الجـــــمهورية الـــــجزائرية الديمقراطية الشـــــــعبية

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE SAAD DAHLEB DE BLIDA

FACULTE DES SCIENCES

DEPARTEMENT D'INFORMATIQUE

MEMOIRE

En vue de l'obtention du diplôme d'ingénieur d'état en

GENIE INFORMATIQUE

Option :

S.I .Avancés

Présenté Par :

ATTABA ASSIA

BENSAID NORA

THEME

LES WEB SERVICES

Soutenu le jeudi 11Decembre 2003 devant le jury composé de :

Melle MOKHTARI	**Université De BLIDA**	**Président**
Mme EL-MAOUHEB	**CERIST**	**Promoteur**
Mme HESSAS	**CERIST**	**CO- Promoteur**
Melle ADLI	**CERIST**	**Encadreur**
Melle BOUSTIA	**Université De BLIDA**	**Examinateur**
Mr HADJ YAHIA	***Université De BLIDA***	***Examinateur***

Année Universitaire 2002/2003

MIG-004-12-1

MIG-004-12-1

Dédicaces

جامعة البليدة
المكتبة
كلية العلوم

Nous dédions ce travail à :

Nos Chères Mères, Nos Chers Pères, à vous. Vous n'avez cessé de nous aider et nous propulser vers le succès et la réussite depuis notre enfance, nous vous dédions de tout cœur notre travail, notre réussite vous êtes notre source de vie, notre joie et notre Bonheur que dieu vous garde pour nous.

Mes Frères et Soeurs: Amel, Amina, Mehdi.
Pour votre aide, soutien, pour m'avoir donné envie de continuer pour des succès futurs Inchallah.

Nos Grands Parents, nos Tantes, nos Oncles, nos Cousins, nos Cousines et nos Familles :

« BENSAID, FERHAH » et «ATTABA, CHERROUK ».

Spéciale dédicace : à toi Tante « Samou ».

A tous les maîtres de nos Ecoles Primaires.
A tous nos professeurs des Collèges, Lycées et Université.
A vous Mr Bala et Mr HadjYahia.
À vous Mme Benstitti -Chef de département Informatique-
Première promotion d'Ingénieurs d'Etat de l'Université de Blida.
A tous nos Amis et à tous nos camarades de la première Promotion

« Années : Septembre 1998-Décembre 2003 »

Remerciements

Nous Remercions :

Le Bon Dieu de nous avoir aidé à accomplir notre travail et nous a accordé patience, courage et bienveillance tout le long de notre parcours.

Notre chef de département Mme Benstitti pour son soutien et ses encouragements.

Notre promoteur le chef de département Réseau au CERIST : Mme EL-Maouheb, notre Co-Promoteur Mme Hessas de nous avoir accordé ce stage au sein du centre ainsi que Melle Adli.

Notre commission de suivi Mr Bala et Mr Hadj-Yahia pour leurs conseils et aide tout le long de l'année.

Tous nos professeurs à l'université de Blida, ainsi que nos professeurs de la cinquième année.

Nos parents, pour nous avoir encouragé, soutenu, Merci à vous, nous sommes le fruit de votre Bonté, Patience ainsi que tous vos Efforts et Sacrifices.

Nos frères et sœurs pour leurs permanents encouragements ; Merci à toi Mehdi.

Résumé

*Dans les systèmes distribués une nouvelle technologie appelée «**Web Services**» a vu le jour. Il s'agit d'une technologie permettant à des applications de dialoguer à distance via Internet, et ceci indépendamment des plateformes et des langages sur lesquels elles reposent.*

La simplicité et la standardisation sont deux caractéristiques qui ont incité les acteurs du monde informatique à adopter cette évolution qui remplacera peut être les anciens systèmes connus par leur complexité et leur difficulté de mise en œuvre.

Ayant comme terrain d'action le Web, les Web Services peuvent bénéficier de son universalité et de son évolution.

L'objectif de notre projet est d'explorer ce domaine,afin de découvrir le fonctionnement de l'ensemble de protocoles standardisant le mode d'invocations mutuelles, qui comprend le langage XML pour le format des données,le protocole SOAP pour la structure des messages XML,le langage WSDL pour la description des Web Services.

Dans le but de comprendre le fonctionnement des Web Services, nous avons eu recours à l'étude des plateformes J2EE et dotNET ; deux technologies de développement des Web Services, pour des raisons de disponibilité nous avons choisi la seconde afin de développer un environnement basé sur cette technologie nouvelle.

L'échange entre applications/Web Services est illustré selon le modèle client/serveur et cela grâce à l'atelier de développement gratuit, conçu et élaboré par Microsoft, permettant la création dans différents langages offerts par la plateforme, le test et la compilation des Web Services ainsi que leur consommation par des applications ASP.NET.

CHAPITRE III : ETUDE DU PROTOCOLE SOAP 22

III-1-Architecture SOAP : 22
III-1-1-Utilité de XML dans les messages SOAP : 23

III-2-La structure des messages SOAP : 23
III-2-1- L'enveloppe SOAP : <SOAP-ENV:Envelope> 24
III-2-2-L'en-tête SOAP : <SOAP-ENV:Header> : 25
III-2-3-Le corps du message SOAP : <SOAP-ENV:Body> 26

III-3- Les règles d'encodage des types de données : 27

III-4-Les pièces jointes : 30

III-5-Le routage des messages SOAP « WS-Routing » : 31
III-5-1-Modèle de routage des messages WS-Routing : 32
III-5-2- Rôle des intermédiaires WS-Routing : 33

III-6-Le transport SOAP/HTTP: 34

III-7-Conclusion: 35

CHAPITRE IV : ETUDE DE LA NORME WSDL 37

IV-1-Structure d'un document WSDL : 37

IV-2-Exemple de fichier WSDL : 40

IV-3-Les messages SOAP : 46
IV-3-1- Requête envoyée à partir d'un client pour réaliser un appel de fonction : 47
IV-3-2- Réponse reçue en retour du serveur par le client : 47

IV-4-Un Web Service qui comporte plusieurs méthodes : 48

IV-5-Conclusion : 50

CHAPITRE V : ETUDE DES PLATEFORMES J2EE/.NET 53

V-1-L'architecture type 53

V-2-les plateformes J2EE/.NET 54
V-2-1- Couche présentation : 55
V-2-2- Couche de service : 55
V-2-3- Couche d'objets métiers (ou domaine) : 55
V-2-4- Couche d'accès aux données ou de Persistance : 55

V-3-L'implémentation des Web Services dans J2EE/.NET : 56
V-3-1-Les Web Services dans .NET 56
V-3-2-Les Web Services dans J2EE: 57

V-4-Scénario d'utilisation d'un Web Service /.NET 58

V-5-Conclusion 60

CHAPITRE VI : WEB SERVICES ET OUTILS DE DEVELOPPEMENT .NET 62

VI-1-Le Serveur des Web Services (Module 1) : 63

VI-2-Les applications clientes : (Module 2) 63

VI-3-les messages SOAP/XML (Module 3): 65

VI-4- La plateforme .NET et le « .Net Framework » : 70
VI-4-1- Common Language Runtime (CLR) 73
VI-4-2- Common Type System (CTS) 73
VI-4-3- Common Language System (CLS) 73
VI-4-4- L'Intermediate Language (IL) 73
VI-4-5- Accès aux données avec ADO.NET 74

VI-5- Choix de l'architecture ASP.NET, WebMatrix et MSDE 75
VI-5-1-Choix de la technologie de développement Web 75
VI-5-2-Choix de l'environnement de développement ASP.NET Web Matrix 79
VI-5-3-Choix de la base de données 83
VI-5-4-Configuration materiel ET logiciel : 85

VI-6-Conclusion 86

CHAPITRE VII : REALISATION 87

VII-1-Comment créer un Web service en .Net ? 87

VII-2-Test et consommation d'un Web Service en .Net 90

VII-3-Le Rôle de la Classe Proxy 94
VII-3-1-Génération de la classe Proxy 95
VII-3-2-l'utilitaire WSDL.EXE de .Net 97
VII-3-3-Classe Proxy 98.

VII-4-Application ASP.NET consommant les Web Services 99
VII-4-1-Conception du site 101
VII-4-2- Gestion des données 102
VII-4-3- Mémorisation des commandes par le Panier 105

VII-5- Sécurité SOAP des Web services 108

VII-6- Conclusion 110

CONCLUSIONS GENERALE 113

REFERENCES 115

BIBLIOGRAPHIES 116

ANNEXES 118

ANNEXE A 119

ANNEXE B 121

ANNEXE C 123

Sommaire

Première Partie :
Etude des Protocoles et des Standards
« WEB SERVICES »

DEDICACES I

REMERCIEMENTS II

RESUME III

ABSTRACT IV

SOMMAIRE V

LISTE DES ILLUSTRATIONS IX

LISTE DES TABLEAUX XI

INTRODUCTION GENERALE 1

CHAPITRE I : INTRODUCTION AUX WEB SERVICES 5

I-1- Le Service Web: 5

I-2- Présentation des Web Services : 6

I-3- L'innovation des Web Services : 8

I-4- L'entreprise et les Web Services: 9

I-5- Conclusion : 10

CHAPITRE II : ETUDE DU LANGAGE XML 12

II-1-XML le langage : 12
II-1-1- Anatomie d'un document XML : 13
II-1-2- Les DTD (Data Type Definition) : 16
II-1-3- Les Domaines Nominaux (XML Namespaces) : 17
II-1-4- XMl schéma: 17

II-2- XML et les Web Services : 19

II-3- Conclusion : 20

Abstract

In distributed system, a new technology called web services emerged. It consists of a technology allowing remote communication via Internet independently from the languages and the platforms technology.

The standardisation and the simplicity of this technology are the two characteristics inciting the actors in the world of Information technology to adopt this evolution which will may be in future replace the old known systems.

The aims of our work consist on the study and exploration of this field to be able to understand the remote standardised protocols XML for text format, SOAP for message transport and WSDl for Web services description.

Web services allow the components of one application to be used by other applications in different locations via the Web using a set of underlying protocols, such as HTTP, Simple Object Access Protocol (SOAP) and XML. Web services are the building blocks for creating open distributed systems and offer a way to design systems with modularity, flexibility and platform independence using standardized components. Web services provide the tools enabling government and businesses to collaborate with customers and trading partners in real-time, develop and deploy new products and services more quickly with less resources, improve efficiency and productivity while taking advantage of the distributed nature of modern corporate networks.

There are many platform allowing the development of web services, two of them has been considered in this work J2EE and .Net, the last one did meet most of the requirement of availability was selected. With it's free development tool Web matrix from Microsoft, the communication Application/Web Services was demonstrated.

ANNEXE D 124

ANNEXE E 125

ANNEXE F 126

ANNEXE G 128

ANNEXE H 130

GLOSSAIRE 132

Liste des Illustrations

FIGURE I- 1--PRINCIPE DES WEB SERVICES 8

FIGURE II- 1- STRUCTURE GENERALE D'UN DOCUMENT XML. 13
FIGURE II- 2- ANATOMIE D'UN DOCUMENT XML 13
FIGURE II- 3-STRUCTURE D'UN ELEMENT SIMPLE. 14
FIGURE II- 4- STRUCTURE D'UN ELEMENT COMPOSE. 14
FIGURE II- 5- DOCUMENT XML AVEC ATTRIBUTS 15
FIGURE II- 6-AFFICHAGE D'UN DOCUMENT XML 15
FIGURE II- 7-UN DOCUMENT XML ET SON DTD. [4] 16
FIGURE II- 8- L'INFORMATION ENTRE WEB SERVICES [4] 19

FIGURE III- 1- LA STRUCTURE DU MESSAGE SOAP [4] 24
FIGURE III- 2- L'ENTETE DANS DES TRANSFERTS SUCCESSIFS DU MESSAGE SOAP 26
FIGURE III- 3- LE MESSAGE SOAP INCLUS DANS UNE REQUETE OU UNE REPONSE HTTP [4] 34

FIGURE IV- 1 STRUCTURE D'UN DOCUMENT WSDL [6] 38
FIGURE IV- 2 -DEFINITIONS CONCRETES ET ABSTRAITES [5] 39
FIGURE IV- 3 REQUETE SOAP 47
FIGURE IV- 4 REPONSE SOAP 47
FIGURE IV- 5 WSDL D'UN WEB SERVICE AVEC PLUSIEURS METHODES 49

FIGURE V- 1-LES COUCHES DE L'ARCHITECTURE N-TIERS. [7] 54
FIGURE V- 2 -LA MISE EN OEUVRE DES TECHNOLOGIES J2EE/.NET [7]. 55
FIGURE V- 3-L'ARCHITECTURE WEBSERVICES .NET [7] 56
FIGURE V- 4-L'ARCHITECTURE WEBSERVICES J2EE 57
FIGURE V- 5- ETAPE DE MISE EN ŒUVRE EN VISUAL STUDIO .NET [8] 58
FIGURE V-6-ROLE DE LA CLASSE PROXY [8] 59
FIGURE V- 7-.APPEL DES WEB SERVICE A TRAVERS INTERNET 60

FIGURE VI - 1-MODELE DE SYSTEME BASE SUR LES WEB SERVICES. 62
FIGURE VI - 2- L'INTERACTION WEB SERVICES ET LES APPLICATIONS CLIENTES. 64
FIGURE VI - 3 DOCUMENTATION DU KIT DE DEVELOPPEMENT .NET FRAMEWORK SDK 70
FIGURE VI - 4 - VUE D'ENSEMBLE DU .NET FRAMEWORK 71
FIGURE VI - 5. LES CLASSES DU FRAMEWORK 72
FIGURE VI - 6. L'ARCHITECTURE FONCTIONNELLE DE .NET 72
FIGURE VI - 7 –VUE D'ENSEMBLE DE L'ARCHITECTURE ET DES COMPOSANTS DE ADO.NET 74
FIGURE VI - 8. TRAITEMENT D'UNE REQUETE VERS UNE PAGE ASP.NET. [6] 78
FIGURE VI - 9.CONCEPTEUR DE PAGES ASP.NET WEBMATRIX 79
FIGURE VI - 10- FICHIERS GENERAUX. 81
FIGURE VI - 11- FICHIERS DE DONNEES. 81
FIGURE VI - 12- FICHIERS WEBSERVICES 81
FIGURE VI - 13- ASP.NET WEBMATRIX TOOLBOX 82
FIGURE VI - 14- SERVEUR WEB DE DEVELOPPEMENT 82
FIGURE VI - 15- CONFIGURATION DU SERVEUR 82
FIGURE VI - 16 - MICROSOFT SQL SERVER DESKTOP ENGINE (MSDE) 83
FIGURE VI - 17- ACCES AUX BASE DE DONNEES AVEC WEBMATRIX 84
FIGURE VI - 18- ACCES ET CREATION DE BASE DE DONNEES SQL AVEC WEBMATRIX 84

FIGURE VII - 1- MONTANTTTC WEB SERVICE EN C# ET EN VB.NET 89

FIGURE VII- 2 - WEB SERVICE D'ECHANGE MONETAIRE (DINAR EURO ET DOLLAR) EN VB.NET89
FIGURE VII - 3 - CODE C# D'UN WEB SERVICE SIMPLE DE CALCULATRICE 90
FIGURE VII - 4 - WEB SERVICES DE CALCUL D'ECHANGE MONETAIRE (DINAR EURO ET DOLLAR) 90
FIGURE VII - 5 - ERREUR DE COMPILATION SUR UN WEB SERVICE 90
FIGURE VII - 6 - TEST DU WEB SERVICE DANS LE .NET FRAMEWORK. 91
FIGURE VII - 7 - DESCRIPTION DU WEB SERVICE « FICHIER WSDL » 91
FIGURE VII - 8 - PROTOCOLE SOAP DE LA METHODE MULTIPLICATION. 92
FIGURE VII - 9 - PROTOCOLE HTTP POST DE LA METHODE MULTIPLICATION. 92
FIGURE VII - 10 - RESULTAT APRES INTERROGATION DE LA METHODE. 92
FIGURE VII - 11- APPLICATION ASP.NET POUR CONSOMMER LE WEBSERVICE MATHSERVICE . 93
FIGURE VII - 12 - SCHEMAS DU ROLE DE LA CLASSE PROXY 94
FIGURE VII - 13 - GENERATION DE LA CLASSE PROXY AVEC WEBMATRIX. 95
FIGURE VII - 14 - LE GENERATEUR DE LA CLASSE PROXY ET DE L'ASSEMBLAGE APRES COMPILATION. 96
FIGURE VII - 15- CODE SOURCE DE L'APPLICATION CLIENTE DU WEB SERVICE 97
FIGURE VII - 16 - LA CLASSE PROXY DU WEB SERVICE MATHSERVICE 98
FIGURE VII - 17 -PAGE DE PRESENTATION. 99
FIGURE VII - 18 -VERIFICATION ET DE VALIDATION DES COMMANDES. 100
FIGURE VII - 19 –PAGE DE CONFIRMATION DES COMMANDES. 100
FIGURE VII-20–WEBSERVICE CALCULATRICE MATHEMATIQUE. 101
FIGURE VII-21-WEBSERVICE CONVERSIONS MONETAIRES. 101
FIGURE VII - 22 - PLAN DE L'APPLICATION ASP.NET ET APPEL DES WEB SERVICES. 102
FIGURE VII - 23- LA COUCHE DE DONNEES (DATALAYER.VB) LIEE AU FICHIER XML ET LA BASE DE DONNEES 103
FIGURE VII - 24 - FICHIER XML (COMPUTER.XML) DETAILLANT LES CHOIX DISPONIBLES POUR LE CLIENT 104
FIGURE VII - 25 - PAGE DE PRESENTATION ASP.NET A PARTIR DU FICHIER XML. 104
FIGURE VII - 26 - TABLES DE LA BASE DE DONNEES PCOMPUTERWS 105
FIGURE VII - 27 - PROCEDURES STOCKEES DE LA BASE DE DONNEES 105
FIGURE VII - 28 -CONTROLE WEB POUR LA BARRE DE NAVIGATION 105
FIGURE VII - 29 –CONTROLE DATAGRID DU PANIER 106
FIGURE VII - 30-- CODE DE CONSOMMATION DU WEB SERVICE MONTANT TTC 106
FIGURE VII - 31 – ELEMENTS DU PANIER. 107
FIGURE VII - 32-.PAGE DE VERIFICATION ET DE CONFIRMATION 107
FIGURE VII - 33 -PAGES DE GESTION DES COMMANDES PAR LE WEB MASTER. 108
FIGURE VII - 34 - PAGES DE LOGIN, LOGOUT ET INSCRIPTION 108

Liste des Tableaux

TABLEAU VI - 1. EXEMPLE DE LA BIBLIOTHEQUE DE CLASSES [9] 72
TABLEAU VI 2. DIFFERENTE TECHNOLOGIE DE DEVELOPPEMENT WEB. [6] 76
TABLEAU VI-3. LES DIFFERENTS TYPES DE FICHIERS DU CONCEPTEUR DE PAGES WEB ASP.NET. 80

TABLEAU VII- 1-DESCRIPTION DES ATTRIBUTS DE LA DIRECTIVE <%@ *WEBSERVICE*> 87
TABLEAU VII 2- COMMANDE DE L'UTILITAIRE WSDL.EXE 97

Introduction Générale

Auparavant pour mettre en place des applications distribuées, il fallait utiliser des technologies assez complexes telles que COM. Certes ces technologies étaient abordables pour un développeur, mais il fallait que le développeur passe du temps à établir un protocole de transmission. Les Web Services sont alors apparus pour faciliter tout d'abord la tâche des développeurs. Avant toute chose, Microsoft, contrairement aux idées reçues, n'a pas créé les Web Services mais Microsoft a participé avec de grandes entreprises telles que IBM, SUN à la standardisation des Web Services. Ceci montre bien que la technologie des Web Services est une technologie très jeune, ce qui bien sûr peut être un inconvénient pour son intégration au sein des entreprises. Mais les plus grands spécialistes prévoient une « explosion » de l'utilisation des Web Services d'ici 2004 toutes technologies confondues (.NET, Java ...).

L'un des plus gros avantages des Web Services est qu'ils reposent sur des protocoles standardisés. Cela permet que cette technologie soit exploitable par de nombreux langages.

En effet, les Web Services reposent sur des protocoles tels que XML et http, donc SOAP. Pour vulgariser ce dernier protocole, SOAP permet de faire circuler du XML via du HTTP. Donc lorsqu'on interroge un Web Service, les données sont transmises en XML via le port 80 (HTTP). Rien de plus simple ensuite pour le développeur de traiter l'information reçue. A l'heure actuelle, la quasi totalité des langages informatiques supporte ces protocoles : ils disposent en effet de fonctions pour lire un fichier XML (Parseur XML).

Notre travail consiste à comprendre le fonctionnement des Web Services, expliquer comment s'effectue l'échange entre client et serveur de Web Services, comment créer un Web Service et comment le consommer à travers une application. Pour cela nous avons entrepris l'étude selon les sept chapitres suivants :

Le *Chapitre I* : est consacré à une introduction aux Web Services, qui les présente et situe leur rôle dans l'informatique distribuée, nous avons donné l'idée générale pour la quelle les entreprises ont opté pour cette technologie qui hérite des fonctionnalités existantes du Web et qui offre un langage universel le XML, qui sera étudié dans le chapitre suivant.

Le *Chapitre II* : est consacré à l'étude du premier standard : le langage XML qui permet de coder les données, les structurer ; c'est le standard de base pour le reste des standards qui accompagnent cette technologie. La force d'XML a fait que le Web adopte cette norme pour tous les échanges entre entités d'applications, les Web Services en font l'usage de force dans

leur protocole de transmission qui sera étudié dans le *chapitre III* Ainsi que dans le standard WSDL qui sera étudié dans le *chapitre IV*.

Le *chapitre III* : est consacré à l'étude du deuxième standard : c'est le protocole SOAP, un protocole qui formalise les messages XML et permet leur transport via HTTP; ce protocole leur donne une structure standard en proposant une enveloppe qui contiendra le message et toutes les informations nécessaires pour les requêtes et les réponses émises depuis et vers des Web Services. Pour effectuer un transport qui comprend les données correctes d'une réponse à une requête d'un Web Service on a prévu un document WSDL écrit en XML ; c'est l'étude du chapitre qui suit.

Le *Chapitre IV* : est consacré à l'étude du troisième standard : c'est la norme WSDL qui permet de décrire le Web Service, car pour le retrouver il faut connaître les points d'accès, pour envoyer le bon message SOAP qui correspond à la représentation XML des données du service il faut connaître les bons paramètres et pour connaître l'adresse et le nom du Web Service il faut connaître les bonnes URI ; un document WSDL répond à tous ces besoins et présente le Web Service de façon standard.

Cette partie d'étude nous a permis de connaître les standards associés à cette technologie que sont les standards de base, chaque développeur de systèmes de Web Services doit passer par ces trois standards, pour réussir son système en Web Services. Mais reposer seulement sur ces trois standards demande beaucoup d'expérience et beaucoup de tests pour réaliser des systèmes qui génèrent et supportent la technologie Web Services. Pour ces raisons nous nous sommes orientés vers une plateforme qui propose un fonctionnement plus avancé et qui permet d'automatiser la partie la plus laborieuse des Web Services, c'est la génération du document WSDL ,le choix d'une plateforme est donc nécessaire,c'est l'objet d'étude du prochain chapitre.

Chapitre V : ce chapitre est consacré à l'étude des plateformes existantes permettant d'aborder la technologie Web Services ; deux candidates : J2EE et dotNET ; ce sont deux implémentations du modèle client/serveur ; chacune possède son architecture ; mais pour des raisons de disponibilité ainsi que d'intégrité nous avons élu .NET car elle possède les caractéristiques les plus simples de standards Web Services .Microsoft a pu fournir un fonctionnement généralisé sous le concept de standardisation. Donc l'étude des différents composants de cette dernière est nécessaire pour arriver à l'objectif principal qui est de comprendre le fonctionnement des Web Services .le chapitre suivant répond à ce besoin d'étude.

Le *Chapitre VI* : est destiné à l'étude détaillée des différentes parties de la plate forme .NET avec une présentation du Framework .NET et ses différents composants : ADO.NET, le CLR et bien d'autres parties. Ensuite le choix d'une architecture proposée par cette plateforme ainsi que l'outil Webmatrix permet de développer un environnement .NET accompagné du système MSDE, et de la technologie ASP .NET qui permet de créer des applications Web. Après l'étude de tout cet environnement vient l'étude du processus de développement des Web Services au sein de cette même plateforme et qui sera détaillée dans le dernier chapitre.

Chapitre VII : consacré à la réalisation, c'est la phase qui permet de comprendre comment fonctionne les Web Services de façon globale, montrer comment les créer, les tester, et les consommer; comment intervient chacun des standards étudiés dans les premiers chapitres, comment les messages SOAP permettent le transport des messages de données en requêtes et réponses ; comment le fichier WSDL décrit les Web Services, comment l'utiliser pour créer la classe Proxy, comment utiliser les URI.

Enfin nous avons créé des Web Services .NET : un convertisseur de monnaie, une calculatrice d'opérations mathématiques élémentaires et un Web Service qui calcule le montant TTC (ou TVA) ; nous les avons testé selon les techniques de NET ;ensuite nous avons créé une application ASP.NET qui les consomme. C'est une application de commande de microordinateurs, au moment de calculer le montant total le montant TTC est inséré grâce à un appel au Web Service qui le calcule et cela de façon transparente pour les utilisateurs. Dans la même page des liens vers des Web Services (convertisseur monétaire, calculatrice mathématique) sont présentés, ils possèdent leur propres interfaces.

Première Partie :

Etude des Protocoles et des Standards « WEB SERVICES »

Chapitre I : Introduction aux Web Services

La question qui vient à l'esprit pour un étranger à ce domaine est : « Que signifient exactement les **Web Services** ? ». Pour certains la réponse parait évidente : « Les **Web Services** sont des éléments d'offre d'une entreprise ». Par exemple : une offre est le service de réponse à une demande d'un utilisateur client par une entreprise à titre commercial. Cette réponse définie les Web Services dans un domaine précis pour mieux comprendre nous allons donner la définition de deux concepts dans le monde informatique: le Web et le Service.

- Le **WEB** abréviation de **World Wide Web,** a vu le jour en 1989 au CERN, le laboratoire européen de physique des particules à Genève. Le projet W3 ou Web, est présenté par son auteur, Tim Berners-Lee pour fournir un accès convivial à un système d'information global en employant des techniques réseau et hypertexte pour des équipes de chercheurs dans le monde entier et accédant à différents types de documents sur un grand nombre d'ordinateurs hétérogènes dispersés géographiquement. Le Web est souvent confondu à l'Internet; mais en fait *le **Web** est le service majeur de l'**Internet**.*

L'**Internet** est la fondation sur laquelle est construit le Web. Il en utilise la plus part des mécanismes et en intègre l'essentiel des services (mails, news, ftp, gopher, wais...).

Le **Web** est le service qui offre sur l'Internet une interface graphique permettant par simple pointage d'accéder, suivant une technique dérivée de celle de l'hypertexte, à des documents sur le réseau contenant du texte, des images, des sons, des liens. L'accès aux documents se fait en utilisant un logiciel de consultation (browser ou navigateur) qui gère l'interface graphique de l'utilisateur et les liens permettant par un simple clic de la souris d'accéder au document ainsi désigné.pour plus d'information sur ce concept (cf.Annexe A).

Le concept de «client universel» du Web permet à n'importe quel client d'accéder à, l'ensemble des informations disponibles sur l'Internet en utilisant un ensemble de standards qui inclut des outils tels que les navigateurs, des règles et conventions d'adressages, des protocoles et différents formats. [1]

- Le **SERVICE** est un Processus capable de fournir des réponses à une requête ou à une interrogation sur un ordinateur distant. Un service de gestion de fichiers distants, par exemple, peut effacer ou transmettre un fichier en réponse à la demande d'un ordinateur client. Comme l'envoi d'une page HTML (HyperText Markup Language) pour un serveur HTTP (HyperTtext Transfer Protocol) qui fait fonctionner un service HTTP. [2].

I-1- Le Service Web:

Le Web a connu une rapide évolution, qui a donnée naissance aux Web Services, cette évolution est passée par les étapes suivantes :

1-<u>le document Web</u> : pour satisfaire les besoins des scientifiques, et partager leurs documents de recherches.

2-les serveurs Web : qui parlent du HTTP et délivrent de l'information en HTML afin de l'afficher dans les navigateurs.

3-les sites Web : crées pour que les entreprises puissent faire de la publicité pour leurs produits ainsi que leurs services.

4-les applications Web : créées pour que les entreprises fassent des transactions avec leurs clients.

5-les serveurs d'applications Web : qui ont acquis des capacités de la programmation distribuées, et arriver à faire des transactions sur plusieurs machines à la fois. Ces serveurs ont appris à parler le Protocole du Sans fil WAP (Wireless Access Protocol) et délivrent de l'information en WML (Wireless Markup Language), afin que les clients mobiles puissent participer à l'application Web.

C'est la floraison du e_business (electronic business)

6-le service Web : les entreprises voulaient intégrer leurs processus avec d'autres entreprises, donc le Service Web a été créé. Le Service Web est propulsé par les serveurs d'applications Web qui parlent le SOAP (Simple Object Access Protocol), et délivrent de l'information en XML (Extensible Markup Language).Ainsi, le Service Web est aujourd'hui dans son enfance.

La question qui se pose à présent le Service Web est-il le parent des Web Services ?

"The building block of the Service Web is the Web Service, a set of related application functions that can be programmatically invoked over the Internet." [3]

C'est à dire que : «*Le jeu de construction du Service Web est le Web Service, un ensemble de fonctions d'une application qui peuvent être invoqué sur l'Internet* » d'où le cri «*Software as Service*».Donc les Web services sont le noyau du service Web et l'intégralité des Web Services disponibles sur Internet est appelée le Service Web.

