2éme année Master Informatique

Option: **Génie Logiciel et Systèmes Distribués**

Université de Biskra

Année 2021/2022

Architectures Orientées Services

Dr. Mohamed Lamine KERDOUDI

Email : **l.kerdoudi@univ-biskra.dz**

4. Concepts de bases et principes de fonctionnement des services

4.1 Service as Web Service:

La technologie des services Web représente maintenant le moyen le plus important et le plus populaire pour implémenter des architectures orientées services.

4.2 Service as REST Service

❑ Définition d'un Service Web

Un service peut être défini comme suit:

«*Un composant logiciel réutilisable et faiblement couplé qui encapsule des fonctionnalités discrètes, qui peuvent être distribuées et accessibles par des programmes. Un service Web est un service auquel on accède en utilisant des standards d'Internet et des protocoles basés XML [Sommerville, 2011]* »

Un service Web est un le corps d'une solution qui fournit un contrat de service composé d'une définition WSDL et d'une ou plusieurs définitions de schéma XML, ainsi que d'éventuelles **expressions WS-Policy (à voir plus tard).**

4. Concepts de bases et principes de fonctionnement des services

❑ Caractéristiques de services Web

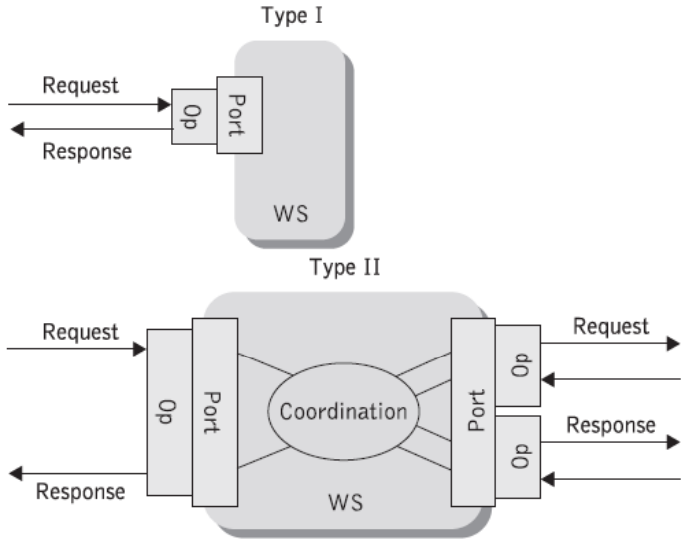
- Les services Web sont très différents des pages Web qui fournissent également un accès aux applications sur Internet et à travers les frontières organisationnelles.
- Les pages Web sont destinées aux utilisateurs humains, tandis que les services Web sont développés pour l'accès par les applications automatisées.
- Il est facile de confondre quelqu'un en décrivant un «service» comme un service Web alors que ce n'est pas le cas.

4. Concepts de bases et principes de fonctionnement des services

❑ Types de services Web

❖ **Services Web informatifs** : qui prennent en charge uniquement les opérations de requête/ réponse simples et attendent toujours une demande; ils le traitent et répondent.

❖ **Les services Web complexes** (business processes): implémentent une certaine forme de coordination entre les opérations entrantes et sortantes.



❖ Chacun de ces deux modèles présente plusieurs caractéristiques importantes.

4. Concepts de bases et principes de fonctionnement des services

❑ Propriétés fonctionnelles et non-fonctionnelles

- La description fonctionnelle détaille les caractéristiques opérationnelles qui définissent le comportement global du service

  ➢ Elle définit les détails de « comment le service est invoqué ? », l'emplacement « où il est invoqué ? », et ainsi de suite.

  ➢ Cette description concerne la syntaxe des messages et comment configurer les protocoles réseau pour délivrer des messages.

- La description non fonctionnelle se concentre sur les attributs de qualité de service, tels que:

  ➢ mesure de coût, métriques de performance, temps de réponse ou précision,

  ➢ attributs de sécurité, autorisation, authentification, intégrité (transactionnelle), fiabilité, évolutivité et disponibilité.

4. Technologies d'implémentation de services

Interface et implémentation de service Web

- La **partie interface de service** définit les fonctionnalités de service visibles par le monde extérieur et fournit les moyens d'accéder à cette fonctionnalité.
- Le service décrit ses propres caractéristiques d'interface, en terme d'opérations disponibles, de paramètres, de types de données et de protocoles d'accès,
- De cette manière, **d'autres modules logiciels** puissent déterminer ce qu'il fait, comment invoquer une fonctionnalité et quel résultat attendre. en retour.
- À cet égard, les services sont des modules logiciels fournissent des **descriptions accessibles au public** qui sont utilisées pour accéder au service.
- Le client de service utilise la description de l'interface du service pour se lier au fournisseur de services et invoquer sa fonctionnalité.

