Table des matières

Liste des Figures

Introduction Générale

De nos jours, les entreprises expriment un grand besoin pour échanger des informations et des services. Ceci nécessite des langages communs de communication.
Les efforts d'élaboration de ces langages ont donné lieu a un nouveau domaine de recherche connu sous le nom de « business protocoles ». Une technologie émergeante dans ce domaine a permis de tracer quelques pistes intéressantes pour la communication entre entreprises. Cette technologie est celle de **web services**.

Les **web services** sont un paradigme naissant qui vise à la transposition des architectures ,par composant dans le cadre du Web, d'autres technologies telles que RMI, DCOM et CORBA ont précédemment adopté ce style architectural mais ont généralement échoué en raison de la diversité des plates-formes utilisées dans les organisations et aussi parce que leur usage n'était pas adapté à Internet (problème de passage à travers des FireWalls, etc.).

D'où la lenteur, voire l'absence de réponses sur ce réseau. Les applications réparties fondées sur ces technologies offrent des solutions caractérisées par un couplage fort entre les objets. Les solutions proposées par les services Web, permettent néanmoins un couplage moins fort. De plus, l'utilisation des technologies standards du Web telles HTTP et XML par les services Web facilite le développement d'applications réparties sur Internet, et permet d'avoir des applications très faiblement couplées. L'intégration est sans doute le facteur essentiel qui favorise l'utilisation des services Web.

C'est pourquoi, nous nous somme intéressé a étudie d'une manière peu approfondie, le domaine du web services et ses applications.

Notre travail sera donc répartit en deux chapitres, initié par une recherche bibliographique ou nous apportons dans le premier chapitre les principaux concepts liés aux services web. Le deuxième chapitre est une partie pratique, son objectif est de réaliser une application web services pour la création d'un compte bancaire ainsi que ses transactions.

1. Introduction

Les services web sont des mots tendance, et sont actuellement promus par, entre autres, SUN,ORACLE, HP, Microsoft et IBM. C'est un mot nouveau de concept ancien car il s'agit ni plus ni moins que de déporter le traitement de données d'un poste client, vers un poste serveur sur lequel tourne l'application.

Cette évolution du monde informatique a entraîné le développement de nouveaux paradigmes d'interaction entre applications tels que la SOA. Cette dernière a été mise en avant afin de permettre des interactions entre applications distantes. L'architecture orientée service SOA est une méthode de conception basée sur des standards (SOAP, WSDL,...), permettant de créer une infrastructure informatique intégrée capable de répondre rapidement aux nouveaux besoins d'un utilisateur. Elle fournit les principes et directives permettant de transformer un réseau existant de ressources informatiques hétérogènes, distribuées, complexes et rigides en ressources intégrées, simplifiées et particulièrement souples pouvant être modifiées et combinées afin de mieux satisfaire les objectifs de l'utilisateur. **[1]**

Dans ce chapitre, nous présenterons l'architecture orientée services et sa principale réalisation : les services Web. Nous donnons les définitions des services Web et décrivons brièvement leurs principe standards et technologies. Enfin, nous abordons le concept de composition des services Web en explicitant les méthodes de sélection des services web.

2. L'architecture orientée service SOA

2.1. Définition

Plusieurs définitions sont utilisées pour définir et expliquer l'architecture SOA. Les définitions suivantes illustrent différentes vues de la SOA. Cependant, elles convergent toutes vers un seul sens :

- **Métier :**

« L'architecture orientée service est un ensemble de méthodes techniques, métiers, procédurales, organisationnelles et gouvernementales pour réduire ou éliminer les frustrations avec les technologies d'information, et pour mesurer quantitativement la valeur métier des technologies d'information, pendant la création d 'un environnement métier agile pour un intérêt concurrentiel. » **[2].**

- **Technique :**

« Une architecture SOA est une structure d'intégration de processus métier qui supporte une infrastructure des technologies d'information comme étant des composants et services sécurisés, standardisés et qui peuvent être combinés pour s'adresser aux priorités de changements métiers. » **[3].**

Par conséquence, un SOA (Services Oriented Architecture ; architecture orientée service) repose sur la réorganisation des applications à partir d'un ensemble de services élémentaires. Ces applications (visibles) représentent les services en question s'appuyant fonctionnellement parlant que des interfaces standards (langages SOAP WSDL ou REST pour Representational State Transfer), connue sous le nom de Web Services, couches d'invocation compréhensibles potentiellement par l'ensemble des systèmes en présence, pour peu qu'elles intègrent le module d'interprétation nécessaire. Au sein d'un tel environnement, des services (dits "producteurs") sont ainsi exposés à d'autres services (dits "consommateurs"). **[4]**

Comme la montre la figure ci-dessous, les fournisseurs de services enregistrent et publient leurs services dans un annuaire de services pour respecter l'architecture SOA, les clients de services ou les utilisateurs consultent cet annuaire pour trouver des services qui vérifient les critères qui correspondent à certaines descriptions, si c'est vérifié l'annuaire répond aux clients en donnant les descriptions de la requête avec un contrat d'utilisation. Alors le client fait son choix en s'adressant au fournisseur pour invoquer le service Web. **[5]**

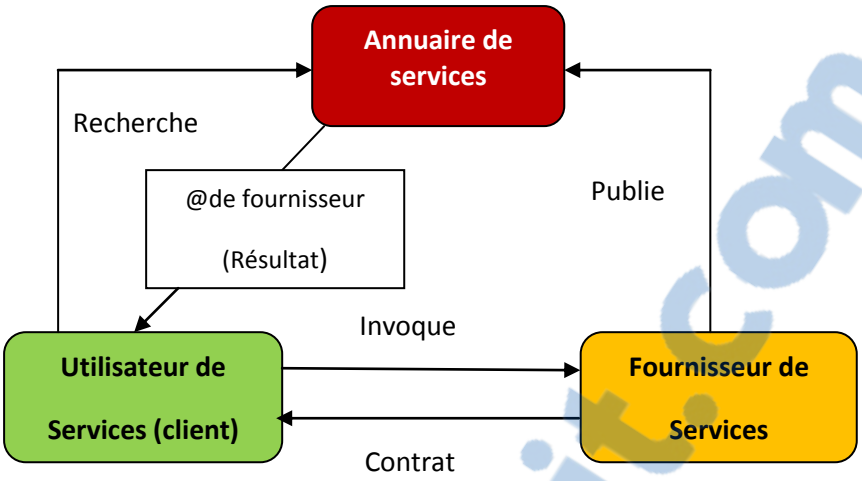


Figure 1.1: architecture orientée services (SOA)

2.2. Les avantages de SOA

Une architecture orientée services permet d'obtenir tous les avantages d'une architecture client-serveur et notamment :

- Une modularité permettant de remplacer facilement un composant (service) par un autre.
- Une réutilisabilité possible des composants (par opposition à un système tout-en-un fait sur mesure pour une organisation).
- De meilleures possibilités d'évolution (il suffit de faire évoluer un service ou d'ajouter un nouveau service).
- Une plus grande tolérance aux pannes.
- Une maintenance facilitée. **[6]**

3. Les services web

Plusieurs d´définitions des services Web ont été mises en avant par différents auteurs.

