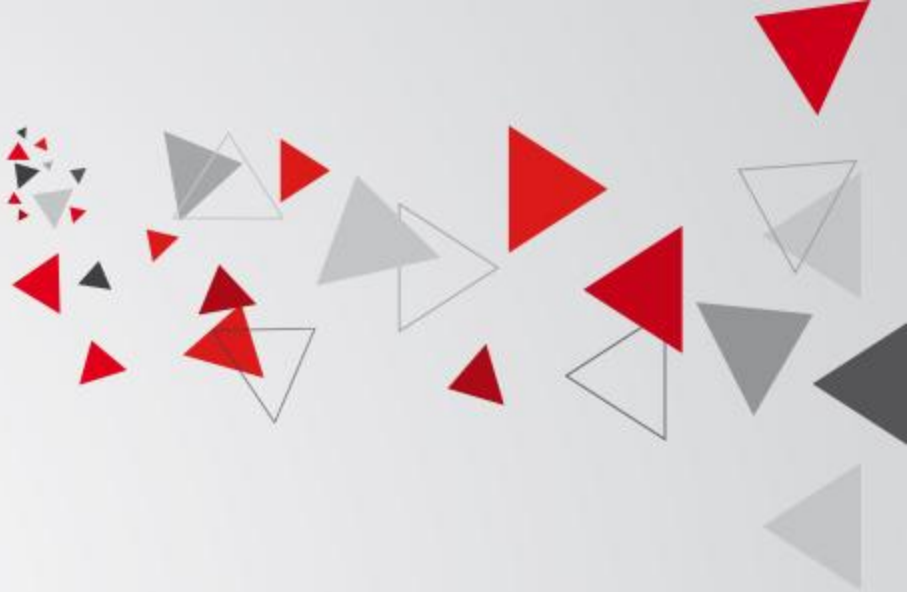
esprit
Se former autrement



JAX-WS

Module SOA

A.U 2019-2020

cdio
CONCEIVE DESIGN IMPLEMENT OPERATE

Enropean Accreditation of Engineering Programmes EUR-ACE®

Délivré par Cti Commission des Titres d'Ingénieur

CONFÉRENCE DES GRANDES ÉCOLES

Objectifs

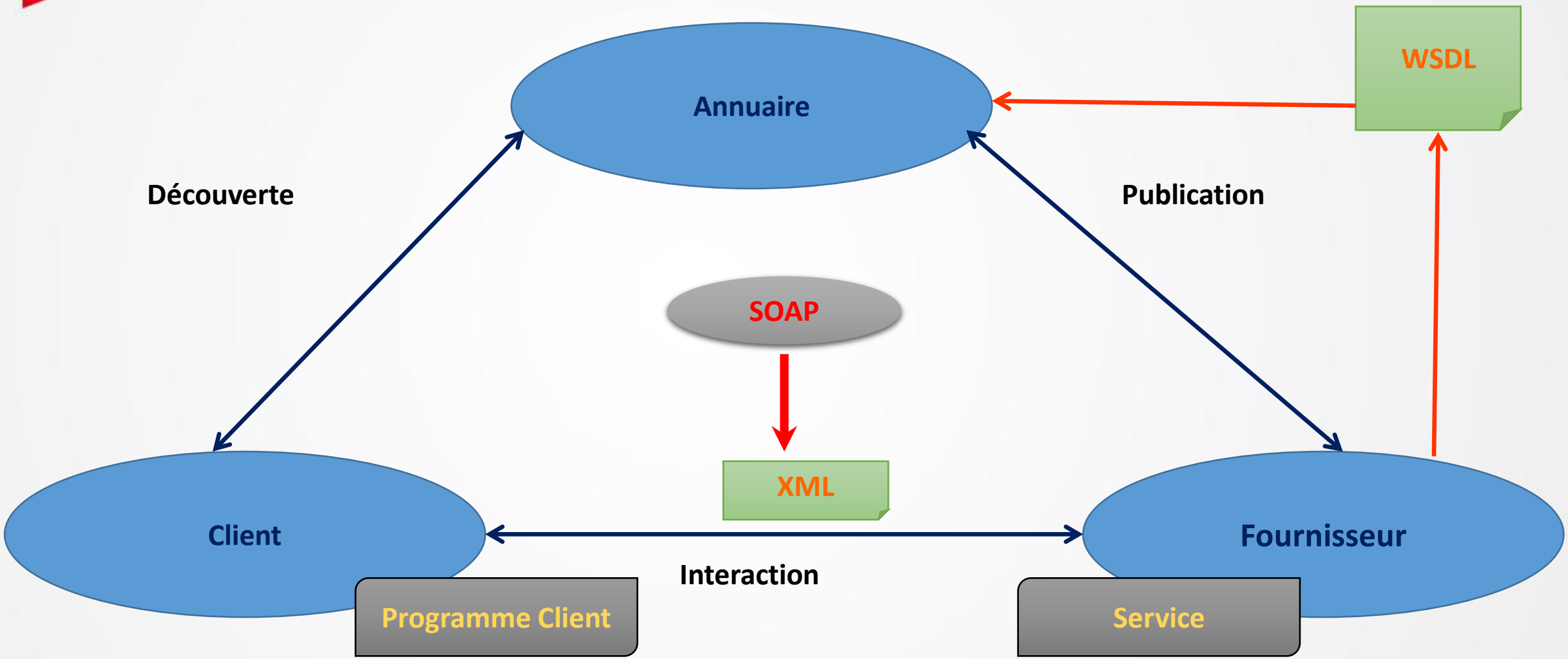
- Présentation générale des Web Services
- Apprendre l'architecture de JAX-WS
- Différences entre SOAP et REST
- Développer un service Web avec JAX-WS
- Développer un Client Web

