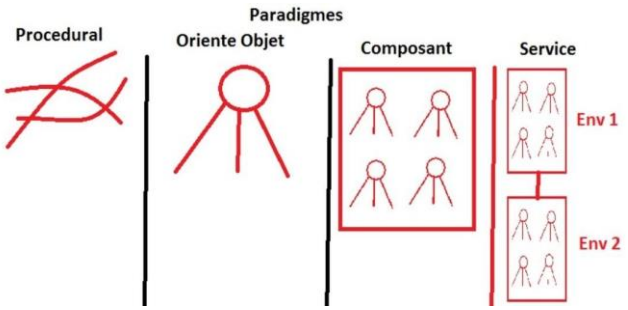
# 1. Introduction aux services web

B2C : utilisateur , Application ( unidirectionnel) ➔ interaction entre une app et acteur humain

B2B : Application , Application (Bidirectionnel) comm entre #ents apps

-**Paradigme** (namouthej )est employé pour exprimer la façon dont un système a été conçu et pensé dans ses grandes lignes.

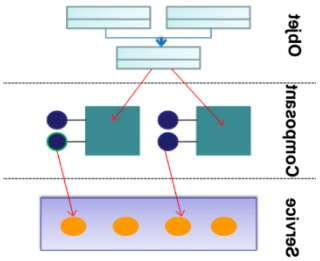
-Le Niveau d'abstraction grandissant avec l'évolution des paradigme




Le paradigme procédural {liste des taches et operation }➔ P objet { encapsulation , heritage , polymorphisme , donné } ➔ P composant {externaliser le code d'une app afin de le rendre réutilisable dans d'autre app } => P service

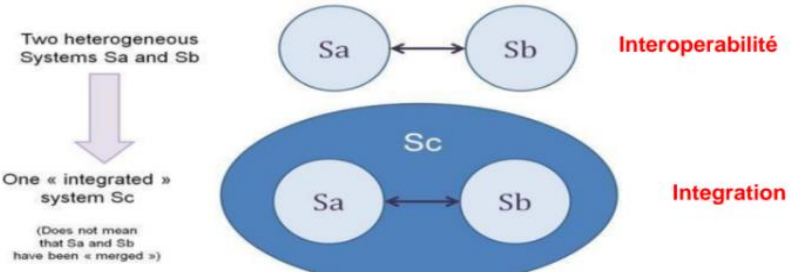
**Le paradigme service** permet de:

- réduire le couplage
- améliorer la réutilisation
- augmenter l'abstraction.

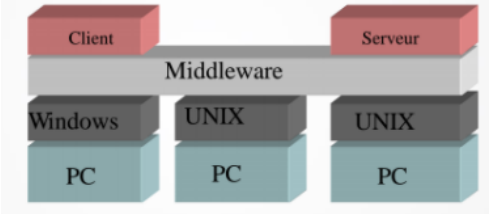



## -L'**interopérabilité** ou **interfonctionnement en informatique**

L'interopérabilité signifie que deux (ou plus) systèmes fonctionnent ensemble sans changement, même s'ils n'ont pas nécessairement été conçus pour fonctionner ensemble.( malgré que les 2 app ne sont pas programmé de fonctionné ensemble )




**Middleware(intergiciel) :** Un intermédiaire de communication entre des applications complexes et distribuées




**Rôles de base d'un middleware:**

- Résoudre l'interopérabilité : Unifier l'accès à des machines distantes
- Résoudre l'hétérogénéité : Etre indépendant des SE et du lang de prog des app

**Service web :**

Web service is a software system designed to support interoperable machine-to-machine interaction over a network

• Les services Web interagissent à travers l'échanges de messages

• Il existe deux grandes familles de services web:

– Les services web étendus (SOAP/WSDL)

– Les services web REST

## Présentation SOA :

•"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 lâche pour exécuter l'application désirée".

Objectifs :

- ✓ Décomposer une fonctionnalités en sous ensemble de fonctions basiques ( services)
- ✓ Décrire finement le schéma d'interaction entre ces services

### Caractéristiques d'un service :

• Contrat standardisé : L'ensemble des services d'un même Système Technique sont exposés au travers de contrats respectant les mêmes règles de standardisation.

• Couplage lâche : Le contrat d'un service doit imposer un couplage lâche de ses clients.

• Abstraction : Le contrat d'un service ne doit contenir que les informations essentielles à son invocation. Un service est vu comme une boîte noire.

• Réutilisabilité : Un service doit être positionné comme une ressource réutilisable.

• Autonomie : Un service ne doit être dépendant d'aucun contexte ou service externe

• Stateless (sans état) : Un service doit minimiser la consommation de ressources en déléguant la gestion des informations d'état quand cela est nécessaire.