3.1. Définition 1

Le consortium W3C (World Wide Web Consortium) définit un service Web comme étant: «A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interacact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. » **[7]**:

- Il est identifié par un URI (Uniform Resource Identifier) **[8].**

- Ses interfaces et ses liens peuvent être décrits en XML.

- Sa définition peut être découverte par d'autres services Web.

- Il peut interagir directement avec d'autres services Web à travers le langage XML en utilisant des protocoles Internet standards.

3.2. Définition 2

Les services web représentent un domaine de recherche jeune. IBM donne dans un tutorial la definition suivante des services web: « Web services are a set of emerging standards that enable interoperable integration between heterogeneous IT processes and systems. You can think of them as a new breed of web application that is self-contained and self-describing, and that can provide functionality and interoperation ranging from the basic to the most complicated business and scientific processes. In short, web services hold the promise for providing a common standard mechanism for interoperable integration among disparate systems, and the key to their utility is their standardization. This common mechanism for delivering a "service" makes them ideal for implementing a service-oriented architecture (SOA). »**[9]**

3.3. L'intérêt des services Web

Les services Web fournissent un lien entre applications. Ainsi, des applications utilisant des technologies différentes peuvent envoyer et recevoir des données au travers de protocoles compréhensibles par tout le monde. Les services Web sont normalisés car ils sollicitent les standards XML et généralement HTTP pour transférer des données et ils sont en affinité avec de nombreux autres environnements de développement. Ils sont donc exploitables à distance via n'importe quel type de plate-forme. C'est dans ce contexte qu'un intérêt très spécial a été attribué à la conception des services Web puisqu'ils permettent aux entreprises de délivrer des applications profitables à distance par d'autres entreprises. **[10]**

3.4. Les caractéristiques des services Web

Cette technologie est devenue la base de l'informatique distribuée sur Internet et offre beaucoup d'opportunités au développeur Web qui possède les caractéristiques suivantes: **[11]**

- Il est accessible via le réseau.
- Il dispose d'une interface publique (ensemble d'opérations) décrite en XML.
- **Web based** : les Web services sont basés sur les protocoles et les langages du Web, en particulier HTTP et XML.
- **Self-described**, **self-contained** : le cadre des Web services contient en lui-même toutes les Informations nécessaires à l'utilisation des applications, sous la forme de trois fonctions : trouver, Décrire et exécuter.
- **Modular** : les Web services fonctionnent de manière modulaire et non pas intégrée. Cela signifie qu'au lieu d'intégrer dans une seule application globale toutes les fonctionnalités, on crée (ou on récupère) plusieurs applications spécifiques qu'on fait inter-opérer entre elles, et qui remplissent chacune une de ces fonctionnalités. Une fonctionnalité développée sous forme de Web services peut dorénavant être réutilisée et recombinée à une suite d'autres fonctionnalités pour composer une Nouvelle application.
- Ses descriptions (fonctionnalités, comment l'invoquer et où le trouver ?) sont stockées dans un annuaire.

3.5. Architecture des services web

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 :

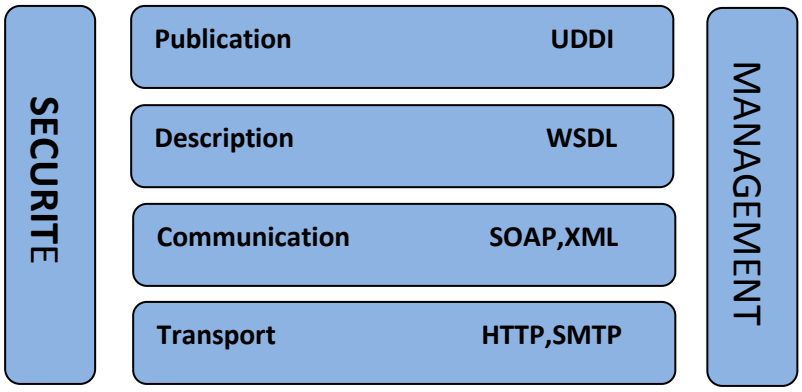


Figure 1.2 : Architecture en Pile des services Web. [13]

Couche transport

Cette couche est responsable du transport des messages XML échangés entre les applications. Actuellement, cette couche inclut HTTP, SMTP (Simple Mail Transfer Protocol), FTP (File Transfer Protocol), Le transport de messages XML-RPC (Remote

Procedure Call) ou SOAP (Simple Object Access Protocol) est assuré par le standard HTTP.

Couche communication

Cette couche est responsable du formatage des données échangées de sorte que les messages peuvent être compris à chaque extrémité. Cette couche utilise des protocoles reposants sur le langage XML, car sa syntaxe unique résout les conflits syntaxiques lors de l'encodage des données. Actuellement, SOAP est le protocole le plus utilisé pour cette couche.

Couche description de service

Cette couche est responsable de la description de l'interface publique du service Web. Le langage utilisé pour décrire un service Web est WSDL (Web Services Description Language) [11] qui est la notation standard basée sur XML pour construire la description de l'interface d'un service.

