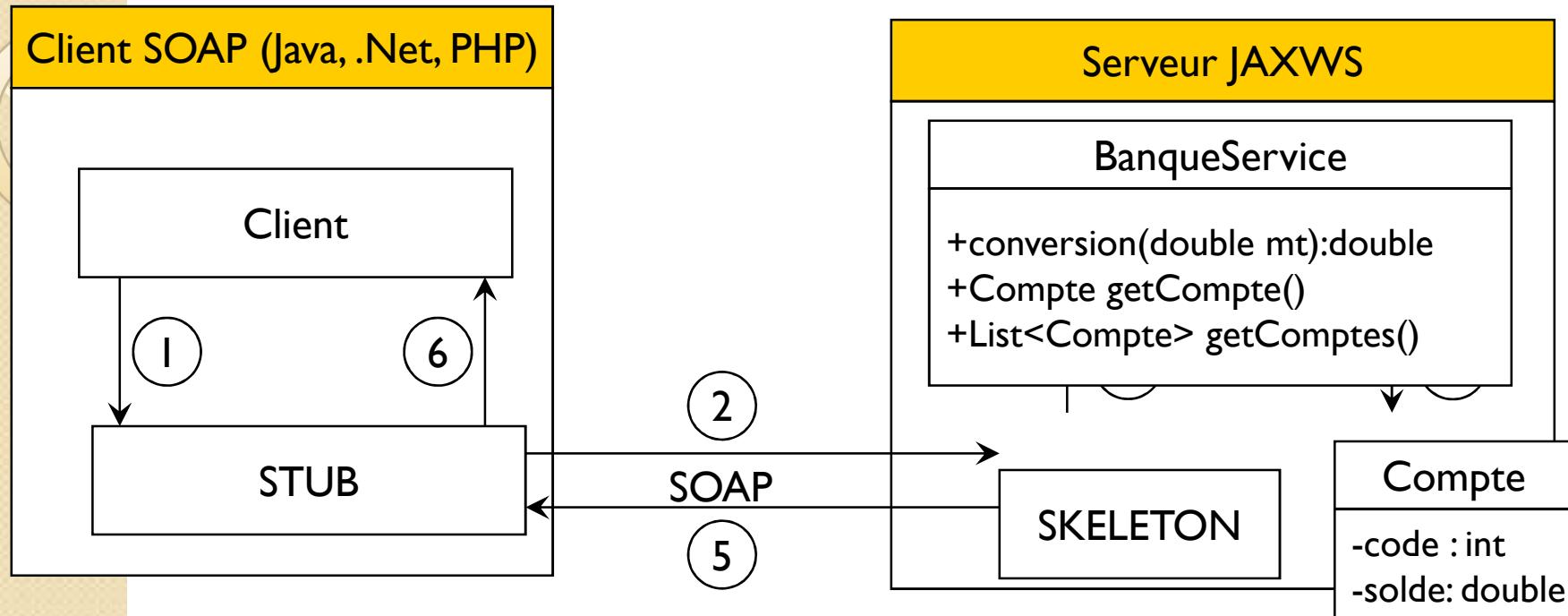
- Un client Java
- Un client .Net
- Un client PHP



Application

- Créer un web service java en utilisant JaxWS qui permet de :
 - Convertir un montant de l'auro en DH
 - Consulter un compte sachant son code
 - Consulter une liste de comptes
- Tester le web service avec un analyseur SOAP
- Créer un client Java
- Créer un client .Net
- Créer un client PHP

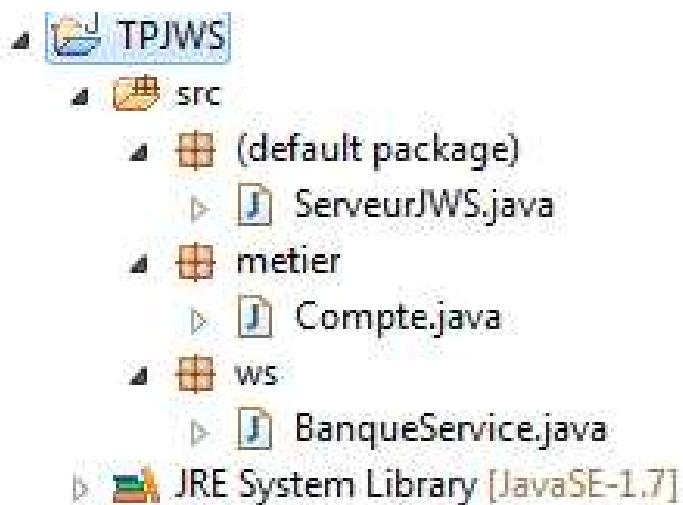
Architecture I



- 1 Le client demande au stub de faire appel à la méthode conversion(12)
- 2 Le Stub se connecte au Skeleton et lui envoie une requête SOAP
- 3 Le Skeleton fait appel à la méthode du web service
- 4 Le web service retourne le résultat au Skeleton
- 5 Le Skeleton envoie le résultat dans une la réponse SOAP au Stub
- 6 Le Stub fournie le résultat au client

Développer le web service avec JAX-WS

- Outils à installer :
 - JDK1.7
 - Editeur java : Eclipse.
- Créer un projet java en utilisant JDK1.7 comme compilateur et environnement d'exécution java.
 - Structure du projet :



Implémentation de la classe Compte

```
package metier;
import java.util.Date;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
@XmlRootElement
@XmlAccessorType(XmlAccessType.FIELD)
public class Compte {
    private Long code;
    private double solde;
    @XmlTransient
    private Date dateCreation;
    // Constructeur sans paramètre et avec paramètre
    // Getters et setters
}
```

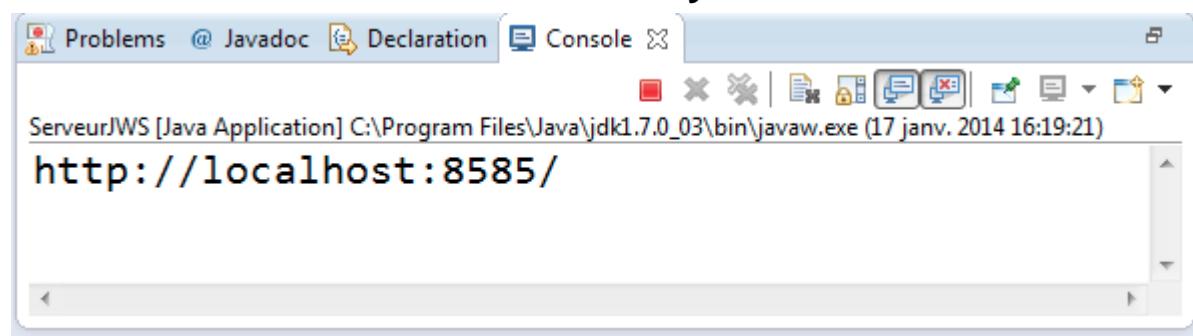
Implémentation du web service

```
package ws;
import java.util.*;
import javax.jws.*;
import metier.Compte;
@WebService(serviceName="BanqueWS")
public class BanqueService {
    @WebMethod(operationName="ConversionEuroToDh")
    public double conversion(@WebParam(name="montant")double mt){
        return mt*11;
    }
    @WebMethod
    public Compte getCompte(@WebParam(name="code")Long code){
        return new Compte (code,7000,new Date());
    }
    @WebMethod
    public List<Compte> getComptes(){
        List<Compte> cptes=new ArrayList<Compte>();
        cptes.add (new Compte (1L,7000,new Date()));
        cptes.add (new Compte (2L,7000,new Date()));
        return cptes;
    }
}
```

Simple Serveur JAX WS

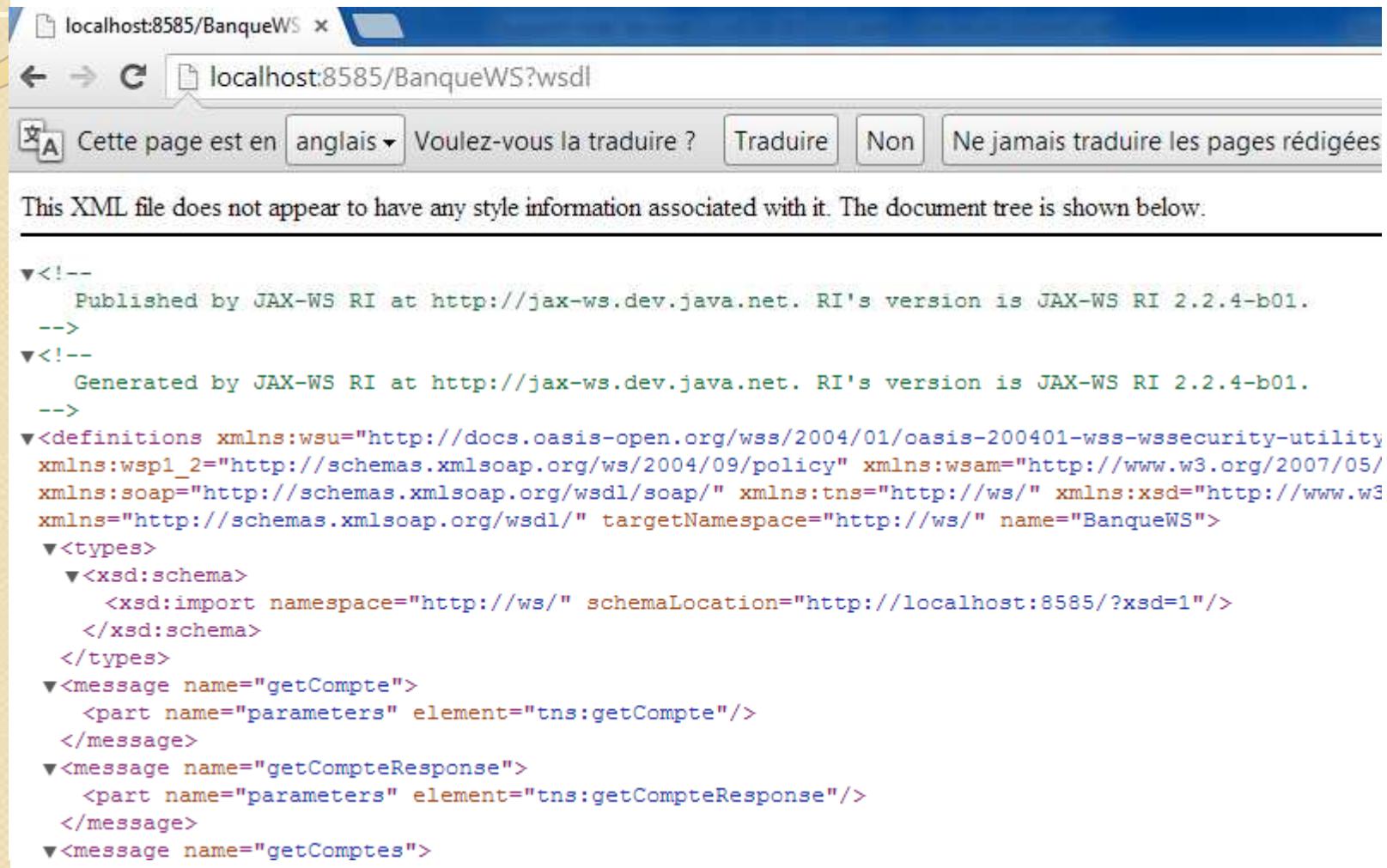
```
import javax.xml.ws.Endpoint;
import ws.BanqueService;
public class ServeurJWS {
    public static void main(String[] args) {
        String url="http://localhost:8585/";
        Endpoint.publish(url, new BanqueService());
        System.out.println(url);
    }
}
```

Exécution du serveur Java



Analyser le WSDL

- Pour Visualiser le WSDL, vous pouvez utiliser un navigateur web



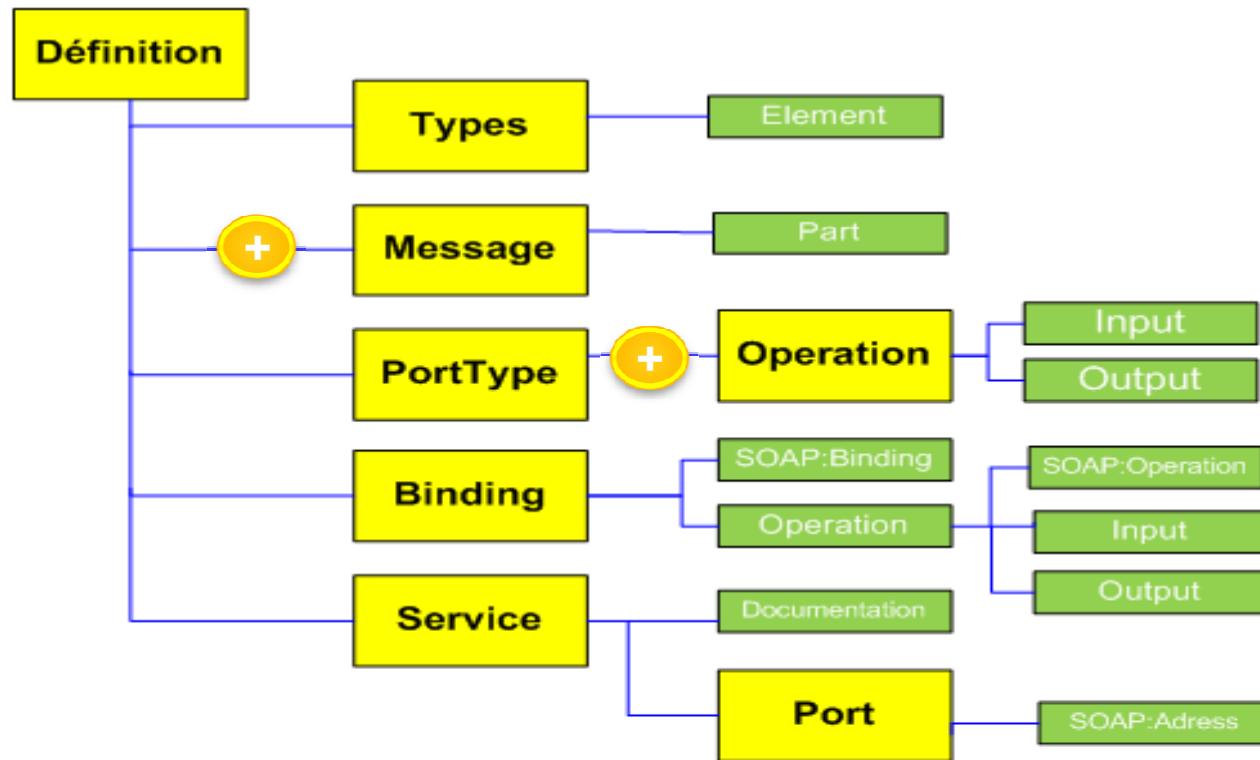
The screenshot shows a web browser window with the URL `localhost:8585/BanqueWS?wsdl`. The page content is as follows:

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2.4-b01. -->
<!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2.4-b01. -->
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility"
    xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
    xmlns:wsam="http://www.w3.org/2007/05/
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://ws/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    targetNamespace="http://ws/"
    name="BanqueWS">
    <types>
        <xsd:schema>
            <xsd:import namespace="http://ws/" schemaLocation="http://localhost:8585/?xsd=1"/>
        </xsd:schema>
    </types>
    <message name="getCompte">
        <part name="parameters" element="tns:getCompte"/>
    </message>
    <message name="getCompteResponse">
        <part name="parameters" element="tns:getCompteResponse"/>
    </message>
    <message name="getComptes">
```

Structure du WSDL

- Un document WSDL se compose d'un ensemble d'éléments décrivant les types de données utilisés par le service, les messages que le service peut recevoir, ainsi que les liaisons SOAP associées à chaque message.
- Le schéma suivant illustre la structure du langage WSDL qui est un document XML, en décrivant les relations entre les sections constituant un document WSDL.



Structure d'un document WSDL

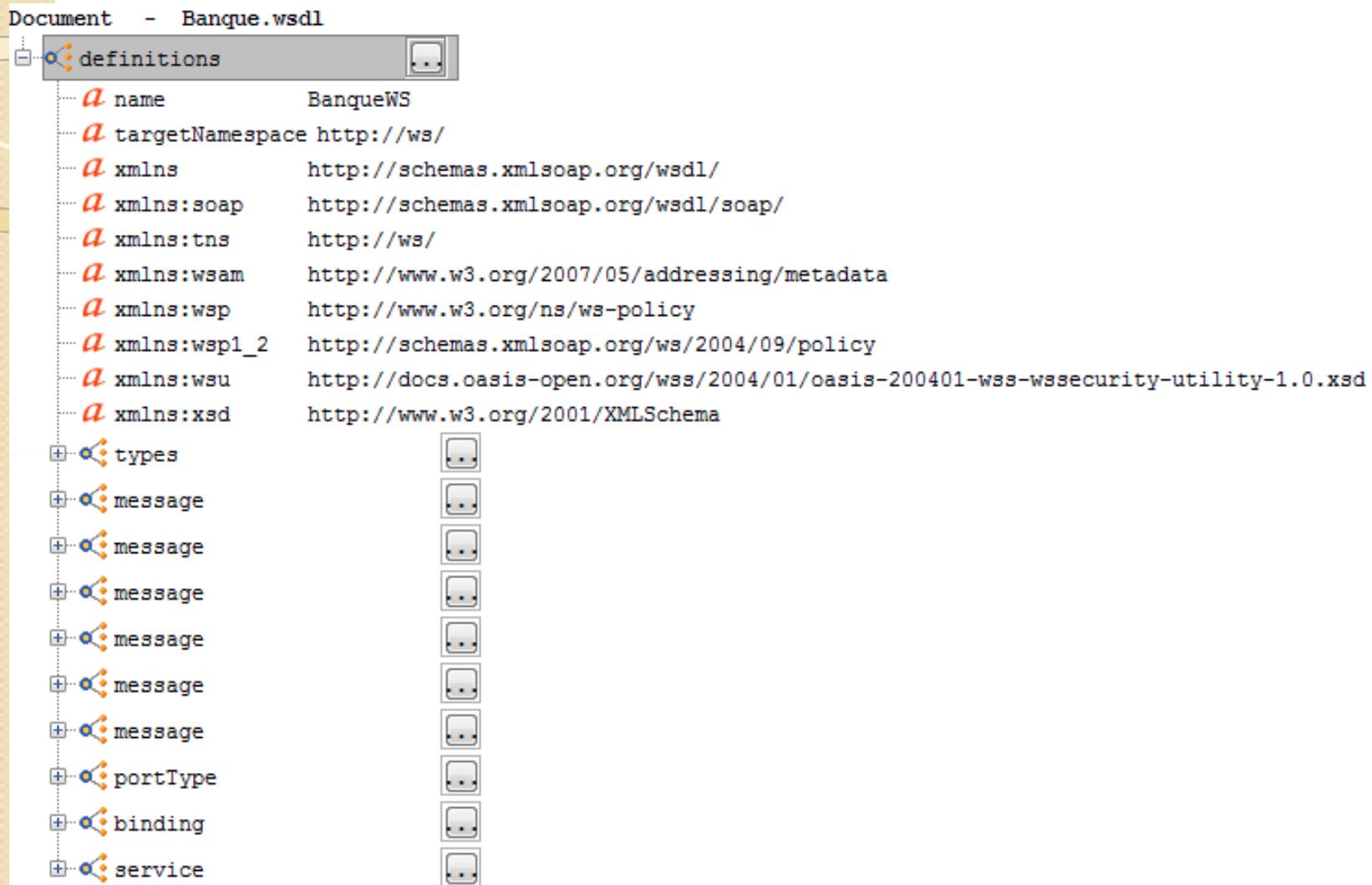
medyoussiti.net



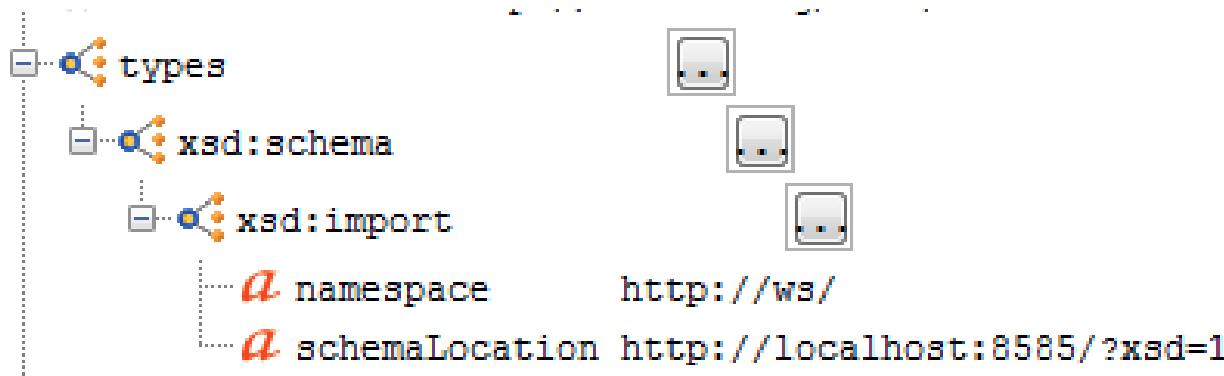
Structure du WSDL

- Un fichier WSDL contient donc sept éléments.
 - **Types** : fournit la définition de types de données utilisés pour décrire les messages échangés.
 - **Messages** : représente une définition abstraite (noms et types) des données en cours de transmission.
 - **PortTypes** : décrit un ensemble d'opérations. Chaque opération a zéro ou un message en entrée, zéro ou plusieurs messages de sortie ou d'erreurs.
 - **Binding** : spécifie une liaison entre un `<portType>` et un protocole concret (SOAP, HTTP...).
 - **Service** : indique les adresses de port de chaque liaison.
 - **Port** : représente un point d'accès de services défini par une adresse réseau et une liaison.
 - **Opération** : c'est la description d'une action exposée dans le port.

Structure du WSDL



Elément Types



XML Schema

localhost:8585/?xsd=1

Cette page est en **anglais** ▾ Voulez-vous la traduire ? **Traduire** **Non** Ne jamais traduire les pages rédigées en anglais

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2.4-b01. -->
<xsd:schema xmlns:tns="http://ws/" xmlns:xs="http://www.w3.org/2001/XMLSchema" version="1.0" targetNamespace="http://ws/">
  <xsd:element name="ConversionEuroToDh" type="tns:ConversionEuroToDh"/>
  <xsd:element name="ConversionEuroToDhResponse" type="tns:ConversionEuroToDhResponse"/>
  <xsd:element name="compte" type="tns:compte"/>
  <xsd:element name="getCompte" type="tns:getCompte"/>
  <xsd:element name="getCompteResponse" type="tns:getCompteResponse"/>
  <xsd:element name="getComptes" type="tns:getComptes"/>
  <xsd:element name="getComptesResponse" type="tns:getComptesResponse"/>
  <xsd:complexType name="ConversionEuroToDh">
    <xsd:sequence>
      <xsd:element name="montant" type="xs:double"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="ConversionEuroToDhResponse">
    <xsd:sequence>
      <xsd:element name="return" type="xs:double"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="getCompte">
    <xsd:sequence>
      <xsd:element name="code" type="xs:long" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="getCompteResponse">
    <xsd:sequence>
      <xsd:element name="return" type="tns:compte" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="compte">
    <xsd:sequence>
      <xsd:element name="code" type="xs:long" minOccurs="0"/>
      <xsd:element name="solde" type="xs:double"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="getComptes">
    <xsd:sequence/>
  </xsd:complexType>
  <xsd:complexType name="getComptesResponse">
    <xsd:sequence>
      <xsd:element name="return" type="tns:compte" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:tns="http://ws/" xmlns:xs="http://www.w3.org/2001/XMLSchema" version="1.0"
targetNamespace="http://ws/">
  <xs:element name="ConversionEuroToDh" type="tns:ConversionEuroToDh"></xs:element>
  <xs:element name="ConversionEuroToDhResponse" type="tns:ConversionEuroToDhResponse"/>
  <xs:element name="compte" type="tns:compte"></xs:element>
  <xs:element name="getCompte" type="tns:getCompte"></xs:element>
  <xs:element name="getCompteResponse" type="tns:getCompteResponse"></xs:element>
  <xs:element name="getComptes" type="tns:getComptes"></xs:element>
  <xs:element name="getComptesResponse" type="tns:getComptesResponse"></xs:element>
<xs:complexType name="ConversionEuroToDh">
  <xs:sequence>
    <xs:element name="montant" type="xs:double"></xs:element>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="ConversionEuroToDhResponse">
  <xs:sequence>
    <xs:element name="return" type="xs:double"></xs:element>
  </xs:sequence>
</xs:complexType>
```

XML Schema

```
<xs:complexType name="getCompte">
  <xs:sequence>
    <xs:element name="code" type="xs:long" minOccurs="0"></xs:element>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="getCompteResponse">
  <xs:sequence>
    <xs:element name="return" type="tns:compte" minOccurs="0"></xs:element>
  </xs:sequence>
</xs:complexType>

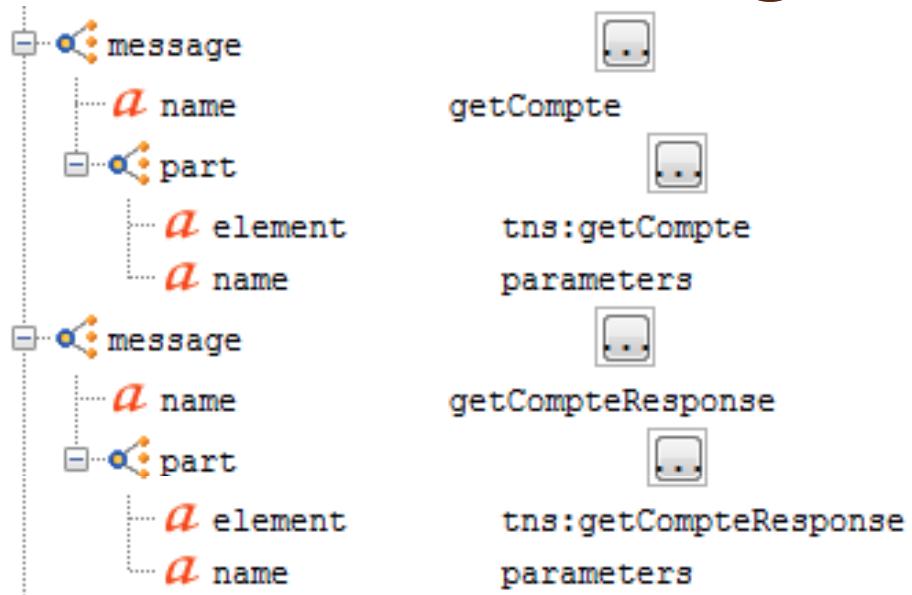
<xs:complexType name="compte">
  <xs:sequence>
    <xs:element name="code" type="xs:long" minOccurs="0"></xs:element>
    <xs:element name="solde" type="xs:double"></xs:element>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="getComptes">
  <xs:sequence></xs:sequence>
</xs:complexType>

<xs:complexType name="getComptesResponse">
  <xs:sequence>
    <xs:element name="return" type="tns:compte" minOccurs="0" maxOccurs="unbounded"></xs:element>
  </xs:sequence>
</xs:complexType>

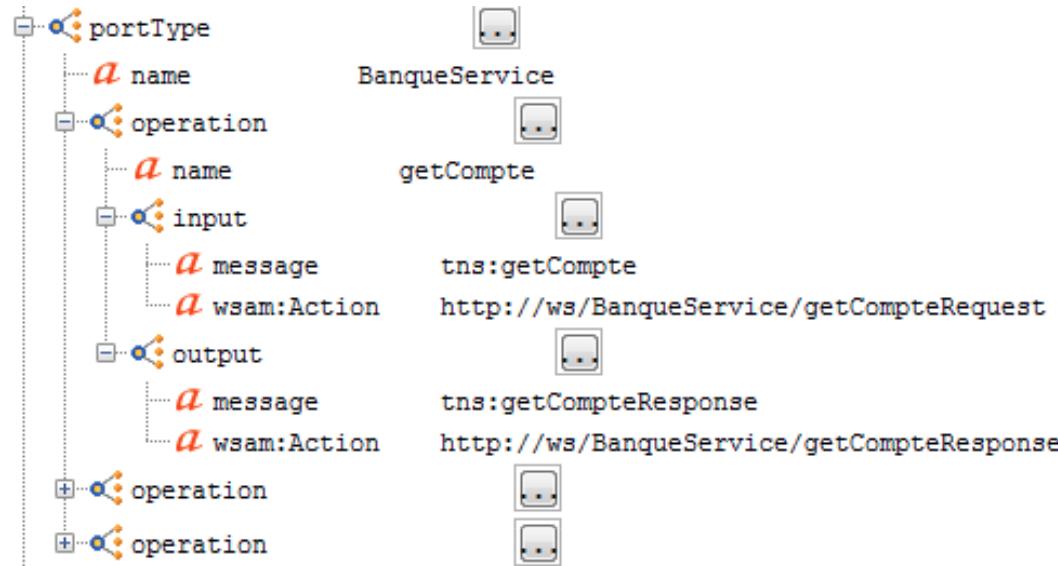
</xs:schema>
```

Elément message



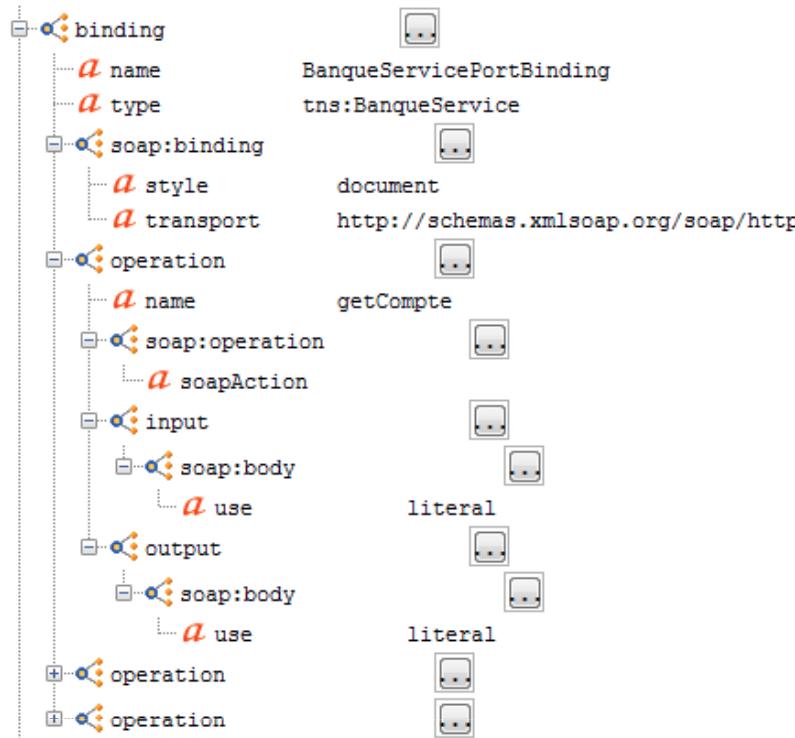
```
<message name="getCompte">
    <part name="parameters" element="tns:getCompte"></part>
</message>
<message name="getCompteResponse">
    <part name="parameters" element="tns:getCompteResponse"></part>
</message>
```

Elément portType



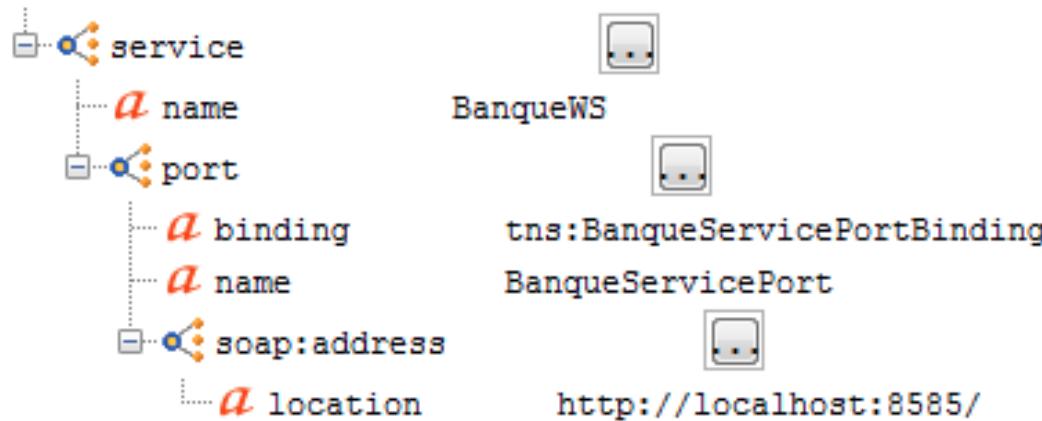
```
<portType name="BanqueService">
  <operation name="getCompte">
    <input wsam:Action="http://ws/BanqueService/getCompteRequest" message="tns:getCompte"></input>
    <output wsam:Action="http://ws/BanqueService/getCompteResponse" message="tns:getCompteResponse"></output>
  </operation>
  <operation name="getComptes">
    <input wsam:Action="http://ws/BanqueService/getComptesRequest" message="tns:getComptes"></input>
    <output wsam:Action="http://ws/BanqueService/getComptesResponse" message="tns:getComptesResponse"></output>
  </operation>
  <operation name="ConversionEuroToDh">
    .....
  </operation>
</portType>
```

Elément binding



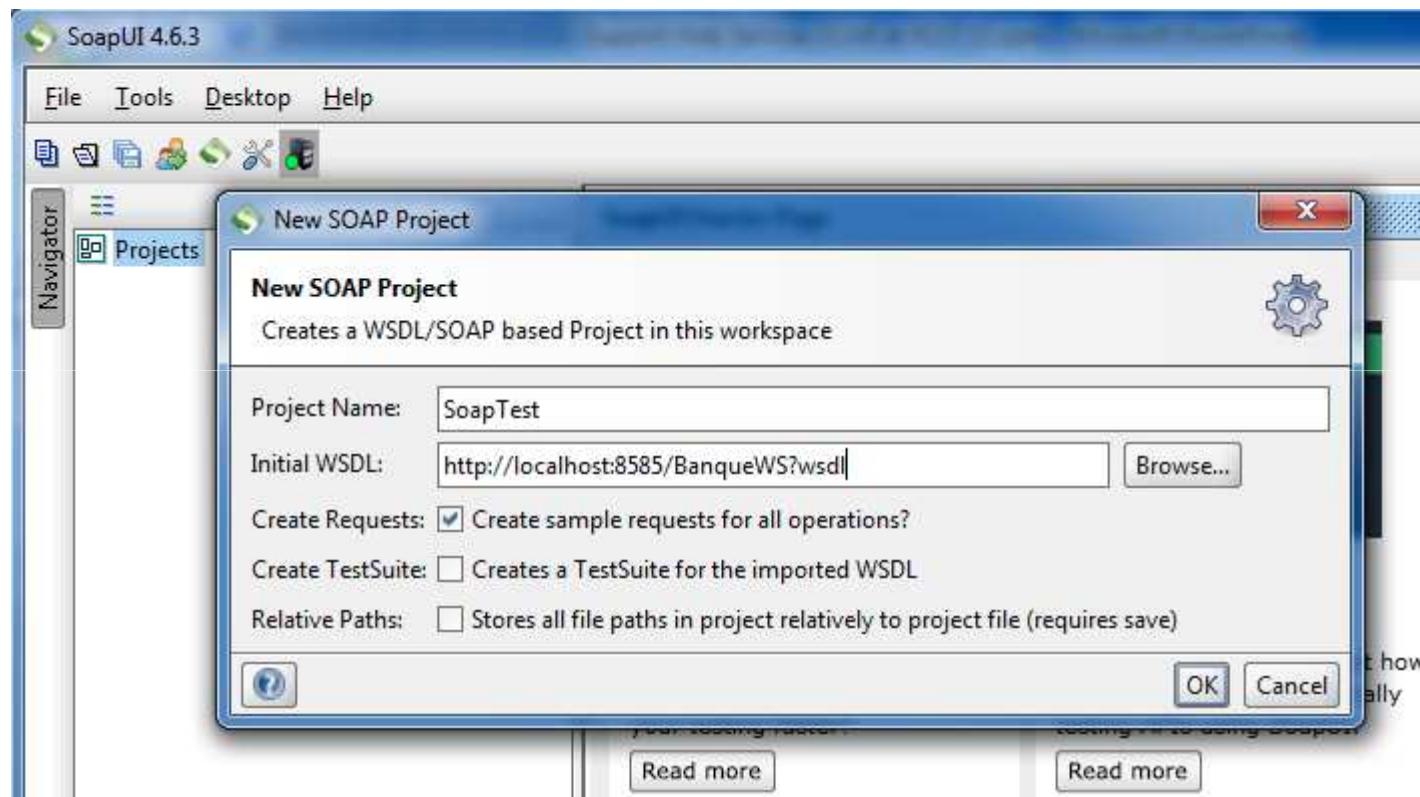
```
<binding name="BanqueServicePortBinding" type="tns:BanqueService">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"></soap:binding>
    <operation name="getCompte">
        <soap:operation soapAction=""></soap:operation>
        <input>
            <soap:body use="literal"></soap:body>
        </input>
        <output>
            <soap:body use="literal"></soap:body>
        </output>
    </operation>
    <operation name="getComptes">
        ....
    </operation>
</binding>
```

Elément service

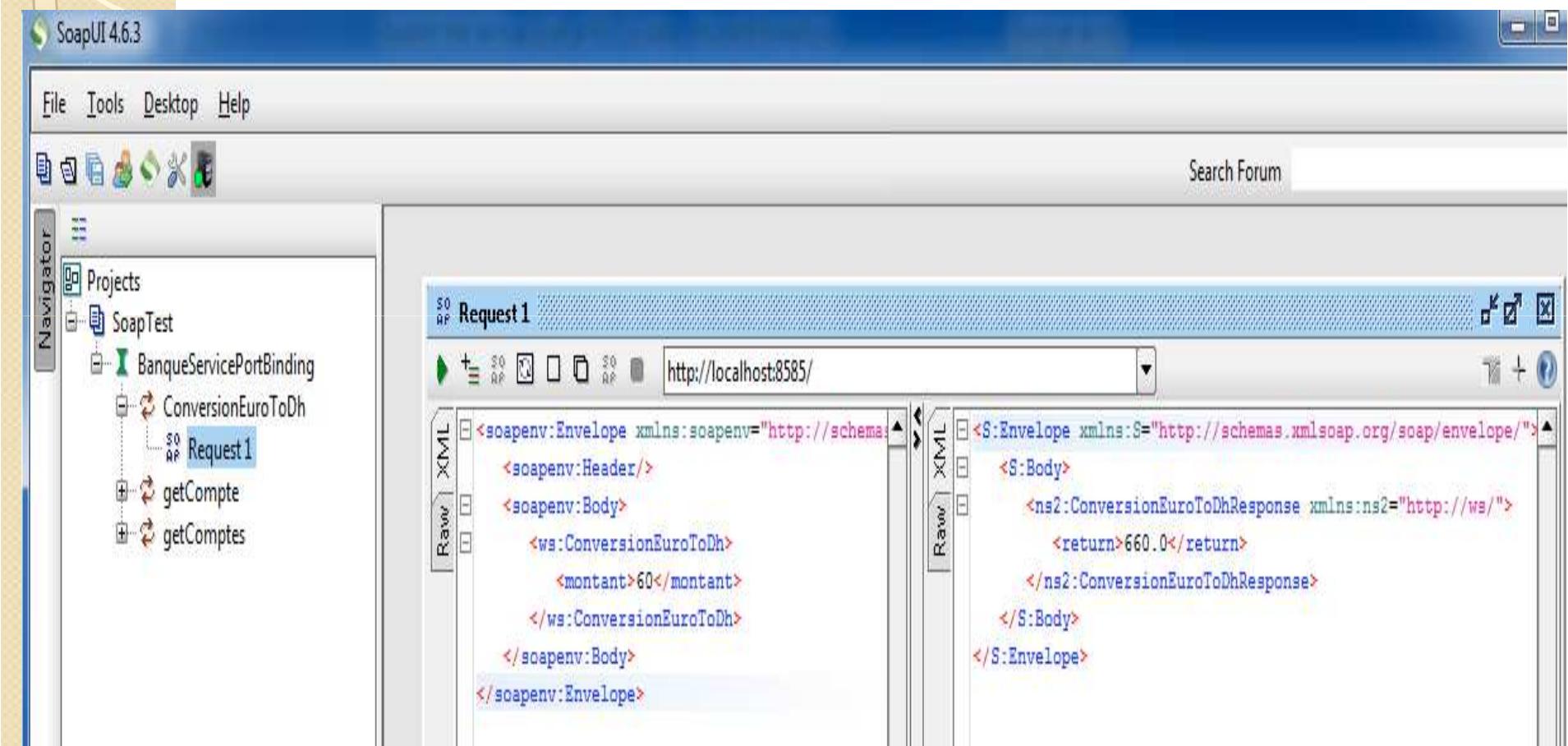


```
<service name="BanqueWS">
  <port name="BanqueServicePort" binding="tns:BanqueServicePortBinding">
    <soap:address location="http://localhost:8585/"></soap:address>
  </port>
</service>
```

Tester les méthodes du web service avec un analyseur SOAP : SoapUI



Tester les méthodes du web service avec un analyseur SOAP : SoapUI



Tester la méthode conversion

Requête SOAP

- <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ws="http://ws/">
 <soapenv:Header/>
 <soapenv:Body>
 <ws:ConversionEuroToDh>
 <montant>60</montant>
 </ws:ConversionEuroToDh>
 </soapenv:Body>
</soapenv:Envelope>

Réponse SOAP

- <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
 <ns2:ConversionEuroToDhResponse xmlns:ns2="http://ws/">
 <return>660.0</return>
 </ns2:ConversionEuroToDhResponse>
 </S:Body>
</S:Envelope>

Tester la méthode getCompte

Requête SOAP

- <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ws="http://ws/">
 <soapenv:Header/>
 <soapenv:Body>
 <ws:getCompte>
 <code>2</code>
 </ws:getCompte>
 </soapenv:Body>
</soapenv:Envelope>

Réponse SOAP

- <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
 <ns2:getCompteResponse xmlns:ns2="http://ws/">
 <return>
 <code>2</code>
 <solde>7000.0</solde>
 </return>
 </ns2:getCompteResponse>
 </S:Body>
</S:Envelope>

Tester la méthode getComptes

Requête SOAP

- <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ws="http://ws/">
 <soapenv:Header/>
 <soapenv:Body>
 <ws:getComptes/>
 </soapenv:Body>
</soapenv:Envelope>

Réponse SOAP

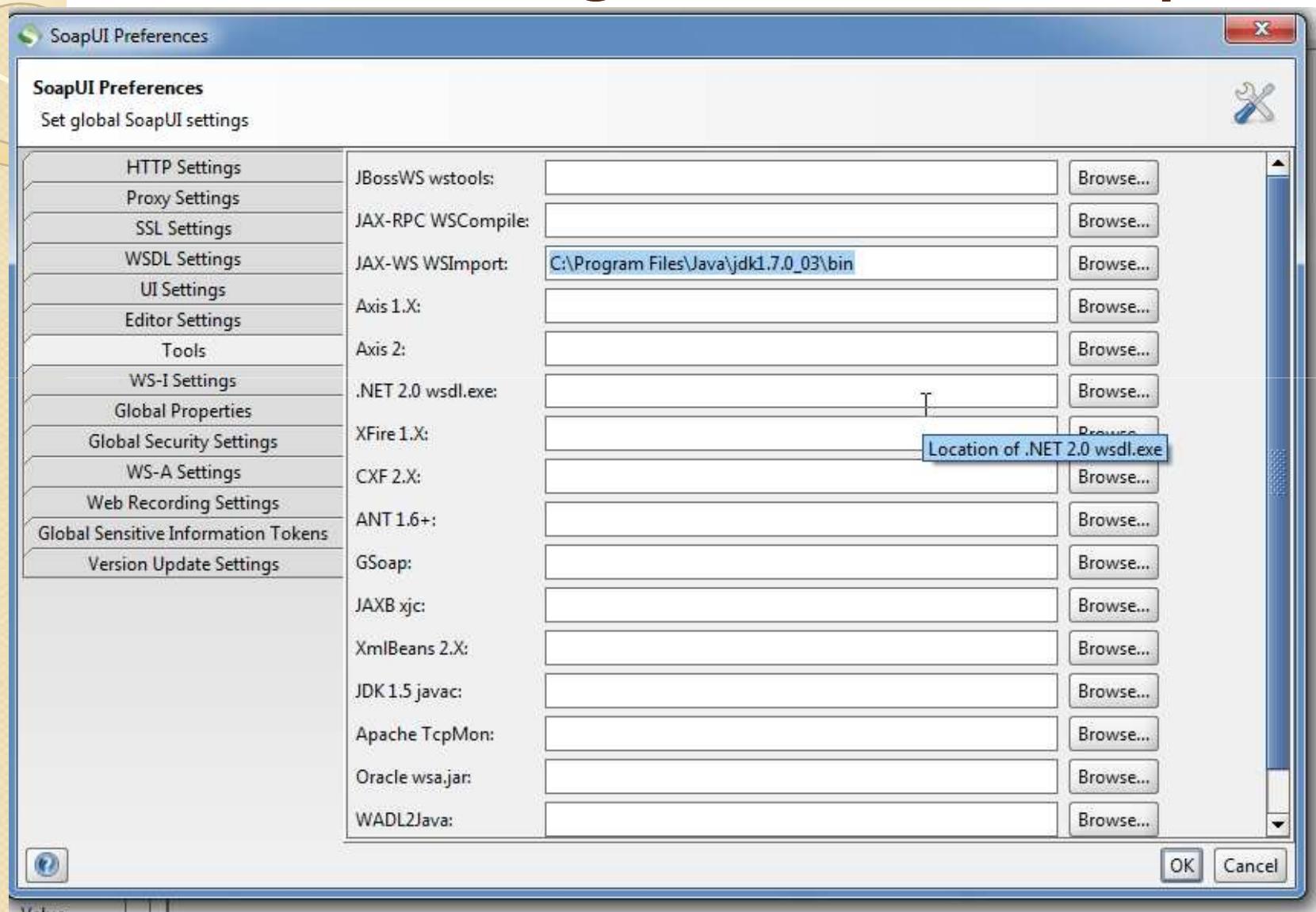
- <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
 <ns2:getComptesResponse xmlns:ns2="http://ws/">
 <return>
 <code>1</code>
 <solde>7000.0</solde>
 </return>
 <return>
 <code>2</code>
 <solde>7000.0</solde>
 </return>
 </ns2:getComptesResponse>
 </S:Body>
</S:Envelope>



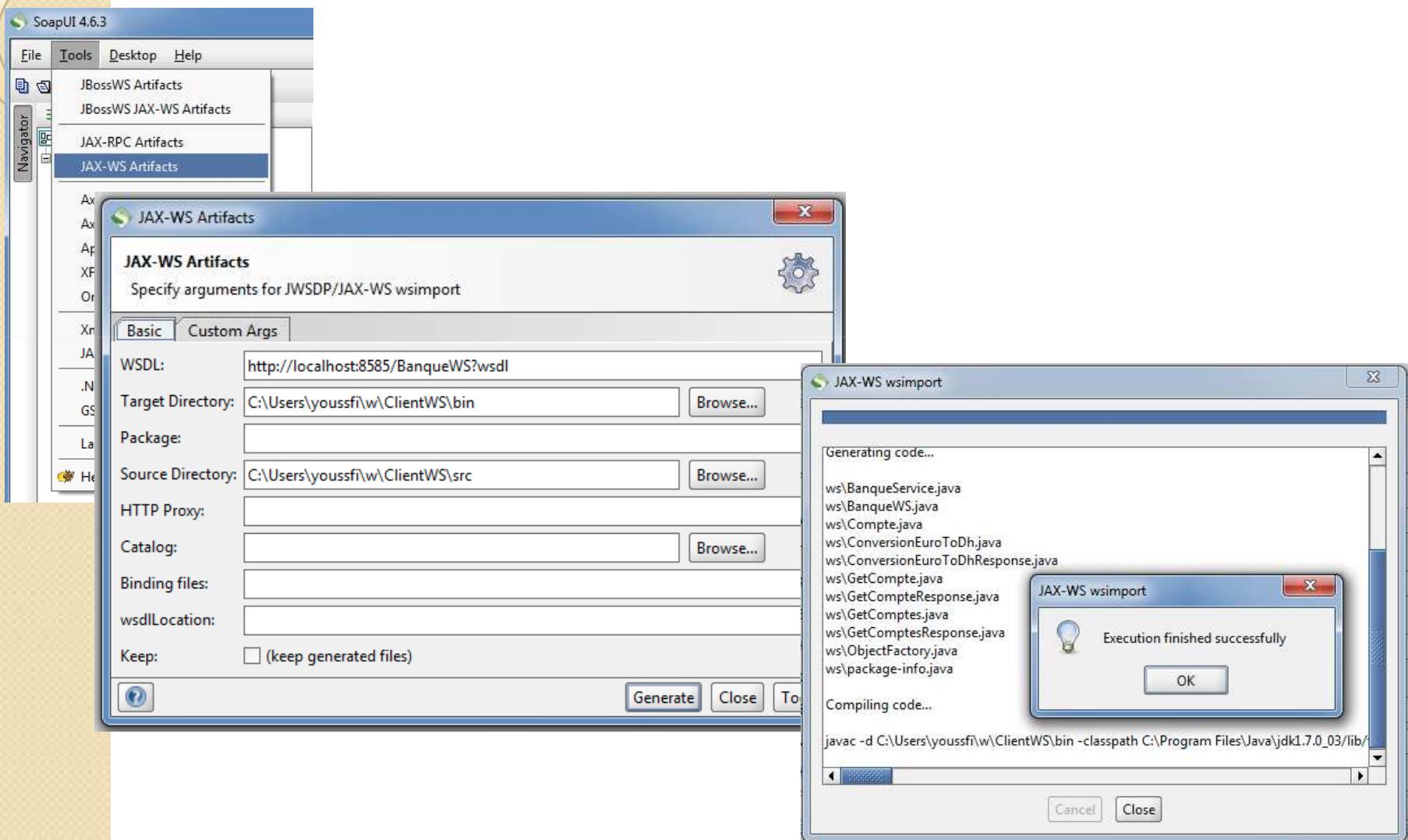
Client Java

- Créer un nouveau projet Java
- Générer un proxy
 - SoapUI est l'un des outils qui peuvent être utilisés pour générer les artefacts client en utilisant différents Framework (Axis, CXF, JaxWS, etc...)
 - Le JDK fournit une commande simple qui permet de générer un STUB JaxWS pour l'accès à un web service. Cette commande s'appelle **wsimport**.
 - SoapUI a besoin de savoir le chemin de cette commande
 - Avec la commande File > Preferences > Tools , vous pouvez configurer ce chemin comme le montre la figure suivante :

Préférence générale de SoapUI

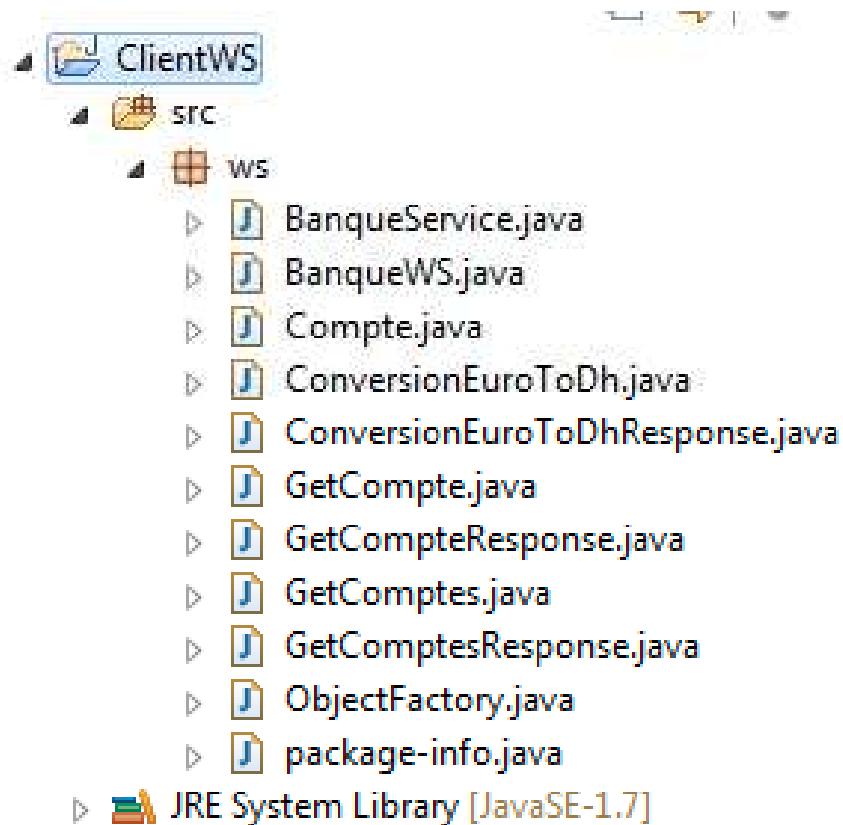


Générer le STUB JaxWS



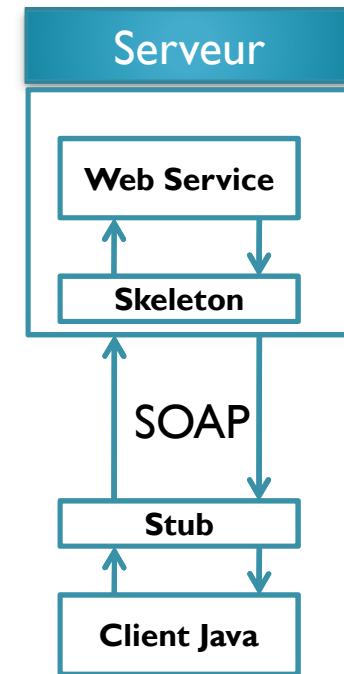
med@youssfi.net

Fichiers Générés



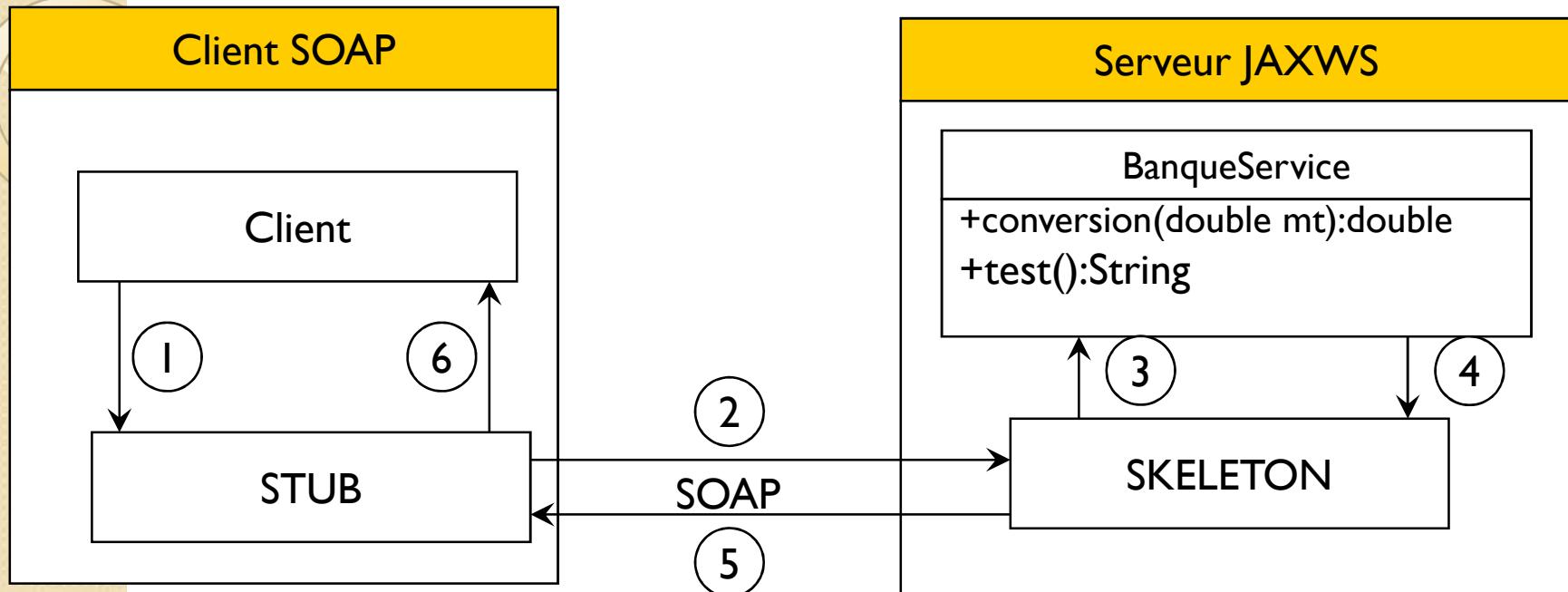
Client Java

```
import java.util.List;
import ws.BanqueService;
import ws.BanqueWS;
import ws.Compte;
public class ClientWS {
public static void main(String[] args) {
    BanqueService stub=new BanqueWS().getBanqueServicePort();
    System.out.println("Conversion");
    System.out.println(stub.conversionEuroToDh(9000));
    System.out.println("Consulter un compte");
    Compte cp=stub.getCompte(2L);
    System.out.println("Solde="+cp.getSolde());
    System.out.println("Liste des comptes");
    List<Compte> cptes=stub.getComptes();
    for(Compte c:cptes){
        System.out.println(c.getCode()+"----"+c.getSolde());
    }
}
}
```