I-2- Présentation des Web Services :

Les Web Services prennent leur origine dans **l'informatique distribuée** et dans **l'avènement du Web**. Le but de l'informatique distribuée est de permettre à une application sur une machine d'accéder à une fonction d'une autre application sur une machine distante et ce, de la même manière que l'appel d'une fonction locale, indépendamment des plates-formes et des langages utilisés. Trois standards sont aujourd'hui utilisés : CORBA/IIOP (Common Object Request Broker Architecture / Internet Inter-ORB Protocol), DCOM (Distributed Component Object Model) et RMI (Remote Method Invocation). Ces modèles offrent des mécanismes de récupération d'espace mémoire, de sécurité, de gestion du cycle de vie des

objets, mais sont complexes, et difficilement interopérables entre eux. Ils restent donc souvent confinés à l'intérieur des entreprises.

Le second point qui a favorisé l'essor des Web Services, c'est le développement d'Internet par la démocratisation du haut débit, la structuration des données via XML et la recherche d'interopérabilité. Les entreprises publiaient déjà de l'information via des sites Web, utilisaient la messagerie et faisaient du commerce électronique, elle l'utilise maintenant pour leurs applications métiers (ressources humaines, ventes, finances, etc.).(cf.I-1).

Les Web Services sont la technologie logicielle qui s'impose aujourd'hui comme une réalité incontournable ; appelée aussi les **Services Web XML** Elle repose sur un ensemble de spécifications et normes conçu, adopté et soutenu par tous les grands acteurs du marché (Microsoft, IBM, Sun Microsystems,Hewlett-Packard, ...), qui travaillent sur le sujet dans le cadre du W3C (*World Wide Web Consortium)* et aussi, depuis janvier 2002, dans le cadre du nouveau consortium WS-I (*Web Services Interoperability)*.

Ils représentent une infrastructure logicielle de communication qui permet aux applications et aux systèmes d'informations des entreprises, administrations et associations.de proposer, rechercher, fournir, consommer, échanger des services informatisés sur Internet ; et qui appartient à un espace global d'échange et de dialogue ,ce dit espace s'appuie sur le Web (en intranet et sur Internet) et se déploie sur la base de cette infrastructure ainsi que la technologie normalisée XML pour le format de données.

Les Web services ne sont pas un standard, mais plutôt un ensemble de standards associés à trois spécifications XML :

-SOAP : (Simple Object Access Protocol) : le protocole de transport des données.

-WSDL (Web Service Description Language) : le langage de description des services offerts.

-UDDI (*Universal Description Discovery and Interoperability*) : annuaire pour le référencement des services par les fournisseurs et leur découverte par les utilisateurs.

Ce sont les standards de base pour les Web Services, d'autres standards dérivés existent en compléments pour les Web Services (cf Annexe B).

objets, mais sont complexes, et difficilement interopérables entre eux. Ils restent donc souvent confinés à l'intérieur des entreprises.

Le second point qui a favorisé l'essor des Web Services, c'est le développement d'Internet par la démocratisation du haut débit, la structuration des données via XML et la recherche d'interopérabilité. Les entreprises publiaient déjà de l'information via des sites Web, utilisaient la messagerie et faisaient du commerce électronique, elle l'utilise maintenant pour leurs applications métiers (ressources humaines, ventes, finances, etc.).(cf.I-1).

Les Web Services sont la technologie logicielle qui s'impose aujourd'hui comme une réalité incontournable ; appelée aussi les **Services Web XML** Elle repose sur un ensemble de spécifications et normes conçu, adopté et soutenu par tous les grands acteurs du marché (Microsoft, IBM, Sun Microsystems,Hewlett-Packard, ...), qui travaillent sur le sujet dans le cadre du W3C (*World Wide Web Consortium)* et aussi, depuis janvier 2002, dans le cadre du nouveau consortium WS-I (*Web Services Interoperability)*.

Ils représentent une infrastructure logicielle de communication qui permet aux applications et aux systèmes d'informations des entreprises, administrations et associations.de proposer, rechercher, fournir, consommer, échanger des services informatisés sur Internet ; et qui appartient à un espace global d'échange et de dialogue ,ce dit espace s'appuie sur le Web (en intranet et sur Internet) et se déploie sur la base de cette infrastructure ainsi que la technologie normalisée XML pour le format de données.

Les Web services ne sont pas un standard, mais plutôt un ensemble de standards associés à trois spécifications XML :

-SOAP : (Simple Object Access Protocol) : le protocole de transport des données.

-WSDL (Web Service Description Language) : le langage de description des services offerts.

-UDDI (*Universal Description Discovery and Interoperability*) : annuaire pour le référencement des services par les fournisseurs et leur découverte par les utilisateurs.

Ce sont les standards de base pour les Web Services, d'autres standards dérivés existent en compléments pour les Web Services (cf Annexe B).

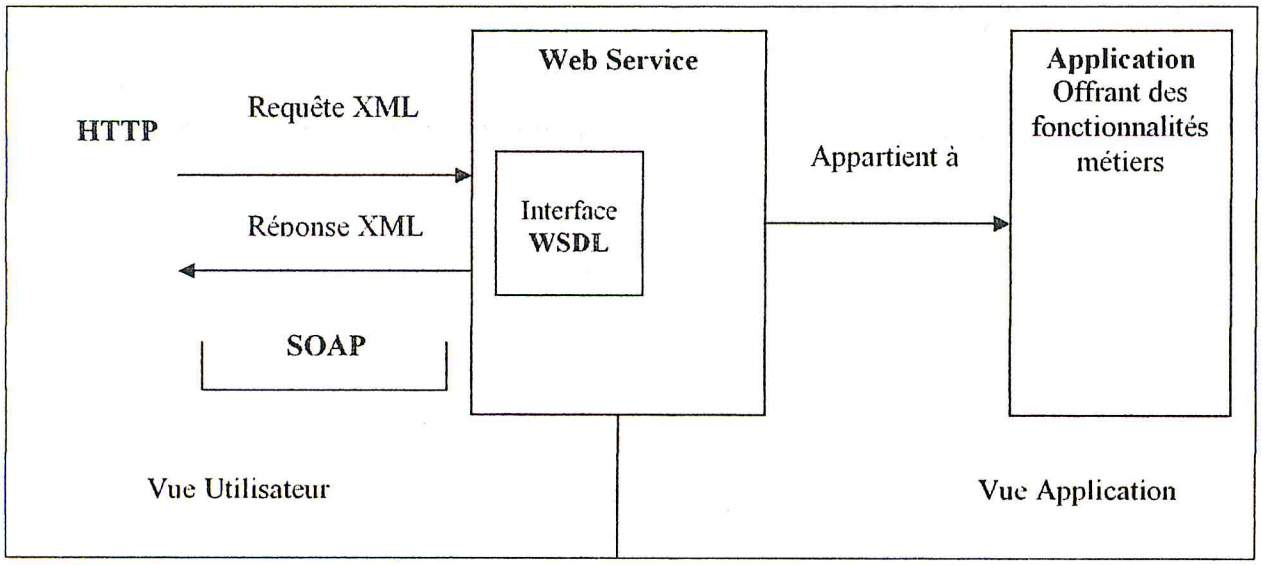


Figure I- 1–principe des Web Services

Les Web services sont des composants logiciels encapsulant des fonctionnalités métier de l'entreprise et accessibles via des protocoles standards du Web. Un Web service est décrit dans un document WSDL , précisant les méthodes pouvant être invoquées, leur signature, et les points d'accès du service (URL, port, etc.). Ces méthodes sont accessibles via SOAP: la requête et la réponse sont des messages XML transportés par HTTP. Un service Web est accessible depuis n'importe quelle plate-forme ou langage de Programmation.

I-3- L'innovation des Web Services :

L'idée n'est pas nouvelle en soi : il s'agit d'une norme universelle de communication entre systèmes informatiques répartis sur un réseau. En revanche, l'approche est bien nouvelle sur quatre points importants :

1-les Web Services *sont totalement compatibles avec les technologies, les outils, les pratiques courants sur Internet aujourd'hui*. Mais les spécifications des Services Web XML sont aussi indépendantes de ces technologies et pratiques, ce qui leur permet d'intégrer facilement les évolutions des protocoles de base.

2- Les technologies Services Web XML *sont totalement indépendantes des spécificités de la mise en œuvre des applications et systèmes d'information participant à l'échange* (l'architecture logicielle, le langage d'implémentation, le système d'exploitation, la technologie des bases de données).

3- Ils sont totalement *« non intrusives »*. Non seulement les systèmes participant à l'échange ne sont obligés en aucun cas de connaître les choix de mise en oeuvre de leurs

interlocuteurs, mais, en plus, aucune installation de technologie dépendante de ces choix n'est demandée (« couplage faible » entre applications et systèmes). Les Services Web XML permettent notamment l'échange « transparent » de services informatisés entre systèmes de technologie **J2EE** (normalisée par Sun Microsystems et mise en œuvre par des grands éditeurs comme IBM et BEA Systems) et systèmes de technologie **dotNet** (NET développée par Microsoft).

4- *les clients et serveurs n'ont pas besoin d'implémenter leur propre format propriétaire pour utiliser et publier des méthodes publiques* dans l'intégration des applications au niveau méthode/fonction Les RPC (Remote Procedure Call) et les API (Application Programming Interface) ne permettent pas aux serveurs de présenter aux clients, de façon standard, leurs méthodes publiques. Chaque programme serveur peut avoir sa propre implémentation. Par contre, les Web services sont toujours exposés par les serveurs dans un format standard utilisant le langage WSDL (Web Services Definition Language).

I-4- L'entreprise et les Web Services:

Les Web Services permettent d'ouvrir les applications, y compris les plus anciennes, pour les exploiter dans toute sorte de nouveaux processus métiers. Ce qui fait que l'intégration d'application est devenue sans limites, L'une des entreprises clientes s'est appuyée sur les Services Web pour mettre en place un canal de vente révolutionnaire au moyen d'un antique HP 3000 ! Ce système propriétaire, datant d'il y a vingt ans, voit ainsi sa durée de vie prolongée dans le 21ème siècle et ce, grâce aux Web Services.

La séparation de l'interface et de l'exécution est au cœur des Services Web. C'est ce principe qui libère les applications pour autoriser leur intégration généralisée dans de nouveaux processus. Les entreprises ont la possibilité d'encapsuler des applications et des processus métiers entiers sous forme de Web Services, puis de relier ceux-ci entre eux pour créer de nouveaux processus.

Une fois encore, ce principe reprend les atouts mis en avant pour les composants logiciels, mais avec bien plus de souplesse et de puissance du fait que les Services Web favorisent l'interopérabilité de différentes plates-formes : Java, .NET, CORBA...etc. Il en résulte une transformation complète du marché de l'intégration des applications, avec la disparition des risques élevés et du manque de souplesse liés à la précédente génération de solutions d'intégration. Par exemple, si une entreprise doit intégrer sa gestion de stock avec les systèmes de commande de ses partenaires, *les Web Services rendent une telle intégration*

possible au moyen des standards Internet. Inutile pour cela que l'entreprise et ses partenaires disposent des mêmes logiciels. Autre exemple : lorsqu'il s'agit de créer un nouveau canal de vente en ligne afin de générer davantage de revenus, les Web Services permettent d'adosser ce canal aux applications existantes, ce qui réduit les délais et les coûts de mise en œuvre, tout en préservant la cohérence de l'entreprise. Pour d'autres exemples (cf.Annexe E).

Deux domaines d'utilisation des Web services sont mis en avant par les acteurs du marché : la location d'application (ASP Application Service Providing) et les applications distribuées. Le premier domaine permet par exemple à un particulier de consulter son agenda en ligne, son portefeuille, de sauver des documents sur un serveur distant, etc. C'est un modèle économique qui pour l'instant ne décolle pas.Quant au second, il permet aux entreprises de passer d'un modèle de commerce interne à un modèle externe. Au lieu de seulement automatiser leurs processus de commerce interne, elles construisent des sites Web qui intègrent à la fois leurs Web services mais aussi ceux de leurs partenaires. Cette collaboration est appelée «Automatisation du commerce Internet ».

I-5- Conclusion :

Les Web services présentent plusieurs avantages qui sont les suivants :

- -L'indépendance au type de client, au système d'exploitation, au langage de développement, au logiciel, au modèle objet, au serveur supportant les composants des services web.
- Réduction des coûts d'encadrement par automatisation interne et externe des processus commerciaux.
- Réduction des efforts humains et techniques pour interfacer les processus commerciaux des entreprises par l'utilisation d'interfaces standardisées et auto-descriptives (XML+WSDL).
- Un code de développement écrit plus rapidement, plus stable, plus flexible, plus simple.
- -Grâce à des standards d'interopérabilité, ces technologies uniformisent la présentation des services offerts par une entreprise et rendent l'accès transparent à tout type de plate-forme : ce qui permet leur consommation via n'importe qu'elle plate forme.
- -Livrer à l'utilisateur final une agrégation de services intelligents.
- -Outils puissants d'intégration entre applications distantes.

- -Ils offrent le passage réussi de l'intégration intra-entreprise d'aujourd'hui à celle de l'inter –entreprise de demain qui nécessite des modèles de procédures métiers impliquant plusieurs Web Services.

Une constatation faite de manière plus nuancée par l'auteur «Jean-Marie Chauvet» qui distingue la vision stratégique à long terme des **Web Services** qui devraient permettre d'orchestrer les services de l'entreprise étendue grâce à l'exposition de leurs processus métiers et une vision tactique des Web Services par les développeurs qui les utilisent comme des RPC ;Notant que la vision tactique des Web Services est déjà bien réalisée, cet auteur souligne le chemin qui reste à accomplir pour réaliser la vision stratégique et notamment le risque de "babélisation" de Web Services qui ne serait pas compréhensible.

Divergence des discours techniques et marketing ou divergence des visions tactiques et stratégiques? Le mot de la fin reviendra peut-être aux réalisations concrètes qui suivent parfois des chemins imprévus.

Chapitre II : Etude du langage XML

XML (Extensible Markup Langage) ou langage extensible de balisage, standardisé par le W3C en février 1998, adopté très rapidement par les éditeurs de progiciels (logiciels professionnels) d'entreprise, c'est un format d'intégration facilitant la migration des document vers l'Internet. XML a permis à Internet de se transformer en une plateforme informatique intégrée, ainsi participe dans le développement du Web en permettant à tous les acteurs industriels de partager une base commune XML.

Les raisons d'un tel consensus autour d'une norme technique sont à chercher du côté de la simplicité et de la richesse d'expression d'XML :

- XML repose sur le balisage d'un flux de texte, comme HTML.
- XML est un format pratique de sauvegarde de données.
- XML est un format pratique d'échange de données, à la fois entre programmes (de serveur à serveur) et entre programme et utilisateur (de serveur à client).
- XML est un format de sauvegarde de documents.
- XML est le format standard sur le Web pour les applications d'entreprises [4].

XML résout les problèmes d'incompatibilités de l'ancienne génération de développement d'application réparties à base de composants logiciels, qui intègre des applications internes au système d'information de l'entreprise, et aussi impose une infrastructure et des modèles composants et d'assemblage d'applications.

Les Web Services sont l'étape suivante de l'intégration d'applications interentreprises via le réseau d'information du Web, ils ont l'avantage de ne pas imposer une infrastructure car elle est déjà en place, l'interopérabilité règne et s'impose grâce à l'ubiquité et l'universalité du Web. Aujourd'hui, les efforts de recherche et de standardisation XML ne portent plus sur la fonctionnalité de publication de documents; mais sur les scénarios d'usage de documents XML dans l'intégration d'application intra et interentreprises.

II-1-XML le langage :

XML reprend les idées contenues dans HTML et SGML (Standard Generalized Markup Langage); les balises HTML sont prédéfinies par la norme, elles décrivent l'aspect graphique de la page Web, tandis que XML permet de définir et d'interpréter des balises.

Il est conçu pour des documents complexes mais reposant sur des principes simples et clairs :

- Lisibilité à la fois par les machines et par les utilisateurs.

- Définition sans ambiguïté du contenu d'un document.
- Définition sans ambiguïté de la structure d'un document.
- Séparation entre documents et relations entre documents.
- Séparation entre structure du document et présentation du document.

Pour ces deux derniers principes, XML fait recours à des standards dérivés de cette même norme XML. (cf.Annexe C).

II-1-1- Anatomie d'un document XML :

On distingue deux classes de documents : les documents **bien formés** (syntaxiquement correctes) et les documents **valides** (obéissant explicitement à une DTD).

La structure du document comprend un prologue qui contient des déclarations (la version, le codage des caractères…etc.).

Un document XML est une collection de balises qui constituent l'arbre d'éléments, il prend en général la structure suivante :

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <racine>
    <mabalise monattribut="valeur1" />
    <mabalise monattribut="valeur2" />
  </racine>
```

Figure II- 1- structure générale d'un document XML.

On utilise cette structure pour écrire un document XML bien formé, par exemple:

```
1  <?xml version="1.0" ?>                                   2
   <!-- Fichier Documentation.txt -->                       3
   - <Documentation>
     - <LIVRE>
         <TITRE>Services Web avec SOAP, WSDL, UDDI, ebXML</TITRE>
         <AUTEUR>Jean-Marie Chauvet</AUTEUR>
         <PAGES>300</PAGES>
         <PRIX>3000 DA</PRIX>
4      </LIVRE>                                              5
     - <LIVRE>
         <TITRE>Inside ASP.NET Web Matrix</TITRE>
         <AUTEUR>Homer A et Sussman D</AUTEUR>
         <PAGES>98</PAGES>
         <PRIX>1800 DA</PRIX>
       </LIVRE>
   </Documentation>
```

Figure II- 2- Anatomie d'un document XML

Description :

1- **Le prologue** : contient des déclarations et des commentaires.

2- **Déclaration XML** : définit la version d'XML. Pour les codages (cf.Annexe D).

3- **Un commentaire.**

4- **La racine :** du document, qui délimite le début et la fin du document.

5- **Les éléments** contenus dans le document.

Cette structure définit comment écrire un document bien formé pour cela il faut respecter les règles de syntaxes suivantes :

1. le document doit avoir un unique élément racine qui contiendra les autres éléments.
2. si un élément est déclaré fils d'un autre élément, il doit être fermé (clos) dans le même élément père.
3. chaque élément doit avoir une balise d'ouverture et une balise de fermeture.
4. le nom de l'élément dans la balise d'ouverture est le même dans la balise de fermeture.
5. le nom de l'élément est sensible à la casse par exemple : <TITRE>Web Service</titre>; < !- - élément illégal-->

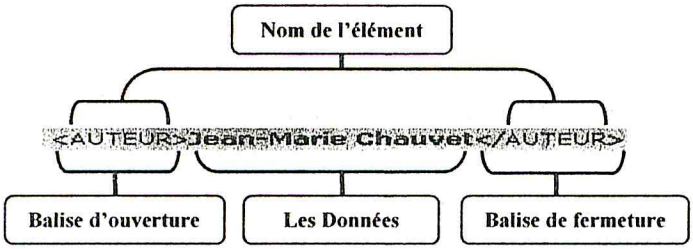


Figure II- 3-Structure d'un élément simple.

Chaque élément peut avoir des fils : l'élément père <LIVRE> possède quatre éléments fils (TITRE, AUTEUR, PAGES, PRIX).

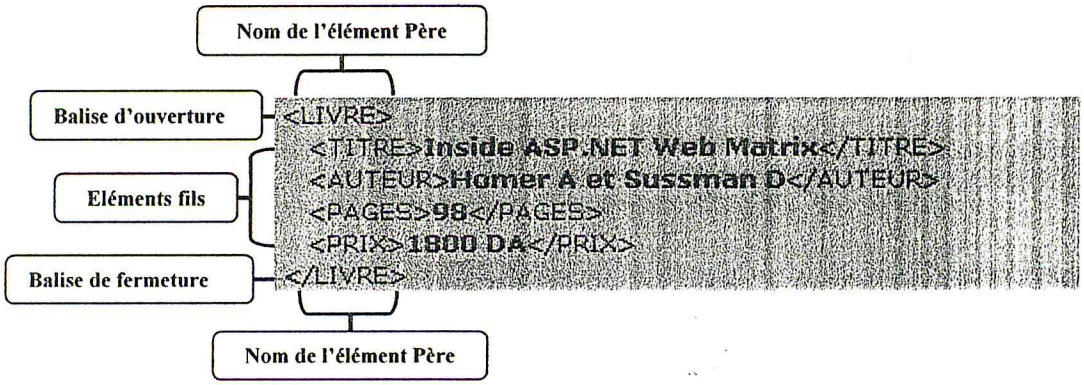


Figure II- 4- Structure d'un élément composé.

Les éléments peuvent contenir des attributs, pour spécifier leur valeur, voir figure ci-dessous :

```
<?xml version="1.0" encoding="utf-8" ?>
<PC>
  <computer name="Portable HD40G-GC32M" caracteristiques="40 Go de Disque dur." />
  <computer name="PC HD120G-GC32M" caracteristiques="120 Go de Disque dur." />
  <computer name="Portable HD20G-GC64M" caracteristiques="20 Go de Disque dur." />
  <computer name="PC HD60G-GC64M" caracteristiques="60 Go de Disque dur." />
  <computer name="PC HD120G-GC16M" caracteristiques="120 Go de Disque dur." />
  <ram name="128 Mo" price="2500" />
  <ram name="256 Mo" price="4500" />
  <ram name="512 Mo" price="8500" />
  <speed name="750 MHz (35000 DA)" price="35000" />
  <speed name="1,0 GHz (40000 DA)" price="40000" />
  <speed name="1,7 GHz (45000 DA)" price="45000" />
  <speed name="2,2 GHz (50000 DA)" price="50000" />
</PC>
```

Valeur

Attributs

Figure II- 5- Document XML avec attributs

Une spécification d'attribut est une paire nom-valeur, associée à l'élément; c'est une nouvelle façon d'inclure de l'information dans un élément (il peut être vide comme dans l'exemple ci-dessus l'élément 'computer'); car l'attribut permet de stoker une variété de propriétés d'éléments. Les règles pour créer des attributs sont les suivantes :

-le nom de l'attribut débute par une lettre ou le caractère souligné.

-on peut assigner le nom de l'attribut à un Namespace, en plaçant ce dernier comme préfixe, suivi du caractère (:) ensuite le nom de l'attribut.

Les règles pour des valeurs valides d'attributs : la valeur est une série de caractères délimitées par des cotes

-la chaîne peut contenir des références à des caractères ou à des entités internes ou générales.

-la chaîne ne peut contenir des balises.

Vous pouvez ouvrir un document XML directement dans le navigateur Internet Explorer, tout comme une page Web HTML, en donnant au document une extension (.xml) ; le navigateur affiche alors le document avec des balises structurées automatiquement dans un arbre d'éléments.XSTL (eXtensible Steel sheeT Langage): est la norme qui donne des techniques plus robustes d'affichage de document XML.

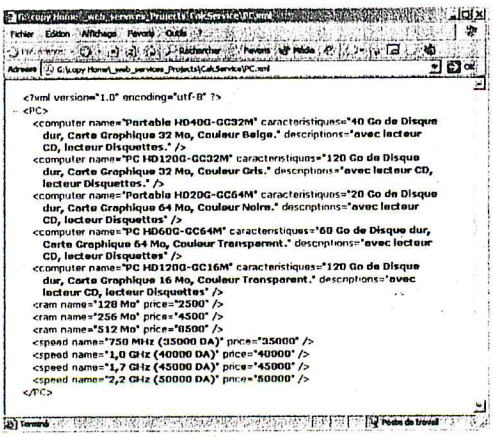


Figure II- 6-Affichage d'un document XML.

II-1-2- Les DTD (Data Type Definition) :

Afin de distinguer sans ambiguïté le contenu du document ; les DTD ou structure type de document ont été introduite dans la norme XML, pour définir les balises c'est à dire la structure d'une famille de documents XML ; Un document valide sera toujours une instance de la structure définie dans une DTD. Ainsi le contenu est séparé de la structure. La déclaration d'une DTD dans un document XML, se fait avec l'élément réservé « **DOCTYPE** » qui spécifie à quel type de document, le document XML fait référence. La DTD est un bloc de balises XML, qu'on ajoute dans le prologue d'un document valide .la déclaration d'une DTD prend cette forme générale :**< ! DOCTYPE Nom DTD>**; le nom spécifie le nom du document.

Exemple :

Exemple.dtd

```
ELEMENT  carte_visite (nom, titre, contacts)>
< ! ELEMENT  nom (prenom, initiale ?,nom_famille)>
< ! ELEMENT  prenom (#PCDATA)>
< ! ELEMENT  initiale (#PCDATA) >
< ! ELEMENT  nom_famille (#PCDATA)>
< ! ELEMENT  titre (#PCDATA)>
< ! ELEMENT  contacts (numero*)>
< ! ELEMENT  numero ((#PCDATA)>
```

Exemple.Xml

```
<?Xml version='1.0'  encoding='UTF-8' ?>
<!DOCTYPE xq PUBLIC  'exemple.dtd'>
< carte_visite >
     <nom>
          <prenom>Harry</prenom>
          <initiale>V</initiale>
          <nom_famille>Tuxen</nom_famille>
     </nom>
     <titre>Auteur</titre>
     <contacts>
        <numero>00 11 234 5678</numero>
         <numero>00 11 234 6789<numero>
     </contacts>
</carte_visite>
```

Figure II- 7-Un document XML et son DTD. [4]

Description : le document DTD définit les balises « **ELEMENT** » et leur imbrication dans les documents XML ; les caractères spéciaux comme « **?** » ou « ***** » marquent l'aspect facultatif ou la répétition éventuelle de certaines balises ; «**#PCDATA**» indique simplement la présence de texte (sans balises).

II-1-3- Les Domaines Nominaux (XML Namespaces) :

C'est une recommandation du W3C, qui résout le problème d'ambiguïté des balises, Dans un document XML, car ce dernier contient des balises réutilisables dans diverses applications.

Exemple :

```
<bonDeCommande:expedition
xmlns :bonDeCommande="http://www.ebXML.org">
<adresse>
61, bd. St Germain, 75005 Paris
</adresse>
</expedition>
<contacts:serviceClient xmlns: contacts ="intranet/contacts/xml">
<adresse>
info@eyrolles.com
</adresse>
</serviceClient>
```

L'exemple ci-dessus comporte la balise **<adresse>** qui est utilisée deux fois dans le document et comporte deux informations différentes. La première désigne une adresse postale d'expédition dans un bon de commande, la seconde désigne une adresse de courrier électronique dans une application intranet. Pour distinguer les deux types d'adresses, on associe un préfixe aux balises ou à l'attribut par « : » ; c'est le « Namespace », reconnu de manière unique grâce à la valeur de l'attribut réservé « **Xmlns** ».

-La balise **expedition** comporte le Namespace bonDeCommande.

-La balise **serviceClient** comporte le Namespace contacts.

Le dispositif des URI qui suit la valeur du Namespace, assure l'unicité.

II-1-4- XMl schéma:

Une recommandation du W3C, qui a marquée une innovation dans l'utilisation d'XML, la DTD ne définit que la structure du document, un schéma définit l'imbrication des éléments entre eux c'est-à-dire la structure ; (héritée des DTD) de plus le type des éléments et leurs attributs ; c'est une information plus riche que celle représentée par les DTD.XML schéma précise comment représenter en XML les structures de données en général.

Exemple :

Description du schéma :

-schema ; racine du schema xml.

-Xsd=http://www.w3.org/2001/XMLSchema ; le Namespace conventionnel des schémas XML. La spécification définit deux Namespaces pour XML schéma :

• « xsd » ou « s »associé à l'URI_http://www.w3.org/2001/XMLSchema.

• xsi_associé à l'URI_http://www.w3.org/2001/XMLSchema-instance.

- **TargetNamespace=http://www.livres.org** ; indique que les éléments du schéma appartiennent à ce Namespace cible par défaut.

La DTD : livre.DTD

```
< ! ELEMENT  Livre (titre, auteur, date, ISBN, editeur)>
< ! ELEMENT  titre (prenom, initiale ?,nom_famille)>
< ! ELEMENT  auteur (#PCDATA)>
< ! ELEMENT  date (#PCDATA)>
< ! ELEMENT  ISBN (#PCDATA)>
< ! ELEMENT  editeur (#PCDATA)>
```

Le schéma livre.xsd

```
<xsd :schema    xmlns:xsd='http://www.w3.org/2001/XMLSchema'
 TargetNamespace='http://www.livres.org'
xmlns='http//www.livres.org'>
<xsd :element  name='livre'>
<xsd :complexetype>
<xsd :sequence>
<xsd :element  ref='titre' minoccurs='1' maxoccurs='1'/>
<xsd :element  ref='auteur' minoccurs='1' maxoccurs='1'/>
<xsd :element  ref='ISBN' minoccurs='1' maxoccurs='1'/>
<xsd :element  ref='editeur' minoccurs='1' maxoccurs='1'/>
</xsd :sequence>
</xsd :complexetype>
</xsd :element>
<xsd: element name='titre' type='xsd:string'/>
<xsd: element name='auteur' type='xsd:string'/>
<xsd: element name='date' type='xsd:string'/>
<xsd: element name='ISBN' type='xsd:string'/>
<xsd: element name='editeur' type='xsd:string'/>
</xsd:schema>
```

- **xmlns='www.livres.org'** ; Namespace par défaut.

-l'information associée aux balises identifie la nature du texte : le type utilisé est chaîne de caractères **String**, il appartient aux types primitifs (booléens, entiers, réels, chaînes de caractères, dates, entités, notations des DTD), la spécification permet à l'utilisateur de définir des types composés pour modéliser des données complexes, cela en utilisant la balise prédéfinie **Complexetype .**

-**minoccurs,maxoccurs** : l'information sur l'élément est plus précise en indiquant le nombre d'occurrences (minimum ;maximum).

-un autre outil que XML permet d'utiliser, c'est définir d'autres types, plus complexes à l'aide de la balise **complexetype** qui indique où commencer la définition du nouveau type dans le même document.

-En général, la valeur de l'attribut **ref** doit être celle d'un élément global, c'est à dire un élément déclaré directement à l'intérieur du schéma et non à l'intérieur d'une définition de type complexe, selon l'exemple l'élément **titre** est défini et référencé dans le même schéma.

L'instance du schéma est la suivante :

Document XML : livre.xml

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

<livre xmlns='http//www.livres.org'
xmlns='http://www.w3.org/2001/XMLSchema-instance'
xsi :schemalocation='http://www.livres.org/livre.xsd'>

<titre>Forsyth</titre>

<auteur>Harry V.</auteur>

<ISBN>102NR23RP34B1p16p1p2PK1p2PK1p12P1n32nlk</ISBN>

<editeur>Skakbladet</editeur>

</livre>
```

-xmlns='http//www.livres.org' ; le Namespace par défaut.

