Université Abderrahmane Mira de Béjaia
Faculté des Sciences Exactes
Département d'Informatique

Évolution des Systèmes d'informations (Cours 1)

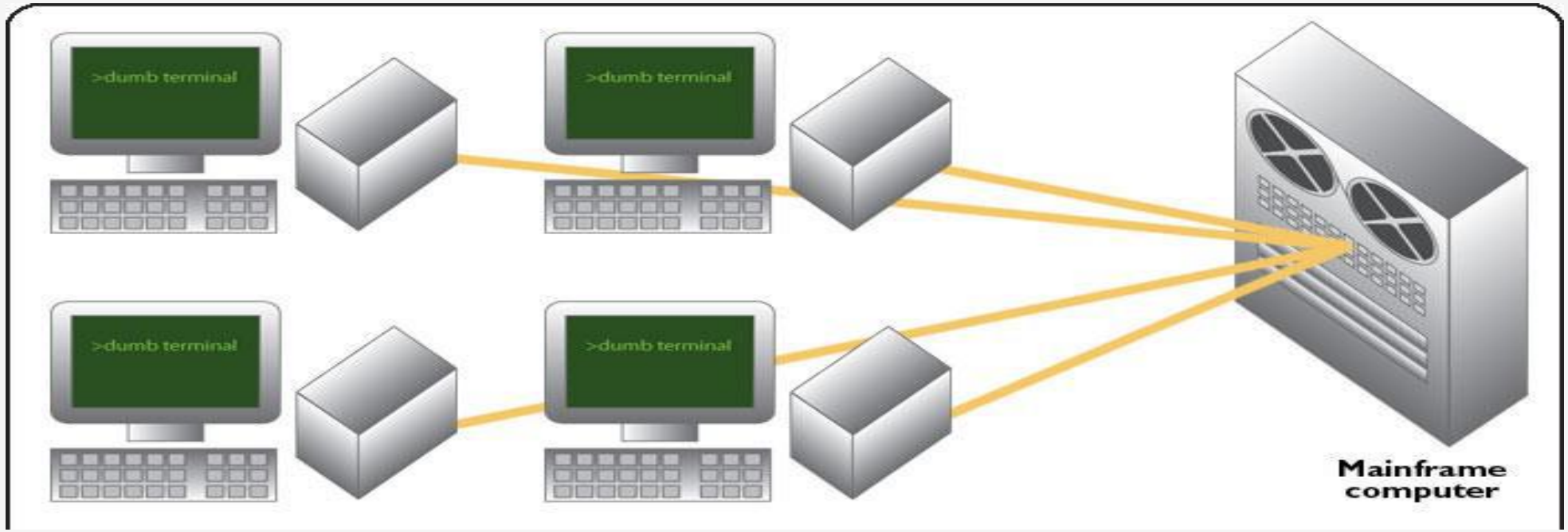
Dr H. EL BOUHISSI

Novembre 2021

Objectif du cours

Découvrir l’évolution des systèmes d’information

1ÈRE Génération : Mainframe



Ordinateur central , Terminaux , Serveur unique

Avantage : Assure la haute disponibilité et l'intégrité des données et offre à l'entreprise un système cohérent et fiable.

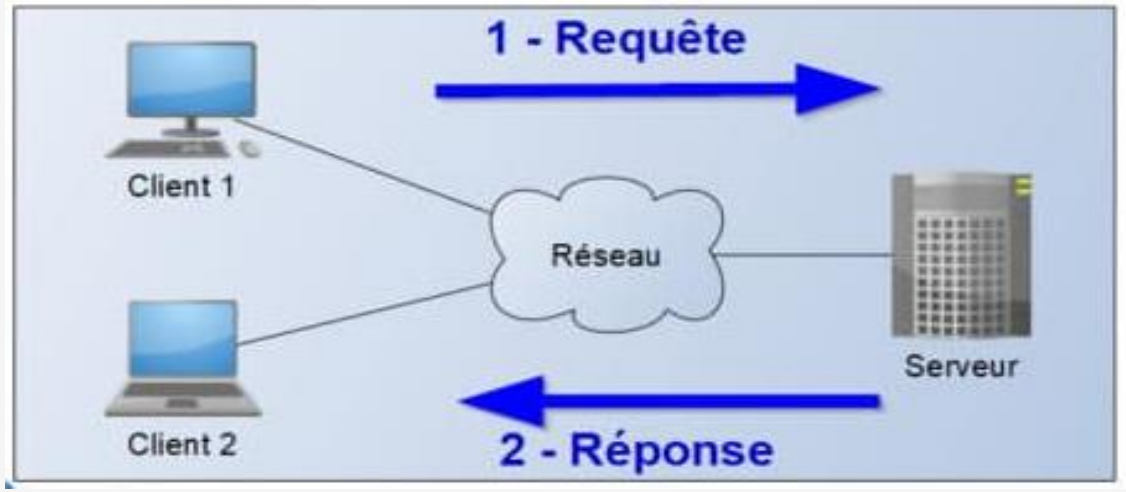
Inconvénient : Coûts d'acquisition et d'exploitation sont élevés.

2ÈME Génération : Application client/serveur

L'ordinateur client envoie une demande de données au serveur via Internet ou sur un réseau local, le serveur accepte le processus demandé et renvoie les paquets de données demandés au client.

Les clients ne partagent aucune de leurs ressources, exemple : courrier électronique, le World Wide Web, etc.

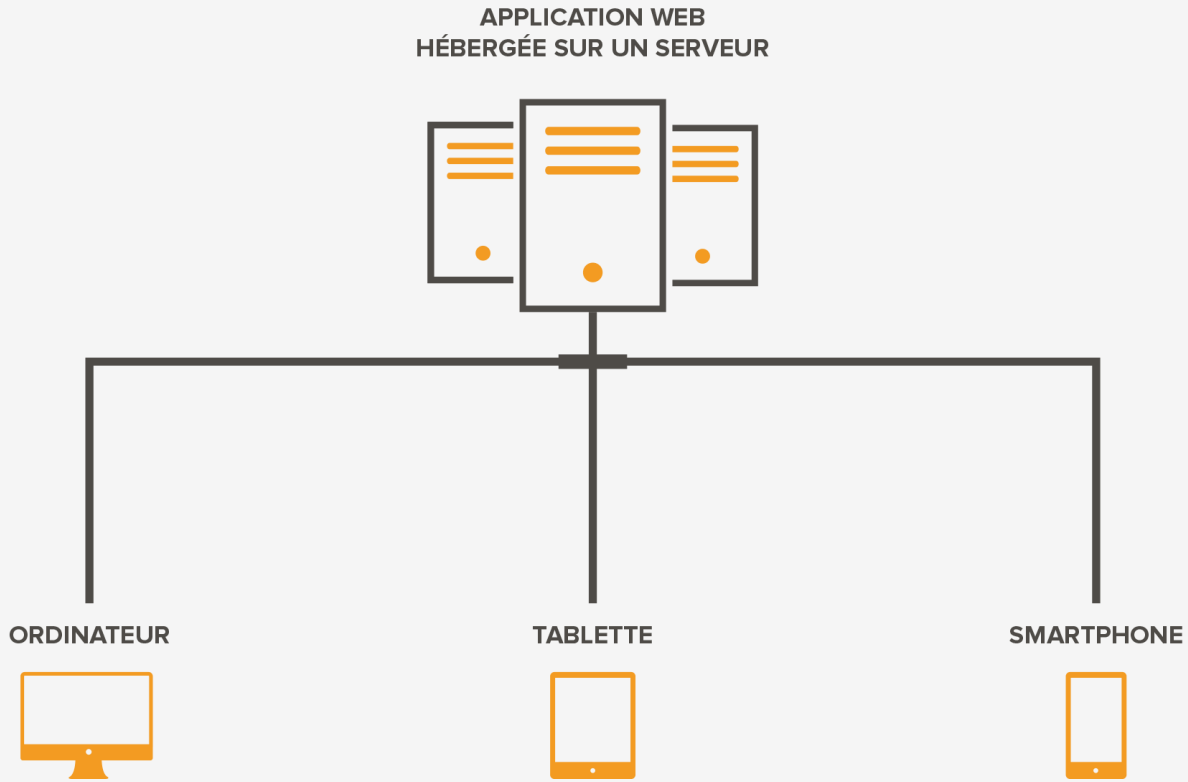
- **Système centralisé** avec toutes les données en un seul endroit. C'est d'autant plus vrai pour les serveurs d'applications.
- **Rentabilité** nécessite moins de coûts de maintenance et la récupération des données est possible.
- **Capacité** du client et des serveurs peut être modifiée séparément.



Inconvénients : Duplications d'informations , le poste de travail chargé de plusieurs exécutables , les serveurs sont sujets aux attaques par déni de service.

3ÈME Génération : Application Web

Une **application web** désigne un logiciel applicatif hébergé sur un serveur et accessible via un navigateur web. Contrairement à un logiciel traditionnel, l'utilisateur d'une application web n'a pas besoin de l'installer sur son ordinateur. Il lui suffit de se connecter à l'application à l'aide de son navigateur favori



4ÈME Génération : SOA et Services Web

SOA est apparu en 1996 dans une note de recherche du Gartner* Group.

- « L'architecture orientée service constitue un **style d'architecture** basée sur le principe de séparation de l'activité métier en une série de **services**. »
- « Ces services peuvent être **assemblés et liés entre eux** selon le principe de couplage faible pour exécuter l'application désirée. »

« Ces services sont définis à un niveau supérieur de la traditionnelle approche composants »

Gartner - Septembre 2005

Selon le Gartner Group, plus de 75% des projets d'entreprise des années 2008 reposeront sur les SOA (Service Oriented Architecture).

*Gartner, Inc., fondée en1979, est une entreprise américaine de conseil et de recherche dans le domaine de la technologie.

Université Abderrahmane Mira de Béjaia
Faculté des Sciences Exactes
Département d'Informatique

SOA et Services Web (Cours 2)

Dr H. EL BOUHISSI

Novembre 2021

Objectifs du cours

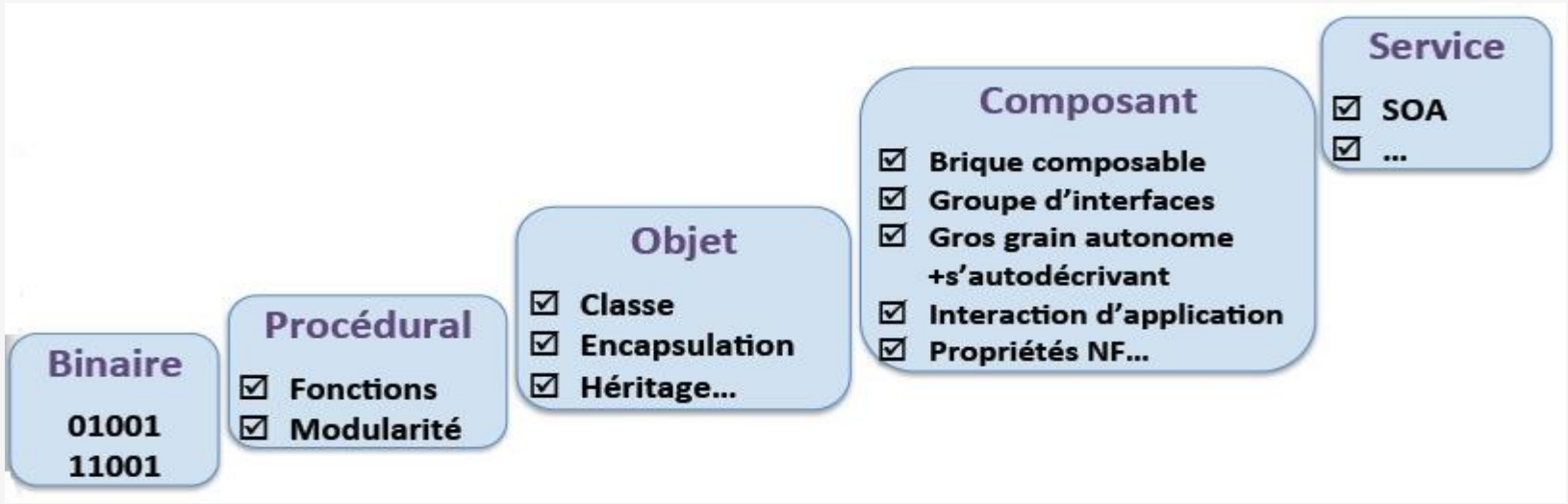
1. Comprendre le principe de l'architecture SOA
2. Comprendre l'intérêt et les enjeux de l'architecture SOA
3. Comprendre le concept de Services Web

Plan

1. Problématique
2. L’architecture SOA et le principe de service
3. Les Services Web

Evolution des paradigmes de programmation

- La conception d'un programme informatique s'effectue conformément à un paradigme de programmation (PP).
- Un PP définit un concept pour représenter le monde et des techniques pour traiter ce concept.
- Différents PP ont vu le jour et ont évolué du binaire, à différents modèles de programmation puis à **l'architecture SOA.**



Problématique

Une architecture distribuée est une architecture où le traitement des données des applications est distribué sur plusieurs machines en réseau : Architectures client-serveur, N-Tiers, Web

Le SI d'une entreprise souvent confronté à des changements :

- Interopérabilité ?
- Réutilisabilité ?
- Communication entre applications hétérogènes ? (Java/C#)

L'interopérabilité exprime le besoin d'échanger des données entre 2 systèmes distribués et **éventuellement hétérogènes** pour un besoin **d'intégration** **(par exemple les données d'une facture).**

Le Paradigme SOA

- Style d'architecture distribuée qui permet de fournir ou consommer un processus métier en tant que service.
- Offre des services réutilisables et interopérables via des interfaces standards (construites autour de XML).
- Plusieurs partenaires peuvent communiquer et échanger des données dans le contexte de SOA indépendamment des Plateformes et langages.

