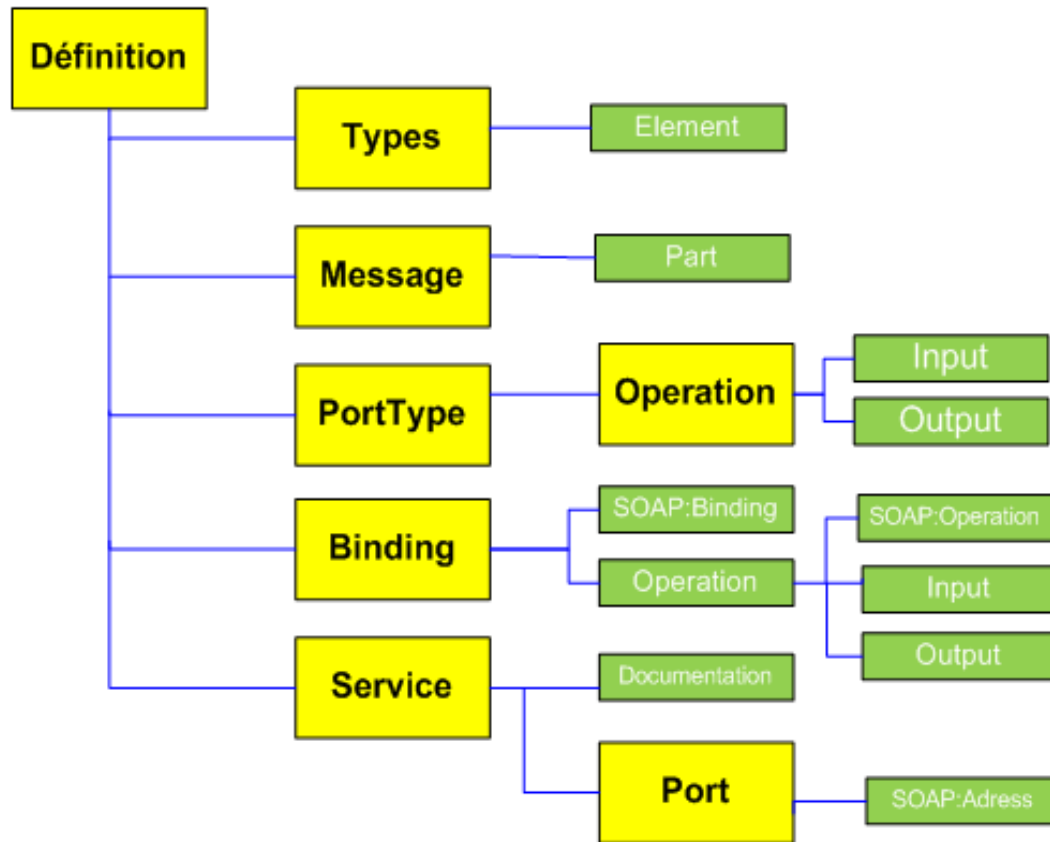


# Les Services Web

- **WSDL – Décrire, configurer, et personnaliser un Service Web**

# Contrat Service Web - WSDL



Structure d'un document WSDL

**<wsdl:definitions>**

**<wsdl:types />**

**<wsdl:message />**

**<wsdl:portType>**

**<wsdl:operation>**

**<wsdl:input />**

**<wsdl:output />**

**</wsdl:operation>**

**</wsdl:portType>**

**<wsdl:binding />**

**<wsdl:service />**

**</wsdl:definitions>**

# Contrat Service Web - WSDL

## Annotations et personnalisation du WSDL

- JAX-WS repose sur l'utilisation massive d'annotations pour configurer un service web et customiser son document WSDL.
- Les principales **annotations** sont les suivantes :
  - **@WebService**: annote une interface/classe Java pour décrire/implémenter un service web (annotation obligatoire).
  - **@WebMethod** : Paramétrer une opération.
  - **@WebParam** : Paramétrer un message paramètre.
  - **@WebResult** : Paramétrer un message de sortie (return).
  - **@OneWay** : Message input seulement.