4. Technologies d'implémentation de services

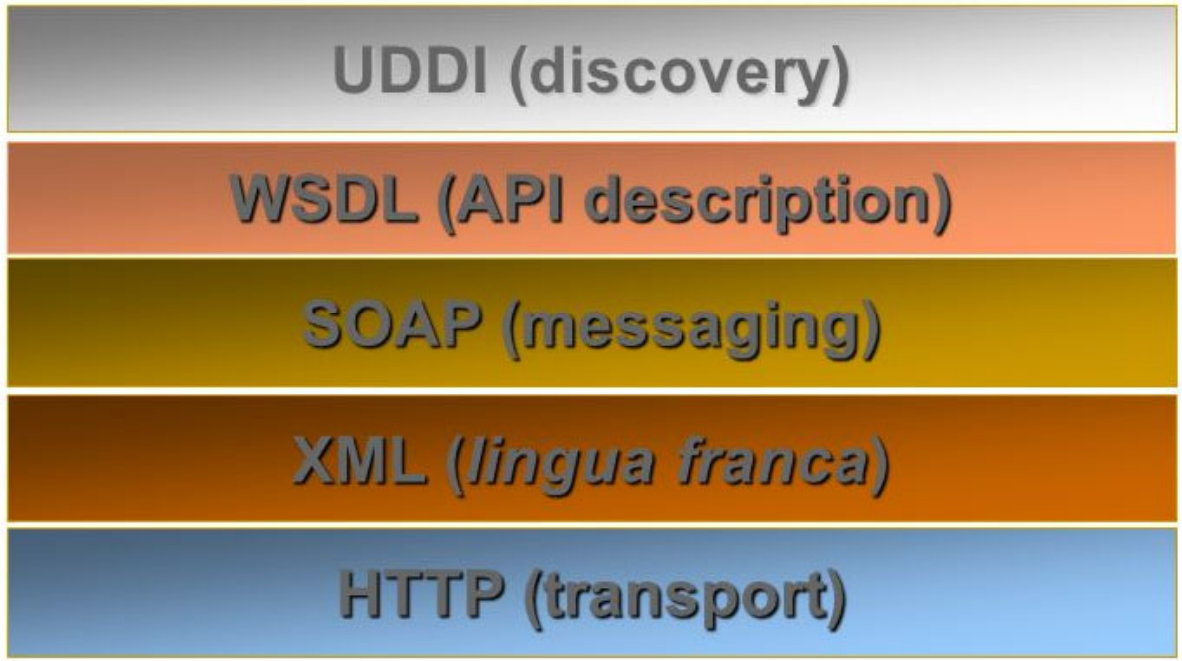
Interface et implémentation de service Web

- La **partie implémentation de service** permet de réaliser une interface de service spécifique dont les détails d'implémentation sont cachés aux utilisateurs du service.
- **Différents fournisseurs** de services peuvent implémenter la même interface en utilisant n'importe quel langage de programmation.
- Une implémentation de service peut fournir **directement la fonctionnalité elle-même**, alors qu'une autre implémentation de service peut utiliser **une combinaison d'autres services** pour fournir la même fonctionnalité.
- Dans la plupart des cas, les organisations qui fournissent des interfaces de service ne sont pas les mêmes que celles qui implémentent les services.

4. Concepts de bases et principes de fonctionnement des services

Pile des Standards des Services Web

- Trois organisations sont la clé à l'évolution des standards de services Web: W3C, OASIS, Organisation WSI
- Ces standards définissent à la fois les **protocoles** utilisés pour communiquer et le **format des interfaces** utilisées pour spécifier et les services et les contrats de service.
- Cinq standards fondamentaux: **XML, HTTP, WSDL, SOAP, UDDI.**



4. Concepts de bases et principes de fonctionnement des services

❑ Extensible Markup Language (XML):

❑ **XML** est utilisé comme format standard pour décrire les modèles, les formats et les types de données. Toutes les standards de services Web sont basées sur XML.

- ➢ Développé par W3C
- ➢ Norme ouverte sans frais
- ➢ Son objectif principal est de faciliter le partage de données structurées entre différents systèmes d'information, notamment via Internet.
- ➢ Un sous-ensemble de SGML (Standard Generalized Markup Language)

4. Concepts de bases et principes de fonctionnement des services

❑ **Hypertext Transfer Protocol (**HTTP) (y compris HTTPS) est le protocole de bas niveau utilisé par Internet.

➢ HTTP (S) est un des protocoles qui peut être utilisé pour envoyer des messages aux services Web sur les réseaux, en utilisant la technologie Internet.

➢ Dans le protocole HTTP, une méthode est une commande spécifiant un type de requête tel que:

- **GET** demander une ressource. Une requête GET est sans effet sur la ressource, il doit être possible de répéter la requête sans effet.