Présentation des Services web étendus

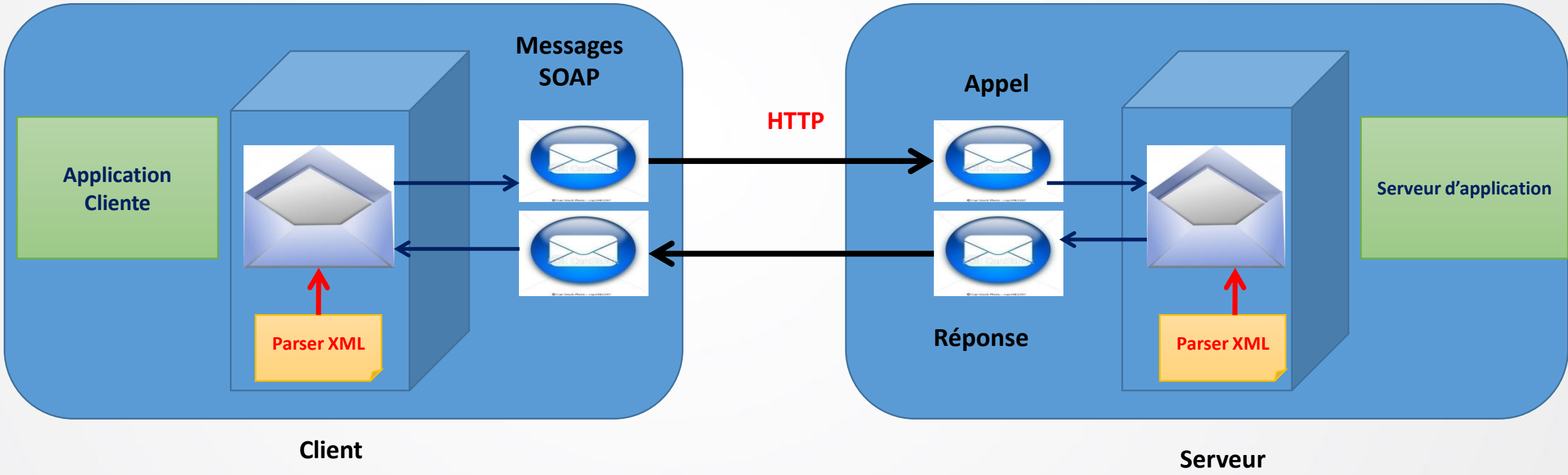
Un service web est:

- Un composant logiciel encapsulant des fonctionnalités métier accessible via des protocoles standards basés sur XML.
- Expose un contrat décrivant les modalités d'utilisation
- Abstrait
  Un service web est une **boîte noire** dont l'implémentation interne est masquée au consommateur
- Sans état
  Ne nécessite pas la présence d 'un contexte d 'exécution
- Réutilisable

Architecture des Services Web



Architecture des Services Web



Les technologies

L'architecture des web services étendus repose essentiellement sur les technologies suivantes:

- **SOAP-S**imple **O**bject **A**cces **P**rotocol: Protocole pour la communication entre Web Services.
- **WSDL-W**eb **S**ervice **D**escription **Langage**: langage de description de l'interface du Web Service
- **UDDI-U**niversal **D**escription, **D**iscovery and **I**ntegration: Annuaire pour le référencement du Web Service.

JAX-WS

JAX-WS (**J**ava **A**PI for **X**ML-based **W**eb **S**ervices) :

- Est un modèle standard de programmation des services web étendus en Java.
- Permet de développer des services web et leurs clients en Java.
- JAX-WS est une **spécification** supportée par plusieurs plateformes Java comme:
  - **AXIS2**
  - **CXF**
  - **Glassfish**

Généralités JAX-WS

- l'implémentation JAX-WS est intégrée nativement à la JRE depuis la version 6.

- Il est possible de développer des Services Web en dehors d'un serveur d'application en mode autonome.

- Le développement de Services Web avec JAX-WS est basé sur les POJO (Plain Old Java Object).

- Les fonctionnalités de base pour le développement de Web Services avec JAX-WS sont tout simplement l'utilisation des annotations Java.