## Contrat Service Web - WSDL

1. **@WebService**: *name*, *serviceName*, *portName*

```
@WebService(name="Hello")
```

```
public interface HelloWorld {
```

```
    @WebMethod
```

```
    public String simpleHello();
```

```
    @WebMethod
```

```
    public String makeHello(String name);
```

```
}
```

```
<wsdl:portType>
```

```
<wsdl:operation>
```

```
<wsdl:input />
```

```
<wsdl:output />
```

```
</wsdl:operation>
```

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

# Contrat Service Web - WSDL

## 1. **@WebService**: *name*, **serviceName**, **portName**

```
@WebService(endpointInterface = "org.soa.ws.tp.HelloWorld",  
             serviceName="HelloService",  
             portName="HelloPort")
```

```
public class HelloWorldImpl implements HelloWorld{
```

```
    @Override
```

```
    @WebMethod
```

```
    public String simpleHello() {  
        return "Hello World";  
    }
```

```
}
```

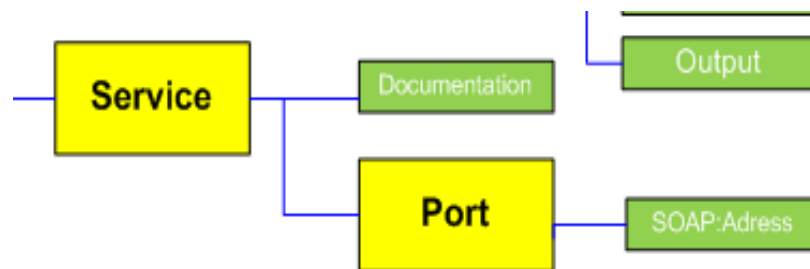
```
    @Override
```

```
    @WebMethod
```

```
    public String makeHello(String name) {  
        return "Hello World, " + name;  
    }
```

```
}
```

```
}
```



## Contrat Service Web - WSDL

### 1. **@WebMethod**: **operationName**, *exclude*

```
@WebService(name="Hello")  
public interface HelloWorld {
```

```
    @WebMethod(operationName="operationSimple")  
    public String simpleHello();
```

```
    @WebMethod(operationName="operationMake")  
    public String makeHello(String name);
```

```
}
```

```
<wsdl:portType>  
    <wsdl:operation>  
        <wsdl:input />  
        <wsdl:output />  
    </wsdl:operation>  
</wsdl:portType>
```

## Contrat Service Web - WSDL

### 1. **@WebMethod**: *operationName*, **exclude**

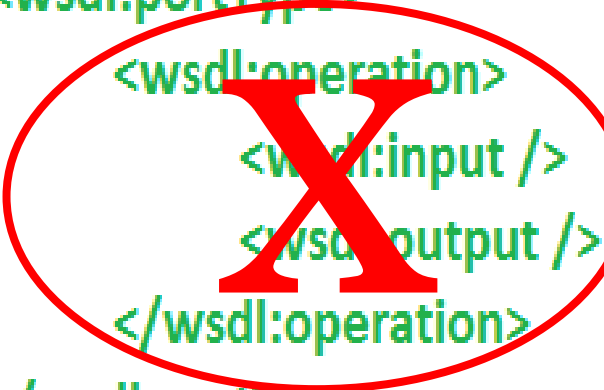
```
@WebService(name="Hello")  
public interface HelloWorld {
```

```
    @WebMethod(operationName="operationSimple")  
    public String simpleHello();
```

```
    @WebMethod(exclude=true)  
    public String makeHello(String name);
```

```
}
```

```
<wsdl:portType>  
  <wsdl:operation>  
    <wsdl:input />  
    <wsdl:output />  
  </wsdl:operation>  
</wsdl:portType>
```



# Contrat Service Web - WSDL

## 1. @OneWay:

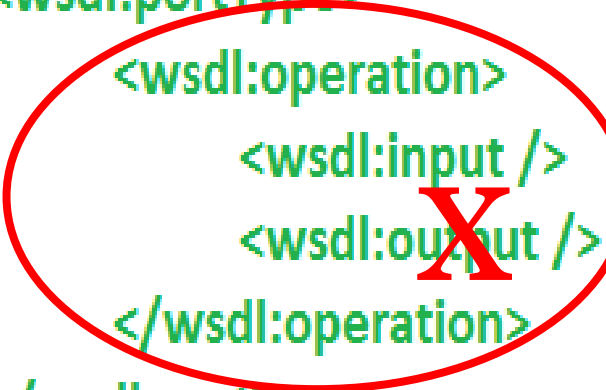
```
@WebService(name="Hello")
public interface HelloWorld {

    @WebMethod
    public String simpleHello();

    @WebMethod
    @OneWay
    public String makeHello(String name);

}
```

```
<wsdl:portType>
  <wsdl:operation>
    <wsdl:input />
    <wsdl:output />
  </wsdl:operation>
</wsdl:portType>
```