Exemple: **GET /page.html Host: www.perdu.com HTTP/1.1**

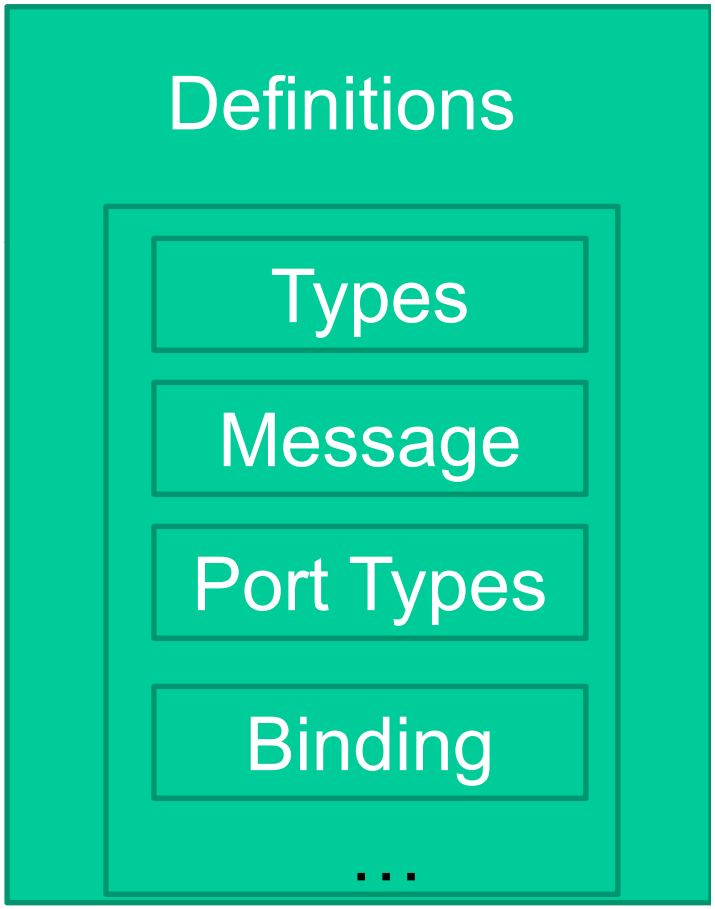
- **HEAD** Cette méthode ne demande que des informations sur la ressource (entête), sans demander la ressource elle-même.
- **POST** Cette méthode est utilisée pour transmettre des données en vue d'un traitement à une ressource (le plus souvent depuis un formulaire HTML).
- **PUT** Cette méthode permet de remplacer ou d'ajouter une ressource sur le serveur. L'URI fourni est celui de la ressource en question.
- **DELETE** Cette méthode permet de supprimer une ressource du serveur.
- ….

4. Concepts de bases et principes de fonctionnement des services

- **Web Service Description Language (WSDL)** WSDL est langage basé XML qui est utilisé pour décrire les interfaces de service Web.

  - **Structure d'un document WSDL**

```
<definitions>
   <types>
   définitions de type de données.....
   </types>
   <message>
    définition des données communiquées
   </message>
   <portType>
   ensemble d'opérations......
   </portType>
   <binding>
   spécification de protocole et de format
   de données....
   </binding>
</definitions>
```



4. Concepts de bases et principes de fonctionnement des services

Structure d'un document WSDL

Le fichier WSDL décrit l'interface d'un service Web contient les éléments suivants:

- **Types :** Contient les définitions de types décrits par un schéma XML;
- **Message :** Décrit les noms et types d'un ensemble de champs à transmettre (Paramètres d'une invocation, valeur du retour…)
- **Opération :** Correspond à une abstraction décrivant une action implémentée par un service Web

Les types d'opérations :

1- *One-way* : L'opération peut recevoir un message mais ne retournera pas de réponse

2- *Request-response* : L'opération peut recevoir un message et retourner une réponse

3- *Solicit-response* : L'opération peut envoyer un message et attendra une réponse.

4- *Notification* : L'opération peut envoyer un message mais n'attendra pas de réponse

4. Concepts de bases et principes de fonctionnement des services

▪ Structure d'un document WSDL

- **Les types de ports (PortType) :** Décrit un ensemble d'opérations. Chaque opération à 0 ou * messages en entrée, 0 ou 1 message en sortie ou une faute (exception) ;
- **Liaison (binding) :** Spécifie une liaison d'un <PortType> à un protocole concret (SOAP, HTTP, MIME (Multipurpose Internet Mail Extensions)…).
  - ❑ Un PortType peut avoir plusieurs liaisons ;
- **Service Web :** Est une collection de points d'entrée.
  - ❑ **Port :** Définit un point d'entrée (endpoint) comme la combinaison d'une <Liaison> et d'une adresse Internet (réseau) ;

➔ **Ces différentes parties peuvent être développées en tant que documents séparés.**