Le paradigme SOA :

- Augmenter la flexibilité
- Faiblement couplé
- Basé sur des standards
- Hétérogène
- Plusieurs propriétaires
- Distribué

Service : composant logiciel qui exécute une action pour le compte d'un client , il traduit le niveau logique d'accès aux traitements, plutôt que le niveau physique d'implémentation.

Acteurs de la SOA

Fournisseur de service :

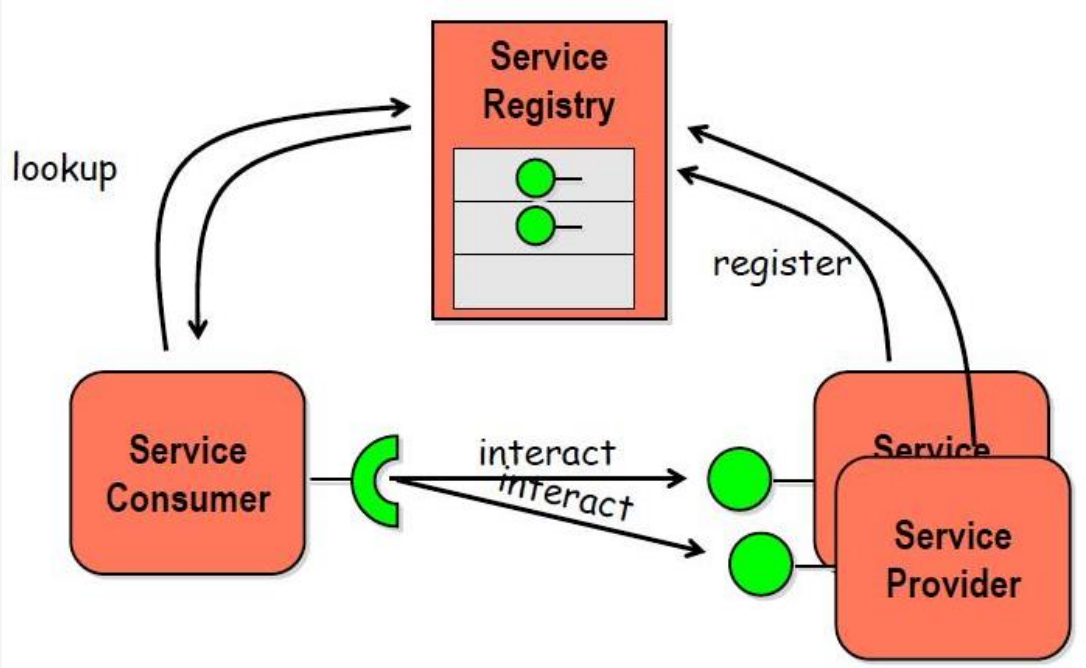
- Fournit un service accessible via une adresse
- Publie son contrat dans le registre de services
- Exécute les requêtes des consommateurs (un Proxy et un cache peuvent être utilisés du côté consommateur pour délocaliser le traitement et réduire le nombre d'appels réseau)

Consommateur de service

Application, service…, Cherche le service dans le registre (son adresse) - Se lie dynamiquement au service. Invoque le service via une requête conforme au contrat

Registre de services

Annuaire des contrats de services, Le Contrat décrit le format d'échange (format des requête/réponse, les pré et post conditions du service et sa QoS, ex: temps de réponse). Le contrat est renouvelable par demande de nouveau bail à partir du registre



Technologies d'implémentation de SOA

L'architecture SOA peut être implémentée par différentes Technologies :

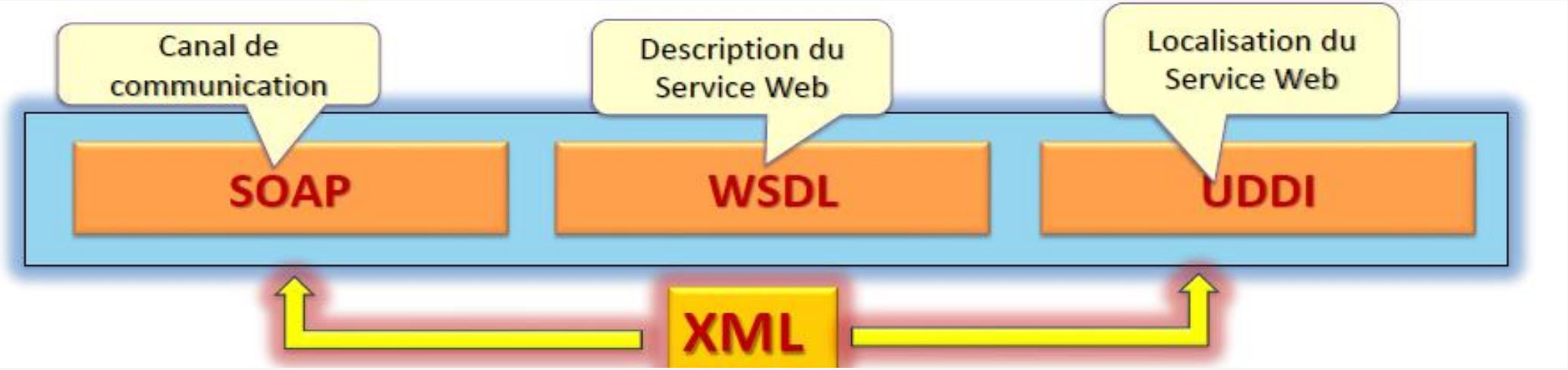
- CORBA : Multi-langage , Multiplateforme, installation coûteuse.
- DCOM (Microsoft) : Multi-langage , Mono-plateforme, faible diffusion (non disponible pour certains OS).
- RMI : Multi-langage , Multiplateforme, problème de performance et de sécurité.
- Services Web

Les Services Web demeurent la technologie émergente pour l'implémentation de l'architecture SOA :

- Déployés : sur n'importe quelle plateforme
- Modulaires : implémenté dans n'importe quel langage
- Enveloppés dans une couche de standards dérivés du XML
- Ne nécessitant pas une configuration réseau particulière
- Publié, localisé et invoqué de n'importe quel point du Web
- Accessible via des protocoles standards Internet

Services Web

- Les services Web sont une technologie d'implémentation de la SOA.
- Les services Web constituent la meilleure solution disponible (standardisée).
- Selon le W3C (2004) : Un service web est un système logiciel identifié par un URI, dont les interfaces publiques et les « bindings » sont définies et décrites en XML. Sa définition peut être découverte dynamiquement par d'autres systèmes logiciels. Ces autres systèmes peuvent ensuite interagir avec le service web d'une façon décrite par sa définition, en utilisant des messages XML transportés par des protocoles Internet.



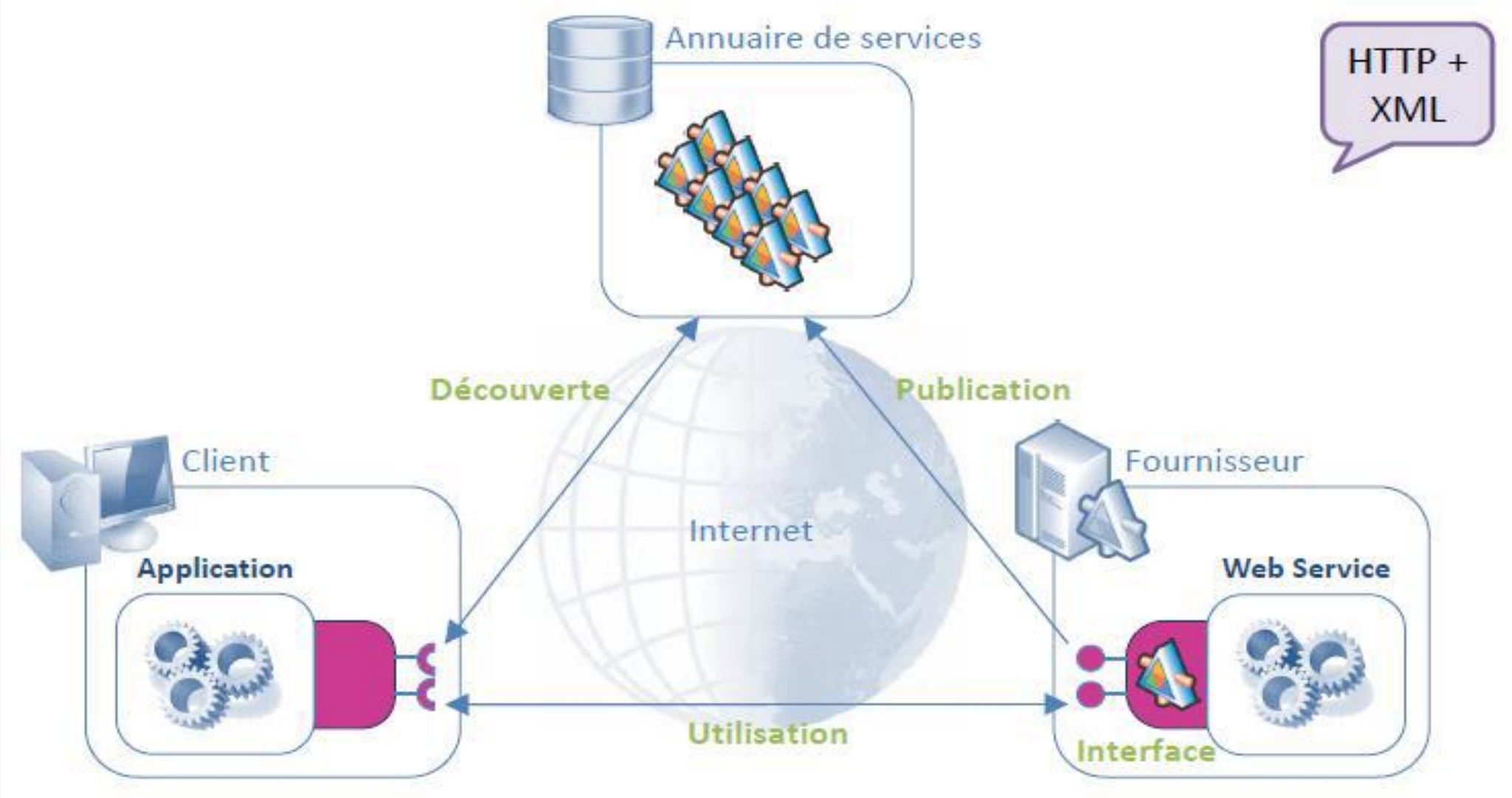
Infrastructure des Services Web

SOAP (Simple Object Access Protocol) : assure la communication avec et interServices Web.