Couche de découverte de service

Cette couche est chargée de centraliser les services dans un registre commun, et de simplifier les fonctionnalités de recherche et de publication des services Web. Actuellement, la découverte des services est assurée par un annuaire UDDI (Universal Description, Discrovery, and Integration) [11],

3.6. Les principales technologies de développement de service Web

Les technologies utilisées par les services Web sont XML, REST, SOAP, WSDL, UDDI et HTTP qui sont détaillé comme suit :

3.6.1. Communication

- **XML**

XML {Extensible MarkupLanguage, ou Langage Extensible de Balisage) est un standard promulgué par le W3C, l'organisme chargé de standardiser lesévolutions du Web et le langage destiné à succéder à HTML. Comme HTML (HypertextMarkupLanguage) c'est un langage de balisage (markup) : il représente de l'information encadrée par des balises. XML est un métalangage, ce qui veut dire que contrairement à HTML qui possède un ensemble de balises de présentation prédéfinies,

il va permettre d'inventer de nouvelles balises d'isolement d'informations ou d'agrégats élémentaires que peut contenir une page Web. **[12]**

XML a été conçu pour des documents arbitrairement complexes, tout en s'appuyant sur cinq grands principes simples et clairs :

- Lisibilité à la fois par les machines et par les utilisateurs ;
- Définition sas ambiguïté du contenu d'un document ;
- Définition sans ambiguïté de la structure d'un document ;
- Séparation entre documents et relation entre document ;
- Séparation entre structure du document et présentation du document

Le contenu d'un document XML est structuré par une suite d' «éléments», qui sont des blocs de texte encadré par des paires de balises ouvrantes et fermante (<, >), qui sont les « unités de contenu». Ces éléments sont liés entre eux par une hiérarchie, certains éléments apparaissant imbriqués dans d'autres (arborescence). XML et DTD permettent aussi la différenciation réelle du contenu de la structure de document. **[13]**

Voici un exemple de schéma XML définissant le type de document de la bibliographie. Ce schéma est volontairement rudimentaire pour un premier exemple. Il n'est pas très précis sur les contenus de certains éléments. Un exemple plus complet pourrait être donné pour la bibliographie.

Un schéma XML définit, d'une part, l'imbrication des éléments entre eux, ce qui s'apparente aux DTD, et d'autre part, le type des éléments et de leurs attributs. L'information fournie par le schéma est donc plus riche que celle dans le DTD. **[14]**

```
bib>
    <book year= “1994“>
      <title>TCP/IP Illustrated < /title >
     <author>
      <last>Stevens</last>
      <first>W.</first>
     </author>
      <publisher>Addision-Wesly</ publisher >
      <price>65.95</ price >
    </book>
   <book year= “1992“>
      <title>Advanced Programming in the Unix environment <
/title >
      <author><last> Stevens</last><first>W.</first></author>
      <publisher>Addision-Wesly</ publisher >
      <price>65.95</ price >
      <abstract>65.95</ abstract >
   </book>
```

Figure 1. 3 : Exemple Fichier XML de bibliothèque

- **REST**

REST (Representational State Transfer) est une architecture propre aux services Web. Élaborée en l'année 2000 par Roy Fiedling, l'un des créateurs du protocole HTTP, du serveur Apache HTTP et d'autres travaux fondamentaux, REST est une alternative qui permet de bâtir une application pour les systèmes distribués comme le World Wide Web. **[15]**

- **Le Protocol SOAP**

SOAP {simple Object Access Protocol) est un protocole de communication définit à l'origine par Microsoft, puis standardisé par le W3C, avec l’élaboration de IBM. Un protocole pour l’échange d’information dans un environnement repartie, basé sur le standard XML.

SOAP permettant de définir les mécanismes d'échanges d'information entre des clients et des fournisseurs de services Web. Il s'appuie sur n'importe quel protocole de communication (HTTP, SMTP, FTP ...) pour transmettre les messages. **[16]**

SOAP définit un format pour l'envoi du messages. Les messages SOAP sont structuré en un document XML et deux parties obligatoires : l'enveloppe SOAP et le corps SOAP ; et une partie optionnelle : l'en-tête SOAP.

✓ **Structure d'un message SOAP**

Tout d'abord un message SOAP est un document XML qui doit avoir la forme suivante :

- **Envelope** (enveloppe) c'est lui qui contient le message et ses différents sous-blocs. Il s'agit du bloc racine XML. Il 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** (entête) c'est un bloc optionnel qui contient des informations d'en-têtes sur le message. S'il est présent, ce bloc doit toujours se trouver avant le bloc Body à l'intérieur du bloc Envelope.
- **Body** (corps) c'est le 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. SOAP ne définit pas comment est structuré le contenu de ce bloc. Cependant, il définit le bloc Fault qui peut s'y trouver.
- **Fault** (erreur) ce bloc est 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. **[17]**

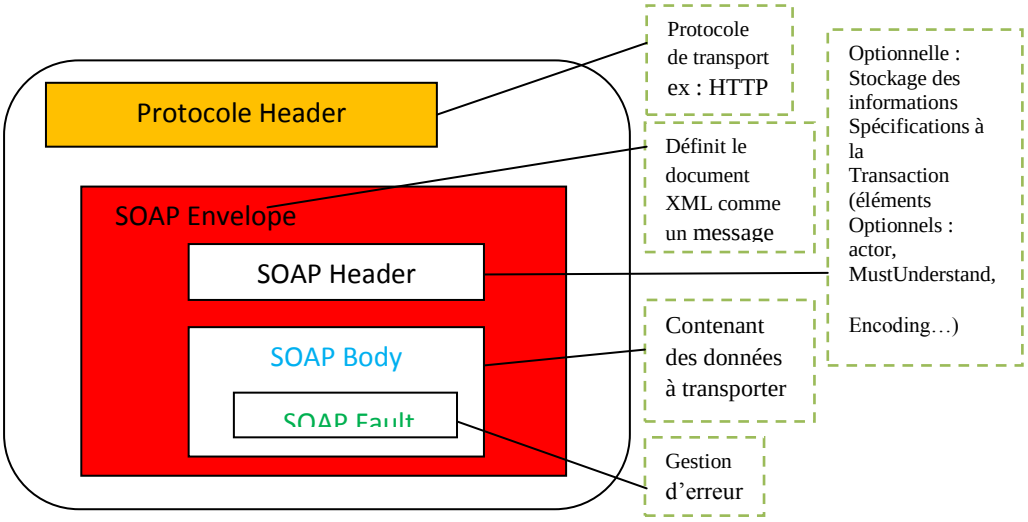


Figure 1. 4 : Structure d'un message SOAP

3.6.2. Description

➢ La technologie WSDL (Web Service Description Language)

✓ Définition général

WSDL est un langage qui permet de décrire les services web, et en particulier, les interfaces des services web. Ces descriptions sont documents XML. Le WSDL permettant de fournir les spécifications nécessaires à l'utilisation d'un service Web en décrivant les méthodes, les paramètres et ce qu'il retourne. **[18]**