4. Concepts de bases et principes de fonctionnement des services

- **Exemple d'un document WSDL**

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>
<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>
```

- Dans cet exemple, l'élément <**portType**> définit "glossaryTerms" comme nom d'un port et "**getTerm**" comme nom d'une opération.
- L'opération "**getTerm**" a un message d'entrée : "**getTermRequest**" et un message de sortie : "**getTermResponse**".
- Les éléments <**message**> définissent les parties de chaque message et les types de données associés.

4. Concepts de bases et principes de fonctionnement des services

- **WSDL One-Way Operation**

```
<message name="newTermValues">
  <part name="term" type="xs:string"/>
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="setTerm">
    <input name="newTerm" message="newTermValues"/>
  </operation>
</portType >
```

- **WSDL Request-Response Operation**

```
<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>
```

4. Concepts de bases et principes de fonctionnement des services

WSDL Binding to SOAP

```
<binding type="glossaryTerms" name="b1">
   <soap:binding style="document"
   transport="http://schemas.xmlsoap.org/soap/http" />
   <operation>
     <soap:operation soapAction="http://example.com/getTerm"/>
     <input><soap:body use="literal"/></input>
     <output><soap:body use="literal"/></output>
  </operation>
</binding>
```

- L'élément « **binding** » a deux attributs: nom et type.
  - L'attribut **name** (n'importe quel nom) définit le nom de la liaison et
  - L'attribut **type** pointe vers le port, dans ce cas le port "**glossaryTerms**".
- élément « **soap:binding** » a deux attributs - style et transport.
  - L'attribut **style** peut être "**rpc**" ou "**document**".
  - L'attribut de **transport** définit le protocole à utiliser. Par exemple, **HTTP**.
- L'élément « **operation** » définit chaque opération que le portType expose.
  - Pour chaque opération, l'**action SOAP** correspondante doit être définie. Vous devez également spécifier comment E/S sont codées.
  - Dans ce cas, nous utilisons "**littéral**".

4. Concepts de bases et principes de fonctionnement des services

- **Exemple d'un document WSDL complet**

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
             xmlns:xs="http://www.w3.org/2001/XMLSchema"
            xmlns:tns="http://www.sun.com/AirlineReservationService"
             xmlns:ota="http://www.opentravel.org/OTA/2003/05"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
targetNamespace="http://www.sun.com/AirlineReservationService">
<types> <xs:schema
         targetNamespace="http://www.sun.com/AirlineReservationService">
        <xs:import namespace="http://www.opentravel.org/OTA/2003/05"
        schemaLocation="OTA_TravelItinerary.xsd"/>
 </xs:schema>
</types>
<message name="CancelAirlineIn">
    <part name="itinerary" element="ota:ItineraryRef"/>
</message>
<message name="CancelAirlineOut">
    <part name="succeeded" element="ota:CancellationStatus"/>
</message>
<portType name="AirlineReservationPortType">
   <operation name="cancelAirline">
    <input message="tns:CancelAirlineIn"/>
     <output message="tns:CancelAirlineOut"/>
   </operation>
</portType>
```