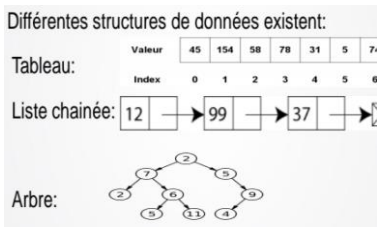
• Découvrabilité : Un service est complété par un ensemble de métas données de communication au travers desquelles il peut être découvert et interprété de façon effective.

• Composabilité : Un service doit être conçu de façon à participer à des compositions de services.

## 2. XML : Extensible Markup Language :

**Structure des donnés** : baisser la complexité d'une app et diminuer les erreurs

**Structure physique** **structure logique**



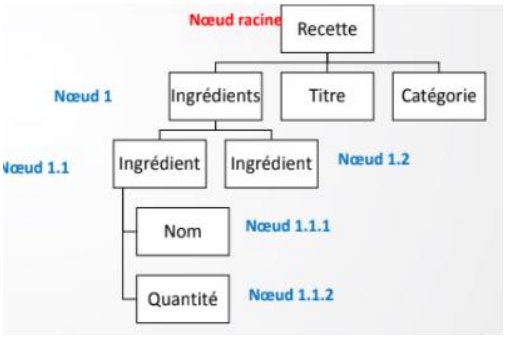
**Presentation de Xml**

Langage de balises , Archiver des données , Lisible ,

Extensible : supporte les évolutions applicatives.

Mise en forme avec des feuilles de style.

Un méta langage permettant la définition de langages adaptés à des besoins variés.




- L'arborescence d'un document XML est la structure hiérarchique des nœuds.

-Un document XML est composé de plusieurs nœuds

**Structure d'un document XML :**

Un document XML comporte : une prologue , l'arbre des éléments ,des commentaires.

•La prologue : (facultative, mais fortement conseillée)

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

est une instruction de traitement destinée à l'application chargée du traitement du document XML. Elle décrit:

- la version du langage XML

- le codage des caractères (par défaut UTF-8)

- La dépendance à des document extérieurs

•Les nœuds XML : ( 3 types)

1) Les éléments : s'ouvre et se ferme par une balise <categorie>Dessert</categorie>
2) Les attributs : se trouve dans la balise ouvrante d'un élément
   L'attribut n'est pas repris dans la balise fermante
   Un élément peut contenir plusieurs attributs
   Un même attribut ne peut pas être présent qu'une seule fois dans un élément
   L'ordre des attributs n'a pas d'importance au sein d'un élément
   La valeur de l'attribut est indiquée entre guillemets
   <quantite unite ="g" >100</quantite>

3) Les entités : une chaîne de caractère commençant par & et se terminant par ;
   ( &entite; )

| Caractère | Entité |
|---|---|
| & | &amp; |
| < | &lt; |
| > | &gt; |
| " | &quot; |
| ' | &apos; |

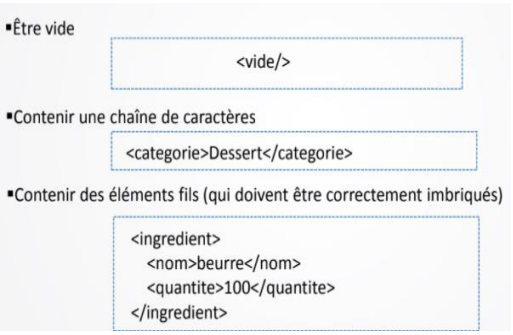
<message>salaire < 1000</message>

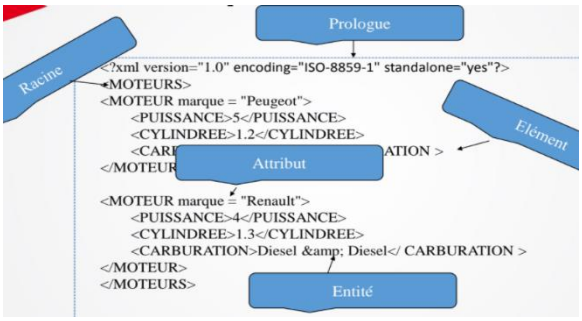
Commentaire < !—commentaire--!>

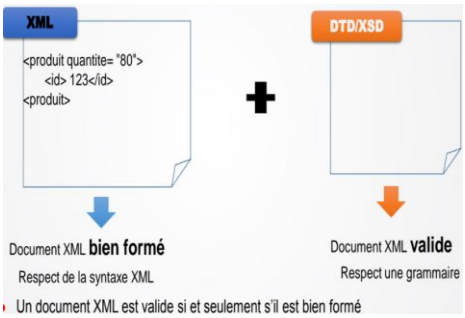
**Les règles syntaxiques :**

-Un élément peut: Être vide : <vide/>

- XML est sensible à la casse : <Categorie>incorrect</categorie> ||| Document XML valide = Respect une grammaire XSD