## Contrat Service Web - WSDL

### 1. @WebParam: *name*

<wsdl:types />  
<wsdl:message />

```
@WebService(name="Hello")  
public interface HelloWorld {
```

```
    @WebMethod(operationName="operationSimple")  
    public String simpleHello();
```

```
    @WebMethod  
    public String makeHello(@WebParam(name="nom") String name);  
}
```

## Contrat Service Web - WSDL

### 1. @WebParam: *name*

`<wsdl:types />`  
`<wsdl:message />`

```
@WebService(name="Hello")  
public interface HelloWorld {
```

```
    @WebMethod(operationName="operationSimple")  
    public String simpleHello();
```

```
    @WebMethod  
    public String makeHello(@WebParam(name="nom") String name,  
                           @WebParam(name="age") String age);  
}
```

## Contrat Service Web - WSDL

### 1. @WebResult: *name*

<wsdl:types />  
<wsdl:message />

```
@WebService(name="Hello")  
public interface HelloWorld {
```

```
    @WebMethod(operationName="operationSimple")  
    public String simpleHello();
```

```
    @WebMethod
```

```
    @WebResult(name="MonResultat")
```

```
    public String makeHello(@WebParam(name="nom") String name);  
}
```

# Contrat Service Web - WSDL

## Exercice 1 – Série TP 3

Ecrire un service web Convertisseur fournissant deux opérations :

- Une opération ***getDinarFromEuro*** : pour convertir de l'euro au dinar.
- Une opération ***getEuroFromDinar*** : pour convertir du dinar à l'euro.

1. Ecrire les classes de ce **service web**, à savoir, Convertisseur (interface), ConvertisseurImpl, et ConvertisseurPublisher.
2. Changer depuis l'interface le nom du service web en ConvertisseurDinarEuro.
3. Changer depuis l'interface le nom de l'opération 2 en dinarVersEuro.
4. Changer depuis l'interface le nom du paramètre de l'opération 1 en montantEuro et le nom de son résultat retourné en montantDinar.

<https://github.com/GitTeaching/ConvertisseurWS/tree/master/src/org/soa/ws/tp3>

<https://github.com/GitTeaching/ConvertisseurWSClient/blob/master/src/ConvertisseurWSClient.java>

# Contrat Service Web - WSDL

## Exercice 1 – Série TP 3

Changer depuis l'interface le nom du service web en ConvertisseurDinarEuro.

Changer depuis l'interface le nom de l'opération 2 en dinarVersEuro.

Changer depuis l'interface le nom du paramètre de l'opération 1 en montantEuro et le nom de son résultat retourné en montantDinar.

```
@WebService(name="ConvertisseurDinarEuro")
```

```
public interface Convertisseur {
```

```
    @WebMethod
```

```
    @WebResult(name="montantDinar")
```

```
    public double getDinarFromEuro(@WebParam(name="montantEuro")
                                     double euro);
```

```
    @WebMethod(operationName="dinarVersEuro")
```

```
    public double getEuroFromDinar(double dinar);
```

```
}
```

# Contrat Service Web - WSDL

## Exercice 1 – Série TP 3

### Client - Changement

```
public class ConvertisseurWSClient {  
  
    public static void main(String[] args) {  
        ConvertisseurImplService service = new ConvertisseurImplService();  
  
        ConvertisseurDinarEuro convertisseur =  
            service.getConvertisseurImplPort();  
  
        System.out.println(convertisseur.dinarVersEuro(100));  
    }  
}
```