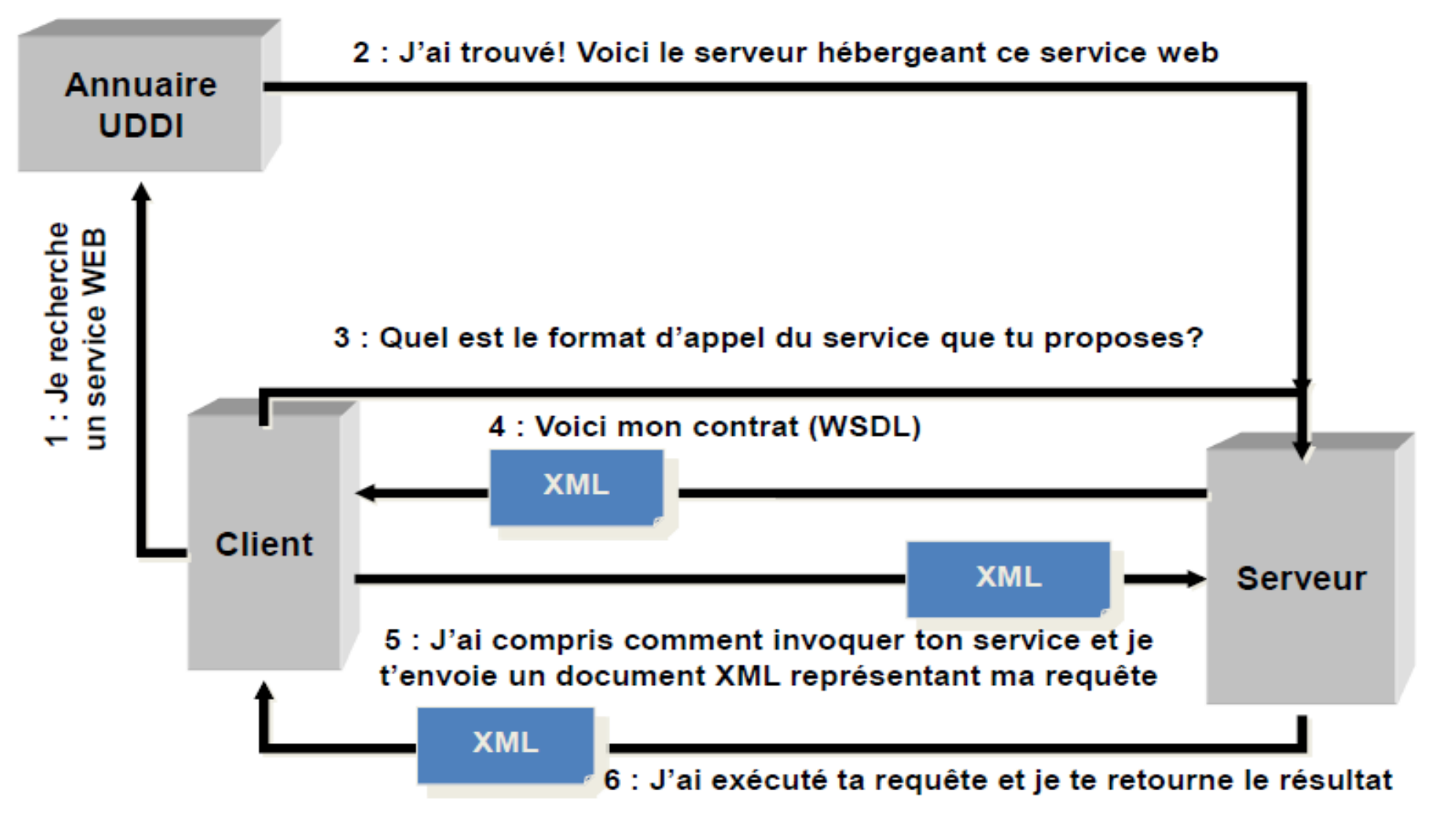
WSDL (Web Services Description Langage) : offre un schéma formel de description des Services Web.

UDDI (Universal Description, Discovery and Integration) : offre une manière uniforme de définir des registres des Services Web et un schéma uniformément extensible de descriptions des Services Web.

Infrastructure des Services Web



Cycle de vie d'utilisation d'un Service Web



Types de Services Web

- **Services Web 1.0 (Première génération = Etendus) :**
  utilisent les standards UDDI (annuaire)/ WSDL(contrat) / SOAP (consommation)

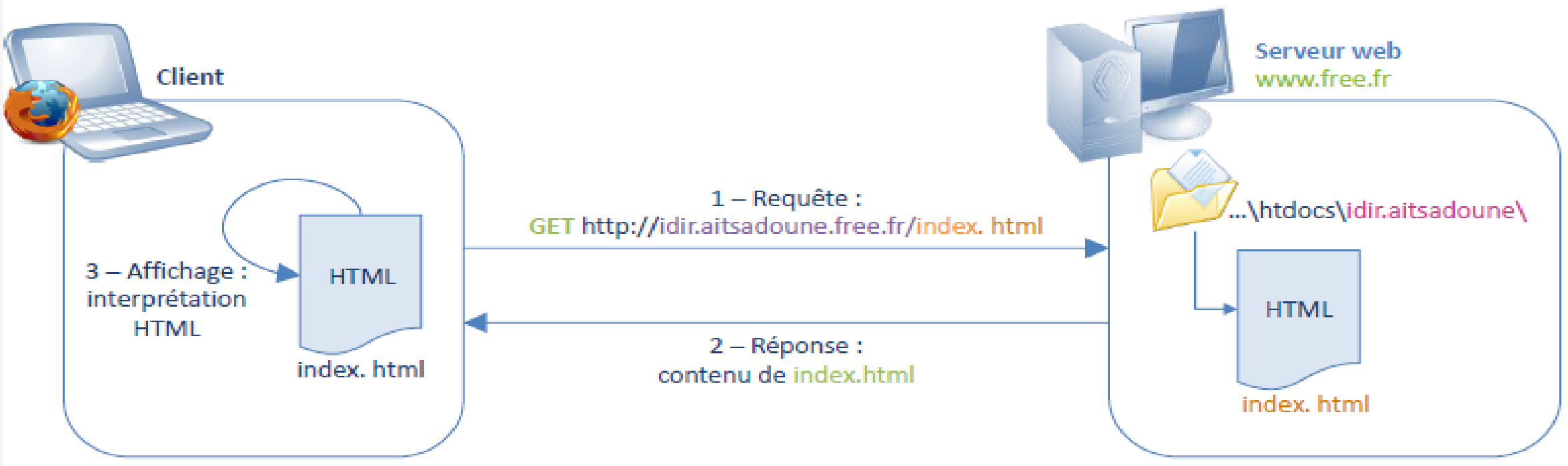
- **Services Web 2.0 (Deuxième génération = REST(Representational State Transfer)) :**
  utilisent :

  - Directement HTTP au lieu d'une enveloppe SOAP
  - Un URI pour nommer et identifier une ressource
  - Les méthodes HTTP (POST, GET, PUT et DELETE) pour effectuer les opérations de base CRUD

- REST n'est pas un remplacement générique de SOAP (ne couvre qu'une partie des besoins)
- REST utilise WADL (Web Application Description Langage) pour décrire les Contrats, et qui est non standardisé : initiative isolée de SUN

HTTP (Hyper Text Transfer Protocol)

Protocole de communication dédié au web
Chaque ressource du web est identifiée par une URL
Mode de communication = requête / réponse



Exemples de Services Web existants

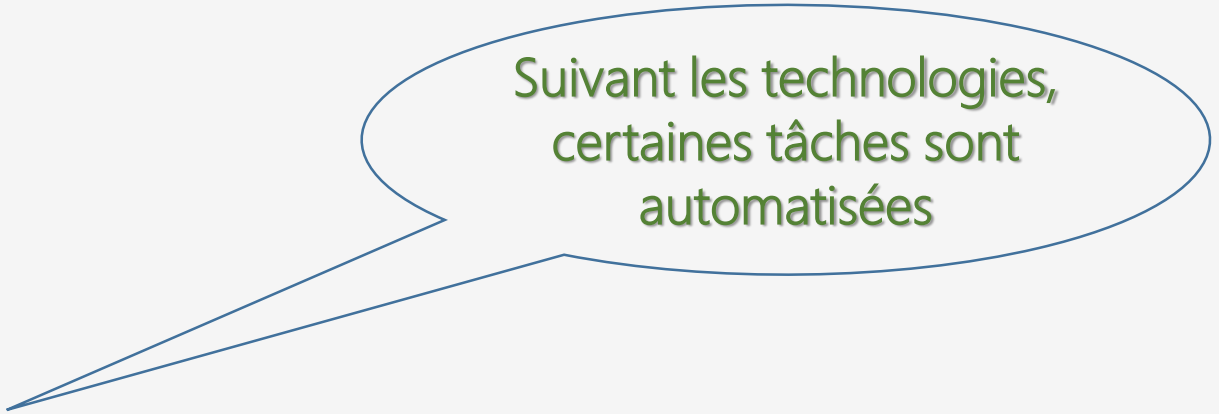
Google (http://www.google.com/apis/) :
•Accès gratuit mais limité (1000 requêtes par jour après enregistrement)

Amazon (http://aws.amazon.com/fr/) •accès gratuit mais limité (1 requête par seconde après enregistrement)

Bien d'autres (http://webservicex.net par exemple)

Implémentation des Services Web

Côté fournisseur

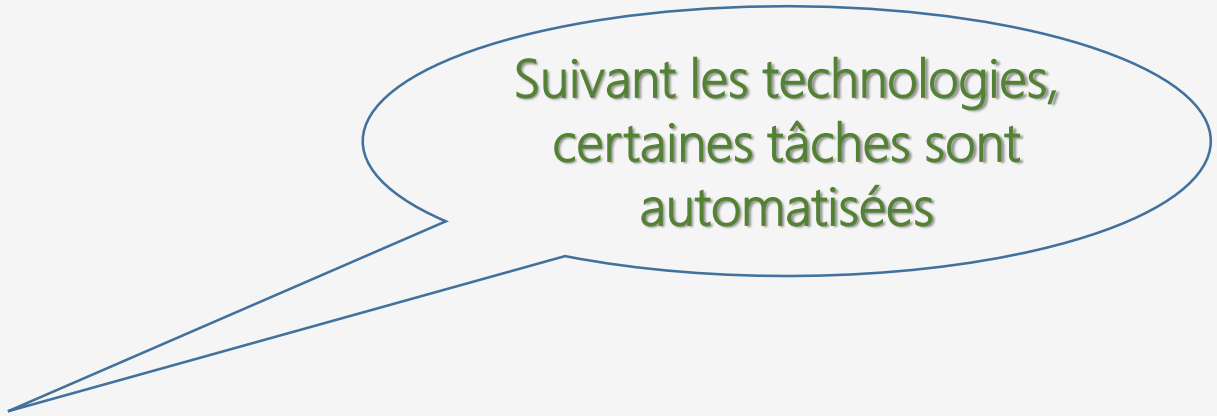


Pour créer un Web Service :

1. Définir le contrat du service
2. Développer le service
3. Développer la couche de traitement XML
4. Déployer sur le serveur
5. Publier dans l'annuaire

Implémentation des Services Web

Côté client



Pour créer une application cliente :

1. Rechercher le service dans l'annuaire
2. Récupérer le contrat du service
3. Développer la couche de traitement XML
4. Utiliser le service et présenter les résultats (rendu)

Tâches relatives aux Services Web

Invocation : vise à établir la communication entre le client et le fournisseur en décrivant la structure des messages échangés.

Découverte : permet de localiser un service web particulier dans un annuaire de services décrivant les fournisseurs ainsi les services fournis.

Composition : consiste à combiner des services pour former un nouveau service dit composé ou composite. Le but de la composition est avant tout la réutilisation de services (simples ou composés) et de préférence sans aucune modification de ces derniers.

Sélection : choisir parmi les services web découverts, ceux qui répondent au mieux aux exigences de l'utilisateur sur la base des besoins fonctionnels et/ou non fonctionnels.

Université Abderrahmane Mira de Béjaia
Faculté des Sciences Exactes
Département d'Informatique

Le WSDL
(Cours 3)

Dr H. EL BOUHISSI

Novembre 2021

Objectifs du cours

1. Découvrir le contrat de description d'un Service Web
2. Comprendre les briques de base d'un document WSDL
3. Apprendre comment extraire les informations utiles à partir d'un document WSDL

Plan

1. Définition et Versions
2. Contenu d’un document WSDL
3. Exemples de documents WSDL

WSDL : Définition

WSDL signifie Web Service Description Language.

Un métalangage permettant de décrire en détail les services Web. Un service Web est un service mis à disposition des clients par un serveur via Internet (ou un autre réseau). Les services Web sont multiplateformes, c'est-à-dire qu'ils fonctionnent avec les systèmes et les applications les plus divers.

Pour qu'un client puisse s'informer sur les possibilités et les processus du service Web, un fichier WSDL est disponible sur le serveur. Les détails contenus dans le fichier permettent au client de savoir comment consulter le service Web.