4. Concepts de bases et principes de fonctionnement des services

- **Exemple d'un document WSDL --- suite**

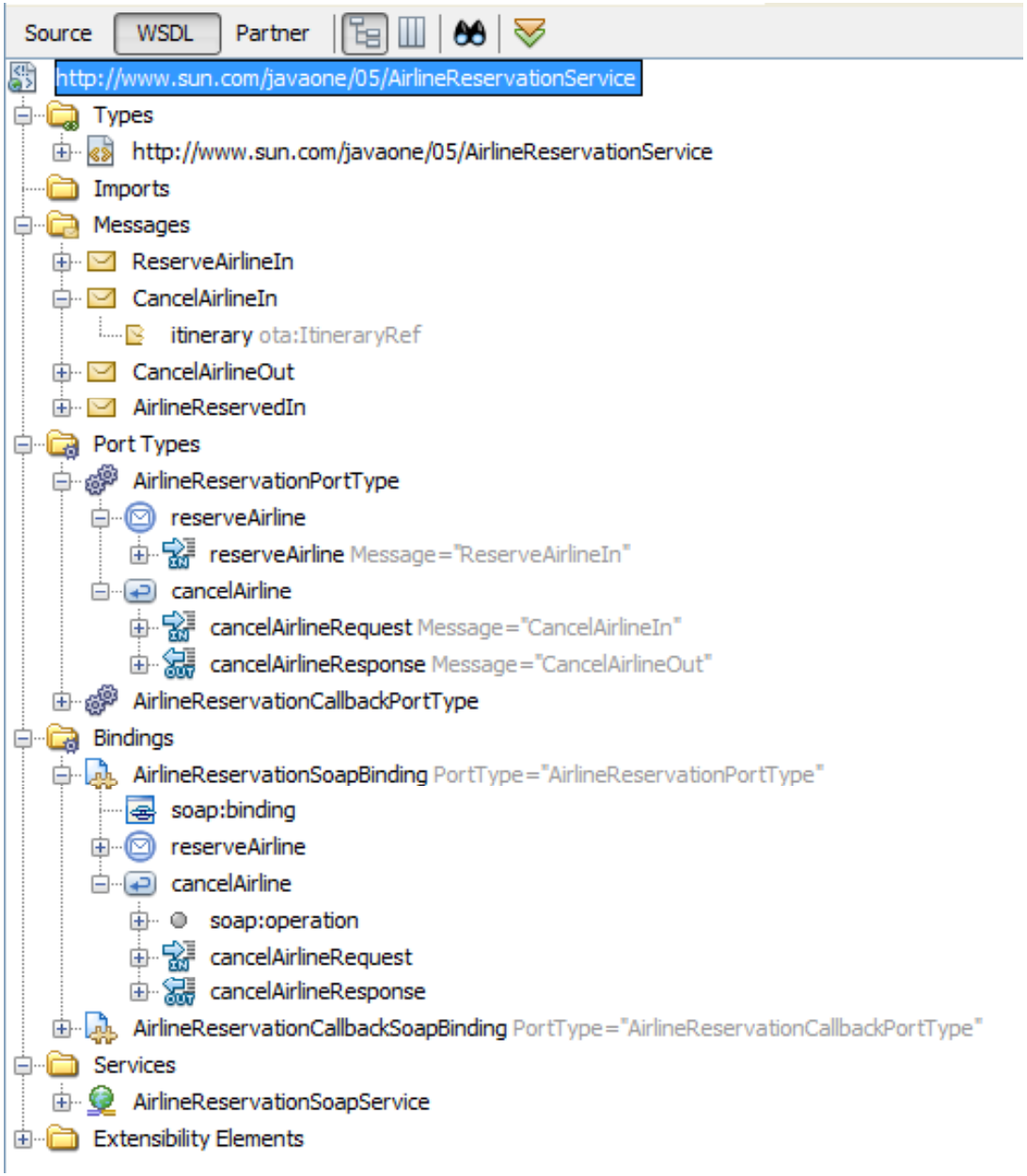
```
<binding name="AirlineReservationSoapBinding“
        type="tns:AirlineReservationPortType">
  <soap:binding style="document"
                transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="cancelAirline">
 <soap:operation soapAction=http://www.sun.com/AirlineService/cancelAirline
   style="document"/>
   <input><soap:body use="literal"/></input>
   <output><soap:body use="literal"/></output>
  </operation>
</binding>

<service name="AirlineService">
   <port name="AirlineReservationSoapHttpPort"
       binding="tns:AirlineReservationSoapBinding">
       <soap:address
     location="http://localhost:8080/webservice/AirlineReservationService"/>
   </port>
</service>

</definitions>
```

4. Concepts de bases et principes de fonctionnement des services

- **Exemple d'un document WSDL ouvert avec Netbeans**



4. Concepts de bases et principes de fonctionnement des services

- **Structure d'un document WSDL :** Les caractères ajoutés aux éléments, attributs sont comme suit:
  - "?" (0 ou 1),
  - "*" (0 ou plus),
  - "+" (1 ou plus).

```
<wsdl:definitions name="nmtoken"? targetNamespace="uri"?>
   <import namespace="uri" location="uri"/>*

   <wsdl:documentation .... /> ?

   <wsdl:types> ?
      <wsdl:documentation .... />?
      <xsd:schema .... />*
      <-- extensibility element --> *
   </wsdl:types>

   <wsdl:message name="nmtoken"> *
      <wsdl:documentation .... />?
      <part name="nmtoken" element="qname"? type="qname"?/> *
   </wsdl:message>
```

```
<wsdl:portType name="nmtoken">*
      <wsdl:documentation .... />?
      <wsdl:operation name="nmtoken">*
        <wsdl:documentation .... /> ?
        <wsdl:input name="nmtoken"? message="qname">?
          <wsdl:documentation .... /> ?
        </wsdl:input>
        <wsdl:output name="nmtoken"? message="qname">?
          <wsdl:documentation .... /> ?
        </wsdl:output>
        <wsdl:fault name="nmtoken" message="qname"> *
          <wsdl:documentation .... /> ?
        </wsdl:fault>
      </wsdl:operation>
  </wsdl:portType>
```

4. Concepts de bases et principes de fonctionnement des services

```
<wsdl:binding name="nmtoken" type="qname">*
      <wsdl:documentation .... />?
      <-- extensibility element --> *
      <wsdl:operation name="nmtoken">*
        <wsdl:documentation .... /> ?
        <-- extensibility element --> *
        <wsdl:input> ?
            <wsdl:documentation .... /> ?
            <-- extensibility element -->
        </wsdl:input>
        <wsdl:output> ?
            <wsdl:documentation .... /> ?
            <-- extensibility element --> *
        </wsdl:output>
        <wsdl:fault name="nmtoken"> *
            <wsdl:documentation .... /> ?
            <-- extensibility element --> *
        </wsdl:fault>
      </wsdl:operation>
  </wsdl:binding>
```