✓ Structure d'un document WSDL

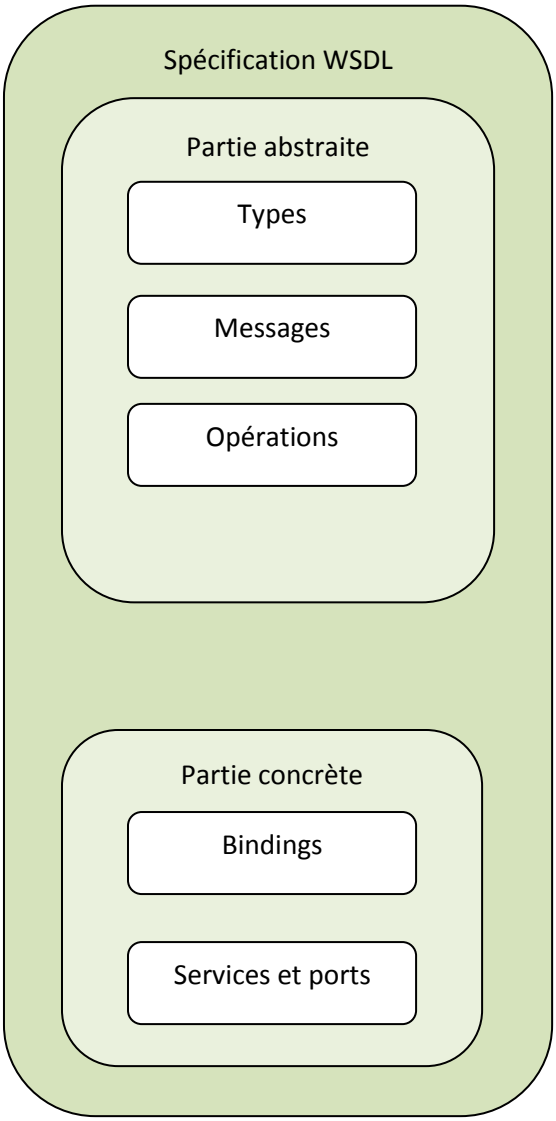


Figure 1. 5 : Structure d'un document WSDL

Défini la structure générale d'un document WSDL.

- **Types :** fournit la définition de types de données utilisés pour décrire les messages
- **Messages :** représente une définition abstraire (noms et types) des données en cours de transmission.
- **Opération :** c'est la description d'une action exposée dans le port.
- **Types de Port :** décrit un ensemble d'opérations. Chaque opération a zéro ou un message en entrée, zéro ou plusieurs messages de sortie ou d'erreurs.
- **Binding :** spécifie une liaison entre un <portType> et un protocole concret (SOAP, HTTP...).
- **Service :** indique les adresses de port de chaque liaison.
- **Port :** représente un point d'accès de services défini par une adresse réseau et une liaison.

Le document WSDL peut être divisé en deux parties. Une partie pour **les définitions abstraites**, tandis que la deuxième contient **les descriptions concrètes**. **[19]**

3.6.3. Publication

- **UDDI (Universal Description Discovery and Integration)**
  - ✓ **S Définition**

UDDI (OASIS), a été conçu en 2000 à l'initiative d'un ensemble d'industriels (Ariba, IBM, Microsoft), en vue de devenir le registre standard de la technologie des services Web. Pour convenir à la technologie des services Web, les services référencés dans UDDI sont accessibles par l'intermédiaire du protocole de communication SOAP, et la publication des informations concernant les fournisseurs et les services doit être spécifiée en XML afin que la recherche et l'utilisation soient faites de manière dynamique et automatique. UDDI est une spécification définissant la manière de publier et de découvrir les services Web sur un réseau. **[20]**

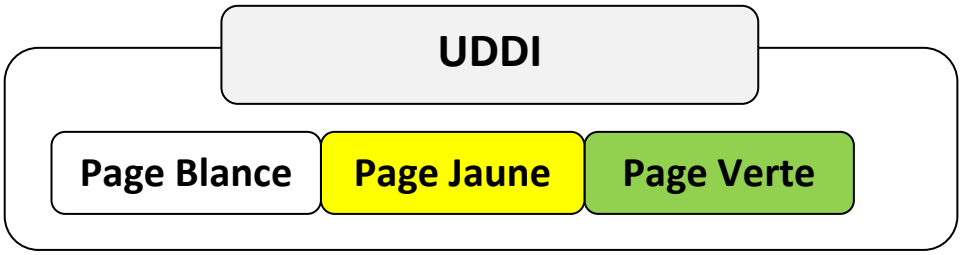


Figure 1.6 : les trois types de l'annuaire UDDI

Les informations qu'il contient peuvent être séparées en trois types :

- **Les pages blanches** incluant l'adresse, le contact et les identifiants relatifs aux services Web.
- **Les pages jaunes** identifiant les secteurs d'affaires relatifs aux services Web.
- **Les pages vertes** fournissent des informations techniques précises sur les services fournis.

Une fois ceci est fait, le service Web peut alors être connu de tous ceux qui le recherchent. Le modèle UDDI comporte quatre types de structures de données décrites sous forme de schéma XML : (Business entity, Business service, BindingTmodel, Publisher Assertion). **[21]**

3.7. Les avantages et inconvénients des services Web

3.7.1. Avantages

L'idée essentielle derrière les services Web est de partager les applications et les programmes en un ensemble d'éléments réutilisables appelés service, de sorte que, chacun de ces éléments effectuent une tâche principale et efficace, afin de faciliter l'interopérabilité entre tous ces services Web.

- Les services Web fournissent l'interopérabilité entre divers logiciels fonctionnant sur diverses plates-formes.
- Permettent de profiter de différents environnements et langages de développement par une publication, localisation, description et une invocation via XML. Les services
- Web sont très flexibles, indépendants des langages de programmation et des systèmes d'exploitation.

- Les protocoles et les formats de données sont au format texte dans la mesure du possible, facilitant ainsi la compréhension du fonctionnement global des échanges.
- Basés sur le protocole HTTP, les services Web peuvent fonctionner au travers de nombreux pare-feux sans nécessiter des changements sur les règles de filtrage.
- Les outils de développement, s'appuyant sur ces standards, permettent la création automatique de programmes utilisant les services Web existants. **[22] [23]**

3.7.2 Inconvénients

- La sémantique n'est pas prise en charge de façon efficace car le WSDL décrit les services de manière syntaxique.
- Les services web ont de faible performances par rapport à d'autres approches de l'informatique répartie telles que le RMI, CORBA, ou DCOM.
- En l'utilisation du protocole http, les services Web peuvent contourner les messures de sécurité mises en place à travers les firewalls.
- Ils sont pas sécurisés à 100 % **[24]**

4. Conclusion

L'introduction des services web a donc changé l'approche des applications en effectuant le passage d'une orienté objet à une architecture orienté service.