WSDL : Définition

Standard du W3C :

Version 1.1 en 2001
Version 2.0 en 2007, encore peu supporté par les outils

Objectif : décrire l'interface publique d'un Web Service (contrat de service)

Grammaire dérivée d'XML

Service Web = ensemble de ports de connexions mettant à disposition des opérations qui reçoivent et envoient des messages

Regroupe les informations nécessaires pour interagir avec le service (fonctionnelles et techniques) :
•Les méthodes, les paramètres et valeurs retournées, le protocole de transport utilisé, la localisation du service
Document indispensable au déploiement de Services Web
•Publication et recherche de services au sein de l'annuaire se font via les documents WSDL
• Pour l'accès à un service particulier, un client se voit retourné l'URL du fichier WSDL décrivant l'implémentation du service

Eléments du WSDL

<definitions> : racine du document
<types> : Types de données sous format schéma XML

<Part>

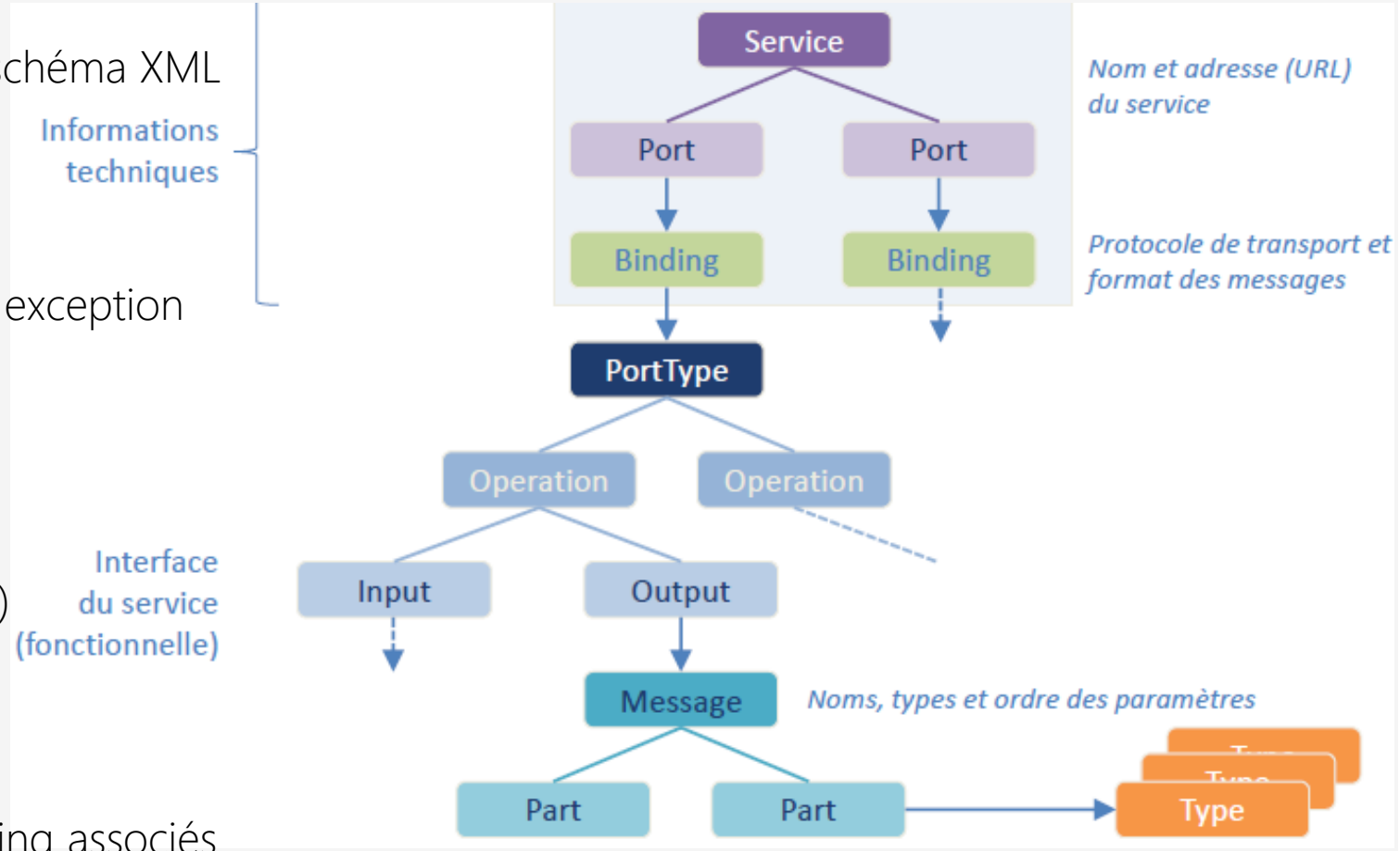
<Message> : paramètre, valeur de retour, exception
<Operation>

<portType> : ensemble d'opérations

<binding> : détail technique (protocoles…)

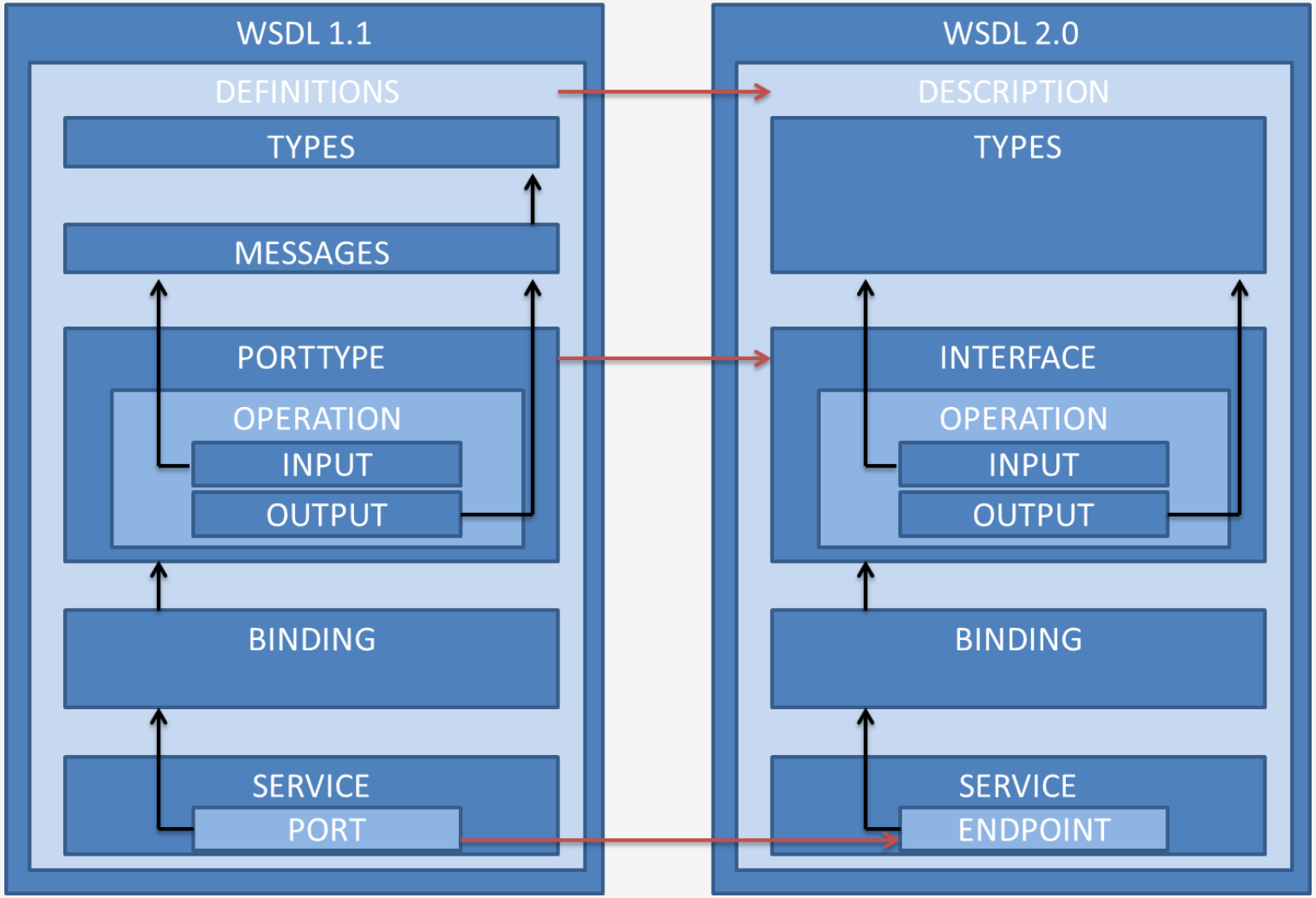
<port> : point d'accès

<service> : ensemble de port et leur binding associés



Description à 2 niveaux: Séparation entre la partie abstraite et concrète

WSDL : Version



Exemple : service Carnet d'adresses

3 opérations à mettre en œuvre :

- addPerson : Entrée (Person) + Sortie (booléen pour indiquer l'état de création)
- addPerson : Entrées (3 string : name, address et birthyear)
- getPersonByName : Entrée (string) + Sortie (Person)
- getPersons : Sortie (tableau d'objets Person)

Protocole SOAP : pour décrire les messages

Protocole HTTP : pour l'échange de messages

WSDL : l'élément types

2 variantes :

- Contient la définition des types de données (Schéma XSD, schéma XML…)
- Facultatif si les types sont simples (Integer, Boolean…)

- Importe un fichier Schéma XML contenant la définition des types
- Avantage : réutiliser des types et d'alléger le fichier WSDL

Exemple WSDL : définition des types

```
<definitions
      name="Notebook"
      targetNamespace="http://notebookwebservice.lisi.ensma.fr/"
      xmlns="http://schemas.xmlsoap.org/wsdl/"
      xmlns:tns="http://notebookwebservice.lisi.ensma.fr/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
      <types>
            <xsd:schema targetNamespace="http://notebookwebservice.lisi.ensma.fr/">
                  <xsd:complexType name="person">
                        <xsd:sequence>
                              <xsd:element name="address" type="xs:string" minOccurs="0"/>
                              <xsd:element name="birthyear" type="xs:string" minOccurs="0"/>
                              <xsd:element name="name" type="xs:string" minOccurs="0"/>
                        </xsd:sequence>
                  </xsd:complexType>
                  <xsd:complexType name="personArray" final="#all">
                        <xsd:sequence>
                              <xsd:element name="item" type="tns:person" minOccurs="0" maxOccurs="unbounded" nillable="true"/>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:schema>
      </types>
      ...
</definitions>
```

Une personne est définie par une *adresse*, une *année de naissance* et un *nom*

Définition d'un type tableau de personne

WSDL : l'élément message

Décrit un message échangé par le service et est défini par les attributs :

- Name : nom du message

- 1 ou plusieurs éléments <part> : inputs/outputs ou exception d'une opération

Chaque élément <part> est défini par les attributs :
- name : nom du paramètre
- type : type de données du paramètre

WSDL : l'élément message

```
<definitions
      targetNamespace="http://notebookwebservice.lisi.ensma.fr/"
      name="Notebook"
      ...>
      <types>
            ...
      </types>
      <message name="addPersonWithComplexType">
            <part name="newPerson" type="tns:person"/>
      </message>
      <message name="addPersonWithComplexTypeResponse">
            <part name="addPersonWithComplexTypeResult" type="xsd:boolean"/>
      </message>
      <message name="addPersonWithSimpleType">
            <part name="name" type="xsd:string"/>
            <part name="address" type="xsd:string"/>
            <part name="birthyear" type="xsd:string"/>
      </message>
      <message name="getPerson">
            <part name="personName" type="xsd:string"/>
      </message>
      <message name="getPersonResponse">
            <part name="getPersonResult" type="tns:person"/>
      </message>
      <message name="getPersons"/>
      <message name="getPersonsResponse">
            <part name="getPersonsResult" type="tns:personArray"/>
      </message>
</definitions>
```

Message utilisé pour l'appel d'une opération avec une seule partie

Message utilisé pour le résultat d'une opération avec une seule partie

Message utilisé pour l'appel d'une opération avec trois parties

Une partie qui pointe sur un type défini par l'élément *<types>*

WSDL: les éléments portType et opération

Interface contenant les prototypes d'opérations fournies par le service et définie par les attributs :

- Name : nom du portType
- <opération> : 1 ou plusieurs décrivant chacun le prototype d'une méthode fournie par le service

Chaque élément <opération> est défini par les attributs :

- name : nom de l'opération
- <input message>/<output message>/ <fault> : référence à 1 message décrivant les paramètres d'E/ S et le message d'erreur de l'opération

Exemple WSDL : l'élément portType

```
<definitions
      targetNamespace="http://notebookwebservice.lisi.ensma.fr/"
      name="Notebook"
      ...>
      <types>
            ...
      </types>
      <message>
            ...
      </message>
      <portType name="Notebook">
            <operation name="addPerson">
                  <input message="tns:addPersonWithComplexType"/>
                  <output message="tns:addPersonWithComplexTypeResponse"/>
            </operation>
            <operation name="addPerson" parameterOrder="name address birthyear">
                  <input message="tns:addPersonWithSimpleType"/>
            </operation>
            <operation name="getPerson">
                  <input message="tns:getPerson"/>
                  <output message="tns:getPersonResponse"/>
            </operation>
            <operation name="getPersons">
                  <input message="tns:getPersons"/>
                  <output message="tns:getPersonsResponse"/>
            </operation>
      </portType>
</definitions>
```

L'opération *addPerson* est surchargée

Possibilité de fixer l'ordre des paramètres définis par cette opération

Types d'échanges assurés par une opération

One-way : Message input sans réponse

```
<operation name="addPerson" parameterOrder="name address birthyear">
    <input message="tns:addPersonWithSimpleType"/>
</operation>
```

Notification : Seul un message <output> est utilisé

```
<operation name="personStatus">
    <output message="trackingInformation"/>
</operation>
```

Request/Response : <input>, <output> et <fault>
Le service reçoit un message du client et répond à sa Requête

```
<operation name="addPerson">
    <input message="tns:addPersonWithComplexType"/>
    <output message="tns:addPersonWithComplexTypeResponse"/>
</operation>
```

Solicit - response : <input>, <output> et <fault>
Le client reçoit un message du service et répond au service

```
<operation name="clientQuery">
    <output message="bandWithRequest"/>
    <input message="bandwidthInfo" />
    <fault message="faultMessasge" />
</operation>
```

WSDL : l'élément binding

Décrit un portType du point de vue technique , les informations sur le protocole de transport utilisé ;
Protocole (SOAP 1.1 ou 1.2, HTTP GET & Post : ex, transfert d'images) utilisé pour manipuler un <portType>