Développement Serveur

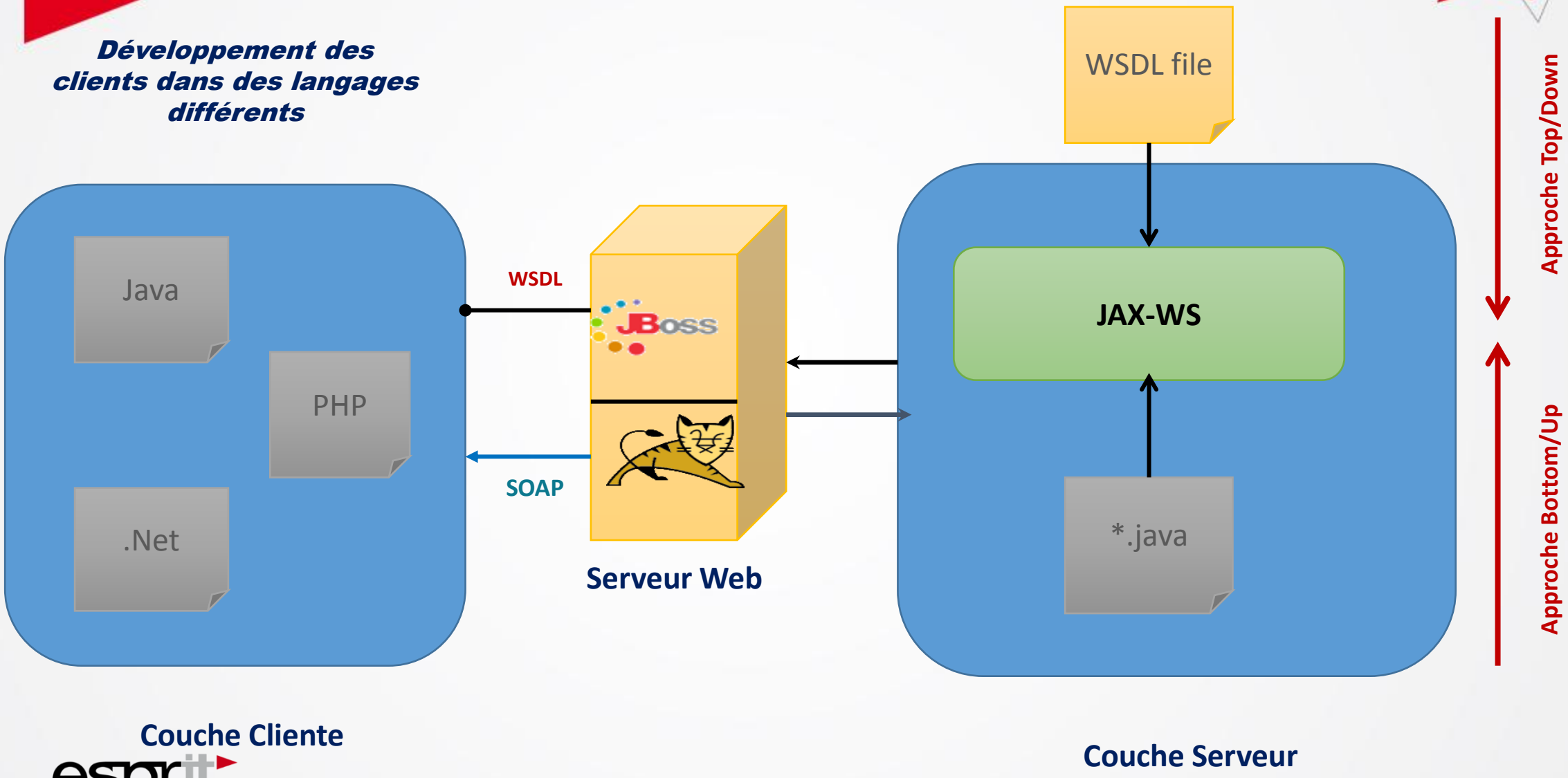
- Deux façons pour développer un Service Web avec JAX-WS:

  ❖**Approche Bottom / Up** (à partir d'un POJO)
  - ✓Créer et annoter un POJO.
  - ✓Compiler, déployer et tester.
  - ✓Le document WSDL est automatiquement généré.

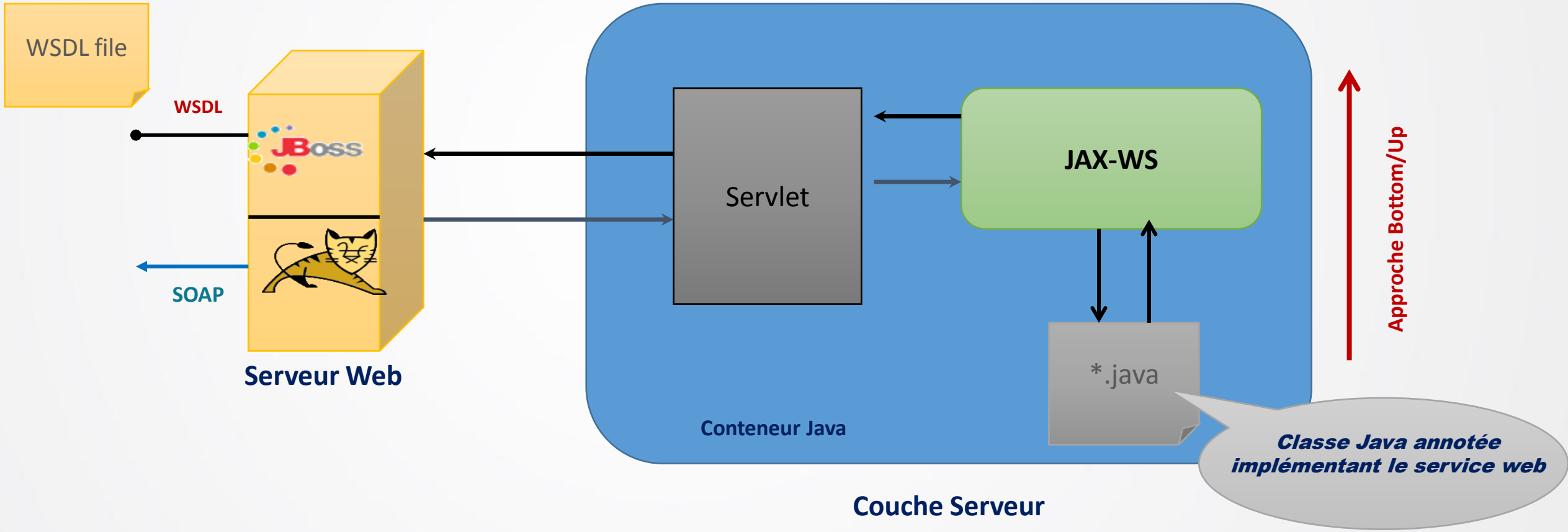
  ❖**Approche Top / Down** (à partir d'un document WSDL)
  - ✓Génération des différentes classes Java (JAXB et squelette du Web Service) en utilisant l'outil « **wsimport** ».
  - ✓Compléter le squelette de classe de l'implémentation.
  - ✓Compiler, déployer et tester.