**Grammaire :**

| DTD (Document Type Definition) | XSD (XML Schema Definition) |
|---|---|
| DTD est une grammaire qui permet de définir une structure type de document XML.<br>- Nouveau langage :Syntaxe particulière<br>- Types de données limités<br>- Aucune contrainte sur le contenu des éléments et attributs | est un langage de description de format de document XML permettant de définir la structure et le type de contenu d'un document XML qui permet de vérifier sa validité<br>- Langage issu de XML Syntaxe XML<br>- Types de données plus riches ( int , float, ..)<br>- Définition des contraintes sur le contenu des éléments/attributs<br>- Extensible |

- XML est un langage de structuration de données
- Un document XML est structuré à l'aide d'éléments et d'attributs
- Un document XML doit respecter les règles syntaxiques pour qu'il soit bien formé

## 3/ XSD (XML Schema Definition) :

**Structure d'un schéma XML :**


Un document schema XML est défini dans un fichier dont l'extension est *.xsd

Comme tout document XML, un schéma XML commence par la prologue XML et a un élément racine

• L'élément <xs:schema> est la racine de tout document Schema XML

**Fichier XSD**

```
<?xml version="1.0" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
        ...
</xs:schema>
```

**Déclaration des éléments :**

**<xs:element name="theName" type="theType" />**

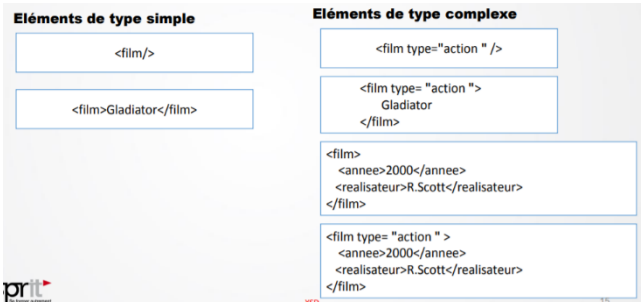
The type : peut etre Par defaut ou Fixed

**Déclaration des attributs :**

**<xs:attribute name="theName" type="theType" use="required" /> ( or optional)**

Les attributs sont de types simples

Les éléments sont de: o Types simples ou Types complexes




**Les types simples: restriction**

Les restrictions sur les types simples permettent de dériver de nouveaux types à partir de types existants
Les restrictions passent par l'utilisation des facettes.

Une facette permet de définir des contraintes sur le nouveau type à créer

La création de nouveaux types simples est réalisée avec la balise <xs:simpleType>

```
<xs:simpleType name="newType" >
        ...
</xs:simpleType>
```

La restriction est exprimée avec la balise <xs:restriction>

```
<xs:simpleType name="newType" >
        <xs:restriction base="type" >
            ...
        </xs:restriction>
</xs:simpleType>
```

***Les principals Facettes**

Facette length

```
<xs:element name=" password" type ="passwordType" />
<xs:simpleType name ="passwordType" >
    <xs:restriction base="xs:string">
      <xs:length value="8"/>
    </xs:restriction>
</xs:simpleType>
```

Facette minlength , maxlength

```
<xs:element name="password"  type="passwordType" />
<xs:simpleType name =" passwordType" >
   <xs:restriction base="xs:string">
     <xs:minLength value="5"/>
     <xs:maxLength value="8"/>
   </xs:restriction>
</xs:simpleType>
```

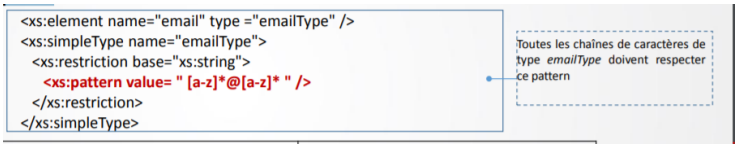
Facette minLInclusive, minExclusive, maxInclusive, maxExclusive

```
<xs:element name=" age" type ="ageType" />
<xs:simpleType name="ageType">
  <xs:restriction base="xs:int">
    <xs:minInclusive value="1" />
    <xs:maxInclusive value="100" />
  </xs:restriction>
</xs:simpleType>
```

♣ Facette enumeration

```
<xs:element name=" age" type ="ageType" />
<xs:simpleType name="ageType">
  <xs:restriction base="xs:int">
    <xs:minInclusive value="1" />
    <xs:maxInclusive value="100" />
  </xs:restriction>
</xs:simpleType>
```