4. Concepts de bases et principes de fonctionnement des services

```
<wsdl:service name="nmtoken"> *
      <wsdl:documentation .... />?
      <wsdl:port name="nmtoken" binding="qname"> *
        <wsdl:documentation .... /> ?
        <-- extensibility element -->
      </wsdl:port>
      <-- extensibility element -->
  </wsdl:service>

  <-- extensibility element --> *

</wsdl:definitions>
```

❑ Simple Object Access Protocol (SOAP)

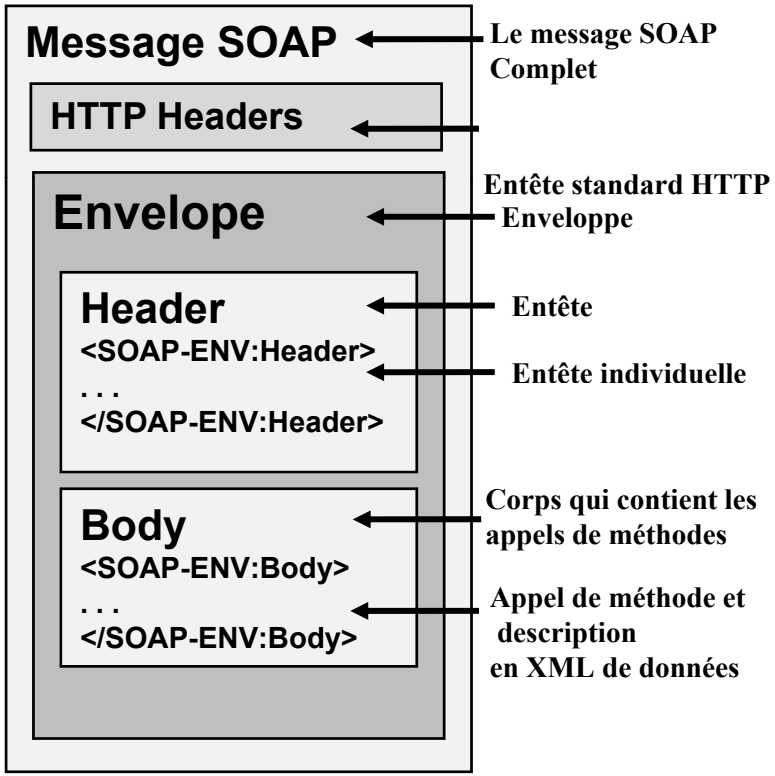
- SOAP a été initialement défini par Microsoft et IBM, mais il est devenu une recommandation du W3C.
- SOAP est un protocole de la famille XML servant à **l'échange d'informations** dans un environnement distribué et décentralisé,
- **Indépendant** de toute plate-forme.
- La technologie la plus importante des services Web

❑ SOAP peut être utilisé sur **n'importe quel protocole** de transport tel que TCP, HTTP, SMTP

❑ Format de message SOAP

Les messages SOAP sont des documents XML qui contiennent trois éléments composant un message:

- **Enveloppe** : SOAP Envelope
- **Entête** : SOAP Header (optionnel)
- **Corps Du Message:** SOAP Body



❑ Enveloppe : SOAP Envelope

- Une enveloppe contient une description du contenu d'un message (mais pas le message).
- Elle définit un Framework global pour exprimer ce qui est dans un message et qui doit le traiter.
- Elle définit l'ensemble de règles de codage pour exprimer des instances de types de données définis par l'application
- L'élément racine d'un message SOAP.
- L'élément **Envelope** contient un élément **Header** facultatif suivi d'un élément **Body** obligatoire.

❑ SOAP Envelope Code

```
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Header> <!-- optional -->
        <!-- header blocks go here... -->
    </soap:Header>
    <soap:Body>
        <!– Message Content... -->
    </soap:Body>
</soap:Envelope>
```

❑ SOAP Header (facultatif)

- L'élément **Header** est un conteneur générique pour inclure des fonctionnalités supplémentaires de contrôle à un message SOAP de manière décentralisée sans accord préalable entre les parties communicantes.
  - Sécurité,
  - Transactions et
  - Autres attributs de qualité de service
- L'en-tête peut contenir n'importe quel nombre d'éléments de n'importe quel espace de noms.

❑ SOAP Body

- SOAP **Body** (Corps) est un conteneur pour les informations obligatoires destinées à l'ultime destinataire du message.