Est défini par les attributs : name : nom du binding , type : portType concerné
Sa structure (éléments qu'ils contient) dépend du protocole utilisé
Plusieurs <binding> peuvent être définis pour appeler un portType de différentes manières

```
<definitions>
     ...
     <binding name="NamePortBinding" type="tns:portType">
          <!-- Décrit le protocole à utiliser -->
          <operation name="operation1">
                    <!-- Action du protocole sur l'opération -->
               <input>
                    <!-- Action du protocole sur les messages d'entrés (input) -->
               </input>
               <output>
                    <!-- Action du protocole sur les messages de sorties (ouput) -->
               </output>
               <fault>
                    <!-- Action du protocole sur les messages d'erreurs (fault) -->
               </fault>
          </operation>
     </binding>
     ...
</definitions>
```

Ces informations sont spécifiques au protocole utilisé

WSDL : Binding SOAP

A généralement la forme <soap:binding> et est défini par l'espace de noms :

http://schemas.xmlsoap.org/wsdl/soap/

Utilise les principales balises suivantes :

- <soap:binding>
- <soap:operation>
- <soap:body>, <soap:header>, <soap:headerfault>
- <soap:fault>

WSDL : les éléments service et port

Un élément <service> définit les ports (points d'accès) du service
Chaque <port> spécifie 1 adresse pour 1 binding donné.
Un élément <port> est défini par les attributs :

- name : nom du port
- binding : nom du binding (défini précédemment)

La structure de l'élément <port> est spécifique au protocole utilisé dans le binding
- <soap:address> : URI du port dans le cas d'un binding SOAP

```
<definitions ...>
     <!-- Définition de la partie Abstraite du WSDL -->

     <binding ...>
     </binding>
     <service name="Notebook">
          <port name="NoteBookPort" binding="tns:NoteBookPortBinding">
               <soap:address location="http://localhost:8080/NotebookWebService/notebook"/>
          </port>
     </service>
</definitions>
```

Le portType Notebook est accessible en SOAP/HTTP via cette URL

Un autre service : HelloWorld

2 opérations : 1) makeHello : Entrée (string) et Sortie (string) 2) simpleHello : Sortie (string)
Protocole SOAP : pour décrire les messages , et le Protocole HTTP : pour l'échange de messages

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<definitions name="HelloWorld"
     targetNamespace="http://helloworldwebservice.lisi.ensma.fr/"
     xmlns="http://schemas.xmlsoap.org/wsdl/"
     xmlns:tns="http://helloworldwebservice.lisi.ensma.fr/"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  <types/>
  <message name="makeHelloWorld">
    <part name="value" type="xsd:string"/>
  </message>
  <message name="makeHelloWorldResponse">
    <part name="helloWorldResult" type="xsd:string"/>
  </message>
  <message name="simpleHelloWorld"/>
  <message name="simpleHelloWorldResponse">
    <part name="helloWorldResult" type="xsd:string"/>
  </message>
  <portType name="HelloWorld">
    <operation name="makeHelloWorld">
      <input message="tns:makeHelloWorld"/>
      <output message="tns:makeHelloWorldResponse"/>
    </operation>
    <operation name="simpleHelloWorld">
      <input message="tns:simpleHelloWorld"/>
      <output message="tns:simpleHelloWorldResponse"/>
    </operation>
  </portType>
```

```
<?xml version="1.0" encoding="
<definitions name="AktienKurs"
  targetNamespace="http://loca
  xmlns:xsd="http://schemas.xmlsoap.or
  xmlns="http://schemas.xmlsoap.org/wsd
  <service name="AktienKurs">
    <port name="AktienSoapPort" binding
      <soap:address location="http://loc
    </port>
    <message name="Aktie.HoleWert">
      <part name="body" element="xsd:Tra
    </message>
    …
  </service>
</definitions>
```

WSDL

Un autre service : HelloWorld

```
<binding name="HelloWorldPortBinding" type="tns:HelloWorld">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
  <operation name="makeHelloWorld">
    <soap:operation soapAction=""/>
    <input>
      <soap:body use="literal" namespace="http://helloworldwebservice.lisi.ensma.fr/"/>
    </input>
    <output>
      <soap:body use="literal" namespace="http://helloworldwebservice.lisi.ensma.fr/"/>
    </output>
  </operation>
  <operation name="simpleHelloWorld">
    <soap:operation soapAction=""/>
    <input>
      <soap:body use="literal" namespace="http://helloworldwebservice.lisi.ensma.fr/"/>
    </input>
    <output>
      <soap:body use="literal" namespace="http://helloworldwebservice.lisi.ensma.fr/"/>
    </output>
  </operation>
</binding>
<service name="HelloWorld">
  <port name="HelloWorldPort" binding="tns:HelloWorldPortBinding">
    <soap:address location="TODO"/>
  </port>
</service>
</definitions>
```

Exercice

Annoter ce WSDL de vos commentaires :à quoi il sert, comment s'en servir d'un point de vue client, quelles données vont circuler quand on l'utilise, etc.

Cliquez ici pour visualiser le Fichier WSDL

Solution

Ce service web offre une unique opération : sur envoi d'un message de type GetLastTradePrice, l'opération nommée GetLastTradePrice va s'exécuter. L'URL pour déclencher cette opération depuis un navigateur web devrait :

http://example.com/stockquote/method=GetLastPrice?tickerSymbol=MonEntreprisePreferee

Le message sera véhiculé par une enveloppe SOAP (on a défini un soap binding). Le type de donnée en entrée est un tickerSymbol prenant une valeur de type chaine de caractères. En réponse, on obtient un price que l'on peut interpréter comme un flottant.

Outils pour manipuler des documents WSDL

Edition

Notepad++ (texte : XML)
Eclipse JavaEE
Netbeans
Visual Studio
Environnements de développement de SW

Validation

www.validwsdl.com

Test

SOAPUI

En résumé……

En résumé WSDL c'est un contrat entre un client et un serveur qui fait état :

- Des spécifications d'interfaces qui décrivent toutes les méthodes publiques,
- Des spécifications relatives aux types de donnée de messages mis en œuvre dans les questions-réponses.
- Des informations liées au protocole de transport utilisé.
- Des informations d'adresse permettant de localiser le service décrit.

En un mot, WSDL définit le contrat existant entre un client et un serveur sans dépendance particulière pour une plateforme ou un langage.

Université Abderrahmane Mira de Béjaia
Faculté des Sciences Exactes
Département d'Informatique

Le SOAP
(Cours 4)

Dr H. EL BOUHISSI

Novembre 2021

Objectifs du cours

1. Découvrir le protocole de consommation d'un Service Web

2. Comprendre les briques de base d'un message SOAP

Plan

1. Définition et Versions
2. Contenu de messages SOAP
3. Exemples de messages SOAP

SOAP (Simple Object Access Protocol)

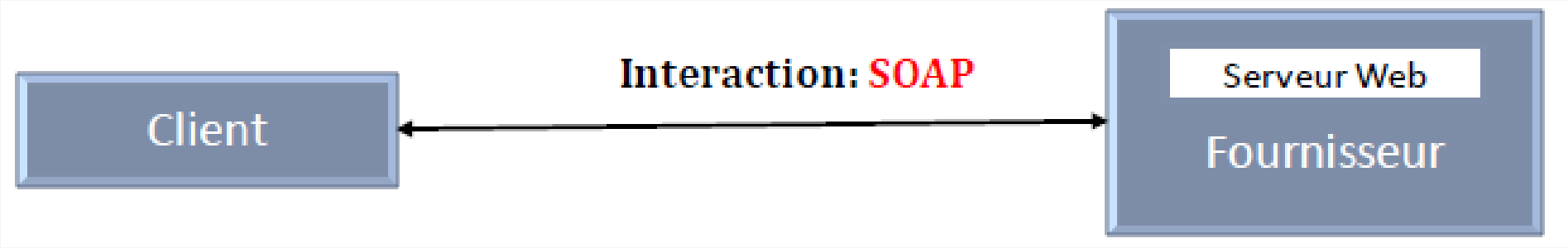
Le message SOAP est destiné au fournisseur du Service après avoir contacté l'annuaire pour chercher le service correspondant au besoin du client, les informations obtenues permettent au client de connaitre la localisation du service pour pouvoir l'invoquer à l'aide de messages SOAP.

- Protocole minimal pour l'appel des méthodes sur des serveurs, services, composants, objets.
- Protocole d'échange d'informations XML entre client et serveur HTTP.
- Standard de W3C, initié par Microsoft et IBM).

  SOAP 1.0 : 1999
  SOAP 1.1 : 2000, plus générique
  SOAP 1.2 : recommandation W3C, 2007: https://www.w3.org/TR/soap/
- Portable sur toutes les plateformes et technologies.

SOAP : Fonctionnement

La requête SOAP intervient sur le réseau entre le client et le serveur



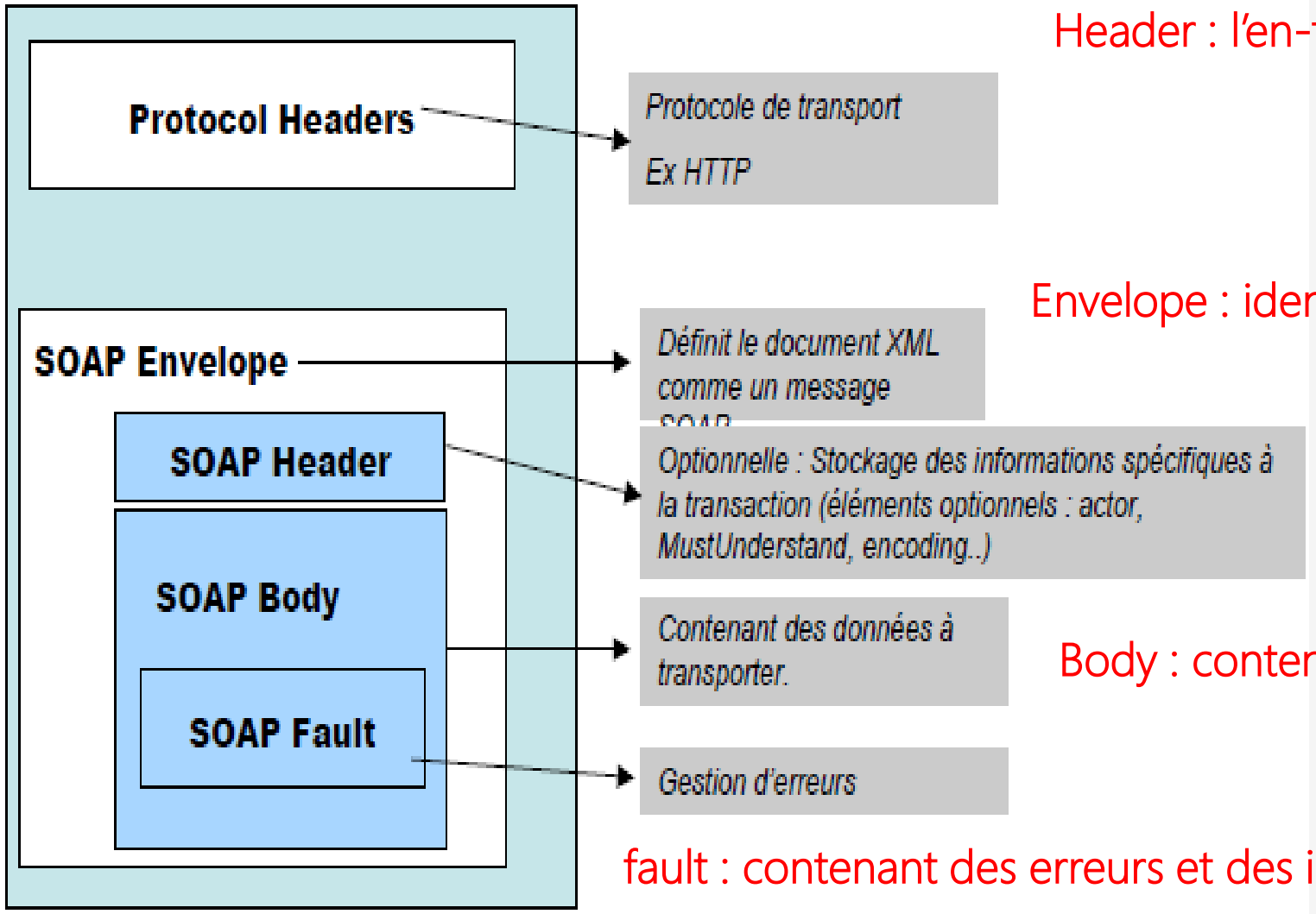
SOAP Côté client

Ouverture d'une connexion HTTP
Envoi d'une requête SOAP: document XML décrivant la méthode à invoquer sur la machine distante et les paramètres de cette méthode.

SOAP Côté Serveur

Récupération de la requête
Exécution de la méthode avec les paramètres
Renvoie une réponse SOAP (document XML) au client

Structure d'un message SOAP



Header : l'en-tête qui contient des informations d'en-tête

Envelope : identifie le document XML en tant que message SOAP

Body : contenant des informations d'appel et de réponse

fault : contenant des erreurs et des informations d'état

Structure d'un message SOAP

Envelope : contient le message et ses différentes sous-blocs. Il s'agit du bloc racine XML. Peut contenir un attribut *encodingStyle dont la valeur est une* URL vers un fichier de typage XML qui décrira les types applicables au message SOAP.

Header: bloc optionnel qui contient des informations d'en-têtes sur le message. Si il est présent, ce bloc doit toujours se trouver avant le bloc **Body à** l'intérieur du bloc **Envelope.**

Body: bloc qui contient le corps du message. Il doit absolument être présent de manière unique dans chaque message et être contenu dans le bloc **Envelope.**

Fault: la seule structure définie par SOAP dans le bloc Body. Il sert à reporter des erreurs lors du traitement du message, ou lors de son transport. Il ne peut apparaître qu'une seule fois par message. Sa présence n'est pas obligatoire.

Structure d'un message SOAP

Tous les éléments du SOAP sont déclarés dans l'espace de noms par défaut de l'enveloppe SOAP :

http://www.w3.org/2003/05/soap-envelope/

Et l'espace de noms par défaut pour le codage SOAP et les types de données est:

http://www.w3.org/2003/05/soap-encoding

SOAP : Exemple

SOAP REQUEST : demander le prix des pommes

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Body>
  <m:GetPrice xmlns:m="http://www.w3schools.com/prices">
    <m:Item>Apples</m:Item>
  </m:GetPrice>
</soap:Body>

</soap:Envelope>
```

Les éléments **m: GetPrice** et Item sont des éléments spécifiques à l'application et ils ne font pas partie de l'espace de noms SOAP.