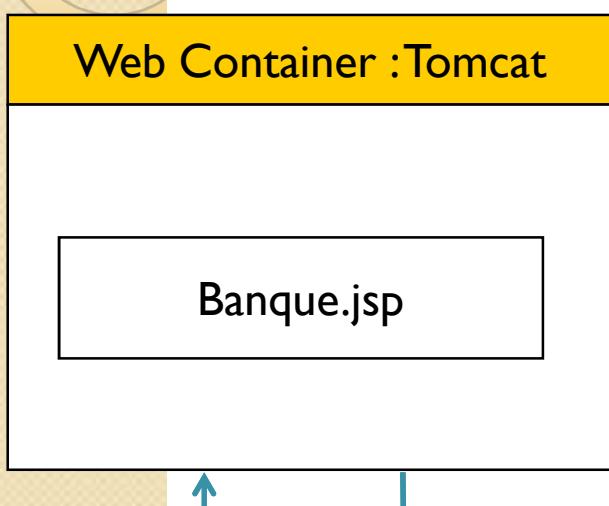
```
Problems @ Javadoc Declaration Console
<terminated> ClientWS (1) [Java Application] C:\Program Files\Java\...
Conversion
99000.0
Consulter un compte
Solde=7000.0
Liste des comptes
1----7000.0
2----7000.0
```

Architecture



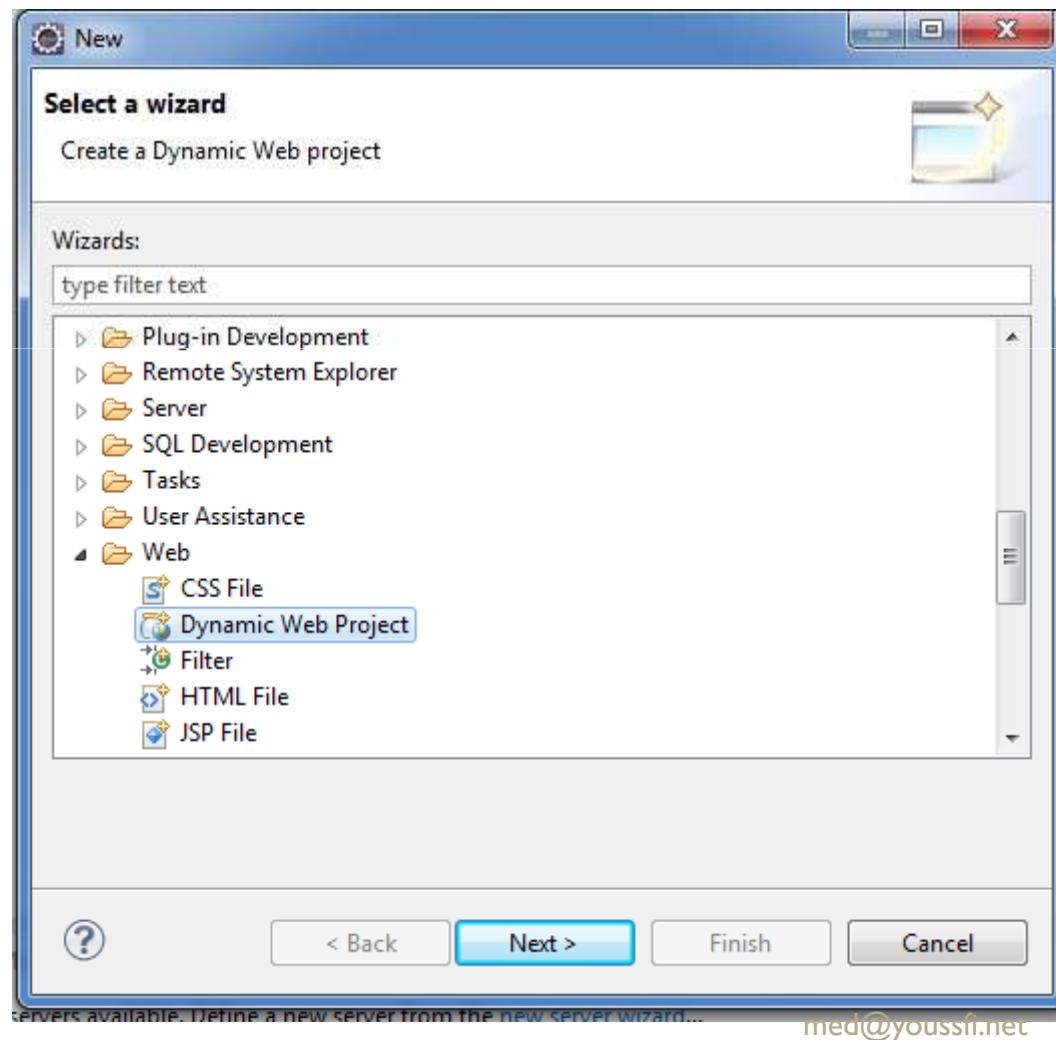
- 1 Le client demande au stub de faire appel à la méthode conversion(12)
- 2 Le Stub se connecte au Skeleton et lui envoie une requête SOAP
- 3 Le Skeleton fait appel à la méthode du web service
- 4 Le web service retourne le résultat au Skeleton
- 5 Le Skeleton envoie le résultat dans une la réponse SOAP au Stub
- 6 Le Stub fournie le résultat au client

Client JSP

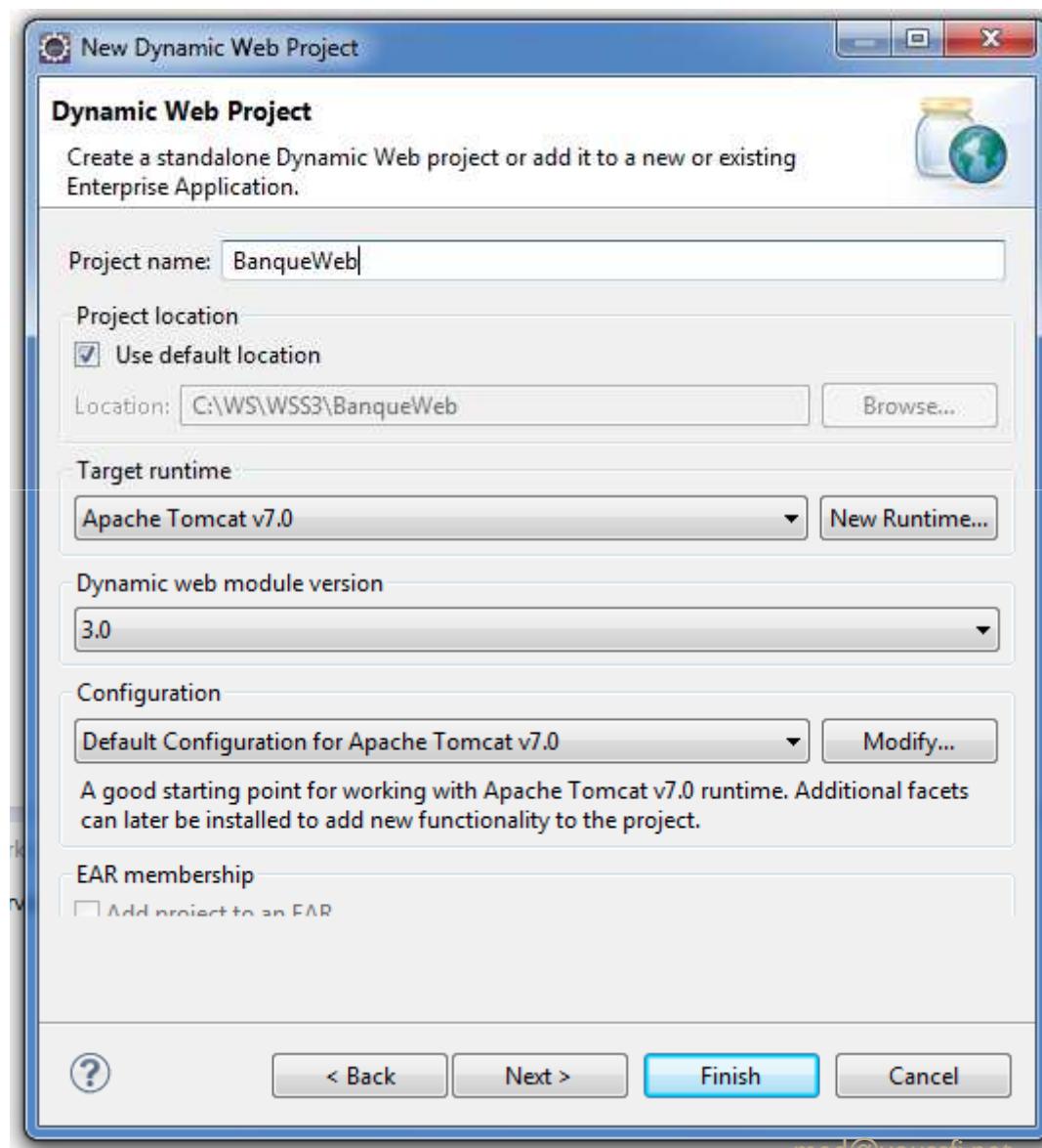


Création d'un projet Web Dynamique basé sur Tomcat 7

- Créer un projet Web Dynamique basé sur Tomcat 7

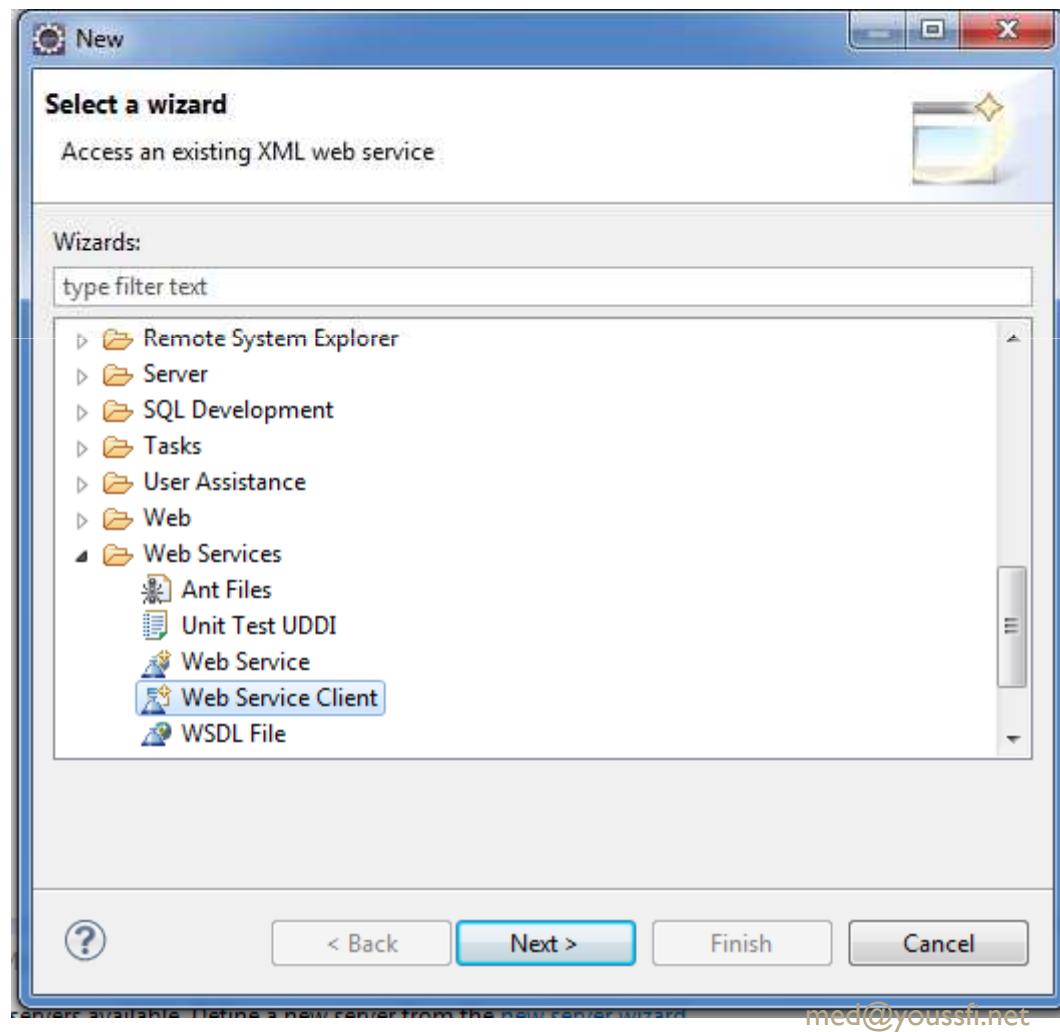


Projet Web Dynamique



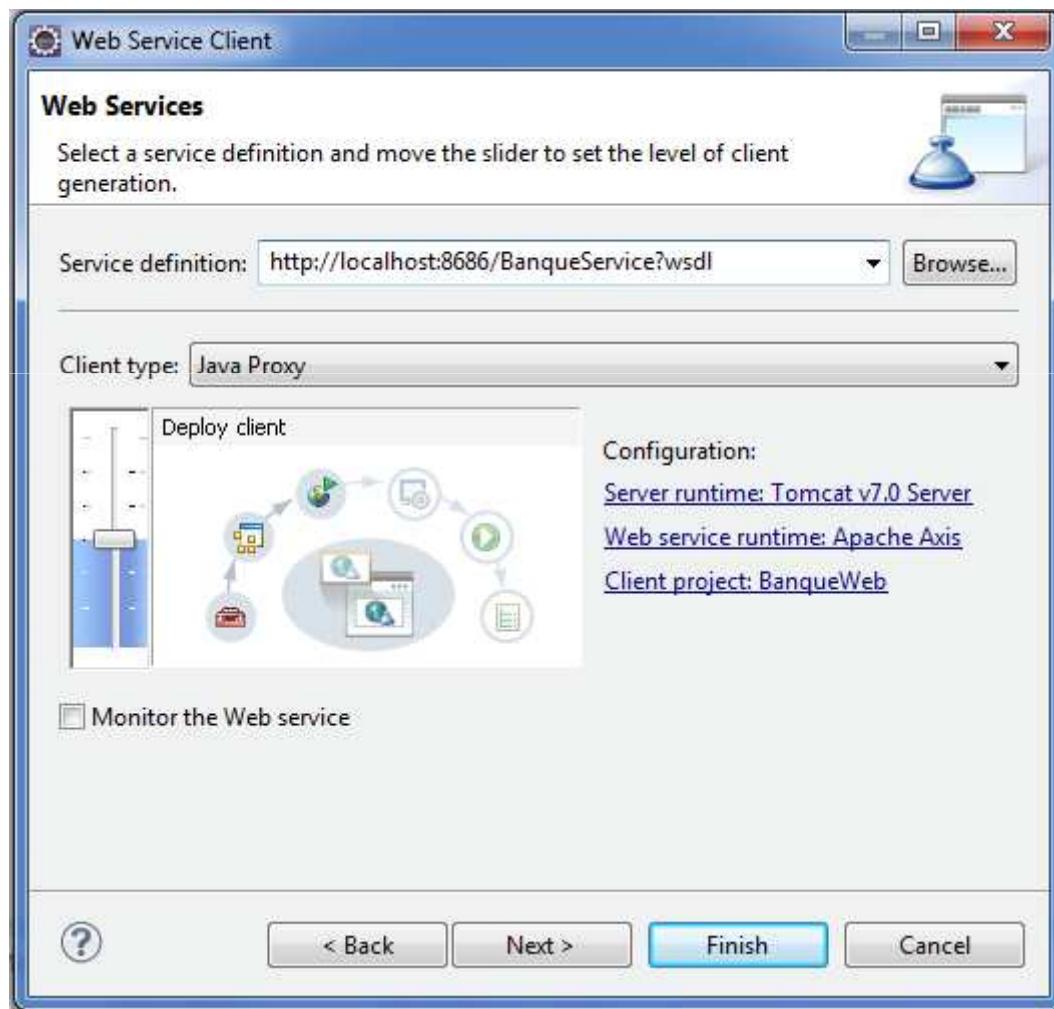
Générer un proxy (Stub) pour le web Service

- Fichier > Nouveau > Web Service Client



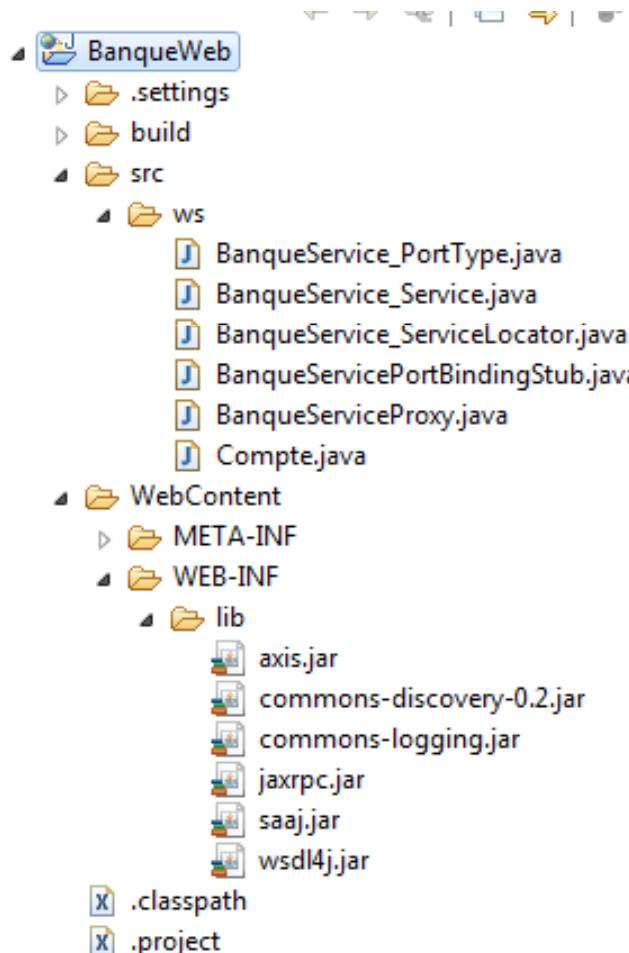
Génération du proxy

- Générer le proxy à partir du WSDL

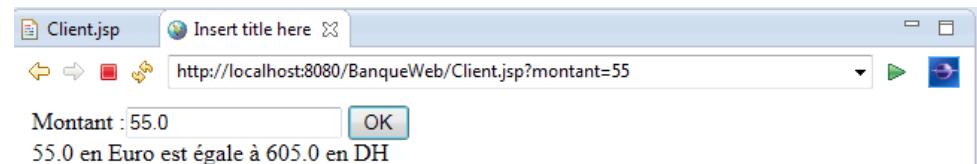


Fichier Générés

- Le proxy généré est basé sur AXIS



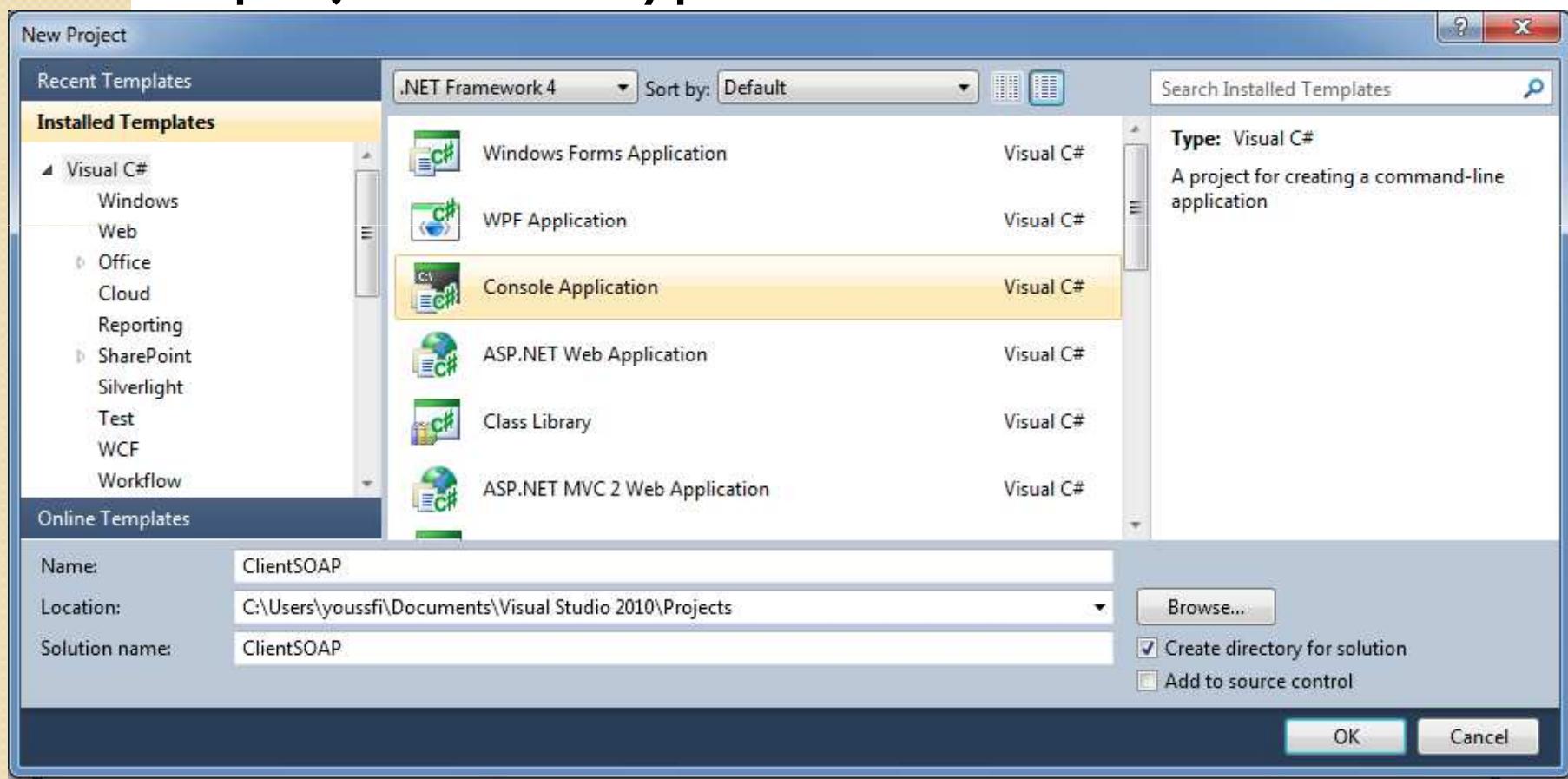
Client JSP



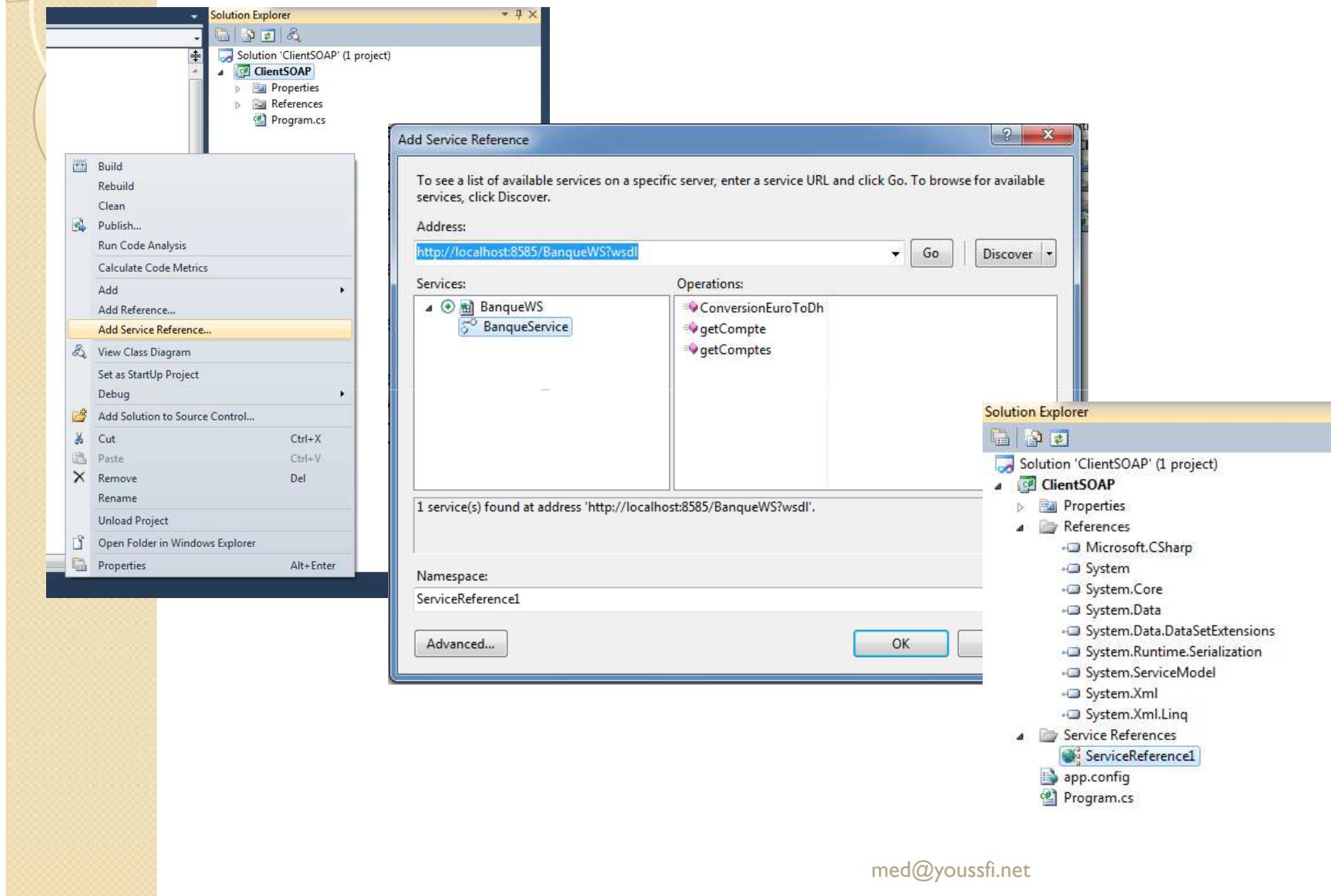
```
<%@page import="ws.BanqueServiceProxy"%>
<%
    double montant=0;  double resultat=0;
    if(request.getParameter("montant")!=null){
        montant=Double.parseDouble(request.getParameter("montant"));
        BanqueServiceProxy service=new BanqueServiceProxy();
        resultat=service.conversionEuroDH(montant);
    }
%>
<html><body>
<form action="Client.jsp">
    Montant :<input type="text" name="montant" value="<%=montant%>">
    <input type="submit" value="OK">
</form>
<%=montant %> en Euro est égale à <%=resultat %> en DH
</body></html>
```

Client SOAP avec .Net

- En utilisant Visual Studio (2010), Créer un projet C# de type Console

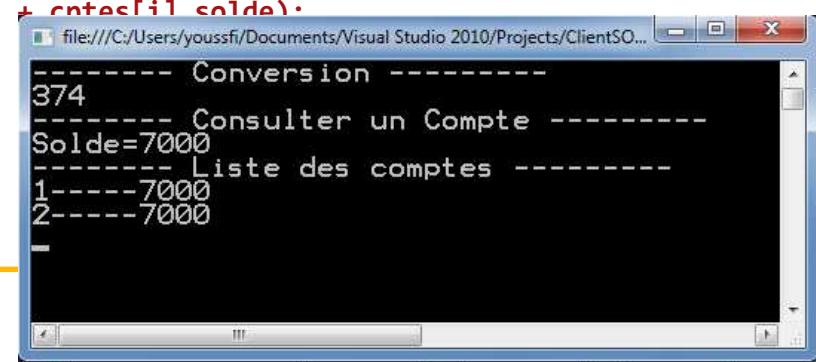
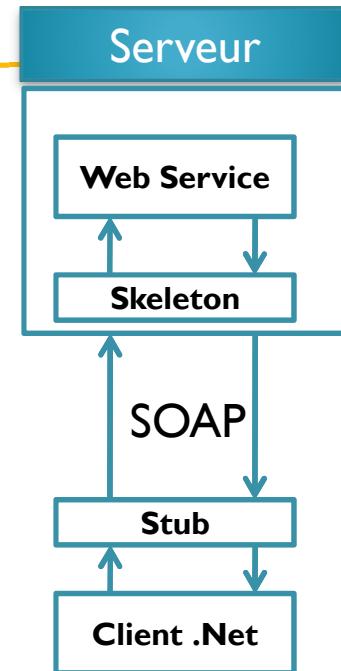


Générer Le Proxy client Soap



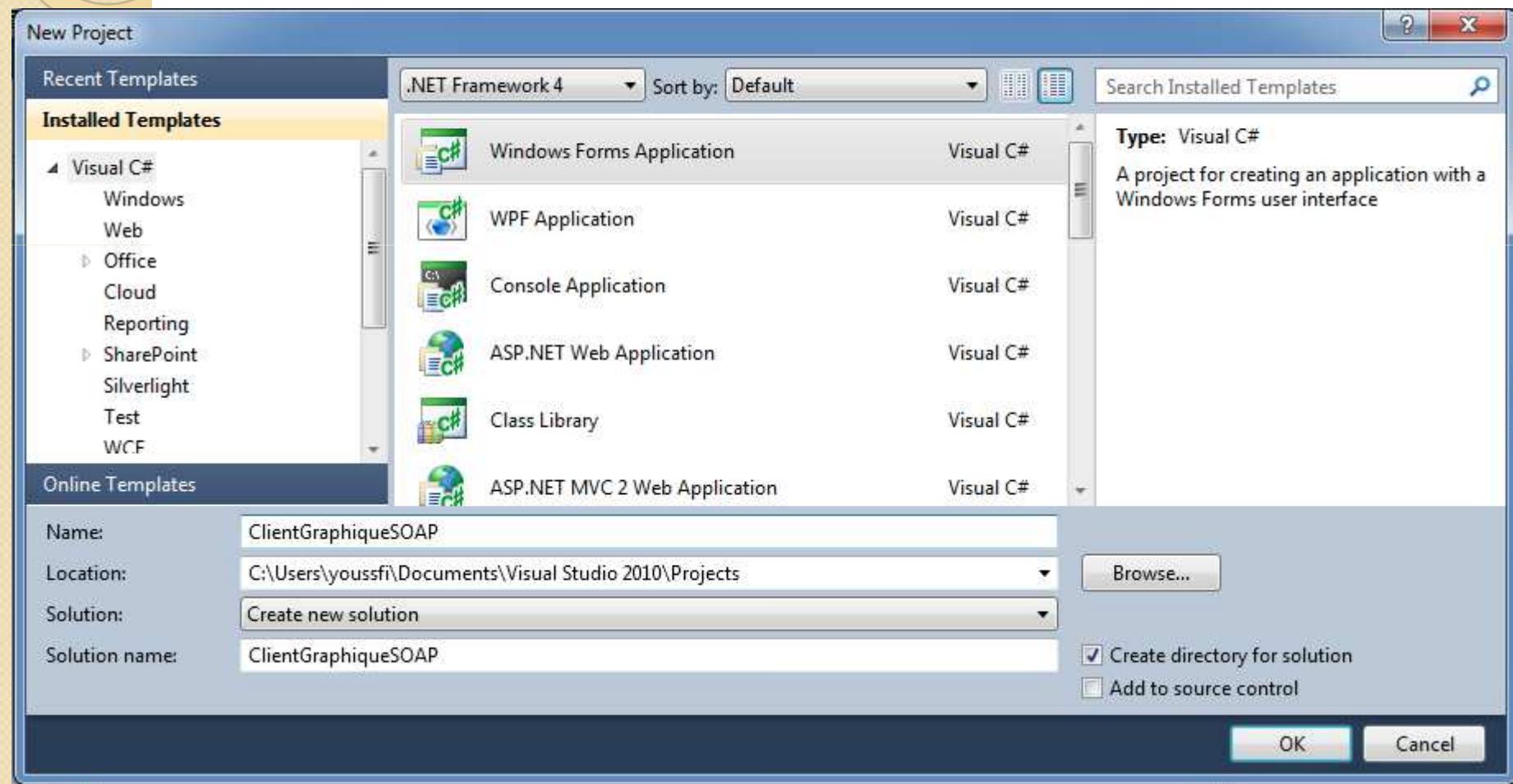
Code C# du client

```
using System;
namespace ClientSOAP
{
    class Program
    {
        static void Main(string[] args)
        {
            ServiceReference1.BanqueServiceClient stub =
                new ServiceReference1.BanqueServiceClient();
            Console.WriteLine("----- Conversion -----");
            Console.WriteLine(stub.ConversionEuroToDh(34));
            Console.WriteLine("----- Consulter un Compte -----");
            ServiceReference1.compte cp = stub.getCompte(2L);
            Console.WriteLine("Solde=" + cp.solde);
            Console.WriteLine("----- Liste des comptes -----");
            ServiceReference1.compte[] cptes = stub.getComptes();
            for (int i = 0; i < cptes.Length; i++)
            {
                Console.WriteLine(cptes[i].code + "----" + cptes[i].solde);
            }
            Console.ReadLine();
        }
    }
}
```

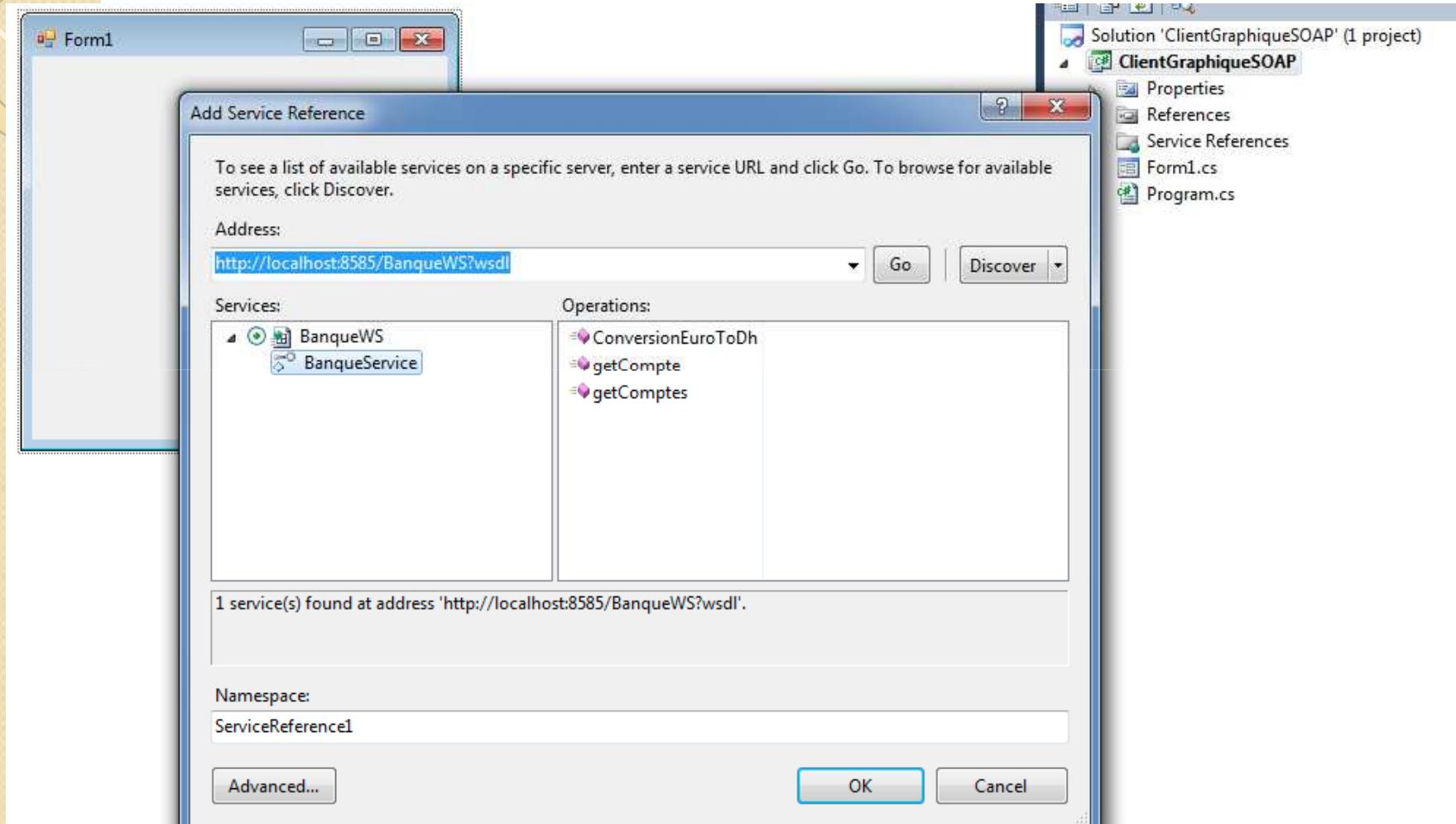


Client Graphique .Net

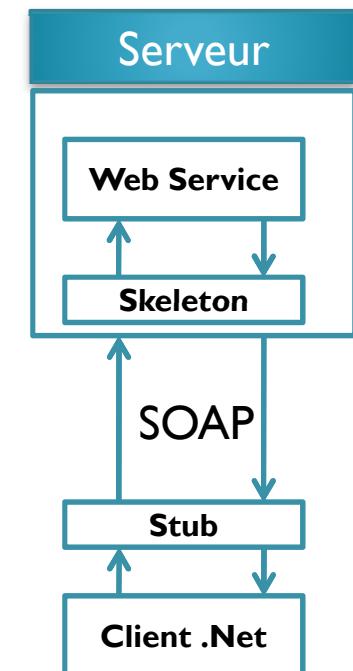
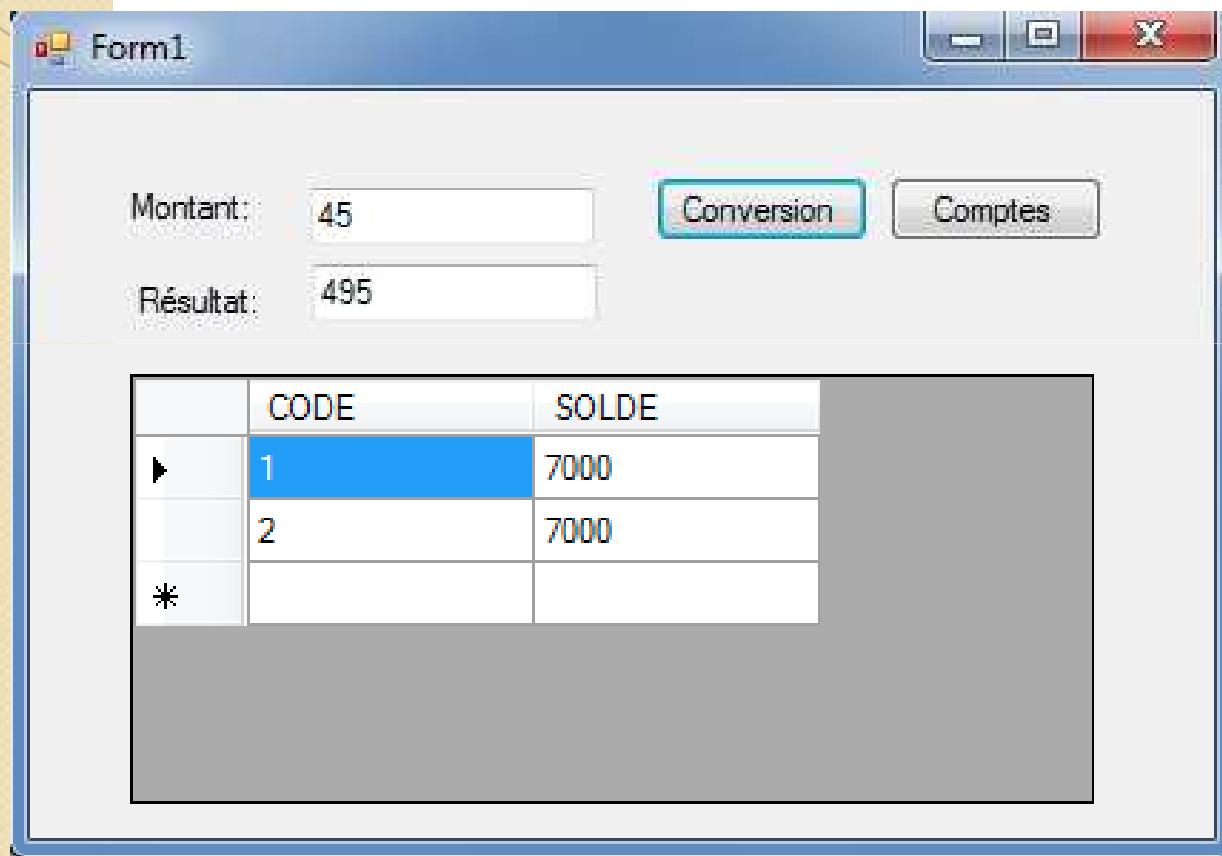
- Créer un nouveau Projet Visual Studio de type Windows Forms Application



Générer à nouveau le proxy SOAP

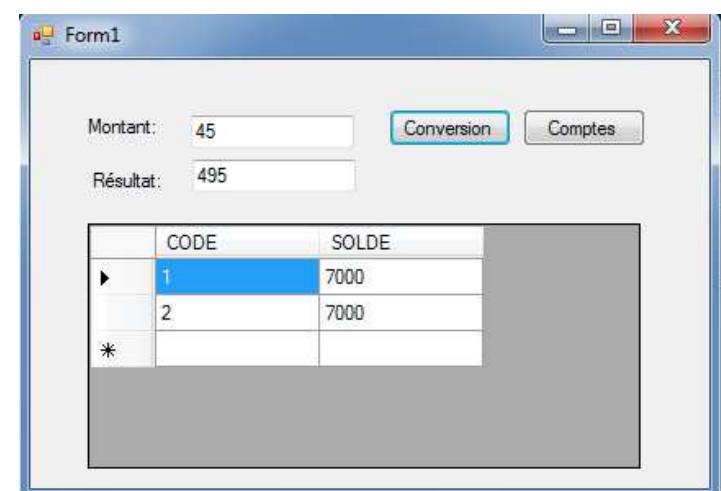
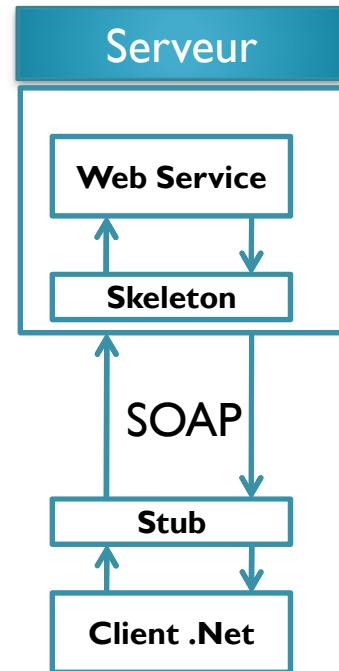


Dessiner les composants graphique de l'interface

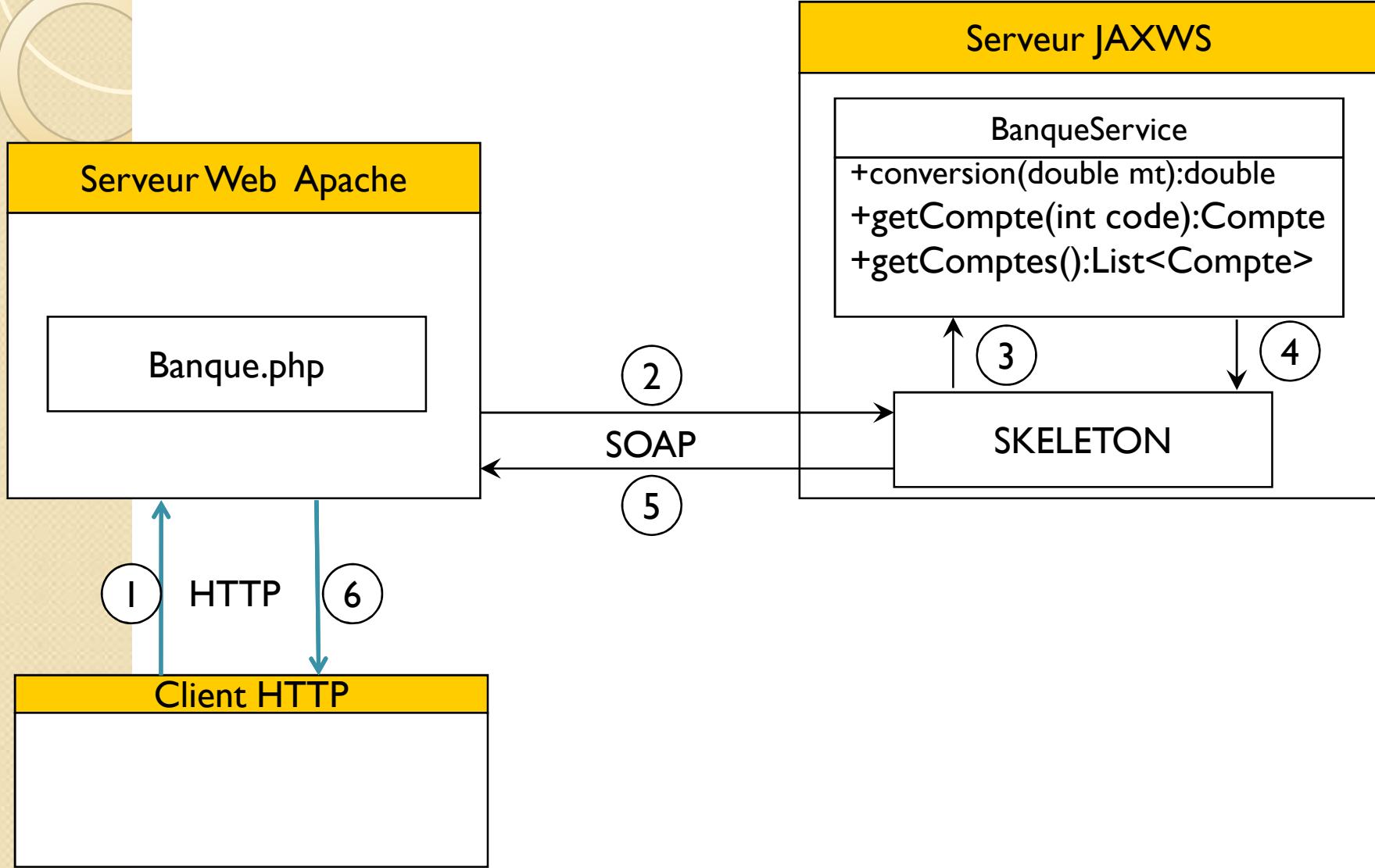


Code C#

```
using System;using System.Collections.Generic; using System.ComponentModel;using System.Data;
using System.Drawing;using System.Linq; using System.Text;using System.Windows.Forms;
namespace ClientGraphiqueSOAP {
    public partial class Form1 : Form      {
        private ServiceReference1.BanqueServiceClient stub;
        public Form1()  {
            InitializeComponent();stub = new ServiceReference1.BanqueServiceClient();
        }
        private void button1_Click(object sender, EventArgs e)      {
            double mt = Double.Parse(textBox1.Text);
            double res = stub.ConversionEuroToDh(mt);
            textBox2.Text = res.ToString();
        }
        private void button2_Click(object sender, EventArgs e) {
            ServiceReference1.compte[] cptes = stub.getComptes();
            DataTable dt = new DataTable("comptes");
            dt.Columns.Add("CODE");
            dt.Columns.Add("SOLDE");
            for (int i = 0; i < cptes.Length; i++)  {
                dt.Rows.Add(cptes[i].code, cptes[i].solde);
            }
            dataGridView1.DataSource = dt;
        }
    }
}
```



Web Service Java et Client PHP

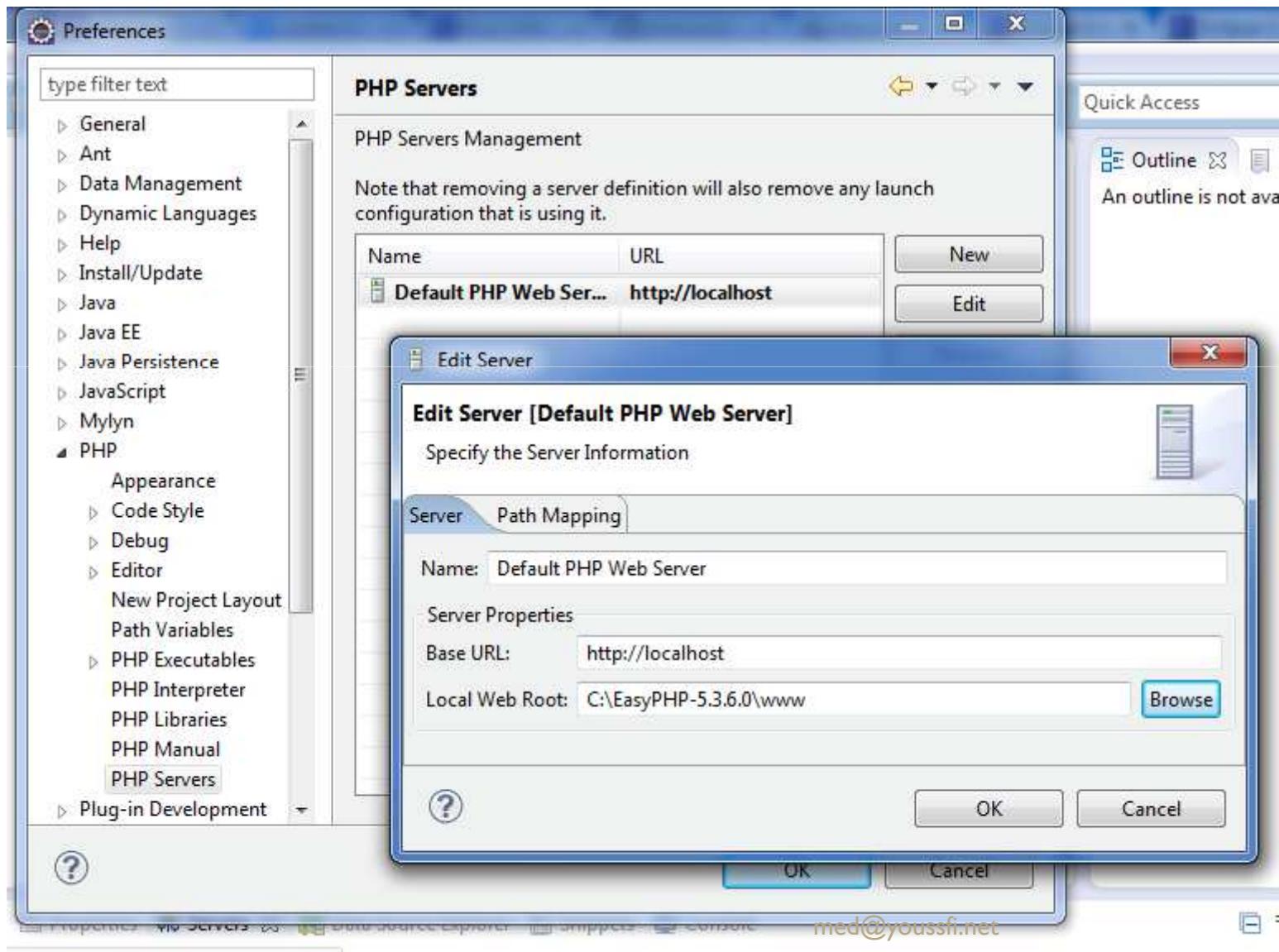


Exemple de Client SOAP PHP

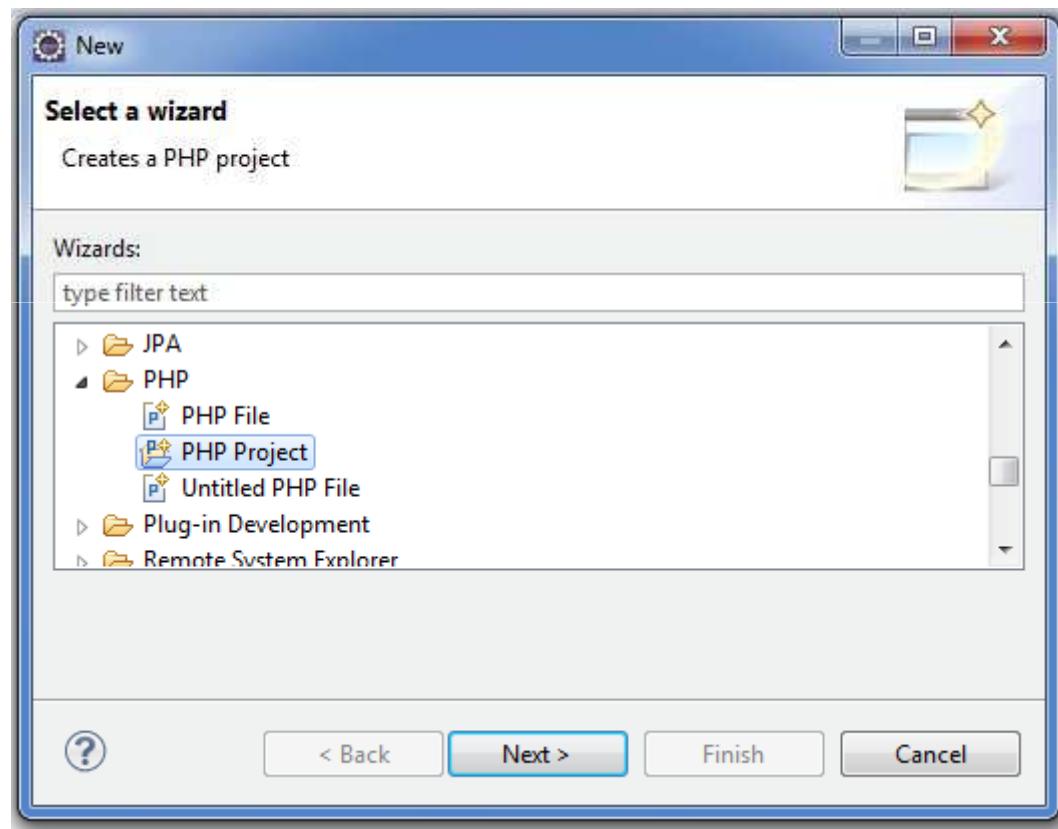
```
<?php
$client = new SoapClient('http://localhost:8585/BanqueWS?wsdl');
$param=new stdClass();
$param->montant=23;
$res=$client->__soapCall("conversionEuroDH",array($param));
//var_dump($res);
echo($res->return);
$param2=new stdClass();
$param2->arg0=2;
$res2=$client->__soapCall("getCompte",array($param2));
//var_dump($res2);
echo("Code=". $res2->return->code);
echo("<br/>Solde=". $res2->return->solde);
$res3=$client->__soapCall("getComptes",array());
//var_dump($res3);
echo ("<hr/>");
foreach($res3->return as $cpt){
    echo("Code=". $cpt->code);
    echo("<br/>Solde=". $cpt->solde);
    echo("<br/>");
}
?>
```