♣ Facette pattern



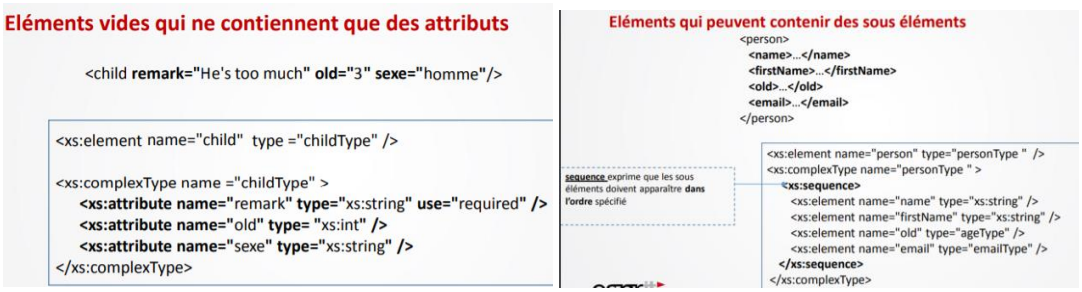

[a-z]* => 0 ou plusieurs lettre(s)

([a-z][A-Z])+ 1 => ou plusieurs paires de lettres min et maj sToP, Stop,STOP,stop

[a-zA-Z0-9]{8} =>8 caractères (chiffre, lettre min, lettre maj)

**Les types complexes : 4 combinaisons**

La création d'un éléments de type complexe est réalisée avec la balise <xs:complexType>




♣ XSD permet d'exprimer trois sortes d'indicateurs d'ordre:

• sequence : exprime que les sous éléments doivent apparaître dans l'ordre spécifié

• all : all tous les sous éléments peuvent apparaître dans n'importe quel ordre : <xs :all>

• choice : choice exprime qu'un seul élément parmi tous les sous éléments peut apparaître : <xs :choice>

Indicateurs d'occurrence

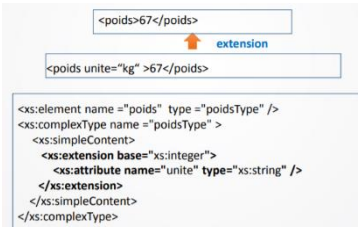
♣ maxOccurs : précise le nombre d'occurrence maximum

♣ minOccurs : précise le nombre d'occurrence minimum

♣ Si les valeurs de maxOccurs ou minOccurs ne sont pas explicitement précisées, la valeur par défaut est de 1

♣ Pour définir une valeur infinie, fixer la valeur à unbounded

**L'héritage en XSD : <xs:extension>**




Héritage d'un élément simple : <xs:simpleContent>

Héritage d'un élément Complexe : <xs:complexContent>

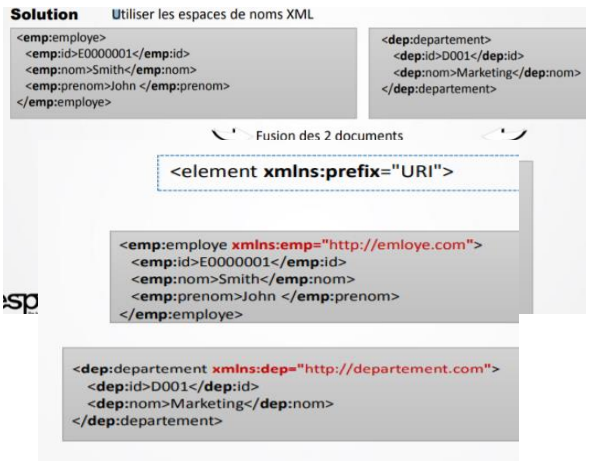
Balise content type avant extension

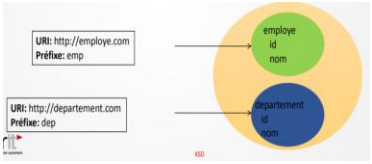
**Déclaration des espaces de noms :**

-Un espace de nom associe un préfixe à un URI

-L'URI (Uniform Resource Identifier) sert à identifier un espace de noms

Le préfixe est une chaîne utilisée pour référencer l'espace de nom dans un fichier XML.






-La déclaration de l'espace de noms se fait au moyen de l'attribut xmlns

- XSD est un langage permettant la définition de la structure d'un document XML
- XSD offre une richesse de types : Types simples, types complexes , Restriction, extension
- **L'association d'un fichier XML à un fichier XSD passe par l'utilisation des espaces de noms**.

| | |
|---|---|
| **xmlns:xs** | ▪espace de nommage des éléments et types XSD |
| **xmlns:dep** | ▪espace de nommage des nouveaux types définis par le programmeur |
| **targetNamespace** | ▪espace de nommage du schema XSD cible<br>▪C'est l'espace de noms qui sera référé par le fichier XML |

Espace de nom pour XSL

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
...
</xsl:stylesheet >
```

Espace de nom pour XSD