SOAP : Exemple

SOAP RESPONSE : Fournir le prix des pommes

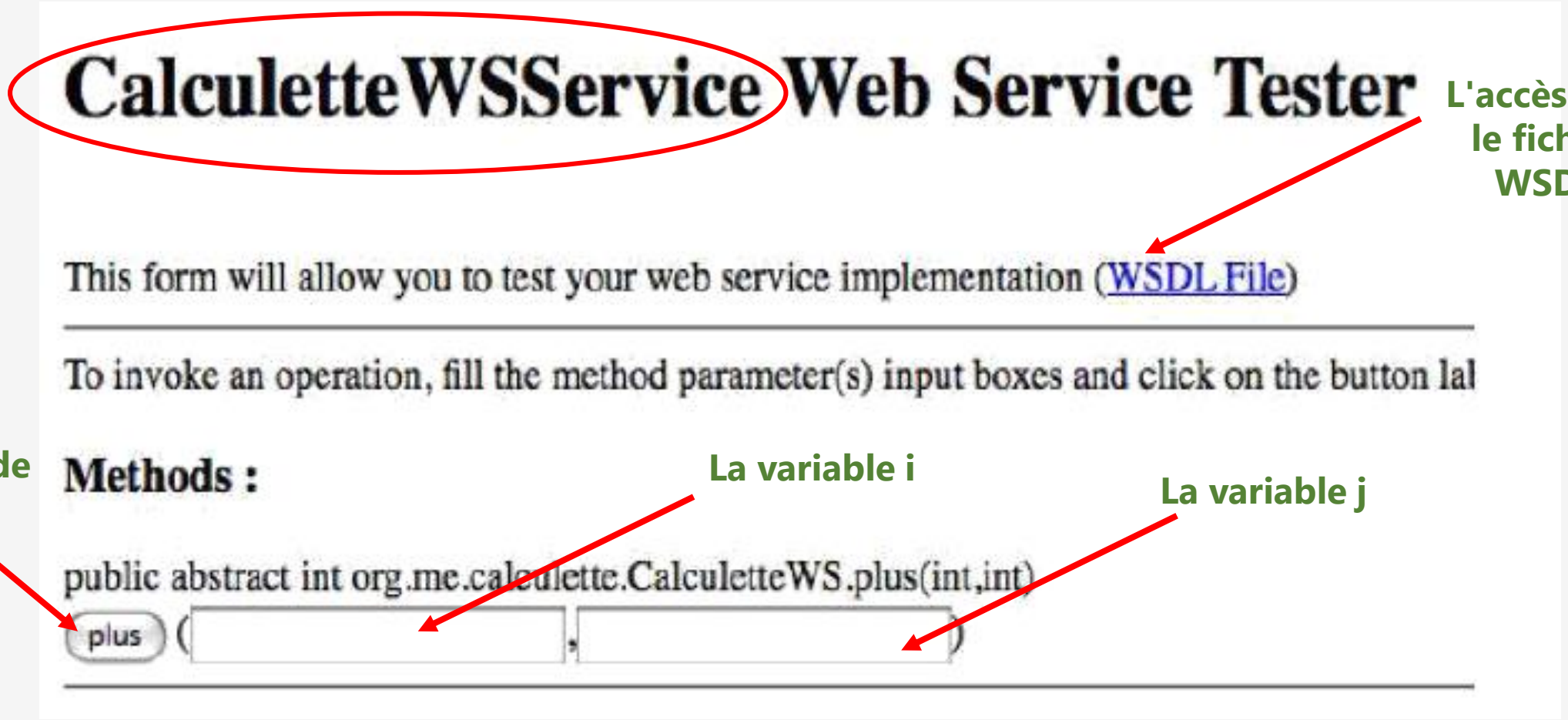
```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Body>
  <m:GetPriceResponse xmlns:m="http://www.w3schools.com/prices">
    <m:Price>1.90</m:Price>
  </m:GetPriceResponse>
</soap:Body>

</soap:Envelope>
```

SOAP : Exemple



Méthode plus qui calcule la somme k de deux entiers : i, j

SOAP : Exemple

plus Method invocation

Method parameter(s)

Type	Value
int	5
int	7

Method returned

int : "12"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
<S:Header/>
<S:Body>
<ns2:plus xmlns:ns2="http://calculette.me.org/">
<i>5</i>
<j>7</j>
</ns2:plus>
</S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
<S:Body>
<ns2:plusResponse xmlns:ns2="http://calculette.me.org/">
<return>12</return>
</ns2:plusResponse>
</S:Body>
</S:Envelope>
```

Université Abderrahmane Mira de Béjaia
Faculté des Sciences Exactes
Département d'Informatique

Le REST
(Cours 5)

Dr H. EL BOUHISSI

Novembre 2021

Objectifs du cours

1. Découvrir le les Service Web REST

2. Comprendre les briques de base des Services REST

REST - Définition

REST acronyme de REpresentational State Transfert : Concept introduit en 2000 dans la thèse de Roy FIELDING (un des créateurs du protocole HTTP).

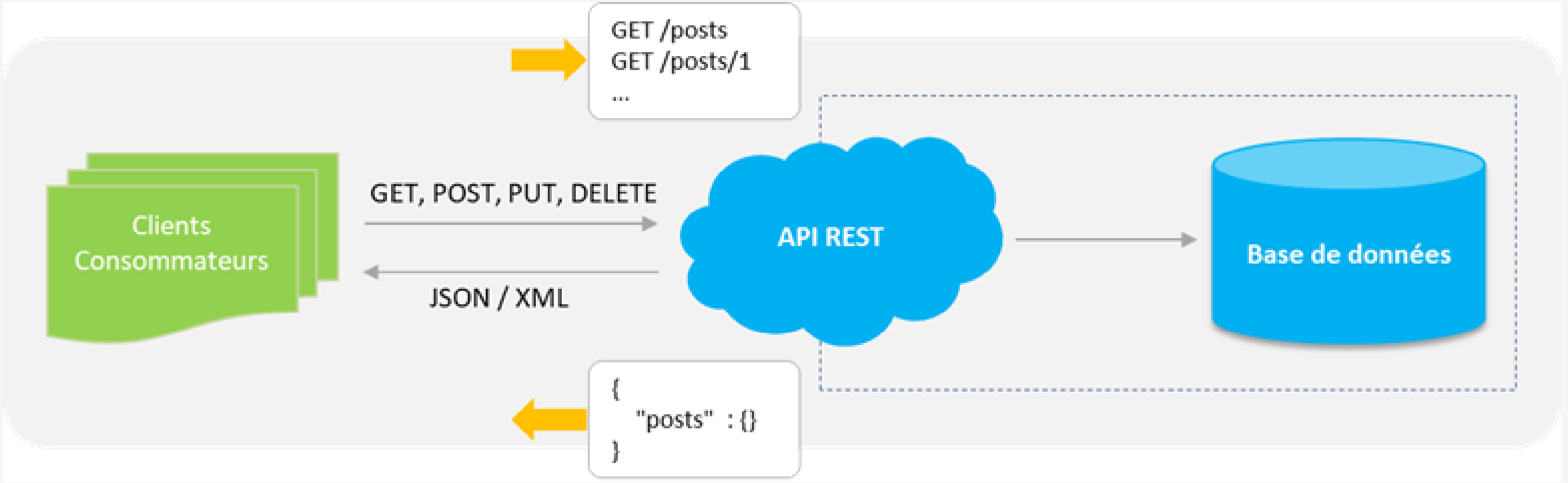
REST est une manière de construire une application pour les systèmes distribués comme le World Wide Web.

Ce n'est pas:

- Un format
- Un protocole
- Un standard

REST n'est pas un standard: Pas de recommandation du W3C

Les 5 règles à suivre pour implémenter REST



Style d'architecture inspiré de l'architecture WEB et repose sur le HTTP.

Permet l'envoi de messages sans enveloppe SOAP et dans un encodage libre (XML, JSON, binaire, simple texte).

Les 5 règles à suivre pour implémenter REST

Règle n°1 : **l'URI comme identifiant des ressources**

REST se base sur les URI (Uniform Resource Identifier) afin d'identifier une ressource. Il est nécessaire de prendre en compte la hiérarchie des ressources et la sémantique des URL pour les éditer. Par exemple : une ressource : « liste de livres »

Non OK : http://mywebsite.com/book OK : http://mywebsite.com/books

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) → POST
Afficher (read) → GET
Mettre à jour (update) → PUT
Supprimer (delete) → DELETE

Les 5 règles à suivre pour implémenter REST

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.

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.

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

Comment authentifier une requête ? La réponse est très simple et est massivement utilisée par des APIs renommées (Google, Yahoo, etc.) : **le jeton d'authentification**.

Services REST : Fournisseurs

Description de services Web REST

Langages : WSDL 2.0,WADL, RSDL (Restful Service Description Language), SERIN (Semantic RESTFul Interfaces).

- **WSDL2.0** : Évolution de Web Service Description Language recommandé en 2007 par le W3C. Il permet de spécifier un binding HTTP au lieu de SOAP
- **WADL** (Web Application Description Language) : Un langage de description XML de services de type REST.

Permet une description de services par éléments de type: ressource, méthode, paramètre, requête, réponse.

Pas assez de Framework qui supportent la description WADL

Dr

Description de services Web REST : WADL

ELEMENTS WADL	DESCRIPTION
<application>	La racine d'une description WADL
<ressources base=…….>	• Un conteneur pour les ressources que l'application fournit • L'attribut base définit l'URI pour les ressource
<resource id="" path="">	• Décrit la ressource que l'application fournit • L'attribut id identifie l'élément ressource • L'attribut path fournit une URI relative pour l 'identifiant de la ressource.
<method name=" " id="">	• Fils d'élément ressource ou application • L'attribut name définit les méthodes HTTP
<request> <response>	• Request: décrit une requête à la méthode HTTP sur une ressource • Response: décrit la sortie en réalisant la méthode HTTP sur le ressource

Description de services Web REST : WADL

EXEMPLE de description WADL pour l'application Yahoo News Search