Configuration du plugin Eclipse PDT

- Après avoir installé Easy PHP ou Wamp Server (Apage, MySQL, PHP, PhpMyAdmin)
- Window > preferences

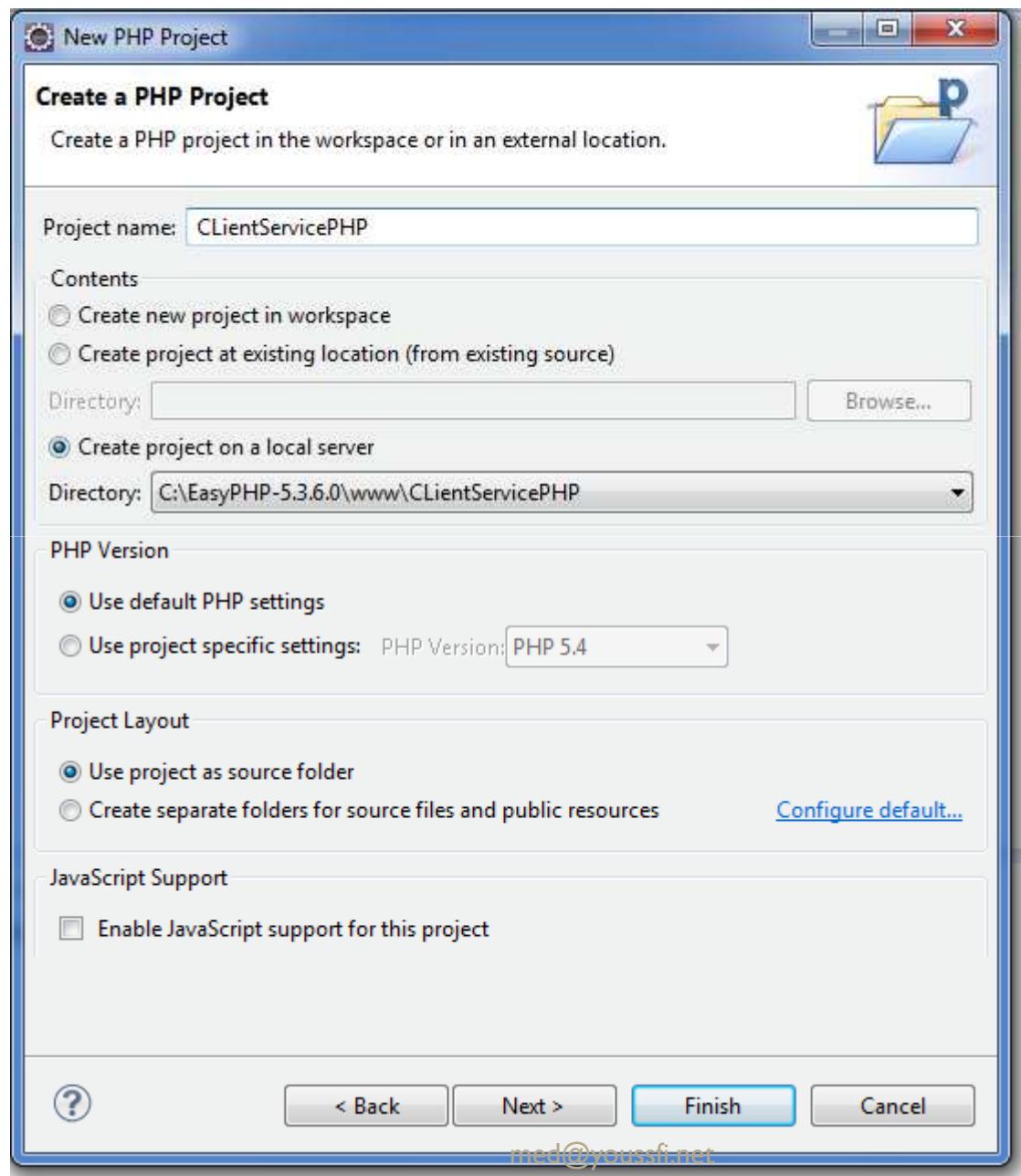


Projet PHP



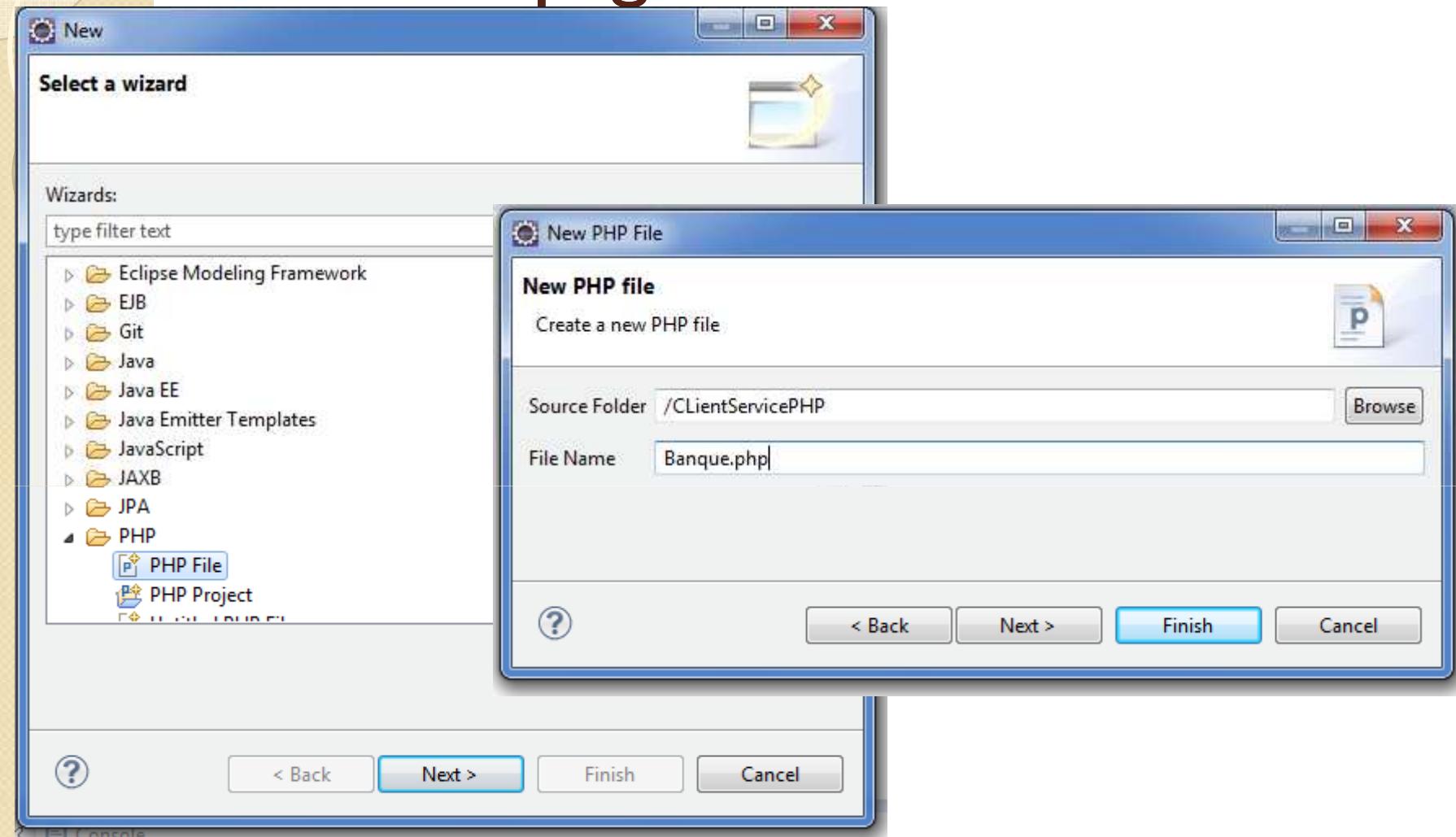
med@youssfi.net

Projet PHP

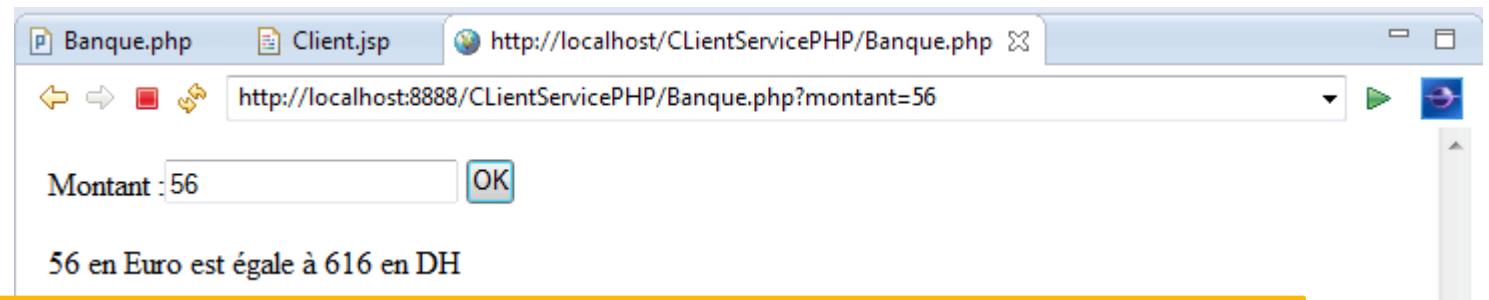


med@yousfi.net

Nouvelle page PHP



Code PHP



```
<?php
$montant=0;$resultat=0;
if (isset($_GET['montant'])){
$montant=$_GET['montant'];
$client = new SoapClient('http://localhost:8686/BanqueService?wsdl');
$param=new stdClass();
$param->montant=$montant;
$rep=$client->__soapCall("conversionEuroDH",array($param));
$resultat=$rep->return;
}
?>
<html>
<body>
<form action="Banque.php">
Montant :<input type="text" name="montant" value="<?php
echo($montant)?>">
<input type="submit" value="OK">
</form>
<?php echo($montant)?> en Euro est égale à <?php echo($resultat)?> en DH
</body>
</html>
```

Un autre exemple Client SOAP PHP

```
<?php
$mt=0;
if(isset($_POST['action'])){
    $action=$_POST['action'];
    if($action=="OK"){
        $mt=$_POST['montant'];
        $client=new SoapClient("http://localhost:8585/BanqueWS?wsdl");
        $param=new stdClass();
        $param->montant=$mt;
        $rep=$client->__soapCall("conversionEuroToDh",array($param));
        $res=$rep->return;
    }
    elseif($action=="listComptes"){
        $client=new SoapClient("http://localhost:8585/BanqueWS?wsdl");
        $res2=$client->__soapCall("getComptes",array());
    }
}
?>
```

localhost/ClientPHP/clientSOAP.php

Montant: 0 OK listComptes

Rsultat:

CODE	SOLDE
1	7000
2	7000

Suite de l'exemple du Client SOAP PHP

```
<html>
<body>
    <form method="post" action="clientSOAP.php">
        Montant:<input type="text" name="montant" value="<?php echo($mt)?>">
        <input type="submit" value="OK" name="action">
        <input type="submit" value="listComptes" name="action">
    </form>

    Rsltat:
    <?php if (isset($res)){
        echo($res);
    }
    ?>
    <?php if(isset($res2)){?>
        <table border="1" width="80%">
            <tr>
                <th>CODE</th><th>SOLDE</th>
            </tr>
            <?php foreach($res2->return as $cp) {?>
                <tr>
                    <td><?php echo($cp->code)?></td>
                    <td><?php echo($cp->solde)?></td>
                </tr>
            <?php }?>
        </table>
    <?php }?>
    </body>
</html>
```

localhost/ClientPHP/clientSOAP.php

Montant: 0

OK listComptes

Rsltat:

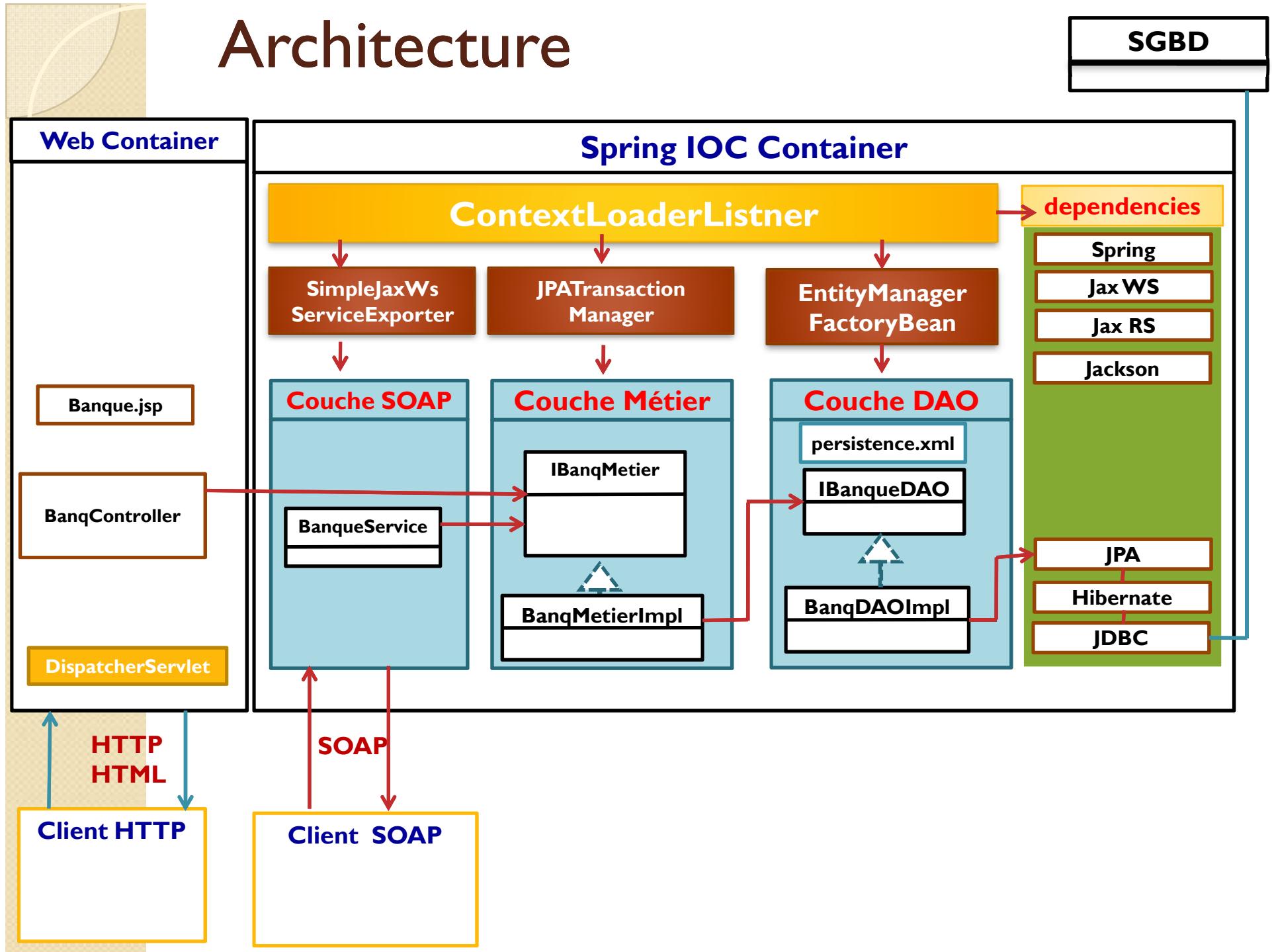
CODE	SOLDE
1	7000
2	7000



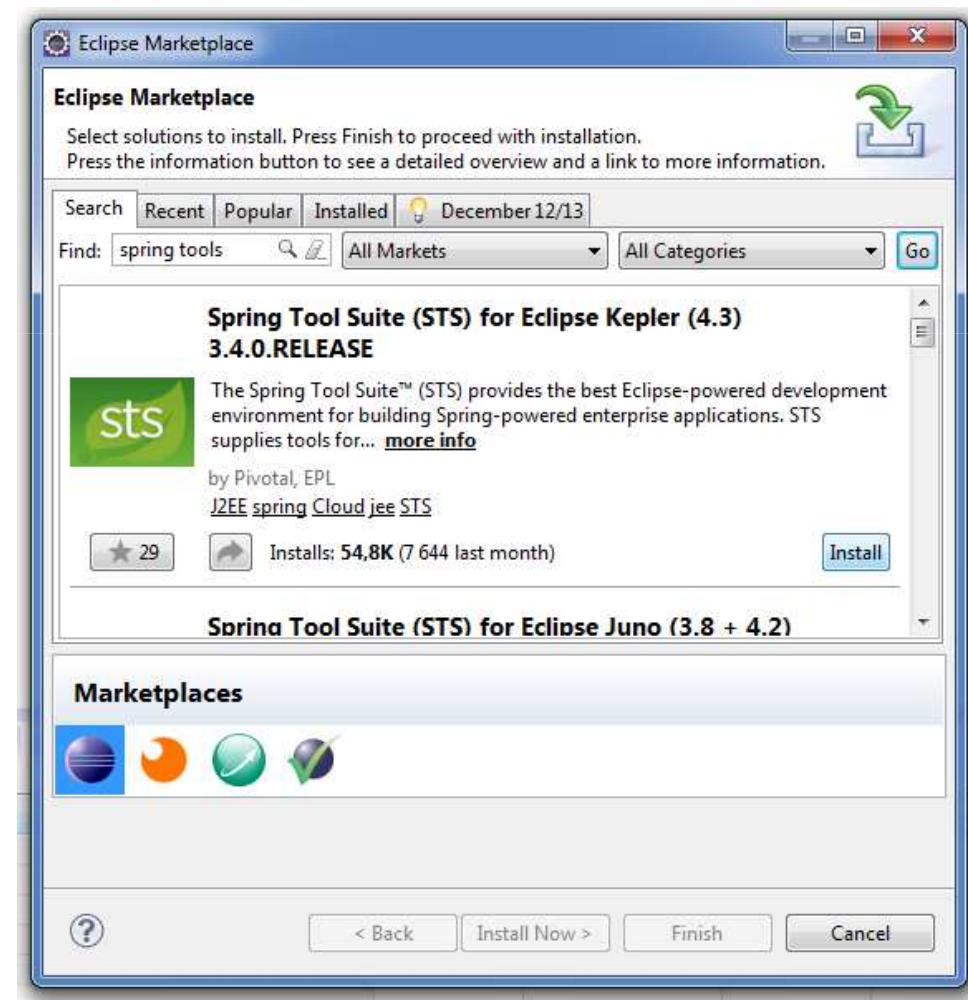
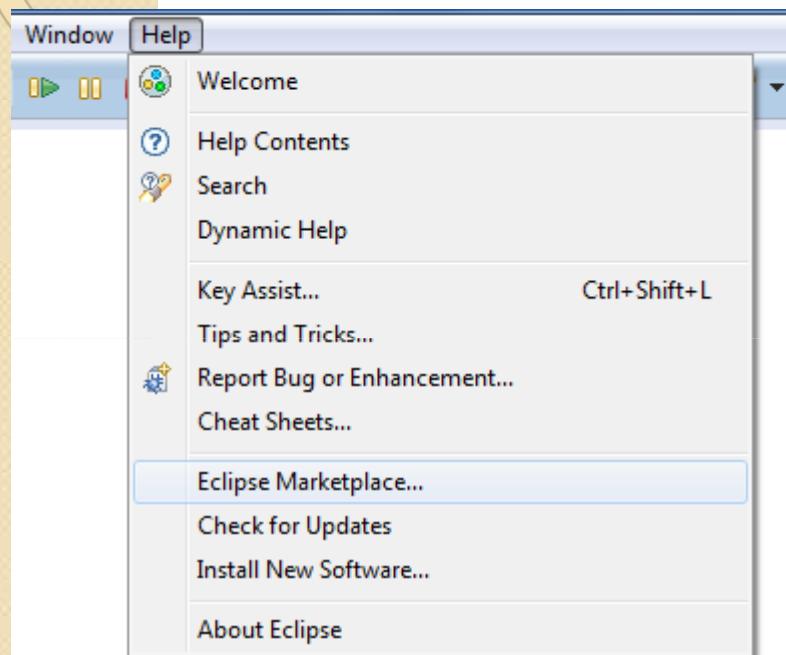
Web Service dans une application web J2EE basée sur Spring

- On souhaite créer une application web J2EE basée sur Spring qui permet de déployer un web service qui permet de :
 - Ajouter un compte
 - Consulter un compte
 - Consulter tous les comptes
 - Effectuer un versement d'un montant dans un compte
 - Effectuer un retrait d'un montant sur un compte
 - Effectuer un virement d'un montant d'un compte vers un autre
 - Supprimer un compte
- Chaque compte est défini par le code, le solde et la date création
- Les comptes seront stockés dans une base de données MySQL

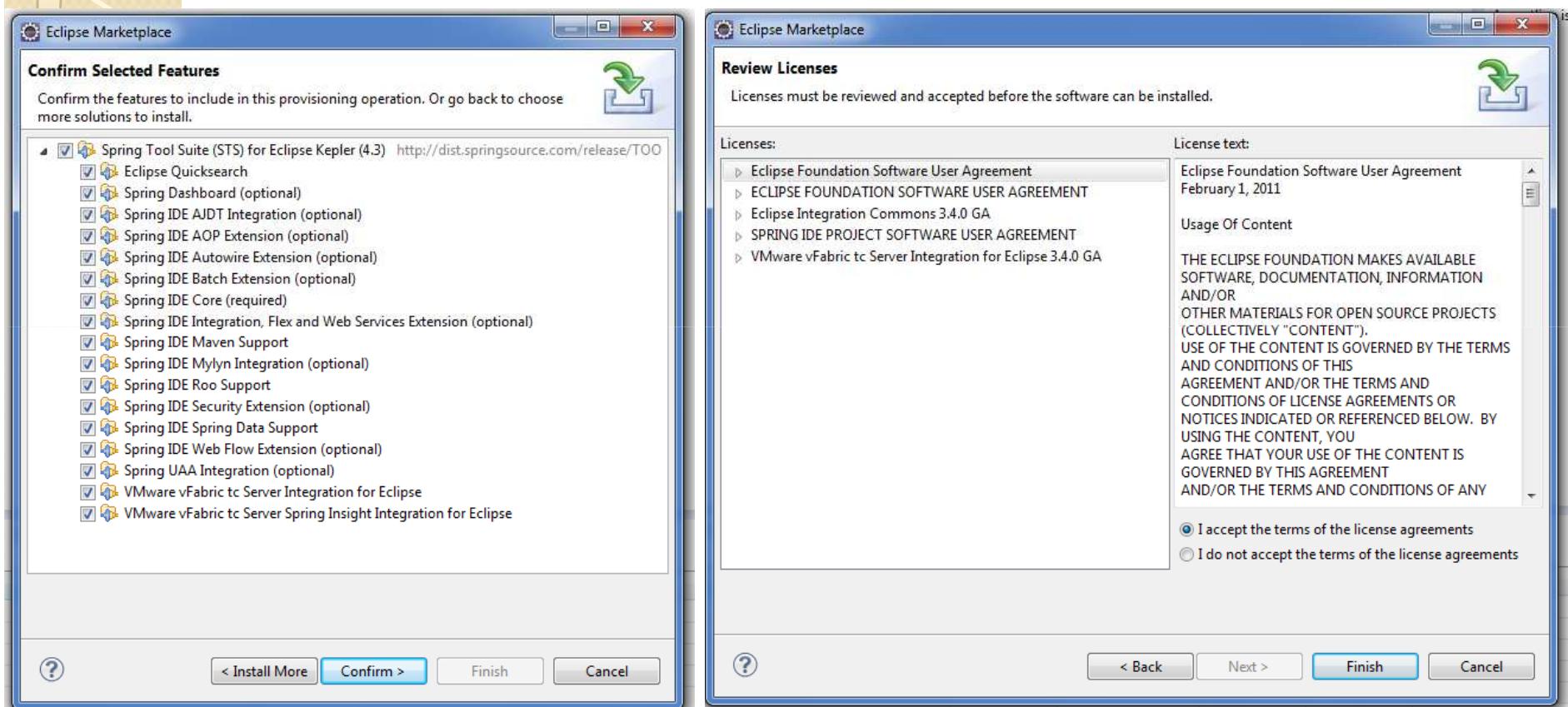
Architecture



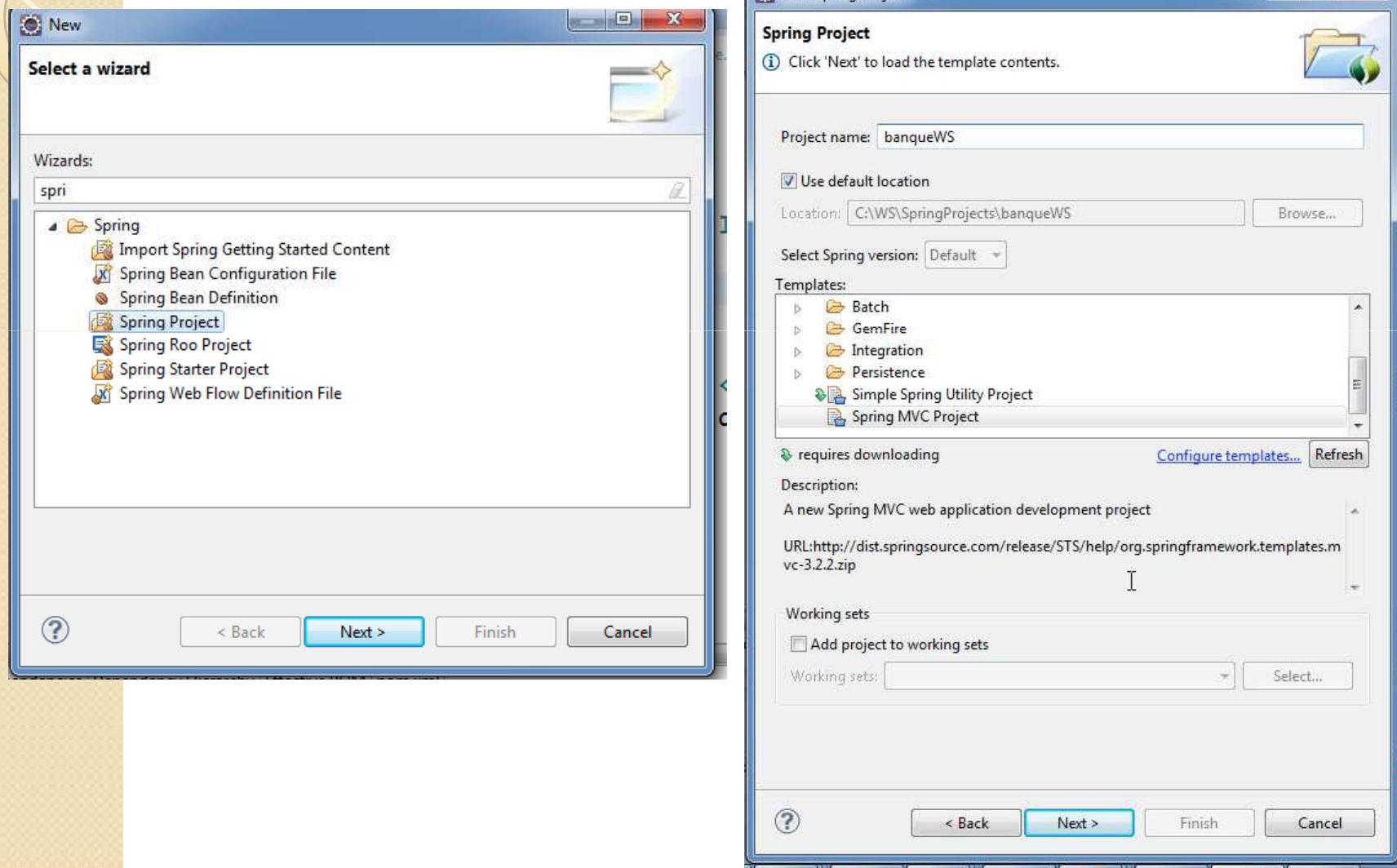
Installation du plugin : spring tools pour eclipse



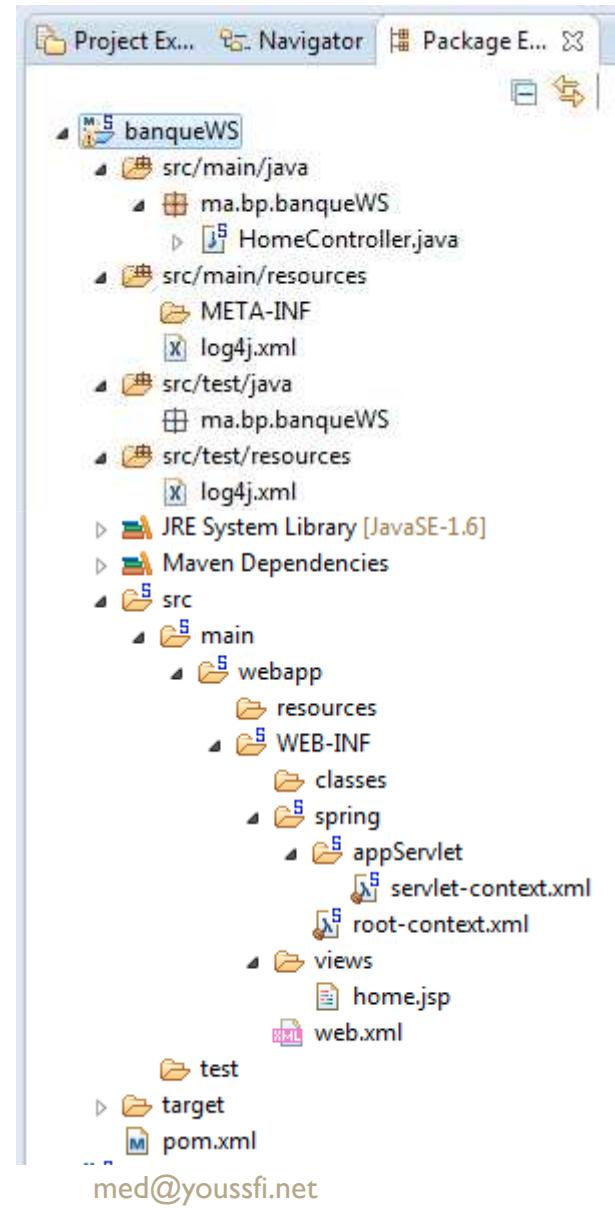
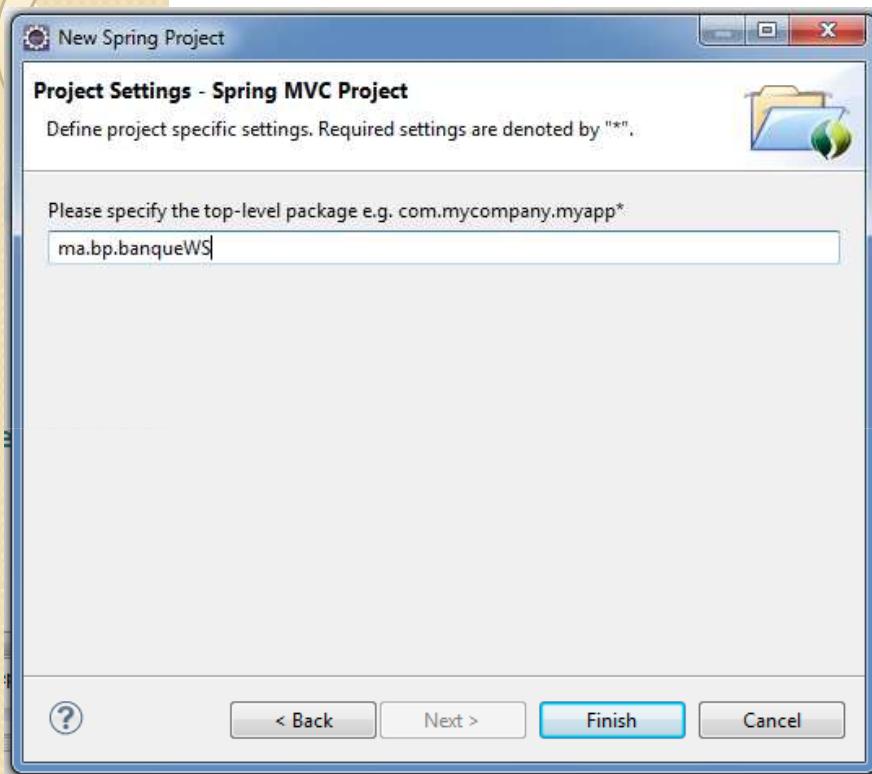
Installation du plugin : spring tools pour eclipse



Création d'un projet Spring



Création d'un projet Spring



web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
<!-- The definition of the Root Spring Container shared by all Servlets and
Filters --&gt;
&lt;context-param&gt;
    &lt;param-name&gt;contextConfigLocation&lt;/param-name&gt;
    &lt;param-value&gt;/WEB-INF/spring/root-context.xml&lt;/param-value&gt;
&lt;/context-param&gt;
<!-- Creates the Spring Container shared by all Servlets and Filters --&gt;
&lt;listener&gt;
&lt;listener-
    class&gt;org.springframework.web.context.ContextLoaderListener&lt;/listener-
    class&gt;
&lt;/listener&gt;</pre>
```

web.xml

```
<!-- Processes application requests -->
<servlet>
    <servlet-name>appServlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet
    </servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring/appServlet/servlet-context.xml
        </param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>appServlet</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
</web-app>
```



>/WEB-INF/spring/root-context.xml

- Ce fichier est lu par ContextLoaderListener, au démarrage du serveur .
- C'est un fichier dans lequel contexte de l'application sera construit
- ContextLoaderListener représente Spring IOC
- c'est donc un fichier pour l'injection des dépendances
- Pour le moment, il est vide

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-
    beans.xsd">

    <!-- Root Context: defines shared resources visible
        to all other web components -->

</beans>
```

>/WEB-INF/spring/appServlet/servlet-context.xml

- Ce fichier est lu par DispatcherServlet qui représente le contrôleur web de l'application

```
<?xml version="1.0" encoding="UTF-8"?>

<beans:beans xmlns="http://www.springframework.org/schema/mvc"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:beans="http://www.springframework.org/schema/beans"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd
        http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd">

    <!-- Enables the Spring MVC @Controller programming model -->
    <annotation-driven />

    <!-- Handles HTTP GET requests for /resources/** by efficiently serving up static
        resources in the ${webappRoot}/resources directory -->
    <resources mapping="/resources/**" location="/resources/" />

    <!-- Resolves views selected for rendering by @Controllers to .jsp resources in the
        /WEB-INF/views directory -->
    <beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <beans:property name="prefix" value="/WEB-INF/views/" />
        <beans:property name="suffix" value=".jsp" />
    </beans:bean>

    <context:component-scan base-package="ma.enset.myCataogue" />
</beans:beans>
```

Un exemple de contrôleur Spring MVC

```
package ma.enset.myCataogue;

import java.text.*;import java.util.*;import org.slf4j.*;import
    org.springframework.stereotype.Controller;

import org.springframework.ui.Model; import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
/** Handles requests for the application home page. */
@Controller
public class HomeController {

    private static final Logger Logger = LoggerFactory.getLogger(HomeController.class);
    /** Simply selects the home view to render by returning its name. */
    @RequestMapping(value = "/", method = RequestMethod.GET)
    public String home(Locale locale, Model model) {
        Logger.info("Welcome home! The client Locale is {}.", locale);
        Date date = new Date();
        DateFormat dateFormat = DateFormat.getDateInstance(DateFormat.LONG, DateFormat.LONG, Locale);
        String formattedDate = dateFormat.format(date);
        model.addAttribute("serverTime", formattedDate );
        return "home";
    }
}
```

Un exemple de vue JSP

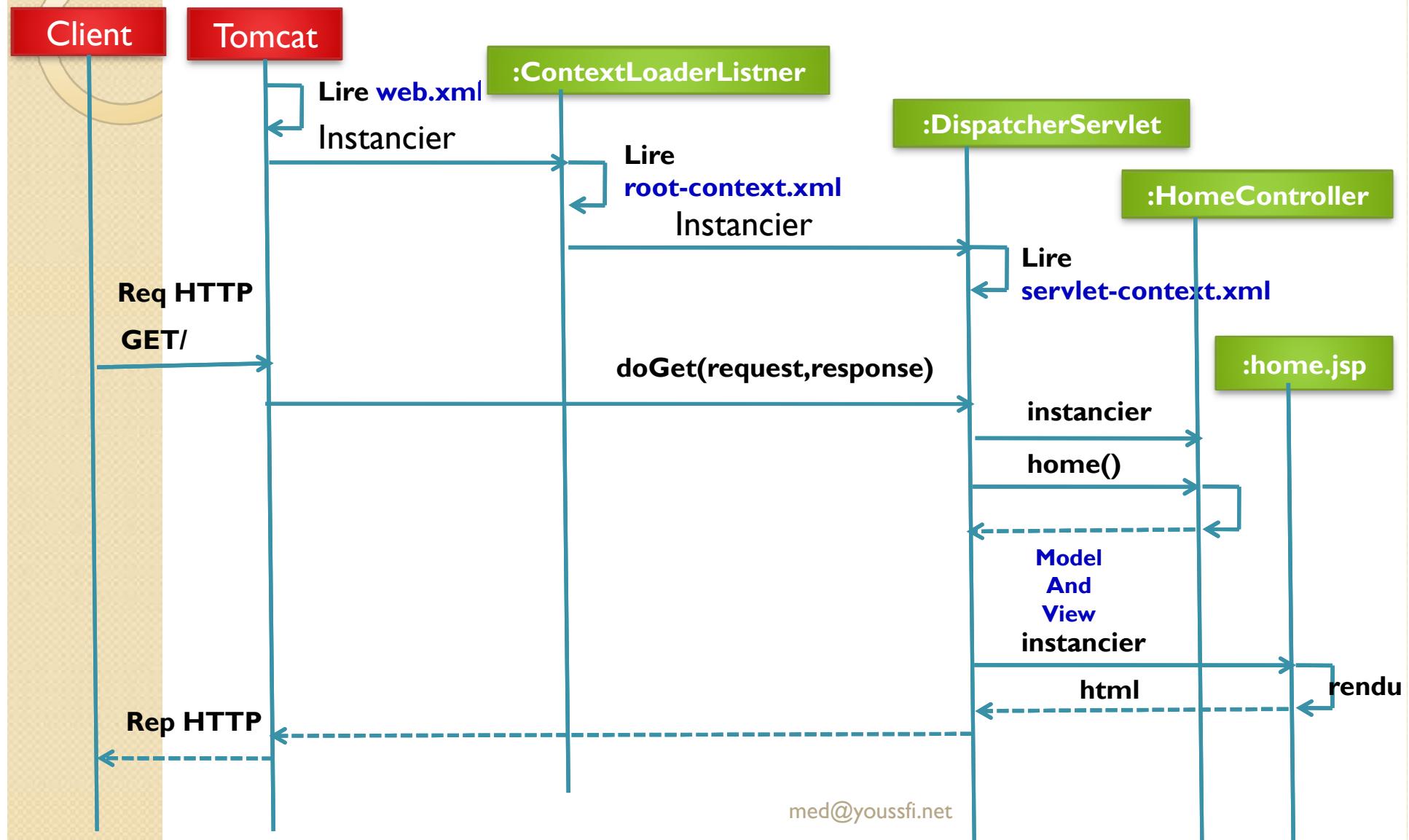
```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ page session="false" %>
<html>
    <head>
        <title>Home</title>
    </head>
    <body>
        <h1> Hello world! </h1>
        <P> The time on the server is ${serverTime}. </P>
    </body>
</html>
```

◀ ▶ ■ ⌂ http://localhost:8080/myCataogue/

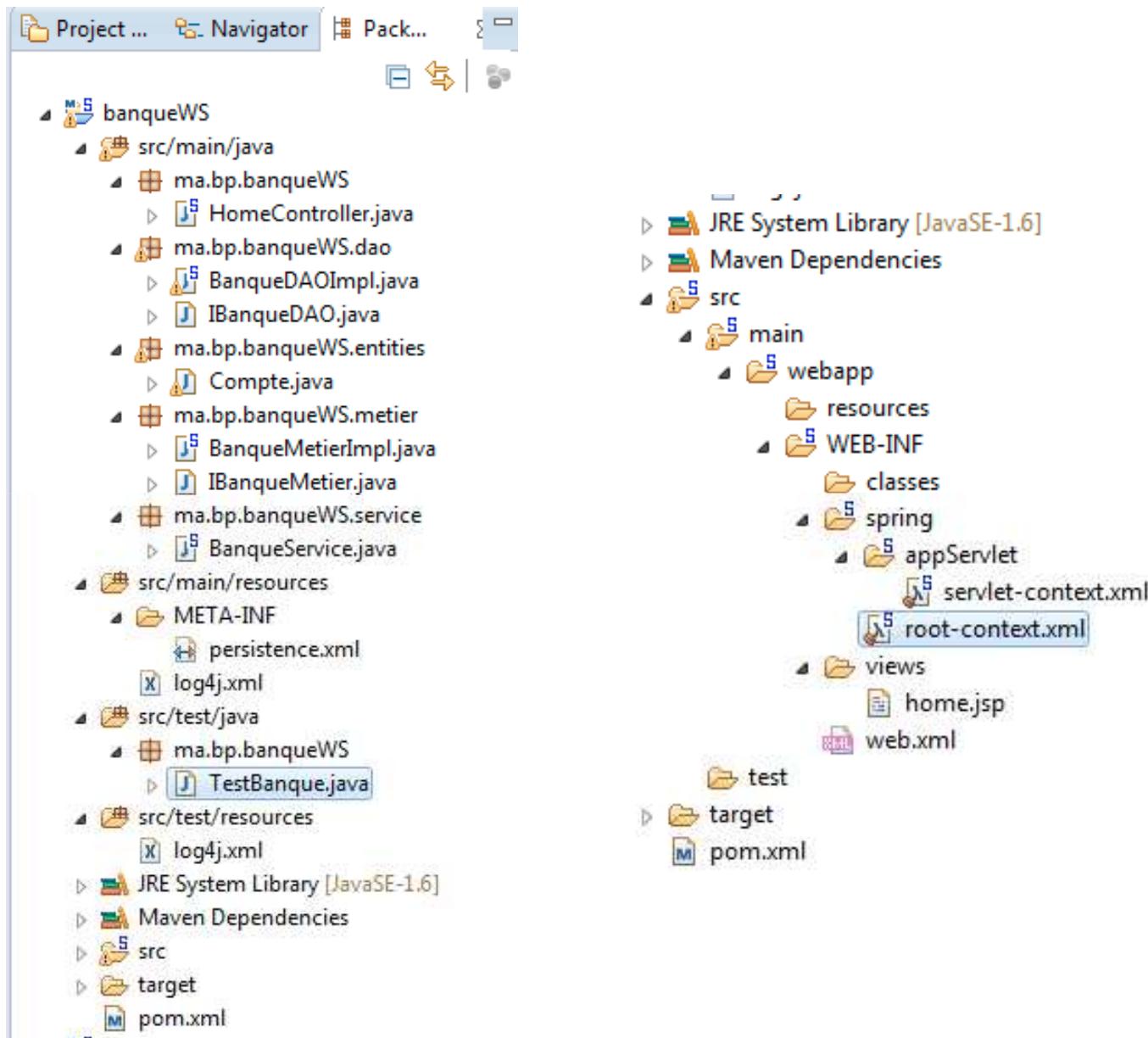
Hello world!

The time on the server is 23 décembre 2013 13:47:12 WET.

Fonctionnement



Structure du projet



Création des entités

```
package ma.bp.banqueWS;
import java.io.Serializable;
import java.util.Date;
import javax.xml.bind.annotation.XmlRootElement;
@XmlRootElement
public class Compte implements Serializable {
    private Long code;
    private double solde;
    private Date dateCreation;
    // Getters et Setters
    // Constructeurs
}
```

Web Service

```
package ma.bp.banqueWS;  
import java.util.*;  
import javax.jws.*;  
  
@WebService  
public class BanqueService {  
    @WebMethod(operationName="conversionEuroToDH")  
    public double conversion(double mt){  
        return mt*11;  
    }  
    @WebMethod  
    public Compte getCompte(@WebParam(name="code")Long code){  
        return new Compte(code,Math.random()*4000, new Date());  
    }  
    @WebMethod  
    public List<Compte> getComptes(){  
        List<Compte> cptes=new ArrayList<Compte>();  
        cptes.add (new Compte(1L,Math.random()*4000, new Date()));  
        cptes.add (new Compte(2L,Math.random()*4000, new Date()));  
        return cptes;  
    }  
}
```

Déploiement du web service avec Spring

/WEB-INF/spring/root-context.xml

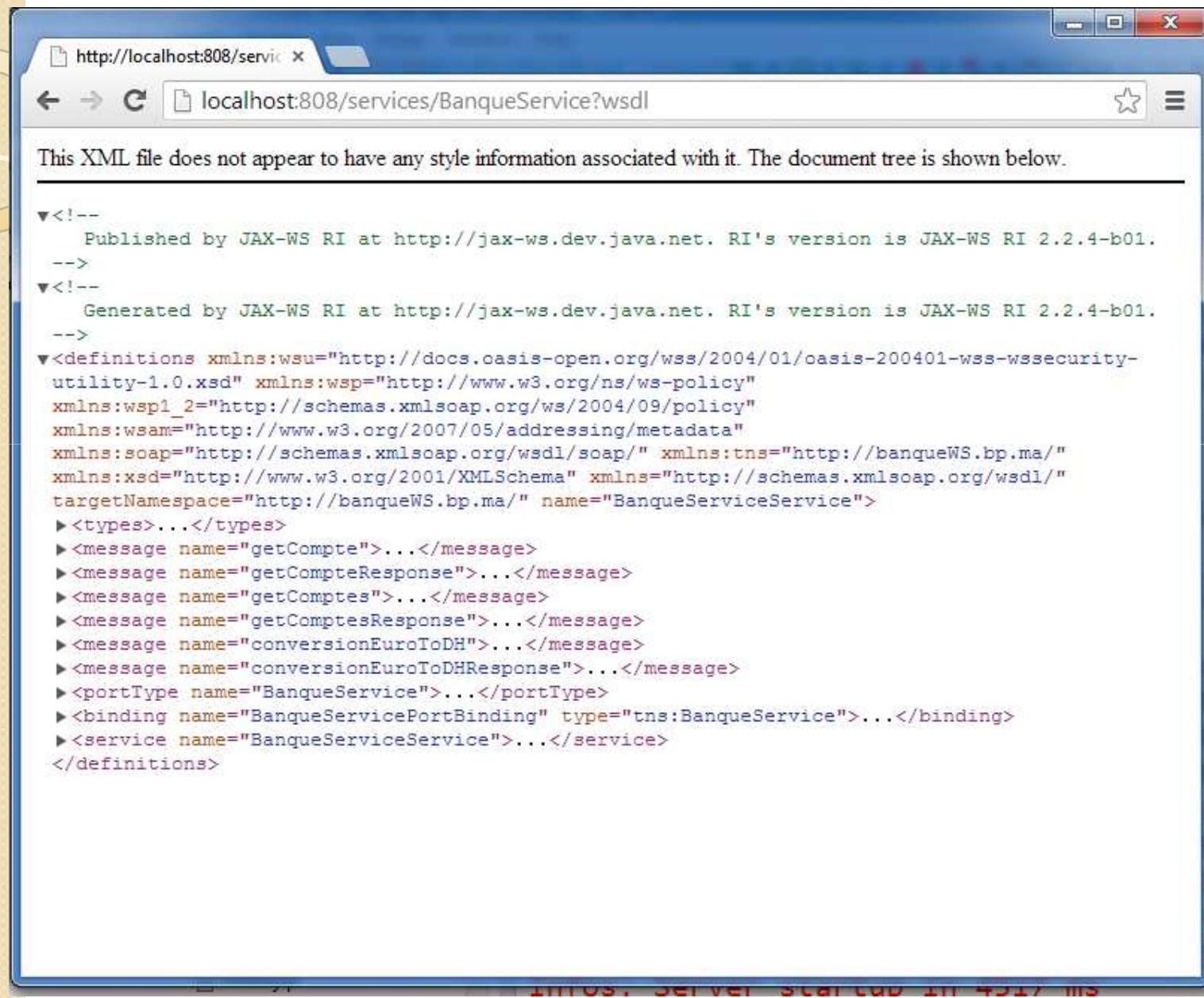
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans.xsd">
    <!-- Root Context: defines shared resources visible to all other web components -->
    <bean id="service" class="ma.bp.banqueWS.BanqueService"></bean>
    <bean class="org.springframework.remoting.jaxws.SimpleJaxWsServiceExporter">
        <property name="baseAddress" value="http://localhost:808/services/"></property>
    </bean>
</beans>
```



Déploiement du projet

- Déployer le projet en utilisant Tomcat7
- Démarrer le serveur Tomcat

Tester le web service



The screenshot shows a Microsoft Internet Explorer window displaying the WSDL (Web Services Description Language) document for a web service. The URL in the address bar is `localhost:808/services/BanqueService?wsdl`. The page content is as follows:

```
This XML file does not appear to have any style information associated with it. The document tree is shown below.