```
<?xml version="1.0" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
...
</xs:schema>
```

## 4/JAX-B : Java Architecture for XML Binding

-JAX-B est un API Java permettant la gestion de données XML.

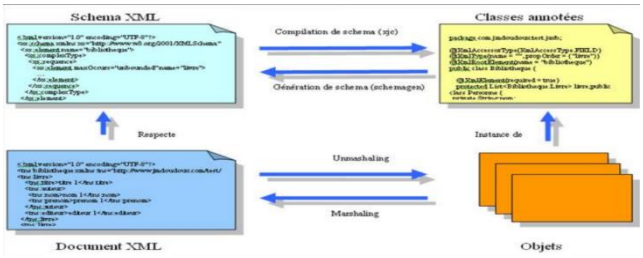
-JAX-B permet plus particulièrement l'utilisation du "Data Binding"

Le Data Binding est une technologie permettant d'automatiser la transformation des fichiers XML en objets Java et inversement. ( sérialisation et desérialisation )

**Fonctionnement :**

Classes java <-> Schéma XML

Instances de classes java <-> XML




**Cas d'usages :** JAX-B est utilisé par :

**JAX-WS :** utilisent JAX-B pour la conversion de données entre les classes Java et, WSDL et SOAP.

**JAX-RS :** Les services web RESTful utilisent JAX-B pour la transformation des données échangées en XML

Types de données définis par l'utilisateur:

| Types Java | Types XSD |
|---|---|
| Javabean | <xsd:complexType> |
| Variable de Javabean | Un élément sous <xsd:complexType> |
| Variable de Javabean de type List | Un element sous <xsd:complexType> avec l'attribut maxOccurs="unbounded". |

L'outil xjc permet de générer les classes à partir d'un schéma XML . Exemple: xjc personne.xsd

L'outil schemagen permet de générer le schéma XML à partir d'une classe Ex : schemagen Personne.java

**Sérialisation / Marshalling** : Génération d'un document XML à partir d'une instance Java

1) Ajouter les annotations nécessaires à la classe java

2) Utiliser la classe Marshaller de l'API JAX-B pour générer le document XML à partir des objets déjà créés.

**Les annotations** JAX-B utilisées dans les classes java permettent la génération et la personnalisation:

- Des schémas XSD générés
- Des documents XML générés

Les annotations JAX-B sont définies dans le package **javax.xml.bind.annotation.**

| Annotation | Description |
|---|---|
| XmlRootElement | Spécifier la racine du document XML. |
| XmlElement | Convertir une propriété de la classe en un élément dans le document XML. |
| XmlAttribute | Convertir une propriété de la classe en un attribut dans le document XML. |
| XmlTransient | Retire des éléments pris en compte pour la création des schémas et des documents XML. |

| Annotation | Description |
|---|---|
| XmlType | Permet de fixer l'ordre dans lequel les champs de cette classe doivent être enregistrés dans le document XML. |
| XmlAccessorOrder | Contrôler l'ordre des attributs et des propriétés dans la classe. |
| XmlSchema | Associer un espace de noms à un paquetage. |

## 5) JAX-RS :

-REST est l'acronyme de REpresentational State Transfert , est une alternative à SOAP

- REST est un **style d'architecture** inspiré de l'architecture du Web pour construire des services web