# Les Services Web

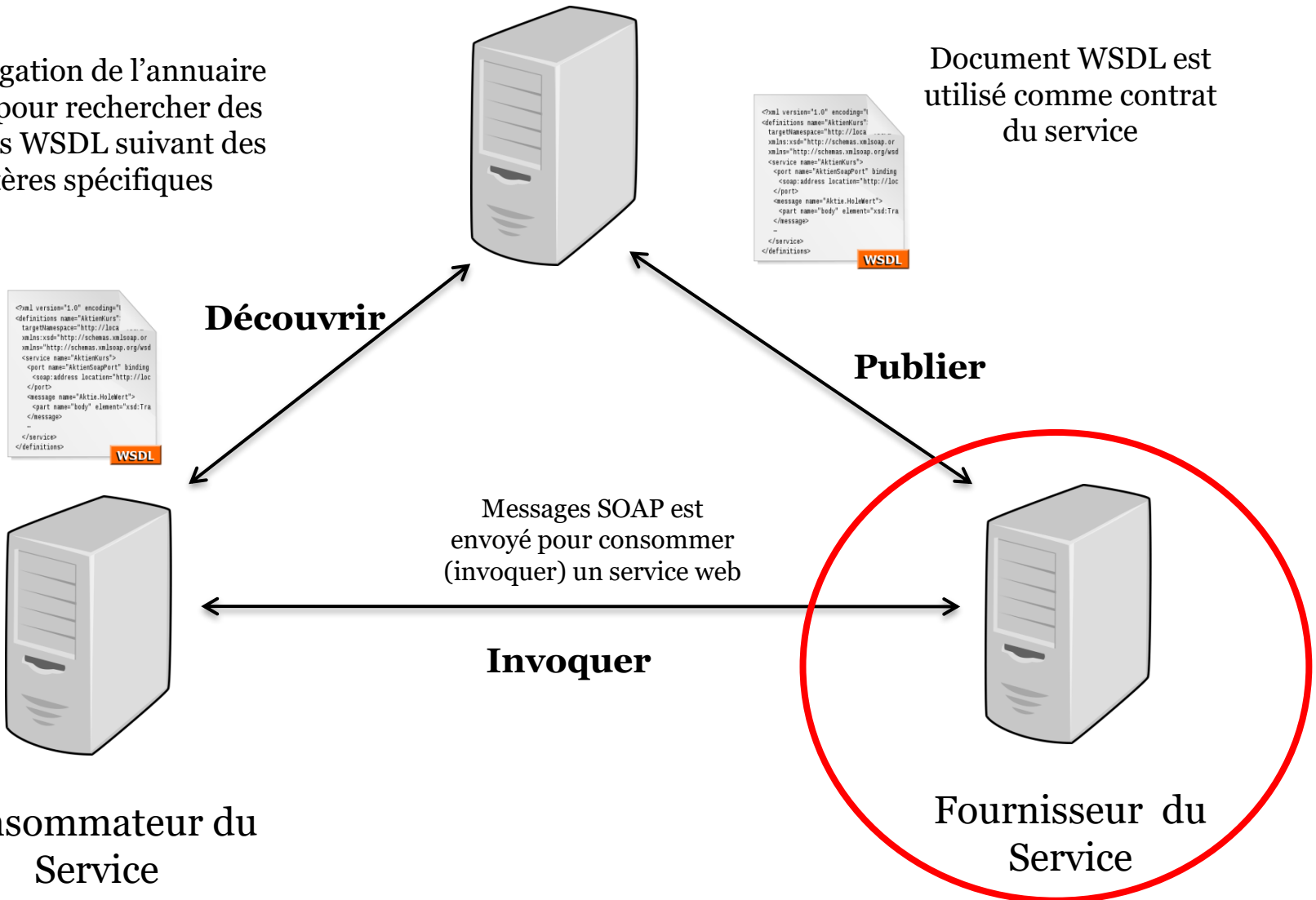
**- Implémentation : Client consommateur d'un service web**

# Style architectural - Acteurs

<http://localhost:4848/helloworldws?wsdl>

Interrogation de l'annuaire  
UDDI pour rechercher des  
contrats WSDL suivant des  
critères spécifiques