Développement Serveur

Développement des clients dans des langages différents



Développement Serveur: Bottom/Up



Développement Serveur: Bottom/Up

- L'approche **Bottom /Up** consiste à démarrer le développement à partir d'une classe Java (POJO).
- Ajouter l'annotation @**WebService**.
- Déployer l'application sur un serveur d'application (ou via directement Java SE 6).
- Le document WSDL est généré automatiquement en respectant les valeurs par défauts.
  - *URL du WSDL : http://monserveur/app/Service?WSDL*
- Toutes les méthodes public du POJO sont des opérations du Web Service.
- La surcharge de méthodes n'est pas supportée.

Développement Serveur: Bottom/Up

- ***Exemple: Implémentation du Web Service HelloWorld:***

```
package edu.esprit.demo.HelloWorldWebService;

import javax.jws.WebService;

@WebService
public class HelloWorldService {

    public String helloWorldParam(String param){
        return "Hello World to: "+param;
    }

    public String helloWorldSimple(){
        return "Hello World to everybody";
    }

}
```

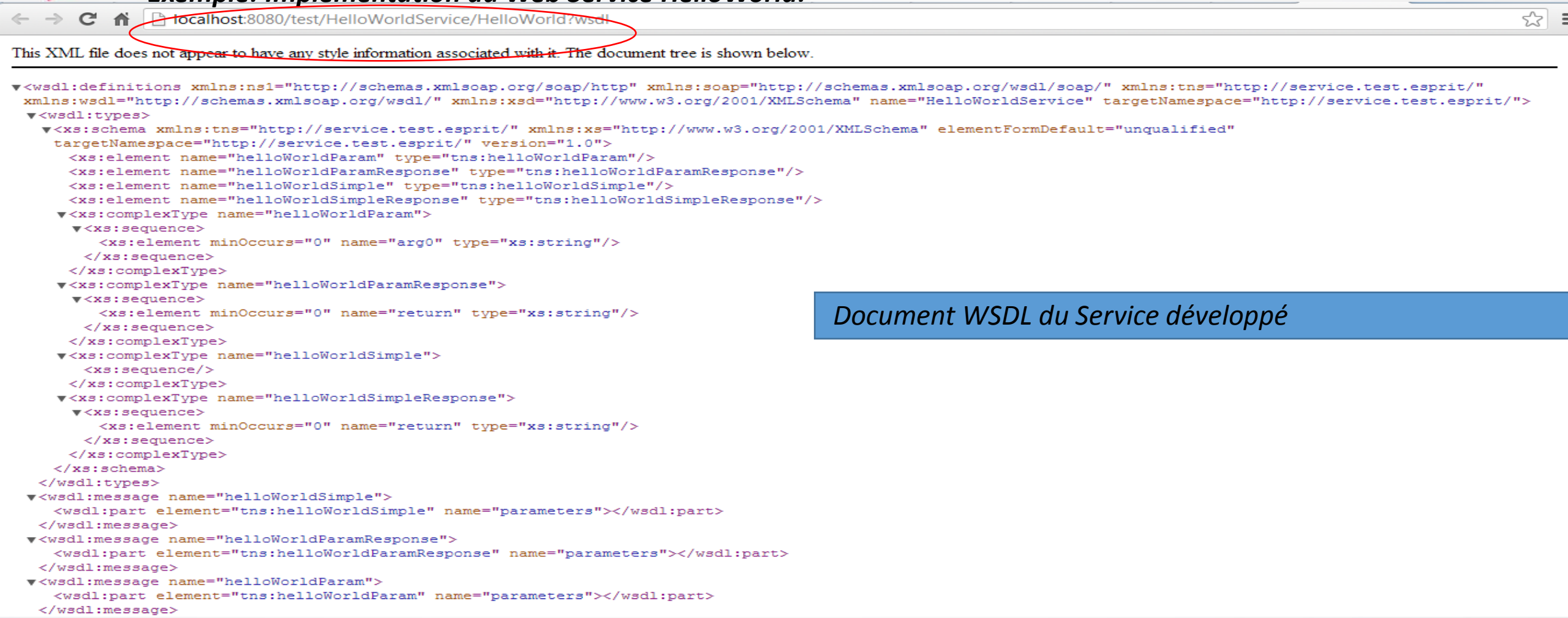
package intégré dans la JRE6

Deux opérations sont définies dans la classe HelloWorldService:

- *L'opération* ***helloWorldParam*** *contenant un message input et un message output.*
- *L'opération* ***helloWorldSimple*** *contenant un message output uniquement*

Développement Serveur: Bottom/Up

- ***Exemple: Implémentation du Web Service HelloWorld:***



Document WSDL du Service développé

Développement Serveur: Bottom/Up

- ***Exemple: Paramétrer le service web HelloWorld:***

Utilisation d'une interface pour définir les paramètres du Servi

```
package edu.esprit.demo.helloWorldWebService;

import javax.jws.WebMethod;

@Remote
@WebService (name="HelloWorld", targetNamespace="http://helloWorldWebService.demo.esprit.edu/")
public interface HelloWorldRemote {

    @WebMethod(operationName="helloWithParam")
    @WebResult(name="helloWorldResult")
      public String helloWorldParam(@WebParam (name="to") String param);

    @WebMethod(operationName="helloWorldSimple")
    @WebResult(name="helloWorldResult")
      public String helloWorldSimple();
}
```

Développement Serveur: Bottom/Up

- Comme indiqué précédemment, la JRE 6 fournit des API et des outils pour manipuler des Services Web.