❑ SOAP in Code: Example *Requête HTTP*

POST /soap HTTP/1.1

Host: localhost

Content-Type: text/xml; charset="utf-8"

Content-Length: length

<?xml version='1.0' ?>

<SOAP: **Envelope** // **Enveloppe du message**

<!-- *Espace de nommage pour l'enveloppe SOAP :-->*

xmlns:SOAP ="http://schemas.xmlsoaporg/soap/envelope/"

<!-- *Le type d'encodage du message (entête) :*

SOAP:**encodingStyle**="http://schemas.xmlsoap.org/soap/encoding/"

<!-- *Espace de nommage pour les types de variables :-->*

xmlns : xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns : xsd= "http://www.w3.org/2001/XMLSchema"

<!-- *Espace de nommage pour l'appel des méthodes : -- >*

xmlns :tns ="http.//soapinterop.org/">

❑ SOAP in Code: Example *Requête HTTP*

< SOAP: **Body**> *<!-- **Corps** qui contient le message lui-même :-->*

<!-- Appel à la fonction **getStudentNumber** *qui retourne le nombre d'étudiants d'une classe -- >*

<getStudentNumberRequest>

<!-- Paramètre de l'opération-- >

<className xsi:type="xsd:string"> **B** </className>

</getStudentNumberRequest>

< SOAP*:* **Body**>

</SOAP: **Envelope**>

❑ SOAP in Code: Example *Réponse HTTP*

Response header:
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: **length**
<?xml version='1.0' ?>

<SOAP: **Envelope**
<!– ajouter les espaces de noms ici -->

< SOAP: **Body**>
<**getStudentNumberResponse**>
<**StudentNumber** xsi:type="xsd:int "> **50** </NombreEtudiant>
</ **getStudentNumberResponse**>
< /SOAP: **Body**>
</SOAP: **Envelope>**

❑ Universal Description Discovery and Integration Registry (UDDI):

Specification

- ❑ Le projet UDDI a commencé en 10/2000, par:
  - ➢ Ariba, IBM, Microsoft +260 autres sociétés
- ❑ Une version 2 a été mise au point en 2002.
  - ➢ La version 3 a été mise au point en 2003,
  - ➢ UDDI v3 est devenu un standard de OASIS en février 2005.

Objectifs

- ❑ annuaire mondial d'entreprises pour permettre d'automatiser les communications entre fournisseurs, clients, etc.
- ❑ UDDI est un annuaire de services fondé sur XML et plus particulièrement destiné aux services Web.
- ❑ Il fournit un moyen accessible au public pour **stocker** et **récupérer** des informations sur les **interfaces de services Web.**
- ❑ UDDI définit un registre des services web sous un format XML
- ❑ Ce registre peut être public, privé ou partagé.

❑ Structure de UDDI

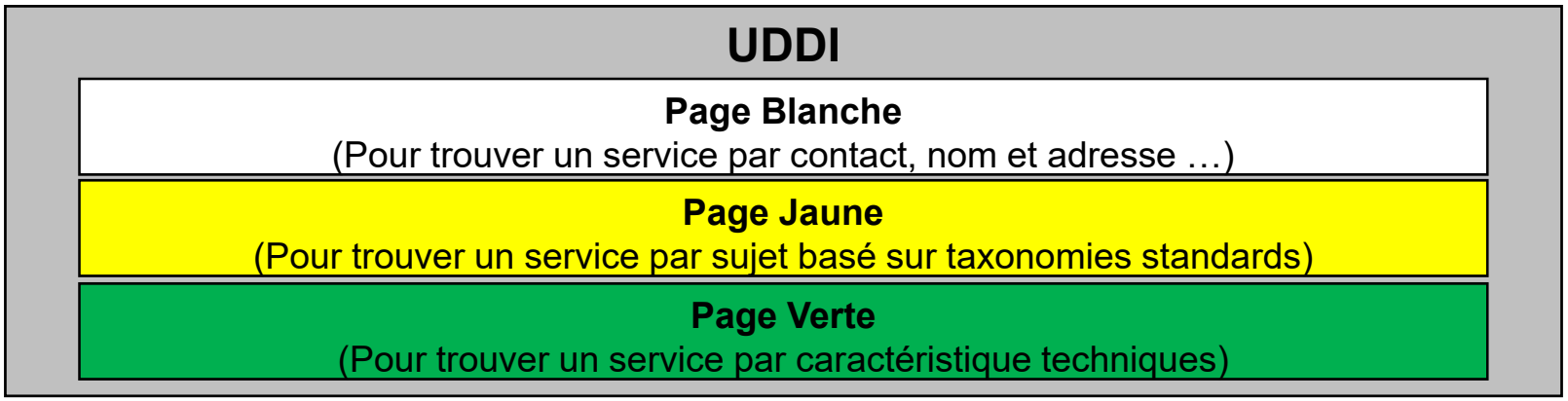
❖ UDDI sont souvent divisées en trois catégories d'information

➢ **Page blanche :** Un nom d'entreprise, information de contact (adresse, numéro tel, fax, site Web,…) et système de numération universel de données (DUNS), ou autre nom d'identifiant.

➢ **Page jaune :** Catégorie d'affaires, endroit, et produits.