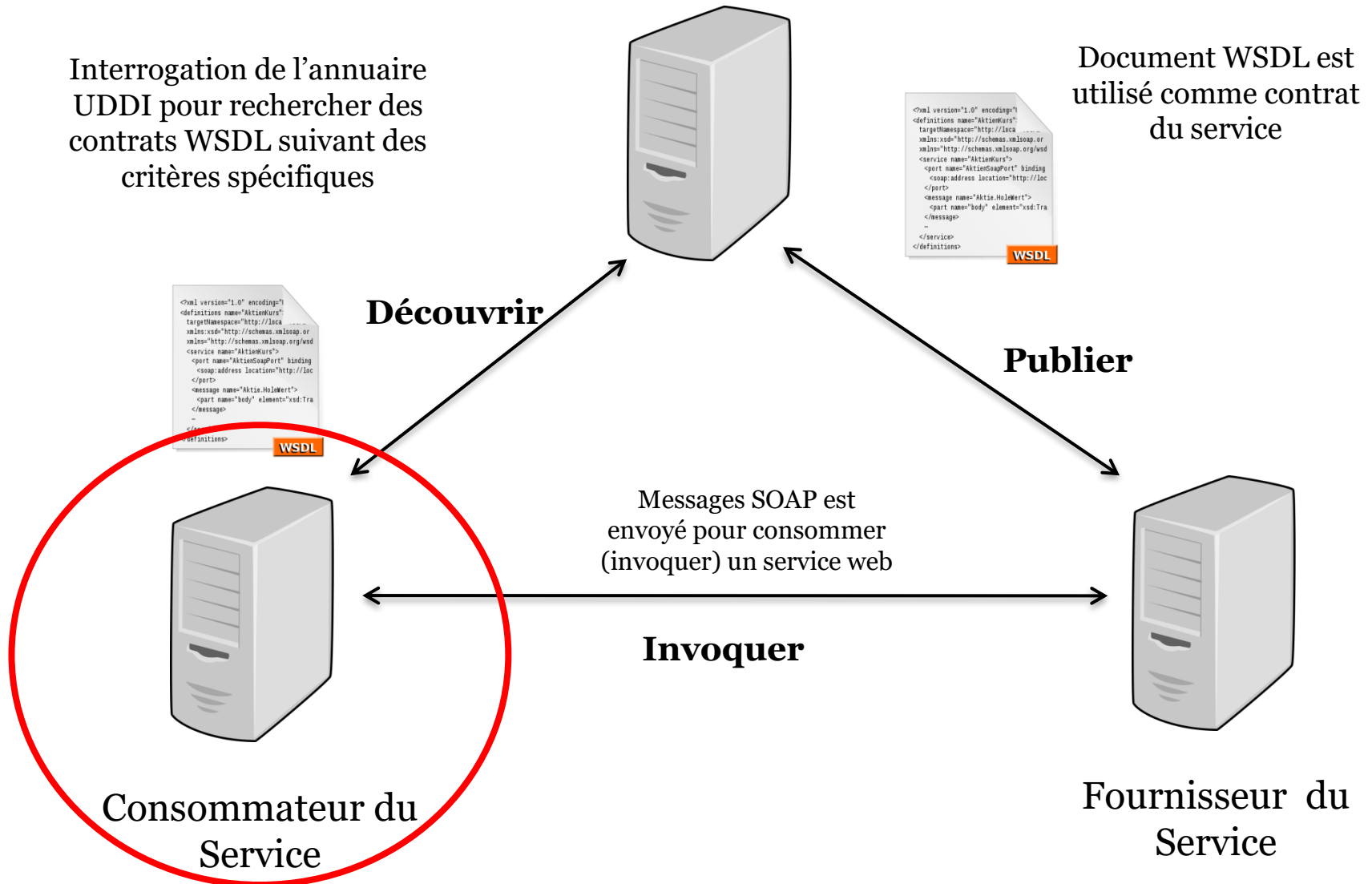
Document WSDL est  
utilisé comme contrat  
du service





# Style architectural - Acteurs

<http://localhost:4848/helloworldws?wsdl>



## Service Web SOAP - Client Consommateur

### Formule :

A extraire depuis le **WSDL** et **XSD**:

```
<ServiceName>    service = new <ServiceName>( );
```

```
<PortType>       stub = service.get<PortName>();
```

```
<ReturnType>     resultat = stub.operationName(args);
```

# Service Web SOAP - Client Consommateur

## Formule :

Exemple : Consommer **HelloWorldWS**

<**ServiceName**> : HelloWorldImplService

<**PortType**> : HelloWorld

<**PortName**> : HelloWorldImplPort

**operationName** : makeHello (String) -> String

## Service Web SOAP - Client Consommateur

**Formule** :

Exemple Consommer **HelloWorldWS** :

```
HelloWorldImplService  service = new HelloWorldImplService( );
```

## Service Web SOAP - Client Consommateur

### Formule :

Exemple Consommer **HelloWorldWS** :

```
HelloWorldImplService    service = new HelloWorldImplService( );
```

```
HelloWorld    hello = service.getHelloWorldImplPort();
```

## Service Web SOAP - Client Consommateur

### Formule :

Exemple Consommer **HelloWorldWS** :

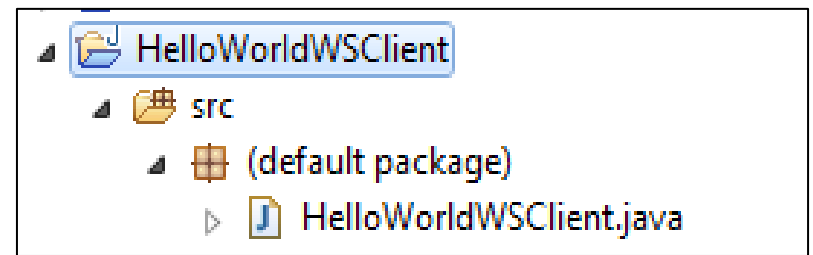
```
HelloWorldImplService    service = new HelloWorldImplService( );
```

```
HelloWorld      hello = service.getHelloWorldImplPort();
```

```
System.out.println(hello.makeHello("ISIL"));
```