-xmlns='http://www.w3.org/2001/XMLSchema-instance' indique où trouver le schéma pour ce document.

-cette instance « livre.xml »qui est conforme au schéma, utilise les balises définies dans le schéma «livre.xsd ».

Des validateurs de schéma xml sont utilisés pour vérifier la conformité d'un document XML à un schéma, de façon automatique, donc ils déchargent le code de l'application qui exploite les documents de cette tache.

II-2- XML et les Web Services :

Les Web Services et les applications communiquent via des messages XML.

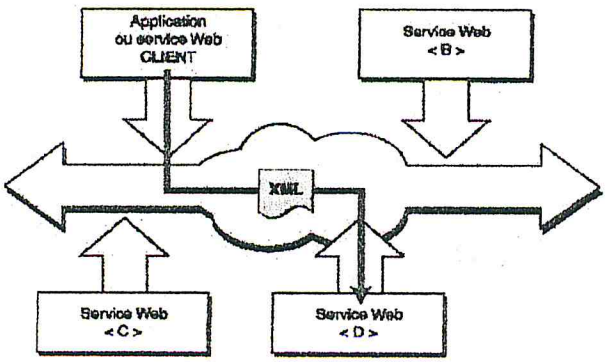


Figure II- 8- L'information entre Web services [4]

Dans le mécanisme de transport entre Web services, les requêtes, leurs résultats et les erreurs éventuelles résultant de leur invocation sont tous écrits en XML. Ce sont donc des documents émis par le client vers le service Web et traités sur le serveur en réponse aux demandes.

Notons d'emblée que le choix de documents XML comme format pour les données circulant entre applications clientes et Web services ou entre Web services présente des avantages sur les middlewares de la génération précédente tels que Corba, RMI ou DCOM. Il n'y a pas de programmation à proprement parler et l'on est loin des mécanismes de compilation particuliers que le recours à des RPC ou à de tels middlewares imposent aux programmeurs [4].La conséquence de remplacer une requête traditionnelle par un document texte est que le document XML devient plus complexe car il faut décrire la destination exacte de l'appel, préciser les arguments éventuels et leur type, la ou les valeurs attendues en retour et leur type, les codes d'erreur et, plus généralement, expliciter toute information de contexte requise par le serveur. XML est de taille pour représenter les données passées entre Web services qui sont a priori complexes (références à d'autres messages, types de données complexes, pointeurs, etc.) ; corrélativement on a prévu un « interpréteur XML, ou parseur », qui décode les balises XML à la réception des messages XML, et qui exécute les commandes correspondantes, ensuite les mécanismes de traitements sont déclanchés. La performance est un peu plus considérable à ce niveau ,car un document XML peut à lui seul remplacer une multitude d'appels RPC traditionnels au travers d'un middleware.

La standardisation se ressent aussi à ce niveau car l'utilisation d'un document XML n'est pas propriétaire à un protocole particulier ou une infrastructure particulière,les anciens systèmes imposaient un protocole particulier,une configuration particulière,mais avec des messages XML le besoin se résume à un interpréteur XML.

II-3- Conclusion :

Les Namespaces et XML Schema, sont des recommandations du W3C, c'est une preuve de standardisation et de richesse en représentation pour XML. Un langage flexible et facile à employer. Se sont deux spécifications techniques qui indiquent comment utiliser XML dans des applications informatiques réparties sur le Web pour représenter et manipuler les abstractions indispensables mais indépendantes du secteur d'application.

Pour les Web Services, XML est le langage de base des standards qui accompagnent cette technologie ; le protocole SOAP, et le langage WSDL reposent sur XML .Ces deux nouveaux standards décrivent le fonctionnement de base d'un échange avec les Web Services et leur

description mais aussi répondent à la question d'aujourd'hui ; qui est celle de pouvoir trouver d'autres scénarios d'usage pour XML ; pour cela une étude préalable de ce langage est utile.

Chapitre III : Etude du protocole SOAP

SOAP (Simple Object Access Protocol) est un protocole léger d'échange de données dans un réseau de pair à pair, c'est-à-dire décentralisé ; s'appuyant sur les standards d'Internet Simple et facile à implémenter dans les serveurs Web ou dans les serveurs d'applications, SOAP est indépendant des langages de programmation ou des systèmes d'exploitation employés pour l'implémentation des Web services. S'inspirant des générations précédentes de RPC et de RPC objet (Corba, DCOM), sa définition est formalisée dans des documents XML échangés sur le Web, sous forme de dialogue entre le client et le serveur. il a comme objectif de permettre la *normalisation des échanges de données au sein d'architectures distribuées orientées objet*, servir de protocole de communication sur les intranets, dans une optique d'intégration d'applications d'entreprise, et permettre la communication entre applications et Web services, en particulier dans un contexte d'échanges inter-entreprises, proposé par un groupe de sociétés Microsoft, DevelopMentor et Userland Software, IBM, Intel, le 8 mai 2001 diffusion des spécifications SOAP 1.1 par le W3C ensuite le 17 décembre 2001 le SOAP 1.2 Working draft du W3C.SOAP est simple et neutre,il n'impose pas l'utilisation d'une API particulière,ou l'utilisation d'un ORB tel que CORBA ou DCOM,ou aucun autre modèle de programmation,de plus il est fondé sur le langage XML qui utilise un format textuel.Donc il peut être utilisé dans tous les styles de communication : synchrone ou asynchrone, point à point ou multipoint, intranet ou Internet.

III-1-Architecture SOAP :

C'est un protocole qui peut être transporté par les protocoles réseaux, car il est conçu pour s'appuyer sur HTTP ou un autre protocole de transport de messages (comme SMTP), SOAP n'impose pas de modifications aux serveurs Web déjà installés sur la Toile, plutôt il en ajoute des extensions, aussi il peut s'interfacer avec des applications préexistantes grâce à XML Schéma en saisissant leurs structures de données même jugées complexes. Le protocole SOAP se décompose en quatre parties autonomes.

L'Enveloppe SOAP : définit l'ensemble qui décrit la structure du message, son contexte, son destinataire, son contenu et différentes options.

Les Règles d'encodage SOAP : définissent un mécanisme de *sérialisation* pouvant être utilisé pour échanger des données entre différentes applications, et la représentation des données d'une application dans le corps d'un message SOAP (en particulier leur structure) en

utilisant XML Schéma, sachant que la *sérialisation* est un mécanisme qui permet la lecture et l'écriture de concepts objets, d'où la transformation d'un appel de méthode en texte. Les règles de codage utilisent abondamment XML Schéma pour décrire la structure des données constitutives des messages SOAP.

La Représentation SOAP RPC : définit une convention représentant les procédures d'appels et de réponses définissant la succession des requêtes et des réponses. SOAP hérite des principes traditionnels de RPC ainsi se rapproche des protocoles antérieurs, comme DCOM dans Windows NT ou IIOP dans la norme Corba. Ce qui facilite l'intégration des différents protocoles et assure une excellente interopérabilité à moindre coût à la jonction de l'intranet et de l'Internet dans les réseaux d'entreprises.

l'utilisation de HTTP comme couche de transport des messages SOAP.HTTP est un protocole simple et robuste qui peut s'accommoder sur le Web où la qualité de service et la latence sont variables, le réseau résulte de l'agrégation de nœuds et de sous-réseaux hétérogènes partageant la couche TCP/IP la plus profonde.En s'appuyant sur HTTP, on ne se heurte pas aux problèmes de pare-feu(Firewalls) ou de configuration de réseau IP qui rendent parfois difficile le déploiement d'applications à objets répartis au-delà du périmètre du réseau d'entreprise.

III-1-1-Utilité de XML dans les messages SOAP :

SOAP dispose d'une richesse expressive permettant l'expression de messages entre applications et services Web, aussi complexes soient-ils Soap propose d'associer XML comme format neutre de représentation de données pour le protocole HTTP ayant la même fonction de NDR (Network Data Representation) et CDR (Common Data Representation) dans les protocoles objets DCOM et IIOP. XML a plus d'avantages que ces formats pour représenter textuellement des structures de données complexes (mieux qu'une représentation binaire qui est moins lisible); en permettant une large disponibilité pour les outils de développement ou d'interpréteurs facilitant son implémentation et sa diffusion et qui évolue progressivement sans réécriture systématique des programmes émetteurs ou destinataires; à l'aide des normes NameSpaces et XML Schema qui permettent l'extension ou la spécialisation.

III-2-La structure des messages SOAP :

Un message SOAP est un document XML constitué d'une enveloppe contenant un en-tête (*header*) facultatif et le corps (*body*) du message ; l'enveloppe contient le message qui peut

disposer de son propre langage, ou vocabulaire spécifique aux messages d'une application ainsi que le vocabulaire de l'enveloppe SOAP.

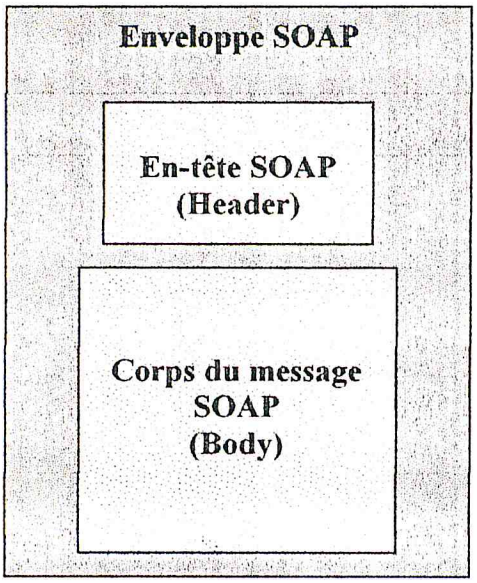


Figure III- 1- la structure du message SOAP [4].

III-2-1- L'enveloppe SOAP : <SOAP-ENV:Envelope>

La balise **<Envelope>**, représente la racine du document XML du message SOAP ;la spécification SOAP impose que cette balise et celles imbriquées soient explicitement associées à un namespace, de manière à supprimer toute ambiguïté. Deux namespace sont utilisés :

-**SOAP-ENV** associé à l'URI **http://schemas.xmlsoap.org/soap/envelope** pour définir le namespace de l'enveloppe dans la version 1.1, et à **http:// www.w3.org/2001/06/soap-envelope** dans la version 1.2 reprise par le W3C.

-**SOAP-ENC** associé à l'URI **http://schemas.xmlsoap.org/soap/encoding** pour la définition des formats de types de données dans la version 1.1, et à **http://www.w3.org/2001/06/soap-encoding** dans la version 1.2.

Exemple1 : message SOAP d'un programme client à un serveur offrant un service Web d'information financière en ligne pour connaître la valeur d'un titre :

```
<SOAP-ENV:Envelope
Xmlns:SOAP-ENV=«http://schemas.xmlsoap.org/soap/envelope"
SOAP-ENV: encodingStyle="http://schemas.xmlsoap.org/ soap/encoding">
<SOAP-ENV:Body>
<!-- Requête de la valeur d'un titre -->
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

La réponse du serveur aurait la même structure :

```
<SOAP-ENV:Envelope
Xmlns:SOAP-ENV= " http://schemas.xmlsoap.org/soap/
Envelope "
SOAP-ENV :encodingStyle="http://schemas.xmlsoap.org/
soap/encoding">
<SOAP-ENV:Body>
<!-- Valeur du titre -->
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

L'attribut **encodingStyle**, a comme valeur un URI, spécifie les conventions employées dans cet élément (et dans son contenu) pour le format des données. La spécification propose des formats identifiés par l'URI **SOAP-ENC**.

III-2-2-L'en-tête SOAP : <SOAP-ENV:Header> :

La balise **<Header>** élément facultatif, *pour passer dans le même message des informations complémentaires;* il Doit être le premier dans l'enveloppe SOAP du message. L'en-tête contient des informations authentifiant l'émetteur ou le contexte d'une transaction dont le message n'est q'une des étapes. Pour les couches de transport (comme FTP) ne fournissant pas, à l'émission, d'adresse de retour, ou bien pour identifier l'émetteur du message SOAP.L'élément **Header**, et les éléments qu'il contient, sont qualifiés par l'attribut **mustUnderstand** à valeur 0 ou 1.

La valeur 1 : le récepteur doit reconnaître l'information présente dans l'entête, son traitement est obligatoire.

La valeur 0 : l'en-tête peut être ignoré par le serveur, donc étendre les fonctions du serveur, par exemple, sans remettre en cause les clients.

Exemple2 : un en-tête transportant une adresse de courrier électronique :

```
<SOAP-ENV:Header
<exemple:emetteur xmlns:exemple=="http://monURI"
SOAP-ENV:mustUnderstand="0">
MonNom@monDomaine.com
</exemple:emetteur>
</SOAP-ENV:Header>
```

Un message SOAP peut « traverser » plusieurs intermédiaires avant d'atteindre son destinataire final. Dans ce cas, l'en-tête joue un rôle primordial, à chaque arrêt, le serveur intermédiaire extrait de l'en-tête ce qui le concerne en propre et ajoute ce qui est nécessaire au serveur intermédiaire suivant. Les éléments concernant un serveur intermédiaire sont marqués par la présence de l'attribut **SOAP-ENV: ACTOR** avec une valeur conventionnelle

SOAP-ENV:Actor=**http://schemas.xmlsoap.org/soap/actor/next**.

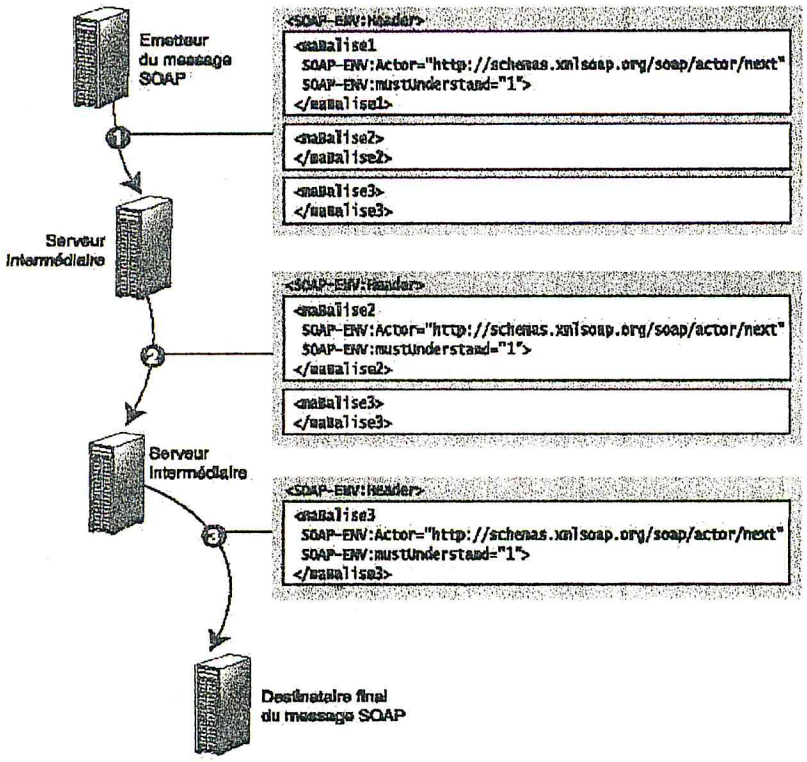


Figure III- 2- L'entête dans des transferts successifs du message SOAP

III-2-3-Le corps du message SOAP : <SOAP-ENV:Body>

Avec un seul élément **BODY**, contenant un ou plusieurs sous-éléments:

• **FAULT,** indiquant une erreur ou une défaillance en réponse à une requête.Dans ce cas, de nouveaux sous-éléments sont employés pour signaler le type d'erreur et pour renvoyer à l'émetteur des informations supplémentaires indiquant la raison de l'échec de l'appel. Les codes d'erreurs sont normalisés par la spécification SOAP.

• **Des données** destinées au destinataire du message, à un format défini par les règles de codage SOAP.

Au début le corps de message SOAP était conçu pour passer des appels de procédures distants, c'est-à-dire des requêtes au sens OMG (Object Management Group) du terme (un objet, une méthode et les valeurs de ses arguments) et des résultats ou des messages d'erreur. Mais, en fait, l'usage s'est élargi et le corps du message est souvent utilisé pour simplement passer des données structurées entre applications.

III-3- Les règles d'encodage des types de données :

Les règles d'encodage font appel à des balises définies dans le second namespace propre à SOAP **(http://schemas.xmlsoap.org/soap/encoding/).** Il est également possible de spécifier des namespaces de substitution, propres à l'utilisateur, pour définir ou redéfinir ces balises de codage de données en réponse à des besoins particuliers.

Toutes les valeurs des données d'une requête / réponse respectent les règles suivantes :

• Les valeurs **simples** (comme les entiers ou les chaînes de caractères) apparaissent directement entre les balises.

• Les valeurs dites **composées :** appelée un **accesseur** composé de champs qualifiés d'un nom, d'un numéro d'ordre, ou des deux. On définit les **structures**, où les accesseurs sont tous des **noms**, et les **tableaux,** où les accesseurs sont tous des **numéros d'ordre**.

• Les éléments représentant ces valeurs sont dits **inclus** ou **indépendants.** Un élément inclus dans une structure composée représente directement la valeur du champ considéré de la structure. Mais la valeur du champ peut également être un pointeur vers un élément, dit alors indépendant et qui peut être ainsi partagée par plusieurs éléments dans le corps d'un même message.

Exemple classique3 : une opération sur un compte bancaire défini, par exemple, par la méthode Java suivante sur le serveur :

```
package banque
public void operation{
public int compte;
public float montant ;
}
```

Une instance particulière de cette classe apparaîtrait alors dans le corps d'un message sous la forme de la structure SOAP suivante :

```
<t:operation xmlns:t="http://monServeur/banque">
<compte>125612</compte>
<montant>250.75</montant> </t:operation>
```

Les accesseurs sont les noms **Compte** et **Montant** utilisés comme balises. Une transaction **débit-crédit**, à peine plus complexe La classe Java correspondante étant :

```
package banque
public class transaction{
public operation credit;
public operation debit;
}
```

La structure SOAP est la suivante :

```
<t:transaction xmlns:t="http://monServeur/banque">
<credit>
<compte>125612</compte>
<montant>250.75</montant>
</credit>
<debit>
<compte>125760</compte>
<montant>250.75</montant>
</debit>
</t:transaction>
```

Remarque1: Une fois les structures définies, l'appel d'une méthode Java sur ces opérations est effectué.

Les finesses supplémentaires de ces règles générales sont : L'apport dans la représentation en SOAP de **la nullité** ou de **l'absence de certains champs d'une valeur composée**, qui fait appel à des mécanismes standardisés de représentation des données complexes, traités par la norme XML Schema. Le cas dans la situation où la valeur d'un champ d'une structure se trouve être une instance d'un sous-type du type prévu (il suffit d'imaginer par exemple une sous-classe de la classe **Operation** contenant des champs supplémentaires : ces derniers doivent alors figurer également dans le message SOAP).

Exemple4 : En ajoutant au scénario bancaire une seconde transaction **débit-crédit** faisant circuler le même montant entre trois comptes. La représentation est la suivante :

```
<t:double-transaction xmlns :t="http://monServeur/banque">
<transaction-B-vers-C>
<credit>
<compte>125612</compte>
<montant>250.75</montant>
</credit>
<debit>
<compte>125760</compte>
<montant>250.75</montant>
</debit>
</transaction-B-vers-C>
<transaction-A-vers-B>
<credit>
<compte>125760</compte>
<montant>250.75</montant>
</credit>
<debit>
<compte>125900</compte>
<montant>250.75</montant>
</debit>
</transaction-A-vers-B>
</t:double-transaction>
```

Défaut : Cette représentation ne détecte pas le compte B, intermédiaire, qui est le même dans les deux transactions, la première fois au débit, la seconde au crédit.

Solution :

Il faudrait, utiliser les éléments indépendants qui ont pour rôle de factoriser la structure commune. La représentation SOAP devient:

```
<t:double-transaction xmlns:t="http://monServeur/banque">
<transaction-B-vers-C>
<credit>
<compte>125612</compte>
<montant>250.75</montant>
</credit>
<debit href="#id1" />
</transaction-B-vers-C>
<transaction-A-vers-B>
<credit href="#id1" />
<debit>
<Compte>125900</compte>
<Montant>250.75</montant>
</debit>
</transaction-A-vers-B>
</t:double-transaction>
<t:operation soap:id="id1" xmlns:t="http://monServeur/
banque">
<compte>125760</compte>
<montant>250.75</montant>
</t:operation>
```

Résultat : le compte intermédiaire est définit à part où la transaction principale fait référence par **l'identifiant id1**.

Remarque 2: un **accesseur** peut être soit :

-une constante :entier,chaine de caractére...(comme dansl'exemple ci-dessus **<montant>250.75</montant>**).

-un URI agissant comme *un pointeur* (comme dans l'expression **<debithref="#id1"/>**). Cas, où plusieurs valeurs composées peuvent alors faire référence au même élément indépendant à l'intérieur d'un message SOAP.

Exemple 5:un tableau et de sa représentation dans un message SOAP. Le tableau suivant, définit dans l'IDL de DCOM, décrit un ensemble de points :

```
struct POINTLIST{
long cElem;
[size_is(cElem)] POINT points[];
};
```

Il sera représenté dans un message sous la forme d'un tableau SOAP, pour un jeu donné de valeurs :

```
<t:POINTLIST xmlns:"uri-de-POINTLIST">
<cElem>3</cElem>
<points xsd:type="t:POINT[3]">
<POINT><x>3</x><y>4</y></POINT>
<POINT><x>5</x><y>6</y></POINT>
<POINT><x>7</x><y>8</y></POINT>
</points>
</t:POINTLIST>
```

L'attribut **xsd** dans l'élément **points** fait référence à la norme XML Schema. Il permet de stipuler le type, au sens de la structure de données, du champ représenté par l'élément points. Dans l'exemple, ce schéma est supposé défini dans l'URI du namespace spécifié dans la balise de l'élément **t:POINTLIST.**

III-4-Les pièces jointes :

Dans ce cas les données ne sont pas au format d'une représentation textuelle ; ce qui fait appel à un mécanisme standard d'inclusion du message SOAP dans un message MIME composite *(multipart /related message).*le message SOAP doit apparaître le premier avec le type : **text/xml** dans le message MIME dans l'entête *multipart /related.* Son traitement se fait selon les règles de la norme SOAP.L'entête MIME comportant l'attribut : *content –ID* où bien l'attribut *Content-Location* désigne le reste des éléments du message MIME.

-Les références aux pièces jointes : se font depuis l'entête ou depuis le corps d'un message SOAP ; en utilisant l'astuce des règles de codage des messages SOAP. (Selon la remarque 2 où l'accesseur est une référence URI).la norme se charge de stipuler les conventions de construction d'URI qui pointent vers les pièces jointes au message MIME encapsulant le message SOAP.Le message SOAP fait référence aux URI qui identifient chaque partie du message *multipart /related :*

- Si l'attribut Content-Location est présent il contient un URI absolu utilisé comme identifiant de la partie MIME en question (le plus simple).

- Si ce même attribut est présent mais contient un URI relatif, l'URI absolu est constitué en référence au plus proche contenant MIME qualifié par un Content-Location absolu ou en référence au défaut « thismessage:/ » en cas d'absence d'une telle racine (pour les pièces jointes complexes ou imbriquées).

• Si l'attribut Content-ID est présent, il existe une méthode normalisée de construction d'un URI à partir de sa valeur (RFC 2111).

Un message SOAP peut parfaitement ne pas faire référence aux pièces jointes groupées dans le même message MIME,la résolution effective des URI n'est pas du rôle de l'interpréteur SOAP lui-même mais en fonction du traitement du message, le service Web sollicité aura à résoudre ou non ces références.

Le protocole SOAP avec pièces jointes provoque quelques changements mineurs dans les requêtes d'après le protocole de transport HTTP ou SMTP :

HTTP:L'attribut Content-Type:Multipart/Related apparaît dans l'en-tête HTTP à l'exclusion de tous les attributs HTTP redondants avec ceux de MIME.

SMTP : Les en-têtes SMTP et MIME sont simplement fusionnés (pas de redondance).

SOAP avec pièces jointes a permis d'attacher des documents non textuels ; plutôt des données binaires, par exemple dans le secteur du bâtiment des dessins d'architectes doivent circuler, ou bien dans le domaine du tourisme où des photos au format GIF ou JPEG doivent circuler

III-5-Le routage des messages SOAP « WS-Routing » :

-le routage des messages : c'est la détermination des chemins de diffusion des messages, car il faut assurer à la fois : un bon ordre de transmission des messages et la rémission éventuelle de ceux qui se sont égarés.

-WS-Routing(Web Services Routing Protocol) : c'est une spécification d'Octobre 2001, affinement de (SOAP-RP)SOAP Routing Protocol .

-fondé sur SOAP définit un échange unidirectionnel entre emetteur et récepteur au travers un nombre d'intermédiaires.

-construit le chemin de retour automatique aux réponses –récepteur vers l'émetteur originel.

-établissement de tout mode de communication avec SOAP : point à point, requête /réponse, accusé de réception, notification des défaillances.

- rajouter des informations à l'entête inclus dans l'enveloppe SOAP ; grâce à cette simplicité le protocole est relativement indépendant de la couche de transport de transmission des messages (TCP, UDP, HTTP).

-en conséquence : une corrélation de divers messages entre serveurs.

III-5-1-Modèle de routage des messages WS-Routing :

Dans le protocole SOAP l'attribut « **SOAPactor**» indique le destinataire final du message (ou partie du message) ;il a l'avantage d'expliciter la séquence d'intermédiaires du même message pour cela il fournit automatiquement les informations nécessaires pour corréler les échanges de messages entre les nœuds de traitements SOAP.

III-5-1-1-Spécification WS-Routing : [4]

Rôle	Commentaire
En-tête WS-Routing	En-tête SOAP utilisé pour conserver l'information de routage, associé au Namespace http:// schema xmlsoap.org/rp
Message WS-Routing	Message SOAP contenant un entête WS-Routing
Défaillance WS-Routing	Notification de défaillance SOAP contenant un entete WS-Routing
Emetteur /Recepteur WS-Routing	Application créant ou recevant un message WS-Routing lié à un protocole de transport de donnée
Chemin WS-Routing	Le chemin et le chemin de retour WS-Routing sont constitués de la séquence (et de la même en ordre inverse) des émetteurs et des récepteurs WS-Routing au travers circule un message WS-Routing
Chemin de retour WS-Routing	Ce chemin est construit automatiquement par le protocole lors de la transmission initiale de l'émetteur au récepteur.ce chemin est éventuellement utilisé pour faire parvenir à l'émetteur les réponses du récepteur.
Intermédiaire WS-Routing	Un nœud jouant à la foi le rôle d'émetteur et de récepteur WS-Routing
Proxy WS-Routing	Intermédiaire jouant un rôle explicite dans la transmission des messages WS-Routing
Tunnel WS-Routing	Intermédiaire WS-Routing jouant un rôle de relais entre deux canaux de communication sans participer explicitement dans la transmission des messages SOAP.
Gateway WS-Routing	Un destinataire final WS-Routing assurant la traduction de SOAP vers un autre protocole (par exemple traduction d'un message SOAP en un flux de commande Telnet)

III-5-1-2-Opérations WS-Routing :

1. l'emetteur initial identifie le destinataire final et les intermédiaires WS-Routing (qui définissent le chemin) dans un en-tête WS-Routing.
2. la transmission se fait au premier nœud WS-Routing indiqué, si c'est un intermédiaire il relaie le message au nœud suivant chemin WS-Routing
3. le chemin peut ne pas être connu à l'émission ; à certaines conditions un intermédiaire WS-Routing ajoute dynamiquement les nouveaux intermédiaires au chemin de routage.
4. Enfin, la reconstitution du chemin de retour des messages SOAP peut être établie par ajout d'informations par l'intermédiaire qui assure ainsi une communication point-à-point bidirectionnelle entre émetteurs et récepteurs originels).

III-5-2- Rôle des intermédiaires WS-Routing :

-permettre l'accès aux informations relatives à la transmission des messages SOAP.

-Permettre le franchissement des barrières spécifiques de communication (Pare-feux, protocoles spécifiques de transport).

-un intermédiaire peut appeler dynamiquement un autre intermédiaire par exemple pour la vérification de l'identité de l'émetteur en cas de détour ; par conséquent l'authentification est automatique.

-gateways (passerelles) WS-Routing : pour assurer la traduction entre le protocole véhiculant le message SOAP et les protocoles extérieurs spécifiques.

-Les Balises de WS-Routing : [4]

1-les éléments des noeuds de routage le long d'un chemin :

Balises WS-Routing	Définition
fwd	Séquence d'éléments via ordonnant la liste des intermédiaires WS-Routing à traverser.
rev	Séquence, inverse de la précédente, d'éléments via indiquant le chemin de retour.