• y compris de diverses taxonomies de catégorisation pour l'endroit géographique, type d'industrie, identification d'affaires…

➢ **Page verte :** Les informations techniques sur les services, tels que la façon d'agir avec eux: des définitions de **processus métier;** et l'URL du WSDL de service.



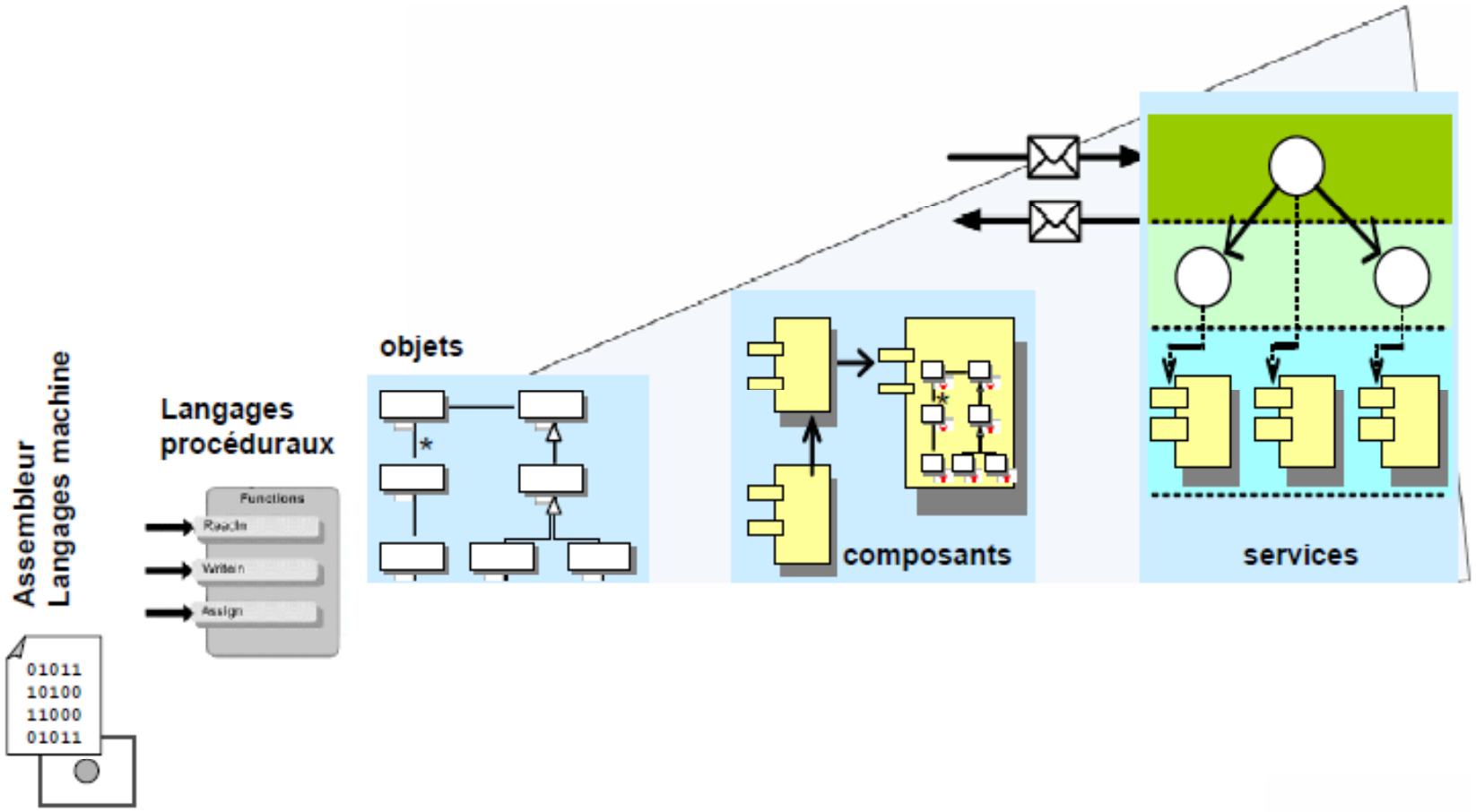
Conclusion

SOA est une évolution des plate-forme passées, tout en préservant les caractéristiques réussies des architectures traditionnelles

- **Contractualisation des services**
  - Design by Contract
- **Découplage Interface/Implémentation**, interopérabilité, transparence des communications, …
  - Middlewares à la CORBA
- **Orchestration des services**
  - Travaux autour des workflows, langages de coordination

SOA est une évolution plutôt qu'une révolution

Conclusion



Niveaux d'abstraction grandissant

Liens utiles

Voir des liens intéressants :

https://docs.oracle.com/cd/E19182-01/821-0539/cnfg_bpel-create-project_t/index.html

https://docs.oracle.com/cd/E19182-01/821-0539/6nlj8ms62/index.html

Spécification BPEL v2:
http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

Schema UDDI v3:
http://www.uddi.org/schema/uddi_v3.xsd

Questions ??