# Service Web SOAP - Client Consommateur

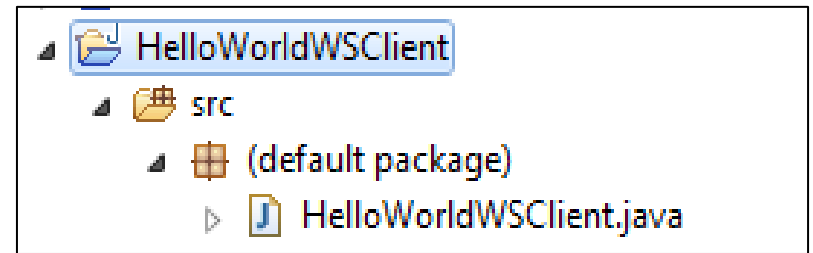
- New => Java Project = New Main Class :



```
public class HelloWorldWSSClient {  
  
    public static void main(String[] args) {  
        HelloWorldImplService service = new HelloWorldImplService();  
        HelloWorld hello = service.getHelloWorldImplPort();  
        System.out.println(hello.makeHello("JAX-WS"));  
    }  
  
}
```

# Service Web SOAP - Client Consommateur

- New => Java Project = New Main Class :

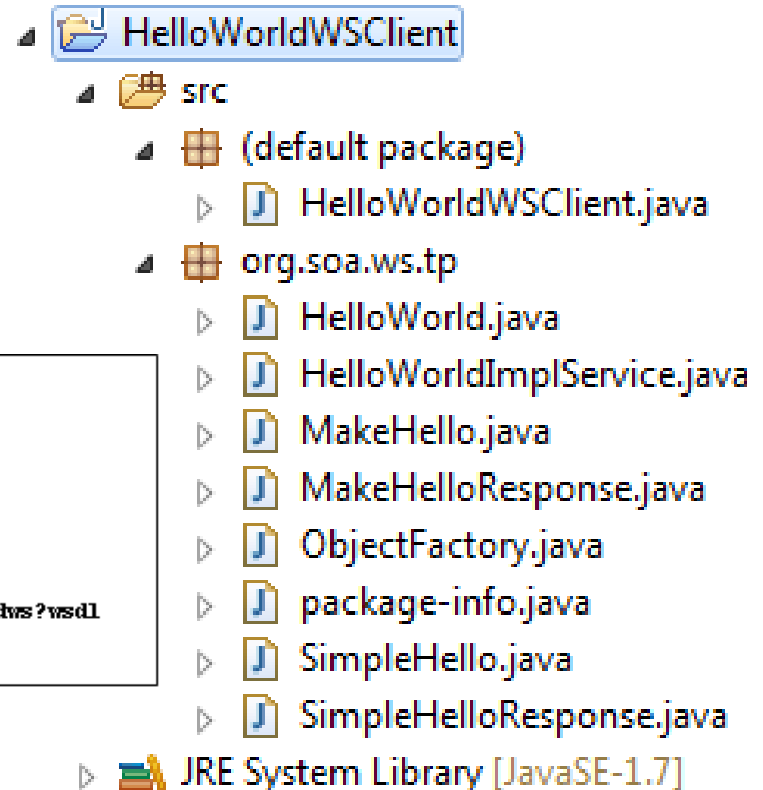


```
public class HelloWorldWSSClient {  
  
    public static void main(String[] args) {  
        ? HelloWorldImplService service = new HelloWorldImplService();  
        HelloWorld hello = service.getHelloWorldImplPort();  
        System.out.println(hello.makeHello("JAX-WS"));  
    }  
  
}
```



# Service Web SOAP - Client Consommateur

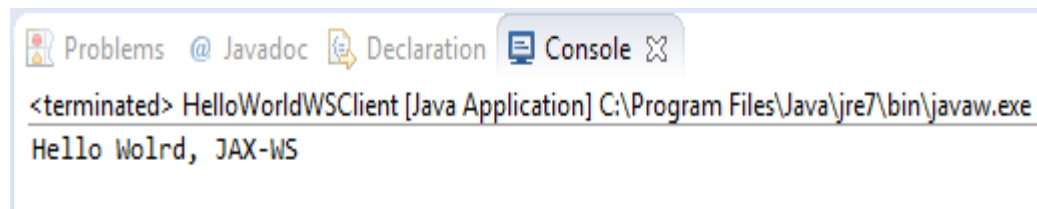
```
C:\Windows\System32> D:
D:\> mkdir SEI
D:\> cd SEI
D:\SEI> mkdir src
D:\SEI> wsimport -keep -s src http://localhost:4848/helloworldws?wsdl
```