```
<?xml version="1.0"?>
<application xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://wadl.dev.java.net/2009/02 wadl.xsd"
 xmlns:tns="urn:yahoo:yn"   xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:yn="urn:yahoo:yn"
 xmlns:ya="urn:yahoo:api"   xmlns="http://wadl.dev.java.net/2009/02">
  <grammars> <include  href="NewsSearchResponse.xsd"/> <include href="Error.xsd"/>
  </grammars>
  <resources base="http://api.search.yahoo.com/NewsSearchService/V1/">
    <resource path="newsSearch">
      <method name="GET" id="search">
        <request>
          <param name="appid" type="xsd:string" style="query" required="true"/>
          <param name="query" type="xsd:string"  style="query" required="true"/>
          <param name="type" style="query" default="all">
            <option value="all"/> <option value="any"/> <option value="phrase"/>
          </param>
          <param name="results" style="query" type="xsd:int" default="10"/>
          <param name="start" style="query" type="xsd:int" default="1"/>
          <param name="sort" style="query" default="rank">
            <option value="rank"/> <option value="date"/>
          </param>
          <param name="language" style="query" type="xsd:string"/>
        </request>
        <response status="200">
          <representation mediaType="application/xml" element="yn:ResultSet"/>
        </response>
        <response status="400">
          <representation mediaType="application/xml" element="ya:Error"/>
        </response>
      </method>
    </resource>
  </resources>
</application>
```

Description des espaces de noms

Description de Grammaire XML utilisée par le serv

Description des ressources Et méthodes HTTP utilisées

Plus facile à comprendre, à interpréter et à écrire qu'un WSDL

Exemples de service web REST

TP 5 et TP 6

REST vs SOAP (Simple Object Access Protocol)

SOAP	REST
SOAP est un protocole.	REST est un style d'architecture.
SOAP signifie Simple Object Access Protocol.	REST signifie REPresentational State Transfer.
SOAP ne peut pas utiliser REST car c'est un protocole.	REST peut utiliser les services Web SOAP car il s'agit d'un concept et peut utiliser n'importe quel protocole comme HTTP, SOAP.
SOAP utilise des interfaces de services pour exposer la logique métier.	REST utilise l'URI pour exposer la logique métier.
JAX-WS est l'API java pour les services Web SOAP.	JAX-RS est l'API java pour les services Web RESTful.
SOAP définit les normes à suivre strictement.	REST ne définit pas trop de normes comme SOAP.
SOAP nécessite plus de bande passante et de ressources que REST.	REST nécessite moins de bande passante et de ressources que SOAP.
SOAP définit sa propre sécurité.	Les services Web RESTful héritent des mesures de sécurité du transport sous-jacent.
SOAP autorise uniquement le format de données XML.	REST autorise différents formats de données tels que le texte brut, HTML, XML, JSON, etc.
SOAP est moins préféré que REST.	REST plus préféré que SOAP.

Université Abderrahmane Mira de Béjaia
Faculté des Sciences Exactes
Département d'Informatique

Le Registre UDDI

(Cours 6)

Dr EL BOUHISSI Houda

Février 2021

UDDI : Universal Description, Discovery and Integration

- À l'origine: annuaire universel pour les services web (à la Google)
- Aujourd'hui: vise plutôt les environnements privés, à petite échelle
- Raisons: peu d'annuaires généraux UDDI (IBM, Microsoft, …), contenu pauvre et non fiable
- Élément d'infrastructure qui aide aussi à stocker des infos absentes en WSDL.
- Version 1: les bases d'un annuaire de services
- Version 2: adaptation à SOAP et WSDL
- Version 3: redéfinition du rôle UDDI, accent sur les implémentations privées, sur l'interaction entre annuaires privés et publics

UDDI : Universal Description, Discovery and Integration

L'annuaire UDDI permet de :

- Publier, découvrir des informations sur une entreprise et ses services
- L'inscription sur UDDI permet à une entreprise de se présenter ainsi que ses services
- L'adoption de UDDI facilite le développement des échanges de type « B2B »
- • L'enregistrement des services dans un annuaire s'effectue auprès d'un opérateur (Microsoft ou IBM actuellement) à travers son site mais on peut créer ses propres registres UDDI (UDDI4J, jUDDI)
- Un annuaire à l'aide d'un browser en ligne:

http://soapclient.com/UDDIAdv.html

Types de structure de données (registre UDDI)

Une fois la connexion au registre UDDI, vous pouvez enregistrer votre service Web. L'enregistrement d'un service implique quatre types de structure de données essentiels : **informations métier**, **informations sur les services, informations sur les liaisons** et **informations décrivant les spécifications relatives aux services**.

Informations métier. Informations contenues dans une structure **businessEntity**. La structure businessEntity regroupe des informations sur l'entreprise ayant publié le service, tel que le nom, la description, les contacts et les identificateurs de cette entreprise.

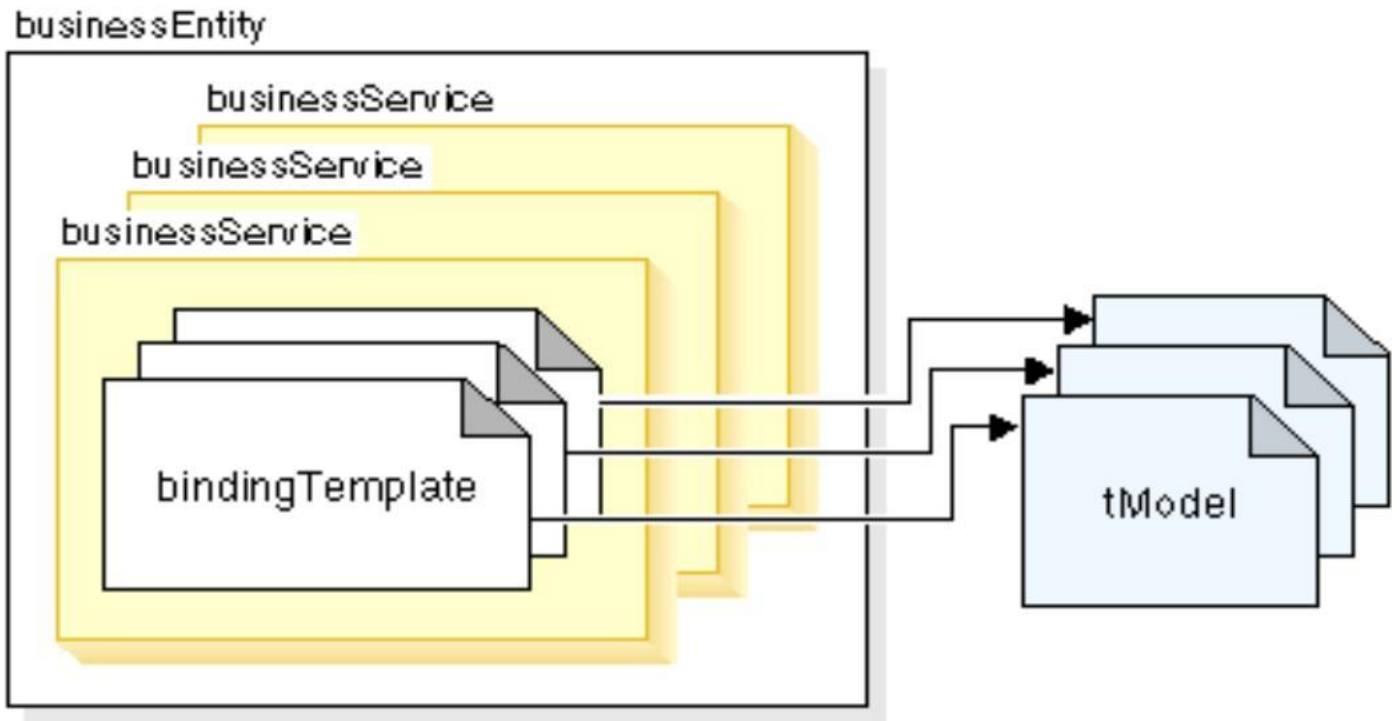
Informations sur le service. Informations décrivant un groupe de services Web. Elles sont contenues dans une structure **businessService**. La structure businessService contient des informations sur les familles de services techniques. Elle regroupe un ensemble de services Web associés à un processus métier ou à un groupe de services.

Types de structure de données (registre UDDI)

- **Informations de liaison.** Informations représentées par la structure **bindingTemplate**. La structure bindingTemplate détient des informations techniques servant à déterminer le point d'entrée et les spécifications de construction pour l'appel d'un service Web. La structure bindingTemplate fournit les descriptions de service Web utiles aux développeurs d'applications souhaitant rechercher et appeler un service Web. La structure bindingTemplate pointe vers des descriptions d'implémentation d'un service, par le biais d'une adresse URL, par exemple.
- **Informations décrivant les spécifications pour les services.** Les métadonnées afférentes aux différentes spécifications implémentées par un service Web donné, sont représentées par la structure **tModel**. tModel fournit un système de références facilitant la reconnaissance des services Web.