<!--
Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2.4-b01.
-->
<!--
Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2.4-b01.
-->
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns: wsp="http://www.w3.org/ns/ws-policy" xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns: wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns: soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns: tns="http://banqueWS.bp.ma/" xmlns: xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://banqueWS.bp.ma/" name="BanqueServiceService">
  <types>...</types>
  <message name="getCompte">...</message>
  <message name="getCompteResponse">...</message>
  <message name="getComptes">...</message>
  <message name="getComptesResponse">...</message>
  <message name="conversionEuroToDH">...</message>
  <message name="conversionEuroToDHResponse">...</message>
  <portType name="BanqueService">...</portType>
  <binding name="BanqueServicePortBinding" type="tns:BanqueService">...</binding>
  <service name="BanqueServiceService">...</service>
</definitions>
```

At the bottom of the browser window, the status bar displays the message "INFO: SERVER STARTED IN 4517 ms".



Persistante des comptes avec JPA Hibernate

med@youssf.net

Maven Dependencies

```
<!-- Hibernate-->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-entitymanager</artifactId>
    <version>3.6.0.Final</version>
</dependency>
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>4.1.0.Final</version>
</dependency>
<!-- MySQL Connector-->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.6</version>
</dependency>
```

Persistance de la classe Compte

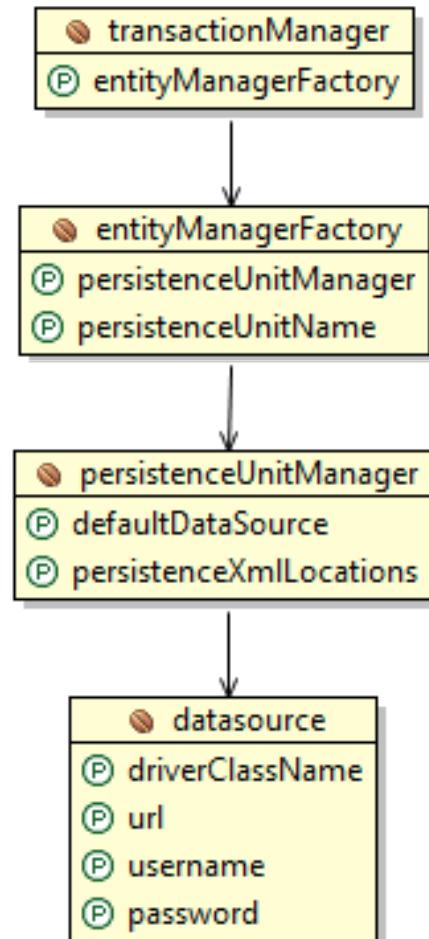
```
package ma.bp.banqueWS.entities;
import java.io.*;
import java.util.*;
import javax.persistence.*;
import javax.xml.bind.annotation.XmlRootElement;
@XmlRootElement
@Entity
public class Compte implements Serializable {
@Id
@GeneratedValue(strategy=GenerationType.IDENTITY)
private Long code;
private double solde;
private Date dateCreation;
// Getters et Setters
// Constructeurs
}
```

src/main/resources/META-INF/persistence.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0" xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
  http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd ">
  <persistence-unit name="UP_EBOUTIQUE" transaction-type="RESOURCE_LOCAL">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <properties>
      <property name="hibernate.show_sql" value="true"/>
      <property name="hibernate.hbm2ddl.auto" value="update"/>
      <property name="hibernate.dialect"
        value="org.hibernate.dialect.MySQLDialect"/>
    </properties>
  </persistence-unit>
</persistence>
```

Configuration JPA avec Spring IOC

Spring Beans



Maven Properties

```
<properties>
<java-version>1.6</java-version>
<org.springframework-
    version>3.2.2.RELEASE</org.springframework-version>
<org.aspectj-version>1.6.10</org.aspectj-version>
<org.slf4j-version>1.6.6</org.slf4j-version>
</properties>
```

Maven Dependencies

```
<!-- Spring -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>${org.springframework-version}</version>
    <exclusions>
        <!-- Exclude Commons Logging in favor of SLF4j -->
        <exclusion>
            <groupId>commons-logging</groupId>
            <artifactId>commons-logging</artifactId>
        </exclusion>
    </exclusions>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>${org.springframework-version}</version>
</dependency>
```

Maven Dependencies

```
<!-- Spring -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-tx</artifactId>
    <version>${org.springframework-version}</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-orm</artifactId>
    <version>${org.springframework-version}</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-beans</artifactId>
    <version>${org.springframework-version}</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>${org.springframework-version}</version>
</dependency>
```

Configuration JPA avec Spring IOC :

/EBoutiqueV2/src/main/resources/applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
                           http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.2.xsd
                           http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.2.xsd">

    <bean id="datasource"
          class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="com.mysql.jdbc.Driver"></property>
        <property name="url" value="jdbc:mysql://localhost:3306/eBanque"></property>
        <property name="username" value="root"></property>
        <property name="password" value=""></property>
    </bean>
```

Configuration JPA avec Spring IOC :

/EBoutiqueV2/src/main/resources/applicationContext.xml

```
<bean id="persistenceUnitManager"
      class="org.springframework.orm.jpa.persistenceunit.DefaultPersistenceUnitManager">
    <property name="defaultDataSource" ref="datasource"></property>
    <property name="persistenceXmlLocations">
        <list>
            <value>classpath*:META-INF/persistence.xml</value>
        </list>
    </property>
</bean>

<bean id="entityManagerFactory"
      class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
    <property name="persistenceUnitManager" ref="persistenceUnitManager"></property>
    <property name="persistenceUnitName" value="UP_EBOUTIQUE"></property>
</bean>

<bean id="transactionManager"
      class="org.springframework.orm.jpa.JpaTransactionManager">
    <property name="entityManagerFactory" ref="entityManagerFactory"></property>
</bean>

<tx:annotation-driven transaction-manager="transactionManager"/>
<context:annotation-config></context:annotation-config>
</beans>
```

Tester la persistance des entités

- Créer la base de données eBanque
- Redéployer le projet
- La table compte devrait être générée



The screenshot shows the MySQL Workbench interface with the following details:

- URL: localhost » ebanque » compte
- Toolbar buttons: Afficher, Structure, SQL, Rechercher, Insérer, Export
- Table structure:

#	Nom	Type	Interclassement	Attributs	Null	Défaut	Extra
1	code	bigint(20)			Non	Aucune	AUTO_INCREMENT
2	dateCreation	datetime			Oui	NULL	
3	solde	double			Non	Aucune	



Couche DAO

med@youssf.net

Interface IBanqueDAO

```
package ma.bp.banqueWS.dao;
import java.util.List;
import ma.bp.banqueWS.entities.Compte;
public interface IBanqueDAO {
    public void addCompte(Compte c);
    public List<Compte> listComptes();
    public Compte getCompte(Long code);
    public void verser(Long code,double mt);
    public void retirer(Long code, double mt);
    public void virement(Long cpte1,Long cpte2,double mt);
    public void deleteCompte(Long code);
}
```

Implémentation DAO : ORM avec JPA

```
package ma.bp.banqueWS.dao;  
import java.util.*; import javax.persistence.*;  
import ma.bp.banqueWS.entities.Compte;  
public class BanqueDAOImpl implements IBanqueDAO {  
    @PersistenceContext  
    private EntityManager em;  
    @Override  
    public void addCompte(Compte c) { em.persist(c); }  
    @Override  
    public List<Compte> listComptes() {  
        Query req=em.createQuery("select c from Compte c");  
        return req.getResultList();  
    }  
    @Override  
    public Compte getCompte(Long code) {  
        Compte cp=em.find(Compte.class, code);  
        if(cp==null) throw new RuntimeException("Compte Introuvable");  
        return cp;  
    }  
}
```

Implémentation DAO : ORM avec JPA

```
@Override  
public void verser(Long code, double mt) {  
    Compte cp=getCompte(code); cp.setSolde(cp.getSolde()+mt);  
}  
  
@Override  
public void retirer(Long code, double mt) {  
    Compte cp=getCompte(code);  
    if(mt>cp.getSolde()) throw new RuntimeException("Solde Insuffisant");  
    cp.setSolde(cp.getSolde()-mt);  
}  
  
@Override  
public void virement(Long cpte1, Long cpte2, double mt) {  
    retirer(cpte1, mt); verser(cpte2, mt);  
}  
  
@Override  
public void deleteCompte(Long code) {  
    Compte cp=getCompte(code); em.remove(cp);  
}  
}
```



Couche Métier

med@youssf.net

Interface IBanqueMetier

```
package ma.bp.banqueWS.metier;
import java.util.List;
import ma.bp.banqueWS.entities.Compte;
public interface IBanqueMetier {
    public void addCompte(Compte c);
    public List<Compte> listComptes();
    public Compte getCompte(Long code);
    public void verser(Long code,double mt);
    public void retirer(Long code, double mt);
    public void virement(Long cpte1,Long cpte2,double mt);
    public void deleteCompte(Long code);
}
```

Implémentation Métier

```
package ma.bp.banqueWS.metier;import java.util.List;
import org.springframework.transaction.annotation.Transactional;
import ma.bp.banqueWS.dao.IBanqueDAO; import ma.bp.banqueWS.entities.Compte;
@Transactional
public class BanqueMetierImpl implements IBanqueMetier {
    private IBanqueDAO dao;
    public void setDao(IBanqueDAO dao) { this.dao = dao; }
@Override
public void addCompte(Compte c) { dao.addCompte(c);}
@Override
public List<Compte> listComptes() { return dao.listComptes(); }
@Override
public Compte getCompte(Long code) { return dao.getCompte(code); }
```

Implémentation Métier (suite)

```
@Override  
public void verser(Long code, double mt) { dao.verser(code, mt); }  
  
@Override  
public void retirer(Long code, double mt) { dao.retirer(code, mt); }  
  
@Override  
public void virement(Long cpte1, Long cpte2, double mt) {  
    dao.virement(cpte1, cpte2, mt); }  
  
@Override  
public void deleteCompte(Long code) {  
    dao.deleteCompte(code); }  
}
```



Web Service

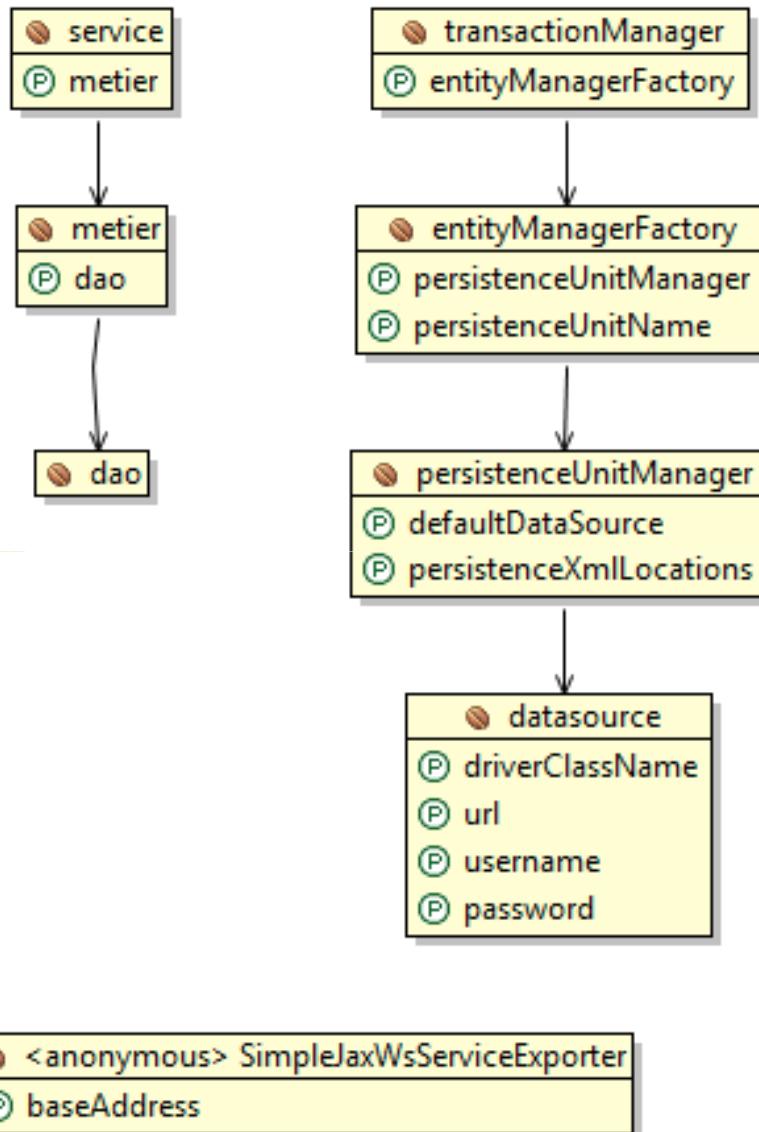
```
package ma.bp.banqueWS.service; import java.util.*; import javax.jws.*;  
import ma.bp.banqueWS.entities.Compte;  
import ma.bp.banqueWS.metier.IBanqueMetier;  
  
@WebService  
  
public class BanqueService {  
    private IBanqueMetier metier;  
  
    @WebMethod(exclude=true)  
    public void setMetier(IBanqueMetier metier) {this.metier = metier; }  
  
    @WebMethod  
    @Oneway  
    public void addCompte(@WebParam(name="solde")double solde){  
        metier.addCompte(new Compte(null, solde,new Date()));  
    }  
  
    @WebMethod  
    public Compte getCompte(@WebParam(name="code")Long code){  
        return metier.getCompte(code);  
    }  
  
    @WebMethod  
    public List<Compte> getComptes(){ return metier.listComptes(); }
```

Web Service

```
@WebMethod  
public void verser(  
    @WebParam(name="code")Long code,  
    @WebParam(name="montant")double mt){ metier.verser(code, mt); }  
  
@WebMethod  
public void retirer(  
    @WebParam(name="code")Long code,  
    @WebParam(name="montant")double mt){ metier.retirer(code, mt); }  
  
@WebMethod  
public void virement(  
    @WebParam(name="cpt1")Long cpt1,  
    @WebParam(name="cpt2")Long cpt2,  
    @WebParam(name="montant")double mt){ metier.virement(cpt1, cpt2,  
    mt); }  
  
@WebMethod  
public void supprimerCompte(@WebParam(name="code")Long code){  
    metier.deleteCompte(code);  
} }
```

Injection des dépendances

Spring Beans





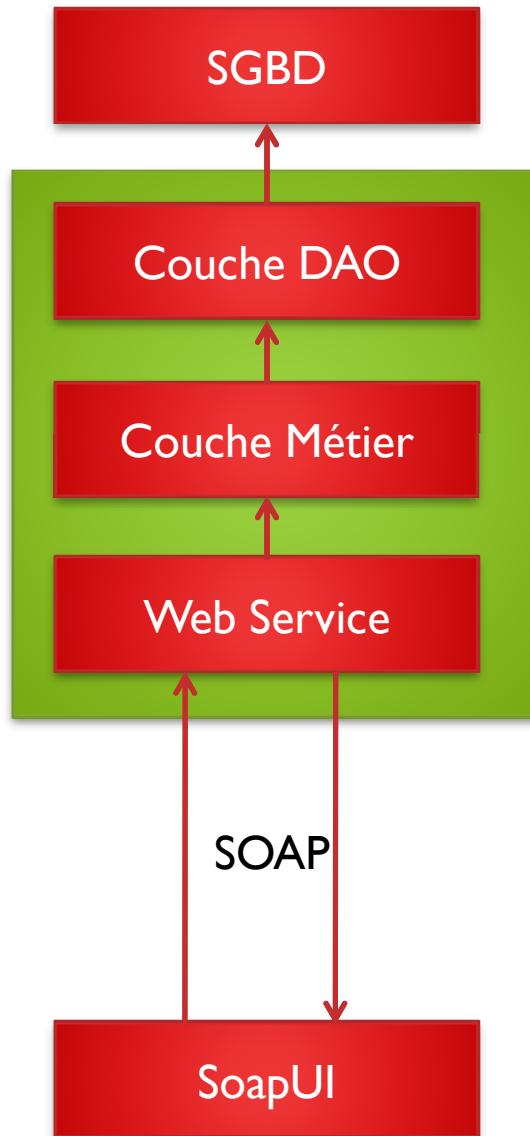
Injection des dépendance :root-context.xml

```
<!-- Root Context: defines shared resources visible to all other web
     components -->
<bean id="dao" class="ma.bp.banqueWS.dao.BanqueDAOImpl"></bean>
<bean id="metier" class="ma.bp.banqueWS.metier.BanqueMetierImpl">
    <property name="dao" ref="dao"></property>
</bean>
<bean id="service" class="ma.bp.banqueWS.service.BanqueService">
    <property name="metier" ref="metier"></property>
</bean>
<bean class="org.springframework.remoting.jaxws.SimpleJaxWsServiceExporter">
    <property name="baseAddress"
        value="http://localhost:808/services/"></property>
</bean>
```

Injection des dépendance :root-context.xml

```
<bean id="datasource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="com.mysql.jdbc.Driver"></property>
    <property name="url" value="jdbc:mysql://localhost:3306/eBanque"></property>
    <property name="username" value="root"></property>
    <property name="password" value=""></property>
</bean>
<bean id="persistenceUnitManager"
      class="org.springframework.orm.jpa.persistenceunit.DefaultPersistenceUnitManager">
    <property name="defaultDataSource" ref="datasource"></property>
    <property name="persistenceXmlLocations">
        <list>
            <value>classpath*:META-INF/persistence.xml</value>
        </list>
    </property>
</bean>
<bean id="entityManagerFactory"
      class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
    <property name="persistenceUnitManager" ref="persistenceUnitManager"></property>
    <property name="persistenceUnitName" value="UP_BANQUE"></property>
</bean>
<bean id="transactionManager" class="org.springframework.orm.jpa.JpaTransactionManager">
    <property name="entityManagerFactory" ref="entityManagerFactory"></property>
</bean>
<tx:annotation-driven transaction-manager="transactionManager"/>
<context:annotation-config></context:annotation-config>
```

Tester les méthodes avec un analyseur SOAP :SoapUI



Tester les méthodes

Requête SOAP pour ajouter un compte

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:ser="http://service.banqueWS.bp.ma/">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <ser:addCompte>  
            <solde>20000</solde>  
        </ser:addCompte>  
    </soapenv:Body>  
</soapenv:Envelope>
```

Pas de réponse SOAP (@oneway)

Tester les méthodes

Requête SOAP pour consulter un compte

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:ser="http://service.banqueWS.bp.ma/">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <ser:getCompte>  
            <code>1</code>  
        </ser:getCompte>  
    </soapenv:Body>  
</soapenv:Envelope>
```

Réponse SOAP

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">  
    <S:Body>  
        <ns2:getCompteResponse xmlns:ns2="http://service.banqueWS.bp.ma/">  
            <return>  
                <code>1</code>  
                <dateCreation>2014-01-24T23:10:01Z</dateCreation>  
                <sold>92600.0</sold>  
            </return>  
        </ns2:getCompteResponse>  
    </S:Body>  
</S:Envelope>
```

Tester les méthodes

Requête SOAP pour consulter les comptes

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:ser="http://service.banqueWS.bp.ma/">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <ser:getComptes/>  
    </soapenv:Body>  
</soapenv:Envelope>
```

Réponse SOAP

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">  
    <S:Body>  
        <ns2:getComptesResponse xmlns:ns2="http://service.banqueWS.bp.ma/">  
            <return>  
                <code>1</code>  
                <dateCreation>2014-01-24T23:10:01Z</dateCreation>  
                <solde>92600.0</solde>  
            </return>  
            <return>  
                <code>2</code>  
                <dateCreation>2014-01-25T08:21:52Z</dateCreation>  
                <solde>20000.0</solde>  
            </return>  
        </ns2:getComptesResponse>  
    </S:Body>  
</S:Envelope>
```

Tester les méthodes

Requête SOAP pour verser un montant

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:ser="http://service.banqueWS.bp.ma/">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <ser:verser>  
            <code>1</code>  
            <montant>2000</montant>  
        </ser:verser>  
    </soapenv:Body>  
</soapenv:Envelope>
```

Réponse SOAP

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">  
    <S:Body>  
        <ns2:verserResponse xmlns:ns2="http://service.banqueWS.bp.ma/">  
    </S:Body>  
</S:Envelope>
```

Retirer un montant

Requête SOAP pour verser un montant

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:ser="http://service.banqueWS.bp.ma/">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <ser:retirer>  
            <code>1</code>  
            <montant>7000000</montant>  
        </ser:retirer>  
    </soapenv:Body>  
</soapenv:Envelope>
```

Réponse SOAP

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">  
    <S:Body>  
        <S:Fault xmlns:ns4="http://www.w3.org/2003/05/soap-envelope">  
            <faultcode>S:Server</faultcode>  
            <faultstring>Solde Insuffisant</faultstring>  
        </S:Fault>  
    </S:Body>  
</S:Envelope>
```

Effectuer un virement

Requête SOAP pour effectuer un virement

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:ser="http://service.banqueWS.bp.ma/">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <ser:virement>  
            <cpte1>1</cpte1>  
            <cpte2>2</cpte2>  
            <montant>8000</montant>  
        </ser:virement>  
    </soapenv:Body>  
</soapenv:Envelope>
```

Réponse SOAP

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">  
    <S:Body>  
        <ns2:virementResponse xmlns:ns2="http://service.banqueWS.bp.ma/">  
    </S:Body>  
</S:Envelope>
```

Supprimer un compte

Requête SOAP pour supprimer un compte

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:ser="http://service.banqueWS.bp.ma/">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <ser:supprimerCompte>  
            <code>2</code>  
        </ser:supprimerCompte>  
    </soapenv:Body>  
</soapenv:Envelope>
```

Réponse SOAP

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">  
    <S:Body>  
        <ns2:supprimerCompteResponse xmlns:ns2="http://service.banqueWS.bp.ma/">  
    </S:Body>  
</S:Envelope>
```



Web services dans une application web J2EE basée sur EJB3 et JBoss7.1

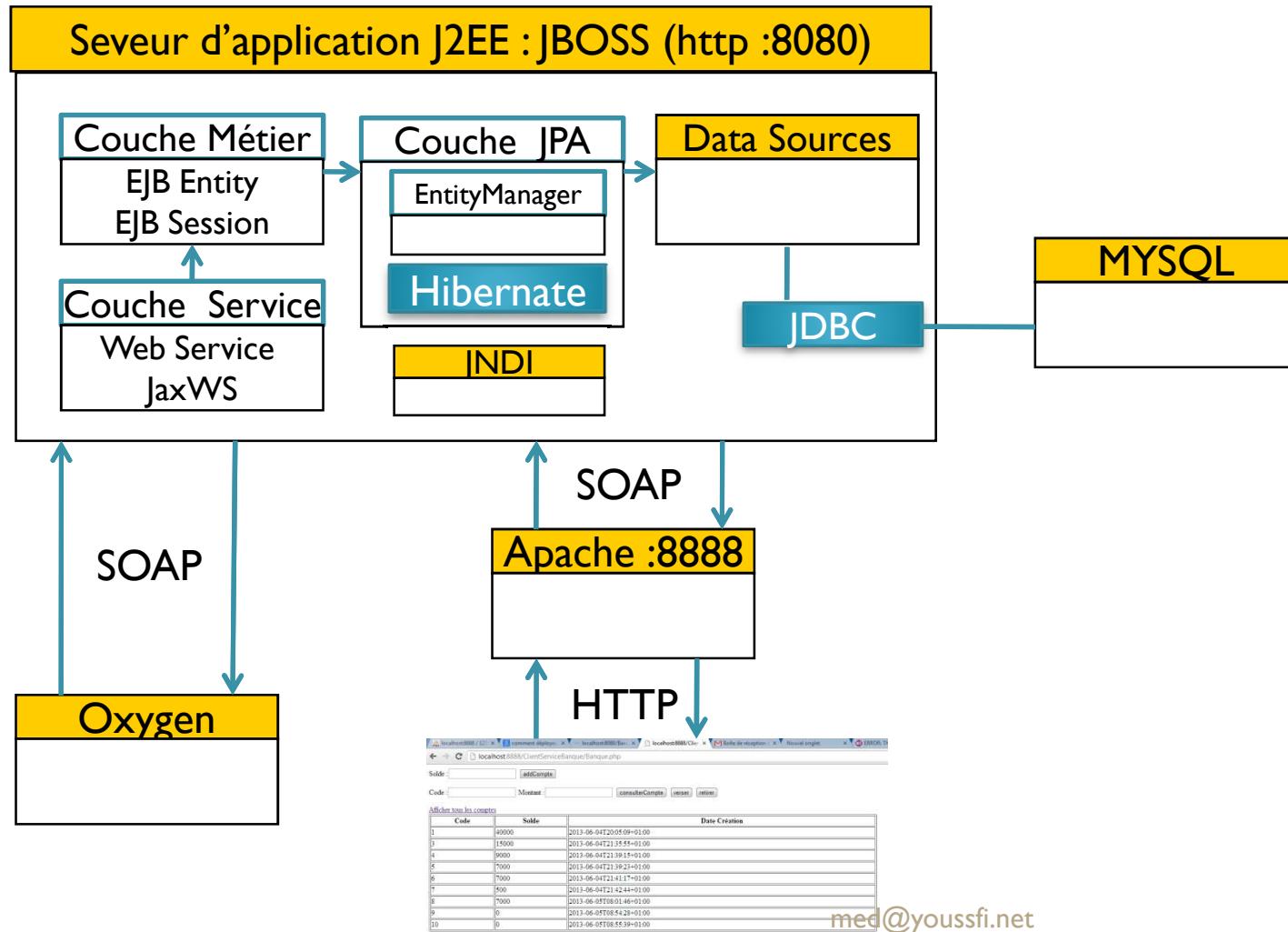
- Dans une application web J2EE basée sur un serveur d'application J2EE
 - Web Sphere
 - Jboss
 - Jonas
 - Glassfish
- Aucune configuration n'est nécessaire pour déployer le web service
- Il suffit de créer un projet web basé sur le serveur d'application
- Créer le web service
- Démarrer le serveur et déployer le projet web
- Chaque serveur d'application possède sa propre implémentation JAX WS : CXF, AXIS, etc...



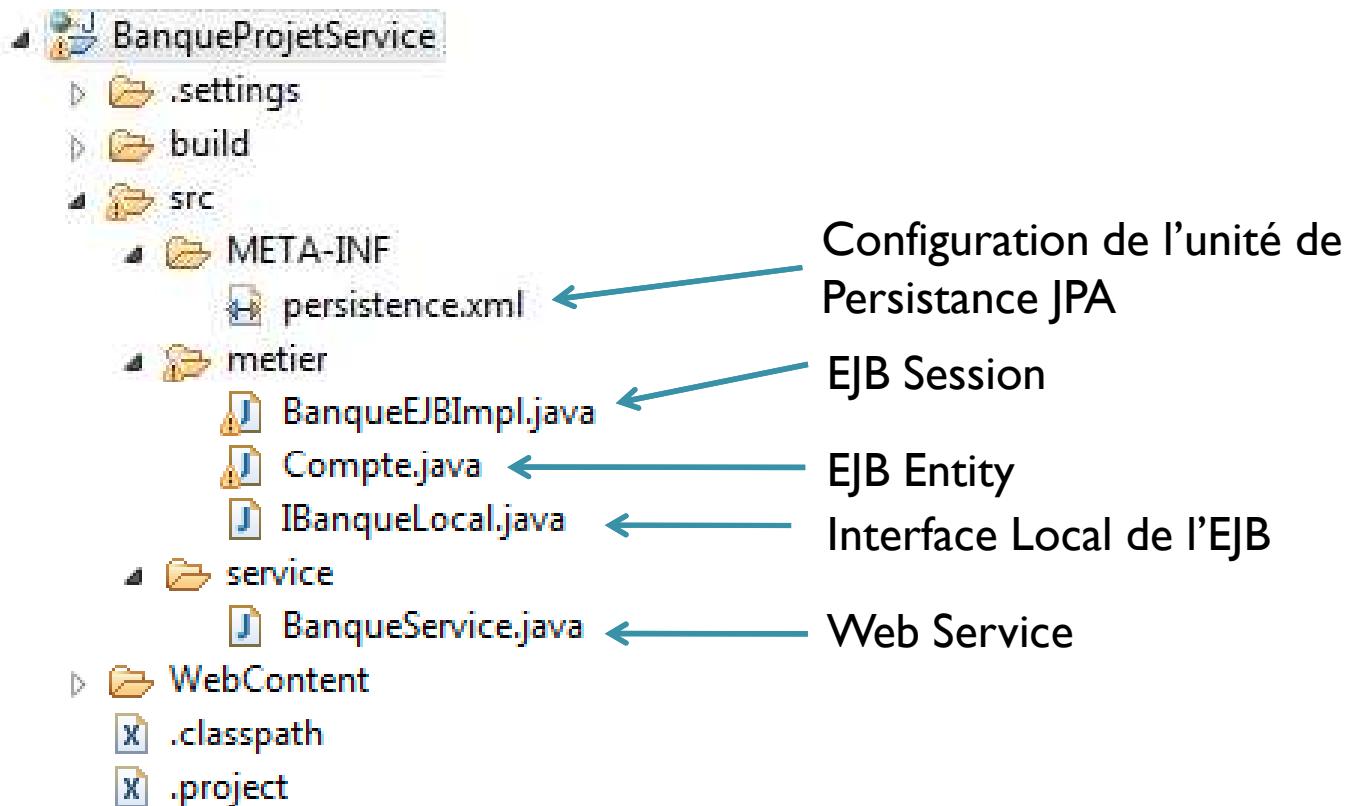
Application

- Créer une application qui permet de gérer des comptes via un service web.
- Chaque compte est défini par un code, un solde, la date de création
- Les comptes sont stockés dans une base de données MYSQL.
- Le mapping objet relationnel est effectué via JPA (EJB Entity et Session)
- La couche service est représenté par un web service de type JAXRS.
- L'application doit permettre de :
 - Ajouter un compte
 - Consulter tous les compte
 - Consulter un compte sachant son code
 - Effectuer un versement d'un montant dans un compte
 - Effectuer un retrait d'un montant sur un compte
- L'application web sera déployé dans un serveur JBoss7
- Créer un client java et un client PHP.

Architecture



Structure du projet J2EE



Données manipulées : Entity Compte

```
package metier;  
import java.io.Serializable; import java.util.Date; import javax.persistence.*;  
@Entity  
@Table(name="COMPTE")  
public class Compte implements Serializable {  
    @Id  
    @GeneratedValue(strategy=GenerationType.IDENTITY)  
    private Long code;  
    private double solde;  
    private Date dateCreation;  
    public Compte() {  
    }  
    public Compte(double solde) {  
        this.solde = solde;  
        this.dateCreation=new Date();  
    }  
    // Getters et Setters  
}
```

Interface Locale

```
package metier;
import java.util.List;
import javax.ejb.Local;
@Local
public interface IBanqueLocal {
    public Compte addCompte(Compte c);
    public List<Compte> getAllComptes();
    public void verser(Long code,double montant);
    public void retirer(Long code,double montant);
    public Compte consulterCompte(Long code);
}
```

Implémentation EJB Session sans état

```
package metier;
import java.util.List;
import javax.ejb.Stateless;
import javax.persistence.*;
@Stateless(name="BP")
public class BanqueEJBImpl implements IBanqueLocal {
    @PersistenceContext(unitName="UP_BP")
    EntityManager em;
    @Override
    public Compte addCompte(Compte c) {
        em.persist(c);
        return c;
    }
    @Override
    public List<Compte> getAllComptes() {
        Query req=em.createQuery("select c from Compte c");
        return req.getResultList();
    }
}
```

Implémentation EJB Session sans état

```
@Override  
public void verser(Long code, double montant) {  
    if(montant<=0) throw new RuntimeException  
    ("Le Montant doit être supérieur à zero");  
    Compte c=em.find(Compte.class, code);  
    c.setSolde(c.getSolde()+montant);  
    em.persist(c);  
}  
  
@Override  
public void retirer(Long code, double montant) {  
    if(montant<=0) throw new RuntimeException  
    ("Le Montant doit être supérieur à zero");  
    Compte c=em.find(Compte.class, code);  
    if(c.getSolde()<=montant) throw new RuntimeException  
    ("Solde Insuffisant");  
    c.setSolde(c.getSolde()-montant);  
}
```

Implémentation EJB Session sans état

```
@Override  
public Compte consulterCompte(Long code) {  
    Compte c=em.find(Compte.class, code);  
    if(c==null) throw new RuntimeException  
        ("Compte introuvable");  
    return c;  
}  
}
```

Configuration de l'unité de persistance : persistence.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence
    xmlns="http://java.sun.com/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd"
    version="1.0">
    <persistence-unit name="UP_BP">
        <jta-data-source>java:/dsBP</jta-data-source>
        <properties>
            <property name="hibernate.hbm2ddl.auto" value="update"/>
        </properties>
    </persistence-unit>
</persistence>
```

Data source

- Fichier standalone.xml de JBoss

```
<datasources>
  <datasource jndi-name="java:/dsBP" pool-name="dsBP" enabled="true">
    <connection-url>jdbc:mysql://127.0.0.1:3306/DB_BP</connection-url>
    <driver-class>com.mysql.jdbc.Driver</driver-class>
    <driver>mysql</driver>
    <security>
      <user-name>root</user-name>
      <password></password>
    </security>
  </datasource>
  <drivers>
    <driver name="mysql" module="com.mysql"/>
  </drivers>
</datasources>
```

Web Service

```
package service;
import java.util.List;
import javax.ejb.EJB;import javax.jws.*;
import metier.Compte;import metier.IBanqueLocal;
@WebService
public class BanqueService {
    @EJB
    private IBanqueLocal metier;
    @WebMethod
    public Compte addCompte(@WebParam(name="solde")double solde) {
        return metier.addCompte(new Compte(solde));
    }
    @WebMethod
    public List<Compte> getAllComptes() {
        return metier.getAllComptes();
    }
}
```

Web Service

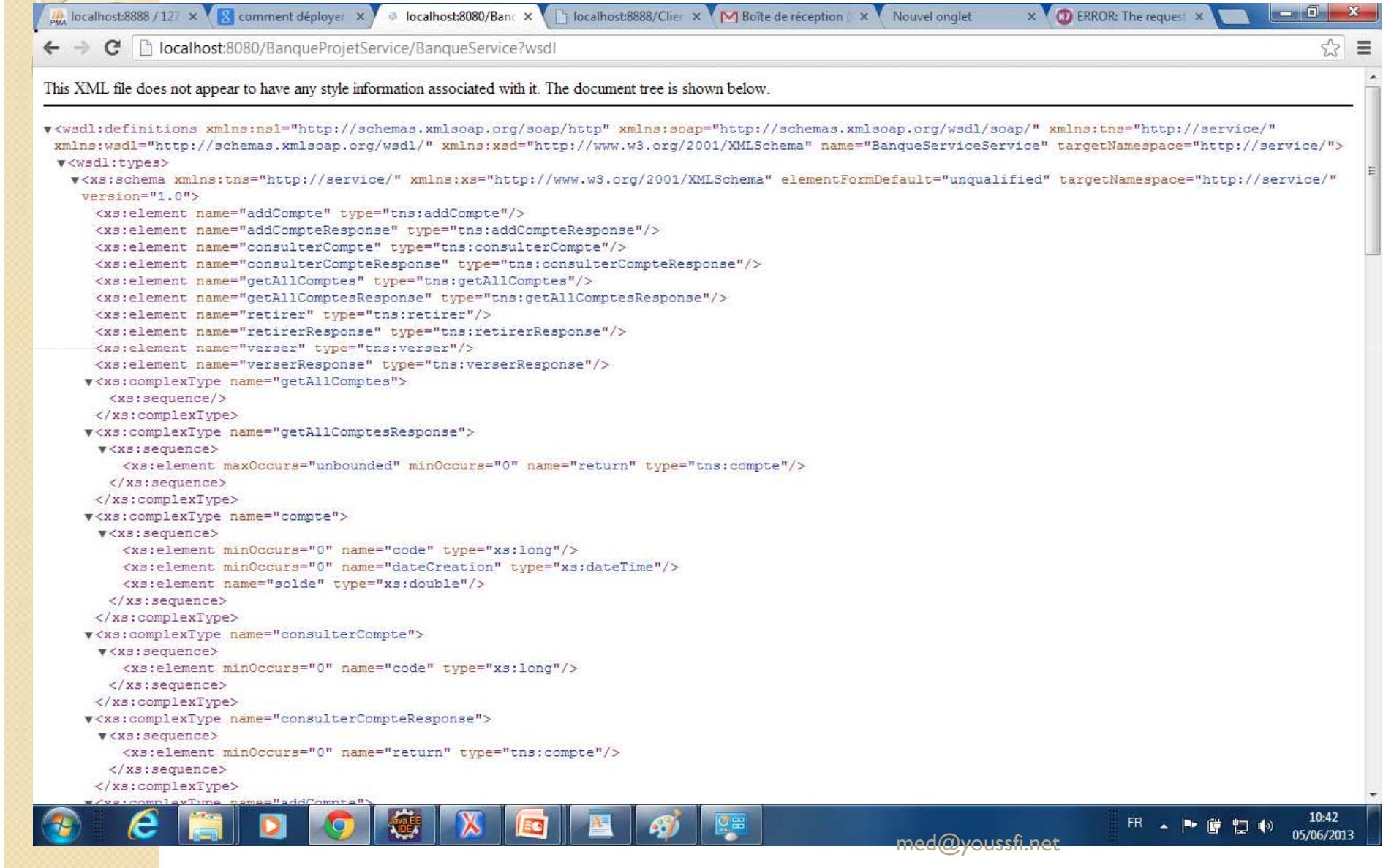
```
@WebMethod  
  
public void verser(@WebParam(name="code")Long code,  
@WebParam(name="montant")double montant) {  
    metier.verser(code, montant);  
}  
  
@WebMethod  
  
public void retirer(@WebParam(name="code")Long code,  
@WebParam(name="montant")double montant) {  
    metier.retirer(code, montant);  
}  
  
@WebMethod  
  
public Compte consulterCompte(@WebParam(name="code")Long code) {  
    return metier.consulterCompte(code);  
}  
}
```



Déployer et Tester le projet Web

- Démarrer le serveur JBoss
- Déployer le projet Web
- Consulter le WSDL
- Tester Le web service avec Oxygen

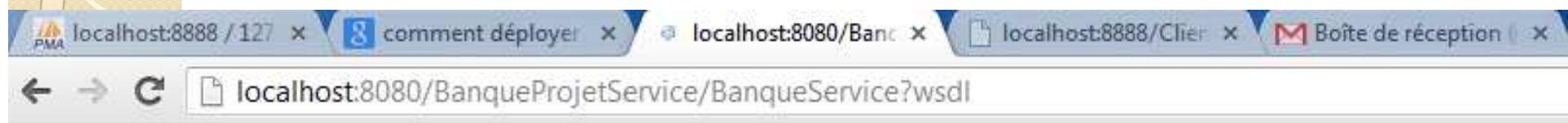
Analyse du WSDL



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<wsdl:definitions xmlns:ns1="http://schemas.xmlsoap.org/soap/http" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://service/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="BanqueServiceService" targetNamespace="http://service/">
  <wsdl:types>
    <xss:schema xmlns:tns="http://service/" xmlns:xss="http://www.w3.org/2001/XMLSchema" elementFormDefault="unqualified" targetNamespace="http://service/" version="1.0">
      <xss:element name="addCompte" type="tns:addCompte"/>
      <xss:element name="addCompteResponse" type="tns:addCompteResponse"/>
      <xss:element name="consulterCompte" type="tns:consulterCompte"/>
      <xss:element name="consulterCompteResponse" type="tns:consulterCompteResponse"/>
      <xss:element name="getAllComptes" type="tns:getAllComptes"/>
      <xss:element name="getAllComptesResponse" type="tns:getAllComptesResponse"/>
      <xss:element name="retirer" type="tns:retirer"/>
      <xss:element name="retirerResponse" type="tns:retirerResponse"/>
      <xss:element name="verser" type="tns:verser"/>
      <xss:element name="verserResponse" type="tns:verserResponse"/>
    </xss:schema>
    <xss:complexType name="getAllComptes">
      <xss:sequence/>
    </xss:complexType>
    <xss:complexType name="getAllComptesResponse">
      <xss:sequence>
        <xss:element maxOccurs="unbounded" minOccurs="0" name="return" type="tns:compte"/>
      </xss:sequence>
    </xss:complexType>
    <xss:complexType name="compte">
      <xss:sequence>
        <xss:element minOccurs="0" name="code" type="xs:long"/>
        <xss:element minOccurs="0" name="dateCreation" type="xs:dateTime"/>
        <xss:element name="solde" type="xs:double"/>
      </xss:sequence>
    </xss:complexType>
    <xss:complexType name="consulterCompte">
      <xss:sequence>
        <xss:element minOccurs="0" name="code" type="xs:long"/>
      </xss:sequence>
    </xss:complexType>
    <xss:complexType name="consulterCompteResponse">
      <xss:sequence>
        <xss:element minOccurs="0" name="return" type="tns:compte"/>
      </xss:sequence>
    </xss:complexType>
  </wsdl:types>
  <xss:complexType name="addCompte">
```

Resumé du WSDL



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<wsdl:definitions xmlns:ns1="http://schemas.xmlsoap.org/soap/http" xmlns:soap="http://schemas.xmls  
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="B  
▶ <wsdl:types>...</wsdl:types>  
▶ <wsdl:message name="retirerResponse">...</wsdl:message>  
▶ <wsdl:message name="addCompte">...</wsdl:message>  
▶ <wsdl:message name="consulterCompteResponse">...</wsdl:message>  
▶ <wsdl:message name="getAllComptesResponse">...</wsdl:message>  
▶ <wsdl:message name="verserResponse">...</wsdl:message>  
▶ <wsdl:message name="retirer">...</wsdl:message>  
▶ <wsdl:message name="consulterCompte">...</wsdl:message>  
▶ <wsdl:message name="getAllComptes">...</wsdl:message>  
▶ <wsdl:message name="addCompteResponse">...</wsdl:message>  
▶ <wsdl:message name="verser">...</wsdl:message>  
▶ <wsdl:portType name="BanqueService">...</wsdl:portType>  
▶ <wsdl:binding name="BanqueServiceServiceSoapBinding" type="tns:BanqueService">...</wsdl:binding>  
▶ <wsdl:service name="BanqueServiceService">...</wsdl:service>  
</wsdl:definitions>
```

Le schéma XML des données échangées :

```
<wsdl:types>
  <xsd:schema elementFormDefault="unqualified" targetNamespace="http://service/" version="1.0"
    xmlns:tns="http://service/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="addCompte" type="tns:addCompte"/>
    <xsd:element name="addCompteResponse" type="tns:addCompteResponse"/>
    <xsd:element name="consulterCompte" type="tns:consulterCompte"/>
    <xsd:element name="consulterCompteResponse" type="tns:consulterCompteResponse"/>
    <xsd:element name="getAllComptes" type="tns:getAllComptes"/>
    <xsd:element name="getAllComptesResponse" type="tns:getAllComptesResponse"/>
    <xsd:element name="retirer" type="tns:retirer"/>
    <xsd:element name="retirerResponse" type="tns:retirerResponse"/>
    <xsd:element name="verser" type="tns:verser"/>
    <xsd:element name="verserResponse" type="tns:verserResponse"/>
    <xsd:complexType name="getAllComptes">
      <xsd:sequence/>
    </xsd:complexType>
    <xsd:complexType name="getAllComptesResponse">
      <xsd:sequence>
        <xsd:element maxOccurs="unbounded" minOccurs="0" name="return" type="tns:compte"/>
      </xsd:sequence>
    </xsd:complexType>
```

Le schéma XML des données échangées :

```
<xs:complexType name="compte">
  <xs:sequence>
    <xs:element minOccurs="0" name="code" type="xs:long"/>
    <xs:element minOccurs="0" name="dateCreation" type="xs:dateTime"/>
    <xs:element name="solde" type="xs:double"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="consulterCompte">
  <xs:sequence>
    <xs:element minOccurs="0" name="code" type="xs:long"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="consulterCompteResponse">
  <xs:sequence>
    <xs:element minOccurs="0" name="return" type="tns:compte"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="addCompte">
  <xs:sequence>
    <xs:element name="solde" type="xs:double"/>
  </xs:sequence>
</xs:complexType>
```

Le schéma XML des données échangées :

```
<xs:complexType name="addCompteResponse">
  <xs:sequence>
    <xs:element minOccurs="0" name="return" type="tns:compte"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="retirer">
  <xs:sequence>
    <xs:element minOccurs="0" name="code" type="xs:long"/>
    <xs:element name="montant" type="xs:double"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="retirerResponse">
  <xs:sequence/>
</xs:complexType>
<xs:complexType name="verser">
  <xs:sequence>
    <xs:element minOccurs="0" name="code" type="xs:long"/>
    <xs:element name="montant" type="xs:double"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="verserResponse">
  <xs:sequence/>
</xs:complexType>
</xs:schema>
</wsdl:types>
```

Description des messages

- <wsdl:message name="retirer">
 <wsdl:part element="tns:retirer" name="parameters">
 </wsdl:part>
 </wsdl:message>
- <wsdl:message name="retirerResponse">
 <wsdl:part element="tns:retirerResponse" name="parameters">
 </wsdl:part>
 </wsdl:message>



L'élément PortType

- ```
<wsdl:portType name="BanqueService">

 <wsdl:operation name="getAllComptes">
 <wsdl:input message="tns:getAllComptes" name="getAllComptes"></wsdl:input>
 <wsdl:output message="tns:getAllComptesResponse" name="getAllComptesResponse">
 </wsdl:output>
 </wsdl:operation>

 <wsdl:operation name="consulterCompte">
 <wsdl:input message="tns:consulterCompte" name="consulterCompte">
 </wsdl:input>
 <wsdl:output message="tns:consulterCompteResponse" name="consulterCompteResponse">
 </wsdl:output>
 </wsdl:operation>


```
- ```
</wsdl:portType>
```

L'élément SoapBinding

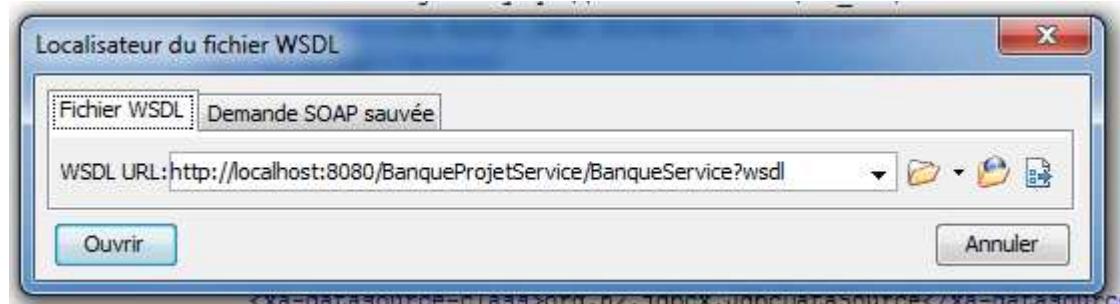
- ```
<wsdl:binding name="BanqueServiceServiceSoapBinding" type="tns:BanqueService">
 <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
 <wsdl:operation name="getAllComptes">
 <soap:operation soapAction="" style="document"/>
 <wsdl:input name="getAllComptes">
 <soap:body use="literal"/>
 </wsdl:input>
 <wsdl:output name="getAllComptesResponse">
 <soap:body use="literal"/>
 </wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="consulterCompte">
 <soap:operation soapAction="" style="document"/>
 <wsdl:input name="consulterCompte">
 <soap:body use="literal"/>
 </wsdl:input>
 <wsdl:output name="consulterCompteResponse">
 <soap:body use="literal"/>
 </wsdl:output>
 </wsdl:operation>

</wsdl:binding>
```

# L'élément Service

```
<wsdl:service name="BanqueServiceService">
 <wsdl:port
 binding="tns:BanqueServiceServiceSoapBinding"
 name="BanqueServicePort">
 <soap:address
 location="http://localhost:8080/BanqueProjetService/Banque
 Service"/>
 </wsdl:port>
</wsdl:service>
```

# Oxygen



- <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">  
    <SOAP-ENV:Header/>  
    <SOAP-ENV:Body>  
        <addCompte xmlns="http://service/">  
            <solde xmlns="">3000</solde>  
        </addCompte>  
    </SOAP-ENV:Body>  
</SOAP-ENV:Envelope>

Requête SOAP pour ajouter un compte

- <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">  
    <soap:Body>  
        <ns2:addCompteResponse xmlns:ns2="http://service/">  
            <return>  
                <code>11</code>  
                <dateCreation>2013-06-05T13:09:01.469+01:00</dateCreation>  
                <solde>3000.0</solde>  
            </return>  
        </ns2:addCompteResponse>  
    </soap:Body>  
</soap:Envelope>

Réponse SOAP après ajout d'un compte

# Oxygen : Effectuer un versement

```
<SOAP-ENV:Envelope xmlns:SOAP-
 ENV="http://schemas.xmlsoap.org/soap/envelope/">
 <SOAP-ENV:Header/>
 <SOAP-ENV:Body>
 <verser xmlns="http://service/">
 <code xmlns="">1</code>
 <montant xmlns="">9000</montant>
 </verser>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Requête SOAP pour verser un montant

```
<soap:Envelope
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
 <ns2:verserResponse xmlns:ns2="http://service/">
 </soap:Body>
</soap:Envelope>
```

Réponse SOAP

# Oxygen : Effectuer le retrait

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
 <SOAP-ENV:Header/>
 <SOAP-ENV:Body>
 <retirer xmlns="http://service/">
 <code xmlns="">1</code>
 <montant xmlns="">600000</montant>
 </retirer>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Requête SOAP pour retirer un montant>solde

```
<soap:Envelope
 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
 <soap:Fault>
 <faultcode>soap:Server</faultcode>
 <faultstring>java.lang.RuntimeException: Solde
Insuffisant</faultstring>
 </soap:Fault>
 </soap:Body>
</soap:Envelope>
```

Réponse SOAP qui retourne une exception

# Oxygen : Consulter les Comptes

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
 <SOAP-ENV:Header/>
 <SOAP-ENV:Body>
 <getallComptes xmlns="http://service/">
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Requête SOAP pour consulter tous les comptes

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
 <ns2:getAllComptesResponse xmlns:ns2="http://service/">
 <return>
 <code>1</code>
 <dateCreation>2013-06-04T20:05:09+01:00</dateCreation>
 <solde>29000.0</solde>
 </return>
 <return>
 <code>3</code>
 <dateCreation>2013-06-04T21:35:55+01:00</dateCreation>
 <solde>15000.0</solde>
 </return>
 </ns2:getAllComptesResponse>
 </soap:Body>
</soap:Envelope>
```

Réponse SOAP qui retourne une exception

# Client SOAP PHP

The screenshot shows a web browser window with multiple tabs open. The active tab is titled "localhost:8888/ClientServiceBanque/Banque.php". The page contains a form for adding a new account and another for managing existing accounts. Below these forms is a table displaying all account details.

**Form 1 (Top):**

Solde :  addCompte

**Form 2 (Middle):**

Code :  Montant :  consulterCompte verser retirer

**Table (Bottom):**

| Code | Solde | Date Création             |
|------|-------|---------------------------|
| 1    | 40000 | 2013-06-04T20:05:09+01:00 |
| 3    | 15000 | 2013-06-04T21:35:55+01:00 |
| 4    | 9000  | 2013-06-04T21:39:15+01:00 |
| 5    | 7000  | 2013-06-04T21:39:23+01:00 |
| 6    | 7000  | 2013-06-04T21:41:17+01:00 |
| 7    | 500   | 2013-06-04T21:42:44+01:00 |
| 8    | 7000  | 2013-06-05T08:01:46+01:00 |
| 9    | 0     | 2013-06-05T08:54:28+01:00 |
| 10   | 0     | 2013-06-05T08:55:39+01:00 |



# Client PHP

```
<?php
try {
$clientSoap=new
SoapClient("http://localhost:8080/BanqueProjetService/BanqueService?wsdl");
if(isset($_GET['action'])){
$operation=$_GET['action'];
if(($operation=="verser")||($operation=="retirer")){
$param1=new stdClass();
$param1->code=$_GET['code'];
$param1->montant=$_GET['montant'];
$clientSoap->__soapCall($operation, array($param1));
$param2=new stdClass();
$param2->code=$_GET['code'];
$repSoap=$clientSoap->__soapCall("consulterCompte", array($param2));
}
}
```



# Client PHP

```
else if($operation=="addCompte"){
$param1=new stdClass();
$param1->solde=$_GET['solde'];
$repSoap=$clientSoap->__soapCall($operation, array($param1));
}

else if($operation=="consulterCompte"){
$param1=new stdClass();
$param1->code=$_GET['code'];
$repSoap=$clientSoap->__soapCall($operation, array($param1));
}

else{
$repSoap=$clientSoap->__soapCall("getAllComptes", array());
}

} catch (Exception $e) {
}

?>
```



# Client PHP

```
<html>
<head>
</head>
<body>
 <div>
 <form action="Banque.php" method="get">
 Solde :<input type="text" name="solde">
 <input type="submit" name="action" value="addCompte">
 </form>
 </div>
 <div>
 <form action="Banque.php" method="get">
 Code :<input type="text" name="code">
 Montant :<input type="text" name="montant">
 <input type="submit" name="action" value="consulterCompte">
 <input type="submit" name="action" value="verser">
 <input type="submit" name="action" value="retirer">
 </form>
 </div>
 Afficher tous les comptes
```



# Client PHP

```
<table border="1" width="80%">
 <tr>
 <th>Code</th><th>Solde</th><th>Date Création</th>
 </tr>
 <?php
 if(isset($repSoap)){
 if(is_array($repSoap->return)){
 $tab=$repSoap->return;
 } else{
 $tab=array($repSoap->return);
 }
 foreach ($tab as $cpt){?>
 <tr>
 <td><?php echo($cpt->code)?></td><td><?php echo($cpt->solde)?></td>
 <td><?php echo($cpt->dateCreation)?></td>
 </tr>
 <?php } } ?>
 </table>
 <?php if(isset($e)){?> <?php echo ($e->getMessage())?> <?php }?>
</body></html>
```



# Client PHP