> **wsimport** -keep -s **src** http://localhost:4848/hello?wsdl

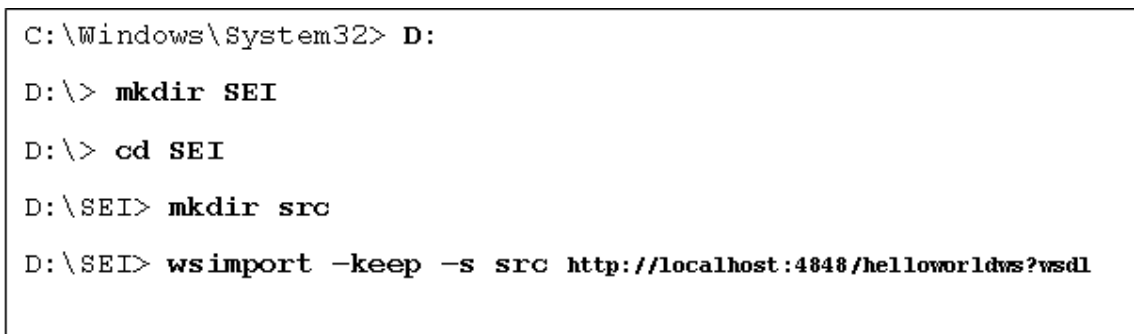
# Service Web SOAP - Client Consommateur

- Run As Java Application :



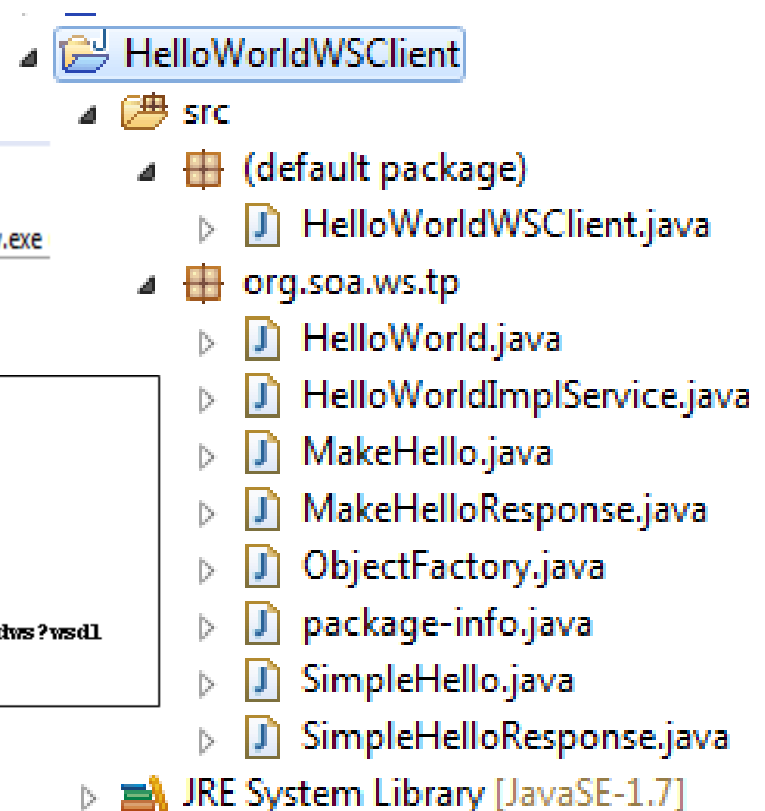
The screenshot shows the Eclipse IDE's Console window. The title bar includes 'Problems', '@ Javadoc', 'Declaration', and 'Console'. The console output shows the application has terminated and printed 'Hello Wolrd, JAX-WS'.

```
<terminated> HelloWorldWSClient [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe  
Hello Wolrd, JAX-WS
```



The screenshot shows a Windows command prompt window with the following commands and output:

```
C:\Windows\System32> D:  
D:\> mkdir SEI  
D:\> cd SEI  
D:\SEI> mkdir src  
D:\SEI> wsimport -keep -s src http://localhost:4848/helloworldws?wsdl
```



> **wsimport** -keep -s **src** http://localhost:4848/hello?wsdl

# Service Web SOAP

## Exercice : Série TP 2

Soit le contrat **WSDL** ci-dessous décrivant un service web SOAP offrant deux opérations :

- *getRandomValue* : retourne une valeur aléatoire entre 0 et 1.
- *getSinusValue* : retourne le sinus d'une valeur passée en paramètres.