2-les éléments du chemin et du chemin de retour :

Balises WS-Routing	Définition
to	URI du destinataire final du message
from.	URI de l'émetteur du message
via	URI de l'intermédiaire (imposé) sur le chemin

exemple1-un chemin minimal sans intermédiaire ni chemin de retour : uniquement l'élément « **to** » est présent, indique une communication directe avec le destinataire dans l'entête WS-Routing.

```
<S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
<S:Header>
<m:path xmlns:m="http://schemas.xmlsoap.org/rp/">
<m:action>http://www.notification.org/update
</m: action>
<m:to>soap://notification.com/some/endpoint</m:to>
<m:id>uuid:20020802-1403-4351-b623-5dsf35sgs5d6
</m:id>
</m:path>
</S:Header>
<S:Body>
...Le corps du message SOAP...
</S:Body>
</S:Envelope>
```

Ainsi encapsulé, le message est prêt à être transporté par TCP, qui est un protocole bi-directionnel, et par UDP, qui ne l'est pas. Dans le premier cas, les éléments '**rev**' sont facultatifs, la couche de transport TCP se chargeant elle-même du calcul du chemin de retour. Dans le second cas, les datagrammes UDP n'étant pas bi-directionnels, les éléments '**rev**' sont obligatoires et renseignés par le traitement aux étapes intermédiaires le long du chemin. En ce qui concerne HTTP, WS-Routing est entièrement compatible avec la liaison SOAP HTTP.

III-6-Le transport SOAP/HTTP:

HTTP est considéré comme protocole privilégié de transport des messages SOAP ; plutôt que SMTP, FTP, POP3 ce qui mène la spécification SOAP à mettre HTTP en avant. Le modèle de spécification requête /réponse de SOAP s'adapte au modèle requête HTTP / réponse HTTP car le message SOAP vient s'imbriquer « naturellement » dans les requêtes et réponses HTTP en bénéficiant de toute la gestion du routage décentralisé du protocole du Web.

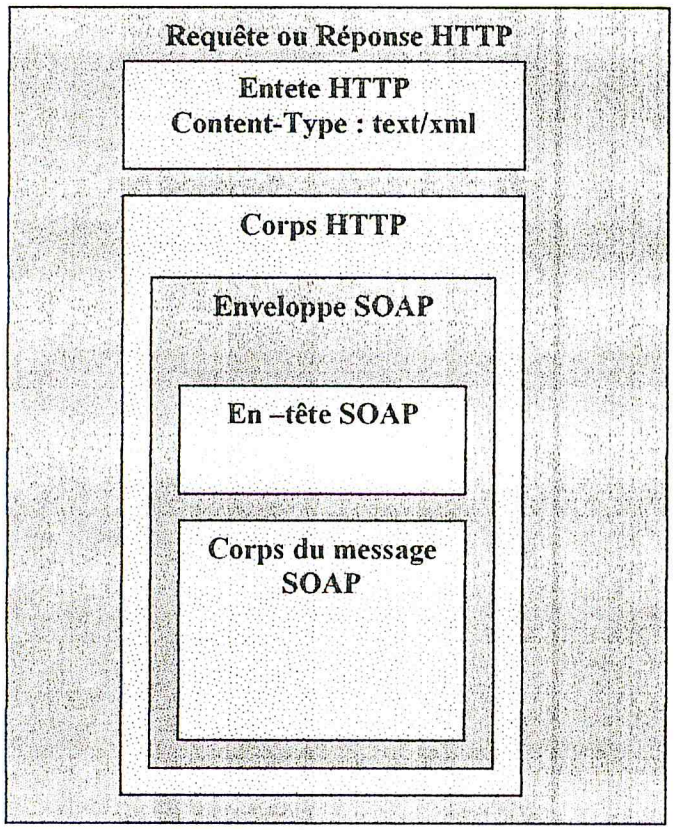


Figure III- 3- Le message SOAP inclus dans une requête ou une réponse http [4]

1-requeteSOAP /HTTP : la spécification utilise les requêtes HTTP type **POST.** Pour transporter les requêtes soap ; il faut ajouter un attribut supplémentaire **SOAPAction** dans l'entête HTTP qui prend valeur comme un URI sans format spécifique ; pour que la requête puisse se manifester au prêt du destinataire qui prend en charge l'interprétation.

Exemple de requête :

```
POST http://www.serveurBoursier.fr HTTP/1.1
Content-type: text/xml ; charset="utf-8"
Content-length: mcdu
SOAPAction: http://uneUriCompriseDuDestinataire#infos
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope"
SOAP-ENV :encodingStyle="http://schemas.xmlsoap.org/soap/encoding">
<SOAP-ENV:Body>
<!-- Requête de la valeur d'un titre -->
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

2-la réponse SAOP/ HTTP: La réponse suit les codes de retour HTTP si la réponse porte le **code 2xx** cela indique que le message SOAP a été reçu (dans l'exemple 200), compris et traité correctement.

Cas contraire : SOAP utilise par convention le **code 500**(Internal Server Error) ; la balise « **body** » doit contenir l'élément « **Fault** » et indique la nature d'erreur.

Exemple de réponse :

```
HTTP/1.1 200 OK
Content-type: text/xml ; charset="utf-8"
Content-length: mcdu
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
envelope"
SOAP-ENV :encodingStyle="http://schemas.xmlsoap.org/
soap/encoding">
<SOAP-ENV:Body>
<!-- Valeur du titre -->
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

III-7-Conclusion:

Le protocole SOAP est nécessaire pour une communication entre entités informatiques via Web Services. La spécification SOAP définit un modèle d'échange de messages qui repose sur trois concepts :

- Les Messages.
- Les Emetteur.
- Les Récepteurs.

Les messages sont des documents XML, ils voyagent d'un émetteur vers un récepteur.Les récepteurs peuvent former une chaîne. Les messages SOAP peuvent donc suivre le modèle

client-serveur classique acheminés via HTTP ou être utilisés suivant le modèle d'échange de message à sens unique. Dans les deux cas on trouvera un émetteur (E), et un récepteur appelé point final (PF) reconnus par des spécifications WS-Routing En chaînant ces messages, on obtient une suite d'émetteurs et de récepteurs SOAP appelés intermédiaires SOAP. Ces intermédiaires peuvent avoir accès aux informations de transport donc spécifier le chemin de transmission.

Puisque la spécification WS-Routing est compatible avec les liaisons SOAP/http, et ce dernier est destiné pour le transport des documents hypertexte, alors cela le favorise pour transporter des documents XML sur le Web et franchir plusieurs pare feux (Firewalls) résidants dans plusieurs nœuds de la toile qui représentent dans certains cas les intermédiaires dans le chemin de transmission.

De plus le message SOAP intègre des données multimédias (où le format des données change) à des messages MIME en s'encapsulant dans ces derniers.

Toutes ces possibilités sont dues à la simplicité de la structure SOAP qui repose sur XML pour représenter les structures de données même jugées complexes grâce à XML schéma.

Une fois le message reçu par le (PF) il faut l'interpréter donc il faut prévoir des interfaces de réceptions et dans certains cas de traductions du langage XML véhiculé par le protocole de transport. On a prévu des spécifications en WSDL considérées comme descriptions des Web Services recherchés et ayant besoin d'être décrits; et précisant les points d'accès pour tout message reçu. Ce sera la phase d'étude qui suit celle-ci.

Chapitre IV : Etude de la norme WSDL

Pourquoi utiliser le langage WSDL ?le protocole SOAP est une norme standard ;Le langage WSDL (*Web Services Description Language*) permet de tirer les bénéfices de SOAP en offrant la possibilité aux fournisseurs et aux utilisateurs de services Web de fonctionner ensemble facilement La première chose à définir est un format standard pour spécifier les interfaces,puisque SOAP n'a pas besoin d'un langage de description d'interface ;c'est une norme pour communiquer du contenu pur, donc utiliser un langage pour décrire ce contenu. Les messages SOAP transportent des informations de type, ce qui fait que SOAP autorise la détermination dynamique du type. Mais on ne peut pas appeler correctement une fonction tant qu'on n'en connaît pas le nom ni le nombre de paramètres et le type de chacun d'entre eux. Sans le langage WSDL, on peut deviner la syntaxe d'appel à partir de la documentation qui doit être fournie, ou en examinant les messages en ligne. D'une manière ou d'une autre, une intervention humaine sera nécessaire, ce qui ouvre la porte à des erreurs. Grâce au langage WSDL, on peut automatiser la génération de Proxy pour les Web services de façon totalement indépendante de la plate-forme et du langage. De même que pour le fichier IDL (Interface Definition Language) pour COM et CORBA, un fichier WSDL est un contrat entre le client et le serveur. [5].WSDL (WebServices Description Langage) c'est un langage pour décrire d'une façon précise les Web Services ; cela en donnant plus de détails sur les protocoles, les serveurs, les ports utilisés, les opérations pouvant être effectuées, les formats des messages d'entrée et de sortie, et les exceptions pouvant être renvoyées.Microsoft a d'abord proposé SDL (Service Definition Language) avec une implémentation fournie dans leur Toolkit SOAP.Ensuite : IBM a proposé NASSL (Network Accessible Service Specification Language), dont une implémentation est fournie dans SOAP4J.Puis : Microsoft modifia sa première spécification et proposa SCL (SOAP Contract Language).Enfin : les différents acteurs s'accordèrent sur WSDL qui a comme Rôle d'Interface masquant l'implémentation du service.

IV-1-Structure d'un document WSDL :

La structure du document WSDL est constituée des balises indispensables suivantes :

<definitions> : Cet élément contient la définition du service. C'est la racine de tout document WSDL. Cette balise peut contenir les attributs précisant le nom du service, et les espaces de nommage. <definitions> contient trois types d'éléments :

-**<message>** et **<portType>** : Ces éléments définissent les opérations offertes par le service, leurs paramètres d'entrée et de sortie, etc. En particulier, un <message> correspond à un paramètre d'entrée ou de sortie d'une <operation>.

Un <portType> définit un ensemble d'opérations. Une **<operation>** définit un couple message entrée / message sortie. Par exemple, dans le monde Java, une opération est une méthode et un portType une interface.

-**<binding>** : Cet élément associe les <portType> à un protocole particulier. Les bindings possibles sont SOAP, CORBA ou DCOM. Actuellement seul SOAP est utilisé. Il est possible de définir un binding pour chaque protocole supporté.

-**<service>** : Cet élément précise les informations complémentaires nécessaires pour invoquer le service, et en particulier l'URI du destinataire. Un <service> est modélisé comme une collection de ports, un **<port>** étant l'association d'un <binding> à un URI.

Il est aussi possible de définir des types de données complexes dans une balise <types> juste avant la balise <message>. Enfin, chaque élément WSDL peut être documenté à l'aide de l'élément <documentation>. Cet élément contient des informations liées à la compréhension du document par les utilisateurs humains du service.

```
<Definitions >
        <Message>
        ...
        </Message>
        <portType>
            <operation>...</operation>
            <operation>...</operation>
            ...
        </portType>
        <binding>
        ...
        </binding>
        <service>
            <port>...</port>
            <port>...</port>
        </service>
</Definitions >
```

Figure IV- 1 Structure d'un document WSDL [6]

Un document WSDL est un document XML décrivant les relations entre les différentes sections constituant le document WSDL.Le document WSDL peut être divisé en deux groupes de sections :

-les définitions abstraites : définissent les messages SOAP de façon totalement indépendante de la plate-forme et de la langue ; elles ne contiennent aucun élément spécifique à la langue ou à la machine. Cela facilite la définition d'un ensemble de services pouvant être implémenté par différents sites Web. Elles sont définies dans les balises suivantes :

Types : Définitions de type indépendantes de la langue et de la machine.

Messages : Contient les paramètres de fonction (les entrées séparées des sorties) ou les descriptions de document.

PortTypes : Réfère aux définitions de message de la section Messages pour décrire les signatures de fonction (nom d'opération, paramètres d'entrée, paramètres de saisie). Chaque section abstraite peut être définit dans un document distinct et importée dans le document principal.

La figure ci-dessous présente les liaisons entre les différentes sections WSDL :

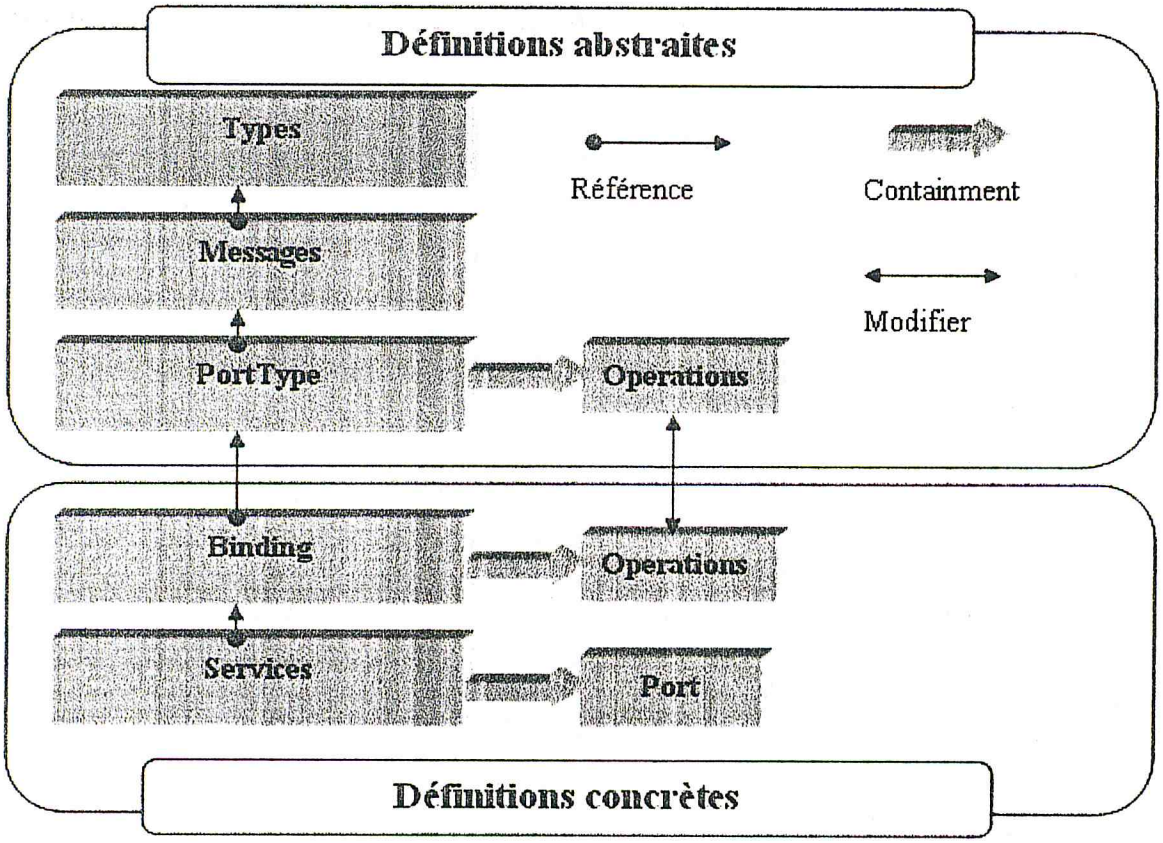


Figure IV- 2 -Définitions concrètes et abstraites [5]

-les descriptions concrètes : illustrent les questions spécifiques au site, telles que la sérialisation. Elles sont définies dans les balises suivantes :

Bindings : Indique la (les) liaison(s) de chaque opération dans la section PortTypes.

Services : Indique les adresses de port de chaque liaison.

IV-2-Exemple de fichier WSDL :

Le fichier WSDL suivant correspond à un Web Service très simple qui a comme nom «calculer» et qui réalise l'addition de deux nombres (a+b) ; ce document décrit le service selon la structure définit dans la *Figure IV-1.*

```
1)   <?xml version="1.0" encoding="utf-8" ?>
     - <definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
2)     xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
       xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:s0="http://tempuri.org/"
2.2)   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
       xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
       xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
2.1)   targetNamespace="http://tempuri.org/" xmlns="http://schemas.xmlsoap.org/wsdl/">
3)     - <types>
3.1)     - <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
3.2)       - <s:element name="Add">
             - <s:complexType>
               - <s:sequence>
                   <s:element minOccurs="1" maxOccurs="1" name="a" type="s:int" />
                   <s:element minOccurs="1" maxOccurs="1" name="b" type="s:int" />
                 </s:sequence>
               </s:complexType>
             </s:element>
3.3)       - <s:element name="AddResponse">
             - <s:complexType>
               - <s:sequence>
                   <s:element minOccurs="1" maxOccurs="1" name="AddResult"
                     type="s:int" />
                 </s:sequence>
               </s:complexType>
             </s:element>
           </s:schema>
         </types>
4.1)   - <message name="AddSoapIn">
           <part name="parameters" element="s0:Add" />
         </message>
4.2)   - <message name="AddSoapOut">
           <part name="parameters" element="s0:AddResponse" />
         </message>
5)     - <portType name="calculerSoap">
5.1)     - <operation name="Add">
5.2)         <input message="s0:AddSoapIn" />
5.3)         <output message="s0:AddSoapOut" />
           </operation>
         </portType>
6)     - <binding name="calculerSoap" type="s0:calculerSoap">
6.1)       <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
             style="document" />
6.2)     - <operation name="Add">
6.3)         <soap:operation soapAction="http://tempuri.org/Add" style="document" />
```

```
6.4)  - <input>
          <soap:body use="literal" />
      </input>
6.5)  - <output>
          <soap:body use="literal" />
      </output>
    </operation>
  </binding>
7)- <service name="calculer">
7.1)- <port name="calculerSoap" binding="s0:calculerSoap">
7.2)    <soap:address location="http://localhost:8080/calcul.asmx" />
    </port>
  </service>
</definitions>
```

Illustration des fonctionnalités les plus évidentes du document ci-dessus:

1)-Déclaration que le document est du langage XML pour aider l'analyseur XML à déterminer si le fichier WSDL doit être analysé ou s'il doit signaler une erreur.

2)- Elément **<définitions>** racine du document WSDL. Il existe plusieurs attributs d'espace de noms (déclarations d'espace de noms) attachés à l'élément racine, de même que dans l'élément enfant <schema> de l'élément <types>.il comprend deux types d'espaces de noms :

2.1) l'attribut **targetNamespace** qui déclare **un espae de nom principal** http://tempuri.org auquel tous les noms déclarés dans un élément du fichier WSDL lui appartiennent.

2.2) déclaration d'attributs d'espace de noms où par la suite est déclaré un alias pour chaque espace de nom utilisé dans le document par exemple xmlns : s= http://www.w3.org/2001/XMLSchema où :

-xmlns : l'attribut de l'espace de nom :"http://www.w3.org/2001/XMLShema"

-s : l'alias qui permet de faire référence à l'espace de nom dans tout le document WSDL cela en préfixant un nom c'est-à-dire qualifiant «**s :**» à un nom par exemple « s:int » :c'est un type qualifié ; le préfixe d'alias suit les règles d'étendues classiques ce qui le rend valable que dans l'élément où il est défini.-Les autres alias utilisés sont :

s0 : pour l'espace de nom principal

soap :pour l'espace de nom qui désigne les schémas des structures des messages SOAP.

soapenc :pour l'espace de nom des schémas des règles de codage du protocole SOAP(cf. III-3).

tm, mime pour les espaces de nom des schémas qui désignent les messages MIME (cf.III-4).

En général un document WSDL peut contenir les préfixes ou alias de noms suivants :

Préfixe	Espace de noms équivalent	Description
soapenc	http://schemas.xmlsoap.org/soap/encoding	Codage SOAP 1.1
wsdl	http://schemas.xmlsoap.org/wsdl/soap	WSDL 1.1
xsd	http://www.w3.org/2001/XMLSchema	Schéma XML

Pour saisir les donnés WSDL il faut se baser sur **"XML Schema: Datatypes"** (XSD) une recommandation de W3C , utilisée dans les sections **Types** et **Messages** du langage WSDL ,connue sous trois versions différentes (1999, 2000/10 et 2001), et le fait de le déclarer comme l'un des attributs d'espace de noms dans l'élément <definitions> permet de préciser quelleversion est utilisée dans un fichier WSDL donné :Xmlns :xsd= http://www.w3.org/2001/XMLSchema,dans notre exemple nous avons l'alias « **s** »(cf. II-1-4). La documentation de MSTK2 (MicroSoft ToolKit 2), énumère tous les types de base XSD pris en charge par MSTK2. Le tableau montre comment le lecteur WSDL, côté client comme côté serveur, fait correspondre les types XSD aux types correspondants en VB, C++ et IDL.XSD définit deux ensembles de types de données intégrés (string,int,float…) : les primitives et les dérivées.

3)- Elément **<Types>** comprend la section Types elle est utilisée selon le cas de déclaration de nouveaux types (il se peut qu'aucun type de données ne soit déclaré). La section Types dans cet exemple est réservée pour déclarer les espaces de noms de schéma utilisés dans le document. Il comprend l'élément enfant :

3.1) **<schema>** à attribut **targetNamespace** qui lui est propre, à valeur **:** http://tempuri.org pour préciser que tous les noms définis dans cet élément <schema> appartiennent à cet espace de noms principal.

Dans l'élément **3)** sont déclarés aussi deux éléments enfants pour définir les types complexes, XML Schema permet de définir les types complexes connu sous le nom de structure en «langage C». Dans un fichier WSDL, les types complexes sont déclarés dans la section **Types** avec l'élément **<complexType>**.d'où la représentation suivante:

3.2) l'élément **Add** qui comporte deux éléments enfants **«a»** et **«b»** de type entier «**Int**», c'est une structure qui comporte deux champs.

3.3) l'élément **AddResponse** qui comporte un seul élément fils «**AddResult**» de type entier «**Int**», c'est une structure qui comporte un seul champ.

Ces deux éléments sont nécessaires pour la déclaration de types complexes spécifiques au service afin d'être ensuite utilisé dans la section messages, lors de l'initialisation de MSTK2 SoapClient, celui-ci analysera le fichier correctement. Toutefois, il ne pourra pas envoyer un appel de fonction à **«AddSoapIn»**. En effet, SoapClient ne sait pas comment traiter les types complexes de lui-même et il a besoin d'un mappeur de type personnalisé pour le faire. La documentation de MSTK2 inclue un mappeur de type personnalisé.

-la Section< **message**> définit les paramètres pour les fonctions qui sont considérées comme section **<opération>**.Dans le cas où le langage WSDL est utilisé pour l'échange des documents, il autorise l'utilisation d'éléments <message> pour décrire le document à échanger. Elle comporte deux messages :

4.1) Premier message **«AddSoapIn»**; d'entrée reconnu au suffixe **«In».**

4.2)Deuxième message **«AddSoapOut»**; de sortie reconnu au suffixe **«Out».**

L'Elément **<part>** enfant de l'élément <message>,il correspond à un paramètre d'entrée ou de sortie défini dans un élément <message> unique; s'ils existent les deux à la fois alors les paramètres d'entrée sont séparés des paramètres de sortie, qui se trouvent dans leur propre élément <message> d'où l'attribution de noms «name="**parameters**"»; on distingue le paramètre de sortie par le mot final «out» dans cet exemple : «**AddSoapOut**».Si un paramètre désigne à la fois un paramètre d'entrée et de sortie alors il possède son élément <part> correspondant dans l'élément <message> d'entrée et de sortie. L'élément <part> possède un type comme ceux d'une fonction, les types autorisés sont : un type de base XSD, un type défini par SOAP (soapenc), un type défini par WSDL (wsdl) ou un type défini par la section Types,dans cet exemple il correspond à **Add** et **AddResponse** les types complexes définis dans la section types .l'utilité de cet élément est de pouvoir lier les messages à une déclaration de type 3): par *un attribut d'élément*; 3.2); 3.3),donc utiliser les types complexes déclarés au-dessus avec les déclarations d'éléments suivantes: element="**s0:Add**" et element="**s0:AddResponse**" les noms déclarés dans ces éléments sont qualifiés par l'alias "**s0**" de la section 2.2).il existe une autre méthode par un attribut de type. (cf..Annexe F).

Lors de l'échange le message d'entrée attend en paramètres **«a»** et **«b »**; le message de sortie donne comme résultat le paramètre **«AddResult»**, pour avoir ce résultat il faut accéder au calcul correspondant; en général toute information qui génère un résultat est contenue dans l'élément d'opération. Toutes les opérations sont groupées dans l'élément portType.

5)-l'Elément **<portType>** qui a le nom "**calculerSoap**",il fait partie de la section abstraite, un même document WSDL peut contenir zéro, un ou plusieurs éléments <portType> ,où chacun d'eux comporte un ou plusieurs éléments <operation>; chaque élément<operation> décrit la syntaxe d'appel de toutes les méthodes de portType; déclarer le nom de la méthode, les paramètres (à l'aide d'éléments <message>) et les types de chacun d'entre eux (à l'aide d'éléments <part> déclarés dans chaque <message>). Cet élément comporte au maximum -un élément <input> -un élément <output> et en cas d'erreur l'élément<fault> (erreur) est introduit ; cet exemple, comporte :

5.1) l'Elément **<operation>** pour la définition des fonctions qui correspondent au même nombre d'élément opération; ici l'élément <operation> est unique, nommé " **Add** "équivaut à un nom de méthode (ou fonction). Ses éléments enfants sont les élément: 5.2) <input> (entrée) ,5.3) <output> (sortie). L'attribut de message de chaque élément <input> et <output> réfère à l'élément <message> correspondant dans la section Messages. La section <portType> de l'exemple est définie en «langage C» par la déclaration de fonction suivante «int Add (int parameters)».les éléments: <operation name="**Add**">;<input message="**s0:AddSoapIn**" />;<output message="**s0:AddSoapOut**" />, ont la signification suivante: pour avoir accès à l'opération **Add** il faut envoyer un message **AddSoapIn** qui transporte les paramètres **« a »** et **«b»**, la fonction **Add** fait le calcul et affecte le résultat au paramètre de sortie **AddResult** qui sera transporté par le message de sortie **AddSoapOut** pour le client. Deux problèmes se pose à présent: quel est le protocole de transport? Et où trouver cette fonction qui effectue le calcul? Les deux sections qui suivent répondent à cette question par une technique d'association.

6)-Eléments < **binding** > qui a le nom "**calculerSoap**", il définit la section binding qui peut avoir zéro, un ou plusieurs éléments <binding>et qui a pour rôle de spécifier comment les appels et réponses d'élément <operation> sont envoyés et transmis. Cette section permet d'associer le **portType** au protocole SOAP via la déclaration 6.1) transport=**http://schemas.xmlsoap.org/soap/http** qui précise aussi que le transport se fait grâce au protocole HTTP; on spécifie quel est l'opération à extraire de cette association par la définition 6.2) <operation name="**Add**"> qui à son tour contient la déclaration de l'attribut supplémentaire 6.3)soapAction=**http://tempuri.org/Add** à inclure dans l'entête HTTP(cf.III-6) ;cet attribut, désigne une URI spécifique à SOAP utilisée dans la formulation du message SOAP.,il est destiné pour la liaison http non pas une autre.SOAP 1.1 établit que soapAction est utilisé pour identifier "l'intention" du message, le serveur peut utiliser cet

attribut pour acheminer le message sans avoir besoin de l'analyser dans sa totalité .en pratique, son utilisation varie.

Dans ces deux déclarations 6.1); 6.2)l'attribut style="**document**"est définie pour préciser que l'information à transporter est sous format "**document**", non pas un autre format (JPEG...etc;cf. III-4),mais la précision dans la deuxième déclaration est utilisée pour ignorer le style spécifié dans l'élément <soap:binding>s'ils sont différents.

Les déclarations 6.4) <input>; 6.5) <output> sont utilisées pour spécifier le corps «body» des messages SOAP d'entrée et de sortie; avec l'élément <soap:body use="**literal**" /> qui spécifie les données de l'élément <body> du message SOAP résultant.

L'attribut use permet de spécifier si les données sont codées ou littérales (cas de cet exemple):

-" **literal** " (littérales) signifie que le message SOAP résultant contient les données formatées exactement comme dans les définitions abstraites (sections Types, Messages et PortTypes).

-"**Encoded**" (codées) qui doit être définie avec:

1-l'attribut «**encodingStyle**» (qu'il faut introduire) décide du codage SAOP et doit avoir la valeur d'URI "http://schemas.xmlsoap.org/soap/encoding".

2-un «**Namespace**»: Chaque corps d'un message SOAP peut avoir son propre espace de noms pour éviter les problèmes de noms. L'URI spécifiée dans cet attribut est utilisée telle quelle dans le message SOAP résultant.

Le nom de l'élément Binding «**calculerSoap**» est choisi de sorte que l'élément <port> 7.1) de la section Services 7) puisse y référer. Et l'attribut de type «**s0:calculerSoap**»associé à cet élément réfère au <portType> 5).Donc la section Binding est la section des liens (vers les protocoles, les URLs, les PortTypes des messages et des opérations, les ports et les services).

7)- Eléments <**service**> définit la section Services qui possède zéro, un ou plusieurs éléments <service> distingué les uns des autres par l'attribut de nom du service.dans cet exemple un seul service est proposé avec un attribut de nom: «name="**calculer**"», il contient des éléments enfants:

7.1) l'élément <**port**>; il peut y avoir plusieurs possédant chacun un attribut de nom, chaque port réfère à un élément <binding> de la section Bindings. Ce service possède un seul port «name="**calculerSoap**"» qui fait référence à au Binding par la déclaration «binding="**s0:calculerSoap**"».