- L'outil **wsgen** génère des artifacts (JAXB, WSDL) à partir des classes Java annotées via JAX-WS.

- L'utilisation de cet outil n'est pas obligatoire puisque cette génération est implicite lors de l'exécution.

- Exemples d'utilisation:

```
wsgen –cp bin edu.esprit.demo.helloWorldWebService.HelloWorldService –keep
```

→ Génère les classes Java annotées JAXB (marshall et unmarshall)

```
wsgen –cp bin edu.esprit.demo.helloWorldWebService.HelloWorldService –keep –wsdl
```

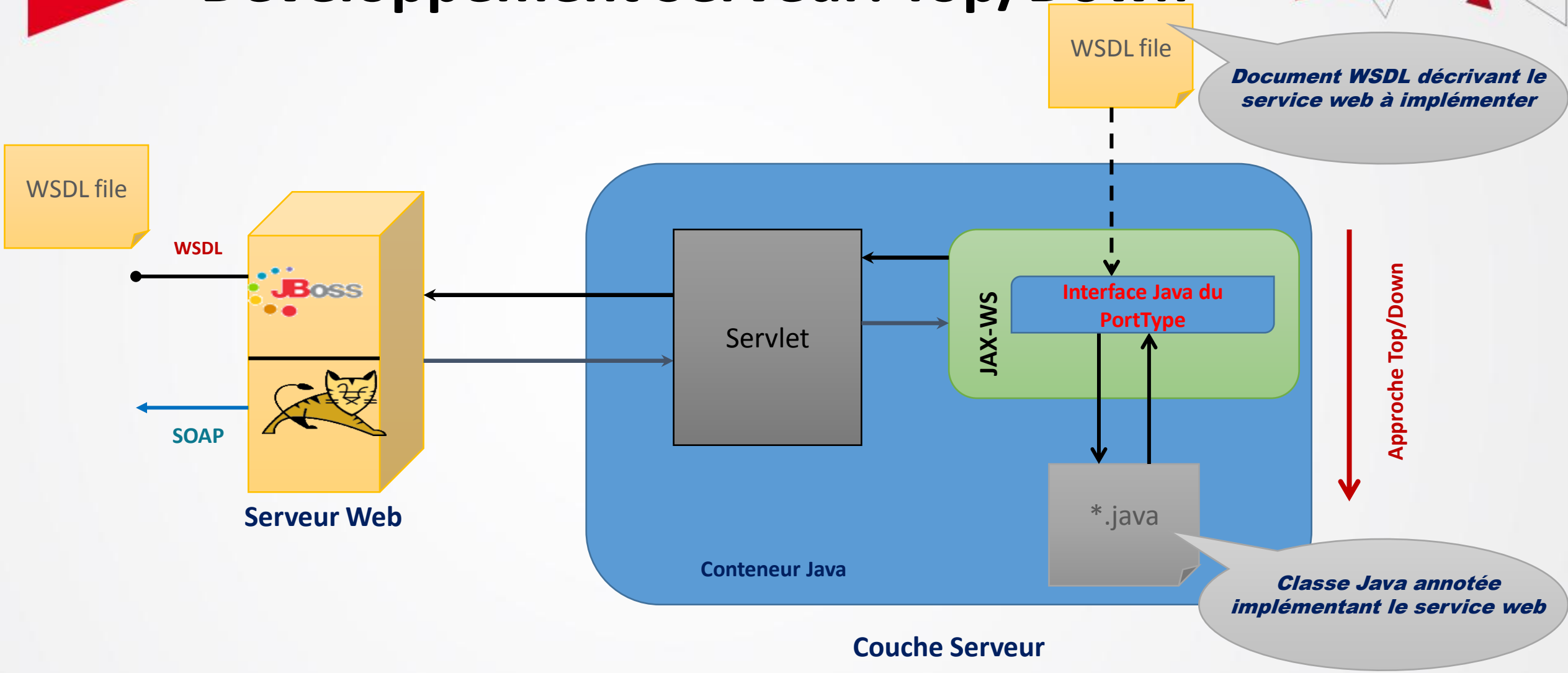
→ Génère le document WSDL

Développement Serveur: Bottom/Up

- Un Service Web est déployé dans une application Web.

- Différentes catégories de serveur d'application pour gérer les Services Web avec JAX-WS.
  - ❖Conteneur respectant JSR 109 (**Implementing Entreprise Web Services**):
    - ✓La gestion du Service Web est maintenue par le serveur d'application (*endpoint EJB, serveur JBOSS)*
  - ❖Conteneur nécessitant une gestion par Servlet:
    - ✓Nécessite une configuration explicite du Service Web (serveur Tomcat)