Ecrire une **classe Client** qui permet d'obtenir 5 valeurs aléatoires, les stocker dans un tableau, puis n'affiche que la somme des sinus des valeurs aléatoires supérieures ou égales à 0.5.

# Service Web SOAP - Client Consommateur

## Formule :

A extraire depuis le **WSDL** – Exemple Consommer **Random WS**

```
<ServiceName>    service = new <ServiceName>( );
```

```
<PortType>       stub = service.get<PortName>();
```

```
<ReturnType>     resultat = stub.operationName(args);
```

## Service Web SOAP - Client Consommateur

```
RandomImplService service = new RandomImplService( );
```

```
Random stub= service.getRandomImplPort();
```

```
double val = stub.getRandomValue();
```

```
double sinus_val = stub.getSinusValue(val);
```

## Service Web SOAP - Client Consommateur

```
RandomImplService service = new RandomImplService( );
```

```
Random stub= service.getRandomImplPort();
```

```
double[] tab = new double[5];
```

```
for (int i=0; i<5; i++) {  
    tab[i] = stub.getRandomValue();  
}
```

```
double sinus_val = stub.getSinusValue(val);
```

## Service Web SOAP - Client Consommateur

```
double [] tab = new double()[5];

RandomImplService service = new RandomImplService( );

Random stub= service.getRandomImplPort();

double som = 0;

for (int i=0; i<5; i++) {
    tab[i] = stub.getRandomValue();

    if (tab[i] >= 0.5)
        som = som + stub.getSinusValue(tab[i]);
}

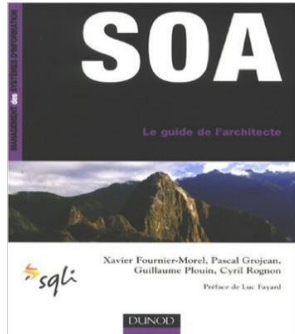
System.out.println(som);
```

## Service Web SOAP - Client Consommateur

```
public class RandomWSClient {  
    public static void main(String [] args) {  
        double [] tab = new double()[5];  
  
        RandomImplService service = new RandomImplService( );  
        Random stub= service.getRandomImplPort();  
        double som = 0;  
        for (int i=0; i<5; i++) {  
            tab[i] = stub.getRandomValue();  
  
            if (tab[i] >= 0,5)  
                som = som + stub.getSinusValue(tab[i]);  
        }  
        System.out.println(som);  
    }  
}
```



# Ressources



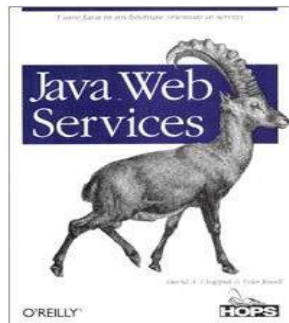
## **Le guide de l'architecte du SI**

- ✓ Auteur : Xavier Fournier-Morel, Pascal Grosjean, ...
- ✓ Éditeur : Dunod
- ✓ Edition : Octobre 2006 - 302 pages - ISBN : 2100499726



## **SOA Principles of Service Design**

- ✓ Auteur : Thomas Erl
- ✓ Éditeur : Prentice Hall Ptr
- ✓ Edition : Juillet 2007 - 608 pages - ISBN : 0132344823



## **Java Web Services**

- ✓ Auteur : David Chappell & Tyler Jewell
- ✓ Éditeur : O'Reilly
- ✓ Edition : Mars 2002 - 276 pages - ISBN : 0-596-00269-6

# Ressources

## **Engineering Long-Lasting Software: An Agile Approach Using SaaS and Cloud Computing**

- ✓ Auteur : Armando Fox and David Patterson
- ✓ Éditeur : Strawberry Canyon LLC
- ✓ Edition : Aout 2012 - 412 pages - ISBN : 0984881212

Cours – Mickael Baron – SOA et Microservices

- ✓ [http://mbaron.developpez.com/#page\\_soa](http://mbaron.developpez.com/#page_soa)

Cours – Koushik Kothagal - Developing SOAP Web Services with JAX-WS

- ✓ [https://javabrain.io/courses/javaee\\_jaxws/lessons/Introduction-to-Web-Services](https://javabrain.io/courses/javaee_jaxws/lessons/Introduction-to-Web-Services)