Les services web sont des applications accessibles par l'échange de documents XML entre deux URL. Ils permettent une souplesse d'utilisation et une accélération du développement d'applications.

1. Introduction

Le couplage des services bancaires aux services web est plus que d'actualité aujourd'hui dans l'univers bancaire.

L'objet de ce chapitre est de réaliser un web service d'une gestion d'un compte bancaire.

2. Modélisation

Notre projet concerne la gestion du compte bancaire via un web service.

Pour cela, le compte bancaire est caractérisé par un code incrémenté automatiquement.

A sa création, un compte bancaire a un nom et un solde.

Il est aussi possible de créer un compte bancaire en précisant son solde initial.

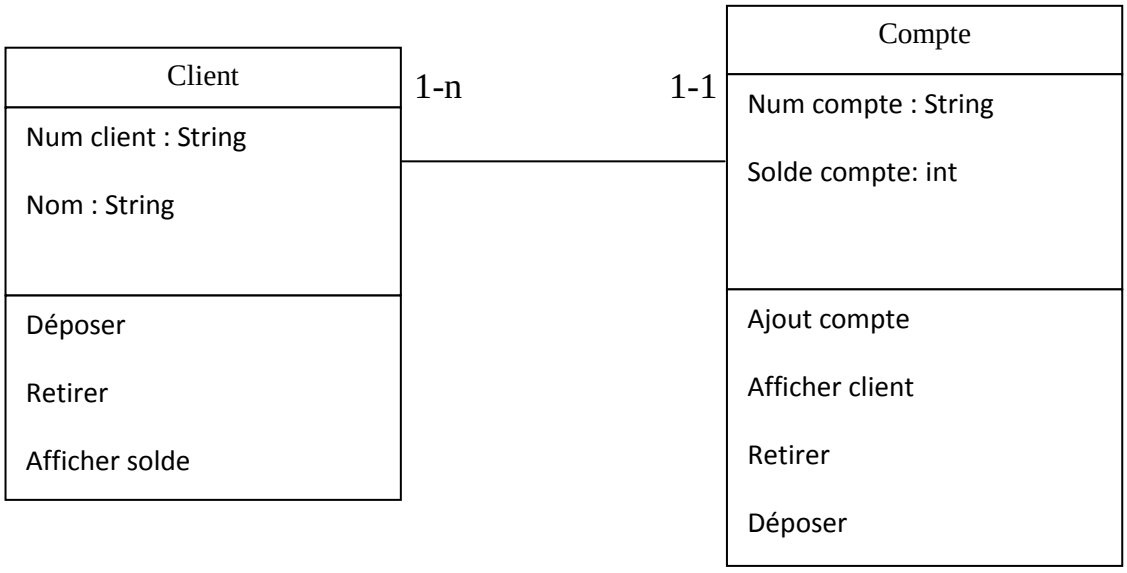
Utiliser son compte consiste à pouvoir y faire des dépôts et des retraits.

Pour ces deux opérations, il faut connaître le montant de l'opération. L'utilisateur peut aussi consulter le solde de son compte par la méthode (affichesolde).

Dans ce chapitre, nous allons présenter les diagrammes UML de notre application.

3. Diagramme de classe

Le diagramme ci-dessous représente les deux classes utilisées dans notre application :



4 .Le langage de programmation JAVA

4.1. Définition

Java est un langage de programmation et une plate-forme informatique créée par Sun Microsystems en 1995. Il s'agit de la technologie sous-jacente qui permet l'exécution de programmes dernier cri, notamment des utilitaires, des jeux et des applications professionnelles. Java est utilisé sur plus de 850 millions d'ordinateurs de bureau et un milliard de périphériques dans le monde, dont des périphériques mobiles et des systèmes de diffusion télévisuelle.

4.2. Objectif

Le langage JAVA participe pleinement à la création des web services.

Généralement, les objectifs du langage JAVA sont les suivants :

- Simple, orienté objet et syntaxiquement familier.
- Robuste et sécurité.
- Utilisant une architecture neutre et etre portable (.NET).
- Disposant de haute performance
- Interprété et disposant de processus lèges (multithreading).

5. Exemple d'Implémentation d'un web service en JAVA

Nous allons montrer comment nous avons implémenté nos web services.

5.1. L'exemple de la classe création du compte bancaire

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package cc;
import java.util.Vector;
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;
/**
 *
 * @author home
 */
```

```
@WebService(serviceName = "creation")
public class creation {
  static Vector <client> clients =new Vector();
   /**
    * Web service operation
    */
      }
   /**
    * Web service operation
    */
   @WebMethod(operationName = "afficheclient")
   public String afficheclient() {
     String ss="";
       //TODO write your implementation code here:
       for(int i=0; i<clients.size();i++)
                return ss;
   }
   /**
    * Web service operation
    */
   @WebMethod(operationName = "ajout")
   public String ajout(@WebParam(name = "numcpte") String numcpte, @WebParam(name
= "solde") int sold) {
       //TODO write your implementation code here:
       for(int i=0; i<clients.size();i++){
       if(clients.elementAt(i).accNo.equals(numcpte)) {
        clients.elementAt(i).deposit(sold);
       }
             }
       return null;
```

```
    }
    /**
     * Web service operation
     */
    @WebMethod(operationName = "affichesolde")
    public int affichesolde(@WebParam(name = "numcompte") String numcompte) {
        //TODO write your implementation code here:
                }
               }
        return 0;
    }
    /**
     * Web service operation
     */
    @WebMethod(operationName = "retrait")
    public int retrait(@WebParam(name = "numCompte") String numCompte,
@WebParam(name = "valeur") int valeur) {
        //TODO write your implementation code here:
         for(int i=0; i<clients.size();i++){
         if(clients.elementAt(i).accNo.equals(numCompte)) {
         return clients.elementAt(i).withdraw(valeur);
        }
               }
        return 0;
    }
}
```

5.2 .l'exemple de la classe client

```
package cc;

/*

 * To change this template, choose Tools | Templates

 * and open the template in the editor.