7.2)- <soap:address location="**http://localhost:8080/MathService.asmx**" /> cette déclaration indique l'URL du service "calculer" que le message SOAP doit accéder par le port"**calculerSoap**",en utilisant le port **8080.**

Un service se compose d'éléments <port>. Chaque élément <port> associe un emplacement à une liaison (<binding>) par paire. S'il y a plusieurs éléments <port> associés au même élément <binding>, les emplacements d'URL supplémentaires peuvent être utilisés comme des alternatives. On peut avoir plusieurs éléments <service> dans un document WSDL; pour cela il existe plusieurs méthodes parmi elles:

-La méthode de regrouper les ports en fonction de la destination de l'URL. Ainsi, on peut rediriger toutes mes demandes d'inventaire en utilisant simplement un autre <service> dans un document WSDL; le programme client continuerait à fonctionner car, dans ce type de service, le regroupement du protocole est indépendant des différents services.

-La méthode de classer les ports en fonction du protocole sous-jacent. Par exemple, on peut placer tous les ports HTTP dans un <service>, et tous les ports SMTP dans un autre. Le client peut alors rechercher le <service> qui correspond au protocole qu'il peut traiter. Le document WSDL est écrit en XML et utilise le protocole SOAP pour établir une communication concrète entre client et serveur. [5]

IV-3-Les messages SOAP :

Un fichier WSDL est un moyen, pour les clients et serveurs qui l'utilisent, de savoir ce qui est envoyé sur le réseau c'est un «contrat d'interface» et plus que ça c'est «un langage de spécification de protocole» qui peut aller au-delà des protocoles "figés", tels qu'IP et http que SOAP utilise, en considérant des protocoles spécifiques aux applications, qui décide du protocole haut niveau à utiliser entre un client et un serveur donné et dans ce cas c'est le langage WSDL qui sera la meilleure manière de spécification entre plusieurs autres manières. L'exemple où un protocole à haut niveau peut être utilisé est dans le service qui fournit une opération « echoInt » (une opération définie dans la documentation de MSDN Library) qui reproduit un entier en entrée, le nombre de paramètres, le type de chaque paramètre et la manière dont ces paramètres doivent être envoyés par le réseau (sérialisation) constituent un protocole spécifique à l'application qui peut être spécifié de nombreuses manières ; dont la spécification WSDL.

La Relation SOAP/WSDL : Le langage WSDL peut indiquer si les messages SOAP sont conformes aux styles **RPC** (appel de procédure distante) ou bien à ceux de **documents**. Un message de style RPC, ressemble à un appel de fonction avec zéro paramètres ou plus. Un message de style documents est plus plat et requiert moins de niveaux d'imbrication. Les messages SOAP correspondants à l'exemple précédant sont les suivants :

IV-3-1- Requête envoyée à partir d'un client pour réaliser un appel de fonction :

```
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/Add"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  <soap:Body>
    <Add xmlns="http://tempuri.org/">
      <a>int</a>
      <b>int</b>
    </Add>
  </soap:Body>
</soap:Envelope>
```

1 2 3

Figure IV- 3 Requête SOAP

1)-la déclaration de l'entête http de la requête à envoyer au serveur

2)- le document XML qui contient l'enveloppe du message SOAP

3)-le corps du message SOAP, qui indique les paramètres «a» et «b» et leur type, de la fonction 'Add', à exécuter par le serveur.

IV-3-2- Réponse reçue en retour du serveur par le client :

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <AddResponse xmlns="http://tempuri.org/">
      <AddResult>int</AddResult>
    </AddResponse>
  </soap:Body>
</soap:Envelope>
```

4 5

Figure IV- 4 Réponse SOAP

4)-la déclaration de l'entête http de la réponse du serveur.

5)- le corps du message SOAP, qui comporte la réponse, comme résultat de la fonction Add. La structure des données des messages SOAP ainsi que leurs types sont conformes à la description contenue dans le document WSDL.

IV-4-Un Web Service qui comporte plusieurs méthodes :

Le document WSDl de l'exemple précédent correspond à un Web Service offrant une seule méthode ; l'exemple qui suit défini un service mathématique appelé « **MathService** » qui offre quatre méthodes différentes (l'addition, la soustraction, la multiplication, la division de deux variables **a** et **b**).

```
<?xml version="1.0" encoding="utf-8" ?>
- <definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:s0="http://tempuri.org/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
  targetNamespace="http://tempuri.org/" xmlns="http://schemas.xmlsoap.org/wsdl/">
  - <types>
    - <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
      - <s:element name="Add">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="1" maxOccurs="1" name="a" type="s:float" />
              <s:element minOccurs="1" maxOccurs="1" name="b" type="s:float" />
            </s:sequence>
          </s:complexType>
        </s:element>
      - <s:element name="AddResponse">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="1" maxOccurs="1" name="AddResult"
                type="s:float" />
            </s:sequence>
          </s:complexType>
        </s:element>
      - <s:element name="Soustraction">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="1" maxOccurs="1" name="a" type="s:float" />
              <s:element minOccurs="1" maxOccurs="1" name="b" type="s:float" />
            </s:sequence>
          </s:complexType>
        </s:element>
      - <s:element name="SoustractionResponse">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="1" maxOccurs="1" name="SoustractionResult"
                type="s:float" />
            </s:sequence>
          </s:complexType>
        </s:element>
      - <s:element name="Multiplication">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="1" maxOccurs="1" name="a" type="s:float" />
              <s:element minOccurs="1" maxOccurs="1" name="b" type="s:float" />
            </s:sequence>
          </s:complexType>
        </s:element>
```

```
  - <portType name="MathServiceSoap">
    - <operation name="Add">
        <input message="s0:AddSoapIn" />
        <output message="s0:AddSoapOut" />
      </operation>
    - <operation name="Soustraction">
        <input message="s0:SoustractionSoapIn" />
        <output message="s0:SoustractionSoapOut" />
      </operation>
    - <operation name="Multiplication">
        <input message="s0:MultiplicationSoapIn" />
        <output message="s0:MultiplicationSoapOut" />
      </operation>
    - <operation name="Division">
        <input message="s0:DivisionSoapIn" />
        <output message="s0:DivisionSoapOut" />
      </operation>
    </portType>
  - <binding name="MathServiceSoap" type="s0:MathServiceSoap">
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
        style="document" />
    - <operation name="Add">
        <soap:operation soapAction="http://tempuri.org/Add" style="document" />
      - <input>
          <soap:body use="literal" />
        </input>
      - <output>
          <soap:body use="literal" />
        </output>
      </operation>
    - <operation name="Soustraction">
        <soap:operation soapAction="http://tempuri.org/Soustraction"
        style="document" />
      - <input>
          <soap:body use="literal" />
        </input>
      - <output>
          <soap:body use="literal" />
        </output>
      </operation>
    - <operation name="Multiplication">
        <soap:operation soapAction="http://tempuri.org/Multiplication"
        style="document" />
      - <input>
          <soap:body use="literal" />
        </input>
      - <output>
          <soap:body use="literal" />
        </output>
      </operation>
```

```
- <s:element name="MultiplicationResponse">
  - <s:complexType>
    - <s:sequence>
        <s:element minOccurs="1" maxOccurs="1" name="MultiplicationResult"
          type="s:float" />
      </s:sequence>
    </s:complexType>
  </s:element>
- <s:element name="Division">
  - <s:complexType>
    - <s:sequence>
        <s:element minOccurs="1" maxOccurs="1" name="a" type="s:float" />
        <s:element minOccurs="1" maxOccurs="1" name="b" type="s:float" />
      </s:sequence>
    </s:complexType>
  </s:element>
- <s:element name="DivisionResponse">
  - <s:complexType>
    - <s:sequence>
        <s:element minOccurs="1" maxOccurs="1" name="DivisionResult"
          type="s:float" />
      </s:sequence>
    </s:complexType>
  </s:element>
</s:schema>
</types>
<message name="AddSoapIn">
  <part name="parameters" element="s0:Add" />
</message>
- <message name="AddSoapOut">
  <part name="parameters" element="s0:AddResponse" />
</message>
- <message name="SoustractionSoapIn">
  <part name="parameters" element="s0:Soustraction" />
</message>
- <message name="SoustractionSoapOut">
  <part name="parameters" element="s0:SoustractionResponse" />
</message>
- <message name="MultiplicationSoapIn">
  <part name="parameters" element="s0:Multiplication" />
</message>
- <message name="MultiplicationSoapOut">
  <part name="parameters" element="s0:MultiplicationResponse" />
</message>
- <message name="DivisionSoapIn">
  <part name="parameters" element="s0:Division" />
</message>
- <message name="DivisionSoapOut">
  <part name="parameters" element="s0:DivisionResponse" />
</message>
```

```
- <operation name="Division">
    <soap:operation soapAction="http://tempuri.org/Division" style="document" />
  - <input>
      <soap:body use="literal" />
    </input>
  - <output>
      <soap:body use="literal" />
    </output>
  </operation>
</binding>
- <service name="MathService">
  - <port name="MathServiceSoap" binding="s0:MathServiceSoap">
      <soap:address location="http://localhost:8080/MathService.asmx" />
    </port>
  </service>
</definitions>
```

Figure IV- 5 WSDL d'un Web Service avec plusieurs méthodes.

La structure du document ci-dessus (IV-4) est la même que celui de l'exemple (IV-2) sauf que le nombre d'opérations diffère ; ce qui a provoqué une évolution dans les déclarations suivantes :

1) le nombre des types complexes.

2) Les messages d'entrée sortie pour chaque opération.

4) le nombres d'opérations : **"Add"** pour l'addition, **"Soustraction"** pour la soustraction, **"Multiplication"** pour la multiplication, **"Division"** pour la division.

Mais ce qui est important à noter est que les éléments 3), 5) et 6) sont uniques :

3) puisque l'élément PortType regroupe toutes les opérations possible d'un service alors il sera unique.

5) l'élément Binding aussi est unique car il spécifie un seul protocole de transport ; tandis que ce qui change c'est le nombre des liaisons de chaque opération par les éléments "soapAction".

6) l'élément Service est unique puisque c'est le même service qui fournit toutes ces opérations.

IV-5-Conclusion :

Un document WSDL se compose d'un ensemble d'éléments décrivant les types de données utilisés par le serveur, les «messages» que le service peut recevoir ainsi que les liaisons SOAP associées à chaque message. Toutes ces informations sont groupées dans un même document écrit en XML. Rechercher un Web Service c'est rechercher le document WSDL associé, car ce dernier possède l'adresse du Web Service recherché.

La spécification WSDL décompose le document en deux sections :

-La section abstraite: elle est réservée aux différentes définitions suivantes :

- les types complexes nécessaires pour la définition de l'interface du service.
- Les messages où chacun peut être tout paramètre transmis à une méthode sur objet présentée par le service, ou tout résultat renvoyé par l'appel de cette méthode.
- Toutes les opérations sont groupées dans un PortType.

Elle comporte les éléments XML suivants : **<Types>, <Messages>, <Operation>, <PortTypes>.**

-La section concrète : elle matérialise les définitions abstraites, où les détails physiques de la transmission des données sont traités car elle permet de :

- Spécifier le protocole.
- La sérialisation et le codage sur le réseau.
- Séparer les spécifications de liaison des déclarations de données et de messages.
- Normaliser un ensemble d'opérations (portType) pour le même type d'activité.
- En même temps se distinguer en proposant des liaisons personnalisées des messages avec les extrémités réceptrices du serveur. Celles-ci permettent d'accéder au service et de connaître le format des données que le service exige pour l'échange avec des messages SOAP comportant impérativement le même format et le même nombre de paramètres contenus dans le document.
- Définir des ports en une unique entité service.

Elle comporte les éléments XML suivants : **<Binding>, <Service>** et **<Port>.**

Les définitions abstraites sont établies comme une norme et sont placées dans leur propre fichier séparé des sections concrètes. C'est *la construction d'importation* du langage Par exemple, les banques peuvent se mettre d'accord pour normaliser un ensemble d'opérations

bancaires, qui sont ensuite décrites en détail dans un document WSDL de définitions abstraites. Chaque banque est libre ensuite de "personnaliser" le protocole sous-jacent, les optimisations de sérialisation et le codage, c'est la standardisation.

Chapitre V : Etude des plateformes J2EE/.NET

Les caractères révolutionnaires (universalité et ubiquité) d'Internet et du Web; ont permis une évolution considérable des applications; depuis le mainframe passant par le client-serveur avec l'architecture à trois niveaux aux applications réparties à objets avec (Corba, DCOM et EJB) qui ont connu une très forte complexité dans leur mise en œuvre. Les applications fondées sur les Web Services sont l'étape actuelle de cette évolution ; L'architecture des Web Services repose sur un mécanisme de transport d'une demande de service entre un client et un serveur, tous deux connectés au réseau le client peut être soit un navigateur Web soit une application. Le serveur, peut être une application s'exécutant sur un moteur de scripts interfacé à un serveur HTTP (PHP/ Apache, par exemple) ou sur un serveur d'applications J2EE ou .NET.

Les deux technologies J2EE/NET reposent sur une architecture de base qui est celle du client-serveur; qui repose à son tour sur plusieurs niveaux, qualifiée aussi d'architecture n-tiers.

V-1-L'architecture type

Le client-serveur a permis de découper d'énormes applications monolithiques des sites centraux ; pour répartir les traitements entre les clients et les serveurs Il existe une architecture client /serveur :

-A deux niveaux (architecture *orientée client*) le plus gros de l'application (logique applicative : requêtes SQL envoyées à une base de donnée) est exécuté coté client ; c'est une architecture qui a fait ses preuves. Mais l'évolution des applications a fait que ce modèle n'est pas adapté aux missions critiques, ce qui a donné naissance à une architecture à trois niveaux.

-A trois niveaux (architecture *orientée serveur* ou *client léger*) qui déplace la logique applicative vers le serveur.

- Niveau1 : les clients exécutent la logique de l'interface graphique utilisateur GUI (Graphical User Interface).
- Niveau2 : le serveur d'applications exécute la logique applicative.
- Niveau3 : pour la base de donnée ou l'application.

Vu que les applications sont découpées en composants et distribuées sur des processeurs multiples, l'architecture peut s'étendre à N niveaux, elle est qualifiée d'architecture N-tiers.

C'est une architecture à base d'objets, elle comprend plusieurs couches ; Cette séparation par couches de responsabilités sert à découpler au maximum une couche de l'autre afin d'éviter l'impact d'évolutions futures de l'application.

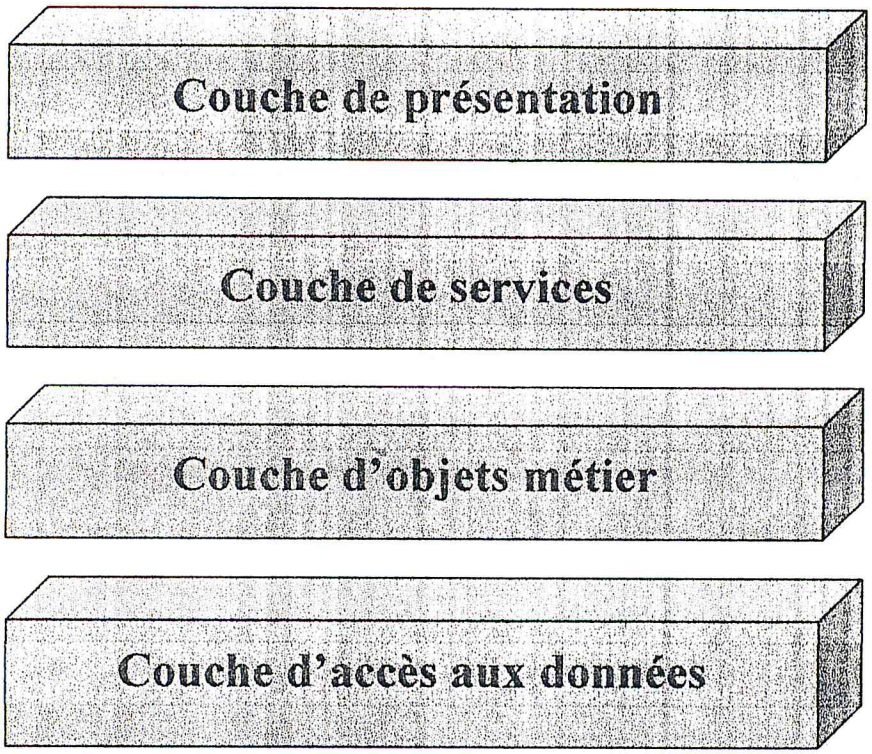


Figure V- 1-les couches de l'architecture n-tiers. [7]

Le contenu de chaque couche :

1-La couche de présentation contient les différents types de clients, léger (Web, ASP, JSP) ou lourd (Swing, WinForm)

2-La couche de service contient les traitements (contrôleurs de Use Case UML) représentant les règles métier (créer un compte, rechercher un client, calculer un amortissement, ...)

3-La couche d'objets métier est représentée par les objets du domaine, c'est à dire l'ensemble des entités persistantes de l'application (Facture, Bon de Commande, Client, ...)

4-La couche d'accès aux données contient les usines d'objets métier, c'est à dire les classes chargées de créer des objets métier de manière totalement transparente, indépendamment de leur mode de stockage (SGBDR, Objet, Fichiers, Legacy, ...) [7].

V-2-les plateformes J2EE/.NET

Les technologies J2EE/.NET sont deux implémentations différentes de ces couches d'architecture. Dans cette présentation nous avons donné une brève information sur le contenu

des couches de chaque technologie, le détail sera pris en considération après avoir fait le choix d'une plate forme en fonction de l'offre en Web Services de chacune d'elles.

La figure ci-dessous illustre leur mise en œuvre.

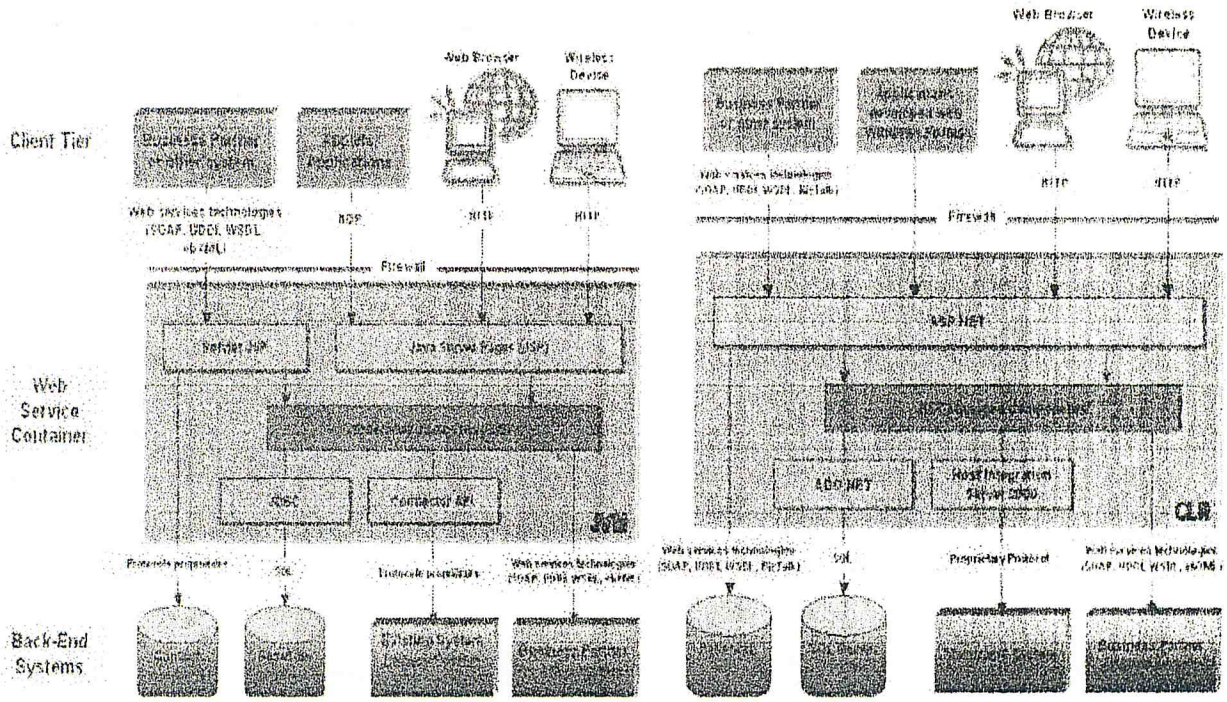


Figure V- 2 -la mise en oeuvre des technologies J2EE/.NET [7].

La CLR (Common Langage Runtime) est l'environnement d'exécution qui est équivalent à la JVM (Java Virtual Machine), un Logiciel qui simule un processeur Java.

V-2-1- Couche présentation :

	Les clients légers ou Web	Les clients lourds ou riches
J2EE	APIs Swing JFC l'AWT	Java Servlet API 2.3 et Java Server Pages 2.2
NET	WinForms (Windows Forms).	ASP.NET et WebForms

V-2-2- Couche de service :

J2EE	EJB Session
NET	ServicedComponent

V-2-3- Couche d'objets métiers (ou domaine) :

J2EE	EJB Entity
NET	objets C# ou VB simples

V-2-4- Couche d'accès aux données ou de Persistance :

J2EE	JAXP, JDBC, Java Persistence Connector(JCA)
NET	ADO.NET pour l'accès aux données

Pour une comparaison un peu plus en détails (cf.Annexe G.)

V-3-L'implémentation des Web Services dans J2EE/.NET :

V-3-1-Les Web Services dans .NET

Microsoft a effectué l'implémentation des Web Services dans la plateforme .NET de la manière suivante :

-La gestion de l'état conversationnel est gérée par **ASP.NET.**

-La communication basée sur HTTP est prise en charge par **IIS** (Internet Information Server).

-Les services tels que la montée en charge, la distribution ou les transactions sont gérés par le **Framework** à l'aide de **COM+**.

-le client : est un navigateur simple ou bien des **WinForms** ou des **WebForms**.

-la communication entre le client et le serveur s'effectue par une implémentation SOAP.

-l'authentification d'un service se fait à partir de la console **IIS ;** qui est considérée comme le chef d'orchestre de la sécurité, avec un Login et un mot de passe clair ou crypté.

-l'outils de développement : c'est le **Visual Studio .NET** ; il permet de créer graphiquement des WebServices avec des assistants. L'interaction entre différents WebServices via les proxy est prise en charge par l'outil sous la forme de **references web**. La création d'un WebService en C# est strictement la même qu'un composant VB.NET ou Cobol.NET.

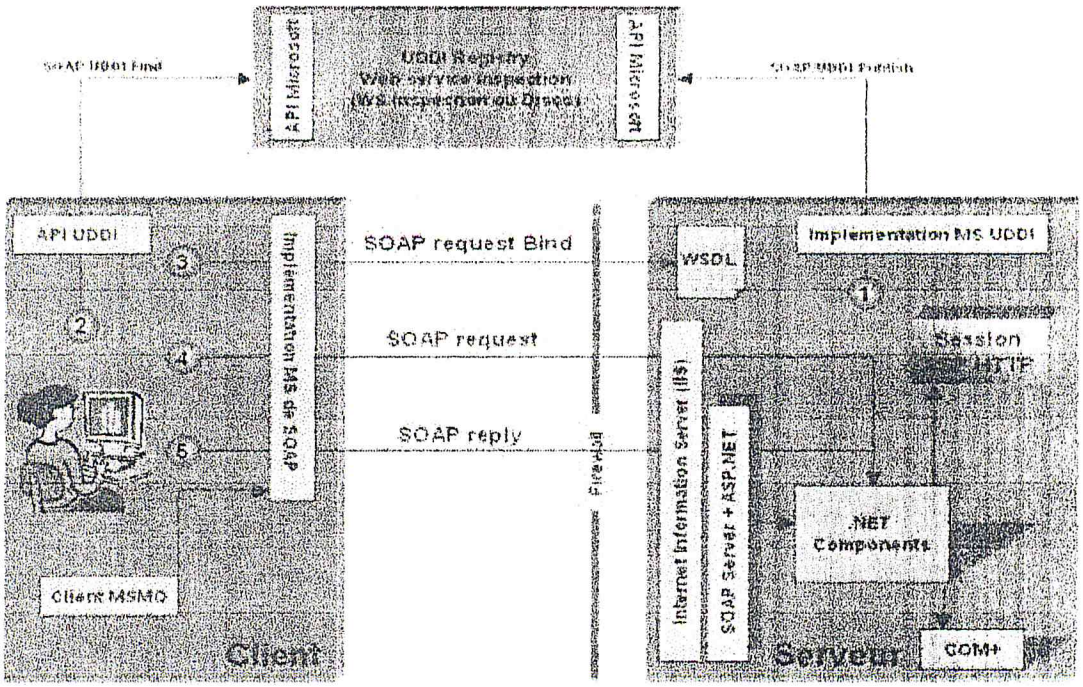


Figure V- 3-L'architecture WebServices .NET [7]

L'avantage tiré est l'homogénéité de l'ensemble de l'architecture permettant d'utiliser pleinement les ressources du système Windows et du Framework .NET,et qui a un impact positif sur les performances et sur l'intégration de l'outil Visual Studio avec la plate-forme. L'inconvénient est de posséder l'ensemble des produits de la gamme Microsoft .NET dont IIS.

V-3-2-Les Web Services dans J2EE:

Sun a pour objectif principal l'ouverture des WebServices à des implémentations tierces ; grâce aux nouvelles APIs : JAXM, JAXR, JAXB.

JAXM (Java API for XML Messaging) est une API qui standardise l'envoi et la réception de messages indépendamment des protocoles. Il est ainsi possible de l'utiliser pour véhiculer de l'ebXML avec un autre protocole que Soap.

JAXP (Java API for XML Processing) est une API intégrant DOM, SAX et XSLT. L'idée ici consiste à "brancher" n'importe quelle implémentation de parseur XML au dessus de JAXP (IBM, Sun, ...)

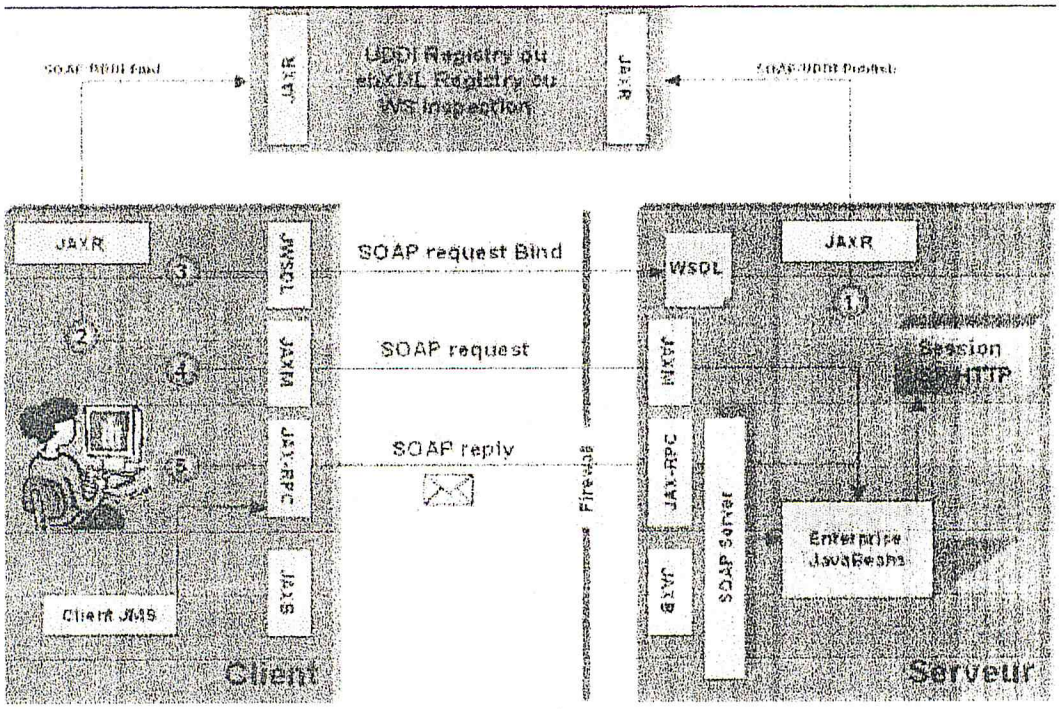


Figure V- 4-L'architecture WebServices J2EE

JAXR (Java API for XML Registries) spécifie l'interface entre un client et un serveur d'annuaire. Il existe deux implémentations :

-pour communiquer avec un serveur UDDI

-pour communiquer avec un repository ebXML.

JAXRPC (Java API for XML **RPC**) spécifie l'utilisation du modèle de développement de composants distribués basé sur le mécanisme de RPC (Remote Procedure Call) associé au format de messages SOAP. [7].

J2EE propose plusieurs outils liés au serveur d'application ; IBM propose un socle de développement : le **WebSphere ToolKit**, Sun propose le socle de développement : le **WSPack**(WebService**Pack**).

Cette plateforme propose des outils qui ne sont pas interopérables avec n'importe quel serveur d'application ; il faut attendre aussi l'intégration définitive des APIs standard, tandis que le besoin en Web Service est grand, les APIs ne sont pas encore crédibles.le seul produit qui intègre ces APIs est le WSPack c'est le seul RI (Reference Implementation).

V-4-Scénario d'utilisation d'un Web Service /.NET

Microsoft a développé l'outil Visual Studio qui permet de créer, tester, et consommer un Web service, le fonctionnement général est décrit dans la figure ci-dessous (Figure V-5):

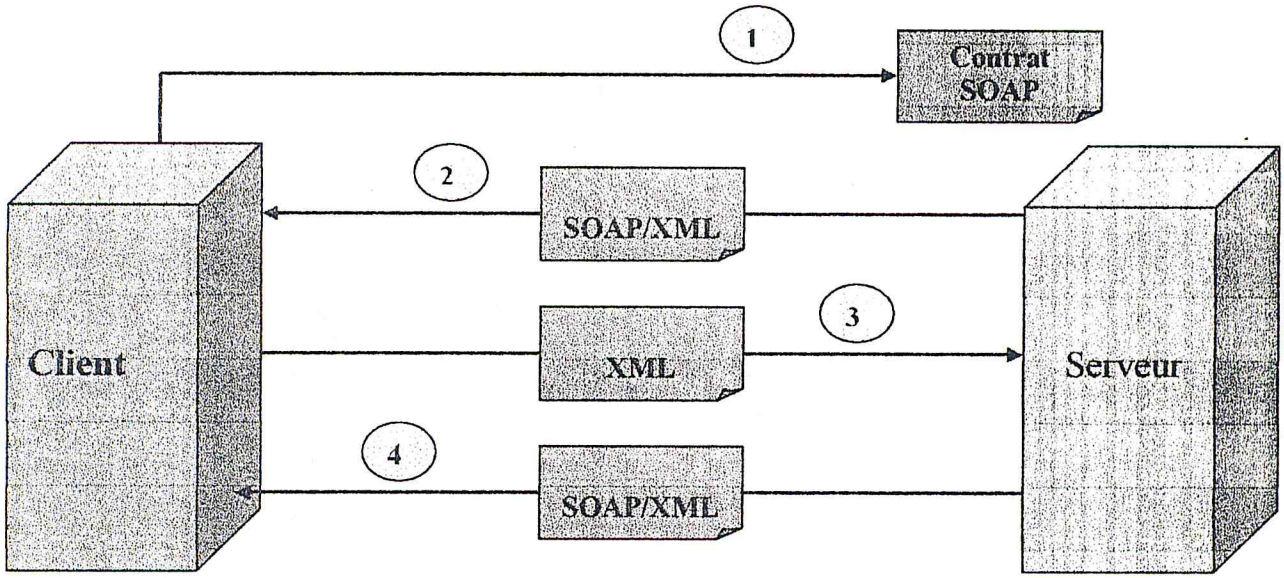


Figure V- 5- Etape de mise en œuvre en Visual Studio .NET [8]

Description :

1. le client demande quel service est proposé et dans quel format il doit être appelé pour construire et émettre une requête vers un Web service il faut connaître sa description qui est contenue dans un contrat SOAP décrit en WSDL, c'est un format de documents XML pour décrire les Web Services comme une suite de méthodes publiques accessibles depuis l'extérieur, leur nom, leurs paramètres et leurs types, il contient aussi les méthodes de transport des messages (SOAP, http GET, http POST).

2. le serveur présente le contrat (WSDL) correspondant : le document WSDL qui décrit le service est fourni, de façon automatique dans le Framework .NET en ajoutant " **?wsdl** " derrière l'url du Web Service: **http://localhost/nomduservice?wsdl**

3. le client a compris comment invoquer le service et envoie un document XML qui représente sa requête : Le client envoie une requête sous forme de document XML, qui sera transporté par le protocole HTTP ou SMTP, mais le problème qui se pose ; si le client ne parle pas du XML comment comprendre ? La réponse est dans des mécanismes de **marshalling-unmarshalling** réalisés par un **Proxy**, généré à partir du contrat WSDL. Le Proxy peut être écrit en C#, VB, JScript ou tout langage supporté par le Framework .NET ayant un générateur de code. Le Proxy **traduit les requêtes** du client en requêtes XML qui seront compréhensibles par le serveur Web et récupérer la réponse XML renvoyée par le serveur et la re-traduire dans au langage client. La figure ci-dessous (Figure V-6) représente le rôle de la classe proxy.

4. le serveur exécute la requête du client et lui retourne le résultat

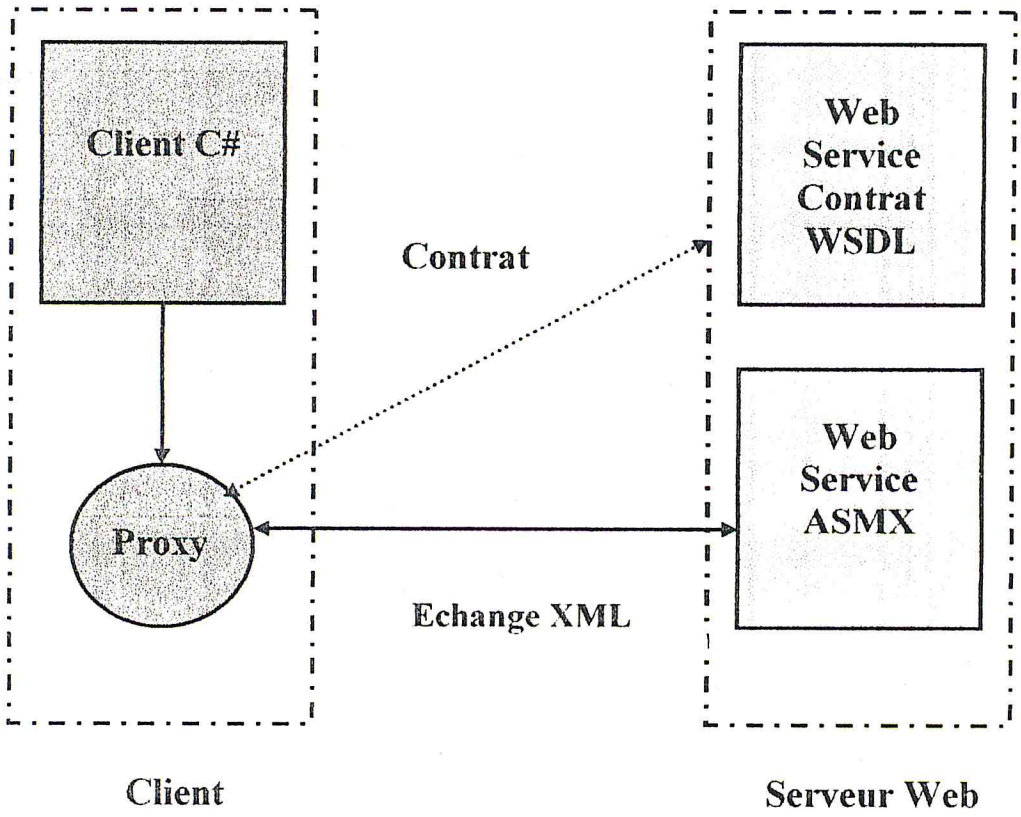


Figure V-6-rôle de la classe Proxy [8]

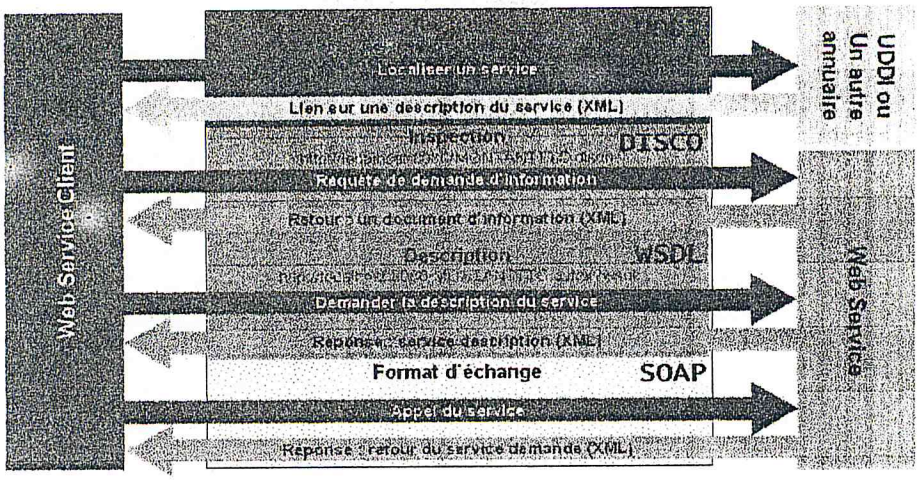


Figure V- 7-.Appel des Web service à travers Internet

V-5-Conclusion

La mise en œuvre des Web Services connaît certains points obscurs car Il n'est pas encore possible de propager un contexte transactionnel entre plusieurs Web Services, http s'y prête mal, la gestion de l'état conversationnel des objets distribués peut poser problème en cas de forte montée en charge, la plate forme doit prendre des mesures appropriées pour utiliser la session HTTP, ou directement une base de données en cas de sessions persistantes.

La plateforme .NET gère mieux les Web Services car elle possède un environnement intégré et qui fonctionne sous le système Windows. Le seul inconvénient est la non disponibilité de l'outil Visual Studio .NET . La plate forme J2EE est moins en avance que .NET; les APIs permettent plusieurs implémentations mais ne sont pas encore crédibles sachant que la création des Web Services en Java est aussi simple qu'en C# ou en Visual Basic. Mais encore un autre problème qui marque un point pour .NET est que les développeurs de J2EE sont familiers avec les concepts des langages de programmation et ils ont tendance à voir tout comme classes Java et Java datatypes pour construire la logique métier. Mais pour représenter les entités du monde réel et traiter les problèmes associés il faut arriver à modéliser des documents, des contrats, des personnes et toute information nécessaire pour une meilleure modélisation. Les WebServices sont le premier effort qui construit honnêtement une technologie qui arrive à une telle modélisation. Leur pouvoir réside dans la modélisation par documents pour construire des applications. C'est une technologie orientée document. Les documents sont une chose que J2EE ne sait pas encore mettre en œuvre. De plus, J2EE est un monde avec 6000 APIs [9]; c'est un environnement sophistiqué ; qui résout des problèmes complexes. Pour cela il est déconseillé d'utiliser tout le conteneur J2EE. La combinaison de Java et d'un moteur Servlet suffit pour *utiliser* les Web Services

Deuxième Partie :

Conception Et Réalisation

Pour cela notre choix est favorable pour la plate forme .NET qui possède une solution plus claire et un environnement intégré par rapport à J2EE. Celui-ci pose des problèmes de complexité dans la mise en œuvre car il faut assembler plusieurs APIs et savoir les combiner pour pouvoir répondre aux besoins.

Tout est dans le critère de choix ainsi donc la deuxième solution demande une étude un peu plus approfondie pour éclaircir ces lacunes d'APIs et d'architecture.

Chapitre VI : Web Services et Outils de Développement .Net

Dans la partie étude, le système qui veut adopter la technologie des Web Services devra impérativement supporter les standards associés à celle-ci. Il existe plusieurs systèmes complexes et évolués dans le domaine d'applications distribuées. Notre objectif est d'établir un système adapté à une architecture client/serveur adoptant la technologie des Web Services, qui incorpore tout standard concerné par cette technologie. Pour illustrer le fonctionnement des Web services le système à établir est fondé sur le modèle suivant :

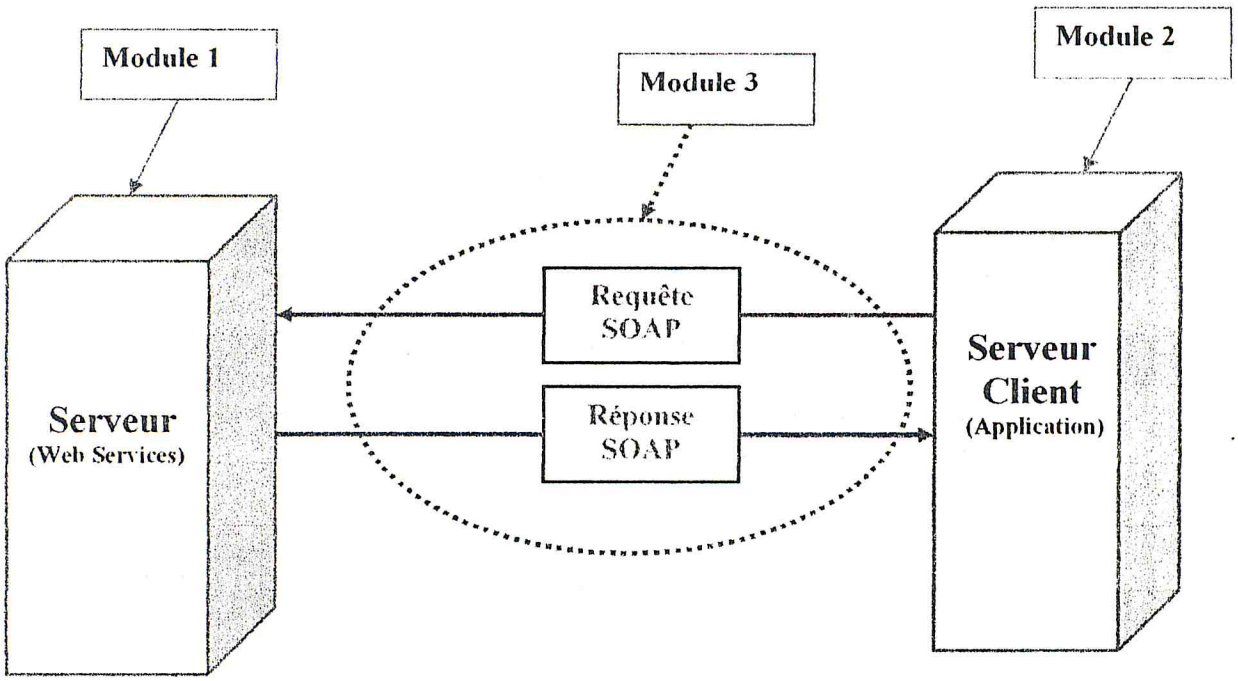


Figure VI - 1-Modèle de système basé sur les Web Services.

Module 1: Ce module comportera tous les Web Services que l'on voudra créer.

Module 2: Ce module comportera les clients consommant tous les Web Services du module1.

Module 3 : il comportera tous les messages de la communication entre client/ serveur.

Le principe de fonctionnement de ce système est le suivant :

Le serveur des Web Services (Module-1-) doit supporter les standards suivants : Le XML, langage de base pour les Web Services et le SOAP, protocole de communication basé sur XML ainsi que le WSDL, norme basée sur XML pour la description des Web Services.

Le client (Module-2-)consomme les Web Services via des applications clientes.

La communication (Module-3-) doit être établie entre client/serveur par des messages sous forme de requêtes /réponses.

1-le client envoie une requête au serveur de Web Services via le protocole SOAP.

2-le serveur identifie le Web Service demandé par le client, et envoie la réponse via le protocole SOAP. Chaque étape dans la communication fait intervenir un standard.

VI-1-Le Serveur des Web Services (Module 1) :

Nous voudrions créer plusieurs Web Services en vue d'être consommés, tous ces derniers seront fournis au client par ce serveur. Les Web Services sont composés d'une ou plusieurs méthodes, la définition des méthodes est selon la nature du service. Les services que nous créons différent dans le nombre d'opérations, nous allons définir trois Web Services :

1. Un Web service pour calculer le montant TTC (Toutes Taxes Comprises)
2. Convertisseur monétaire.
3. Calculatrice mathématique.

Le premier et le deuxième comprenant chacun une seule méthode, alors que le dernier Web service comprend quatre méthodes.

1-1- le premier Web Service : calcul du montant TTC.

Ce Web service comprend une méthode "**MTTC**" qui comprend deux opérandes d'entrées pour les valeurs Montant et TVA et comme sortie le montant TTC.

1-2-le deuxième Web Service : convertisseur monétaire.

Ce Web Service comprend une seule méthode. "**convert**" qui comprend trois entrées pour la conversion,la valeur monétaire à convertir ,le type de monnaie à convertir,le type de monnaie désiré dans la conversion. Le résultat est la valeur de conversion d'un type de monnaie vers l'autre.

1-3-le troisième Web Service : calculatrice mathématique.

Ce Web Service comprend quatre opérations de mathématique élémentaire: "**Add**" pour l'addition, "**Soustraction**" pour la soustraction, "**Multiplication**" pour la multiplication, "**Division**" pour la division. Chaque méthode comprend deux entrées pour les opérandes d'entrées de l'opération et une sortie pour l'opérande résultat.

VI-2-Les applications clientes : (Module 2)

Chaque Web service lui correspond une application cliente qui représente une interface de consommation. Ensuite nous intégrons toutes ces applications dans une application globale qui comprendra toutes les interfaces de ces applications secondaires créées. Pour désigner une seule application on utilise des liens vers ces applications secondaires. Les applications sont :

Pour la Calculatrice mathématique : une interface qui comportera des TextBox au même nombre de paramètres. Le premier champ pour l'entrée «a», le second pour l'entrée

«b» et le troisième champ pour recevoir le résultat envoyé par l'invocation de la méthode du Web Service. Il contiendra la valeur résultat de l'opération. Les opérations seront représentées dans quatre boutons pour les invoquer.

Pour le Convertisseur monétaire : une interface qui comportera des champs au même nombre de paramètres: un champs pour la valeur d'entrée, deux autres pour le sigle de monnaie à convertir ainsi que celui de la conversion. Un quatrième pour recevoir le résultat de la méthode de conversion. Ces deux applications seront intégrées dans la «Page de présentation» de l'application principale

Pour calculer le montant TTC (Toutes Taxes Comprises) : l'application principale qui consomme ce Web Service est une application simple de commande de microordinateur où il sera intégré pour illustrer un appel, par instanciation de la méthode "**MTTC**" à l'intérieur du code de l'application.

Chaque Web Service pour être consommé il lui faut une classe Proxy pour le traduire aux clients ainsi qu'une interface WSDL qui le décrit. Pour cela le système à établir prend la représentation suivante :

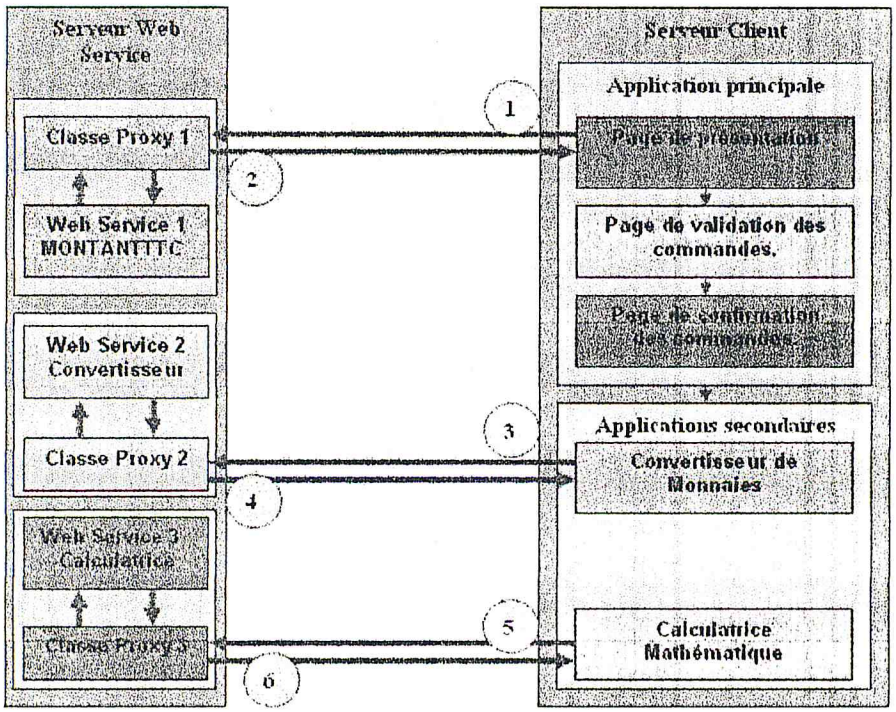


Figure VI - 2- l'interaction Web Services et les applications clientes.

1 Requête SOAP du service de calcul du montant TTC.
2 Réponse SOAP du service de calcul du montant TTC.
3 Requête SOAP du Convertisseur Monétaire.
4 Réponse SOAP du Convertisseur Monétaire.
5 Requête SOAP de la calculatrice mathématique.
6 Réponse SOAP de la calculatrice mathématique.

VI-3-les messages SOAP/XML (Module 3):

C'est la partie qui fait le lien entre l'application et le serveur. Elle permet la communication entre les deux modules. (1 et 2). Pour cela chaque message sera composé d'une requête et d'une réponse. Chaque message appartient à un Web Service. Chaque Web Service comprend le même nombre de message (requête/réponse) que les méthodes contenues dans celui-ci.

Dans le premier Web Service il y a quatre types de messages requête/réponse, alors que dans le second et dans le troisième un seul type de messages. Ces deux derniers Web Services comprennent une seule méthode. Une requête/réponse suffit. SOAP peut couvrir toutes les communications dans la mesure où il spécifie une structure de messages XML standard. Les messages SOAP/XML se composent d'une enveloppe qui comprend un entête et un corps. Tous les messages respectent la structure décrite dans (la Figure III- 1). On utilise un type pour les paramètres, on ne spécifie pas ce type mais on utilise le terme **SType** pour désigner un type simple qui sera au choix dans l'implémentation.

3-1-Messages de la calculatrice mathématique : Dans la requête il faut que le client envoie au service les paramètres a et b ; en réponse le service envoie le résultat contenu dans le paramètre c, le résultat dépend de l'opération demandée.

3-1-1-Méthode Addition : "Add" elle réalise l'opération suivante c :=a+b

Dans la requête il faut que le client envoie au service les paramètres a et b ; en réponse le service envoie le résultat contenu dans le paramètre c.

Requête SOAP :