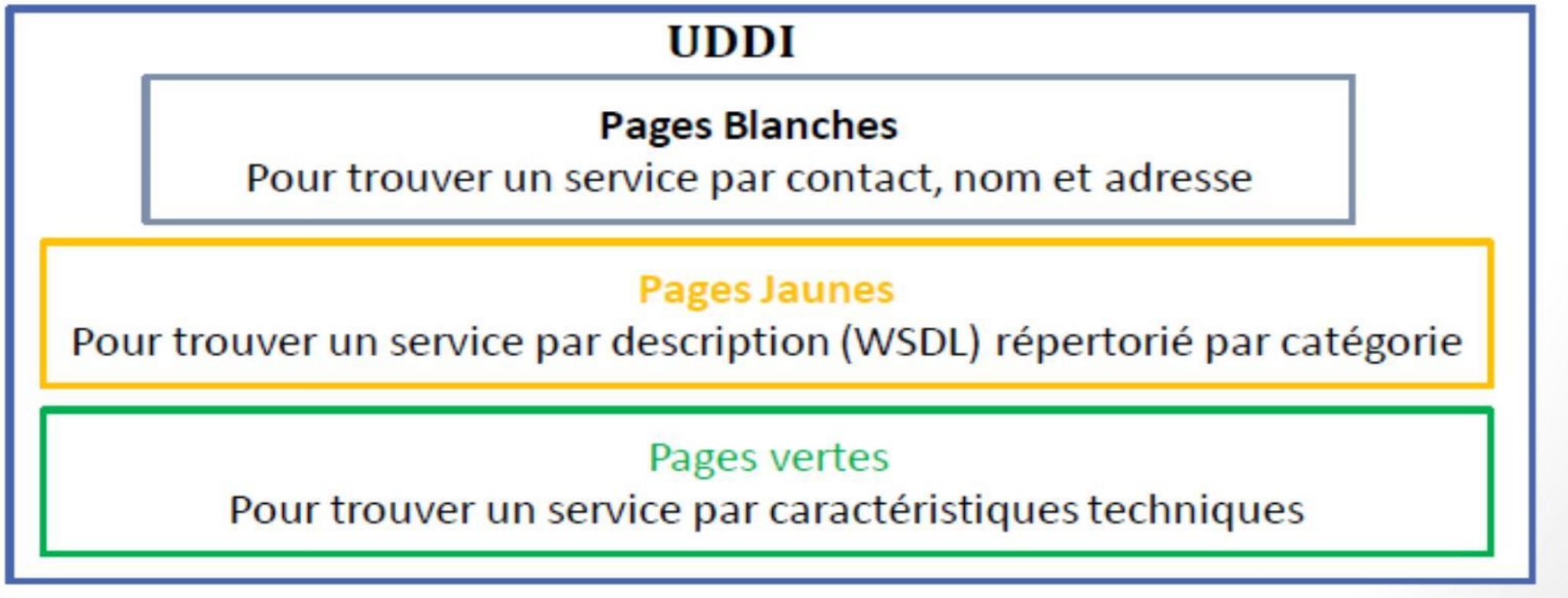
Types de structure de données (registre UDDI)

Chaque structure businessService est détenue par une structure businessEntity spécifique. De même, chaque structure bindingTemplate est détenue par une structure businessService spécifique. Chaque structure bindingTemplate référence des instances uniques de structures tModel, il peut y avoir plusieurs références à des structures tModel à partir de différents modèles parent Entity-Service-Template.



UDDI : Universal Description, Discovery and Integration

Comporte plusieurs catégories de données: Informations organisées en trois méthodes :



Publication de service Web

- La publication d'un service Web requiert que l'entreprise s'authentifie auprès de l'opérateur UDDI

- Il faut fournir les données nécessaire à l'exploitation du service Web (IP, Nom de domaine, modalités d'utilisation …)

Université Abderrahmane Mira de Béjaia
Faculté des Sciences Exactes
Département d'Informatique

Tâches relatives aux Services Web

(Cours 7)

Dr EL BOUHISSI Houda

Février 2021

Services Web : Rappel

Les services Web emploient un ensemble de technologies qui ont été conçues afin de respecter une structure en couches sans être dépendante de façon excessive de la pile des protocoles. Cette structure est formée de quatre couches majeures

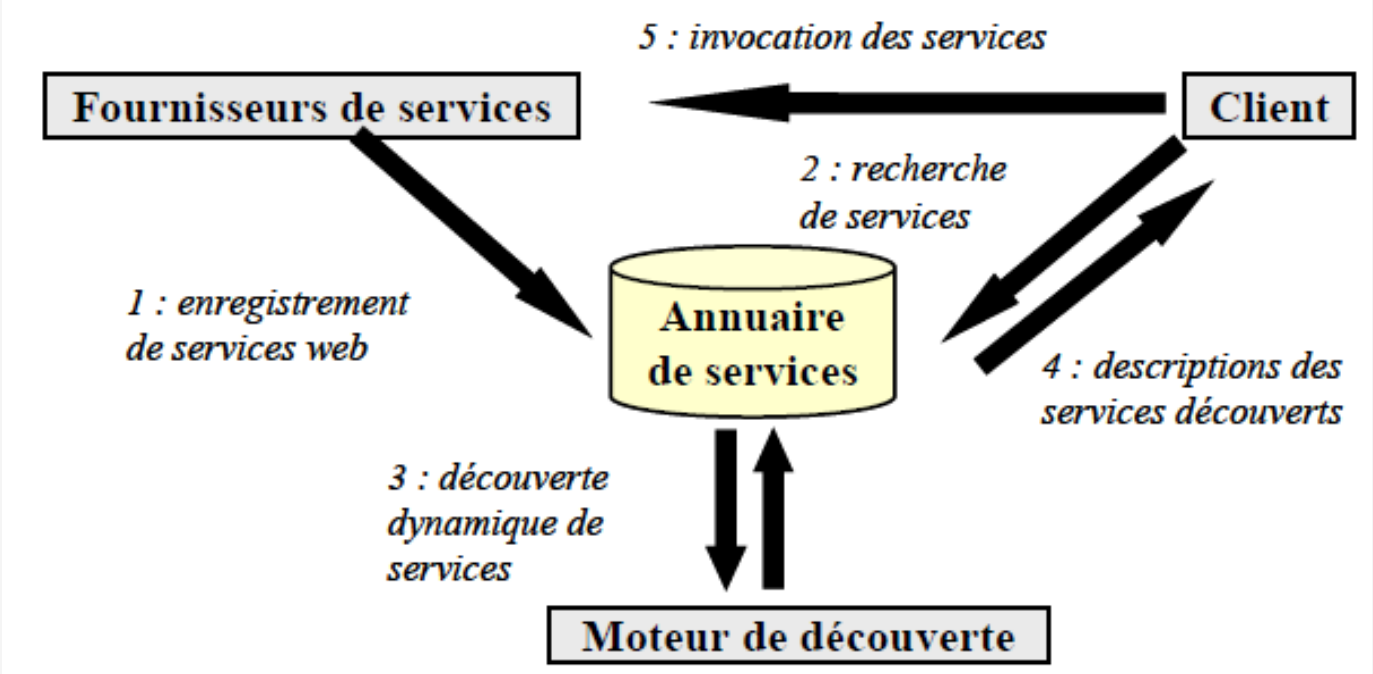
Découverte de services	UDDI
Description de services	WSDL
Communication	SOAP
Transport	HTTP

Problèmes liés aux Services Web

Les problématiques associées aux services web sont nombreuses:

- La découverte (i.e. trouver des services correspondant à une requête utilisateur),
- La sélection (i.e. identifier le meilleur service parmi ceux découvert)
- La composition (i.e. construire un nouveau service à partir d'une combinaison de services existants)
- La supervision (i.e. suivre l'exécution d'un service)
- La simulation (i.e. tester avant déploiement)
- La vérification (i.e. s'assurer qu'un service va bien faire ce qu'il est censé faire)
- Le traitement d'exception (i.e. pouvoir gérer des erreurs au cours de l'exécution d'un service).

Découverte des Services Web



La découverte de Services web désigne le processus par lequel les futurs utilisateurs ou consommateurs d'un service recherchent manuellement ou semi automatiquement le service correspondant à leur besoin. Les services web auront été préalablement créés et publiés dans des annuaires de services web comme l'UDDI.

Sélection des Services Web

Avec la sélection des services web, on cherche à choisir le meilleur fournisseur d'un service web, étant donné un ensemble de fournisseurs de ce service

Composition : Définition

Décomposition d'un grand problème en parties plus petites, donc plus gérables
Comment assembler ces petits services ensemble pour créer des services à valeur ajoutée?

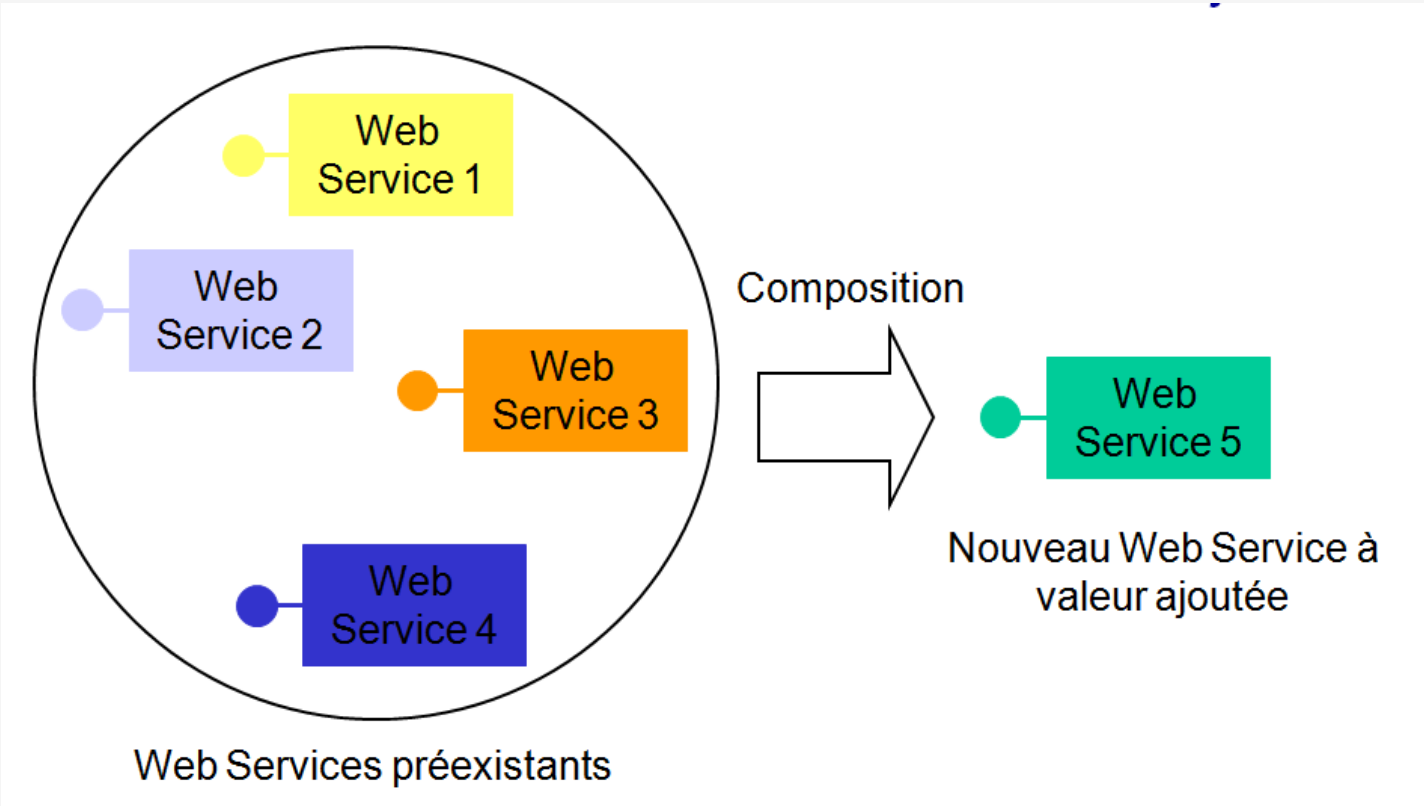
Un service peut participer à des compositions de services
- Un ensemble de services peuvent être composés pour répondre à un besoin complexe

Avantages :
- Apport de valeur ajoutée (répondre à un nouveau besoin complexe)
- Augmentation de la modularité : vu qu'un service complexe peut être décomposé en services simples pouvant être déployés chacun de façon atomique

Exemple de Scénario : **Agence de voyage**
- **Solution 1**: L'utilisateur se rend sur chaque site Web (location de voiture, réservation d'hôtel, ,,etc.)
- **Solution 2**: L'utilisateur utilise un portail de services

Composition : Définition

La composition de services est le mécanisme qui permet l'intégration des services pour construire un nouveau Web service à valeur ajoutée.



Composition : Définition

Implémentation d'une application (offerte comme service) dont la logique implique l'invocation d'opérations offertes par d'autres services.

- Le nouveau service est appelé service composite
- Les services invoqués sont des composants de service

Du point de vue du client, un service composite et un service basique sont in-distinguables.

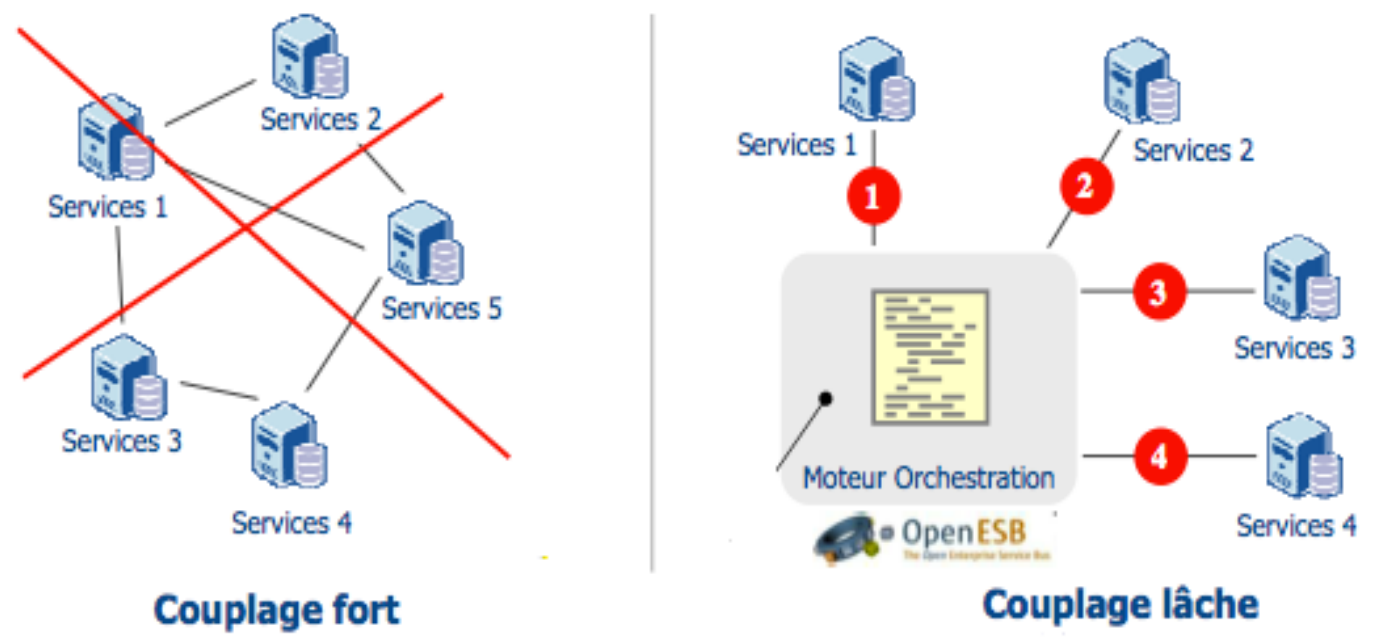
Deux approches principales pour la composition des services web :

- Orchestration
- Chorégraphie

Orchestration

Seul un service « moteur d'orchestration » connaît la logique de composition et appelle les autres pour effectuer un service complexe selon l'enchaînement désiré,

L'orchestration leur permet de communiquer sans avoir à se connaître pour préserver leur couplage lâche (leur indépendance)



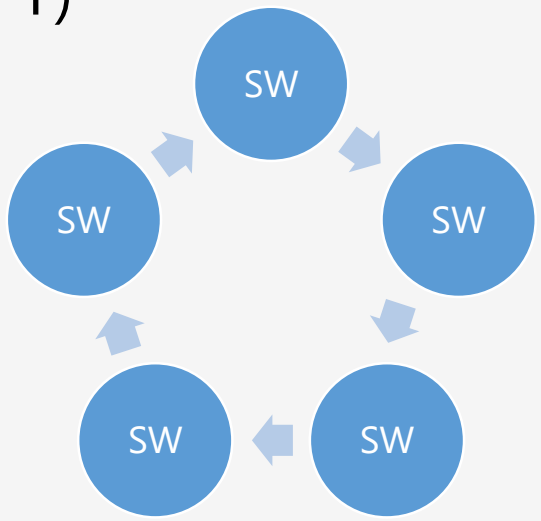
- Approche centralisée autour du moteur de composition (En cas de panne, plus de composition)
- Schéma de composition statique (En cas de changement dans les besoins, le schéma devient invalide

Chorégraphie

Un effort de collaboration dans lequel chaque participant du processus décrit l'itération qui l'appartient.

Le comportement global du processus est basé sur l'interaction des différentes parties, chacune suivant son propre rôle de manière autonome.

Chaque service contribuant au service composite doit connaître sa logique (ex : attendre 50 s quand il est invoqué par le service X avant d'envoyer un message au service Y)



- Collaborations et partenaires statiques (Si les besoins ou partenaires changent, les collaborations deviennent impossibles).
- Pas de langage pour exprimer les besoins

Service Web sémantique

Un service Web sémantique prolonge les capacités d'un service Web en associant des concepts sémantiques au service Web afin de permettre une meilleure recherche, découverte, choix, composition et intégration.

OWLS, la langue de spécifications a été développée pour fournir un mécanisme pour décrire des concepts de domaine en utilisant des ontologies. Elles fournit des constructions pour indiquer des informations sur la façon de composer et d'accéder à des services.

Dans l'état actuel des Services Web non sémantiques, c'est à dire syntaxique, on développe le service à l'aide des outils tels que le Visual Studio.NET et puis générer automatiquement des spécifications appropriées WSDL pour ce service.