 */

/**

 *

 * @author home

 */

public class client {

   String name, actype,accNo;

      int  bal, amt;

   public client (String name, String accNo) {

            this.name = name;

            this.accNo = accNo;

            this.bal = 0;

      }

      int deposit(int amt) {

            //System.out.print("Enter amount to deposit:");

            //amt = input.nextInt();

            if (amt < 0) {

                  System.out.println("Invalid Amount");

                  return 1;

            }
```

```
            bal = bal + amt;
            return bal ;
        }
        int withdraw(int amt) {
            //System.out.println("Your Balance=" + bal);
            //System.out.print("Enter amount to withdraw:");
            //amt = input.nextInt();
            if (bal < amt) {
                System.out.println("Not sufficient balance.");
                return 1;
            }
            if (amt < 0) {
                System.out.println("Invalid Amount");
                return 1;
            }
            bal = bal - amt;
            return bal;
        }


        }
    }
```

5.3 .Présentation des pages de teste

Nous allons illustrer les démarches pour tester notre application.

5.3.1.Page de teste « création un compte » selon la figure en dessous

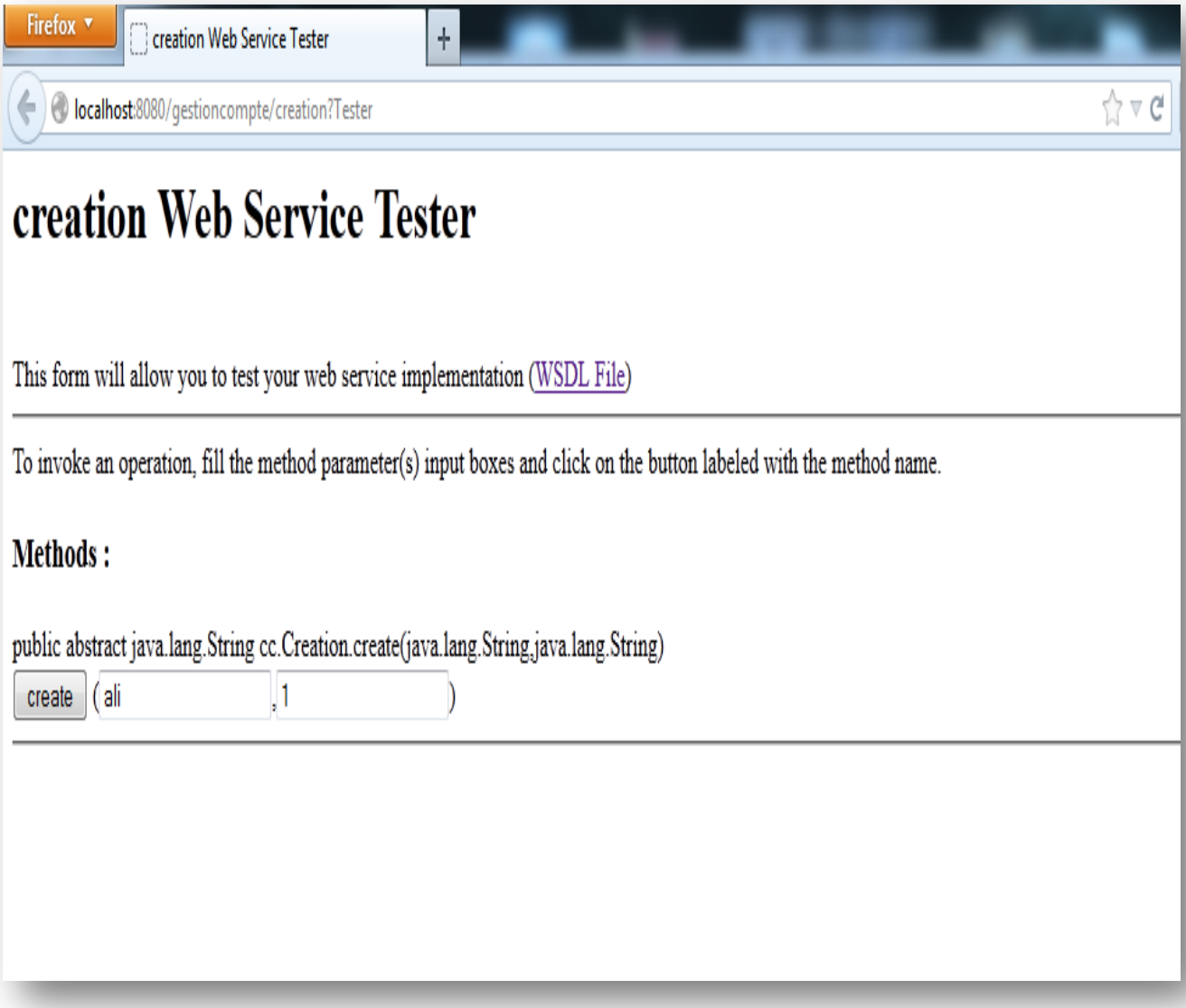
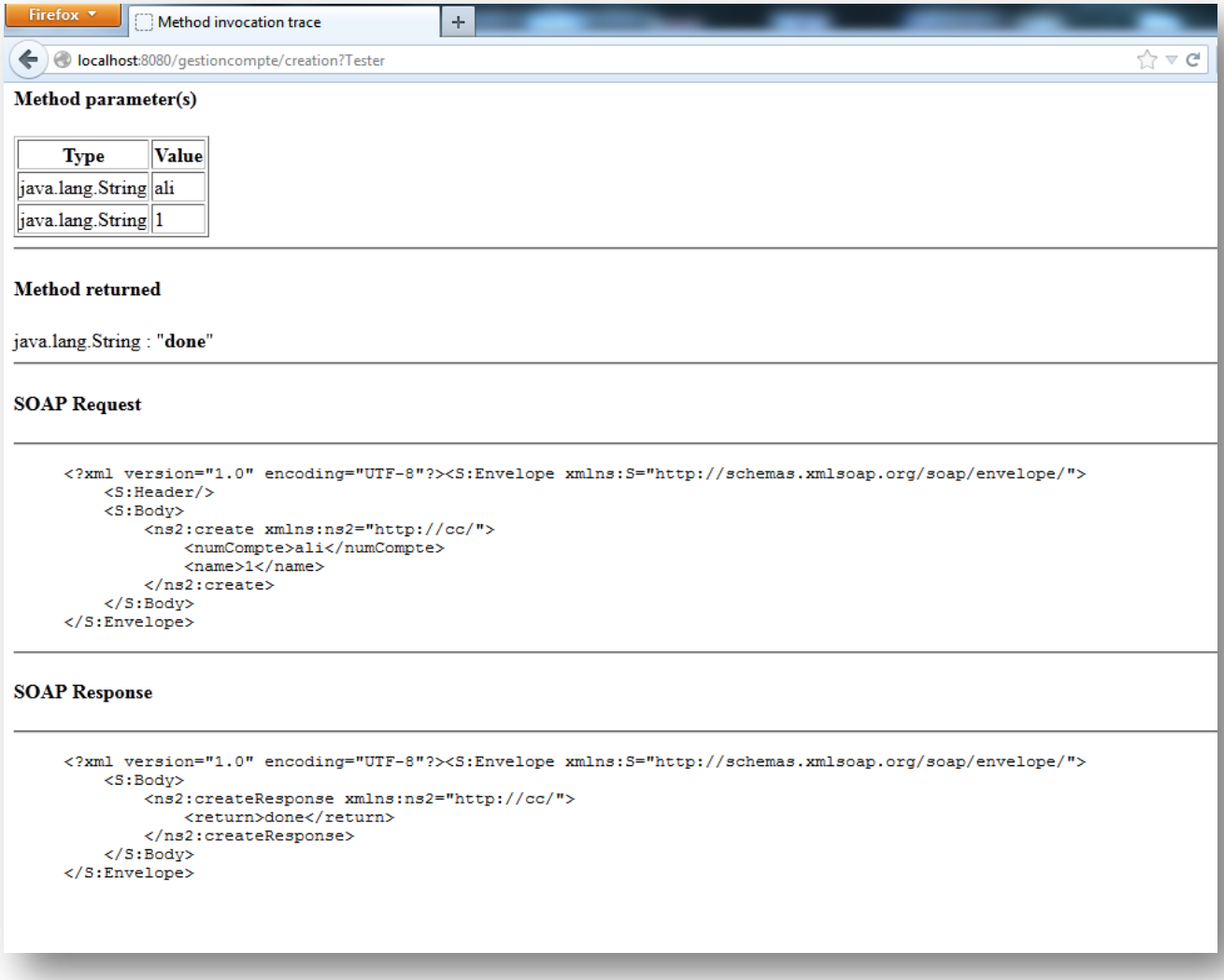


Figure 2.1: Création un compte bancaire

La figure 2.2, représente un nouveau compte bancaire :



La figure 2.2 : Compte bancaire est créé

5.3.2.page de teste « ajout d'un solde» :

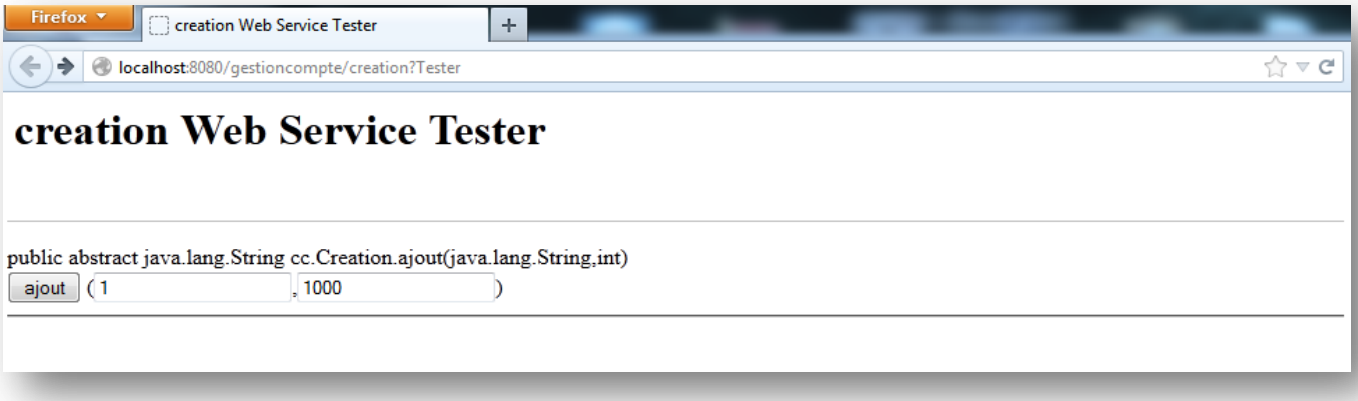


Figure 2.3:Ajout d'un solde pour un compte bancaire

Figure 2.4 :le solde est ajouté par l'utilisateur

6. Le fichier WSDL

Le WSDL est un langage qui permet :

De décrire un service Web, et comment l'invoquer (sa méthode d'invocation), son Objectif est :

● Décrire les services comme un ensemble d'opérations et de messages abstraits relié à des protocoles et des serveurs réseaux

● Permet de décharger les utilisateurs des détails techniques de réalisation d'un appel

● WSDL est un langage qui standardise les schémas XML utilisés pour établir une connexion entre émetteurs et récepteurs.

Nous pouvons accéder aux fichiers de description « WSDL » de notre web service qui est comme suit :