```
<SOAP-ENV:Envelope
Xmlns:SOAP-ENV=«http://schemas.xmlsoap.org/soap/envelope"
SOAP-ENV: encodingStyle="http://schemas.xmlsoap.org/ soap/encoding">
<SOAP-ENV:Body>
      <Add>
          <a>SType</a>
          <b>SType</b>
      </Add>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Réponse SOAP :

```
<SOAP-ENV:Envelope
Xmlns:SOAP-ENV= " http://schemas.xmlsoap.org/soap/
Envelope "
SOAP-ENV :encodingStyle="http://schemas.xmlsoap.org/
soap/encoding">
<SOAP-ENV:Body>
        <AddResponse>
            <c> SType </c>
        </AddResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3-1-2-Méthode Soustraction:"Soustraction" elle réalise l'opération suivante c :=a-b

Requête SOAP :

```
<SOAP-ENV:Envelope
Xmlns:SOAP-ENV=«http://schemas.xmlsoap.org/soap/envelope"
SOAP-ENV: encodingStyle="http://schemas.xmlsoap.org/ soap/encoding">
<SOAP-ENV:Body>
        < Soustraction>
            <a>SType</a>
            <b>SType</b>
        </Soustraction>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Réponse SOAP :

```
<SOAP-ENV:Envelope
Xmlns:SOAP-ENV= " http://schemas.xmlsoap.org/soap/
Envelope "
SOAP-ENV :encodingStyle="http://schemas.xmlsoap.org/
soap/encoding">
<SOAP-ENV:Body>
        < SoustractionResponse>
            <c> SType </c>
        </ SoustractionResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3-1-3-Méthode Multiplication: "Multiplication" elle réalise l'opération suivante c :=a*b

Requête SOAP :

```
<SOAP-ENV:Envelope
Xmlns:SOAP-ENV=«http://schemas.xmlsoap.org/soap/envelope"
SOAP-ENV: encodingStyle="http://schemas.xmlsoap.org/ soap/encoding">
<SOAP-ENV:Body>
        < Multiplication >
            <a>SType</a>
            <b>SType</b>
        </ Multiplication>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Réponse SOAP :

```
<SOAP-ENV:Envelope
Xmlns:SOAP-ENV= " http://schemas.xmlsoap.org/soap/
Envelope "
SOAP-ENV :encodingStyle="http://schemas.xmlsoap.org/
soap/encoding">
<SOAP-ENV:Body>
        < MultiplicationResponse>
            <c> SType </c>
        </ MultiplicationResponse>
</SOAP-ENV:Body>
```

3-1-4-Méthode Division : "Division" elle réalise l'opération suivante c =a/b.

Requête SOAP :

```
<SOAP-ENV:Envelope
Xmlns:SOAP-ENV=«http://schemas.xmlsoap.org/soap/envelope"
SOAP-ENV: encodingStyle="http://schemas.xmlsoap.org/ soap/encoding">
<SOAP-ENV:Body>
        < Division >
            <a>SType</a>
            <b>SType</b>
        </ Division >
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Réponse SOAP :

```
<SOAP-ENV:Envelope
Xmlns:SOAP-ENV= " http://schemas.xmlsoap.org/soap/
Envelope "
SOAP-ENV :encodingStyle="http://schemas.xmlsoap.org/
soap/encoding">
<SOAP-ENV:Body>
        < DivisionResponse>
            <c> SType </c>
        </ DivisionResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3-2-Messages du Convertisseur Monétaire : Dans la requête il faut que le client envoie au service la valeur à convertir de type double par exemple contenue dans le paramètre «dbl Value» ainsi que l'indicateur de la monnaie à convertir de type «string» chaîne de caractères dans un paramètre «strFrom», et l'indicateur de la conversion désirée en réponse. Il sera contenu dans un paramètre «strTo», de type «string»le service envoie le résultat contenu dans le paramètre «ConvertResult», le résultat dépend de la conversion désirée. Par exemple convertir une valeur de l'Euro "€"en Dinar"DA".

Requête SOAP :

```
<SOAP-ENV:Envelope
Xmlns:SOAP-ENV=«http://schemas.xmlsoap.org/soap/envelope"
SOAP-ENV: encodingStyle="http://schemas.xmlsoap.org/ soap/encoding">
<SOAP-ENV:Body>
        < Convert >
            <DblValue>double</dblValue
            <StrFrom>string</strFrom>
            <strTo>string</strTo>
        </ Convert >
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Réponse SOAP :

```
<SOAP-ENV:Envelope
Xmlns:SOAP-ENV= " http://schemas.xmlsoap.org/soap/
Envelope "
SOAP-ENV :encodingStyle="http://schemas.xmlsoap.org/
soap/encoding">
<SOAP-ENV:Body>
        <ConvertResponse>
            <ConvertResult>double</ConvertResult>
        </ConvertResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3-3-Messages du service de calcul du montant TTC : Dans la requête il faut que le client envoie au service les valeurs Montant et TVA contenues dans les paramètres «Montant» et «TVA», le serveur envoie en réponse après invocation de la méthode «MTTC» le montant TTC contenu dans un paramètre de sortie «MTTCResult».tous les paramètres choisis sont de type **double**

Requête SOAP :

```
<SOAP-ENV:Envelope
Xmlns:SOAP-ENV=«http://schemas.xmlsoap.org/soap/envelope"
SOAP-ENV: encodingStyle="http://schemas.xmlsoap.org/ soap/encoding">
<SOAP-ENV:Body>
        <MTTC>
            <Montant> double</ Montant>
            <TVA>double</ TVA>
        </ MTTC>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Réponse SOAP :

```
<SOAP-ENV:Envelope
Xmlns:SOAP-ENV= " http://schemas.xmlsoap.org/soap/
Envelope "
SOAP-ENV :encodingStyle="http://schemas.xmlsoap.org/
soap/encoding">
<SOAP-ENV:Body>
      <MTTCResponse>
          <MTTCResult>double</MTTCResult>
      </ MTTCResponse >
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Ce système est basé sur la technologie des Web Services. Il fait intervenir tous les standards XML, SOAP, et WSDL. Pour ce dernier l'outil de la plateforme choisie le génère automatiquement et permet d'éviter des erreurs de développement. La plateforme .NET offre un outil qui répond à ce besoin. Aussi c'est une plate forme qui permet de séparer le code de l'interface. En séparant le code du Web Service de l'interface qui le décrit (c'est le fichier WSDL) et de celle qui le consomme (c'est l'application cliente). Une étude des technologies offertes par cette plateforme nous aidera à établir ce système. La classe Proxy est générée à partir du document WSDL, donc il est utile que le fichier WSDL ne contienne aucune erreur. Ainsi le client ne trouve pas de difficultés à consommer le Web Service demandé.

VI-4- La plateforme .NET et le « .Net Framework » :

Le *Framework* .NET est très récent et a été développé en parallèle avec le développement des Web services. Il contient toutes les librairies et tous les outils nécessaires pour la réalisation des Web services. Les espaces de nom *System.Web.Services* ou *System.Web.Protocols.Soap* contiennent les objets de base pour la sérialisation ou l'implémentation des services. Tous les Web services réalisés en .NET héritent de l'objet *System.Web.Services.WebService*.

Le code d'un Web service permet de spécifier des attributs pour les méthodes, les classes, etc. Ces attributs contiennent des informations pour les méthodes de sérialisation. Ainsi, un attribut *WebMethod* d'une méthode définit celle-ci comme méthode Web, qui peut alors être appelée par un client. Si une méthode n'est pas précédée par un attribut *WebMethod*, elle ne sera pas disponible pour le client. L'attribut *WebMethod* permet de modifier l'espace de nom XML de la méthode Web, son nom, s'il diffère de celui de la méthode de l'objet ou encore le fait que la méthode utilise les options de sessions du serveur Web.

L'attribut *WebService* quant à lui n'est pas obligatoire. Il permet principalement de modifier l'espace de nom du Web service. Si celui-ci n'est pas redéfini pour les méthodes ou les types, ceux-ci récupèrent celui du service.

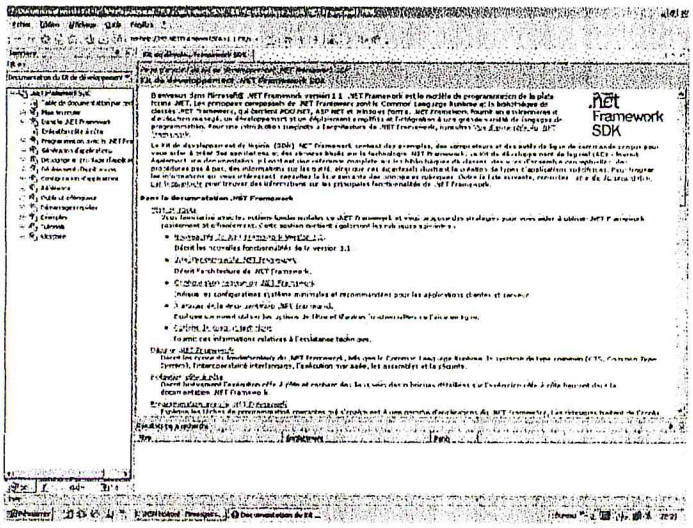


Figure VI - 3 Documentation du Kit de développement .NET Framework SDK

Un autre avantage du *Framework* .NET est le temps de développement, celui-ci est fortement diminué avec Visual Studio, mais dont l'acquisition de ce dernier est onéreuse. Mais, il existe des outils gratuits permettant également d'accélérer le développement de Web services tel que Asp.NET WebMatrix ou SharpDevelop.