Développement Serveur: Top/Down



Développement Serveur: Top/Down

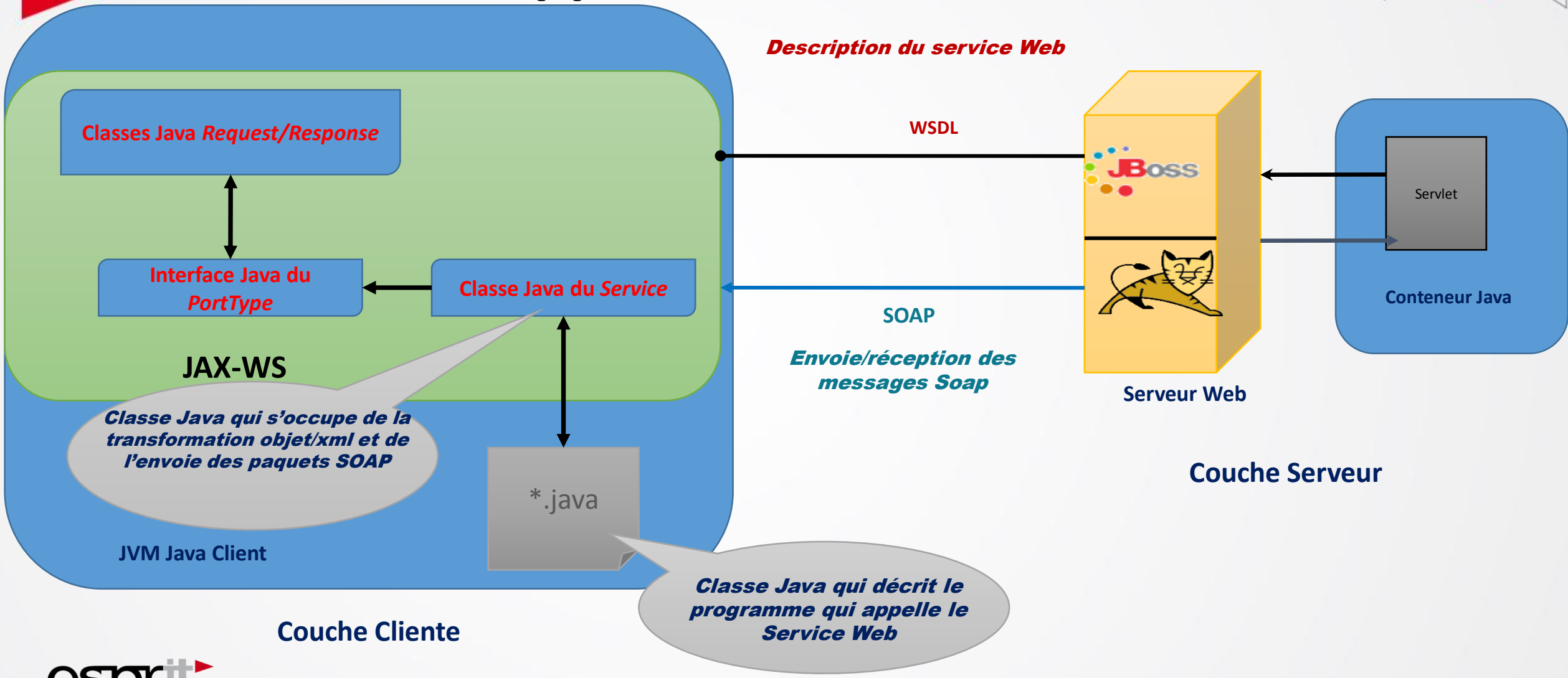
- L'approche **Top/Down** consiste à démarrer le développement du service à partir d'un document **WSDL**.

- L'outil **wsimport** nous permet de générer le squelette du Service Web:
  - ❖Génération des classes Java liées à JAXB.
  - ❖Génération des interfaces WS.

- Déployer l'application sur un serveur d'application.

- Le reste du processus de développement est identique à celui de l'approche **Bottom/Up**.

Développement Client JAX-WS



Développement Client Java

- Le développement du Client se fait à partir d'un programme Java en faisant appel aux opérations de notre Service Web.

- Le client peut être une application développée en:
  - **Java SE** (Swing, Eclipse RCP)
  - **Java EE** avec les EJB (JSP, Servlet, …)

- Possibilité de générer des appels aux Services Web de manière **synchrone** ou **asynchrone**.

- Le développeur ne manipule que du code Java, le code XML est caché (**JAXB**)

Développement Client Java

- Le développement du Client suit une procédure similaire à l'approche Top/ Down où le point de départ est le document WSDL (via une URL ou via un fichier physique).

- Utilisation explicite de l'outil **wsimport** pour la génération du squelette du Service Web.
  - ❖Génération des classes liées à **JAXB**
  - ❖Génération de classes Service Web **(PortType** et **Service**)