- REST n'est pas: un format ni un protocole ni un standard

-il est le dev du serveur web

-Bien que REST ne soit pas un standard, il utilise des standards: http , URL , XML/HTML

Il est léger et simple : Les messages sont courts, faciles à décoder

Il est stateless : Consommation de mémoire inférieure

Rest est auto-descriptif et peut etre gérer en cache

**Principes de REST :**

-Ressources (Identifiant) **:** Identifié par une URI **:http://localhost :8080:/../books**

-Méthodes (Verbes) : pour manipuler la ressource

- Méthodes HTTP : GET, POST, PUT and DELETE
- Une ressource quelconque peut subir quatre opérations de base désignées par CRUD
- REST s'appuie sur le protocole HTTP pour exprimer les opérations via les méthodes HTTP

Create <-> POST , Read <->GET , Update <-> PUT , Delete <-> DELETE

-Représentation : donne une vue sur l'état de la ressource et informations transférées entre le client et le serveur

Exemples : XML, Text, JSON, ...

- ✓ Fournir les données suivant une représentation pour:
  - le client (GET): format de sortie

- le serveur (PUT et POST): format d'entrée

**WADL : Web Application Description Language**

-est un langage de description XML de services de type REST

- l'objectif est de pouvoir générer automatiquement les APIs clientes d'accès aux services REST

**JAX-RS: Java API for RESTful Web Services**

-Spécification décrivant la mise en œuvre et la consommation des services web REST

-JAX-RS est basé sur les annotations

@Produces spécifie le type de la réponse du service

@Path définit le chemin de la ressource

@Consumes spécifie le type accepté en entré du service

- REST est une alternative aux services web étendus (SOAP)
- REST se base sur le protocole HTTP
- JAX-RS est l'API java permettant de développer et consommer des services web REST

## 6) JWT :JSON Web Token

- (JWT) est un standard, Définit une solution, compacte et autonome,

Jwt contient tous les info requise sur l user

-Permet de transmettre de manière sécurisée des informations entre les applications en tant qu'objet structuré au format JSON.

-JWT est constitué de trois parties séparées par un point « . » :

o Header

o Payload

o Signature

- La forme d'un JWT est donc : xxx.yyy.zzz

### A. Header

**L'en-tête se compose généralement de deux parties:**

o Le type du jeton, qui est JWT,

o L'algorithme de hachage utilisé, tel que **: HMAC** (HS512, HS256, HS384) ou **RSA :methode de cryptage**

• La structure du Header est un objet JSON ayant la forme la suivante :

{ "alg": "HS256",

"typ": "JWT" }

• Cet objet JSON est ensuite encodé en Base64URL.