```
<table border="1" width="80%">
 <tr>
 <th>Code</th><th>Solde</th><th>Date Création</th>
 </tr>
 <?php
 if(isset($repSoap)){
 if(is_array($repSoap->return)){
 $tab=$repSoap->return;
 } else{
 $tab=array($repSoap->return);
 }
 foreach ($tab as $cpt){?>
 <tr>
 <td><?php echo($cpt->code)?></td><td><?php echo($cpt->solde)?></td>
 <td><?php echo($cpt->dateCreation)?></td>
 </tr>
 <?php } } ?>
 </table>
 <?php if(isset($e)){?> <?php echo ($e->getMessage())?> <?php }?>
</body></html>
```



# UDDI

med@youssf.net



# UDDI

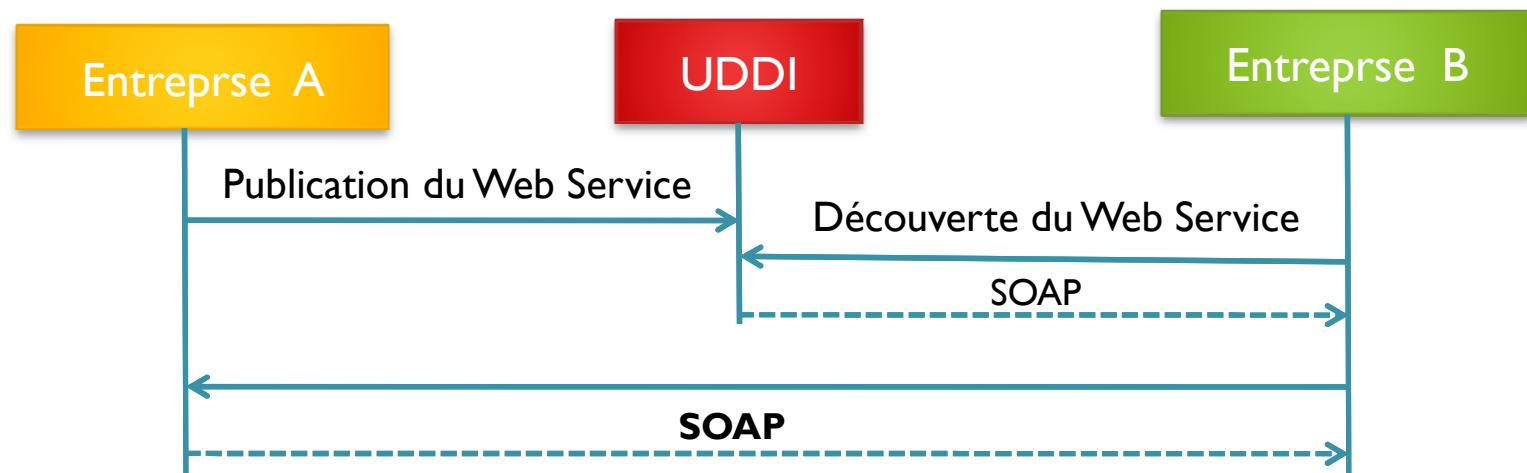
- L'annuaire des services UDDI est un standard pour la publication et la découverte des informations sur les services Web.
- La spécification UDDI est une initiative lancée par ARIBA, Microsoft et IBM.
- Cette spécification n'est pas gérée par le W3C mais par le groupe OASIS.
- La spécification UDDI vise à créer une plate-forme indépendante, un espace de travail (framework) ouvert pour la description, la découverte et l'intégration des services des entreprises.

# Consultation de l'annuaire

- L'annuaire UDDI se concentre sur le processus de découverte de l'architecture orientée services (SOA), et utilise des technologies standards telles que XML, SOAP et WSDL qui permettent de simplifier la collaboration entre partenaires dans le cadre des échanges commerciaux.
- L'accès au référentiel s'effectue de différentes manières.
  - Les pages blanches comprennent la liste des entreprises ainsi que des informations associées à ces dernières (coordonnées, description de l'entreprise, identifiants...).
  - Les pages jaunes recensent les services Web de chacune des entreprises sous le standard WSDL.
  - Les pages vertes fournissent des informations techniques précises sur les services fournis.

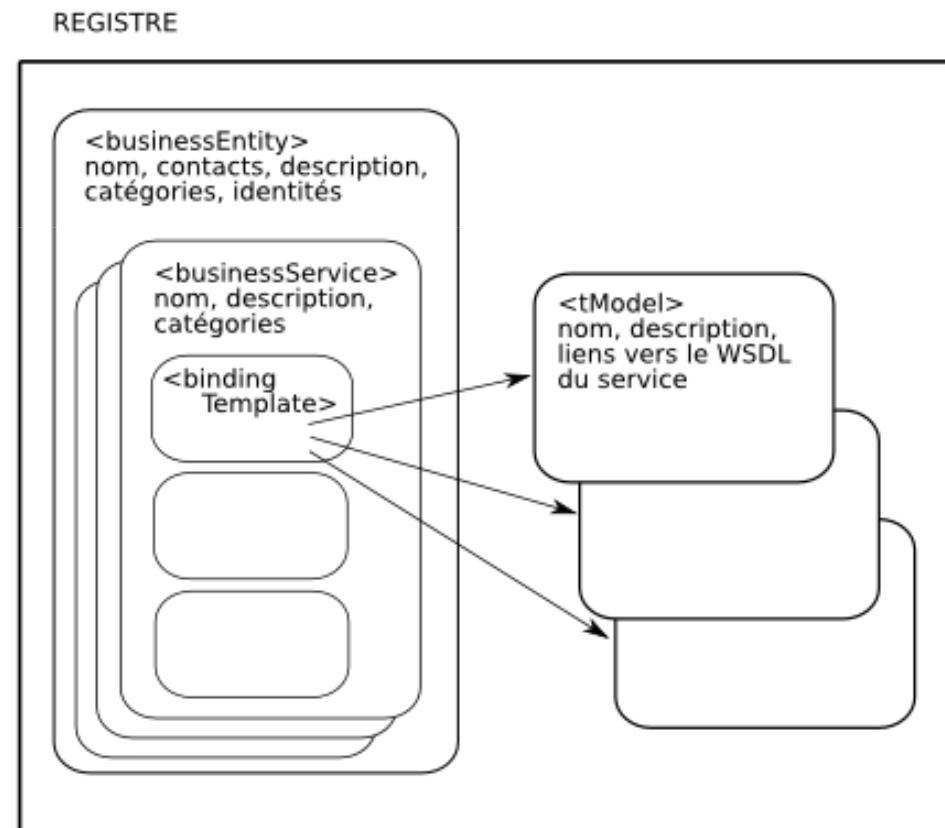
# Architecture

- Les entreprises publient les descriptions de leurs services Web en UDDI, sous la forme de fichiers WSDL.
- Ainsi, les clients peuvent plus facilement rechercher les services Web dont ils ont besoin en interrogeant le registre UDDI.
- Lorsqu'un client trouve une description de service Web qui lui convient, il télécharge son fichier WSDL depuis le registre UDDI. Ensuite, à partir des informations inscrites dans le fichier WSDL, notamment la référence vers le service Web, le client peut invoquer le service Web et lui demande d'exécuter certaines de ses fonctionnalités.
- Le scénario classique d'utilisation de UDDI est illustré ci-dessous. L'entreprise B a publié le service Web S, et l'entreprise A est client de ce service :



# Structures de données UDDI

- Un registre UDDI se compose de quatre types de structures de données,
  - le **businessEntity**,
  - Le **businessService**,
  - le **bindingTemplate**
  - et la **tModel**.





# **BusinessEntity (entité d'affaires)**

- Les « businessEntities » sont en quelque sorte les pages blanches d'un annuaire UDDI.
- Elles décrivent les organisations ayant publié des services dans le répertoire.
- On y trouve notamment
  - le nom de l'organisation,
  - ses adresses (physiques et Web),
  - des éléments de classification,
  - une liste de contacts
  - ainsi que d'autres informations.



# **BusinessService (service d'affaires)**

- Les « businessServices » sont en quelque sorte les pages jaunes d'un annuaire UDDI.
- Elles décrivent de manière non technique les services proposés par les différentes organisations.
- On y trouve essentiellement
  - le nom et la description textuelle des services
  - ainsi qu'une référence à l'organisation proposant le service
  - et un ou plusieurs « bindingTemplate ».



# BindingTemplate (modèle de rattachement)

- UDDI permet de décrire des services Web utilisant HTTP, mais également des services invoqués par d'autres moyens (SMTP, FTP...).
- Les « bindingTemplates » donnent les coordonnées des services.
- Ce sont les pages vertes de l'annuaire UDDI.
- Ils contiennent notamment une description, la définition du **point d'accès** (une URL) et les éventuels « tModels » associés.



# tModel (index)

- Les « tModels » sont les descriptions techniques des services.
- UDDI n'impose aucun format pour ces descriptions qui peuvent être publiées sous n'importe quelle forme et notamment sous forme de documents textuels (XHTML, par exemple).
- C'est à ce niveau que WSDL intervient comme le vocabulaire de choix pour publier des descriptions techniques de services.



# L'interface UDDI

- L'interface UDDI est définie sous forme de documents UDDI et implémentée sous forme de service Web SOAP.
- Elle est composée des modules suivants :
  - **Interrogation inquiry** : Cette interface permet de rechercher des informations dans un répertoire UDDI.
  - **Publication** : Cette interface permet de publier des informations dans un répertoire UDDI.
  - **Sécurité** : cette interface est utilisée pour obtenir et révoquer les jetons d'authentification nécessaires pour accéder aux enregistrements protégés dans un annuaire UDDI.
  - **Contrôle d'accès et propriété custody and ownership transfer**: Cette interface permet de transférer la propriété d'informations (qui est à l'origine attribuée à l'utilisateur ayant publié ces informations) et de gérer les droits d'accès associés.
  - **Abonnement Subscription** : Cette interface permet à un client de s'abonner à un ensemble d'informations et d'être averti lors des modifications de ces informations.



# **WEB SERVICES REST FULL**

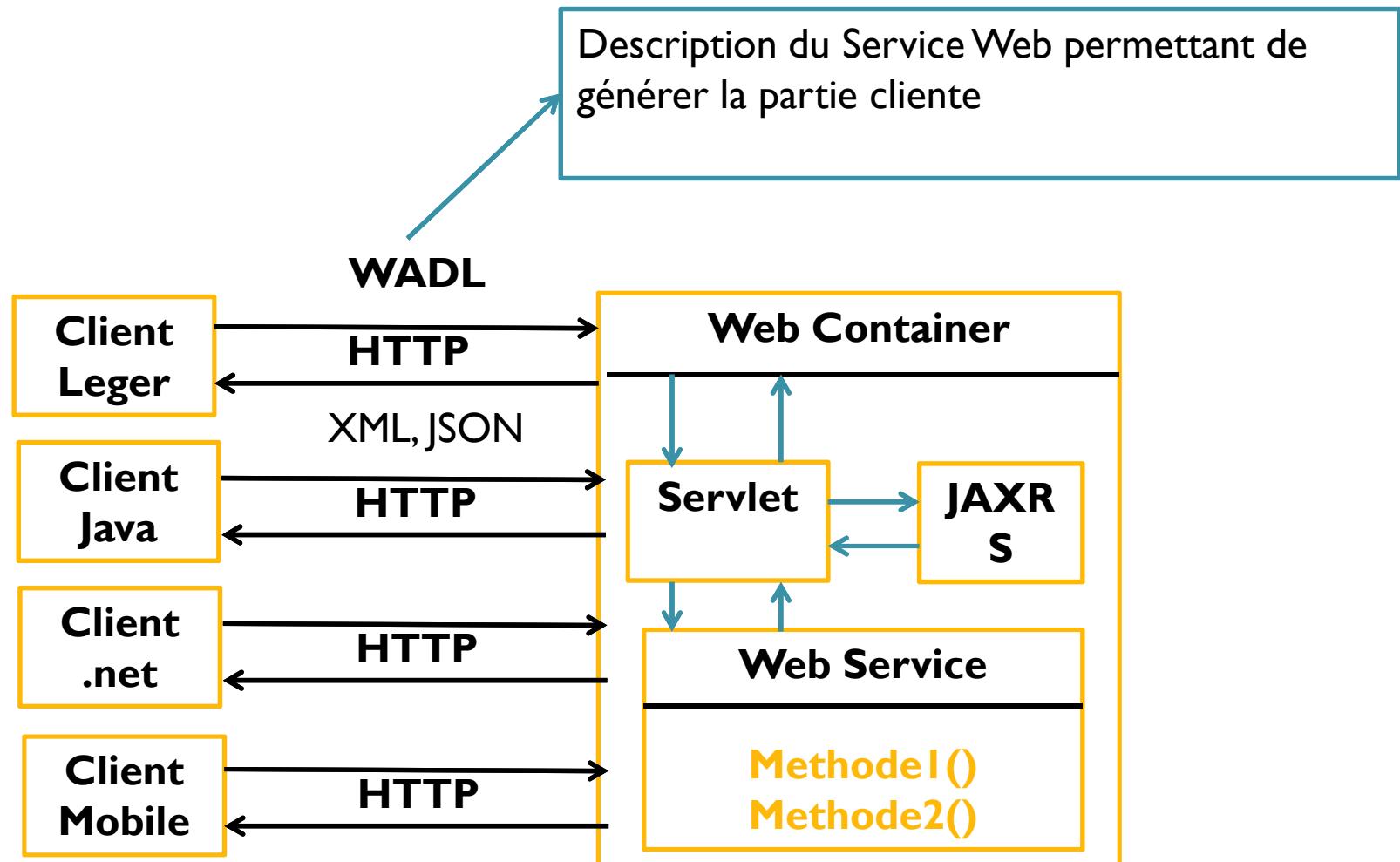
med@youssf.net



# RESTful

- **REST (REpresentational State Transfer) ou RESTful** est un style d'architecture pour les systèmes hypermédia distribués,
- Crée par Roy Fielding en 2000 dans le chapitre 5 de sa thèse de doctorat.
- **REST** est un style d'architecture permettant de construire des applications (Web, Intranet, Web Service).
- Il s'agit d'un ensemble de conventions et de bonnes pratiques à respecter et non d'une technologie à part entière.
- L'architecture REST utilise les spécifications originelles du **protocole HTTP**, plutôt que de réinventer une surcouche (comme le font SOAP ou XML-RPC par exemple).

# REST: Principe





# Règles de RESTful

- Règle n°1 :
  - l'URI comme identifiant des ressources
- Règle n°2 :
  - les verbes HTTP comme identifiant des opérations
- Règle n°3 :
  - les réponses HTTP comme représentation des ressources
- Règle n°4 :
  - les liens comme relation entre ressources
- Règle n°5 :
  - un paramètre comme jeton d'authentification



# Règle N°1 : l'URI comme identifiant des ressources

- REST se base sur les **URI (Uniform Resource Identifier)** afin d'identifier une ressource.
- Ainsi une application se doit de construire ses URI (et donc ses URL) de manière précise, en tenant compte des contraintes REST.
- Il est nécessaire de prendre en compte la hiérarchie des ressources et la sémantique des URL pour les éditer :



## Règle N°1 :

### Quelques exemples de construction d'URL avec RESTful :

- **Liste des livres**
  - NOK : <http://mywebsite.com/book>
  - OK : <http://mywebsite.com/books>
- **Filtre et tri sur les livres**
  - NOK : <http://mywebsite.com/books/filtre/policier/tri/asc>
  - OK : <http://mywebsite.com/books?filtre=policier&tri=asc>
- **Affichage d'un livre**
  - NOK : <http://mywebsite.com/book/display/87>
  - OK : <http://mywebsite.com/books/87>
- **Tous les commentaires sur un livre**
  - NOK : <http://mywebsite.com/books/comments/87>
  - OK : <http://mywebsite.com/books/87/comments>

## Règle N°1 :

Quelques exemples de construction d'URL avec RESTful :

- **Affichage d'un commentaire sur un livre**
  - NOK : <http://mywebsite.com/books/comments/87/1568>
  - OK : <http://mywebsite.com/books/87/comments/1568>
- En construisant correctement les URI, il est possible de les trier, de les hiérarchiser et donc d'améliorer la compréhension du système.
- L'URL suivante peut alors être décomposée logiquement :
  - <http://mywebsite.com/books/87/comments/1568> => **un commentaire pour un livre**
  - <http://mywebsite.com/books/87/comments> => **tous les commentaires pour un livre**
  - <http://mywebsite.com/books/87> => **un livre**
  - <http://mywebsite.com/books> => **tous les livres**



## Règle n°2 : les verbes HTTP comme identifiant des opérations

- La seconde règle d'une architecture REST est d'utiliser les verbes HTTP existants plutôt que d'inclure l'opération dans l'URI de la ressource.
- Ainsi, généralement pour une ressource, il y a 4 opérations possibles (CRUD) :
  - Créer (create)
  - Afficher (read)
  - Mettre à jour (update)
  - Supprimer (delete)
- HTTP propose les verbes correspondant :
  - Créer (create) => **POST**
  - Afficher (read) => **GET**
  - Mettre à jour (update) => **PUT**
  - Supprimer (delete) => **DELETE**



## Règle n°2 : les verbes HTTP comme identifiant des opérations

- Exemple d'URL pour une ressource donnée (un livre par exemple) :
- **Créer un livre**
  - NOK : GET <http://mywebsite.com/books/create>
  - OK : POST <http://mywebsite.com/books>
- **Afficher**
  - NOK : GET <http://mywebsite.com/books/display/87>
  - OK : GET <http://mywebsite.com/books/87>
- **Mettre à jour**
  - NOK : POST <http://mywebsite.com/books/editer/87>
  - OK : PUT <http://mywebsite.com/books/87>
- **Supprimer**
  - NOK : GET <http://mywebsite.com/books/87/delete>
  - OK : DELETE <http://mywebsite.com/books/87>



## Règle n°3 : les réponses HTTP comme représentation des ressources

- Il est important d'avoir à l'esprit que la réponse envoyée n'est pas une ressource, c'est la représentation d'une ressource.
- Ainsi, une ressource peut avoir plusieurs représentations dans des formats divers : **HTML, XML, CSV, JSON, etc.**
- C'est au client de définir quel format de réponse il souhaite recevoir via l'entête **Accept**.
- Il est possible de définir plusieurs formats.
  - **Réponse en HTML**
    - GET /books  
Host: mywebsite.com  
Accept: text/html
  - **Réponse en XML**
    - GET /books  
Host: mywebsite.com  
Accept: application/xml



## Règle n°4 : les liens comme relation entre ressources

- Les liens d'une ressource vers une autre ont tous une chose en commun : ils indiquent la présence d'une relation.
- Il est cependant possible de la décrire afin d'améliorer la compréhension du système.
- Pour expliciter cette description et indiquer la nature de la relation, l'attribut **rel** doit être spécifier sur tous les liens.
- Ainsi l'IANA donne une liste de relation parmi lesquelles :
  - contents
  - edit
  - next
  - last
  - payment
  - etc.
- La liste complète sur le site de l'IANA :
  - <http://www.iana.org/assignments/link-relations/link-relations.xml>



## Règle n°4 : les liens comme relation entre ressources

- Exemple de réponse en XML d'une liste paginée de livres :

```
<?xml>
<search>
 <link rel="self" title="self" href="http://mywebsite.com/books?q=policier&page=1&c=5" />
 <link rel="next" title="next" href="http://mywebsite.com/books?q=policier&page=2&c=5" />
 <link rel="last" title="last" href="http://mywebsite.com/books?q=policier&page=4&c=5" />
 <book>
 //...
 </book>
</search>
```



## Règle n°5 : un paramètre comme jeton d'authentification

- C'est un des sujets les plus souvent abordé quand on parle de REST : comment authentifier une requête ?
- La réponse est très simple et est massivement utilisée par des APIs renommées (Google, Yahoo, etc.) : le **jeton d'authentification**.
- Chaque requête est envoyée avec un jeton (token) passé en paramètre `$_GET` de la requête.
- Ce jeton temporaire est obtenu en envoyant une première requête d'authentification puis en le combinant avec nos requêtes.
- Ainsi, on peut construire le scénario suivant :

## Règle n°5 : un paramètre comme jeton d'authentification

- I. demande d'authentification

- GET /users/123/authenticate?pass=lkdnnssdf54d47894f5123002fds2sd360s0

```
<?xml>
<user>
 <id>123</id>
 <name>Nicolas Hachet</name>
</user>
<token>
 fsd531gfd5g5df31fdg3g3df45
</token>
```

### 2. accès aux ressources :

Cet token est ensuite utilisé pour générer un hash de la requête de cette façon :

```
hash = SHA1(token + requete)
hash = SHA1(fsd531gfd5g5df31fdg3g3df45 + "GET /books")
hash = 456894ds4q15sdq156sd1qsd1qsd156156
```

- C'est ce hash qui est passé comme jeton afin de valider l'authentification pour cette requête:

```
GET /books?user=123&hash=456894ds4q15sdq156sd1qsd1qsd156156
```



# JAX-RS : la spécification

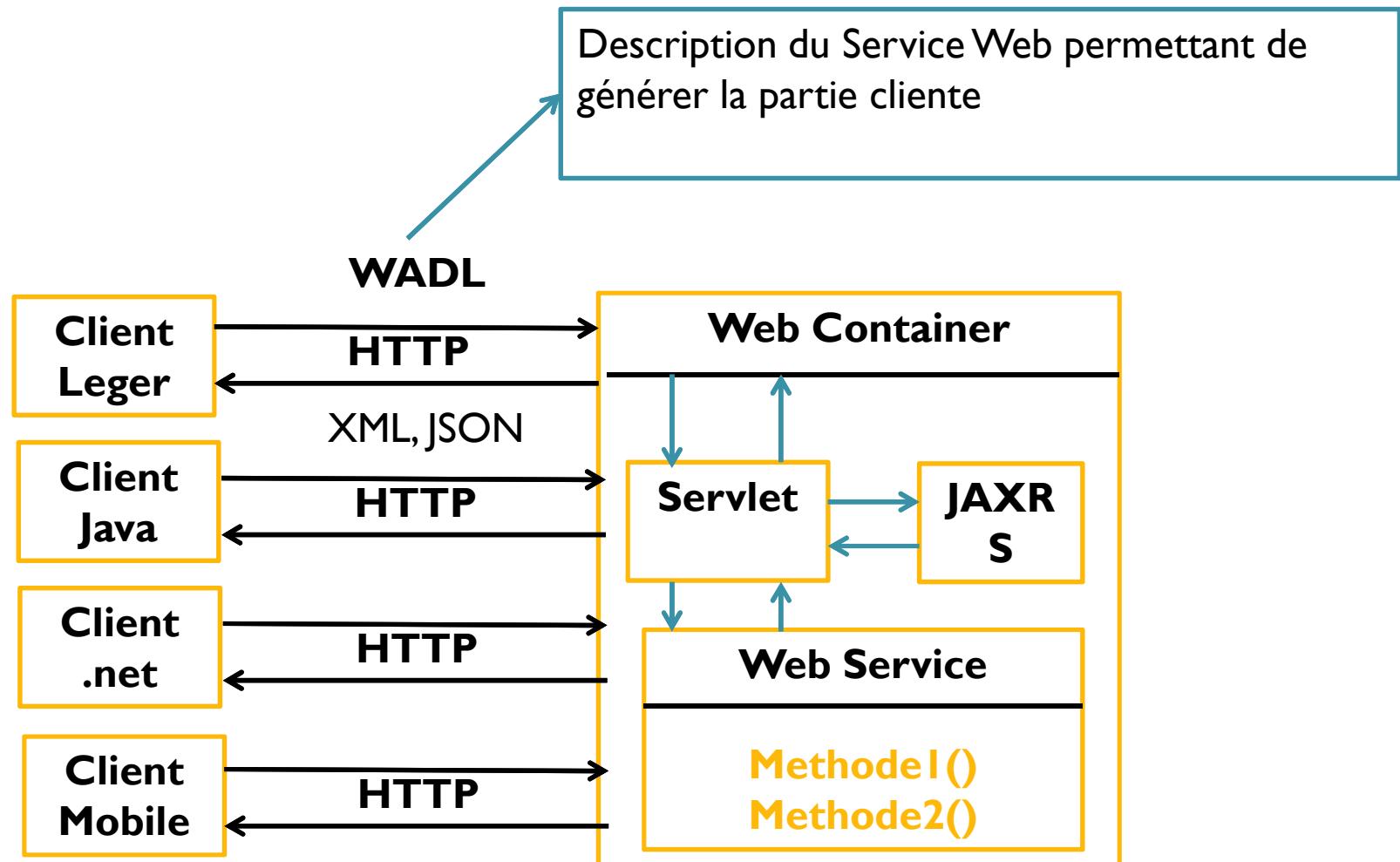
- JAX-RS est l'acronyme Java API for RESTful Web Services
- Décrise par la JSR 311 ([jcp.org/en/jsr/summary?id=311](http://jcp.org/en/jsr/summary?id=311))
- Version courante de la spécification est la 1.1
- Depuis la version 1.1, JAX-RS fait partie intégrante de la spécification Java EE 6 au niveau de la pile Service Web
- Cette spécification décrit uniquement la mise en œuvre de services Web REST coté serveur
- Le développements des Services Web REST repose sur
- l'utilisation de classes Java et d'annotations



# JAX-RS : la spécification

- Différentes implémentations de la spécification JAX-RS sont disponibles
  - **JERSEY** : implémentation de référence fournie par Oracle Site projet : <http://jersey.java.net>
  - **CXF** : fournie par Apache, la fusion entre XFire et Celtix Site projet : [cxf.apache.org](http://cxf.apache.org)
  - **RESTEasy** : fournie par JBoss Site projet : [www.jboss.org/resteasy](http://www.jboss.org/resteasy)
  - **RESTlet** : un des premiers framework implémentant REST pour Java : Site projet : [www.restlet.org](http://www.restlet.org)

# JAX RS: fonctionnement





# JSON

- **JSON** (JavaScript Object Notation – Notation Objet issue de JavaScript) est un format léger d'échange de données.
- Il est facile à lire ou à écrire pour des humains. Il est aisément analysable ou générable par des machines.
- JSON est un format texte complètement indépendant de tout langage, mais les conventions qu'il utilise seront familières à tout programmeur habitué aux langages descendant du C, comme par exemple : C lui-même, C++, C#, Java, JavaScript, Perl, Python et bien d'autres.
- Ces propriétés font de JSON un langage d'échange de données idéal.

# Exemple XML/JSON

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<comptes>
 <compte type="courant">
 <code>1</code>
 <solde>4300</solde>
 <dateCreation>2012-11-11</dateCreation>
 </compte>
 <compte type="epargne">
 <code>2</code>
 <solde>96000</solde>
 <dateCreation>2012-12-11</dateCreation>
 </compte>
</comptes>
```

JSON

```
[
 {type:"courant" , code:1, solde:4300.50,dateCreation:"2012-11-11"},

 {type:"epargne" , code:1, solde:4300.00,dateCreation:"2012-11-11"}

]
```



# Structures de JSON

- JSON se base sur deux structures :
  - Une collection de couples **nom/valeur**.
  - Divers langages la réifient par un *objet*, un enregistrement, une structure, un dictionnaire, une table de hachage, une liste typée ou un tableau associatif.
- Ces structures de données sont universelles. Pratiquement tous les langages de programmation modernes les proposent sous une forme ou une autre.

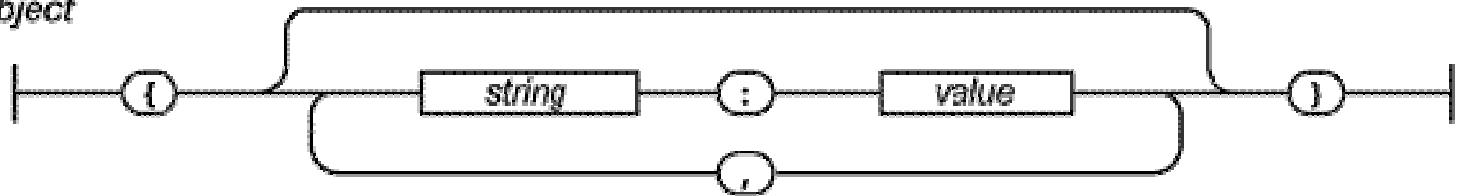


# Les structures de données JSON

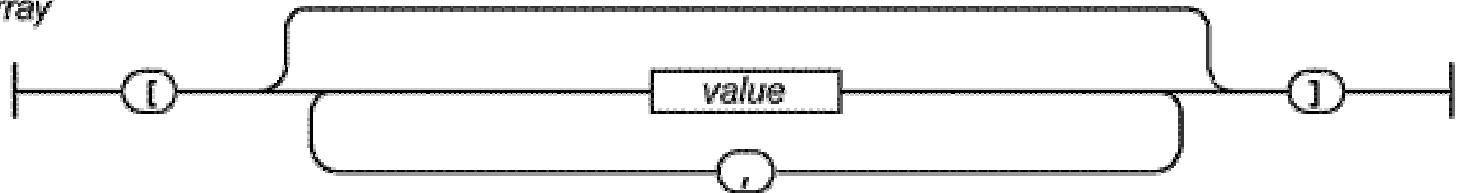
- En JSON, les structures de données prennent les formes suivantes :
  - Un **objet**, qui est un ensemble de couples nom/valeur non ordonnés. Un objet commence par { (accolade gauche) et se termine par } (accolade droite). Chaque nom est suivi de : (deux-points) et les couples nom/valeur sont séparés par , (virgule).
  - Un **tableau** est une collection de valeurs ordonnées. Un tableau commence par [ (crochet gauche) et se termine par ] (crochet droit). Les valeurs sont séparées par , (virgule).
  - Une **valeur** peut être soit une *chaîne de caractères* entre guillemets, soit un *nombre*, soit true ou false ou null, soit un *objet* soit un *tableau*. Ces structures peuvent être imbriquées.
  - Une **chaîne de caractères** est une suite de zéro ou plus caractères Unicode, entre guillemets, et utilisant les échappements avec antislash. Un caractère est représenté par une chaîne d'un seul caractère.

# Structure des données JSON

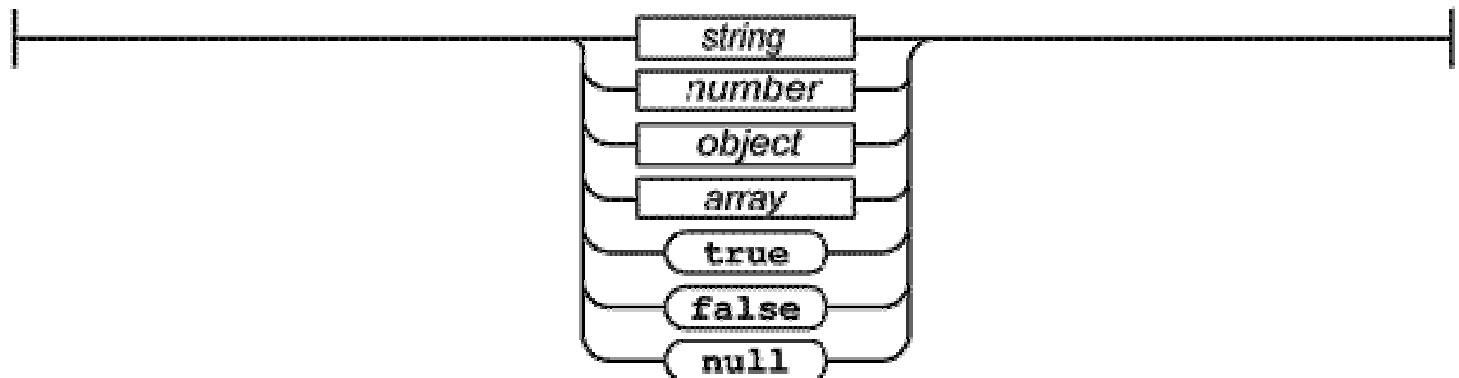
*object*



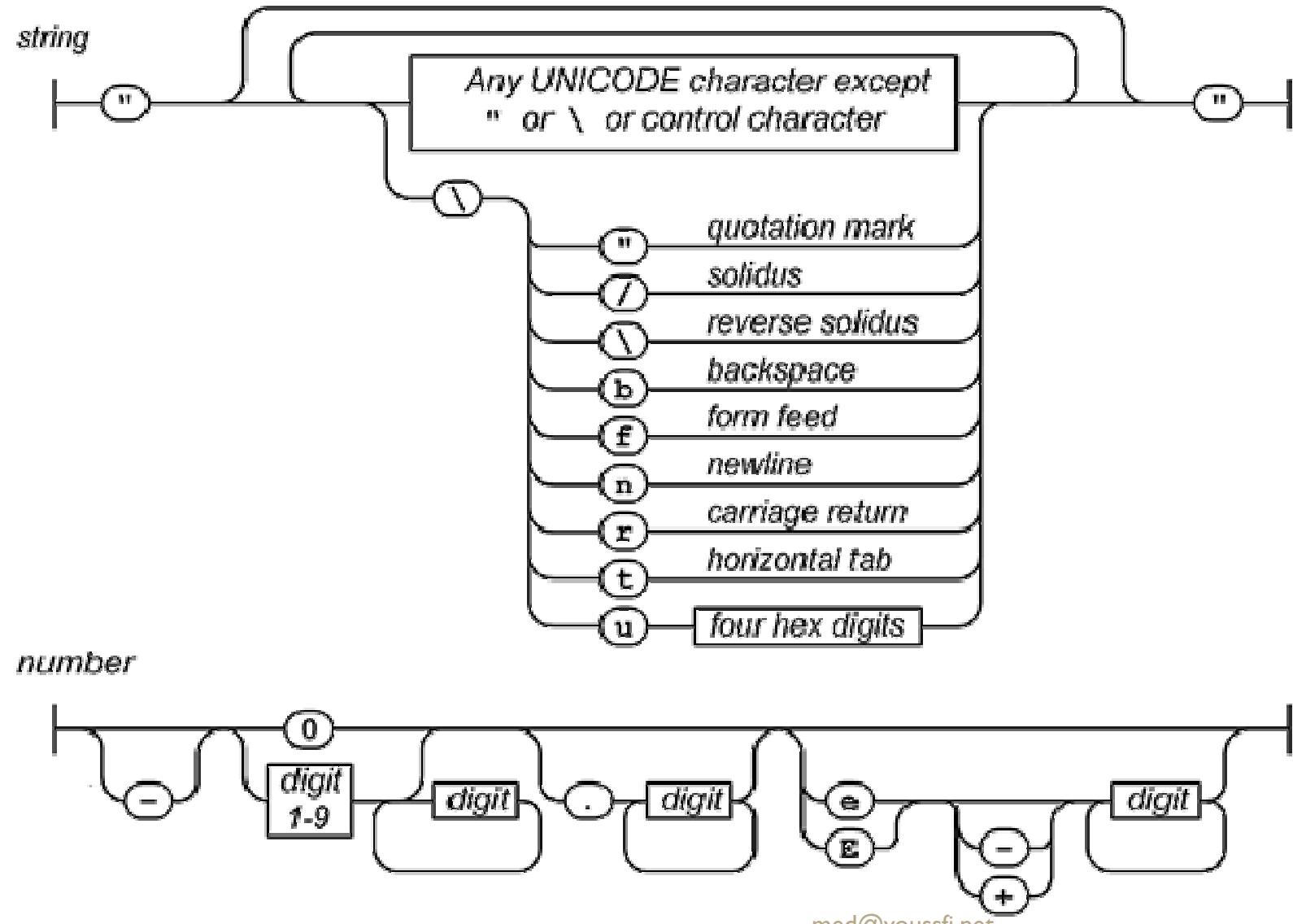
*array*



*value*



# Structure des données JSON





# Généralités sur JAX RS

- Le développement de Services Web avec JAX-RS est basé sur des POJO (Plain Old Java Object) en utilisant des annotations spécifiques à JAX-RS
- Pas description requise dans des fichiers de configuration
- Seule la configuration de la Servlet « JAX-RS » est requise pour réaliser le pont entre les requêtes HTTP et les classes Java annotées
- Un Service Web REST est déployé dans une application Web



# Généralités sur JAX RS

- Contrairement aux Services Web entendus il n'y a pas de possibilité de développer un service REST à partir du fichier de description WADL
- Seule l'approche Bottom / Up est disponible
  - Créer et annoter un POJO
  - Compiler, Déployer et Tester
  - Possibilité d'accéder au document WADL
- Le fichier de description WADL est généré automatiquement par JAX-RS (exemple : <http://host/context/application.wadl>)



# Exemple de web service RESTful

- Consulter un compte
- Consulter la liste des comptes
- Effectuer un versement
- Effectuer un retrait
- Effectuer un virement
- Supprimer un compte

# Exemple de web service JAX RS

```
package ma.bp.banqueWS.service;
import java.util.*; import javax.ws.rs.*;
import javax.ws.rs.core.MediaType;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import ma.bp.banqueWS.entities.Compte;
import ma.bp.banqueWS.metier.IBanqueMetier;

@Component
@Path("/banque")
public class BanqueRestService {
 @Autowired
 IBanqueMetier metier;

 @POST
 @Path("/comptes")
 public void addCompte(@FormParam(value="solde")double solde){
 metier.addCompte(new Compte(null, solde,new Date()));
 }
}
```

# Exemple de web service JAX RS

```
@GET
@Path("/comptes/{code}")
@Produces(value={MediaType.APPLICATION_JSON,MediaType.APPLICATION_XML})
public Compte getCompte(@PathParam(value="code")Long code){
 return metier.getCompte(code);
}

@GET
@Path("/comptes")
@Produces(value={MediaType.APPLICATION_JSON,MediaType.APPLICATION_XML})
public List<Compte> getComptes(){
 return metier.listComptes();
}

@PUT
@Path("/comptes/verser")
public void verser(@FormParam("code") Long code,@FormParam("montant") double
 montant){
 metier.verser(code, montant);
}
```

# Exemple de web service JAX RS

```
@PUT
@Path("/comptes/retirer")
public void retirer(@FormParam("code") Long code,@FormParam("montant")
double montant){
metier.retirer(code, montant);
}

@PUT
@Path("/comptes/virement")
@Produces(value={MediaType.APPLICATION_JSON,MediaType.APPLICATION_XML})
public void virement(@FormParam("cpt1") Long cpt1,
@FormParam("cpt2") Long cpt2,
@FormParam("montant") double montant){
metier.virement(cpt1, cpt2, montant);
}

@DELETE
@Path("/comptes/{code}")
public void supprimer(@PathParam("code")Long code){
metier.deleteCompte(code);
}
}
```

# Maven Dependencies

```
<!-- Jersey -->
<dependency>
 <groupId>com.sun.jersey</groupId>
 <artifactId>jersey-server</artifactId>
 <version>1.8</version>
</dependency>
<!-- Jackson -->
<dependency>
 <groupId>com.fasterxml.jackson.jaxrs</groupId>
 <artifactId>jackson-jaxrs-json-provider</artifactId>
 <version>2.3.0</version>
</dependency>
```

# Maven Dependencies

```
<!-- Jersey + Spring -->

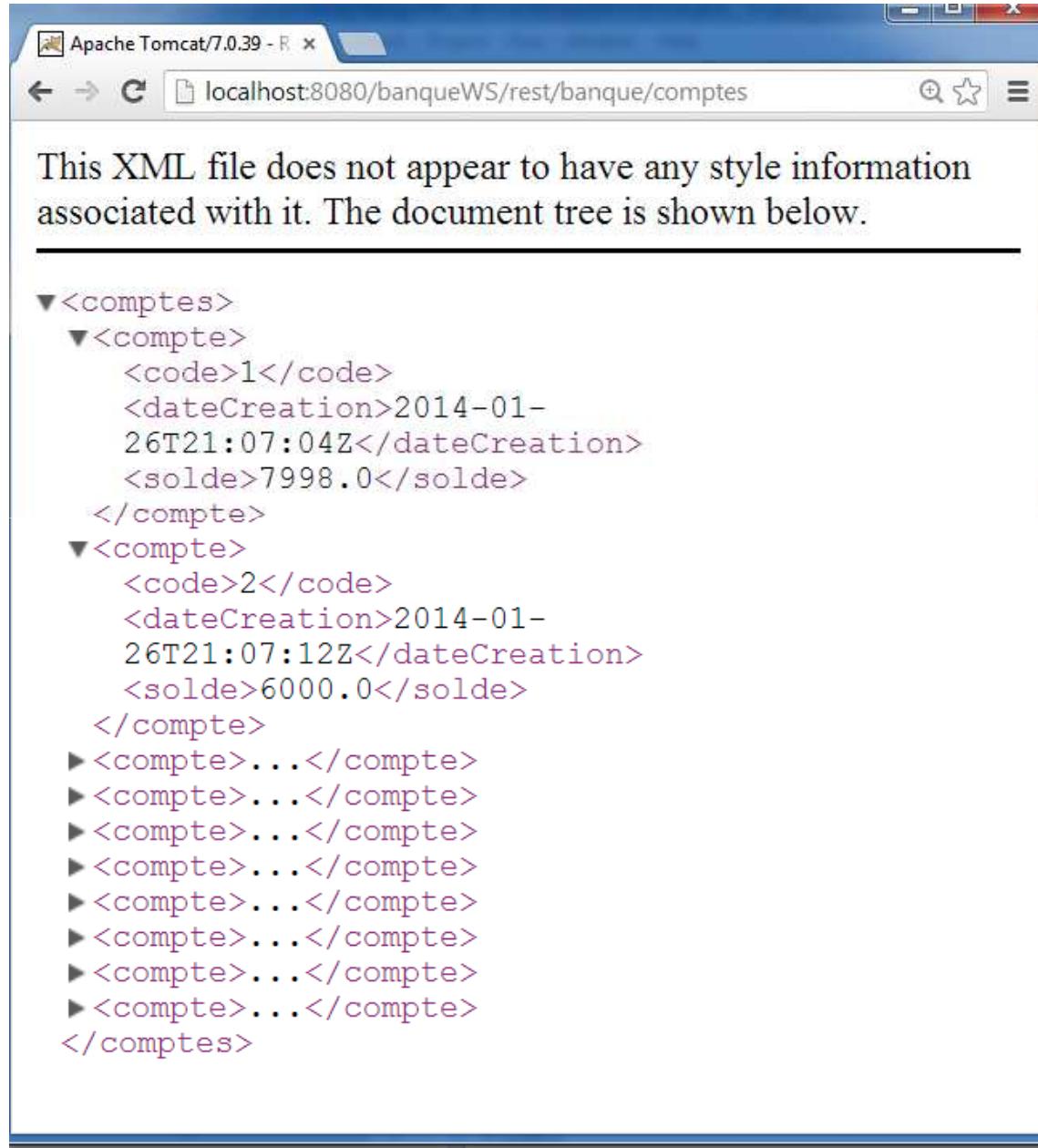
<dependency>
 <groupId>com.sun.jersey.contribs</groupId>
 <artifactId>jersey-spring</artifactId>
 <version>1.8</version>
 <exclusions>
 <exclusion>
 <groupId>org.springframework</groupId>
 <artifactId>spring</artifactId>
 </exclusion>
 <exclusion>
 <groupId>org.springframework</groupId>
 <artifactId>spring-core</artifactId>
 </exclusion>
 <exclusion>
 <groupId>org.springframework</groupId>
 <artifactId>spring-web</artifactId>
 </exclusion>
 <exclusion>
 <groupId>org.springframework</groupId>
 <artifactId>spring-beans</artifactId>
 </exclusion>
 <exclusion>
 <groupId>org.springframework</groupId>
 <artifactId>spring-context</artifactId>
 </exclusion>
 </exclusions>
</dependency>
```



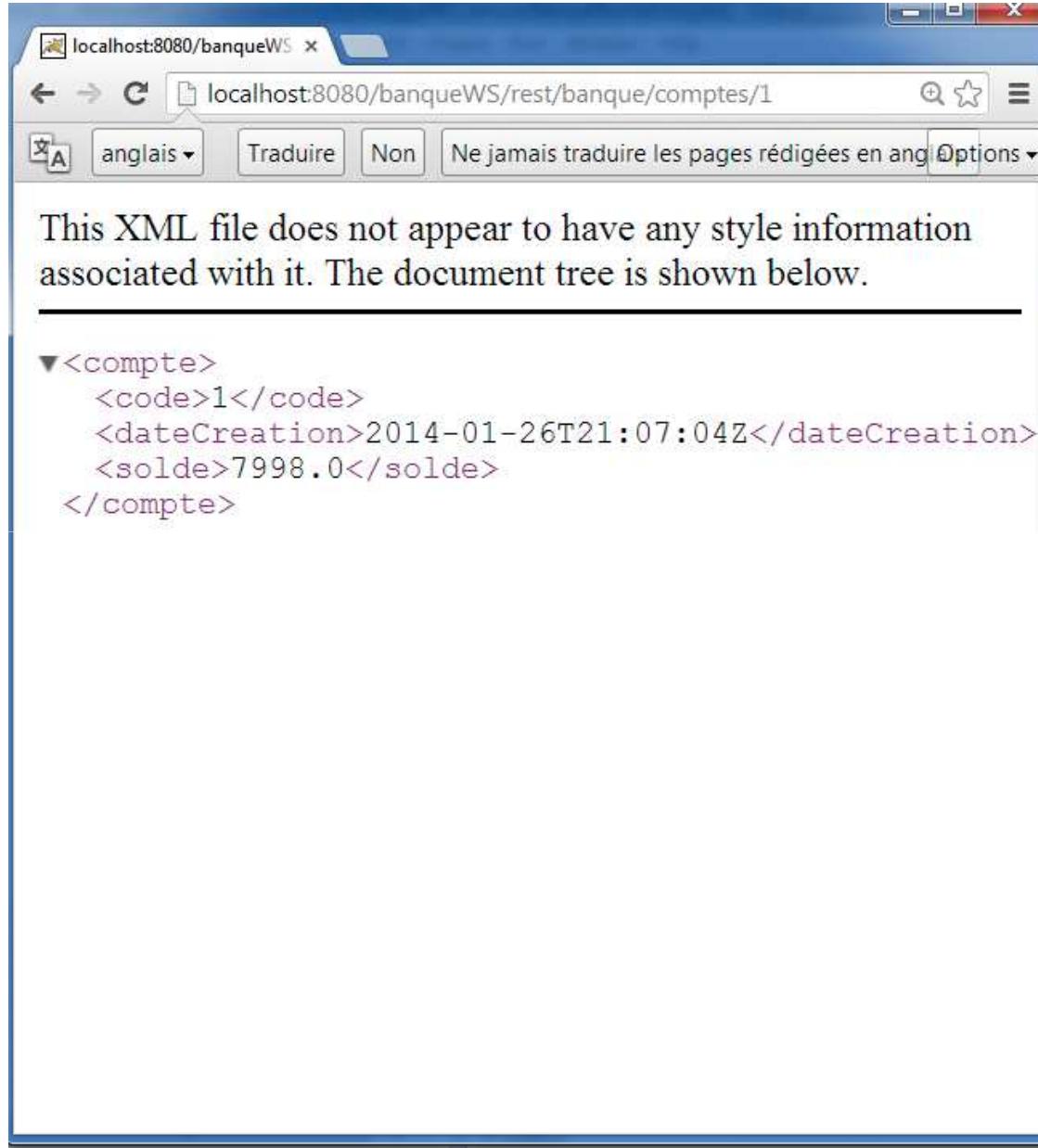
# Web.xml

```
<!-- Jersey Container-->
<servlet>
 <servlet-name>jersey-servlet</servlet-name>
 <servlet-class>
 com.sun.jersey.spi.spring.container.servlet.SpringServlet
 </servlet-class>
 <init-param>
 <param-name>com.sun.jersey.config.property.packages</param-name>
 <param-value>ma.bp.banqueWS.service</param-value>
 </init-param>
 <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
 <servlet-name>jersey-servlet</servlet-name>
 <url-pattern>/rest/*</url-pattern>
</servlet-mapping>
```

## Consulter les comptes avec une requête http GET



## Consulter un comptes avec une requête http GET



The screenshot shows a web browser window with the URL `localhost:8080/banqueWS/rest/banque/comptes/1`. The page content is an XML document:

```
<?xml version="1.0"?>
<compte>
 <code>1</code>
 <dateCreation>2014-01-26T21:07:04Z</dateCreation>
 <solde>7998.0</solde>
</compte>
```

# Récupérer la représentation json d'un compte avec une requête http GET

The screenshot shows a REST client interface with the following details:

- Request 1**: The current request being displayed.
- Method**: GET
- Endpoint**: http://localhost:8080
- Resource**: /banqueWS/rest/banque/comptes/1
- Parameters**: None
- Request Headers**:

Name	Value	Style	Level
Accept	application/json	HEADER	RESOURCE
- Response Body**:

```
1 {
2 "code": 1,
3 "solde": 7998,
4 "dateCreation": 1390770424000
5 }
```

# Récupérer la représentation json des comptes avec une requête http GET

The screenshot shows a REST client interface with the following details:

- Request 1:** Method: GET, Endpoint: http://localhost:8080, Resource: /banqueWS/rest/banque/comptes.
- Request Headers:** Accept: application/json.
- Response Body (JSON):**

```
[{"code": 1, "solde": 7998, "dateCreation": 1390770424000}, {"code": 2, "solde": 6000, "dateCreation": 1390770432000}, {"code": 3, "solde": 5000, "dateCreation": 1390770440000}, {"code": 4, "solde": 4000, "dateCreation": 1390770448000}, {"code": 5, "solde": 3000, "dateCreation": 1390770456000}, {"code": 6, "solde": 2000, "dateCreation": 1390770464000}, {"code": 7, "solde": 1000, "dateCreation": 1390770472000}, {"code": 8, "solde": 500, "dateCreation": 1390770480000}, {"code": 9, "solde": 200, "dateCreation": 1390770488000}, {"code": 10, "solde": 100, "dateCreation": 1390770496000}, {"code": 11, "solde": 50, "dateCreation": 1390770504000}, {"code": 12, "solde": 25, "dateCreation": 1390770512000}, {"code": 13, "solde": 15, "dateCreation": 1390770520000}, {"code": 14, "solde": 10, "dateCreation": 1390770528000}, {"code": 15, "solde": 5, "dateCreation": 1390770536000}, {"code": 16, "solde": 2, "dateCreation": 1390770544000}, {"code": 17, "solde": 1, "dateCreation": 1390770552000}, {"code": 18, "solde": 0.5, "dateCreation": 1390770560000}, {"code": 19, "solde": 0.25, "dateCreation": 1390770568000}, {"code": 20, "solde": 0.125, "dateCreation": 1390770576000}, {"code": 21, "solde": 0.0625, "dateCreation": 1390770584000}, {"code": 22, "solde": 0.03125, "dateCreation": 1390770592000}, {"code": 23, "solde": 0.015625, "dateCreation": 1390770600000}, {"code": 24, "solde": 0.0078125, "dateCreation": 1390770608000}, {"code": 25, "solde": 0.00390625, "dateCreation": 1390770616000}, {"code": 26, "solde": 0.001953125, "dateCreation": 1390770624000}, {"code": 27, "solde": 0.0009765625, "dateCreation": 1390770632000}, {"code": 28, "solde": 0.00048828125, "dateCreation": 1390770640000}, {"code": 29, "solde": 0.000244140625, "dateCreation": 1390770648000}, {"code": 30, "solde": 0.0001220703125, "dateCreation": 1390770656000}, {"code": 31, "solde": 0.00006103515625, "dateCreation": 1390770664000}, {"code": 32, "solde": 0.000030517578125, "dateCreation": 1390770672000}, {"code": 33, "solde": 0.0000152587890625, "dateCreation": 1390770680000}, {"code": 34, "solde": 0.00000762939453125, "dateCreation": 1390770688000}, {"code": 35, "solde": 0.000003814697265625, "dateCreation": 1390770696000}, {"code": 36, "solde": 0.0000019073486328125, "dateCreation": 1390770704000}, {"code": 37, "solde": 0.00000095367431640625, "dateCreation": 1390770712000}, {"code": 38, "solde": 0.000000476837158203125, "dateCreation": 1390770720000}, {"code": 39, "solde": 0.0000002384185791015625, "dateCreation": 1390770728000}, {"code": 40, "solde": 0.00000012020928955078125, "dateCreation": 1390770736000}, {"code": 41, "solde": 0.000000060104644775390625, "dateCreation": 1390770744000}, {"code": 42, "solde": 0.0000000300523223876953125, "dateCreation": 1390770752000}, {"code": 43, "solde": 0.000000015026161193847656, "dateCreation": 1390770760000}, {"code": 44, "solde": 0.000000007513080596923828, "dateCreation": 1390770768000}, {"code": 45, "solde": 0.000000003756540298461914, "dateCreation": 1390770776000}, {"code": 46, "solde": 0.000000001878270149230957, "dateCreation": 1390770784000}, {"code": 47, "solde": 0.0000000009391350746154785, "dateCreation": 1390770792000}, {"code": 48, "solde": 0.0000000004695675373077392, "dateCreation": 1390770800000}, {"code": 49, "solde": 0.0000000002347837686538696, "dateCreation": 1390770808000}, {"code": 50, "solde": 0.0000000001173918843269348, "dateCreation": 1390770816000}, {"code": 51, "solde": 0.0000000000586959421634674, "dateCreation": 1390770824000}, {"code": 52, "solde": 0.0000000000293479710817337, "dateCreation": 1390770832000}]
```

# Ajouter un compte avec requête http POST

The screenshot shows a REST client interface with the following details:

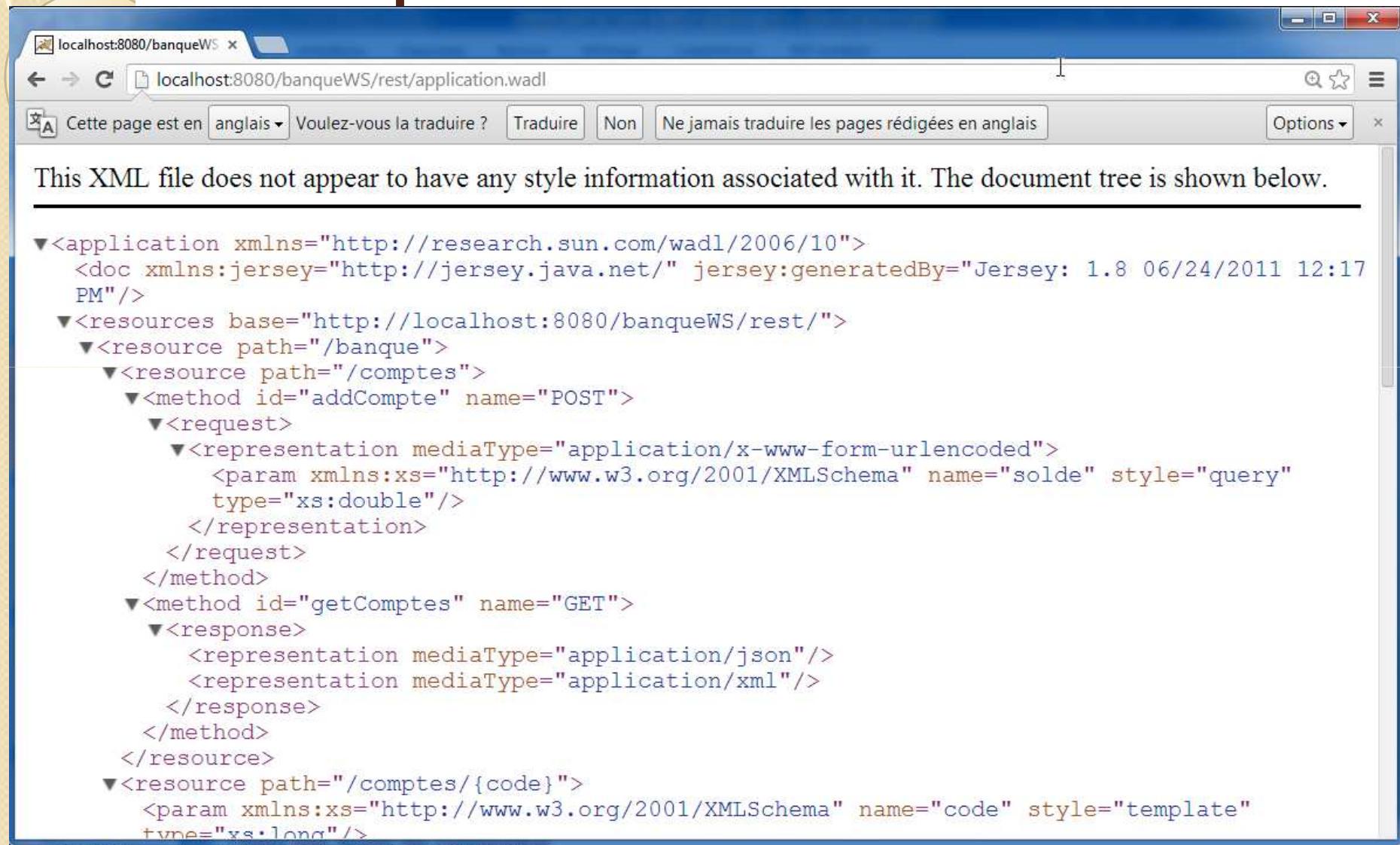
- Request 1**: The current request configuration.
- Method**: POST
- Endpoint**: http://localhost:8080
- Resource**: /banqueWS/rest/banque/comptes
- Parameters**: None
- Request**:
  - Raw**:
    - Name: Value: soldé=7000
  - XML**: None
  - JSON**: None
  - HTML**: None
  - Raw**: None
- Response**:
  - 1 <Not JSON content>



# WADL : Web Application Description Language

- WADL est un fichier XML qui permet de faire la description des services web d'une application basée sur REST.
- Le WADL est généré automatiquement par le conteneur REST.  
<http://localhost:8080/banqueWS/application.wadl>
- Les types de données structurés échangés via ce web service sont décrites par un schéma XML lié au WSDL.
- Le schéma xml de l'application REST peut être consulté par l'adresse de suivante :  
<http://localhost:8080/banqueWS/application.wadl/xsd0.xsd>

# Récupérer le WADL

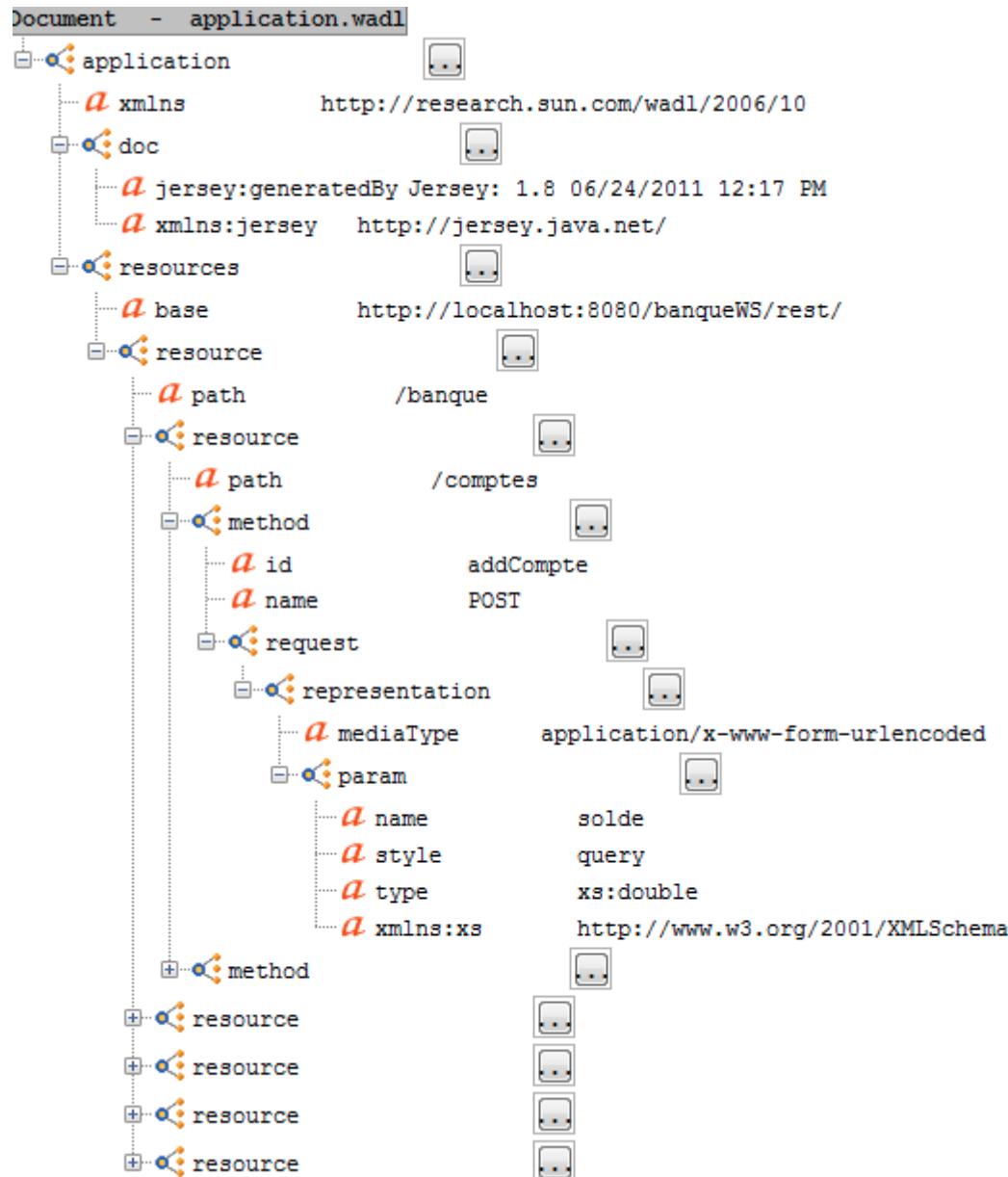


The screenshot shows a web browser window with the URL `localhost:8080/banqueWS/rest/application.wadl`. The page content is as follows:

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<application xmlns="http://research.sun.com/wadl/2006/10">
 <doc xmlns:jersey="http://jersey.java.net/" jersey:generatedBy="Jersey: 1.8 06/24/2011 12:17 PM"/>
 <resources base="http://localhost:8080/banqueWS/rest/">
 <resource path="/banque">
 <resource path="/comptes">
 <method id="addCompte" name="POST">
 <request>
 <representation mediaType="application/x-www-form-urlencoded">
 <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="solde" style="query" type="xs:double"/>
 </representation>
 </request>
 </method>
 <method id="getComptes" name="GET">
 <response>
 <representation mediaType="application/json"/>
 <representation mediaType="application/xml"/>
 </response>
 </method>
 </resource>
 <resource path="/comptes/{code}">
 <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="code" style="template" type="xs:string"/>
 </resource>
 </resource>
 </resources>
</application>
```

# Structure du DADL



# Structure du WADL

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<application xmlns="http://research.sun.com/wadl/2006/10">
 <doc xmlns:jersey="http://jersey.java.net/" jersey:generatedBy="Jersey: 1.8 06/24/2011 12:17 PM"/>
 <resources base="http://localhost:8080/banqueWS/rest/">
 <resource path="/banque">
 <resource path="/comptes">
 <method id="addCompte" name="POST">
 <request>
 <representation mediaType="application/x-www-form-urlencoded">
 <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="solde"
style="query" type="xs:double"/>
 </representation>
 </request>
 </method>
 <method id="getComptes" name="GET">
 <response>
 <representation mediaType="application/json"/>
 <representation mediaType="application/xml"/>
 </response>
 </method>
 </resource>
 </resource>
 </resources>
</application>
```