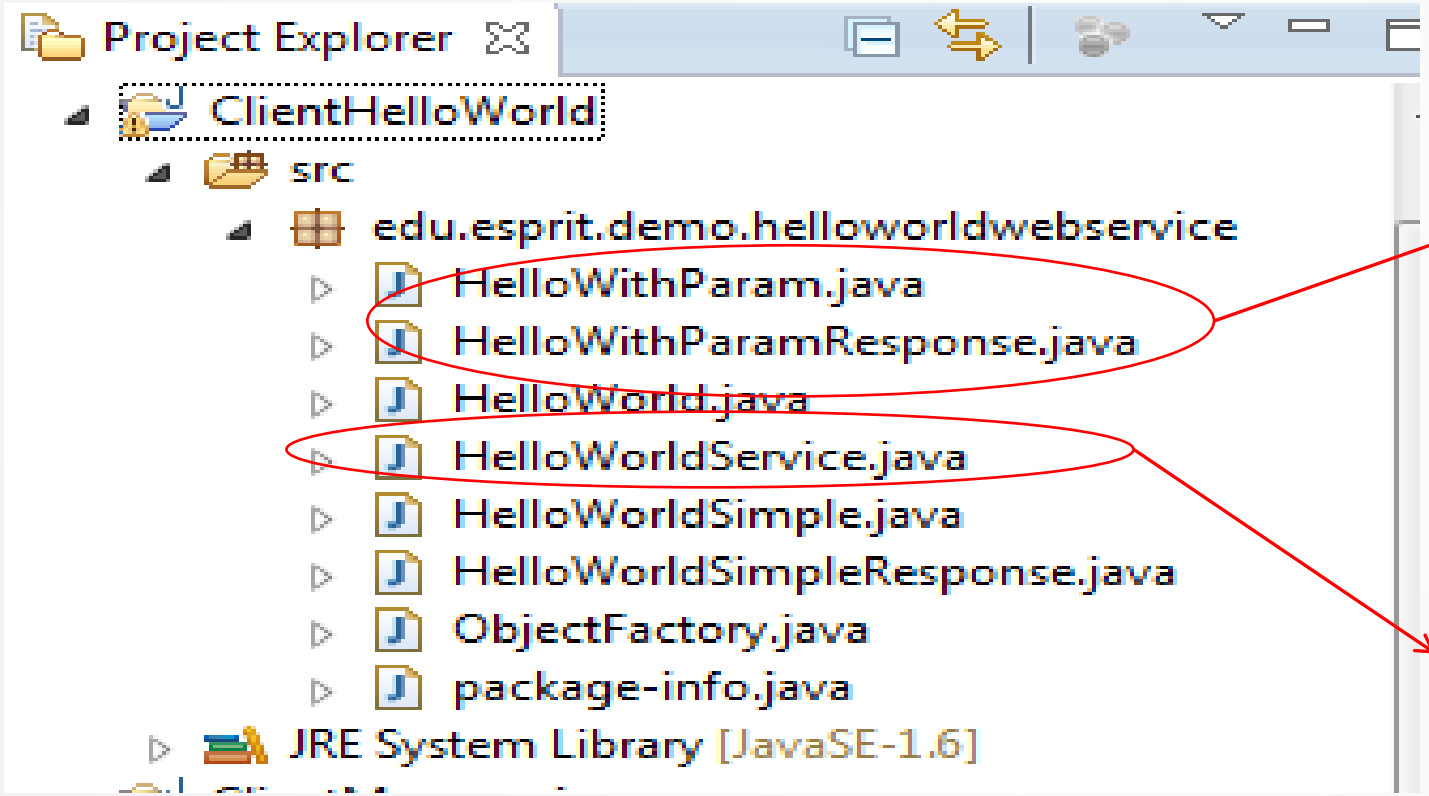
- Création d'une instance de la classe Service.

- Récupération d'un port via **get<ServiceName>Port()**.

- Invocation des opérations.

Développement Client Java

- ***Exemple: Développer un Client Java pour le Service Web HelloWorld:***



Pour chaque type définie dans le WSDL une classe Java (JAXB) est générée

La classe Service s'occupe de gérer les appels distants au Service Web

Développement Client Java

- ***Exemple: Développer un Client Java pour le Service Web HelloWorld* (suite):**

ClientHelloWorld.java

```
package edu.esprit.demo.helloWordlWebService.client;

import edu.esprit.demo.helloworldwebservice.HelloWorld;
import edu.esprit.demo.helloworldwebservice.HelloWorldService;

public class ClientHelloWorld {


    public static void main(String[] args) {

        HelloWorldService service= new HelloWorldService();
        HelloWorld helloWorldPort= service.getHelloWorldPort();

        String response= helloWorldPort.helloWithParam("ESPRIT");

        System.out.println(response);

    }

}
```

Création d'une instance de la classe Service

Récupération d'un Port via getHelloWorldPort()

Appel de l'opération du Service Web

Annotations

- JAX-WS repose sur l'utilisation massive des annotations pour la configurations d'un Service Web.

- Les principales annotations sont:
  - ***@WebService :*** POJO implémentant un Service Web.
  - ***@WebMethod :*** Paramétrer une opération.
  - ***@WebParam :*** Paramétrer un message.
  - ***@WebResult :*** Paramétrer un message de sortie.
  - ***@WebFault :*** Paramétrer un message fault

- Seule l'utilisation de l'annotation ***@WebService*** est necessaire (utilisation des valeurs par défaut).

Annotations: @WebService

- Annoter une classe Java pour définir l'implémentation du Service Web.
- Annoter une interface Java pour définir la description du Service Web.
- Attributs de l'annotation ***@WebService:***
  - ***String name :*** nom du Service Web.
  - ***String endpointInterface :*** nom de l'interface décrivant le Service Web
  - ***String portName :*** nom du port.
  - ***String serviceName :*** nom du service du Service Web.
  - ***String targetNamespace :*** le namespace du Service Web.
  - ***String wsdlLocation :*** l'emplacement décrivant le Service Web.

Annotations: @WebMethod

- Annoter une méthode d'une classe Java qui sera exposée comme opération du Service Web.

- Attributs de l'annotation : **@WebMethod**
  - ***String action :*** l'action de l'opération.
  - ***boolean exclude :*** précise que la méthode ne doit pas être exposée comme une opération. Ne pas utiliser dans une interface Java.
  - ***String operationName :*** précise le nom de l'attribut **name** définit dans l'élément operation du document WSDL.

Annotations: @WebParam

- Décrit la relation entre le paramètre d'entrée d'une méthode et le message part d'une opération.

- Attributs de l'annotation : @**WebParam**
  - ***boolean header :*** précise si le paramètre de sortie doit être transmis dans l'entête du message (**true**) ou dans le corps (**false**).
  - ***String name :*** nom du paramètre.
  - ***String partName :*** le nom du **wsdl:part** représentant ce paramètre.
  - ***String targetNameSpace :*** l'espace de nommage de ce paramètre.