### B. Payload

- C'est la deuxième partie du jeton,

- Elle contient les claims (mtatalibet)suivants:

• iss (issuer : Origine du token),

• exp (heure d'expiration),

• sub (sujet),

• aud (public cible),

• nbf (Not Before : A ne pas utiliser avant cette date) ,

• iat ( issued at : date de création du token),

• jti ( JWT ID identifiant unique du JWT).

### C. Signature

- C'est la dernière partie du jeton,

-Elle est utilisée pour:

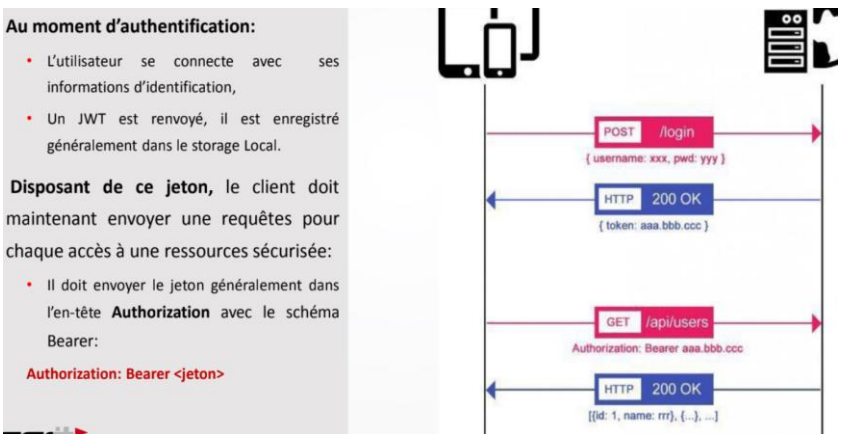
o vérifier que l'expéditeur du JWT est celui qu'il prétend être.bch ythabet eli user howa nafsu

o et pour s'assurer que le message n'a pas été modifié en cours de route. Bch yt1aked eli el msg matbadlch par un hacker MITM

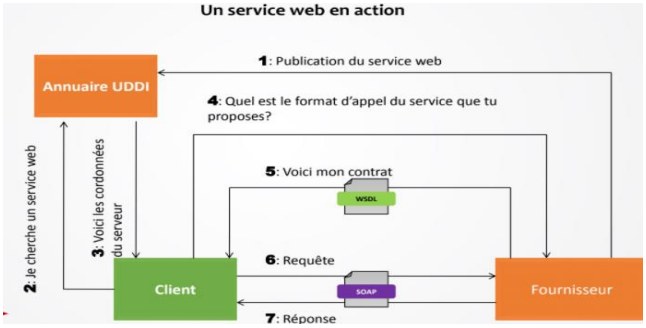
Si vous voulez utiliser l'algorithme HMAC SHA256, la signature sera créée de la façon

suivante:

 HMACSHA256( base64UrlEncode(header) + "." + base64UrlEncode(payload), secret)



## 7) WDSL : Web Service Description Language




Un langage de description des services web basé sur XML.

trois composants sont nécessaires :

*Un langage pour décrire le service web: WSDL

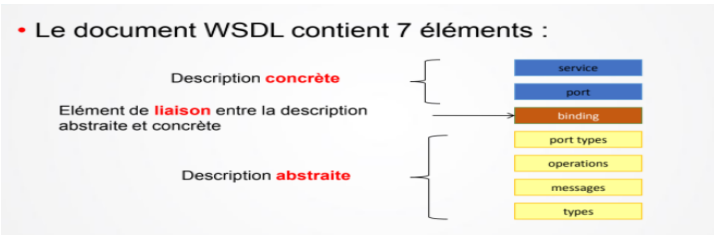
* Un protocole de communication pour écrire les messages échangés entre le consommateur et le fournisseur: SOAP

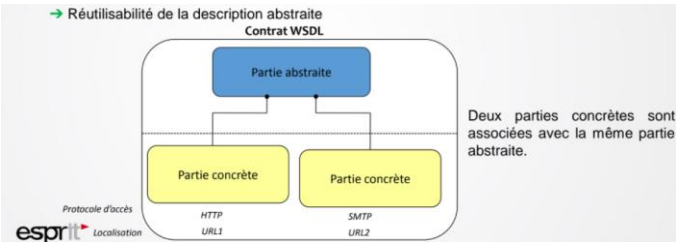
* Un protocole de transport afin de faire circuler les informations sur Internet

Un document WSDL se structure en 2 parties :

**Une description concrète :** Définition du protocole d'accès et de l'URI à partir de laquelle on peut accéder au service web

**Une description abstraite :** nom des opérations, paramètres d'entrée, de sortie, structure des messages

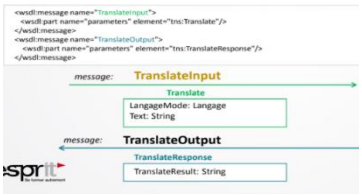


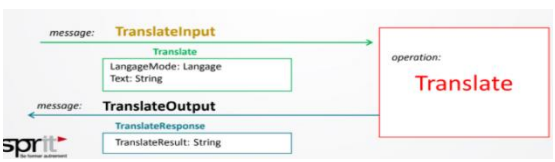

### Partie abstraite :

**Types** : Contient la définition des types de données à transmettre, Exprimé en XSD

**Messages :** Contient la description des messages échangés avec le service web,Paramètres d'entrée des opérations,Paramètres de sortie.



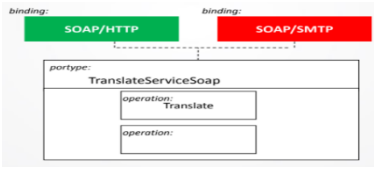

**Operations :** Une opération est comparable à une méthode en Java ,identifiée par un nom,Contient ou non une ou plusieurs entrée(s),Contient ou non une ou plusieurs sorties



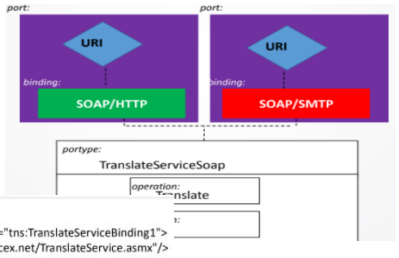
**Portype :** Un portype est comparable à une interface en Java,identifiée par un nom,Contient un ensemble d'opérations


## Une description concrète

**Binding :** Binding: permet de définir le format du msg échangé + le protocole de transportmême portype.

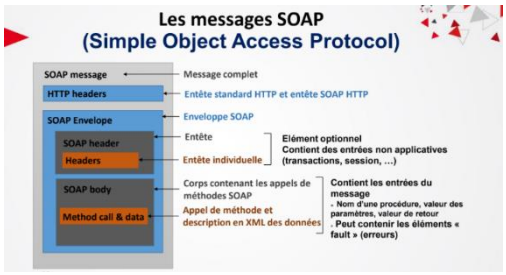



**Port :** Permet de spécifier une adresse pour un binding donné




**service :** Contient une collection de ports