```
- <definitions targetNamespace="http://cc/" name="creation">
  - <types>
    - <xsd:schema>
        <xsd:import namespace="http://cc/" schemaLocation="http://localhost:8080/gestioncompte/creation?xsd=1"/>
      </xsd:schema>
    </types>
  - <message name="create">
      <part name="parameters" element="tns:create"/>
    </message>
  - <message name="createResponse">
      <part name="parameters" element="tns:createResponse"/>
    </message>
  - <message name="ajout">
      <part name="parameters" element="tns:ajout"/>
    </message>
  - <message name="ajoutResponse">
      <part name="parameters" element="tns:ajoutResponse"/>
    </message>
  - <message name="affichesolde">
      <part name="parameters" element="tns:affichesolde"/>
    </message>
  - <message name="affichesoldeResponse">
      <part name="parameters" element="tns:affichesoldeResponse"/>
    </message>
  - <message name="retrait">
      <part name="parameters" element="tns:retrait"/>
    </message>
  - <message name="afficheclient">
      <part name="parameters" element="tns:afficheclient"/>
    </message>
  - <message name="afficheclientResponse">
      <part name="parameters" element="tns:afficheclientResponse"/>
```

```
  </message>
- <portType name="creation">
  - <operation name="create">
      <input wsam:Action="http://cc/creation/createRequest" message="tns:create"/>
      <output wsam:Action="http://cc/creation/createResponse" message="tns:createResponse"/>
    </operation>
  - <operation name="ajout">
      <input wsam:Action="http://cc/creation/ajoutRequest" message="tns:ajout"/>
      <output wsam:Action="http://cc/creation/ajoutResponse" message="tns:ajoutResponse"/>
    </operation>
  - <operation name="affichesolde">
      <input wsam:Action="http://cc/creation/affichesoldeRequest" message="tns:affichesolde"/>
      <output wsam:Action="http://cc/creation/affichesoldeResponse" message="tns:affichesoldeResponse"/>
    </operation>
  - <operation name="retrait">
      <input wsam:Action="http://cc/creation/retraitRequest" message="tns:retrait"/>
      <output wsam:Action="http://cc/creation/retraitResponse" message="tns:retraitResponse"/>
    </operation>
  - <operation name="affricheclient">
      <input wsam:Action="http://cc/creation/affricheclientRequest" message="tns:affricheclient"/>
      <output wsam:Action="http://cc/creation/affricheclientResponse" message="tns:affricheclientResponse"/>
    </operation>
  </portType>
- <binding name="creationPortBinding" type="tns:creation">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
  - <operation name="create">
```

```
      <soap:operation soapAction=""/>
    - <input>
        <soap:body use="literal"/>
      </input>
    - <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  - <operation name="ajout">
      <soap:operation soapAction=""/>
    - <input>
        <soap:body use="literal"/>
      </input>
    - <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  - <operation name="affichesolde">
      <soap:operation soapAction=""/>
    - <input>
        <soap:body use="literal"/>
      </input>
    - <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  - <operation name="retrait">
      <soap:operation soapAction=""/>
    - <input>
        <soap:body use="literal"/>
      </input>
    - <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  - <operation name="affichecliient">
      <soap:operation soapAction=""/>
    - <input>
        <soap:body use="literal"/>
      </input>
    - <output>
```

```
            <soap:body use="literal"/>
          </output>
        </operation>
      </binding>
    - <service name="creation">
      - <port name="creationPort" binding="tns:creationPortBinding">
          <soap:address location="http://localhost:8080/gestioncompte/creation"/>
        </port>
      </service>
    </definitions>
```

Figure 2.5: représentation du fichier WSDL

7. Conclusions

Dans ce chapitre, nous avons présenté la réalisation de notre application,

Suivant les approches de modélisation et le langage utilisé pour atteindre notre but,

Celui de l'élaboration et le perfectionnement de notre présente application.

Conclusion Générale

Ce projet de fin d'étude présente l'aboutissement d'un travail laborieux, il nous a permis de connaitre beaucoup de choses qui ont amélioré nos connaissances dans le domaine des services web.

Nous avons présenté dans ce mémoire les technologies liées aux services web, en l'occurrence une application qui a aboutit la création d'un compte bancaire via services web satisfaisant les besoins de l'utilisateur.

Tout travail est amené à être amélioré, en ce sens, notre application peut encore évoluer

Voire s'améliorer.

Comme perspectives au travail réalisé ci dessus, nous proposons les améliorations suivantes :

Composer notre application (services web) avec d'autres applications (services web).

Références

[1] Adel Boukhadra, La composition dynamique des services Web sémantiques a base d'alignement des ontologies owl-s,2011.

[2] Ben Margolisand, et Joseph Sharpe,SOA for the Business developer: Concepts,BPEL,and SCA, edition MC Press, October 2007.

[3] Frank Jenning, Ramesh Loganathan, et Poornachandra Sarang, Approach to integration , XML, Web services, ESB, and BPEL in real-world SOA projects, edition Packt Publishing Ltd ,November 2007.

[4] Antoine Crochet-Damais « SOA pour service Oriented Archetecture : décryptage »,2007.

[5] Willianm Brown, Robert G laid, Clive Gee, Tilak Mitra : SOA Governance Archieving and Subtaining Business and IT Agility, 2008.

[6] http://commentcamarche.net/contents/web-services-soa-architecture-orientee-services.php3.

[7] D.Booth, H Haas, F,McCabe, E.Newcomer, M.Champion, C.Ferris,et D.Orchad, Web service Archetecture, W3C Working Group Note, Fevrier 2004.

http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

[8] Uniform Ressource Identifier, étudié le 03/05/2013,

http://fr.wikipedia.org/wiki/Uniform_Resource_Identifier, (derniere visite 09/05/2013)

[9] Standard and web services,

http://www.ibm.com/developerworks/webservices/standard#ibm-pcon, (dernière visite 09/05/2013)

[10] Heather Kreger, Web service Conceptual Architecture, edition IBM Software Group, Mai 2001.

[11] XML-RPC ,édité le 03/05/2013, http://php.net/manual/fr/book.xmlrpc.php (dernière visite 09/05/2013).

[12] http://www.ecr-france.org/boite-a-outils/glossaire-ecr/341-xml-extensible-markup-language-?lang

[13] Chauvet J, 'service web avec SOAP ,WSDL,UDDI , ebXML', jouve, paris, mars 2012.

[14] SoftDeath : Archetecture d'un service web 2003.

[15] http://www.sitezero.com/informatique/tutoriels/les -srvices-web/architecture-d-un-srvice-web.

[16] Batiste Bielier, etude de faisabilité d'une application SOAP avec un système embarqué, école HE-ARC ingénierie informatiques , 2005.

[17] W3C World Wide Web Consortium , SOAP version 1.2 part 1 : Messaging Framwork , Disponible à http://www.w3.org/TR/soap12-part1/.

[18] Roberto Chinnici, et Martin Gudgin, web services description language, edition W3C le 22 Aout 2004.

[19] Roberto Chinnici , et Martin Gudgin and Weerawarana S, 'Web services Description Language (WSDL) version 1.2', W3C Working Draft January 2003.

[20] Site official d'UDDI :http:/www.uddi.org/

[21] Céline Lopez-Velasco, Sélection et composition de services Web pour la génération d'applications adaptées au contexte d'utilisation , pour obtenir le grade de Docteur de l'université JOSEPH FOURIER, 18 novembre 2008.

[22] http://fr.wikipedia.org/wiki/Service_Web#Avantages.

[23] Davis Chappel, et Tyler JEWELL, Java Web Service, édition O'Reilly, mars 2002.

[24] http://www.w3.org/2001/XMLSchema-instance