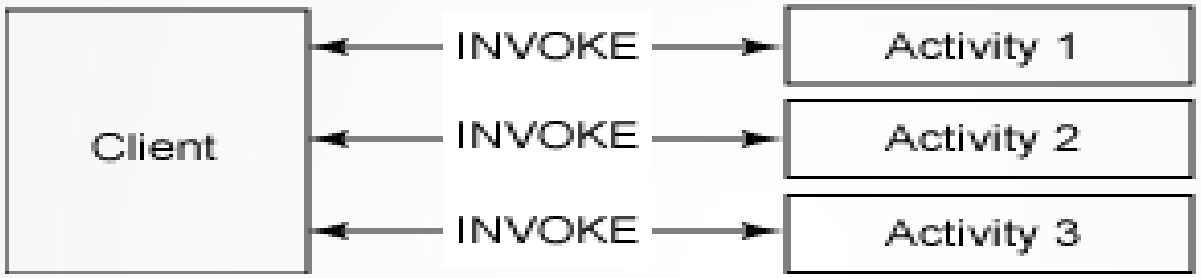
Annotations: @WebResult

- Décrit la relation entre le paramètre de sortie d'une méthode et un message part d'une opération.

- Attributs de l'annotation : @**WebResult**
  - ***boolean header :*** précise si le paramètre de sortie doit être transmis dans l'entête du message (**true**) ou dans le corps (**false**).
  - ***String name :*** nom du paramètre de sortie.
  - ***String partName :*** le nom du **wsdl:part** représentant ce paramètre de sortie.
  - ***String targetNameSpace :*** l'espace de nommage de ce paramètre de sortie.

Services web étendus VS REST 1/5

- Services web étendus sont orientés activité



Les opérations dépendent des types des activités

- Services web REST sont orientés ressource

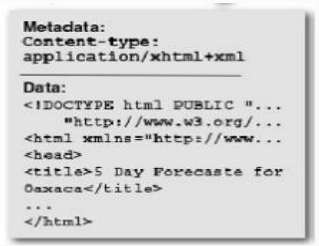
Les mêmes opérations quelque soit la resource

Services web étendus VS REST 2/5

Protocole de communication

- Services web étendus

- REST

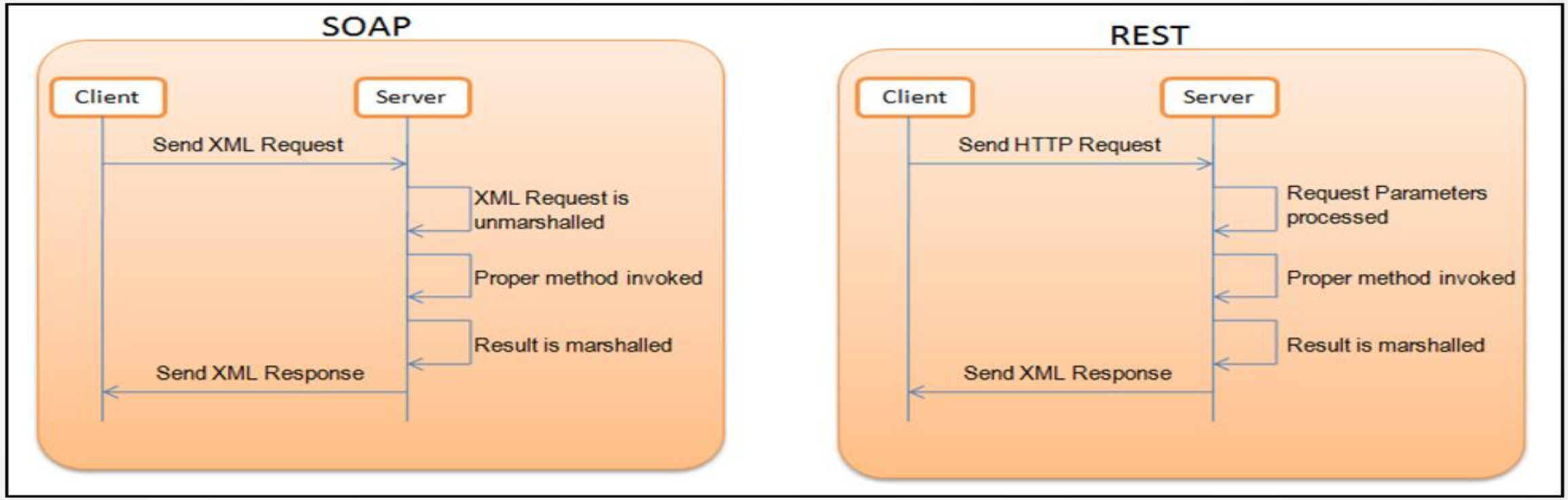


JAX-WS

Services web étendus VS REST 3/5

	Services web étendus	Services web REST
	Exposition des opérations	Exposition des ressources
Protocole de communication	SOAP	HTTP
Protocole de transport	HTTP, autres	HTTP
Description des interfaces	WSDL	WADL
Format des données	XML	XML, Text, JSON…

Services web étendus VS REST 4/5



Services web étendus VS REST 5/5

Services Web étendus

Avantages

-Standardisé
-Sécurité (WS-Security)
-Outillé

Inconvénients

-Complexité, lourdeur

Services Web REST

Avantages

-Simplicité
-Lisibilité par l'humain
-Représentations multiples

Inconvénients

-Sécurité restreinte