```
<wsdl:service name="TranslateService">
  <wsdl:port name="TranslateServiceSoap1" binding="tns:TranslateServiceBinding1">
      <soap:address location="http://www.webservicex.net/TranslateService.asmx"/>
  </wsdl:port>
  <wsdl:port name="TranslateServiceSoap2" binding="tns:TranslateServiceBinding2">
      <soap:address location="mailto:subscribe@translate.com"/>
  </wsdl:port>
</wsdl:service>
```



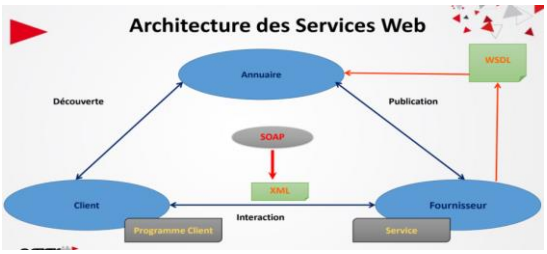

L'architecture des web services étendus repose essentiellement sur les technologies suivantes:

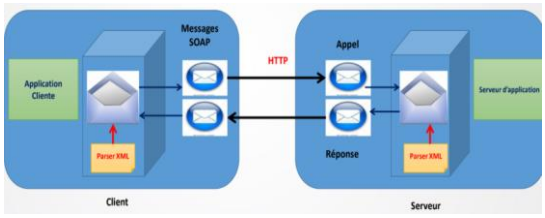
-SOAP-Simple Object Acces Protocol: Protocole pour la communication entre Web Services.

-WSDL-Web Service Description Langage: langage de description de l'interface du Web Service

-UDDI-Universal Description, Discovery and Integration: Annuaire pour le référencement du Web Service.

## 8) JAX-WS (Java API for XML-based Web Services) :




-Est un modèle standard de programmation des services web étendus en Java.

-Permet de développer des services web et leurs clients en Java.

SOAP-Simple Object Acces Protocol: Protocole pour la communication entre Web Services.

*WSDL-Web Service Description Langage: langage de description de l'interface du Web Service

UDDI-Universal Description, Discovery and Integration: Annuaire pour le référencement du

Web Service.

*JAX-WS est une spécification supportée par plusieurs plateformes Java

comme: AXIS2,CXF,Glassfish

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

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

Approche Bottom/Up : génération automatique du document WSDL à partir des classes JAVA(POJO) annotation @WebService

Approche Top/Down:génération des classes JAVA à partir d'un document WSDL
wsimport:démarrer le développement du service à partir d'un document WSDL
sun-jaxws.xml: décrire le endpoint en indiquant la classe+url-pattern

| Approche Bottom / Up (à partir d'un POJO) | Approche Top / Down (à partir d'un doc WSDL) |
|---|---|
| -Créer et annoter un POJO.<br>-Compiler, déployer et tester.<br>-Le document WSDL est automatiquement généré.<br>*Ajouter l'annotation @WebService.<br>*Toutes les méthodes public du POJO sont des opérations du Web Service.<br>*La surcharge de méthodes n'est pas supportée.<br>-L'outil wsgen génère des artifacts (JAXB, WSDL) à partir des classes Java annotées via JAX-WS. | -Génération des différentes classes Java (JAXB et squelette du Web Service) en utilisant l'outil « wsimport ».<br>-Compléter le squelette de classe de l'implémentation.<br>-Compiler, déployer et tester.<br>• L'outil wsimport nous permet de générer le squelette du Service Web:<br>✓ Génération des classes Java liées à JAXB.<br>✓ Génération des interfaces WS. |

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

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

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

| | Web-Etendu | REST |
|---|---|---|
| Architecture | Toute une architecture | Style d'architecture |
| Fonctionnalites | Personnalisee | Predefinis |
| Contract Standarise | WSDL: Web Service Desciption Language | WADL: Web Application Description Language |
| Methodes | Methodes Personnalisables | Predefinis: Post, Get, Put... |
| Utiliste du Contrat | Utile , tres outillee | Inutile, peu outillee |
| Format de donnees | XML | Json, XML, Text.... ✓ |
| Protocole de Communication | SOAP: Simple Object Access Protocol | Basee sur HTTP |
| Protocole de Transport | HTTP, SMTP, FTP ✓ | HTTP |
| Souple et facile a utiliser | complexe, difficile a utiliser | simple, facile a utiliser ✓ |
| Securite | Plus Securite ✓ | Moins Securite |

Import: Javax.jws.WebService
@Annotation Obligatoire: @WebService (Se met au dessus de la classe)

Partie Serveur | Partie Client

1.1.1 POJO: Une seule classe
- Methodes publiques
- Contient une partie Metier

Approche: Bottom-Up

1.2 Contract WSDL URL: chemin?WSDL

Commande: WSGEN dans le Jdk

Pas de partie Metier

1.1.2 SOAP-Binding
Separation entre:
- Classe d'implementation (Partie Metier)
- Interface (Nom des methodes, entrees, sorties)

Pointer vers l'interface: EndPointInterface

Approche: Top-Down
Commande: WSImport dans le JDK

Clients:
- SOAP-UI
- Developper notre propre Client

1.3 Squelette de la classe d'implementation
Pas de partie Metier (A completer)

1.4 Client Jax-WS - Importer:
- Messages d'entree et de sorties
- Classes: Portype et le service
- Object-Factory: Gere la creation des objet
- Package Info: Information du package
- Invocation: Service => Portype=> Operation

@Annotations Optionnelles: Annotations de Personnalisation
- Les parametres de @Webservice (targetNameSpace, Name, ServiceName, Port...)
-@WebMethod: Se met audessus de la methode (OprationName) => Par defaut: le nom de la methode
- @WebParam: Se met avant chaque parametre d'entree (Name) => Par defaut: Arg0, Arg1, Arg2... (Liste d'arguments)
- @WebResult: Se met au dessus de le methode (Name) => Par defaut: return