Cette nouvelle plate-forme informatique simplifie le développement d'applications dans l'environnement fortement distribué d'Internet. Il est bien clair que le développement de Web service est bien intégré dans la technologie .NET, Ce qui devrait amener de nombreux développeurs à se rapprocher de celle-ci.

La plate-forme a été conçue pour atteindre les objectifs suivants :

- Fournir un environnement cohérent de programmation orientée objet que le code objet soit stocké et exécuté localement, ou exécuté localement mais distribué sur Internet ou pour être exécuté à distance.
- Fournir un environnement d'exécution de code qui minimise le déploiement de logiciels et de conflits de versionning.
- Fournir un environnement d'exécution de code qui garantit l'exécution sécurisée de code y compris le code créé par un tiers d'un niveau de confiance moyen ou d'un tiers inconnu.
- Fournir un environnement d'exécution de code qui élimine les problèmes de performance des environnements interprétés ou écrits en scripts.
- Fournir au développeur un environnement cohérent entre une grande variété de types d'applications comme les applications Windows et les applications Web.
- Générer toutes les communications à partir des normes d'industries pour s'assurer que le code basé sur le Framework .NET peut s'intégrer à n'importe quel autre code.

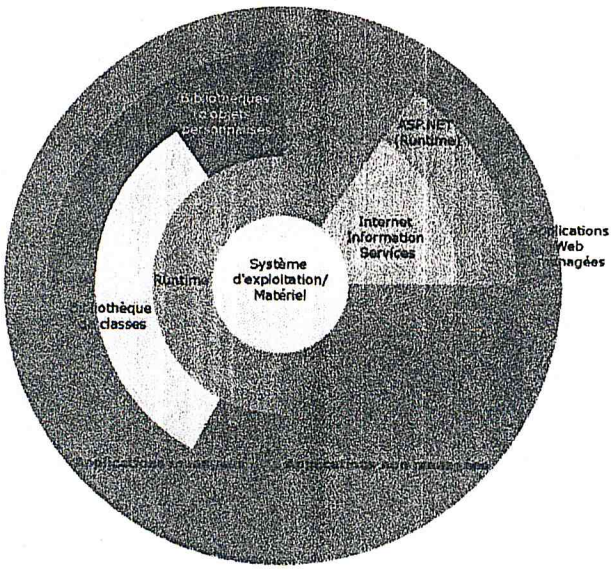


Figure VI - 4 - Vue d'ensemble du .Net Framework

Le Framework .NET contient deux composants principaux : la Common Language Runtime (CLR) et la bibliothèque de classes du Framework .NET. Le CLR est la base du Framework .NET, il peut être considéré comme un agent qui gère le code au moment de l'exécution, fournit des services essentiels tels que la gestion de la mémoire et l'accès distant. Il applique

également une stricte sécurité des types et d'autres formes d'exactitude du code qui garantissent un code sécurisé et robuste.

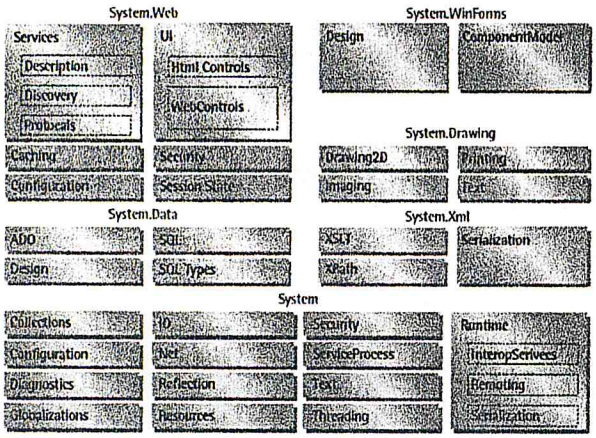


Figure VI - 5. Les classes du Framework

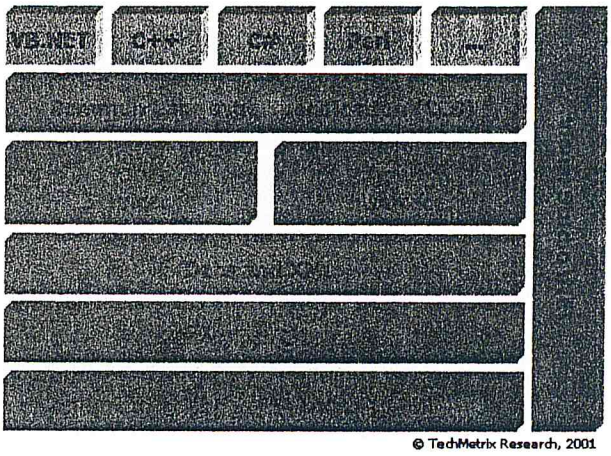


Figure VI - 6. L'architecture fonctionnelle de .NET

En fait, le concept de gestion de code est un principe fondamental du CLR. Le code qui cible le Runtime porte le nom de code managé (par opposition au code non managé).

Namespace	Description	Exemple de classes
System	Types de bases et la Console	Object, Buffer, Byte, Char, Array, Int32, Exception, GC, String
System.Collections	Collections d'objets	ArrayList, BitArray, Dictionary, Hashtable, Queue, SortedList, Stack
System.Data	Interaction avec les bases de données	DataBinding, DataRelation, DataRow, DataSet, DataTable, DataSource
System.Globalization	Types pour l'internationalisation (National Language Support - NLS), comparaison des chaînes de caractères et des calendriers	Calendar, CultureInfo, JulianCalendar, NumberFormatInfo, NumberStyles, RegionInfo
System.IO	Accès aux fichiers	ByteStream, File, FileStream, MemoryStream, Path, StreamReader, StreamWriter
System.Net	Accès au réseau	WebRequest, WebResponse, TcpClient, TcpListener, UdpClient, Sockets
System.Reflection	Inspection des méta-données	Assembly, ConstructorInfo, FieldInfo, MemberInfo, MethodInfo, Module, ParameterInfo
System.Runtime.Remoting	Types pour la gestion d'objets distants	ChannelServices, RemotingServices, IMessage, ImessageSink
System.Security	Types pour la gestion de la sécurité	Permissions, Policy, Principal, Util, Cryptography
System.Web.UI.WebControls	Composants Web	AdRotator, BorderStyle, DataGrid, HyperLink, ListBox, Panel, RadioButton, Table
System.WinForms	Composants graphiques (widgets) classiques	Button, CheckBox, DataGrid, FileDialog, Form, ListBox, MainMenu, MonthCalendar, NewFontDialog, RichEdit, ToolBarTreeView

Tableau VI - 1. Exemple de la Bibliothèque de classes [9]

La bibliothèque de classes, l'autre composant principal du Framework .NET, est une collection complète orientée objet, de types réutilisables que l'on peut utiliser pour développer des applications allant des traditionnelles applications à ligne de commande ou à interface graphique utilisateur (GUI, Graphical User Interface) jusqu'aux applications qui exploitent les dernières innovations fournies par ASP.NET, comme les services Web XML et WebForms.
Les principales composantes de l'environnement .NET, à savoir le CLR, CTS, CLS, IL, ADO.NET et ASP.NET sont détaillées dans les sections qui vont suivre :

VI-4-1- Common Language Runtime (CLR)

La Common Language Runtime (CLR) est un environnement d'exécution sécurisé et robuste qui supporte du code écrit dans plusieurs langages différents (C++, VB, C#, Pascal, Cobol ...) et simplifie le développement, la gestion et le déploiement d'applications. On peut la comparer à la Java Virtual Machine (JVM) ou au Runtime Visual Basic 6 (msvbvm60.dll). La CLR est constituée d'un ensemble de services standards (Modèle de programmation orientée objet, sécurité, ramasse miettes ou GC pour Garbage Collector) dont chaque programme .NET peut tirer profit.

VI-4-2- Common Type System (CTS)

Afin que des classes définies dans plusieurs langages puissent communiquer entres elles, elles ont besoin d'un ensemble commun de types de données. C'est l'objet de la CTS, elle définit les types de données que le Runtime .NET comprend et que les applications .NET peuvent utiliser. A noter que la CTS est un sur-ensemble de la CLS.

VI-4-3- Common Language System (CLS)

C'est un sous-ensemble de la CTS que chaque langage .NET est supposé supporter. Un programme qui utilise des types de la CLS peut interagir avec un autre programme .NET écrit dans un autre langage. Il est donc ainsi possible par exemple qu'une classe C# hérite d'une classe VB .NET.

VI-4-4- L'Intermediate Language (IL)

Tous les programmes .NET avant d'être déployés sont à compiler dans un langage de bas niveau appelé Intermediate Language ou Microsoft Intermediate Language (MSIL) : ce code IL est ensuite compilé dans du code natif au moment de l'exécution. Ce qui signifie que quelque soit le langage utilisé dans votre programme, vos exécutables et DLL seront toujours déployés sous la forme de code IL ; il n'y a donc aucune différence entre un composant écrit en C# et en VB .NET.

VI-4-5- Accès aux données avec ADO.NET

ADO.NET est un ensemble de classes qui exposent les services d'accès aux données au programmeur .NET. ADO.NET propose un large éventail de composants pour la création d'applications distribuées avec partage de données. Il fait partie intégrante du .NET Framework, il permet d'accéder à des données relationnelles, à des données XML et à des données d'application. L'ADO.NET répond à divers besoins en matière de développement, en permettant notamment de créer des clients de bases de données frontaux et des objets métier de couche intermédiaire utilisés par des applications, outils, langages ou navigateurs Internet.

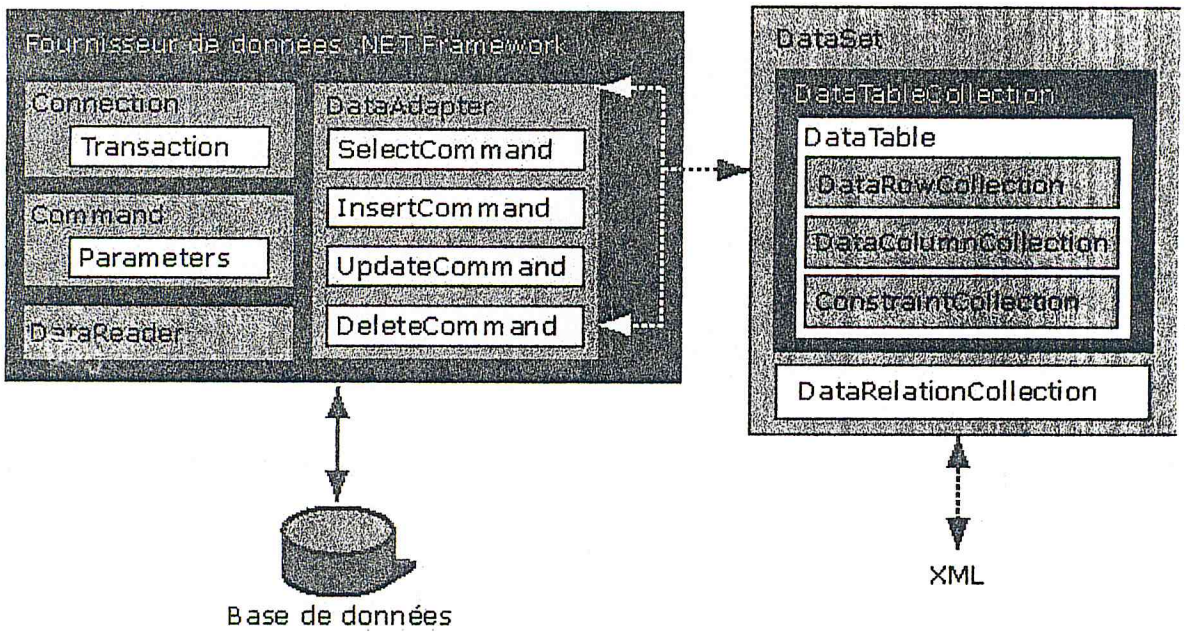


Figure VI - 7 –Vue d'ensemble de l'architecture et des composants de ADO.NET

Data set

Le **DataSet** ADO.NET est une représentation de données résidente en mémoire qui propose un modèle de programmation relationnel cohérent quelle que soit la source des données qu'elle contient. Un **DataSet** représente un jeu complet de données, y compris les tables qui contiennent, organisent et appliquent des contraintes aux données, ainsi que les relations entre les différentes tables.

L'utilisation d'un **DataSet** peut se faire via différentes méthodes qui peuvent être appliquées indépendamment les unes des autres ou combinées. Les différentes méthodes sont comme suit :

- Créer des **DataTables**, **DataRelations** et **Constraints** dans un **DataSet** et remplir les tables de données par programme.

- Remplir le **DataSet** de tables de données provenant d'une source de données relationnelles existante à l'aide d'un **DataAdapter**.
- Charger et rendre persistant le contenu du **DataSet** à l'aide de XML.

Un **DataSet** fortement typé peut aussi être transporté au moyen d'un service Web XML. Le design du **DataSet** le rend idéal pour le transport de données à l'aide des services Web XML.

XML Data set

Avec ADO.NET, il est possible de remplir un **DataSet** à partir d'un flux ou d'un document XML. On peut utiliser le flux ou le document XML pour fournir au **DataSet** soit des données, soit des informations de schéma ou les deux à la fois. Les informations fournies à partir du flux ou du document XML peuvent être combinées aux données ou aux informations de schéma déjà présentes dans le **DataSet**.

ADO.NET il est aussi possible de créer une représentation XML d'un **DataSet**, avec ou sans son schéma, afin de transporter le **DataSet** sur HTTP en vue de son utilisation par une autre application ou une autre plate-forme compatible XML. Dans la représentation XML d'un **DataSet**, les données sont écrites en langage XML et le schéma, s'il est inclus inline dans la représentation, est formulé en langage XSD (XML Schema Definition). XML et XSD proposent un format pratique pour le transfert du contenu d'un **DataSet** vers et à partir des clients distants.

VI-5- Choix de l'architecture ASP.NET, WebMatrix et MSDE

Les utilisateurs étant situés dans des lieux distincts, équipés d'infrastructures techniques différentes, il est naturel de retenir une architecture de type intranet, reposant un serveur HTTP qui génère des pages dynamiques à partir d'informations contenues dans une base de données. Ce type d'architecture nécessite de se déterminer sur les points suivants :

- choix de la technologie de développement Web.
- choix de l'environnement de développement.
- choix de la base de données.

VI-5-1-Choix de la technologie de développement Web

La technologie de développement constitue le moteur d'une application Web : c'est elle qui détermine le mécanisme de génération dynamique des pages Web ainsi que les systèmes d'exploitation, bases de données, langages et outils de développement utilisables.

Parmi les technologies existantes, les choix possibles sont les suivants:

Technologie	Description succincte
ASP	Technologie de génération de sites Web dynamiques introduite par Microsoft en 1996, qui repose sur l'interprétation de scripts et l'activation d'objets ActiveX/COM côté serveur ; fonctionne uniquement sous Windows ; généralement utilisée avec Microsoft SQL Server.
ASP.NET	Nouvelle technologie de développement d'applications et de services Web, introduite par Microsoft en 2001, qui repose sur des pages compilées, des contrôles prédéfinis, un mécanisme de gestion événementielle et une infrastructure technique intégrant des fonctionnalités de sécurité, optimisation et gestion des erreurs ; fonctionne uniquement sous Windows 2000 et XP équipé de l'extension .NET Framework ; généralement utilisée avec Microsoft SQL Server.
ColdFusion	Nom désignant à la fois une technologie de type serveur d'application et un outil de développement, qui repose sur l'utilisation d'un langage spécifique (CFML : Coldfusion Markup Language) permettant d'activer des objets sur le serveur lors de l'interprétation d'une page Web ; fonctionne sous Unix, Linux, Windows, avec la majorité des serveurs HTTP et des bases de données ; historiquement développé par Allaire et récemment acquis par Macromedia (ColdFusion MX).
JSP	Technologie de génération de sites Web dynamiques développée par Sun, qui repose sur l'utilisation de pages compilées contenant du Java et l'activation d'objets serveurs (JavaBeans) ; fonctionne sur toutes les plates-formes équipées d'une machine virtuelle Java.
PHP	Langage de script open source créé en 1994, actuellement dans sa version 4, dont la syntaxe est inspirée de C / Perl et Java, permettant la génération de sites Web dynamiques ; fonctionne sur la majorité des plates-formes Unix, Linux et Windows et est compatible avec un grand nombre de bases de données ; couramment utilisé au sein du trio Linux/Apache/MySQL.

Tableau VI 2. Différente technologie de développement Web. [6]

Aujourd'hui, toutes ces technologies ont atteint un niveau de maturité assez avancé. Néanmoins, ASP.NET se distingue de ses concurrents par son architecture novatrice et son nouveau modèle de programmation qui permet la réalisation d'applications Web avec les mêmes méthodes de conception (découpage en objets métier et composants réutilisables), le même degré de robustesse et de puissance (orientation objet, typage fort des variables, compilation) et le même degré d'organisation du code (séparation entre code et contenu graphique, organisation en gestionnaires d'événements) qu'une application client-serveur classique. Nous allons détailler les différentes caractéristiques de cette technologie.

Qu'est ce que Asp.Net

ASP.NET est une plate-forme de développement Web unifiée qui offre les services nécessaires pour générer des applications Web d'entreprise. ASP.NET n'inclut pas seulement une syntaxe fortement compatible avec ASP (Active Server Pages). Il fournit également un nouveau modèle de programmation et une nouvelle infrastructure qui permettent de créer une nouvelle catégorie d'applications puissantes. Intégré au .NET Framework, ASP.NET permet de tirer pleinement parti des fonctionnalités du Common Language Runtime, telles que la sécurité de type, l'héritage, l'interopérabilité entre les langages et le versioning.

Le nom initial de la technologie ASP.NET était ASP+, et les pages ASP.NET portent l'extension .aspx (pour : ASP eXtended). En pratique, ASP.NET est une extension logicielle qui permet l'exécution d'applications Web (sites Web dynamiques, services Web). Disponible en téléchargement gratuit sur le site www.asp.net et destinée à être installée sur une machine équipée de Windows 2000/XP et d'un serveur HTTP, elle se compose des éléments techniques suivants :

- une extension du serveur HTTP, implémentée sous la forme de filtre ISAPI nommé aspnet_filter.dll qui permet de traiter spécifiquement les requêtes vers les pages comportant une extension particulière (notamment les fichiers .aspx, qui correspondent aux pages ASP.NET standards) ;
- un processus principal nommé aspnet_wp.exe (pour : ASP.NET worker process) qui tourne en tâche de fond et traite les requêtes correspondant à des pages ASP.NET en s'appuyant lui-même sur trois éléments : l'environnement d'exécution des applications .NET constitué du Common Language Runtime (CLR), de la bibliothèque de classes .NET et des compilateurs .NET ;

Donc de par son architecture, ASP.NET est indissociable du Framework .NET. La compilation d'une page ASP.NET n'a lieu que lors du la première requête effectuée vers cette page; l'assemblage produit est alors gardé en cache pour les requêtes successives (à moins que le fichier source ne soit modifié). Ceci garantit une performance optimale lors de l'exécution de l'application.

La figure qui suit illustre le mécanisme d'appel d'une page ASP.NET:

- Lorsque le serveur HTTP reçoit une requête à destination d'une page portant l'extension .aspx, celle-ci est automatiquement traitée par le filtre ISAPI (aspnet_filter.dll) et transmise au processus principal ASP.NET (aspnet_wp.exe).

- La page demandée est alors compilée sous la forme d'un module objet en langage MSIL (Microsoft Intermediate Language) par le compilateur correspondant au langage utilisé dans la page (C#, VB.NET et JScript.NET sont les trois langages utilisables en standard) puis transmise au Common Language Runtime (CLR) qui réalise l'édition de lien avec les modules objet des classes utilisées de la bibliothèque .NET et la production d'un assemblage (composant binaire exécutable) dont l'exécution conduit à la production d'un contenu HTML, transmis en retour au navigateur.

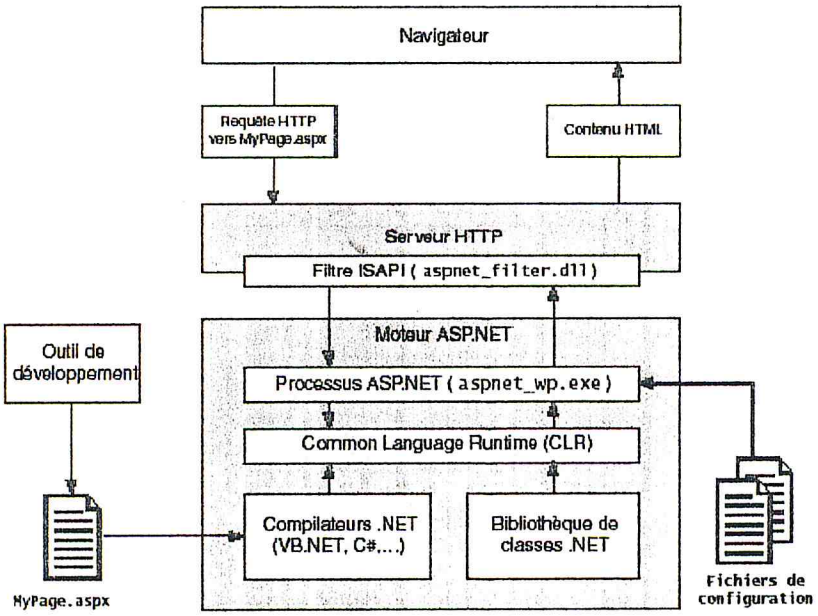


Figure VI - 8. Traitement d'une requête vers une page ASP.NET. [6]

Notons que la configuration du processus principal ASP.NET (langue, sécurité, gestion des erreurs) est entièrement paramétrable à l'aide de fichiers texte au format XML (machine.config et web.config), dont les changements sont détectés et pris en compte dynamiquement, sans nécessiter de redémarrage du serveur HTTP.

Nous avons décrit succinctement l'infrastructure d'exécution des pages ASP.NET: détaillons maintenant les nouvelles possibilités offertes au développeur pour la réalisation de pages Web.

Le .NET Framework ne dispose pas d'interface graphique IDE pour éditer des pages ASP.NET et des Services Web, donc il est nécessaire d'utiliser n'importe quel éditeur de texte pour effectuer cette tache. Cependant il existe un logiciel totalement gratuit « WEBMATRIX » développé par Microsoft et qui permet d'éditer des applications Web et des Webservices

VI-5-2-Choix de l'environnement de développement ASP.NET Web Matrix

Il existe des environnements de développement qui se proposent d'automatiser les tâches laborieuses à l'aide d'assistants, de faciliter le confort de travail du développeur (coloration syntaxique, complétion automatique du code saisi, compilateurs intégrés, aide en ligne), parmi ces outils on trouve Visual Studio.NET, Delphi 7.0 et Web Matrix

Dans le cadre d'une grosse équipe de développement travaillant sur des projets importants, l'emploi de Visual Studio.NET ou de Delphi serait probablement nécessaire ; pour notre étude, limitée à une application ASP.NET consommant des Web Services, réalisée a priori par un ou deux développeurs, Web Matrix répond amplement aux besoins.

ASP.NET WebMatrix (anciennement projet **Saturn**) est un outil de développement léger pour créer des pages Web dynamiques ASP.NET. Il intègre des fonctionnalités permettant aux développeurs d'échanger des informations entre eux et d'accélérer leur développement.

A l'instar du PHP ou du JSP, le langage ASP permet de développer des sites Internet dynamiques. Fonctionnant avec l'environnement d'exécution .NET Framework, la version .NET du langage ASP exécute du code compilé (en Visual Basic ou C#) sur le serveur et permet de créer, ou de consommer, des services Web.

L'intégration avec la messagerie instantanée et les chats sont des innovations importantes dans le domaine des IDE. L'aide est également basée sur les contributions de la communauté des développeurs ASP.NET. Le programme d'installation ne fait que 1,3 Mo.

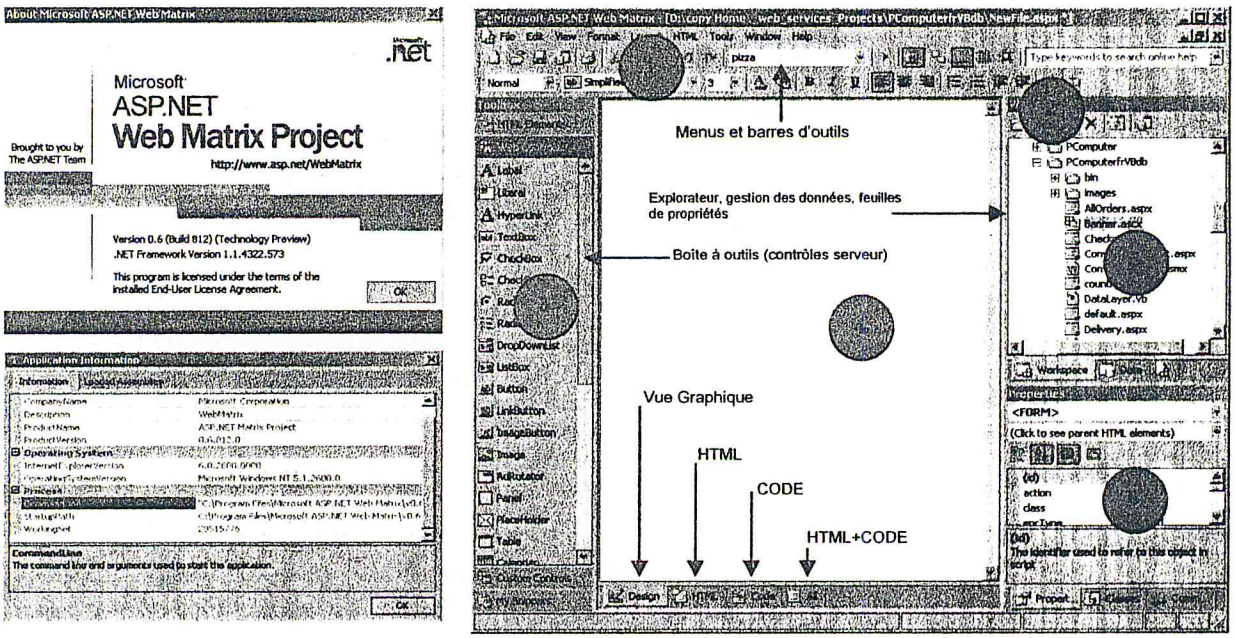


Figure VI - 9.Concepteur de pages ASP.NET WEBMATRIX

WebMatrix Project inclut un serveur Web de test, écrit avec le langage C. Il intègre l'outil de messagerie instantanée Microsoft Messenger pour faciliter le dialogue entre développeurs ainsi qu'une liste de sites Internet consacrés au développement Web. Microsoft autorise la mise en production de l'outil mais ne fournit aucun support pour WebMatrix. Contrairement aux outils Visual Studio .NET, WebMatrix ne possède pas d'outils de debuggage intégrés, de services de développement collaboratif et n'intègre pas la fonction IntelliSense, permettant d'accéder rapidement aux commandes d'un langage de programmation.

Le Concepteur de pages Web ASP.NET (Figure VI-7) permet de créer rapidement des pages ASP.NET avec le concepteur **WYSIWYG** (What You See Is What You Get) il est aussi RAD (Rapid Application Development). Il suffit de glisser les contrôles serveur depuis la boîte à outils vers votre page. Un double clique sur un contrôle serveur suffit pour générer automatiquement le code de gestion de l'événement associé en Visual Basic, C# ou J#. Le concepteur de pages est constitué de plusieurs parties dont la description est la suivante :

1. le **Menu** permet d'accéder aux commandes (**File**, **Edit**, **View**, **Format**, **Layout**, **Html**, **Tools**, **Windows** et **Help**)
2. la **Barre d'Outil** : zone regroupant les raccourcis des commandes fréquemment utilisées.
3. la **Boite à outil** contient différentes sections (Html Elements, Web Controls, Custom Controls, Code Wizards, My snippets
4. Zone de travail contenant quatre onglets (**Design**, **Html**, **Code** et **All**)
5. cette section contient trois onglets : **Workspace**, **Data** et **Open Items**.
6. cette section contient trois onglets : **Properties**, **Classes** et **community**.

Les différents types de fichiers proposés sont comme suite :

Page Asp.net (.apsx)	Contient du html et du code serveur
Contro Utilisateur Asp.net (.ascx)	Contient du code serveur
Page Html (.htm)	Contient du html
Service Web XML (asmx)	Contient du code, et il faut spécifier une classe et un namespace
Classe (.cs, .vb et même .js)	Contient du code, et il faut spécifier une classe et un namespace
Fichier global.asax	Accès à des fonctions particulières d'Asp.
Fichier web.config	Configuration des pages exécutées
Fichir XML (.xml)	Contient du XML
Fichir XSL (.xslt)	Contient du XSLT
Fichir XSD (.xsd)	Contient du XSD
Fichir texte (.txt)	Contient du TXT
Script SQL (.sql)	Contient du SQL

Tableau VI-3. Les différents types de fichiers du Concepteur de pages Web ASP.NET.

En plus de ces fichiers standard, WebMatrix donne aussi accès à des modèles de pages Asp pour la liaison des données, des applications mobiles, la gestion du cache de sortie d'Asp, des pages liées à la sécurité et des modèles pour les Web Services.