# Structure du WADL

```
<resource path="/comptes/{code}">
 <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="code" style="template" type="xs:long"/>
 <method id="getCompte" name="GET">
 <response>
 <representation mediaType="application/json"/>
 <representation mediaType="application/xml"/>
 </response>
 </method>
 <method id="supprimer" name="DELETE"/>
 </resource>
 <resource path="/comptes/verser">
 <method id="verser" name="PUT">
 <request>
 <representation mediaType="application/x-www-form-urlencoded">
 <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="code"
style="query" type="xs:long"/>
 <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="montant"
style="query" type="xs:double"/>
 </representation>
 </request>
 </method>
 </resource>
```



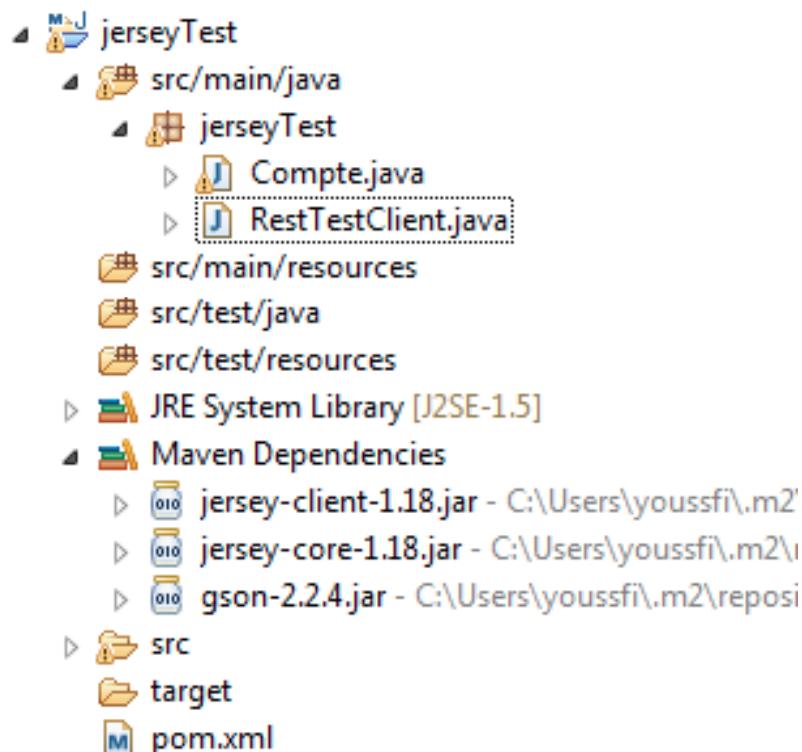
# Structure du WADL

```
<resource path="/comptes/retirer">
 <method id="retirer" name="PUT">
 <request>
 <representation mediaType="application/x-www-form-urlencoded">
 <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="code"
style="query" type="xs:long"/>
 <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="montant"
style="query" type="xs:double"/>
 </representation>
 </request>
 </method>
</resource>
```

# Structure du WADL

```
<resource path="/comptes/virement">
 <method id="virement" name="PUT">
 <request>
 <representation mediaType="application/x-www-form-urlencoded">
 <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="cptel"
style="query" type="xs:long"/>
 <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="cpte2"
style="query" type="xs:long"/>
 <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="montant"
style="query" type="xs:double"/>
 </representation>
 </request>
 </method>
</resource>
</resources>
</application>
```

# Client Java JaxRS



# Maven dependencies

```
<!-- Jersey Client -->
<dependencies>
 <dependency>
 <groupId>com.sun.jersey</groupId>
 <artifactId>jersey-client</artifactId>
 <version>1.18</version>
 </dependency>
 <!-- Google JSON API -->
 <dependency>
 <groupId>com.google.code.gson</groupId>
 <artifactId>gson</artifactId>
 <version>2.2.4</version>
 </dependency>
</dependencies>
```

# Code Java d'un client JaxRS

```
package jerseyTest;

import java.net.URI; import javax.ws.rs.core.UriBuilder;
import com.google.gson.Gson; import com.google.gson.GsonBuilder;
import com.sun.jersey.api.client.Client; import com.sun.jersey.api.client.WebResource;
import com.sun.jersey.api.client.config.ClientConfig; import
com.sun.jersey.api.client.config.DefaultClientConfig;

public class RestTestClient {

 public static void main(String[] args) {
 ClientConfig config=new DefaultClientConfig();
 Client client=Client.create(config);
 URI uri=UriBuilder.fromUri("http://localhost:8080/banqueWS/rest").build();
 WebResource service=client.resource(uri);
 WebResource path=service.path("banque").path("comptes");
 String res=path.get(String.class);
 System.out.println(res);
 Gson gson=new GsonBuilder().create();
 Compte[] cptes=gson.fromJson(res, Compte[].class);
 for(Compte cp:cptes){
 System.out.println(cp.getSolde());
 }
 }
}
```

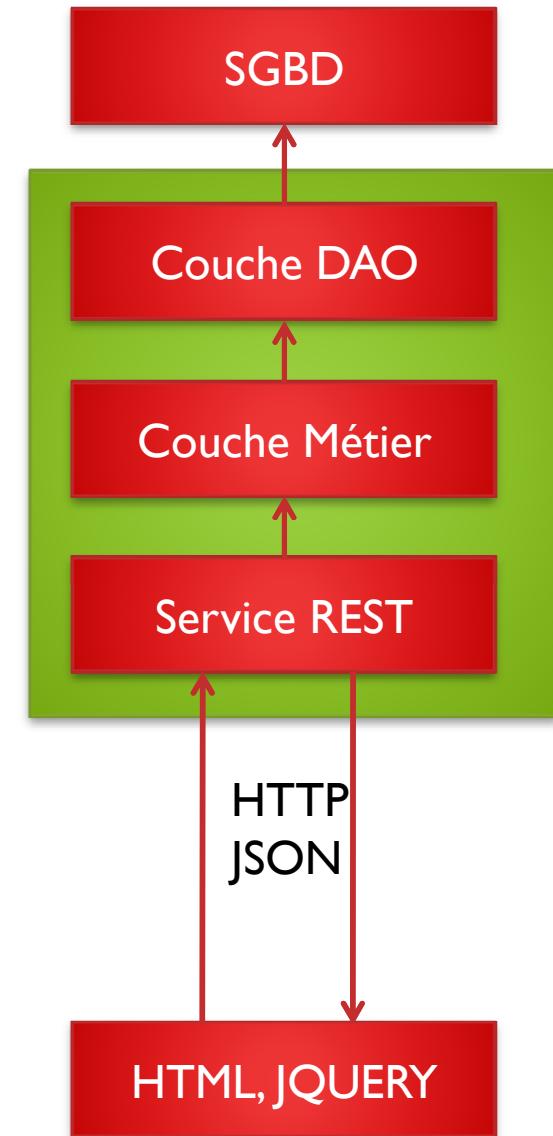
```
package jerseyTest;

public class Compte {
 private Long code;
 private double solde;
 private long dateCreation;
 // Getters et Setters
}
```

# Client HTML5, JQUERY

The screenshot shows a Java project named "banqueWS" with the following structure:

- src/main/java
- src/main/resources
- src/test/java
- src/test/resources
- JRE System Library [JavaSE-1.6]
- Maven Dependencies
- src
  - main
    - webapp
      - resources
        - jquery
          - jquery-1.8.2.js
          - jquery-ui.css
          - jquery-ui.js
        - js
          - application.js
          - index.html
      - WEB-INF
    - test
  - target
  - pom.xml



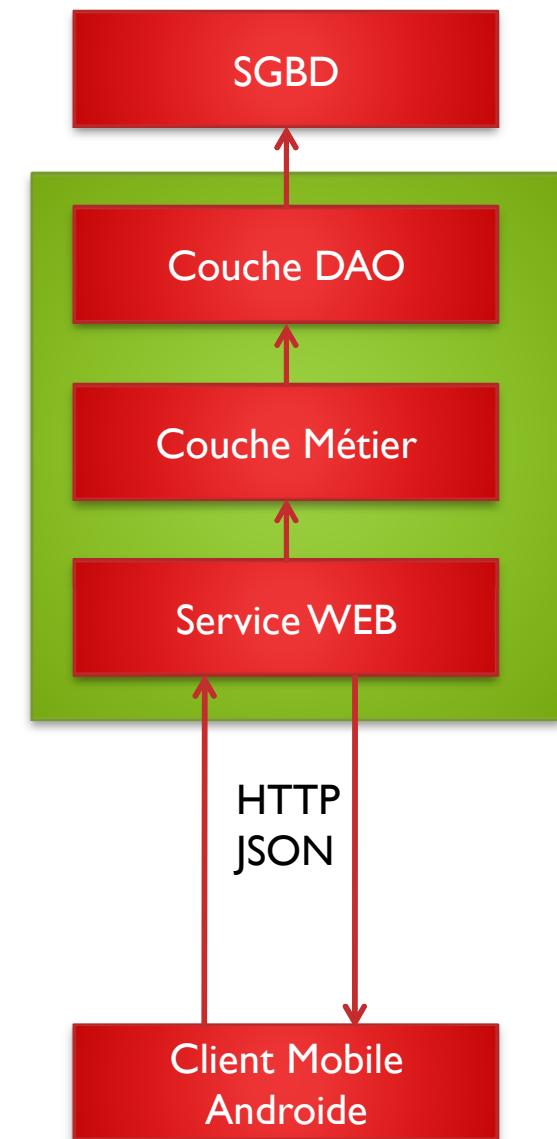
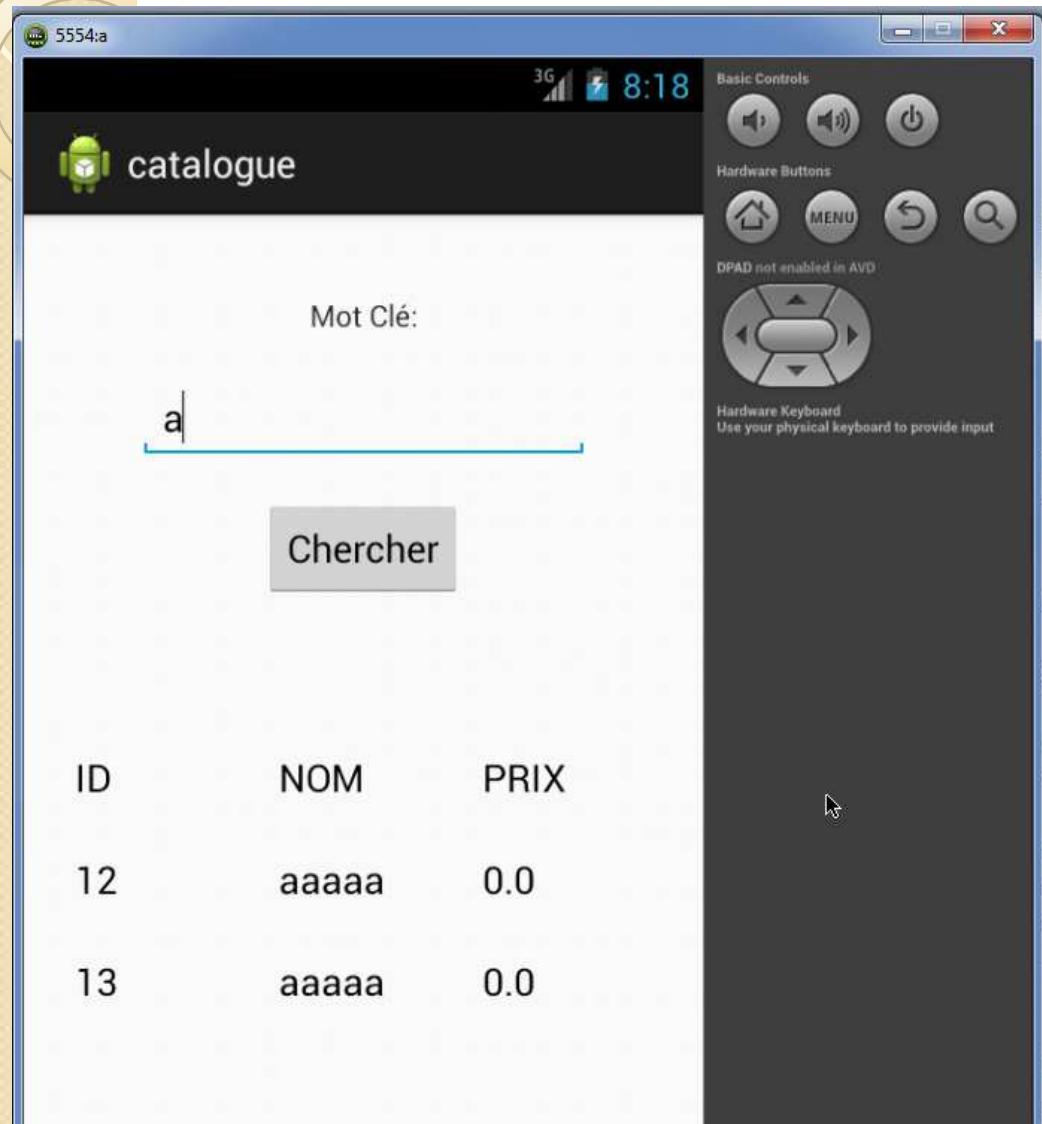
# index.html

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title></title>
 <script type="text/javascript" src="jquery/jquery-1.8.2.js"></script>
 <script type="text/javascript" src="js/application.js"></script>
</head>
<body>
 <input type="button" value="charger Les comptes" onclick="chargerComptes()">
 <div id="listComptes">
 <table>
 <thead>
 <tr>
 <th>Numéro</th><th>Solde</th><th>Date Création</th>
 </tr>
 </thead>
 <tbody id="tableBody">
 </tbody>
 </table>
 </div>
</body>
</html>
```

# Application.js

```
function chargerComptes(){
 $.getJSON("http://localhost:8080/banqueWS/rest/banque/comptes",function(data){
 $("#tableBody").html("");
 for(i in data){
 $tr=$("<tr>");
 $td=$("<td>").append(data[i]['code']);$tr.append($td);
 $td=$("<td>").append(data[i]['solde']);$tr.append($td);
 $td=$("<td>").append(data[i]['dateCreation']);$tr.append($td);
 $("#tableBody").append($tr);
 }
 });
}
```

# Client Androïde



# Vue



**Item 1      Item 2      Item 3**

Sub Item 1    Sub Item 2    Sub Item 3

**Item 4      Item 5      Item 6**

Sub Item 4    Sub Item 5    Sub Item 6

**Item 7      Item 8      Item 9**

Sub Item 7    Sub Item 8    Sub Item 9

# L'activité

```
package com.example.catalogue;

import java.io.*;import org.apache.http.*;import org.apache.http.client.*;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.json.*;import android.os.*;import android.app.Activity;
import android.view.*;import android.view.View.OnClickListener;
import android.widget.*;

public class MainActivity extends Activity implements OnClickListener {

 private EditText editTextMC; private Button buttonOK;
 private GridView gridViewProduits;

@Override
protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 buttonOK=(Button) findViewById(R.id.button1);
 editTextMC=(EditText) findViewById(R.id.editTextMC);
 gridViewProduits=(GridView) findViewById(R.id.gridView1);
 buttonOK.setOnClickListener(this);
 StrictMode.ThreadPolicy threadPolicy=new
 StrictMode.ThreadPolicy.Builder().permitAll().build();
 StrictMode.setThreadPolicy(threadPolicy);
}
```

# L'activité

```
@Override
public void onClick(View arg0) {
 Toast.makeText(getApplicationContext(), "Changement", Toast.LENGTH_LONG).show();
 StringBuilder reponseHTTP = new StringBuilder();
 HttpClient client = new DefaultHttpClient();
 String mc=editTextMC.getText().toString();
 HttpGet httpGet = new HttpGet
("http://192.168.1.79:8080/catalogue/listProduits/"+mc);
 try {
 HttpResponse response = client.execute(httpGet);
 StatusLine statusLine = response.getStatusLine();
 int statusCode = statusLine.getStatusCode();
 if (statusCode == 200) {
 HttpEntity entity = response.getEntity();
 InputStream content = entity.getContent();
 BufferedReader reader = new BufferedReader(new InputStreamReader(content));
 String line;
 while ((line = reader.readLine()) != null) {
 reponseHTTP.append(line);
 }
 }
 }
}
```



# **ANNOTATIONS JAXRS**

med@youssf.net

# @Path

- Une classe Java doit être annotée par **@path** pour qu'elle puisse être traitée par des requêtes HTTP
- L'annotation **@path** sur une classe définit des ressources appelées racines (Root Resource Class)
- La valeur donnée à **@path** correspond à une expression URI relative au contexte de l'application Web
- L'annotation **@path** peut également annoter des méthodes de la classe (facultatif)
- L'URI résultante est la concaténation de l'expression du **@path** de la classe avec l'expression du **@path** de la méthode

http://localhost:8080/TPJAXRS/banque/clients





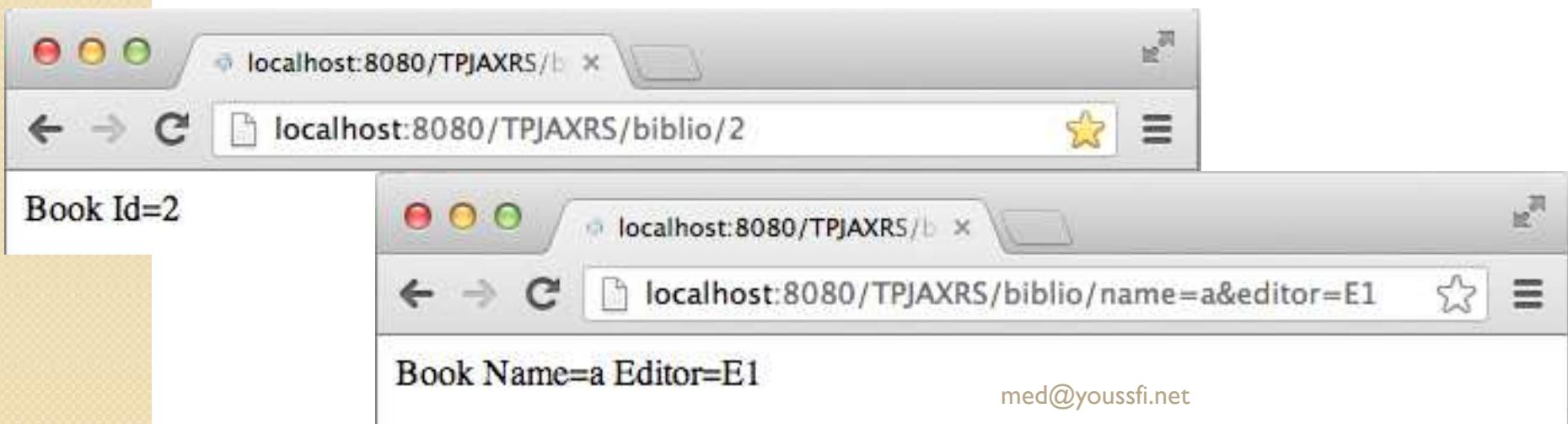
# @Path :Template Parameters

- La valeur définie dans @path ne se limite pas seulement aux expressions constantes
- Possibilité de définir des expressions plus complexes appelées **Template Paramètres**
- Pour distinguer une expression complexe dans la valeur du @path, son contenu est délimité par { ... }
- Possibilité également de mixer dans la valeur de @path des expressions constantes et des expressions complexes
- Les Template Paramètres peuvent également utiliser des expressions régulières

# @Path :Template Parameters

```
package service;
import javax.ws.rs.*;
@Path("/biblio")
public class BookResource {
 @GET
 @Path("{id}")
 public String getBookById(@PathParam("id") int id){
 return "Book Id="+id;
 }
 @GET
 @Path("name={name}&editor={editor}")
 public String getBookByNameAndEditor(@PathParam("name") String name,@PathParam("editor") String editor){
 return "Book Name="+name+ " Editor="+editor;
 }
}
```

## EXAMPLE





## Méthodes HTTP : **@GET, @POST, @PUT, @DELETE**

- L'annotation des méthodes Java permet de traiter de requêtes HTTP suivant le type de méthode (GET, POST, ...)
- Les annotations disponibles par JAX-RS sont les suivantes **@GET, @POST, @PUT, @DELETE et @HEAD**
- Ces annotations ne sont utilisables que sur des méthodes Java
- Le nom des méthodes Java n'a pas d'importance puisque c'est l'annotation employée qui précise où se fera le traitement



## Méthodes HTTP :

### **@GET, @POST, @PUT, @DELETE**

- La spécification JAX-RS, n'impose pas de respecter les conventions définies par le style REST
- Possibilité d'utiliser une requête HTTP de type GET pour effectuer une suppression d'une ressource
- Des opérations CRUD sur des ressources sont réalisées au travers des méthodes HTTP
- Généralement :
  - GET est utilisée pour consulter une ressource
  - POST est utilisée pour Ajouter une nouvelle ressource
  - PUT est utilisée pour mettre à jour une ressource
  - DELETE est utilisée pour supprimer une ressource



# Paramètres de requêtes

- JAX-RS fournit des annotations pour extraire des paramètres d'une requête
- Elles sont utilisées sur les paramètres des méthodes des ressources pour réaliser l'injection du contenu
- Liste des différentes annotations disponibles :
  - `@PathParam` : extraire les valeurs des Template Parameters
  - `@QueryParam` : extraire les valeurs des paramètres de requête
  - `@FormParam` : extraire les valeurs des paramètres de formulaire
  - `@HeaderParam` : extraire les paramètres de l'entête
  - `@CookieParam` : extraire les paramètres des cookies
  - `@Context` : extraire les informations liées aux ressources de contexte
- Une valeur par défaut peut être spécifiée en utilisant l'annotation `@DefaultValue`

# L'annotation `@PathParam`

- L'annotation `@PathParam` est utilisée pour extraire les valeurs des paramètres contenues dans les **Template Parameters**

The diagram illustrates the use of the `@PathParam` annotation. On the left, the Java code for the `BookResource` class is shown:

```
package service;
import javax.ws.rs.*;
@Path("/biblio")
public class BookResource {
 @GET
 @Path("/bookById/{id}")
 public String getBookById(@PathParam("id") int id){
 return "Book Id="+id;
 }
 @GET
 @Path("/book/name-{name}:editor-{editor}")
 public String getBookByNameAndEditor(@PathParam("name") String name,@PathParam("editor") String editor){
 return "Book Name="+name+ " Editor="+editor;
 }
}
```

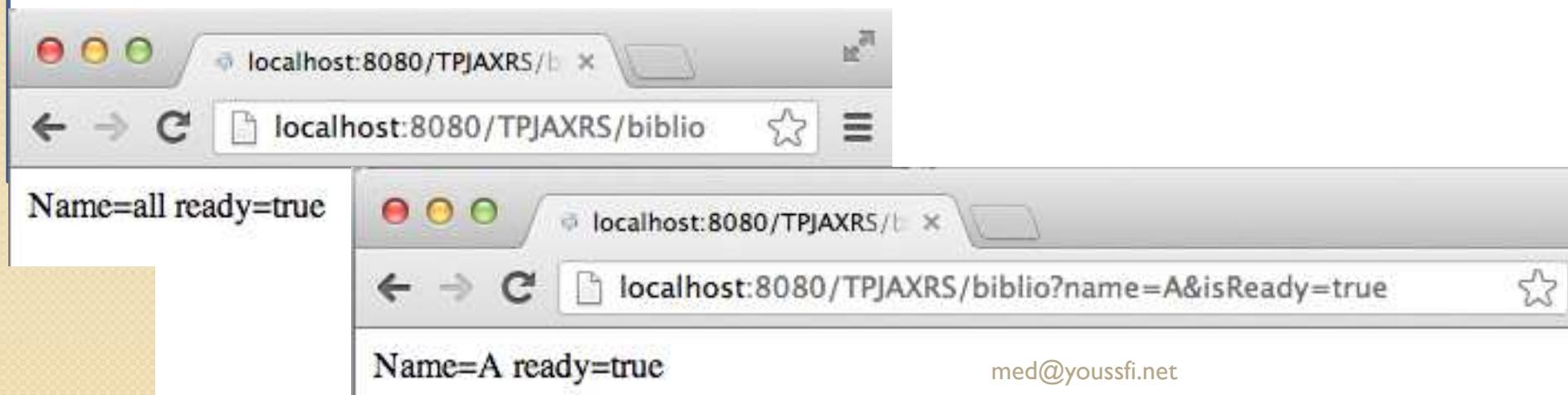
Two browser windows are displayed on the right, each showing the result of a HTTP request:

- The top browser window shows the URL `localhost:8080/TPJAXRS/biblio/bookById/4` and the response "Book Id=4". A red arrow points from the `{id}` placeholder in the URL to the value "4" in the response.
- The bottom browser window shows the URL `localhost:8080/TPJAXRS/biblio/book/name-A:editor-E1` and the response "Book Name=A Editor=E1". A red arrow points from the `{name}` and `{editor}` placeholders in the URL to the values "A" and "E1" in the response.

# L'annotation `@queryparam`

- L'annotation `@QueryParam` est utilisée pour extraire les valeurs des paramètres contenues d'une requête quelque soit son type de méthode HTTP

```
@GET
public String getQueryParameterBook(
 @DefaultValue("all") @QueryParam("name") String name,
 @DefaultValue("true") @QueryParam("isReady") boolean isReady){
 return "Name="+name+" ready="+isReady;
}
```



## Paramètres de requêtes : **@FormParam**

- L'annotation **@FormParam** est utilisée pour extraire les valeurs des paramètres contenues dans un formulaire
- Le type de contenu doit être :
  - **application/x-www-form-urlencoded**
- Cette annotation est très utile pour extraire les informations d'une requête POST d'un formulaire HTML

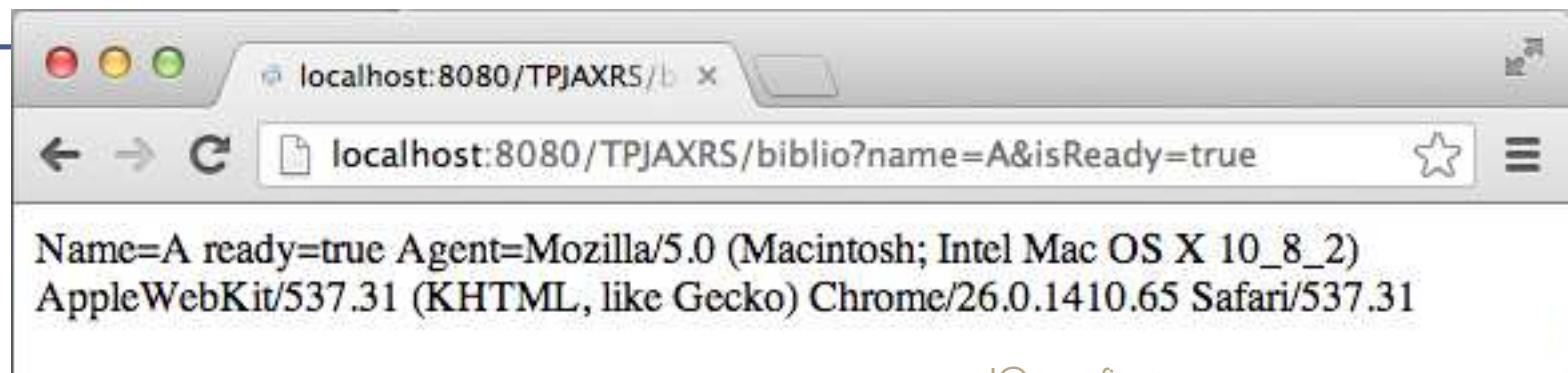
```
@POST
@Path("newBook")
@Consumes("application/x-www-form-urlencoded")
public String newBook(@FormParam("name")String name){
 return name + " Added";
}
```

## Paramètres de requêtes : `@Headerparam`

- L'annotation `@HeaderParam` est utilisée pour extraire les valeurs des paramètres contenues dans l'entête d'une requête

```
@GET
```

```
public String getQueryParameterBook(
 @DefaultValue("all") @QueryParam("name") String name,
 @DefaultValue("true") @QueryParam("isReady") boolean isReady,
 @HeaderParam("User-Agent") String userAgent){
 return "Name=" + name + " ready=" + isReady + " Agent=" + userAgent;
}
```





# Paramètres de requêtes : **@Context**

- L'annotation **@Context** permet d'injecter des objets liés au contexte de l'application
- Les types d'objets supportés sont les suivants :
  - **UriInfo** : informations liées aux URIs
  - **Request** : informations liées au traitement de la requête
  - **HttpHeaders** : informations liées à l'entête
  - **SecurityContext** : informations liées à la sécurité
- Certains de ces objets permettent d'obtenir les mêmes informations que les précédentes annotations liées aux paramètres



# Paramètres de requêtes : @Context / UriInfo

- Un objet de type **UriInfo** permet d'extraire les informations « brutes » d'une requête HTTP
- Les principales méthodes sont les suivantes :
  - **String getPath()** : chemin relatif de la requête
  - **MultivaluedMap<String, String> getPathParameters()** : valeurs des paramètres de la requête contenues dans Template Parameters
  - **MultivaluedMap<String, String> getQueryParameters()** : valeurs des paramètres de la requête
  - **URI getBaseUri()** : chemin de l'application
  - **URI getAbsolutePath()** : chemin absolu (base + chemins)
  - **URI getRequestUri()** : chemin absolu incluant les paramètres

## Paramètres de requêtes :@Context / UriInfo

```
@GET
@Path("/uriInfo/{name}")
public String uriInfo(@Context UriInfo uriInfo, @PathParam("name")String name){
 System.out.println("getPath() :" +uriInfo.getPath());
 System.out.println("getAbsolutePath():"+uriInfo.getAbsolutePath());
 System.out.println("getBaseUri():"+uriInfo.getBaseUri());
 System.out.println("getRequestUri():"+uriInfo.getRequestUri());
 System.out.println("getPathSegments():");
 List<PathSegment> pathSegments=uriInfo.getPathSegments();
for(PathSegment ps:pathSegments)
 System.out.println(ps.getPath());
 System.out.println("getPathParameters()");
 MultivaluedMap<String, String> parameters=uriInfo.getPathParameters();
for(String key:parameters.keySet())
 System.out.println(key+"="+parameters.get(key));
 return "OK";
}
```



## Paramètres de requêtes :@Context / UriInfo

`getPath() : biblio/uriInfo/a`

`getAbsolutePath(): http://localhost:8080/TPJAXRS/biblio/uriInfo/a`

`getBaseUri(): http://localhost:8080/TPJAXRS/`

`getRequestUri(): http://localhost:8080/TPJAXRS/biblio/uriInfo/a?x=1`

`getPathSegments():`

`biblio`

`uriInfo`

`a`

`getPathParameters()`

`name=[a]`



# Paramètres de requêtes :

## @Context / HttpHeaders

- Un objet de type `HttpHeader` permet d'extraire les informations contenues dans l'entête d'une requête
- Les principales méthodes sont les suivantes:
  - `Map<String, Cookie> getCookies()` : les cookies de la requête
  - `Locale getLanguage()` : le langue de la requête
  - `MultivaluedMap<String, String> getRequestHeaders()` : valeurs des paramètres de l'entête de la requête
  - `MediaType getMediaType()` : le type MIME de la requête
- A noter que ces méthodes permettent d'obtenir le même résultat que les annotations `@HeaderParam` et `@CookieParam`

# Paramètres de requêtes :@Context / UriInfo

```
@GET
@Path("httpHeaders")
public String getInformationFromHttpHeaders(@Context HttpHeaders httpheaders) {
 Map<String, Cookie> cookies = httpheaders.get Cookies();
 Set<String> currentKeySet = cookies.keySet();
 for (String currentCookie : currentKeySet) {
 System.out.println(currentCookie+"="+cookies.get(currentCookie));
 }
 MultivaluedMap<String, String> requestHeaders = httpheaders.getRequestHeaders();
 Set<String> requestHeadersSet = requestHeaders.keySet();
 for (String currentHeader : requestHeadersSet) {
 System.out.println(currentHeader+"="+requestHeaders.get(currentHeader));
 }
 return "ok";
}
```

- host=[localhost:8080]
- connection=[keep-alive]
- cache-control=[max-age=0]
- accept=[text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8]
- user-agent=[Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_8\_2) AppleWebKit/537.31 (KHTML, like Gecko) Chrome/26.0.1410.65 Safari/537.31]
- accept-encoding=[gzip,deflate,sdch]
- accept-language=[fr-FR,fr;q=0.8,en-US;q=0.6,en;q=0.4]
- accept-charset=[ISO-8859-1,utf-8;q=0.7,\*;q=0.3]



# Représentations : @Consumes, @Produces

- L'annotation **@Consumes** est utilisée pour spécifier le ou les types MIME qu'une méthode d'une ressource peut accepter
- L'annotation **@Producer** est utilisée pour spécifier le ou les types MIME qu'une méthode d'une ressource peut produire
- Possibilité de définir un ou plusieurs types MIME
- Ces annotations peuvent être portées sur une classe ou sur une méthode
- L'annotation sur la méthode surcharge celle de la classe
- Si ces annotations ne sont pas utilisées tous types MIME pourront être acceptés ou produits
- La liste des constantes des différents type MIME est disponible dans la classe **MediaType**

# Représentations : @Consumes, @Produces

## Requête HTTP

```
GET /books/details/12 HTTP/1.1
```

```
Host: localhost
```

```
Accept: text/html
```

## Réponse HTTP

Type MIME accepté par le client

```
HTTP/1.1 200 OK
```

```
Content-Type: text/html
```

```
Date:Tue, 07 May 2013 09:58:36 GMT
```

```
Server:Apache-Coyote/1.1
```

Type MIME de la réponse

```
<html>
```

```
...
```

```
</html>
```

# Représentations : @Consumes, @Produces

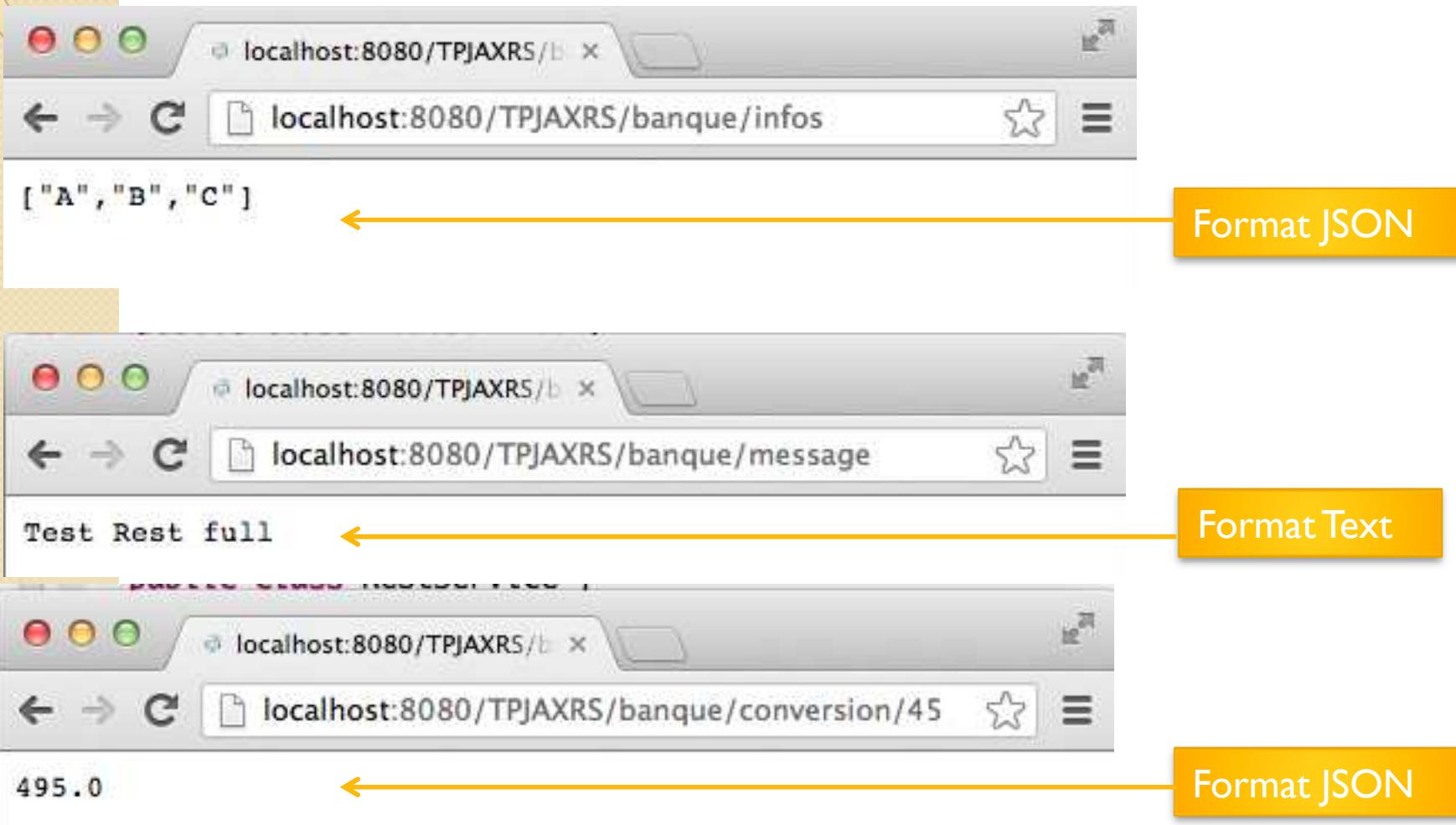
```
package service;
import java.util.*;
import javax.ws.rs.*;
import javax.ws.rs.core.MediaType;
@Path("/banque")
public class RestService {
 @GET
 @Path("/message")
 @Produces(MediaType.TEXT_PLAIN)
 public String getMessage(){
 return "Test Rest full";
 }
 @GET
 @Path("/conversion/{montant}")
 @Produces(MediaType.APPLICATION_JSON)
 public double conversion (@PathParam("montant") double mt){
 return mt*11;
 }
}
```

```
@GET
@Path("/infos")
@Produces(MediaType.APPLICATION_JSON)
public List<String> getInfos(){
 List<String> res=new ArrayList<String>();
 res.add("A");res.add("B");res.add("C");
 return res;
}

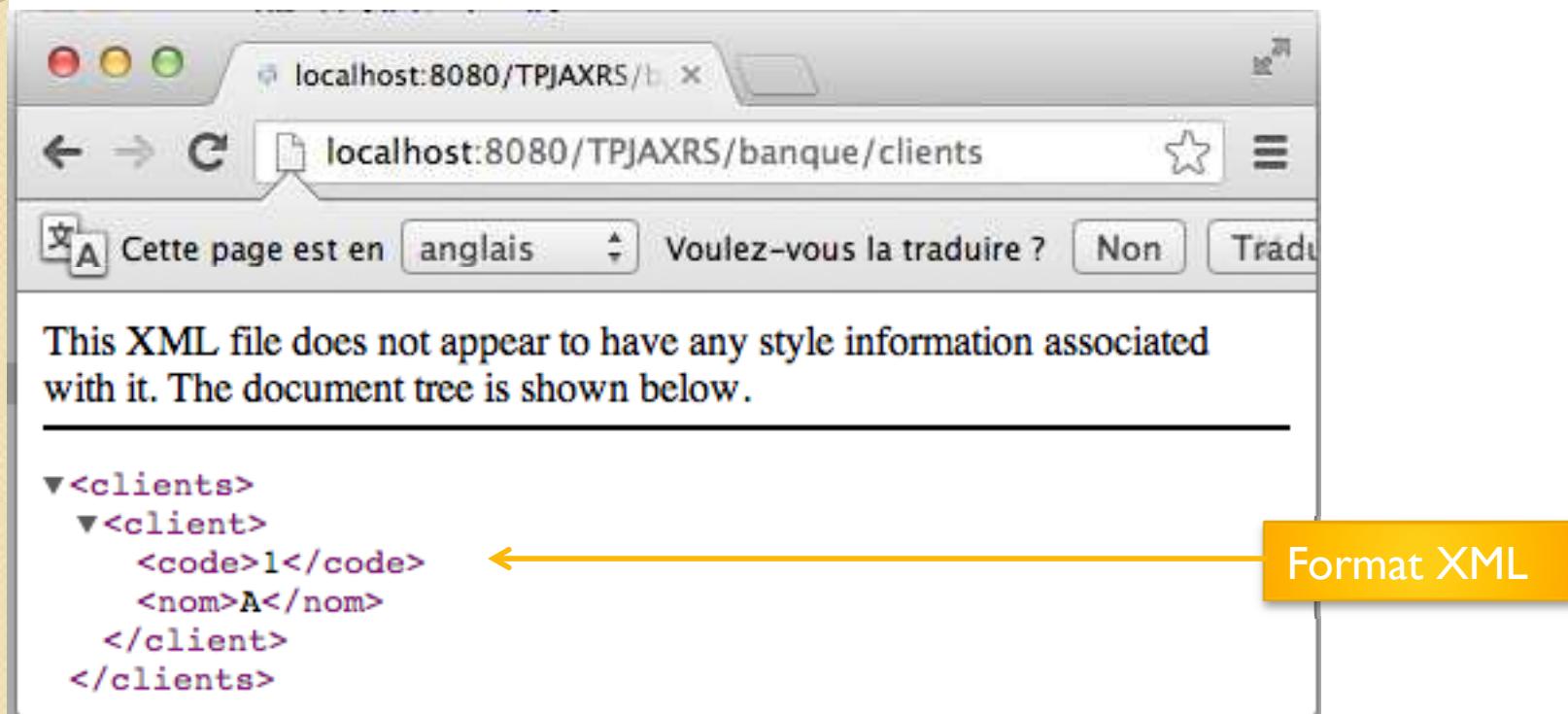
@GET
@Path("/clients")
@Produces(MediaType.APPLICATION_XML)
public List<Client> getClients(){
 List<Client> res=new ArrayList<Client>();
 res.add(new Client(1,"A"));
 return res;
}
```

```
package service;
import java.io.Serializable;import javax.xml.bind.annotation.XmlRootElement;
@XmlRootElement
public class Client implements Serializable {
 private int code;
 private String nom;
 // Getters , Setters et Constructeurs
}
```

# Accès au service avec un browser



# Accès au service avec un browser





# Gestion du contenu : statut des réponses

- Lors de l'envoie de la réponse au client un code statut est retourné
- **Réponse sans erreur :** Les statuts des réponses sans erreur s'échelonnent de 200 à 399
  - Le code est 200 « **OK** » pour les services retournant un contenu non vide
  - Le code est 204 « **No Content** » pour les services retournant un contenu vide
- **Réponse avec erreur :** Les statuts des réponses avec erreur s'échelonnent de 400 à 599
  - Une ressource non trouvée, le code de retour est 404 « **Not Found** »
  - Un type MIME en retour non supporté, 406 « **Not Acceptable** »
  - Une méthode HTTP non supportée, 405 « **Method Not Allowed** »

# Response

- JAX-RS facilite la construction de réponses en permettant de
  - de choisir un code de retour
  - de fournir des paramètres dans l'entête
  - de retourner une URI, ...
- Les réponses complexes sont définies par la classe **Response** disposant de méthodes abstraites non utilisables directement
  - **Object getEntity()** : corps de la réponse
  - **int getStatus()** : code de retour
  - **MultivalueMap<String, Object> getMetaData()** : données de l'entête
- Les informations de ces méthodes sont obtenues par des méthodes statiques retournant des **ResponseBuilder**
- Utilisation du patron de conception **Builder**

# Principales méthodes de Response

- `ResponseBuilder created(URI location)` : Modifie la valeur de Location dans l'entête, à utiliser pour une nouvelle ressource créée
- `ResponseBuilder notModified()` : Statut à « Not Modified »
- `ResponseBuilder ok()` : Statut à « Ok »
- `ResponseBuilder serverError()` : Statut à « Server Error »
- `ResponseBuilder status(Response.Status)` : définit un statut particulier défini dans `Response.Status`

# Méthodes de ResponseBuilder

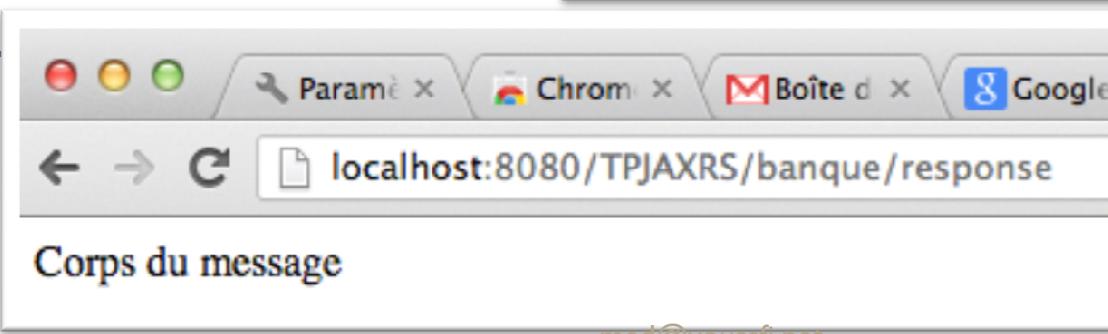
- `Response build()` : crée une instance de Response
- `ResponseBuilder entity(Object value)` : modifie le contenu du corps
- `ResponseBuilder header(String, Object)` : modifie un paramètre de l'entête

# Exemple de Response

```
@Path("response")
@GET
public Response gerReponse(){
 return Response
 .status(Response.Status.OK)
 .header("param1", "valeur1")
 .header("param2", "valeur2")
 .entity("Corps du message")
 .build();
}
```

▼ Response Headers [view source](#)

Content-Type: text/html  
Date: Thu, 09 May 2013 17:18:44 GMT  
param1: valeur1  
param2: valeur2  
Server: Apache-Coyote/1.1  
Transfer-Encoding: chunked



# Exemple de Response

```
@GET
@Path("/comptes/v2/{code}")
@Produces(value={MediaType.APPLICATION_JSON,MediaType.APPLICATION_XML})
public Response compte(@PathParam(value="code")Long code){
 Compte cp=metier.getCompte(code);
 return Response
 .status(Response.Status.OK)
 .entity(cp)
 .build();
}
```

localhost:8080/banqueWS/rest/banque/comptes/v2/1

```
{"code":1,"solde":7998.0,"dateCreation":1390770424000}
```

localhost:8080/banqueWS/rest/banque/comptes/v2/1

This XML file does not appear to have any style information associated with it.

```
▼<compte>
 <code>1</code>
 <dateCreation>2014-01-26T21:07:04Z</dateCreation>
 <solde>7998.0</solde>
</compte>
```



# Développement Client : la création de la requête

- La création de la requête s'appuie sur la patron Builder
- Création d'une chaîne d'appel de méthodes dont le type de retour est **WebResource** ou **WebResource.Builder**
- La chaîne d'appel se termine par les méthodes correspondant aux méthodes HTTP (GET, POST, ...)
- La classe **WebResource.Builder** contient les méthodes de terminaison
  - <T> **get**(Class<T> c) : appelle méthode GET avec un type de retour T
  - <T> **post**(Class<T> c, Object entity) : appelle méthode POST en envoyant un contenu dans la requête
  - <T> **put**(Class<T> c, Object entity) : appelle méthode PUT en envoyant un contenu dans la requête
  - <T> **delete**(Class<T> c, Object entity) : appelle méthode DELETE en envoyant un contenu dans la requête



# Développement Client : la création de la requête

- La classe **WebResource** fournit des méthodes pour construire l'entête de la requête
- Principales méthodes de **WebResource**:
  - WebResource **path(String)** : définition d'un chemin
  - WebResource **queryParam(String key, String val)** : paramètre requête
  - Builder **accept(MediaType)** : type supporté par le client
  - Builder **header(String name, Object value)** : paramètre entête
  - Builder **cookie(Cookie cookie)** : ajoute un cookie
- Possibilité d'appeler plusieurs fois la même méthode

# Exemple de Client java REST

```
import java.net.URI;
import javax.ws.rs.core.UriBuilder;
import com.sun.jersey.api.client.Client;
import com.sun.jersey.api.client.WebResource;
import com.sun.jersey.api.client.config.ClientConfig;
import com.sun.jersey.api.client.config.DefaultClientConfig;
public class ClientJaxRS {
 public static void main(String[] args) {
 ClientConfig config=new DefaultClientConfig();
 Client client=Client.create(config);
 URI uri=UriBuilder.fromUri("http://localhost:8080/TPJAXRS/").build();
 WebResource service=client.resource(uri);
 WebResource path=service.path("banque").path("conversion").path("5");
 String res=path.get(String.class);
 System.out.println(res);
 }
}
```

# Client java REST: Méthodes POST et PUT

// Requête POST avec un paramètre

```
WebResource path2=service.path("banque").path("comptesParClient").queryParam("cc", "45");
String res2=path2.post(String.class);
System.out.println(res2);
```

// Requête PUT pour envoyer un objet Client

```
WebResource path3=service.path("banque").path("newClient");
String res3=path3.put(String.class,new service.Client(3, "ABC"));
System.out.println(res3);
```

```
package service;
import java.io.Serializable;
import javax.xml.bind.annotation.XmlRootElement;
@XmlRootElement
public class Client implements Serializable {
 private int code;
 private String nom;
 // Constructeurs, Getters et Setters
}
```

# Client java REST: ClientResponse

```
System.out.println("-----");
WebResource path4=service.path("banque").path("response");
ClientResponse res4=path4.get(ClientResponse.class);
MultivaluedMap<String, String> headers=res4.getHeaders();
System.out.println(headers.getFirst("param1"));
System.out.println(headers.getFirst("param2"));
System.out.println(res4.getEntity(String.class));
```



# Problème

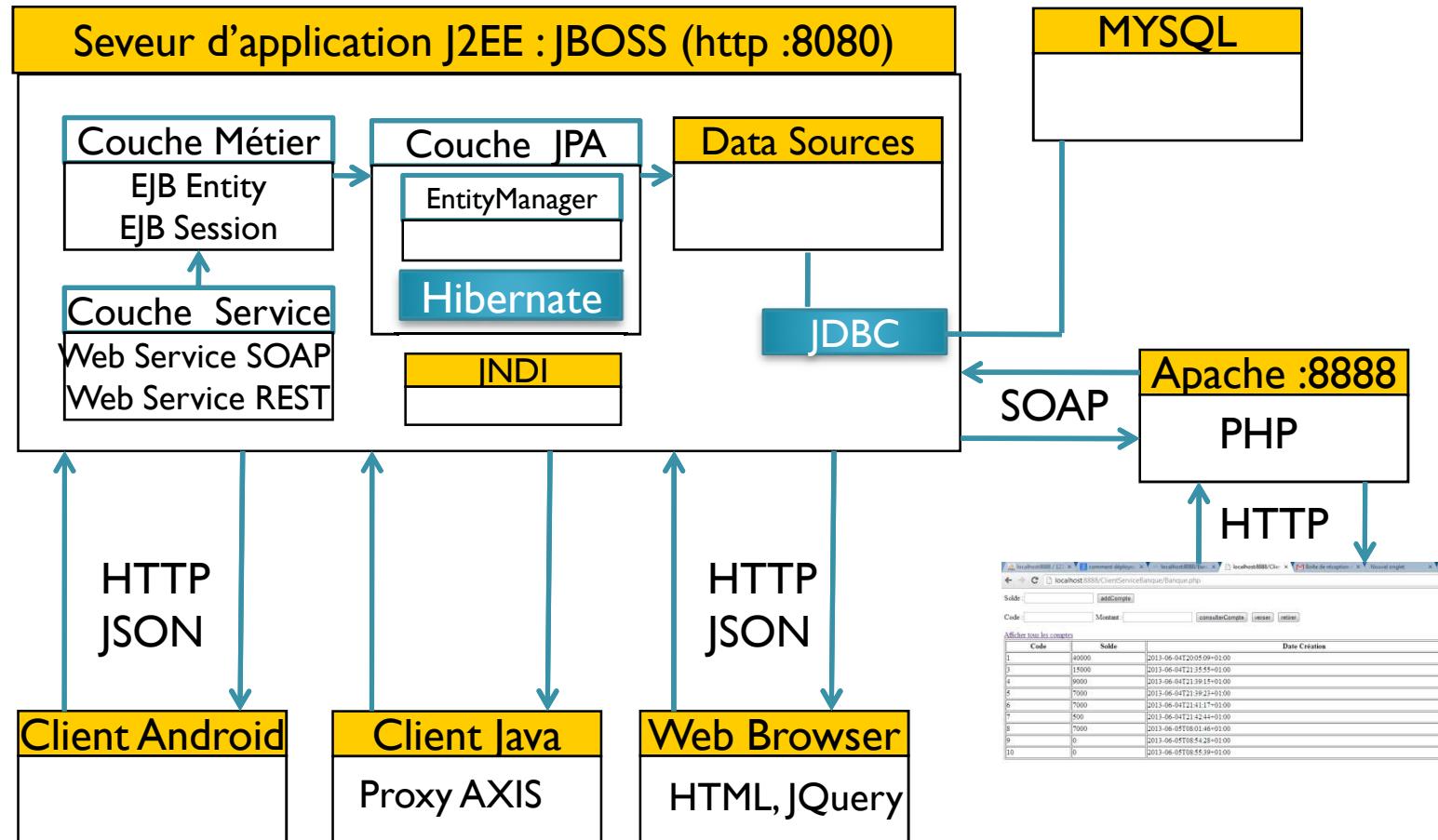
- On souhaite Créer une application distribuée qui permet de gérer le transport des cargaisons contenant des marchandises. Chaque cargaison contient plusieurs marchandises. Il existe deux types de cargaisons : les cargaisons routières et aériennes.
- Chaque marchandise est définie par un numéro de type Long (Auto Incrémenté), le nom de la marchandise, son poids et son volume.
- Chaque cargaison est définie par une référence de type String, la distance de parcours, la date de livraison.
- Une cargaison routière est une cargaison qui possède en plus la température de conservation.
- Une cargaison aérienne est une cargaison qui possède en plus un poids maximal qui ne doit pas être dépassé.



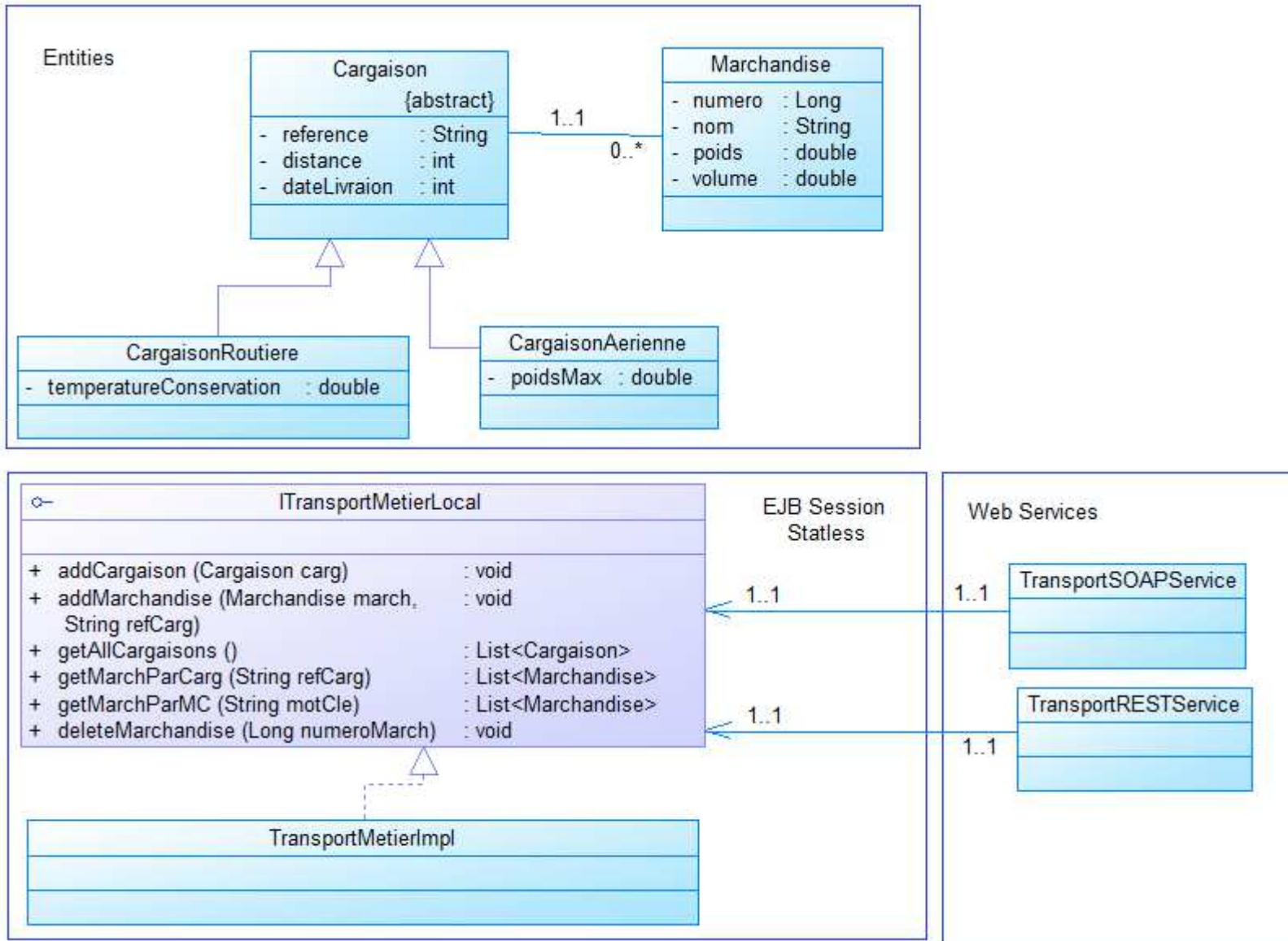
# Problème

- L'application se compose de des couches suivantes :
- Une couche métier basées sur EJB (Entity et Session)
- Une couche service qui permet un accès distant aux services métiers qui contient deux types de services :
  - Un web service SOAP avec JAXWS
  - Un web service REST FULL basé sur JAXRS
- Pour accéder aux services de cette application, d'autres entreprises peuvent développer des applications clientes de différents types :
  - Un client Mobile Androide
  - Un client PHP qui permet de consulter les marchandises dont le nom contient un mot clé. Cette application fait appel à la couche métier via le service SOAP
  - Un client Java qui permet d'ajouter les cargaisons et les marchandises en faisant appel au service REST FULL (Format généré : JSON)
  - Un client WEB (HTML, CSS, JQUERY) qui permet de consulter les marchandises d'une cargaison sélectionnée dans une liste déroulante. Cette application s'appuie sur le service REST FULL

# Architecture



# Diagramme de classes

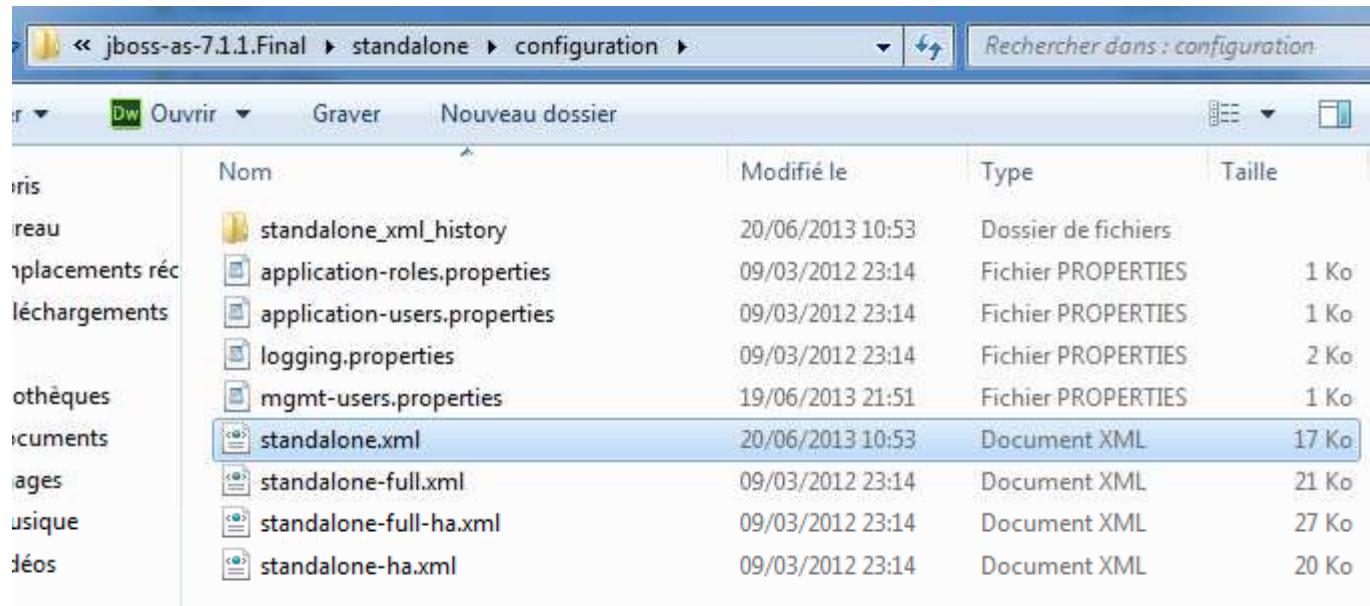




# Base de données

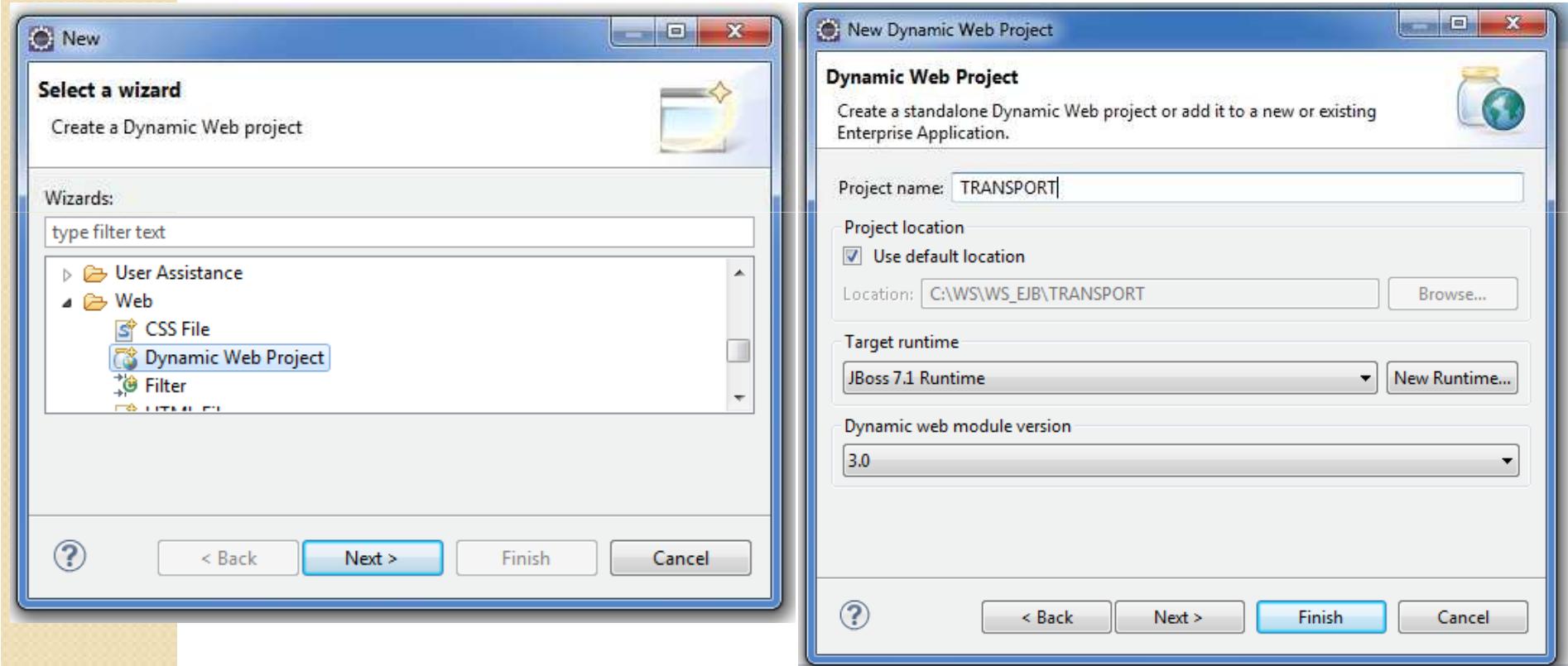
- Démarrer Easy PHP
  - MySQL
  - Apache
  - PHP
  - PhpMyAdmin
- Créer la base de données  
**DB\_TRANSPORT**
- Les tables seront créées par le framework de mapping objet relationnel (Hibernate)

# Déployer le DataSource



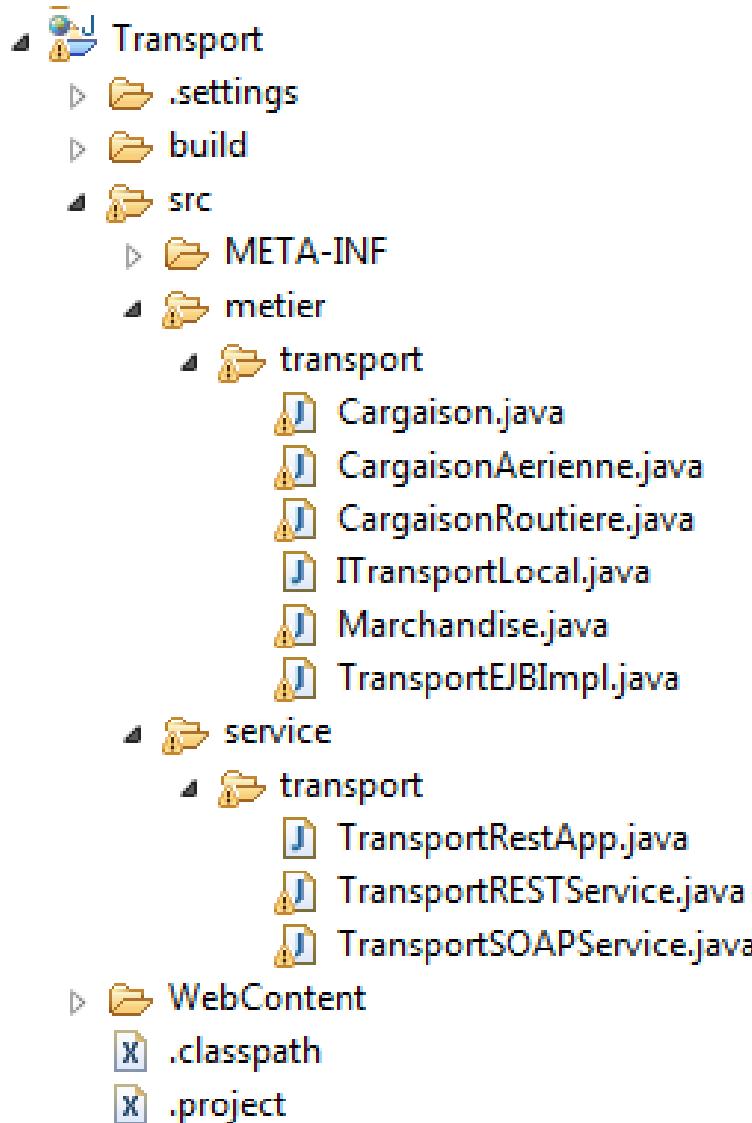
```
<datasource jndi-name="java:/dsTR" pool-name="dsTR" enabled="true">
 <connection-url>jdbc:mysql://127.0.0.1:3306/DB_TRANSPORT</connection-url>
 <driver-class>com.mysql.jdbc.Driver</driver-class>
 <driver>mysql</driver>
 <security>
 <user-name>root</user-name>
 </security>
</datasource>
```

# Projet Web Dynamique



med@youssfi.net

# Structure du projet



# Entity : Marchandise

```
package metier.transport; import java.io.Serializable; import javax.persistence.*;
@Entity
@Table(name="MARCHANDISES")
public class Marchandise implements Serializable {
 @Id
 @GeneratedValue(strategy=GenerationType.IDENTITY)
 @Column(name="NUMERO")
 private Long numero;
 @Column(name="NOM")
 private String nom;
 private double poids;
 private double volume;
 @ManyToOne
 @JoinColumn(name="REF_CARG")
 private Cargaison cargaison;
 public Marchandise(String nom, double poids, double volume) {
 this.nom = nom;this.poids = poids; this.volume = volume;
 }
 public Marchandise() {super(); }
 // Getters et Setters
}
```

# Entity : Cargaison

```
package metier.transport;import java.io.*;import java.util.*;
import javax.persistence.*;import javax.xml.bind.annotation.*;
import org.codehaus.jackson.annotate.JsonIgnore;

@Entity
@Table(name="CARGAISONS")
@Inheritance(strategy=InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name="TYPE_CARG")
@XmlAccessorType(XmlAccessType.FIELD)
@XmlSeeAlso({CargaisonAerienne.class,CargaisonRoutiere.class})
public abstract class Cargaison implements Serializable{
 @Id
 @Column(name="REF_CARG")
 private String reference; private int distance; private Date dateLivraison;
 @OneToMany(mappedBy="cargaison",fetch=FetchType.LAZY)
 @XmlTransient
 private Collection<Marchandise> marchandises;
 public Cargaison(String reference, int distance, Date dL) {
 this.reference = reference; this.distance = distance; this.dateLivraison = dL;
 } public Cargaison() {}
 @JsonIgnore
 public Collection<Marchandise> getMarchandises() { return marchandises;} //Get et Set}
```



# Entity : CargaisonAerienne

```
package metier.transport;
import java.util.Date;
import javax.persistence.*;
@Entity
@DiscriminatorValue("CA")
public class CargaisonAerienne extends Cargaison {
 private double poidsMax;
 public CargaisonAerienne(String reference, int distance, Date dateLivraison,
double poidsMax) {
 super(reference, distance, dateLivraison);
 this.poidsMax = poidsMax;
 }
 public CargaisonAerienne() {
 }

 // Getters et Setters
}
```



# Entity : CargaisonRoutiere

```
package metier.transport;
import java.util.Date;
import javax.persistence.*;
@Entity
@DiscriminatorValue("CR")
public class CargaisonRoutiere extends Cargaison {
 private float temperatureConservation;
 public CargaisonRoutiere(String reference, int distance, Date dateLivraison,
 float temperatureConservation) {
 super(reference, distance, dateLivraison);
 this.temperatureConservation = temperatureConservation;
 }
 public CargaisonRoutiere() {
 }
 // Getters et Setters
}
```

# persistence.xml

Le dossier META-INF devrait être ajouté au classpath :



```
<?xml version="1.0" encoding="UTF-8"?>
<persistence
 xmlns="http://java.sun.com/xml/ns/persistence"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
 http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd"
 version="1.0">
 <persistence-unit name="UP_TRANSPORT">
 <jta-data-source>java:/dsTRANSPORT</jta-data-source>
 <properties>
 <property name="hibernate.hbm2ddl.auto" value="update"/>
 </properties>
 </persistence-unit>
</persistence>
```

# Déployer le projet

- Les deux tables CAGAISONS et MARCHANDISES de la bases de données devraient être générées:

## MARCHANDISES :

Colonne	Type	Interclassement	Attributs	Null	Défaut	Extra
<u>NUMERO</u>	bigint(20)			Non	Aucun	AUTO_INCREMENT
<b>nom</b>	varchar(25)	latin1_swedish_ci		Oui	NULL	
<b>poids</b>	double			Oui	NULL	
<b>volume</b>	double			Oui	NULL	
<b>REF_CARG</b>	varchar(255)	latin1_swedish_ci		Oui	NULL	

## CAGAISONS :

Colonne	Type	Interclassement	Attributs	Null	Défaut	Extra
<u>REF_CARG</u>	varchar(255)	latin1_swedish_ci		Non	Aucun	
<b>TYPE_CARG</b>	varchar(2)	latin1_swedish_ci		Non	Aucun	
<b>dateLivraison</b>	datetime			Oui	NULL	
<b>distance</b>	double			Oui	NULL	
<b>poidsMax</b>	double			Oui	NULL	
<b>temperature</b>	double			Oui	NULL	

# Traitements Métier

- Interface Local

```
package metier.transport;
import java.util.List;
import javax.ejb.Local;
@Local
public interface ITransportLocal {
 public void addCargaison(Cargaison c);
 public void addMarchandise(Marchandise m, String refCarg);
 public List<Cargaison> getAllCargaisons();
 public List<Marchandise> getMarchandisesParCarg(String refCarg);
 public Cargaison getCargaison(String reg);
 public void supprimerMarchandise(Long numeroMarch);
}
```

# Traitements Métier

- EJB Session Stateless

```
package metier.transport;import java.util.List;
import javax.ejb.Stateless;import javax.persistence.*;
@Stateless(name="TRANS")
public class TransportEJBImpl implements ITransportLocal {
@PersistenceContext(unitName="UP_TRANSPORT")
EntityManager em;
@Override
public void addCargaison(Cargaison c) {
em.persist(c);
}
@Override
public void addMarchandise(Marchandise m, String refCarg) {
Cargaison c=em.find(Cargaison.class, refCarg);
m.setCargaison(c);em.persist(m);
}
@Override
public List<Cargaison> getAllCargaisons() {
Query req=em.createQuery("select c from Cargaison c");
return req.getResultList();
}
```

# Traitements Métier

- EJB Session Stateless

```
@Override
public List<Marchandise> getMarchandisesParCarg(String refCarg) {
 Query req=em.createQuery
 ("select m from Marchandise m where m.cargaison.reference=:x");
 req.setParameter("x", refCarg);
 return req.getResultList();
}

@Override
public Cargaison getCargaison(String reg) {
 Cargaison c=em.find(Cargaison.class, reg);
 return c;
}

@Override
public void supprimerMarchandise(Long numeroMarch) {
 Marchandise m=em.find(Marchandise.class, numeroMarch);
 em.remove(m);
}
```



# Couche Service

med@youssf.net

# Web Service SOAP

```
package service.transport;import java.util.*;
import javax.ejb.*;import javax.jws.*;import metier.transport.*;
@Stateless
@WebService
public class TransportSOAPService {
 @EJB
 private ITransportLocal metier;
 @WebMethod
 public void ajouterCargaisonRoutiere(
 @WebParam(name="ref")String ref,
 @WebParam(name="distance")int dist,
 @WebParam(name="dateLivraison")Date dateLiv,
 @WebParam(name="tempConserv")float temp){
 CargaisonRoutiere cr=new CargaisonRoutiere(ref, dist, dateLiv, temp);
 metier.addCargaison(cr);
 }
}
```

# Web Service SOAP

```
@WebMethod
public void ajouterCargaisonAerienne(
 @WebParam(name="ref")String ref,
 @WebParam(name="distance")int dist,
 @WebParam(name="dateLivraison")Date dateLiv,
 @WebParam(name="poidsMax")float poids){
 CargaisonAerienne ca=new CargaisonAerienne(ref, dist, dateLiv, poids);
 metier.addCargaison(ca);
}

@WebMethod
public void ajouterMarchandise(
 @WebParam(name="nom")String nom,
 @WebParam(name="poids")double poids,
 @WebParam(name="volume")double volume,
 @WebParam(name="refCarg")String refCarg){
 Marchandise m=new Marchandise(nom, poids, volume);
 metier.addMarchandise(m, refCarg);
}
```

# Web Service SOAP

```
@WebMethod
public List<Cargaison> getAllCargaisons(){
 return metier.getAllCargaisons();
}

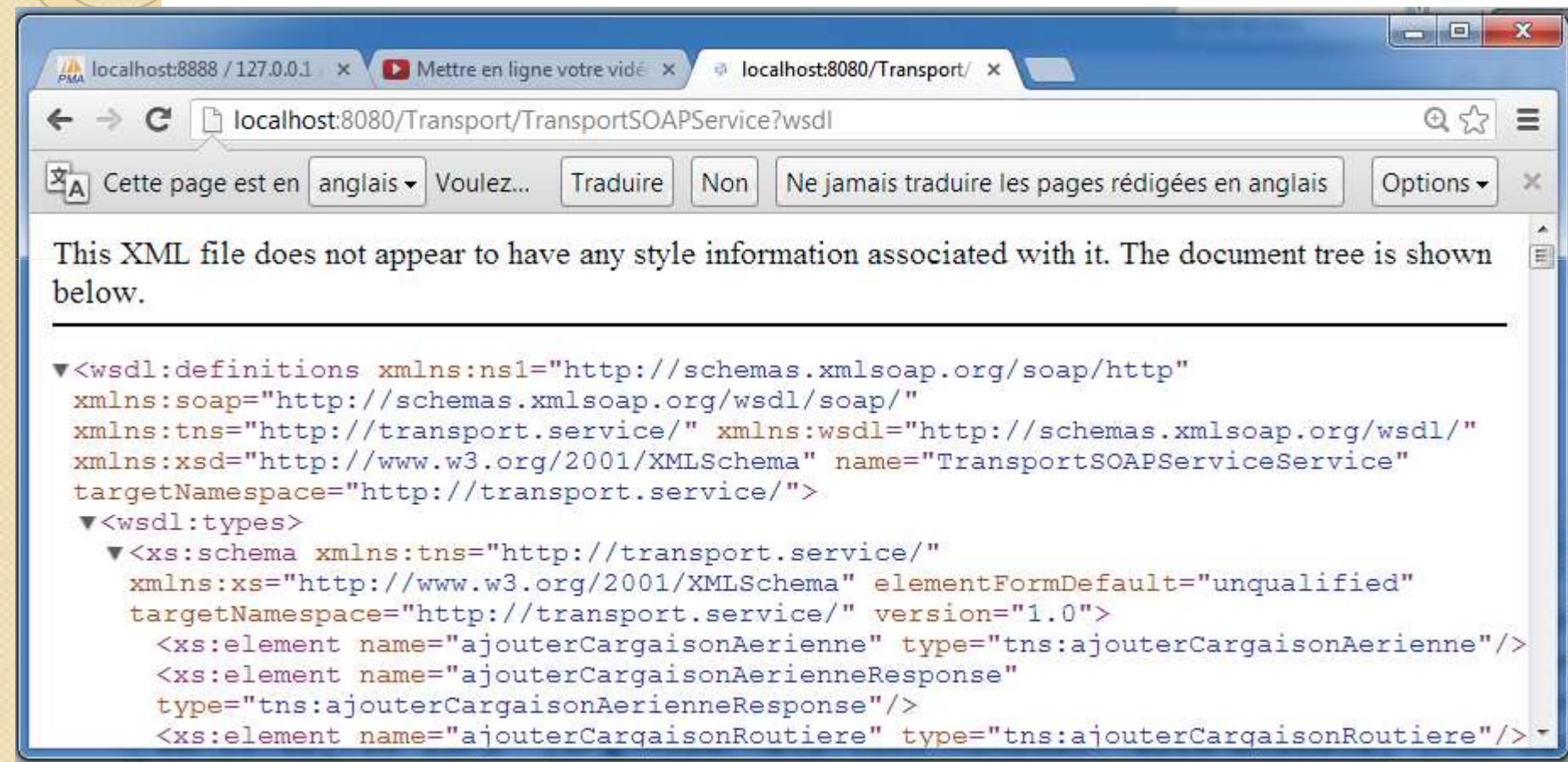
@WebMethod
public List<Marchandise> getMarchParCarg(
 @WebParam(name="refCarg")String refCarg){
 return metier.getMarchandisesParCarg(refCarg);
}

@WebMethod
public Cargaison consulterCargaison(
 @WebParam(name="refCarg")String refCarg){
 return metier.getCargaison(refCarg);
}

@WebMethod
public void supprimerMarchandise(
 @WebParam(name="numMarch")Long numero){
 metier.supprimerMarchandise(numero);
}}
```

# Redéployer le projet

- WSDL



The screenshot shows a web browser window with three tabs at the top: "localhost:8888 / 127.0.0.1", "Mettre en ligne votre vidéo", and "localhost:8080/Transport/". The main content area displays the WSDL XML for the TransportSOAPService. The XML defines a service named "TransportSOAPServiceService" with a target namespace "http://transport.service/". It includes a schema for "transport.service/" with elements like "ajouterCargaisonAerienne", "ajouterCargaisonAerienneResponse", and "ajouterCarqaisonRoutiere".

```
<wsdl:definitions xmlns:ns1="http://schemas.xmlsoap.org/soap/http"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:tns="http://transport.service/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="TransportSOAPServiceService"
 targetNamespace="http://transport.service/">
 <wsdl:types>
 <xsd:schema xmlns:tns="http://transport.service/"
 xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="unqualified"
 targetNamespace="http://transport.service/" version="1.0">
 <xsd:element name="ajouterCargaisonAerienne" type="tns:ajouterCargaisonAerienne"/>
 <xsd:element name="ajouterCargaisonAerienneResponse"
 type="tns:ajouterCargaisonAerienneResponse"/>
 <xsd:element name="ajouterCarqaisonRoutiere" type="tns:ajouterCarqaisonRoutiere"/>
 </xsd:schema>
 </wsdl:types>

```

# Tester les méthodes avec le client SOAP Oxygen

Analyseur WSDL SOAP

WSL

Services: TransportSOAPServiceService  
Ports: TransportSOAPServicePort  
Opérations: ajouterCargaisonRoutiere

Actions

URL: http://WIN-7PA448PU1OR:8080/Transport/TransportSOAPService  
Action SOAP: Version: 1.1 1.2

Requête Fichiers joints

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<ajouterCargaisonRoutiere xmlns="http://transport.service/">
<ref xmlns="">CR2</ref>
<distance xmlns="">120</distance>
<dateLivraison xmlns="">2011-12-11</dateLivraison>
<tempConserv xmlns="">22</tempConserv>
</ajouterCargaisonRoutiere>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Envoyer Configuration de la requête: Ouvrir Enregistrer Régénérer

Ouvrir la réponse dans l'éditeur

Localisateur du fichier WSDL

Fichier WSDL Demande SOAP sauveée

WSDL URL: http://localhost:8080/Transport/TransportSOAPService?wsdl

Ouvrir Annuler

sample response. -->
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<ajouterCargaisonRoutiereResponse xmlns="http://transport.service/">

# Requête SOAP pour ajouter les cargaisons et les marchandises

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
 <SOAP-ENV:Header/>
 <SOAP-ENV:Body>
 <ajouterCargaisonRoutiere xmlns="http://transport.service/">
 <ref xmlns="">CR2</ref>
 <distance xmlns="">120</distance>
 <dateLivraison xmlns="">2011-12-11</dateLivraison>
 <tempConserv xmlns="">22</tempConserv>
 </ajouterCargaisonRoutiere>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

REF_CARG	TYPE_CARG	dateLivraison	distance	poidsMax	temperature
CA1	CA	2013-05-29 10:09:55	500	700	NULL
CA2	CA	2013-05-29 10:09:59	600	600	NULL
CR1	CR	2013-05-29 10:09:59	300	NULL	23



# Requête SOAP pour ajouter les cargaisons et les marchandises

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope">
 <SOAP-ENV:Header/>
 <SOAP-ENV:Body>
 <ajouterCargaisonAerienne xmlns="http://transport.service/">
 <ref xmlns="">CA2</ref>
 <distance xmlns="">123</distance>
 <dateLivraison xmlns="">2012-12-13</dateLivraison>
 <poidsMax xmlns="">200</poidsMax>
 </ajouterCargaisonAerienne>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

NUMERO	nom	poids	volume	REF_CARG
1	Ordinateurs	200	400	CA1
2	Imprimantes	100	200	CA1
3	CD	100	400	CR1

## Requête SOAP pour consulter toutes les cargaisons

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
 <SOAP-ENV:Header/>
 <SOAP-ENV:Body>
 <getAllCargaisons xmlns="http://transport.service/">
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
 <ns2:getAllCargaisonsResponse xmlns:ns2="http://transport.service/">
 <return xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="ns2:cargaisonAerienne">
 <reference>CA1</reference>
 <distance>700</distance>
 <dateLivraison>2012-11-12T00:00:00Z</dateLivraison>
 <poidsMax>500.0</poidsMax>
 </return>
 <return xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="ns2:cargaisonRoutiere">
 <reference>CRI</reference>
 <distance>900</distance>
 <dateLivraison>2012-11-11T00:00:00Z</dateLivraison>
 <temperatureConservation>22.0</temperatureConservation>
 </return>
 </ns2:getAllCargaisonsResponse>
 </soap:Body>
</soap:Envelope>
```

# Requête SOAP pour consulter toutes les marchandises d'une cargaison

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
 <SOAP-ENV:Header/>
 <SOAP-ENV:Body>
 <getMarchParCarg xmlns="http://transport.service/">
 <refCarg xmlns="">CA1</refCarg>
 </getMarchParCarg>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
 <ns2:getMarchParCargResponse xmlns:ns2="http://transport.service/">
 <return>
 <cargaison xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="ns2:cargaisonAerienne">
 <reference>CA1</reference>
 <distance>700</distance>
 <dateLivraison>2012-11-12T00:00:00Z</dateLivraison>
 <poidsMax>500.0</poidsMax>
 </cargaison>
 <nom>PC</nom>
 <numero>1</numero>
 <poids>22.0</poids>
 <volume>66.0</volume>
 </return>
 <return> </return>
 </ns2:getMarchParCargResponse>
 </soap:Body>
</soap:Envelope>
```

# Web Service REST FULL

```
package service.transport;import java.util.Date; import java.util.List;
import javax.ejb.*; import javax.ws.rs.*;
import javax.ws.rs.core.MediaType; import metier.transport.*;
@Stateless
@Path("/transport")
public class TransportRESTService {
 // Injection des dépendances
 @EJB
 private ITransportLocal metier;
 // Méthode pour ajouter une cargaison routière
 @GET
 @Path("addCargRoutiere/{ref}/{dist}/{dateLiv}/{temp}")
 @Produces(MediaType.APPLICATION_JSON)
 public void ajouterCargaisonRoutiere(
 @PathParam(value="ref")String ref, @PathParam(value="dist")int dist,
 @PathParam(value="dateLiv")Date dateLiv,
 @PathParam(value="temp")float temp){
 CargaisonRoutiere cr=new CargaisonRoutiere(ref, dist, dateLiv, temp);
 metier.addCargaison(cr);
 }
}
```

# Web Service SOAP

```
// Méthode pour ajouter une cargaison Aérienne

@GET
@Path("addCargAerienne/{ref}/{dist}/{dateLiv}/{poids}")
@Produces(MediaType.APPLICATION_JSON)
public void ajouterCargaisonAerienne(
 @PathParam(value="ref")String ref, @PathParam(value="dist")int dist,
 @PathParam(value="dateLiv")Date dateLiv, @PathParam(value="poids")float poids){
 CargaisonAerienne ca=new CargaisonAerienne(ref, dist, dateLiv, poids);
 metier.addCargaison(ca);
}

// Méthode pour ajouter une marchandise

@GET
@Path("addMarch/{nom}/{poids}/{volume}/{refCag}")
@Produces(MediaType.APPLICATION_JSON)
public void ajouterMarchandise(
 @PathParam(value="nom")String nom, @PathParam(value="poids")double poids,
 @PathParam(value="volume")double volume, @PathParam(value="refCarg")String refCarg){
 Marchandise m=new Marchandise(nom, poids, volume);
 metier.addMarchandise(m, refCarg);
}
```

# Web Service SOAP

```
// Méthode pour consulter toutes les cargaisons
@GET
@Path("allCarg")
@Produces(MediaType.APPLICATION_JSON)
public List<Cargaison> getAllCargaisons(){
 return metier.getAllCargaisons();
}

// Méthode pour consulter les marchandises d'une cargaison
@GET
@Path("merchandises/{refCarg}")
@Produces(MediaType.APPLICATION_JSON)
public List<Marchandise> getMarchParCarg(
@PathParam(value="refCarg")String refCarg){
 return metier.getMarchandisesParCarg(refCarg);
}
```

# Web Service SOAP

```
// Méthode pour consulter une cargaison
@GET
@Path("consulterCargaison/{refCarg}")
@Produces(MediaType.APPLICATION_JSON)
public Cargaison consulterCargaison(
 @PathParam(value="refCarg")String refCarg){
 return metier.getCargaison(refCarg);
}

// Méthode pour supprimer une marchandise
@GET
@Path("supprimerMarch/{num}")
@Produces(MediaType.APPLICATION_JSON)
public void supprimerMarchandise(
 @PathParam(value="num")Long numero){
 metier.supprimerMarchandise(numero);
}
}
```

# Application REST FULL

```
package service.transport;
import javax.ws.rs.ApplicationPath;
import javax.ws.rs.core.Application;
@ApplicationPath("/")
public class TransportRestApp extends Application {
}
```

# Tester le service REST

The image displays three separate browser windows, each showing a different JSON response from a REST API endpoint.

- Top Window:** Shows the URL `localhost:8080/Transport/transport/allCarg`. The response is a list of four objects:

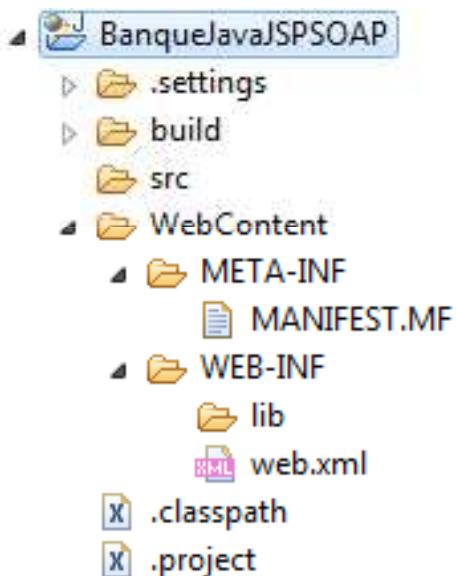
```
[{"reference": "CA1", "distance": 700, "dateLivraison": 1352678400000, "poidsMax": 500.0}, {"reference": "CA2", "distance": 123, "dateLivraison": 1355356800000, "poidsMax": 200.0}, {"reference": "CR1", "distance": 900, "dateLivraison": 1352592000000, "temperatureConservation": 22.0}, {"reference": "CR2", "distance": 120, "dateLivraison": 1323561600000, "temperatureConservation": 22.0}]
```
- Middle Window:** Shows the URL `localhost:8080/Transport/transport/marchandises/CA1`. The response is a list of two objects:

```
[{"numero": 1, "nom": "PC", "poids": 22.0, "volume": 66.0, "cargaison": {"reference": "CA1", "distance": 700, "dateLivraison": 1352678400000, "poidsMax": 500.0}}, {"numero": 2, "nom": "Imprimantes", "poids": 23.0, "volume": 88.0, "cargaison": {"reference": "CA1", "distance": 700, "dateLivraison": 1352678400000, "poidsMax": 500.0}}]
```
- Bottom Window:** Shows the URL `localhost:8080/Transport/transport/consulterCargaison/CA1`. The response is a single object:

```
{"reference": "CA1", "distance": 700, "dateLivraison": 1352678400000, "poidsMax": 500.0}
```

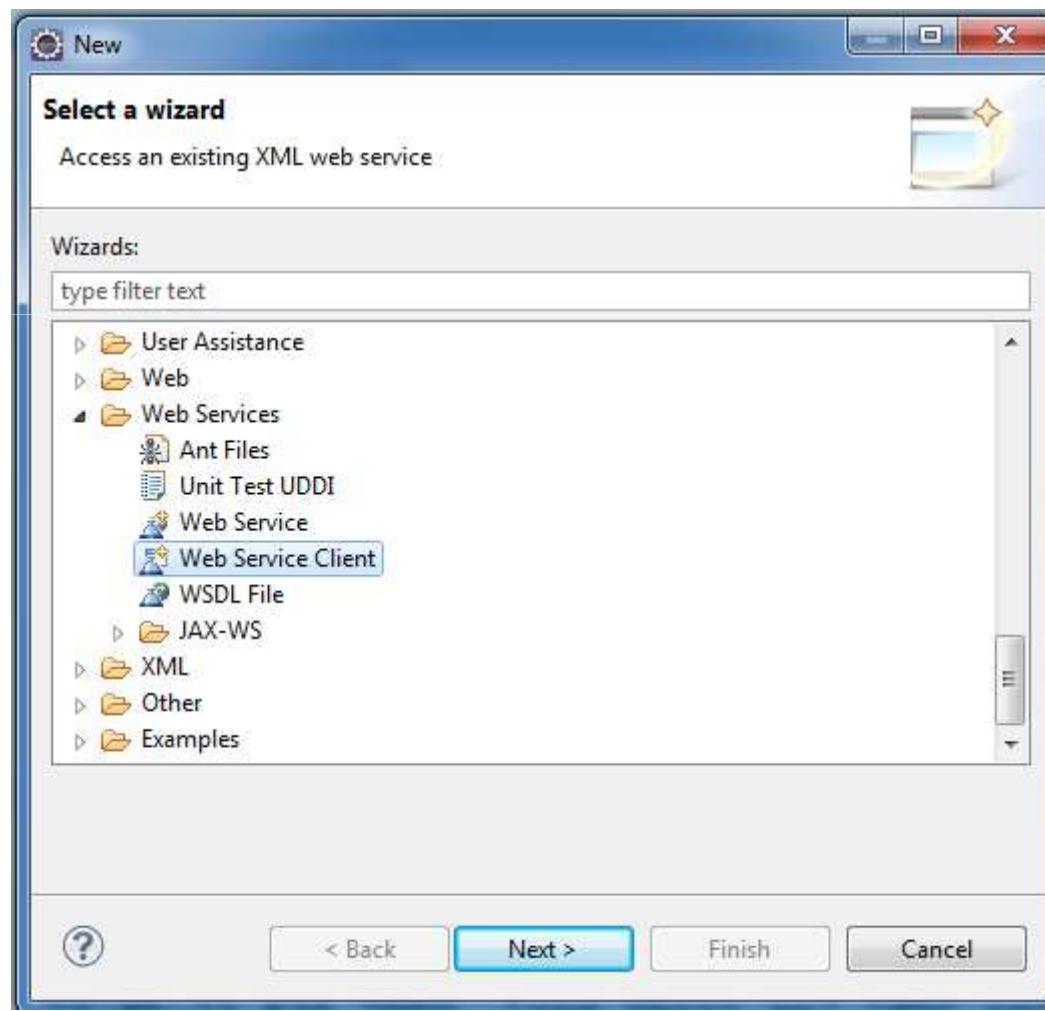
# Client Java, JSP Via un Proxy SOAP

- Crer un Projet Web Dynamique



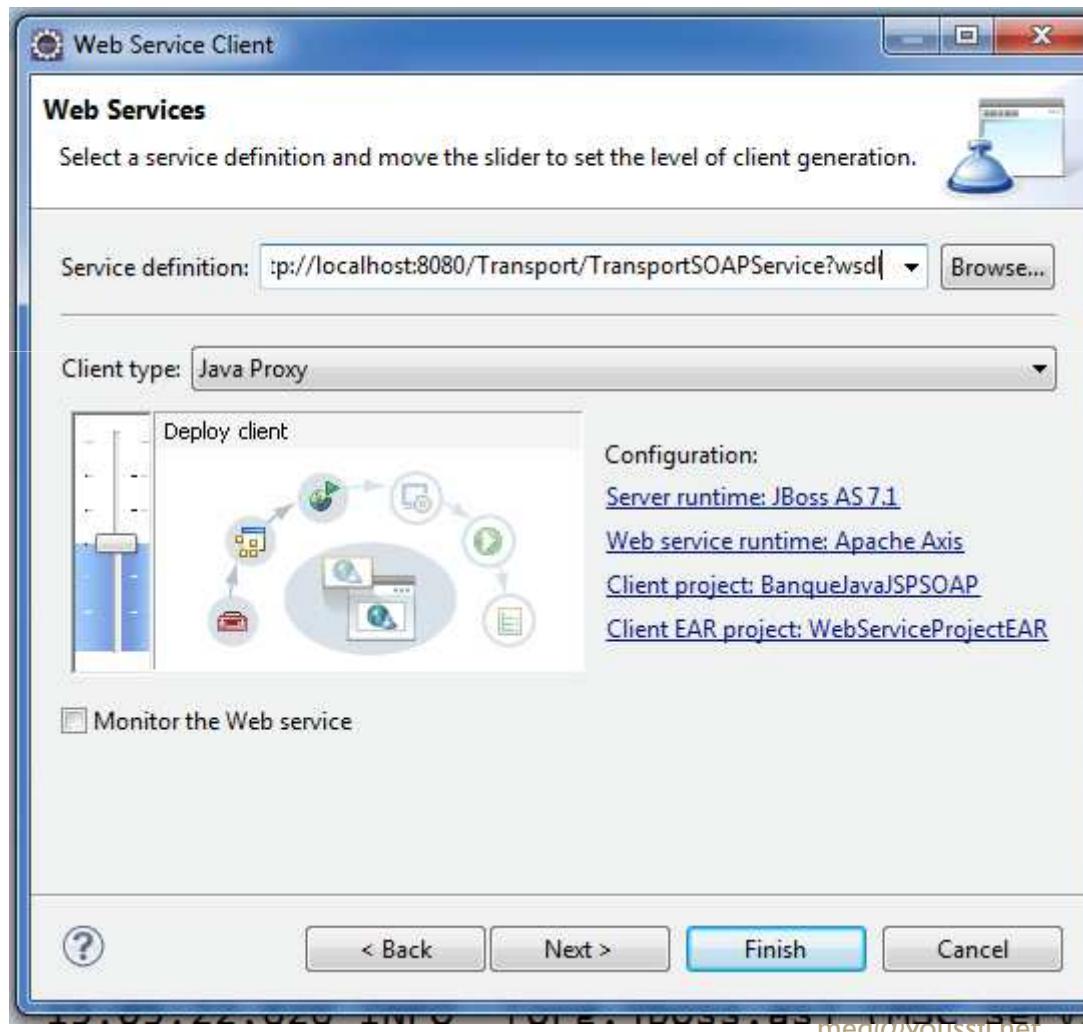
# Générer un Proxy pour l'accès au service SOAP

- Créer un client Web Service



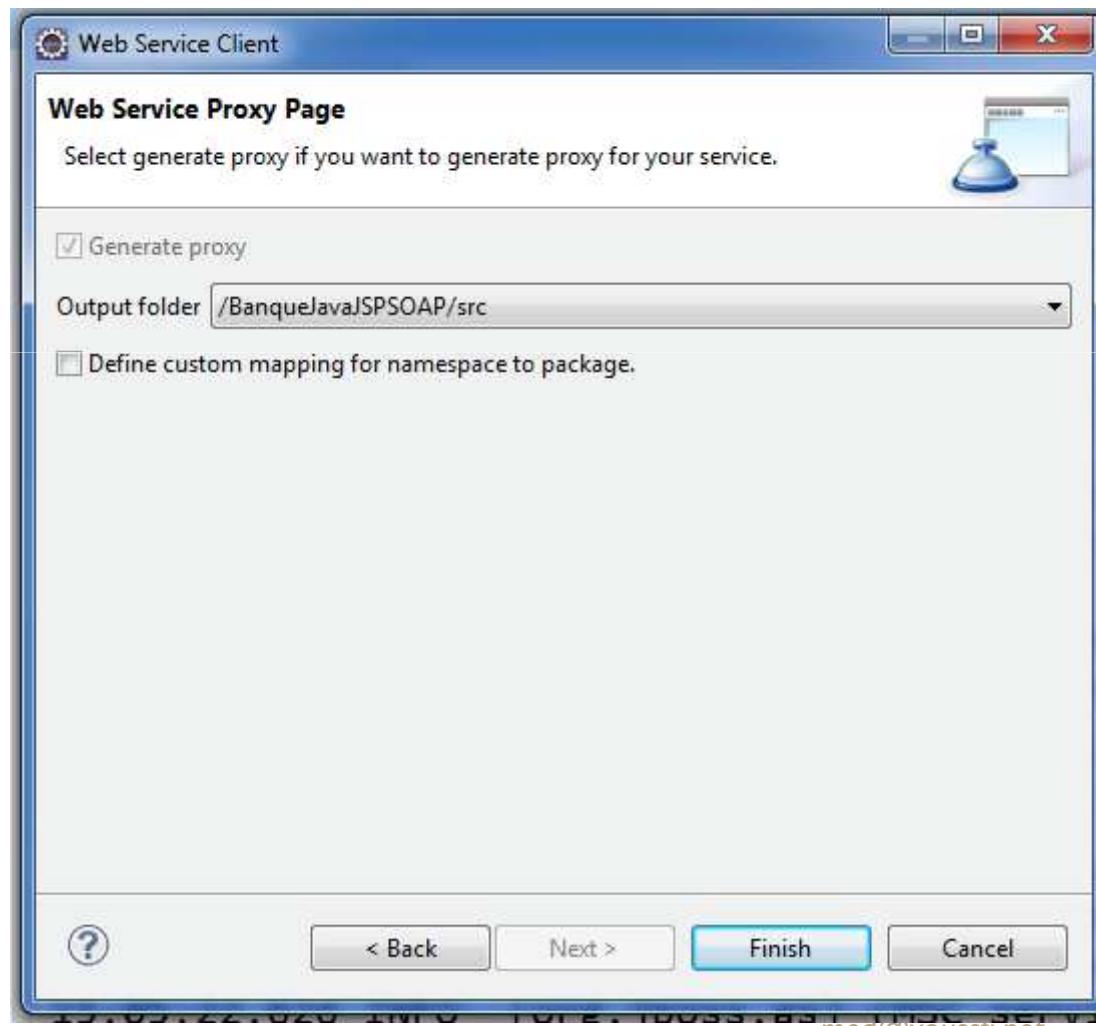
# Générer un Proxy pour l'accès au service SOAP

- Spécifier l'adresse du WSDL

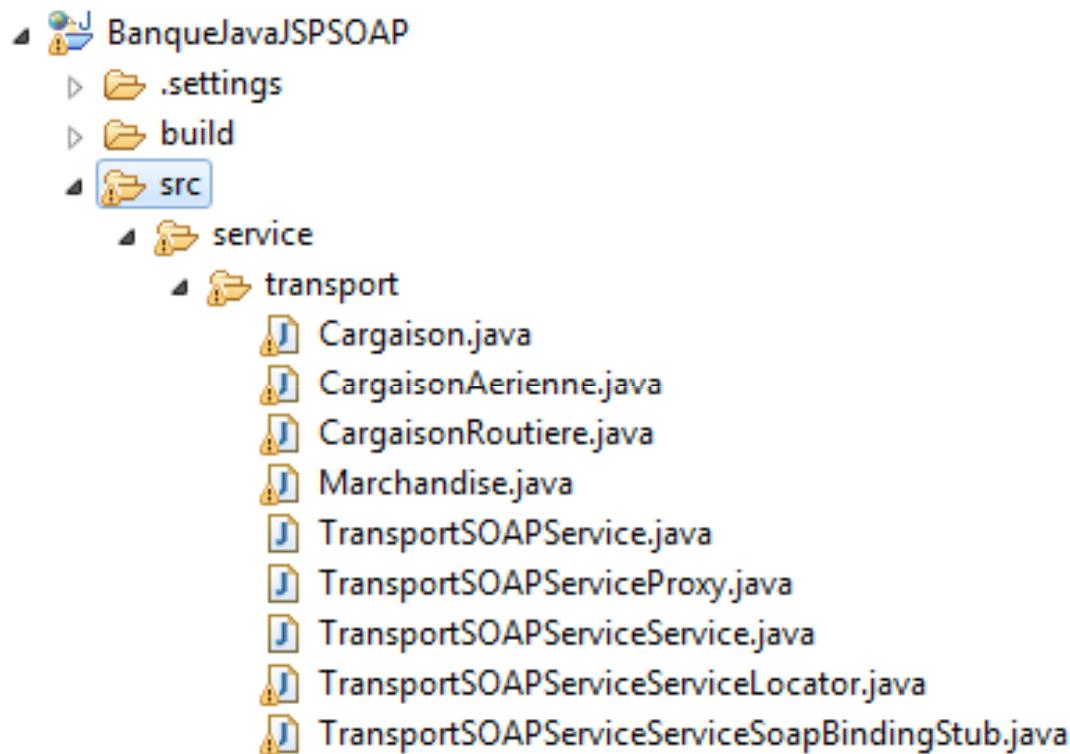


# Générer un Proxy pour l'accès au service SOAP

- L'assistant est prêt à générer le code dans le dossier src



# Proxy Généré



# Client Java Lourd

```
package clientSOAP; import java.rmi.RemoteException; import java.util.*;
import service.transport.*;
public class ClientLourd {
 public static void main(String[] args) {
 try {
 TransportSOAPServiceProxy stub=new TransportSOAPServiceProxy();
 stub.ajouterCargaisonAerienne("CA4", 600, Calendar.getInstance(), 800);
 stub.ajouterCargaisonRoutiere("CR4", 200, Calendar.getInstance(), 22);
 stub.ajouterMarchandise("Ordinateurs", 900, 56, "CA4");
 Cargaison[] cargs=stub.getAllCargaisons();
 System.out.println("---- Totes Les cargaison---");
 for(Cargaison c:cargs){
 System.out.println(c.getReference()+"---"+c.getDistance()+"---"
 "+c.getClass().getSimpleName());
 }
 System.out.println("--- Marchandises d'une cargaison ---");
 Marchandise[] marchandises=stub.getMarchParCarg("CA1");
 for(Marchandise m:marchandises){
 System.out.println(m.getNom()+"---"+m.getPoids()+"---"+m.getVolume());
 }
 } catch (RemoteException e) { e.printStackTrace(); } } }
```

# Exécution du client Lourd

---- Totes Les cargaison---

CA1--700--CargaisonAerienne

CA2--123--CargaisonAerienne

CA4--600--CargaisonAerienne

CR1--900--CargaisonRoutiere

CR2--120--CargaisonRoutiere

CR4--200--CargaisonRoutiere

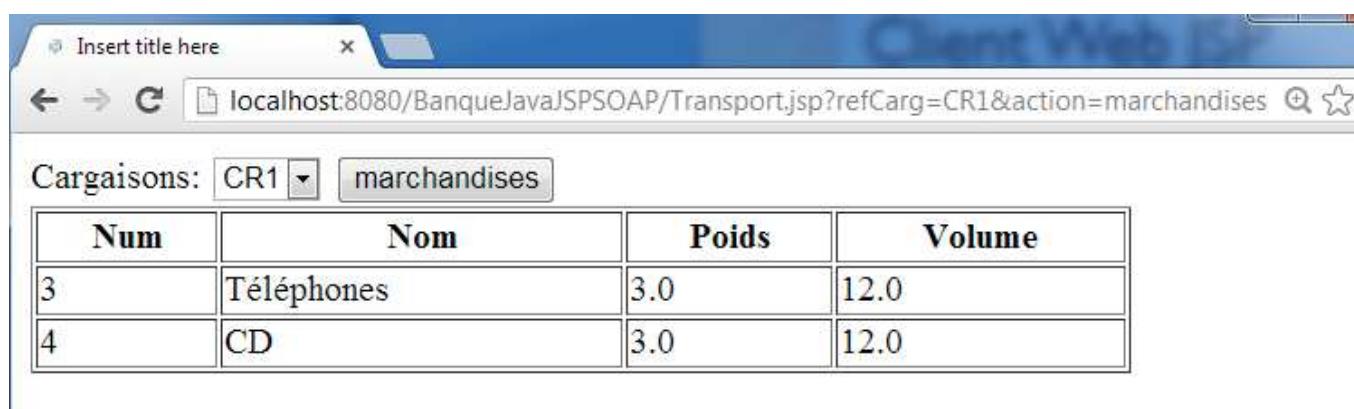
--- Marchandises d'une cargaison ---

PC--22.0--66.0

Imprimantes--23.0--88.0

# Client Web JSP

```
<%@page import="service.transport.*"%>
<%@page import="service.transport.TransportSOAPServiceProxy"%>
<%
 TransportSOAPServiceProxy stub=new TransportSOAPServiceProxy();
 Cargaison[] cargs=stub.getAllCargaisons();
 Marchandise[] mdises=new Marchandise[0];
 String refCarg=null;
 if(request.getParameter("action")!=null){
 String action=request.getParameter("action");
 if(action.equals("marchandises")){
 refCarg=request.getParameter("refCarg");
 mdises=stub.getMarchParCarg(refCarg);
 }
 }
%>
```

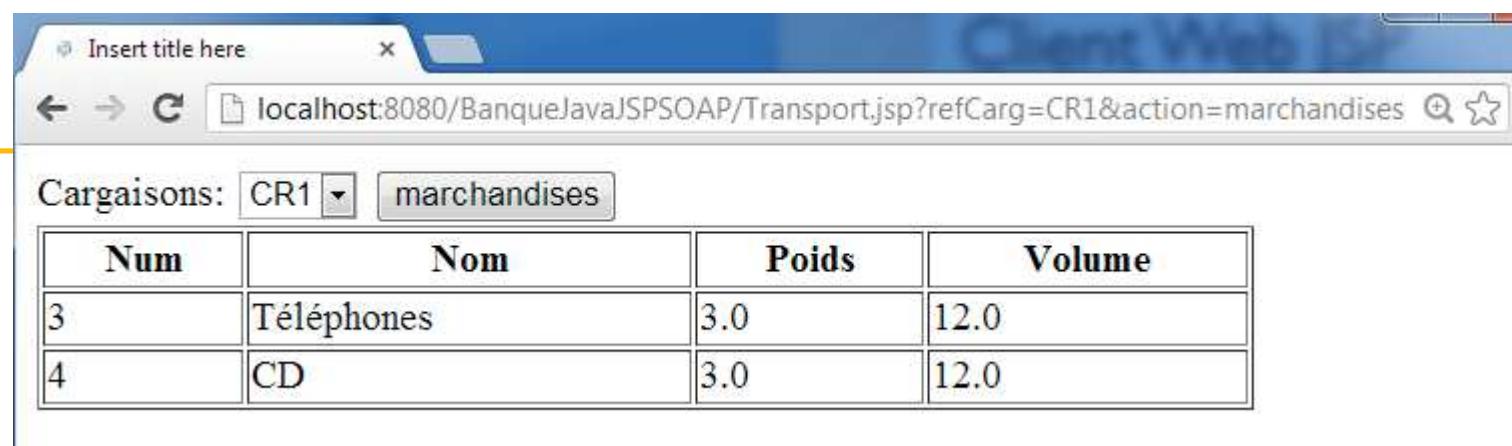


# Client Web JSP

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
 <form action="Transport.jsp" method="get">
 Cargaisons:
 <select name="refCarg">
 <option>-----</option>
 <% for(Cargaison c:cargs) {%
 <option value="<%c.getReference()%>">
 <%if(c.getReference().equals(refCarg)){ %> selected="selected" <%} %> >
 <%=c.getReference()%>
 </option>
 <%} %>
 </select>
 <input type="submit" name="action" value="marchandises">
 </form>
```

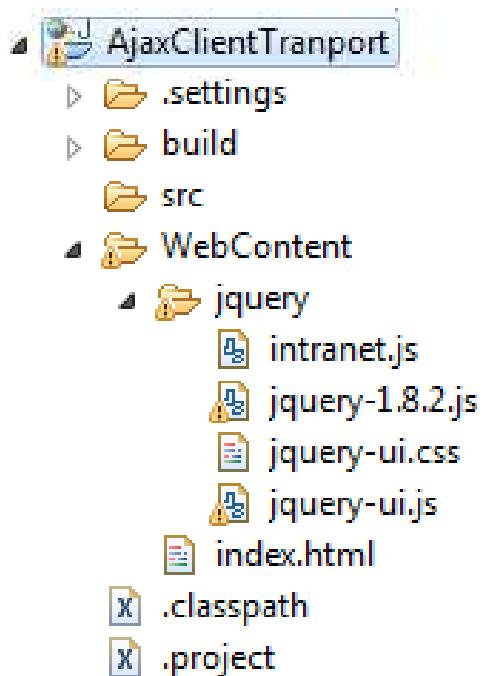
# Client Web JSP

```
<table border="1" width="80%">
 <tr>
 <th>Num</th><th>Nom</th><th>Poids</th><th>Volume</th>
 </tr>
 <% for(Marchandise m:mdises){%>
 <tr>
 <td><%=m.getNumero()%></td><td><%=m.getNom()%></td><td>
 <%=m.getPoids()%></td><td><%=m.getVolume() %></td>
 </tr>
 <%} %>
 </table>
</body>
</html>
```



# Client Web Ajax JQuery

- Créer un Projet Web Statique
- Copier le framework Jquery :
  - Jquery-1.8.2.js



# Transport.html

```
<html>
<head>
<script language="javascript" src="jquery/jquery-1.8.2.js"></script>
<script language="javascript" src="jquery/jquery-ui.js"></script>
<script language="javascript" src="jquery/intranet.js"></script>
<link rel="stylesheet" type="text/css" href="jquery/jquery-ui.css" />
</head>
<body onload="chargerCarg()">
<div>
 Cargaisons :
 <table border="1" width="80%">
 <thead>
 <tr><th>REF CARG</th><th>DATE LIVRAISON</th>
 <th>DISTANCE</th><th>Type</th><th>Poids Max</th><th>Temp Cons</th>
 </tr>
 </thead>
 <tbody id="cargaisons">
 </tbody>
</table>
```

# Transport.html

```
<p></p>

<table border="1" width="80%">
 <thead>
 <tr>
 <th>Numéro</th><th>Nom</th><th>Poids</th><th>Volume</th>
 </tr>
 </thead>
 <tbody id="tableBody">

 </tbody>
</table>
</div>
</body>
</html>
```

# Code Java Script avec Jquery : intranet.js

```
function chargerCarg(){
$.getJSON('/TP_TRANSPORT/transport/consulterCarg', function(data){
filieres=data;
for(i in data){
$tr=$("<tr>");
$lien=$("<a>").append(data[i]['reference']).attr("href","javascript:chargerMarchan
dises('"+data[i]['reference']+')");
$td=$("<td>").append($lien);$tr.append($td);
$td=$("<td>").append(data[i]['dateLivraison']); $tr.append($td);
$td=$("<td>").append(data[i]['distance']); $tr.append($td);
var type=(data[i]['poidsMax']==null)?"Aérienne":"Routière";
$td=$("<td>").append(type); $tr.append($td);
$td=$("<td>").append(data[i]['poidsMax']); $tr.append($td);
$td=$("<td>").append(data[i]['temperatureConservation']); $tr.append($td);
$("#cargaisons").append($tr);
}
}
);
}
```

# Code Java Script avec Jquery : intranet.js

```
function chargerMarchandises(refCarg){
 $.getJSON('/TP_TRANSPORT/transport/consulterMarchParCarg/' +refCarg,
 function(data){
 filieres=data;
 for(i in data){
 $tr=$("<tr>");
 $td=$("<td>").append(data[i]['numero']); $tr.append($td);
 $td=$("<td>").append(data[i]['nom']); $tr.append($td);
 $td=$("<td>").append(data[i]['poids']); $tr.append($td);
 $td=$("<td>").append(data[i]['volume']); $tr.append($td);
 $("#tableBody").html("");
 $("#tableBody").append($tr);
 }
 }
);
}
```

# Aperçu du client JQuery





# JQuery



# Les premiers « Frameworks »

- JavaScript a été créé pour améliorer l'interactivité entre les utilisateurs et les pages HTML.
- Avec une grande maîtrise de JavaScript, associé au CSS, on peut créer des interfaces graphiques web performantes de très grande qualité.
- Les développeurs professionnels JavaScript ont capitalisé leurs expériences en créant des bibliothèques JavaScript qui permettent de faciliter la tâche aux développeurs web.
- Beaucoup de « Frameworks » JavaScript, on vu leurs naissances :
  - PrototypeJS : [www.prototypejs.org](http://www.prototypejs.org)
  - Mootools : [www.mootools.net](http://www.mootools.net)
  - Dojo Toolkit : [www.dojotoolkit.org](http://www.dojotoolkit.org)
  - Yahoo UI : [www.developer.yahoo.com/yui/](http://www.developer.yahoo.com/yui/)
  - ExtJS : [www.extjs.com](http://www.extjs.com)
  - UIZE : [www.uize.com](http://www.uize.com)
  - Spry : [www.adobe.com](http://www.adobe.com)
  - JQuery



# JQuery

- Une bibliothèque javascript open-source
- Elle permet de traverser et manipuler très facilement l'arbre DOM des pages web à l'aide d'une syntaxe simplifiée.
- JQuery permet par exemple
  - de changer/ajouter une classe CSS,
  - créer des animations,
  - Afficher des widgets (Menus, Panneaux à onglets etc..)
  - modifier des attributs,
  - Gérer les événements javascript
  - Envoyer des requêtes HTTP AJAX aux serveurs Web
  - ....



# une simple bibliothèque à importer

- Disponible sur le site de Jquery

<http://jquery.com/>

```
<script src="jquery-1.8.2.js"></script>
```

- Ou directement sur Google code

```
<script type="text/javascript" src=
 "http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js">
</script>
```



# JavaScript / jQuery

- Masquer tous les div du document HTML avec JavaScript pur:

```
divs =
 document.getElementsByTagName('div');
for (i = 0; i < divs.length; i++) {
 divs[i].style.display = 'none';
}
```

- Faire la même chose avec jQuery :  
**`$( "div" ).hide();`**



# La fonction jQuery()

- jQuery repose sur une seule fonction : `jQuery()` ou `$()`.
- C'est une fonction JavaScript
- Elle accepte des paramètres
- Elle retourne un objet
- \$ : Syntaxe issue du framework « Prototype »

# Les sélecteurs d'éléments: \$('anything')

- \$ accepte des ID :
  - `$( '#nomID' )` retourne un élément sachant son id
  - Equivalent javascript :
    - `document.getElementById('nomID')`
- \$ accepte des classes css :
  - `$( '.nomClasse' )` : retourne tous les éléments qui correspondent à cette classe
  - Equivalent javascript :
    - `document.getElementsByClassName('nomClasse')`
- \$ accepte plusieurs sélecteurs
  - `$( '#article, .nouvelles, a' )`
  - Permet de sélectionner les éléments suivants:
    - L'élément dont id est article
    - les éléments ayant la classe css nouvelles
    - Les liens hypertexte (a)

# Les sélecteurs d'éléments: \$('anything')

- **\$ ( document )**
  - Extension de l'objet document à la classe jQuery.
- **\$ ( '\*' )**
  - Sélectionne tous les éléments.
- **\$ ( '#monDiv' )**
  - Sélectionne l'élément ayant l'ID "monDiv".
- **\$ ( 'p.first' )**
  - Sélectionne les éléments <p> ayant la classe "first".
- **\$ ( 'p[title]' )**
  - Sélectionne les éléments <p> ayant un attribut "title".

# Les sélecteurs d'éléments: \$('anything')

- `$('.p[title="Bonjour"]')`
  - Sélectionne les éléments <p> dont l'attribut title est "Bonjour".
- `$('.p[title!="Bonjour"]')`
  - Sélectionne les éléments <p> dont l'attribut title n'est pas "Bonjour".
- `$('.p[title^="H"]')`
  - Sélectionne les éléments dont l'attribut title commence par "H".
- `$('.p[title$="H"]')`
  - Sélectionne les éléments dont l'attribut title fini par "H".

# Les sélecteurs d'éléments: \$('anything')

- **\$('p[title\*="H"]')**
  - Sélectionne les éléments dont l'attribut title contient "H".
- **\$('ul, ol, dl')**
  - Sélectionne les éléments <ul>, <ol> et <dl>
- **\$('div .desc')**
  - Sélectionne les éléments ayant la classe "desc" descendants (au sens CSS) d'éléments <div>.

# Les sélecteurs d'éléments: \$('anything')

- `$('div > .enfant')`
  - Sélectionne les éléments ayant la classe "enfant" enfants d'éléments `<div>`.
- `$('label + input')`
  - Sélectionne les éléments `<input />` dont l'élément précédent (dans le DOM) est `<label>`.
- `$('#debut ~ div')`
  - Sélectionne les éléments `<div>` frères se situant après l'élément dont l'id est "debut".



# Filtrer les sélections

- jQuery offre une très large possibilité de sélection d'éléments en fonction de filtres sur des collections d'éléments.
- Le fonctionnement de la sélection par filtre est simple : on sélectionne d'abord un ensemble d'éléments (par défaut, tous) puis on affine cette sélection à partir de certains critères.



# Filtrer les sélections

- **`$( 'div:first' )`**
  - Sélectionne le premier élément <div>.
- **`$( 'div:last' )`**
  - Sélectionne le dernier élément <div>.
- **`$( 'div:not(.ok)' )`**
  - Sélectionne les <div> n'ayant pas la classe "ok".
- **`$( 'div:[even|odd]' )`**
  - Sélectionne les éléments <div> de rang [pair|impair] (le premier rang est 0).
- **`$( 'div:[eq|lt|gt](n)' )`**
  - Sélectionne le ou les éléments <div> de rang [égal|inférieur|supérieur] à n.



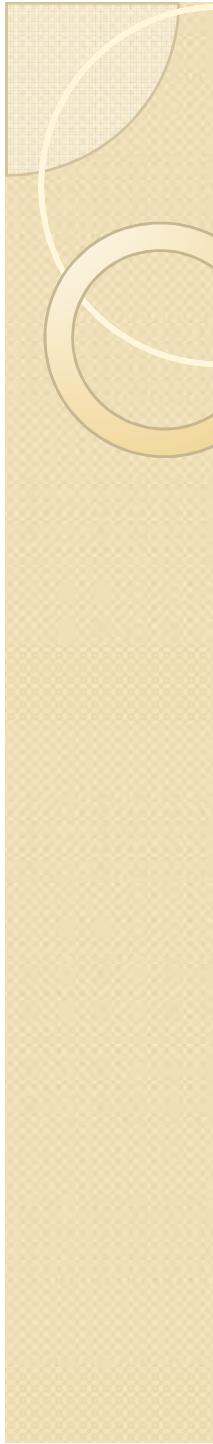
# Filtrer les sélections

- **`$( ':header' )`**
  - Sélectionne les éléments <hn>.
- **`$( ':animated' )`**
  - Sélectionne les éléments actuellement animés.
- **`$( "div:contains( 'dvp' )" )`**
  - Sélectionne les éléments <div> contenant le texte "dvp" (sensible à la casse !)
- **`$( 'div:empty' )`**
  - Sélectionne les éléments <div> vides.
- **`$( 'div:has(p)' )`**
  - Sélectionne les éléments <div> ayant un descendant <p>.



# Filtrer les sélections

- **`$( 'div:parent' )`**
  - Sélectionne les éléments <div> ayant des enfants.
- **`$( 'div:nth-child( [n|even|odd|equation] )' )`**
  - // Les enfants de <div> [de rang n|pairs|impairs| résultat de].
- **`$( 'div: [first-child|last-child]' )`**
  - Les éléments [premier|dernier] enfants d'un élément <div>.
- **`$( 'div:only-child' )`**
  - Les éléments qui sont les seuls enfants d'un élément <div>.



# Les éléments de formulaires

- `$(' :input')`
  - Tous les éléments `<input />`, `<textarea>`, `<select>` et `<button>`.
- `$(' :[text|password|radio|checkbox|submit|image|reset|button|file|hidden]')`
  - Les `<input>` du type choisi.
- `$(' :[enabled|disabled|checked|selected]')`
  - Les `<input />` possédant l'attribut indiqué.

# Exemple

```
<html>
 <script type="text/javascript"
 src="jquery.min.js">
 </script>
 <body>
 <div id="monDiv">Bonjour</div>

 disparition
 </body>
</html>
```

# Méthodes de l'objet jQuery

- Les méthodes s'appliquent généralement à tous les éléments sélectionnés
  - `$('.classe').hide(); $('.classe').show();`
  - `$('.classe').slideUp();$('.classe').slideDown();`
- De nombreuses méthodes utilitaires
  - parcourir le DOM:
    - `find("elements")` :Recherche les éléments répondant aux conditions du sélecteur spécifié. Cette méthode est utile pour retrouver les éléments descendant d'un élément donné.
    - `parent()` :retourne l'élément parent de la sélection
    - `next()` :retourne l'élément suivant de la sélection
    - `children()` :retourne l'élément fils de chaque élément de la sélection
    - `parents()` :Retourne la liste des éléments contenant les ancêtres des éléments recherchés

# Méthodes de l'objet jQuery

- Gérer les styles CSS:
  - **addClass( 'nomClasse' )** :Ajouter une classe aux éléments de la sélection
  - **removeClass( 'nomClasse' )** :Supprimer une classe aux éléments de la sélection
  - **css( 'attributCSS' , 'valeur' )** :permet de modifier les styles CSS aux éléments sélectionnés
- manipuler:
  - **append( html )** :permet d'ajouter du contenu html aux éléments de la sélection.
  - **wrap( html )**:Entoure une structure d'éléments par d'autres éléments
  - **repend( html )** :Ajoute du contenu à l'intérieur des éléments sélectionnés, au début.
  - **attr( 'nomAttribut' , 'valeur' )** :permet de modifier la valeur d'un attribut des éléments sélectionnés
- Intérêt fondamental: la plupart des méthodes de l'objet retournent l'objet lui-même
  - on peut chaîner les appels !
  - **\$( 'anything' ).parent().find('still anything').show();**



# Utilisation des sélecteurs et des événements

- JQuery nous propose 2 approches afin de sélectionner des éléments.
  - La première utilise une combinaison de sélecteurs CSS et XPath passés comme chaîne de caractères au constructeurjQuery (comme par exemple `$("div>ul a")`).
  - La seconde méthode se sert de différentes fonctions de l'objet jQuery.
- Les deux approches peuvent être combinées.



# Utilisation des sélecteurs et des événements

- Au chargement document html :

```
\$(document).ready (
 function(){
 // Code à exécuter au chargement de la page
 }
) ;
```

- Ou encore :

```
\$(
 function(){
 // Code à exécuter au chargement de la page
 }
) ;
```

## Exemple 2:

```
$document.ready(function()
{
 $("#valider").click(function()
 {
 $("#form").submit();
 });
});
```

- Le code précédent permet, quand on click sur le bouton valider, de sélectionner l'élément ayant l'id « form » et appelle sa méthode *submit()*



# Widgets et Effets Jquery

<http://jqueryui.com/>

- Après les interactions, il va falloir nous lancer dans les **widgets** de jQuery UI.
- Ce sont des plugins plus orientés dans le confort des utilisateurs.
- De même, la librairie met à disposition des effets complémentaires, introuvables dans jQuery, et qui sont vraiment très sympathiques.



## Fichiers à inclure dans l'entête de la page HTML

```
<head>
 <link rel="stylesheet" href="jquery-ui.css" />
 <script src="jquery-1.8.2.js"></script>
 <script src="jquery-ui.js"></script>
</head>
```

# Création d'un accordéon

Section1

Contenu Section 1

Section2

Section3

Section1

Section2

Contenu Section 2

Section3

Section1

Section2

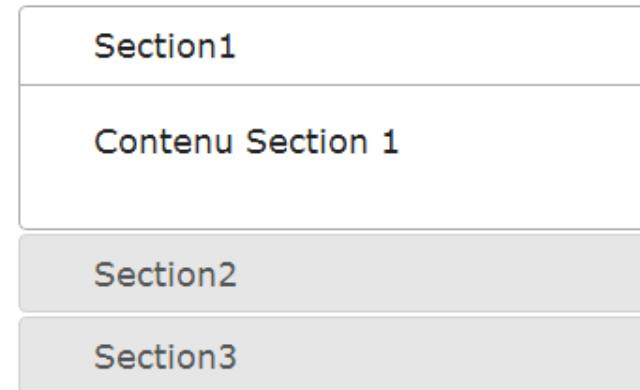
Section3

Contenu Section 3

# Structure d'un accordéon jQuery

- **Structure d'un accordéon JQuery (Code HTML)**

```
<div id="accordeon">
 <h3>Section1</h3>
 <div>Contenu Section 1</div>
 <h3>Section2</h3>
 <div>Contenu Section 2</div>
 <h3>Section3</h3>
 <div>Contenu Section 3</div>
</div>
```



- **Comportement de l'accordéon (Code Java Scrit) :**

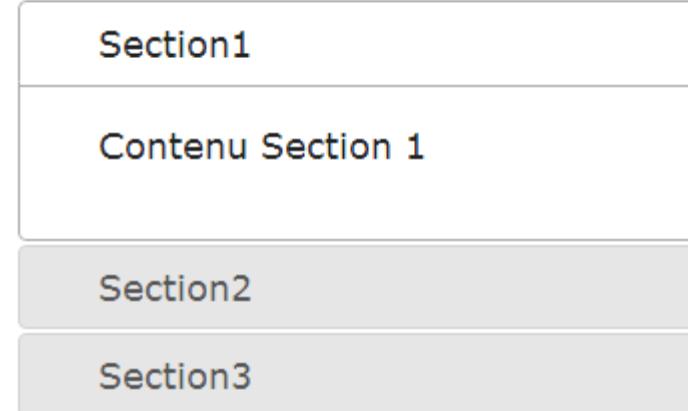
```
<script language="javascript">
 $(function(){
 $("#accordeon").accordion();
 })
</script>
```

- **Pour Que la section 2 soit active au début au lieu de la première :**

```
$("#accordeon").accordion({active:1});
```

# Code Complet de la page

```
<html>
<head>
 <title>Accordéon JQuery</title>
 <script type="text/javascript" src="jquery-1.8.2.js"></script>
 <script type="text/javascript" src="jquery-ui.js"></script>
 <link rel="stylesheet" type="text/css" href="jquery-ui.css" />
</head>
<body>
 <div id="accordeon">
 <h3>Section1</h3>
 <div>Contenu Section 1</div>
 <h3>Section2</h3>
 <div>Contenu Section 2</div>
 <h3>Section3</h3>
 <div>Contenu Section 3</div>
 </div>
 <script language="javascript">
 $(function(){
 $("#accordeon").accordion();
 })
 </script>
</body>
</html>
```



# Une autre manière pour créer l'accordéon

- **Structure d'un accordillon JQuery (Code HTML)**

```
<ul id="accordeon2">
 <li class="titre">Section 1
 Contenu de la section 1

 <li class="titre">Section 2
 Contenu de la section 2

 <li class="titre">Section 3
 Contenu de la section 3

```

- **Comportement de l'accordéon (Code Java Scrit) :**

```
<script language="javascript">
$(function(){
 $("#accordeon2").accordion({header:'.titre'});
})
</script>
```

# Code Complet de la page

```
<html>
<head>
 <title>Accordéon JQuery</title>
 <script type="text/javascript" src="jquery-1.8.2.js"></script>
 <script type="text/javascript" src="jquery-ui.js"></script>
 <link rel="stylesheet" type="text/css" href="jquery-ui.css" />
 <style>
 .titre{
 background:#FFCCCC;
 }
 </style>
</head>
<body>
 <ul id="accordeon2">
 <li class="titre">Section 1
 Contenu de la section 1
 <li class="titre">Section 2
 Contenu de la section 2
 <li class="titre">Section 3
 Contenu de la section 3

 <script language="javascript">
 $(function(){
 $("#accordeon2").accordion({header:' .titre '});
 });
 </script>
</body>
</html>
```

Section 1

Section 2

Contenu de la section 2

Section 3

# Panneau à Onglets

- Structure : (HTML)

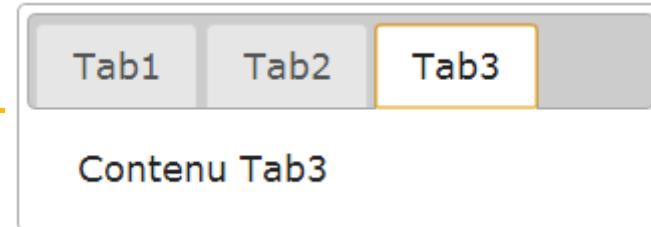
```
<div id="tabs">

 Tab1
 Tab2
 Tab3

 <div id="tabs-1"> Contenu Tab1</div>
 <div id="tabs-2">Contenu Tab2</div>
 <div id="tabs-3">Contenu Tab3 </div>
</div>
```

- Comportement : (JavaScript)

```
<script language="javascript">
 $(function(){
 $("#tabs").tabs();
 })
</script>
```



# Menu déroulant

- **Structure : (HTML)**

```
<ul id="menu">
 <li class="ui-state-disabled">XML
 HTML
 CSS
 Java SCript
 Programmation

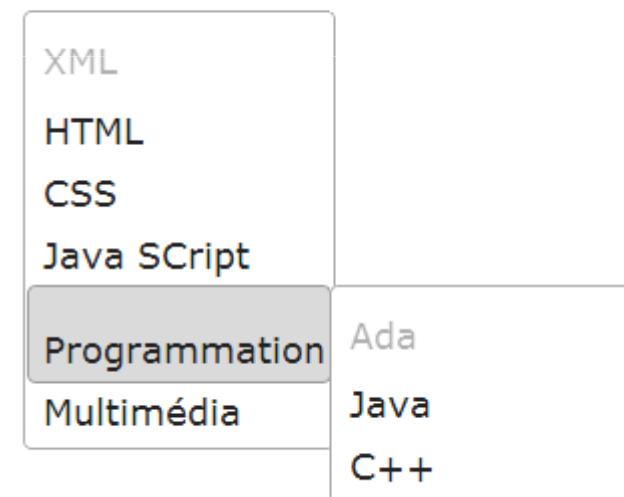
 <li class="ui-state-disabled">Ada
 Java
 C++

 Multimédia

```

- **Comportement : (JavaScript)**

```
<script language="javascript">
$(function(){
 $("#menu").menu();
})
</script>
```



# DatePicker : Calendrier

Date:

November 2012

Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

Date: 11/07/2012

# DatePicker

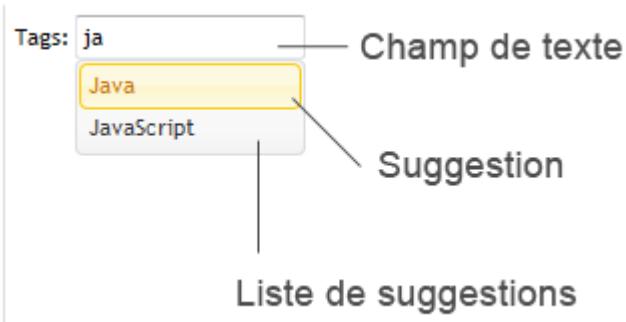
```
<html>
<head>
<title>Date Picker</title>
<script type="text/javascript" src="jquery-1.8.2.js"></script>
<script type="text/javascript" src="jquery-ui.js"></script>
<link rel="stylesheet" type="text/css" href="jquery-ui.css" />
</head>

<body>
<script>
$(function() {
 $("#dateDepart").datepicker();
});
</script>
Date:<input type="text" name="date" id="dateDepart" />
</body>
</html>
```



# Autocomplete

- L'autocomplétion est un widget qui était déjà présent dans la version 1.6 de jQuery UI, mais qui avait été retiré dans la version suivante, la 1.7.
- Pour la version 1.8, les développeurs de la librairie se sont penchés dessus et l'ont amélioré, permettant ainsi de le proposer de nouveau en libre téléchargement.
- Traditionnellement, ce plugin est utilisé dans les champs de recherche, mais vous pouvez tout aussi bien l'implémenter dans une suggestion de pseudonyme ou de prénom, par exemple.



# Autocomplete

- **Code HTML :**

```
<div class="ui-widget">
 <form>
 Tags: <input type="text" id="recherche" />
 </form>
</div>
```

Tags:

j

Clojure  
Java  
JavaScript

- **Code Java Script :**

```
<script>
 $(function() {
 var liste= [
 "ActionScript", "AppleScript", "Asp", "BASIC", "C",
 "C++", "Clojure", "COBOL", "ColdFusion", "Erlang",
 "Fortran", "Groovy", "Haskell", "Java", "JavaScript",
 "Lisp", "Perl", "PHP", "Python", "Ruby", "Scala", "Scheme"
];
 $('#recherche').autocomplete({
 source: liste
 });
 });
</script>
```

# Autocomplete : Paramètres

- ***Spécification d'une longueur minimum***

- Il existe une option qui permet d'exiger une longueur minimum concernant la chaîne de caractère tapée par l'utilisateur. En effet, par défaut, ce paramètre est défini à 1.

```
$('#recherche').autocomplete({
 source : [...],
 /* on inscrit la liste de suggestions*/
 minLength : 3
 /* on indique qu'il faut taper au moins 3
 caractères pour afficher l'autocomplétion*/
});
```

# Autocomplete : Paramètres

- ***Retour sur l'option « source »***

- La liste des paramètres pouvant être définis peut être allongée indéfiniment, mais on utilisera, en général, la syntaxe suivante :
  - value, qui permet d'indiquer la valeur à inscrire dans le champ de texte une fois sélectionnée ;
  - label, qui désigne en fait le titre de la suggestion ;
  - desc, qui donne la possibilité de spécifier une description à chaque proposition ;
  - icon, qui devra prendre une image pour valeur.

# Autocomplete : Paramètres

- **Exemple :**

```
$function(){
 var liste = [
 { value : 'Draggable', label : 'Draggable', desc :
'L\'interaction Draggable permet de déplacer un élément.' },
 { value : 'Droppable', label : 'Droppable', desc :
'L\'interaction Droppable permet de recevoir un élément.' },
 { value : 'Resizable', label : 'Resizable', desc :
'L\'interaction Resizable permet de redimensionner un
élément.' }
];
$('#recherche').autocomplete({
 source : liste,
 select : function(event, ui){
 /* lors de la sélection d'une proposition */
 $('#description').append(ui.item.desc);
 /* on ajoute la description de l'objet dans un bloc */
 }
});
```

L'interaction Draggable permet de déplacer un élément.

Recherche: Draggable

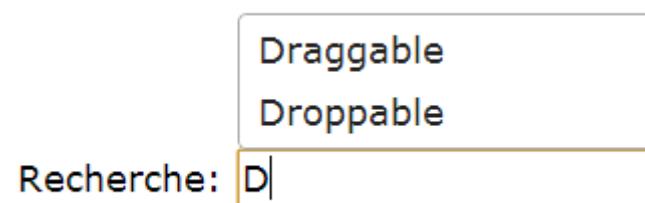
# Autocomplete : Paramètres

- ***l'option « position »***

- Par défaut, la liste de suggestions se place en dessous du champ de texte qui lui est associé.
- Il se peut que ça ne vous convienne pas, et comme le mot d'ordre de la librairie est toujours « flexibilité », vous pouvez définir une nouvelle position grâce à un paramètre.
- Le paramètre **position** prend pour valeur un objet, contenant les positions suivantes :
  - **my**, qui définit la position de la liste autour du champ de texte ;
  - **at**, qui définit la position de la liste dans le champ de texte ;

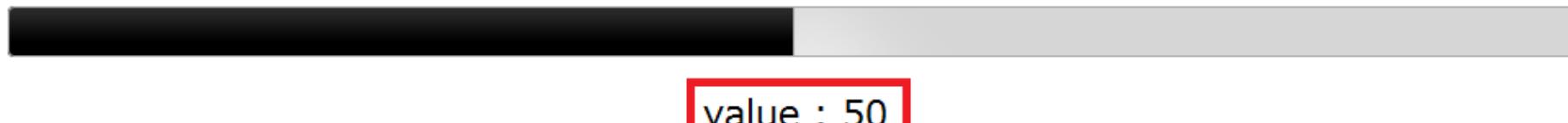
- **Exemple :**

```
$('#recherche').autocomplete({
 source : [...],
 position : {
 my : 'bottom',
 at : 'top'
 } /* ici, ma liste se placera au-dessus
 champ de texte. */
});
```



# Barre de progression

- La barre de progression fonctionne grâce à un bloc div, dans lequel s'implantera un deuxième bloc possédant une largeur variable, qui fera donc office de progression.
- Bien sûr, cela fonctionne en interne, grâce à la librairie, ce qui nous permet de n'avoir qu'un seul bloc à déclarer.
- Code JavaScript :
  - **\$('#barre').progressbar();**
- Le plugin possède un paramètre value, et permet d'indiquer la valeur de progression, en pour cents (0-100%).
  - **\$('#barre').progressbar({**
  - **value : 50 // remplit 50% (la moitié) de la barre**
  - **});**



value : 50

# Boites de dialogue

```
<div id="dialog" title="Boîte de dialogue de base">
```

Cette boîte de dialogue peut être redimensionnée, déplacée et fermée.

```
</div>
```

```
<script>
```

```
$(function() {
```

```
 $("#dialog").dialog({width:300,height:200});
```

```
</script>
```

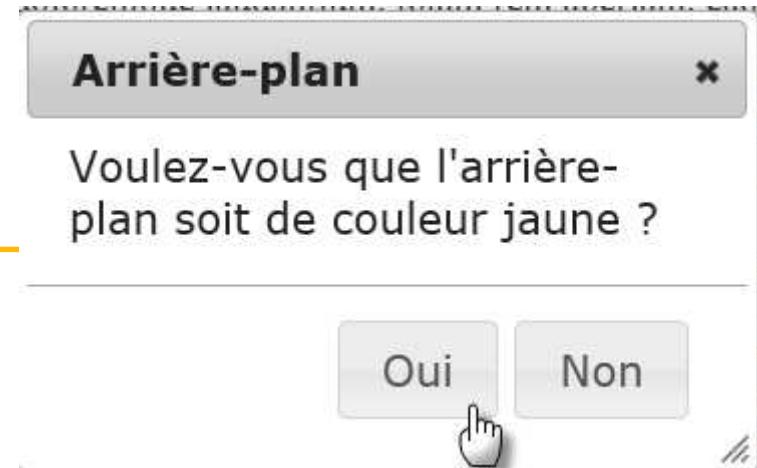
Boîte de dialogue de  
base

Cette boîte de dialogue peut  
être redimensionnée,  
déplacée et fermée.

Options	Signification
height et width	Hauteur et largeur de la boîte de dialogue à l'ouverture.
modal	Initialisé à <b>true</b> , rend la boîte de dialogue modale (c'est-à-dire interdit l'accès à la page). La valeur par défaut est <b>false</b> .
position	Position de la boîte de dialogue sur la page (elle est centrée par défaut).
zindex	Z-index de la boîte de dialogue (1000 par défaut).
buttons	Un ou plusieurs boutons affichés dans la boîte de dialogue.

# Boites de dialogue

```
<script>
$(function() {
 $('#dialog').dialog({
 modal: true,
 buttons: {
 "Oui": function() {
 $('body').css('background', 'yellow');
 $(this).dialog("close");
 },
 "Non": function() {
 $(this).dialog("close");
 }
 });
 });
</script>
```



# Interactions:

- **Selectable :**

- JavaScript

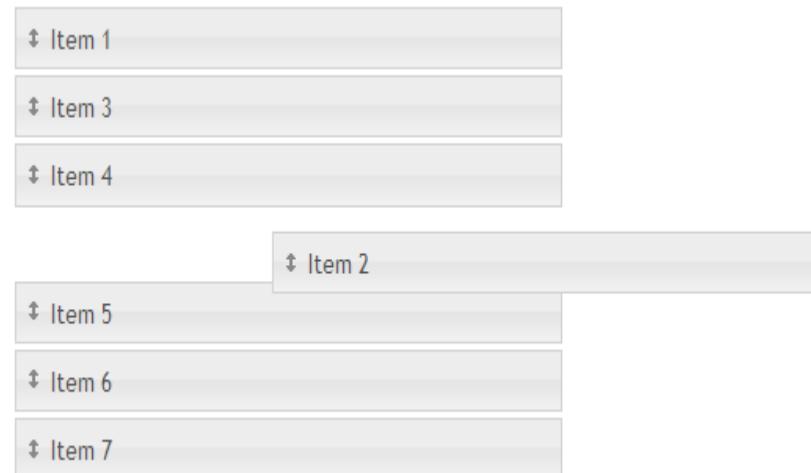
```
<script>
$(function() {
 $("#selectable").selectable();
});
</script>
```



- **Sortable:**

- JavaScript

```
$(function() {
 $("#sortable").sortable();
 $("#sortable").disableSelection();
});
```



# Interactions:

- **Resizable :**

- JavaScript :

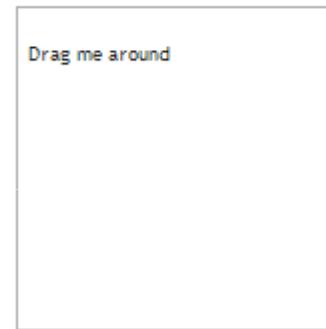
```
$("#resizable").resizable();
```



- **Draggable :**

- JavaScript :

```
$("#draggable").draggable();
```



- **Dropable :**

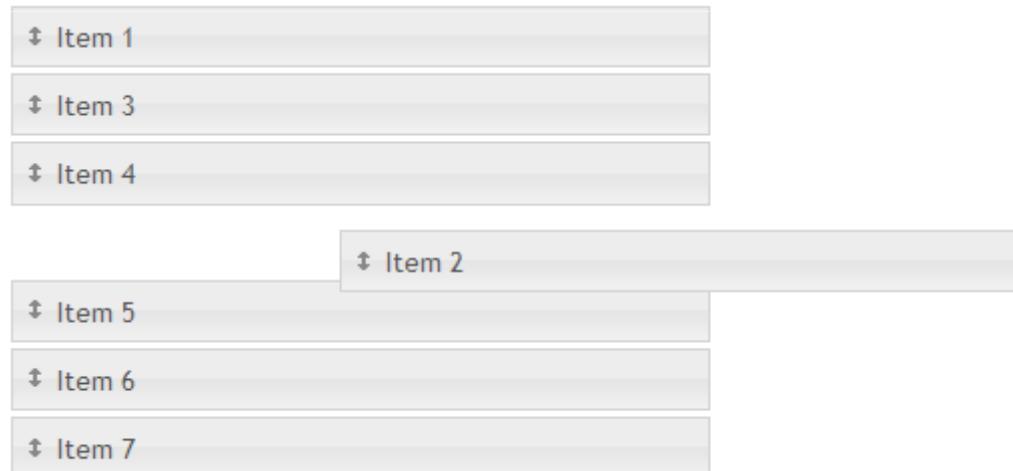
- JavaScript :

```
$("#dropable").droppable();
```

# Interaction : Sotable

- Code JavaScript :

```
<script>
$(function() {
 $("#sortable").sortable();
});
</script>
```





# Interaction : Dragable

- Code JavaScript :

```
<script>
 $(function() {
 $("#draggable").draggable();
 });
</script>
```

# Animation : Exemple

```
<div id="contenu">
 <h3>Titre</h3>
 <p>Contenu</p>
</div>
<script>
$(function() {
 $('#contenu').show('explode');
 $('#contenu').show('slide');
 $('#contenu').show('fold');
});
</script>
```





# Animation des couleurs

- vous pourrez animer la couleur des éléments ! Les propriétés sur lesquelles vous pouvez agir sont les suivantes :  
**backgroundColor**
  - borderBottomColor
  - borderLeftColor
  - borderRightColor
  - borderTopColor
  - color
  - outlineColor
- La syntaxe de la méthode animate() ne change pas :

```
$('sel').animate({ prop1: val1, prop2: val2 }, durée,
modèle, function() {
 //Une ou plusieurs instructions
});
```

# Exemple d'animation des couleurs

## - Code HTML :

```
<div id="contenu">
 <h3>Titre</h3>
 <p>Contenu Contenu Contenu
 Contenu Contenu Contenu</p>
</div>
```

## - Code CSS :

```
#contenu {
 border: 4px gray solid;
 background-color: #aaeae1;
 color: black; width: 100px;
 position: relative;
}
#contenu h3{
 margin: 0; padding: 0.4em;
 text-align: center;
 background-color: #777;
}
```

## - Code JavaScript :

```
$(function() {
 $('#contenu').toggle(
 function() {
 $('#contenu').animate({
 backgroundColor: '#fff',
 color: 'red',
 left: '+=200',
 width: 500
 }, 1000);
 },
 function() {
 $('#contenu').animate({
 backgroundColor: '#aaeae1',
 color: 'black',
 left: '-=200',
 width: 100
 }, 1000);
 });
});});
```

Titre

Contenu  
Contenu  
Contenu  
Contenu  
Contenu  
Contenu

Titre

Contenu Contenu Contenu Contenu Contenu

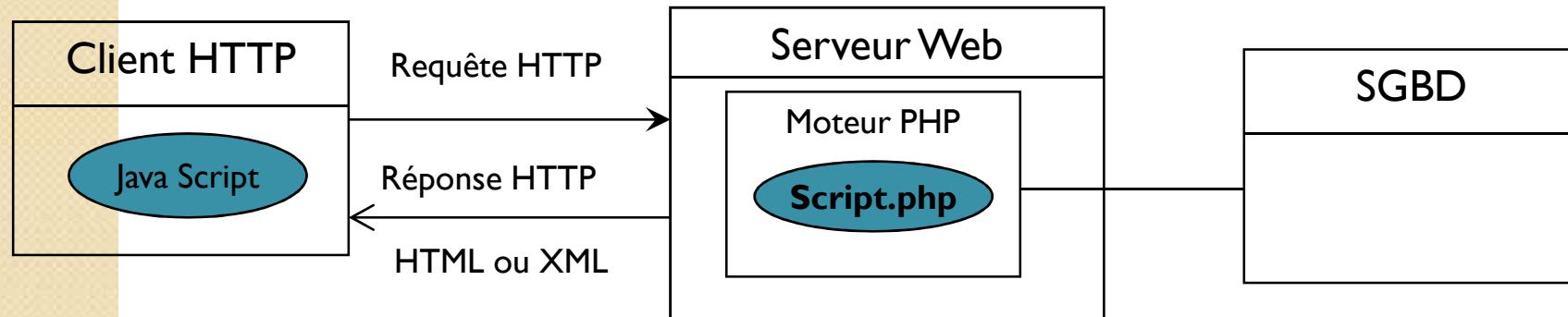
Titre

Contenu  
Contenu  
Contenu  
Contenu  
Contenu  
Contenu



AJAX

# Principe de base HTTP





# Qu'est ce qu'AJAX ?

- **Asynchronous JavaScript And XML.**
- Ajax a exploité les technologies XML et java script
- AJAX est un concept qui permet de faire des appels asynchrones au serveur depuis le client.
- Lors de ces appels, le serveur retournera du XML qui sera "récupéré" par javascript et traité.
- Nous verrons que nous pouvons tout aussi bien faire transiter du texte et faire des appels synchrones si l'on veut.



# Problématique

- Avant toute chose, il serait bon de faire un point sur le processus classique de consultation d'un site ou d'une application web :
  - Vous saisissez une adresse dans votre navigateur.
  - Cette "requête" finie par arriver sur le serveur web qui héberge la page en question.
  - Le serveur vous retourne du texte au format HTML ou XHTML et éventuellement des images, feuilles de style, fichiers JavaScript, applets java ....
  - Votre navigateur les interprète et vous affiche la page.
  - Vous êtes déconnecté du serveur web.
- Donc, quand vous cliquez sur un lien, vous recommencez ce processus en entier avec une nouvelle page.
  - Dans le cas où un formulaire se trouve sur la page, vous envoyez les données sur le serveur qui vous répondra après traitement de ces données.



# Apport de ajax

- L'utilisation d'AJAX va chambouler un peu cette organisation car à tout moment vous pouvez aller chercher des informations sur le serveur pour :
  - Ajouter des éléments a la page
  - Modifier le contenu d'un "bout de la page"
  - Insérer des données dans une base.
  - ...



# Utilisation de Ajax

- L'objet **XmIHttpRequest**
  - AJAX se base sur l'utilisation d'un composant embarqué dans presque tous les navigateurs récents.
  - Par contre, vous vous doutez bien que le comportement va varier en fonction de ces derniers.
  - Pour pouvoir utiliser AJAX, il nous faut donc créer en javascript un objet que l'on nomme **XmIHttpRequest** ou **xhr** pour les intimes,
  - comme son nom l'indique, cet objet nous permet de faire des requêtes http pour échanger du XML.

# Création de l'objet XMLHttpRequest

```
var xhr = null;
if(window.XMLHttpRequest) // Firefox et autres
xhr = new XMLHttpRequest();
else if(window.ActiveXObject){ // Internet Explorer
try {
 xhr = new ActiveXObject("Msxml2.XMLHTTP");
}
catch (e) {
 xhr = new ActiveXObject("Microsoft.XMLHTTP");
}
}else {
 // XMLHttpRequest non supporté par le navigateur
 alert("Votre navigateur ne supporte pas les objets
 XMLHttpRequest...");
 xhr = false;
}
```



## Propriétés et méthode de l'objet xhr

- **open("méthode","url",flag):**
  - Ouvre la connexion avec le serveur.
    - **méthode** -> "GET" ou "POST"
    - **url** -> l'url à laquelle on va envoyer notre requête.
      - Si la méthode est GET, on met les paramètres dans l'url.
    - **flag** -> true si l'on veut un dialogue asynchrone, sinon, false



## Propriétés et méthode de l'objet xhr

- **setRequestHeader("nom","valeur"):**
  - Assigne une valeur à un header HTTP qui sera envoyé lors de la requête.
  - Par exemple, pour un POST :
    - nom -> "Content-Type"
    - valeur -> "text/xml"



# Propriétés et méthode de l'objet xhr

- **send("params"):**
  - Envoi la requête au serveur.
  - Si la méthode est GET, on met null en paramètre.
  - Si la méthode est POST, on met les paramètres à envoyer, sous la forme :
    - "nomparam1=valeur1&nomparam2=valeur2".
- **abort()** : Abandonne la requête.
- **onreadystatechange :**
  - Ici, on va lui affecter notre fonction java script qui sera exécutée à chaque "changement d'état" de notre objet.



## Propriétés et méthode de l'objet xhr

- **readyState :**

- C'est cette propriété qu'on va tester dans le onreadystatechange.
- Elle représente l'état de l'objet et peut prendre plusieurs valeurs :
  - 0 -> Non initialisé.
  - 1 -> Ouverture (open() vient de s'exécuter).
  - 2 -> Envoyé (send() vient de s'exécuter).
  - 3 -> En cours (des données sont en train d'arriver).
  - 4 -> Prêt (toutes les données sont chargées).

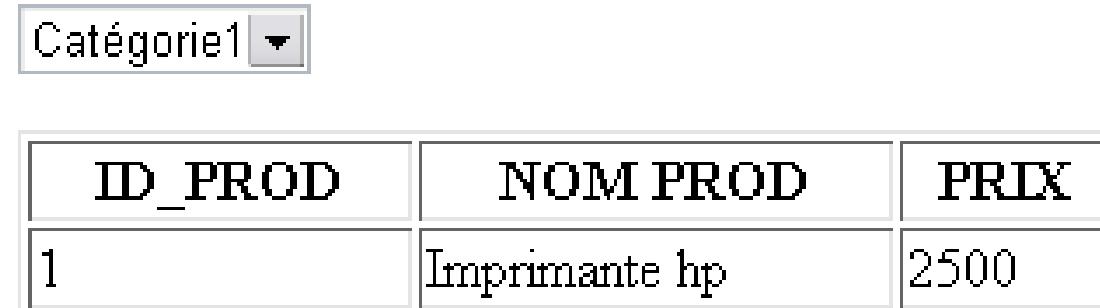


## Propriétés et méthode de l'objet xhr

- **status :**
  - **Le code de la réponse du serveur.**
    - **200 -> OK.**
    - **404 -> Page non trouvée.**
    - ...
- **statusText :** Le message associé à status.
- **responseText :** La réponse retournée par le serveur, au format texte.
- **responseXML:** La réponse retournée par le serveur, au format dom XML.

# Application AJAX PHP

- On souhaite créer une application web, utilisant ajax et JSP, qui permet de présenter un catalogue de produits en ligne. Chaque produit appartient à une catégorie.
- On considère une base de données MYSQL nommée « Catalogue » qui contient deux tables :
  - CATEGORIES (ID\_CAT, NOM\_CAT)
  - PRODUITS (ID\_PROD,NOM\_PROD, PRIX, #ID\_CAT)
- L'application que l'on souhaite créer doit permettre :
  - Au chargement, afficher toutes les catégories dans une zone de liste
  - En sélectionnant une catégorie, afficher les produits de cette catégorie dans un tableau.



The screenshot shows a user interface for an application. At the top, there is a dropdown menu labeled "Catégorie1" with a downward arrow icon. Below it is a table with three columns: "ID\_PROD", "NOM PROD", and "PRIX". The table has two rows of data. The first row contains the values "1", "Imprimante hp", and "2500".

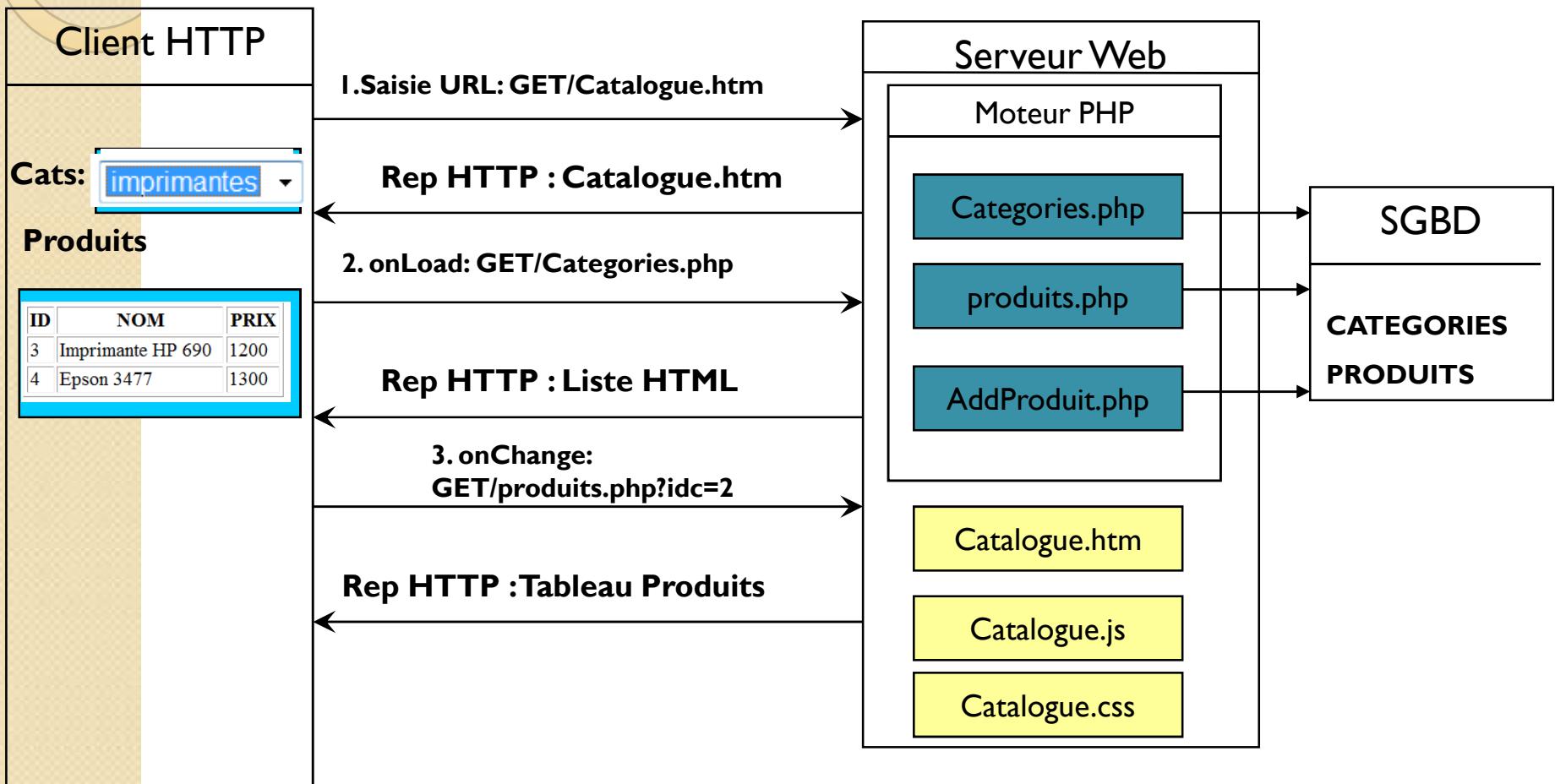
ID_PROD	NOM PROD	PRIX
1	Imprimante hp	2500



## Première solution Ajax avec HTML

- Créer un script PHP qui permet de générer une liste déroulante HTML qui contient toutes les catégories
- Créer un script PHP qui permet de générer un tableau HTML qui contient les produits d'une catégorie donnée.
- Créer une page HTML avec le code Ajax qui permet de :
  - Au chargement afficher la liste des catégories dans un div de la page
  - En sélectionnant une catégorie, afficher les produits de cette catégorie dans un autre div.

# Architecture



# Catalogue.htm

```
<html>
 <head>
 <script language="javascript" src="catal.js">
 </script>
 </head>
 <body onload="chargerCategories()">
 Catégories:
 <div id="cats" style="display:inline"></div>
 <hr/>
 Produits:
 <div id="prods"></div>
 </body>
</html>
```

# catal.js

```
// JavaScript Document
function getXhr(){
 var xhr = null;
 if(window.XMLHttpRequest)
 // Firefox et autres
 xhr = new XMLHttpRequest();
 else if(window.ActiveXObject){
 // Internet Explorer
 try {
 xhr = new ActiveXObject("Msxml2.XMLHTTP");
 catch (e) { xhr = new ActiveXObject("Microsoft.XMLHTTP"); }
 }
 else {
 // XMLHttpRequest non supporté par le navigateur
 alert("Le navigateur ne supporte pas les objets
 XMLHttpRequest...");}
 xhr = false; }
 return xhr;
}
```

# Catal.js (communication synchrone)

```
function chargerCategories(){
 var xhr=getXhr();
 xhr.open("POST","cat.php",false);
 xhr.send("");
 var rep=xhr.responseText;
 document.getElementById("cat").innerHTML=rep;
}

function chargerProduits(idc){
 var xhr=getXhr();
 xhr.open("GET","prods.php?cat="+idc,false);
 xhr.send(null);
 var rep=xhr.responseText;
 document.getElementById("prod").innerHTML=rep;
}
```

# Catal.js (communication asynchrone)

```
function chargerCategories(){
 var xhr=getXhr();
 xhr.open("GET","cat.php",true);
 xhr.onreadystatechange=function(){
 if((xhr.readyState==4)&&(xhr.status==200)){
 var rep=xhr.responseText;
 document.getElementById("cat").innerHTML=rep;
 }
 }
 xhr.send(null);
}

function chargerProduits(idc){
 var xhr=getXhr();
 xhr.open("GET","prods.php?cat="+idc,true);
 xhr.onreadystatechange=function(){
 if((xhr.readyState==4)&&(xhr.status==200)){
 var rep=xhr.responseText;
 document.getElementById("prod").innerHTML=rep;
 }
 }
 xhr.send(null);
}
```

# Ajax avec JQuery : catalogue.js

```
$(document).ready(function(){// Au chargement du document
 chargerCategories(); // Faire appel à la fonction chargerCategories()
});
function chargerCategories(){
// Envoyer une requête Ajax avec la méthode GET au script categories.php
$.ajax({
 url:'categories.php',type:'GET',dataType:'html',
 success:function(reponse,status){ // En cas de succès
 $("#cats").html(reponse); // Afficher la réponse dans le div cats
 $("#cat").change(function(){chargerProduits($(this).val())});
 /* En modifiant la valeur de la liste, faire appel à la fonction
 chargerProduits(valeur de idCat sélectionnée) */
 }
});
}
function chargerProduits(idc){
$.ajax({
 url:'produits.php',type:'GET',dataType:'html',data:'cat='+idc,
 success:function(reponse,status){
 $("#prods").html(reponse);
 },
 error:function(resultat,status,erreur){ // En cas d'erreur
 $("#prods").html(erreur); // Afficher l'erreur
 }
});
}
```

## Ajax avec JQuery : les fonctions get() et post() de JQuery

```
$(document).ready(function(){
 chargerCategories();
});
function chargerCategories(){
 $.get('categories.php',function(reponse){
 $("#cats").html(reponse);
 $("#cat").change(function(){chargerProduits($(this).val())});
 })
}
function chargerProduits(idc){
 $.get('produits.php','cat='+idc,function(reponse){
 $("#prods").html(reponse);
 })
}
```

# JQuery et JSON

- Application : Consulter les produits par catégorie

The screenshot shows a web browser window with the following details:

- Address bar: localhost:8888/TPAJ/catalogueJQueryJSON.html
- Content area:
  - Catégories: **imprimantes** (highlighted with a yellow border)
  - A table displaying product information:

ID	NOM	PRIX
3	Imprimante HP 690	1200
4	Epson 3477	1300

# Catalogue.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>TP Ajax</title>
<script language="javascript" src="jquery-1.8.2.js"></script>
<script language="javascript" src="jquery-ui.js"></script>
<script language="javascript" src="cataloguejQueryJSON.js"></script>
<link rel="stylesheet" type="text/css" href="jquery-ui.css" />
</head>
<body>
 Catégories:<div id="cats" style="display:inline"></div>
 <div id="prods"></div>
</body>
</html>
```

## **cataloguejQueryJSON.js**

```
$(document).ready(function(){
 chargerCategories();
});

function chargerCategories(){
 $.getJSON('categoriesJSON.php', function(data){
 var res='<select name="cat" id="cat"
 onchange="chargerProduits(this.value)">';
 $.each(data, function(key,categorie){
 res+='' +
 categorie['NOM_CAT']+ '</option>';
 })
 res+='</select>';
 $("#cats").html(res);
 });
}
```

# cataloguejQueryJSON.js

```
function chargerProduits(idc){
$.getJSON('produitsJSON.php?cat='+idc, function(data){
 var res='<table border="1">';
 res+='| ID | NOM | PRIX |
';
 $.each(data, function(key,produit){
 res+='|';
 res+=' ' + produit['ID_PROD'] + ' |';
 res+=' ' + produit['NOM_PROD'] + ' |';
 res+=' ' + produit['PRIX'] + ' |';
 res+='
';
 })
 res+='';
 $('#prods').html(res);
})
});
}
```

## **Une autre façon pour générer HTML avec JQuery**

```
function chargerCategories(){
 $.getJSON('categoriesJSON.php', function(data){
 $liste=$("<select>");
 $liste.attr("name","cat");
 $liste.attr("id","cat");
 $liste.attr("onchange","chargerProduits(this.value)");
 $.each(data, function(key,categorie){
 $option=$("<option>");
 $option.attr("value",categorie['ID_CAT']);
 $option.append(categorie['NOM_CAT']);
 $liste.append($option);
 })
 });
 $("#cats").html($liste);
}
});
```

## **Une autre façon pour générer HTML avec JQuery**

```
function chargerProduits(idc){
$.getJSON('produitsJSON.php?cat=' +idc, function(data){
 $tab=$('<table border="1">');
 $tab.append($('<tr><th>ID</th><th>NOM</th><th>PRIX</th>
<th>ID CAT</th></tr>'));
 for(i in data){
 $tr=$("<tr>");
 for(attr in data[i]){
 $td=$("<td>").append(data[i][attr]);
 $tr.append($td);
 }
 $tab.append($tr);
 }
 $("#prods").html($tab);
}
};
};
```



# Client mobile Androïde

Voir Support : Application module android.pdf