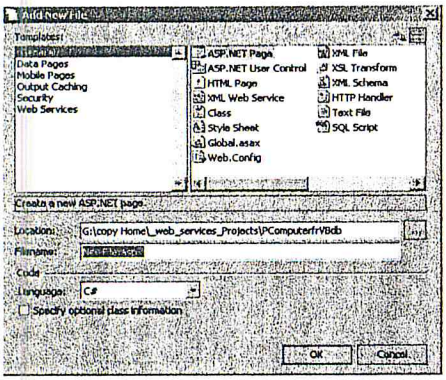


Figure VI - 10- Fichiers généraux.

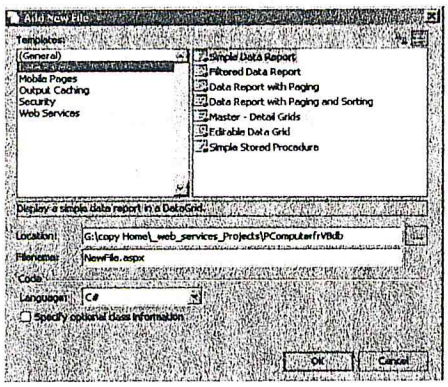


Figure VI - 11- Fichiers de données.

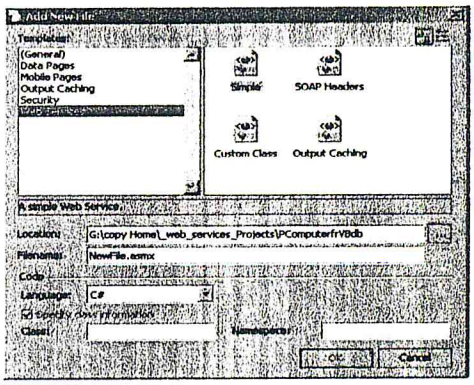


Figure VI - 12- Fichiers WebServices.

Donc après cette breve description on peut résumer les fonctionnalités proposées par WebMatrix comme suit:

- **L'édition de pages** ASP.NET en mode **graphique.**
- La connexion vers des **bases de données** avec **MSDE** (SQL Serveur light).
- La gestion des évènements sur les **WebForms** à travers différentes vues.
- **L'édition de pages HTML** à la manière de Visual Studio .NET.
- **Un atelier de conception** de formulaires HTML via drag & drop en s'appuyant sur des tables et des procédures stockées.
- La création de **WebServices** à l'aide d'assistants en tout genre.

Ce que WebMatrix ne fait pas :

- **La gestion de configuration** intégrée (projets, partages des sources, déploiement).
- La conception d'interfaces graphiques **WinForms.**
- **La complétion des méthodes** (dans la version que nous avons testée)
- La séparation entre la classe du CodeBehind (traitements) et la page HTML. (présentation) : L'outil s'obstine à tout insérer dans une seule et même page pour faciliter les opérations d'édition.
- Le **déboguage** de pages ASP.NET intégré, le déploiement automatisé, la complétion des méthodes.

La **boite à outils** contient les contrôles permettant de construire nos applications ; c'est un menu contextuel (change avec le document présent dans la zone de travail). Un simple glisser-déplacer permet d'ajouter un contrôle.

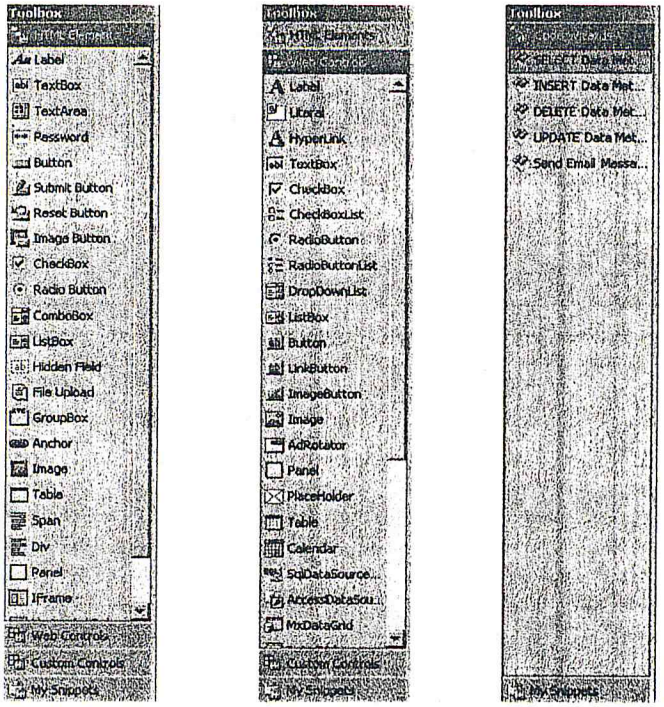


Figure VI - 13- ASP.NET WebMatrix ToolBox

Serveur Web de développement : On peut avec cet outil développer et tester nos applications ASP.NET sans et avec le serveur IIS (Internet Information Services). WebMatrix inclut un serveur Web léger qui gère les pages ASP.NET et les Services Web, pour les requêtes locales uniquement. En effet il n'accepte pas les connexions distantes pour des raisons de sécurité. Ce serveur utilise le port **8080** par défaut. Néanmoins il est aussi possible d'utiliser IIS en spécifiant un nom de répertoire virtuel.

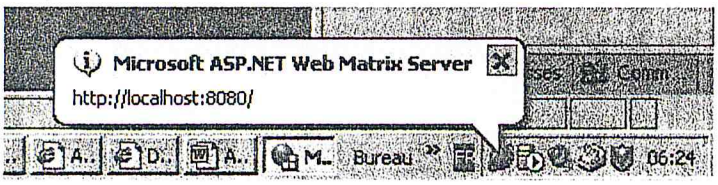


Figure VI - 14- Serveur Web de développement

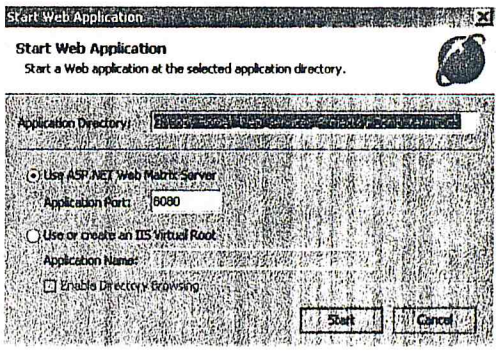


Figure VI - 15- Configuration du serveur

Forte intégration d'outils communautaires : WebMatrix fournit aussi des liens directs vers les sites, newsgroups et forums de discussion dédiés à ASP.NET. L'aide la plus appropriée est disponible plus rapidement en toute simplicité.

VI-5-3-Choix de la base de données

Dans une application ASP.NET comme dans la grande majorité des sites Web dynamiques, la base de données est une composante fondamentale de l'architecture.

L'offre logicielle en la matière est vaste ; parmi les produits les plus connus disponibles sous Windows, citons : Microsoft SQL Server, Oracle, PostgreSQL, Informix, 4D, Microsoft Access et bien d'autres.

À l'heure actuelle, la bibliothèque .NET propose les types d'accès suivants :

- accès natif à Microsoft SQL Server (et à sa version allégée : MSDE) ;
- accès natif à Oracle (fournisseur disponible en téléchargement séparé) ;
- pilote OLE-DB;
- pilote ODBC (fournisseur disponible en téléchargement séparé).

Pour notre étude de cas, nous nous proposons d'utiliser la base Microsoft SQL Server Desktop Engine (MSDE) en raison de sa gratuité.

MSDE : une version allégée et gratuite de SQL Server 2000

Microsoft SQL Server Desktop Engine (MSDE) est une version allégée et gratuite de Microsoft SQL Server, doté d'un support complet de Transaction SQL (y compris la gestion des transactions, des déclencheurs et de la sécurité). En revanche, elle n'offre qu'un support partiel de la réplication, ne dispose pas d'outils graphiques d'administration (comme SQL Enterprise Manager ou Analyseur de requête) et son usage reste limité à un nombre réduit d'utilisateurs simultanés.

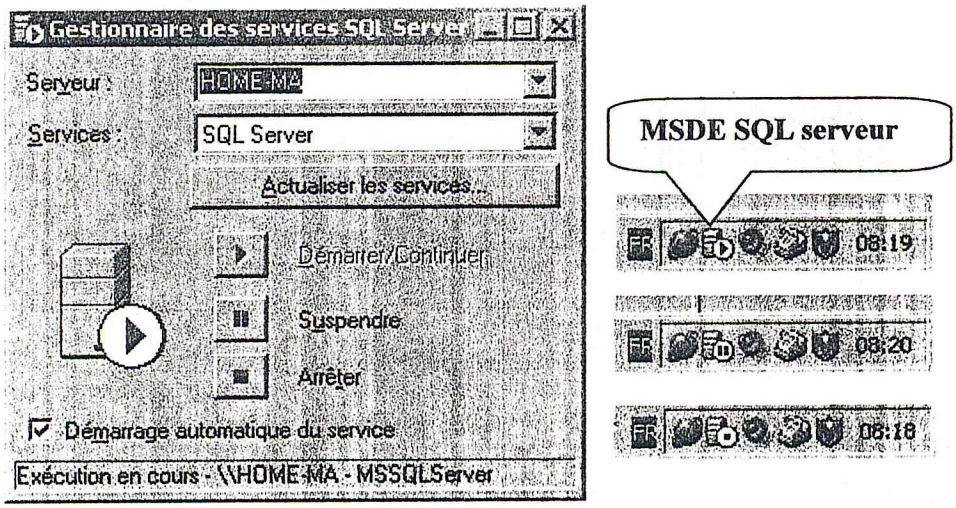


Figure VI - 16 - Microsoft SQL Server Desktop Engine (MSDE)

Cette base constitue néanmoins une solution bien adaptée pour un poste de développement, du fait de sa faible consommation de ressources, d'une part, mais également de sa compatibilité

totale avec SQL Server 2000 d'autre part ce qui facilite un éventuel déploiement vers un serveur de production doté de ce SGBD.

La version précédente du moteur MSDE (1.0) était différente de SQL Server 7.0 ; désormais, MSDE 2000 et SQL Server 2000 sont dotés du même moteur.

Génération simplifiée d'interfaces pour bases de données

WebMatrix permet de créer des pages intégrant des données dynamiques sans la moindre écriture de code. Il suffit de glisser des tables SQL dans la page pour créer des tableaux de données dynamiques.

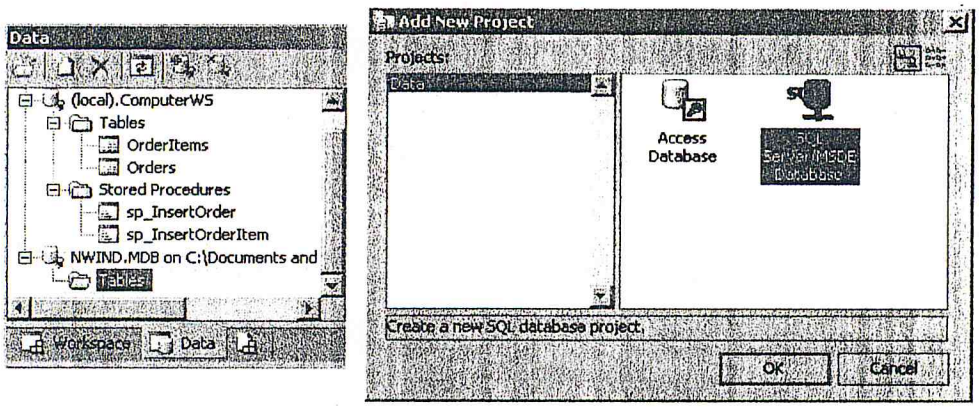
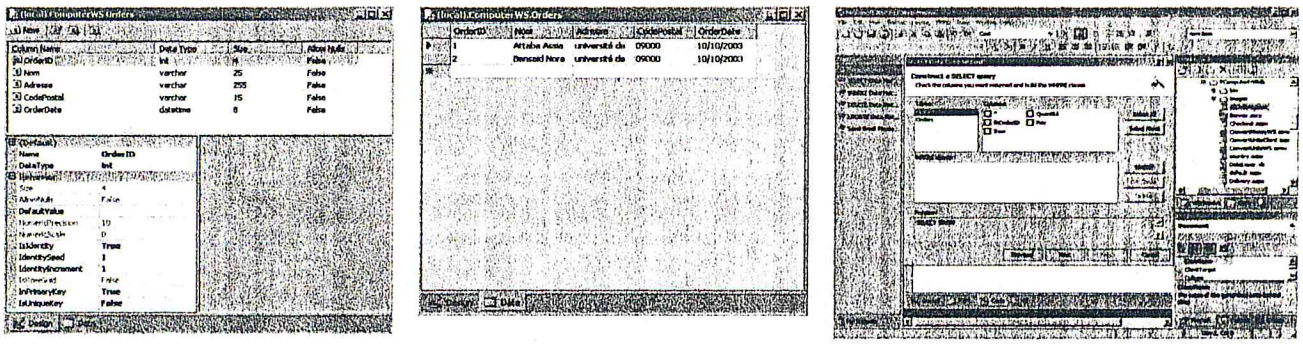


Figure VI - 17- Accès aux base de données avec WebMatrix

Il est possible également de les créer à partir d'un des nombreux modèles de pages dynamiques ; Les assistants aident à générer le code pour les opérations **Select**, **Insert**, **Update** et **Delete**. Il est donc possible d'éditer directement toutes les données depuis l'IDE de WebMatrix. Générer automatiquement le code ADO.NET associé, créer de nouvelles bases de données, ajouter, modifier et supprimer les tables et procédures stockées.



a- Nouvelle table **b- Edition de la table** **c- Requête SQL**

Figure VI - 18- Accès et création de base de données SQL avec WebMatrix

VI-5-4-Configuration materiel ET logiciel :

Dans notre cas les Web services ont été développés ainsi que les applications Asp.Net sur une plateforme Windows 2000 sur P-III 750MHz avec 192Mo de RAM ainsi que Windows XP sur P-IV 1700MHz avec 256 RAM. Sur les deux machines Microsoft Internet Explorer 6.0 et Microsoft Windows Installer 2.0 avait été installés ainsi que Microsoft Internet Information Services (IIS) 6.0 (ce dernier n'a pas été utilisé et a été remplacé par Cassini le serveur de WebMatrix pour les applications locales seulement). Le fournisseur de données SQL Server .NET avec Microsoft Data Access Components (MDAC) 2.7. Pour plus de détails sur les configurations minimales et requises (cf.AnnexeH).

Installation de l'outil de développement:

- Microsoft Windows 2000 ou XP
- Microsoft Internet Explorer 6.0
- Microsoft Windows Installer 2.0
- Microsoft Internet Information Services (IIS) 6.0 existant sur Windows 2000 ou XP.
- .NET Framework Version 1.1 Redistributable Package (23Mo), ce package redistribuable comprend tout ce dont les développeurs ont besoin pour exécuter les applications développées à l'aide du .NET Framework, téléchargeable du site : http://www.asp.net/webmatrix/ ou bien http://download.microsoft.com/download/e/d/a/eda9d4ea-8ec9-4431-8efa-75391fb91421/dotnetfx.exe
- Kit de développement *Framework* .NET SDK Version 1.1 (111Mo) comprend tout ce dont les développeurs ont besoin pour écrire, générer, tester et déployer des applications .NET Framework — documentation, exemples et outils de ligne de commande et compilateurs, téléchargeable du site : http://www.asp.net/webmatrix/ ou bien http://download.microsoft.com/download/3/5/d/35d3a9ee-b371-4843-b14a-b7bc829ad7d4/setup.exe
- Microsoft ASP.NET Web Matrix Project 1.1.4322.573 (1.3Mo) du site : http://www.asp.net/webmatrix/
- Microsoft Data Access Components (MDAC) 2.7. Ce logiciel est requis pour les applications qui utilisent l'accès aux données. téléchargeable du site : http://www.microsoft.com/downloads/details.aspx?familyid=9ad000f2-cae7-493d-b0f3-ae36c570ade8&displaylang=fr

- Microsoft SQL Server 2000 Desktop Engine (MSDE): SQL Server 2000 Service Pack 3 « *FRN_SQL2kdesksp3.exe SP3a* » ; téléchargeable du site: http://www.asp.net/webmatrix/ ou bien http://www.microsoft.com/downloads/details.aspx?FamilyId=90DCD52C-0488-4E46-AFBF-ACACE5369FA3&displaylang=fr#filelist

VI-6-Conclusion

Dans ce chapitre nous avons défini l'offre matérielle et logicielle de la plateforme .NET en Web Services ainsi que toutes les possibilités de développement. C'est une plateforme qui possède toutes les caractéristiques qui simplifient le développement dans un environnement distribué et qui peut être utilisé sur Internet. Ainsi le développement des Web Services est intégré dans le Framework. L'exécution du code peut être sécurisée et peut s'intégrer aux codes produits par l'industrie logicielle ou matérielle autre que .NET. Ses principales composantes sont :

- La CLR pour l'exécution des codes managés, il gère la mémoire, l'accès distant et assure la sécurité.
- La Bibliothèque de types qui permet de développer des applications Web à savoir des Web Services et des WebForms.

Parmi plusieurs offres nous avons choisi l'architecture basée sur la technologie ASP.NET pour la génération dynamiques des pages Web (pour obtenir des sites dynamiques).c'est le moteur d'une application Web qui permet aussi de déterminer le système d'exploitation et la base de données nous avons choisi la MSDE une version allégée pour exécution sur un seul poste de développement, le cas de notre système. En exploitant les classes d'ADO.NET du Framework on peut avoir accès aux données représentées avec les Data Set et XML Data pour notre application.

Ensuite un choix d'environnement de développement a été nécessaire pour faciliter les taches de développement. L'outil WebMatrix permet de l'accélérer et il fonctionne sur le Framework .NET c'est l'interface graphique qui permet de créer des WebServices avec la technologie ASP.NET.

Chapitre VII : Réalisation

VII-1-Comment créer un Web service en .Net ?

Les Web services ont accès aux mêmes options de gestion d'état que les autres applications ASP.NET lorsque la classe implémentant le Web service dérive de la classe WebService. La classe WebService contient de nombreux objets ASP.NET communs.

La forme type d'un Web service sous .Net est comme suit:

1- La première ligne permet de spécifier que le fichier est un Web service (fichier ASMX), le type de langage utilisé et le nom de la classe, de plus de tout ça on peut ajouter le mode debug activé ou non pour détecter une éventuelle erreur lorsqu'on fait nos tests. Ceci se fait par la directive @ ***WebService*** dont la description est comme suit:

```
<%@ WebService attribute="value" [attribute="value"...] %>
```

<%@ WebService language="C#" class="MONTANTTTC" %>

<%@ WebService language="VB" class="MathService" %>

Attribut	Description
Class	Spécifie la classe implémentant le Web service qui est automatiquement compilé lors du premier accès au Web service après modifications. Cette valeur peut correspondre à tout nom de classe valide résidant soit dans le même fichier que la directive **WebService**, soit dans un fichier distinct. Si la classe réside dans un fichier distinct, celui-ci doit être placé dans le répertoire \Bin de l'application Web où réside le Web service. Cet attribut est requis pour la compilation du Web service.
CodeBehind	Spécifie le fichier source qui implémente le Web service, lorsque la classe implémentant le Web service ne réside pas dans le même fichier et n'a pas été compilée en assembly et placée dans le répertoire \Bin.
Debug	Indique si le Web service doit être compilé avec des symboles de débogage. **True** si le service Web XML doit être compilé avec des symboles de débogage ; sinon, **false**.
Language	Spécifie le langage utilisé lors de la compilation de l'ensemble du code incorporé dans le fichier du Web service (.asmx). Ces valeurs peuvent représenter tout langage pris en charge par .NET, notamment **C#**, **VB** et **JS** (C#, Visual Basic .NET et JScript .NET).

Tableau VII- 1-Description des attributs de la directive <%@ *WebService*>

2- Importer les espaces de nom à l'aide des mot-clefs "**Imports**" et "**using**" en VB.NET et C# respectivement, tel que l'espace de nom *System.Web.Services* et *System.Xml.Serialization*.

3- Déclarer la classe qui représente notre Web service et qui dérive de la classe «**WebService**» de l'espace de nom «**System.Web.Services**», ce qui permettra à cette classe d'accéder aux méthodes de la classe «**WebService**» aussi bien qu'à ces

attributs. Dans notre exemple le nom de la classe est **MONTANTTTC**. Il est impératif que les classes Web services soient de type publiques. Ceci permettrait à d'autres utilisateurs d'accéder aux méthodes de notre service.

4- Par la suite il suffit de créer les Web Méthodes de notre classe à savoir **MTTC** dans notre cas. Il faut savoir que les Web méthodes peuvent contenir plusieurs fonctions de caractère privé ou public. Donc l'attribut « **WebMethod** » permet d'indiquer que ces méthodes seront accessibles via le WebService.

Il n'est pas automatiquement possible de communiquer avec les méthodes d'une classe implémentant un Web service sur le Web mais, grâce aux Web services créés à l'aide d'ASP.NET, il est très facile d'ajouter cette capacité. Pour ajouter cette fonctionnalité, il suffit d'appliquer l'attribut WebMethod aux méthodes publiques. Les méthodes d'un Web service avec lesquelles il est possible de communiquer sur le Web sont appelées des méthodes de Web service.

Ces méthodes constituent une partie importante de l'infrastructure de messagerie utilisée par les Web services. C'est-à-dire qu'un client et un Web service communiquent à l'aide de messages, en particulier des messages SOAP, par défaut. Les clients envoient une demande SOAP à un Web service existant, et une des méthodes du Web service retourne une réponse SOAP. Les Web services définissent le type des messages qu'ils acceptent à l'aide d'opérations, comme cela est défini par le langage WSDL. Celles-ci correspondent à chacune des méthodes dans un Web service. Même si chacune de ces méthodes est définie dans ASP.NET à l'aide d'une méthode d'une classe, il faut comprendre que les données qui sont éventuellement transmises sur le réseau doivent être sérialisées en XML. Ainsi, il est important de ne pas oublier que les Web services ne remplacent pas DCOM mais sont plutôt une infrastructure de messagerie pour communiquer sur des plates-formes utilisant les normes industrielles.

Après avoir choisi le langage de programmation, la création de service est assez simple, c'est de la programmation orientée objet. Prenons pour exemple notre premier Web service qui calcul le montant Toutes Taxes Comprises (TTC) **MontantTTC.asmx** (en .Net les Web services ont l'extension .ASMX en J2EE c'est .JSX) d'un montant quelconque, contient une seule méthode nommée «**MTTC**» de la classe « **MONTANTTTC** » avec comme valeurs d'entrées «Montant et TVA» et comme valeur de sortie le montant TTC.

La figure qui va suivre illustre cet exemple en VB et en C#

```
<%@ WebService language="C#" class="MONTANTTTC" %>

using System;
using System.Web.Services;
using System.Xml.Serialization;

public class MONTANTTTC {

    [WebMethod]
    public double MTTC (double Montant, double TVA) {
        return Montant*(100+TVA)/100;
    }
}
```

```
<%@ WebService language="VB" class="MONTANTTC" %>

Imports System
Imports System.Web.Services
Imports System.Xml.Serialization

Public Class MONTANTTC

    <WebMethod> Public Function MTTC(Montant As double, TVA As double) As double
        Return Montant*(100+TVA)/100
    End Function

End Class
```

Figure VII - 1- MONTANTTTC Web service en C# et en VB.NET.

Un second exemple plus au moins complet qui illustre les différentes déclarations citées ci-dessus est un Web service qui effectue la conversion entre monnaies (Dinars Dollar et Euros), dans cet exemple il existe une classe déclarant le Web Service, une Web méthode avec deux fonctions l'une publique et l'autre privée:

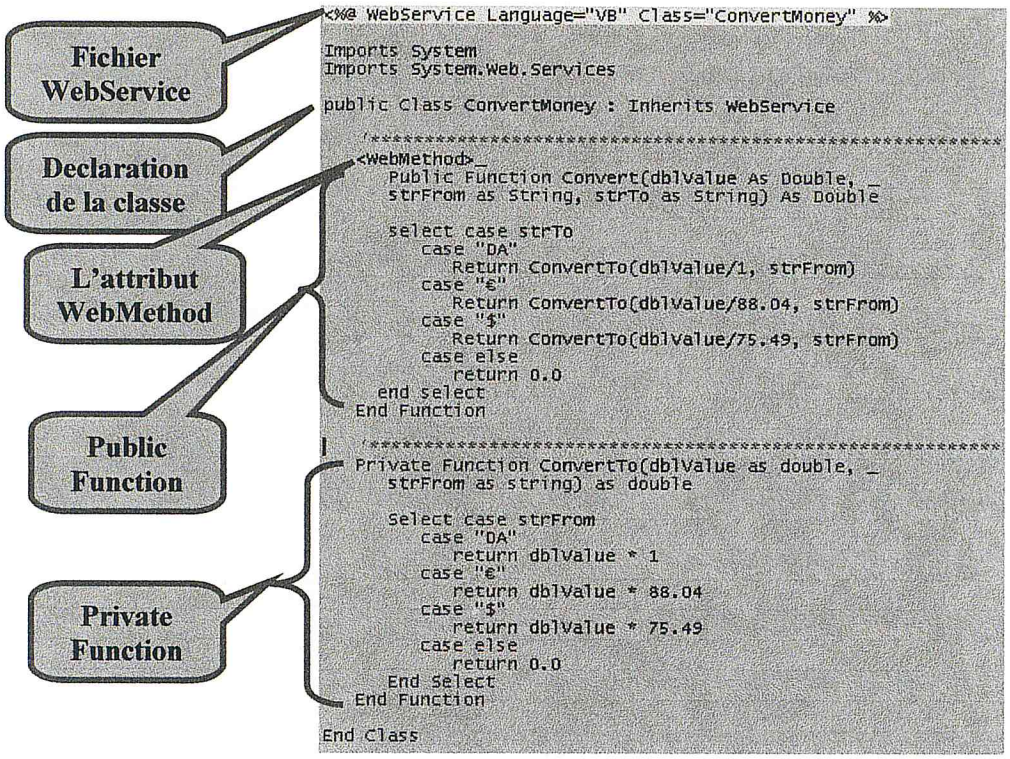


Figure VII- 2 - Web Service d'échange monétaire (dinar Euro et Dollar) en VB.NET

VII-2-Test et consommation d'un Web Service en .Net

Après un survol plus ou moins détaillé de la technologie .NET et de ASP.NET WebMatrix, et des étapes nécessaires à la création de Web services à travers la plateforme .Net. Voyons maintenant comment tester et par la suite consommer des Web services à l'aide de tous ces outils. Tout d'abord voici le code d'un Web service écrit en C# :

```
<%@ WebService Language="C#" Class="MathService" %>

using System;
using System.Web.Services;

public class MathService : WebService {

    [WebMethod]
    public float Addition(float a, float b)
    {
        return a + b;
    }

    [WebMethod]
    public float Soustraction(float a, float b)
    {
        return a - b;
    }

    [WebMethod]
    public float Multiplication(float a, float b)
    {
        return a * b;
    }

    [WebMethod]
    public float Division(float a, float b)
    {
        if (b==0) return -1;
        return a / b;
    }

}
```

Figure VII - 3 - Code C# d'un Web Service simple de calculatrice

```
<%@ WebService Language="VB" Class="ConvertMoney" %>

Imports System
Imports System.Web.Services

public Class ConvertMoney : Inherits WebService

    '*****************************************************
    <WebMethod>_
      Public Function Convert(dblValue As Double, _
      strFrom as String, strTo as String) As Double

      select case strTo
         case "DA"
            Return ConvertTo(dblValue/1, strFrom)
         case "€"
            Return ConvertTo(dblValue/88.04, strFrom)
         case "$"
            Return ConvertTo(dblValue/75.49, strFrom)
         case else
            return 0.0
      end select
    End Function

    '*****************************************************
    Private Function ConvertTo(dblValue as double, _
      strFrom as string) as double

      Select case strFrom
         case "DA"
            return dblValue * 1
         case "€"
            return dblValue * 88.04
         case "$"
            return dblValue * 75.49
         case else
            return 0.0
      End Select
    End Function

End Class
```

Figure VII - 4 - Web services de calcul d'échange monétaire (Dinar Euro et Dollar)

Une fois le code du Web service défini et après avoir vérifié la syntaxe il reste à tester ce dernier sur le Framework, une page asmx est générée ce qui permet de tester toutes les méthodes existantes sur notre service. Il suffit d'appuyer sur la touche **F5** ou bien sous le menu **View** en sélectionnant **Start**. Ainsi le serveur intégré **Cassini** de WebMatrix peut être lancé. Alors l'application sera compilée puis affichée sur Internet Explorer. En cas d'erreur sur les programmes un message d'erreur de compilation apparaît pour signaler l'erreur.

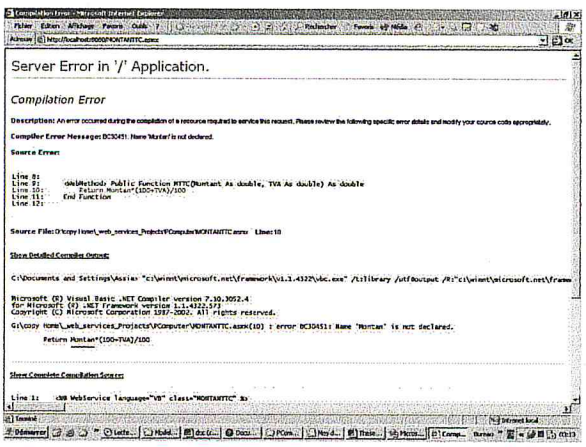


Figure VII - 5 - Erreur de compilation sur un Web service.

En effet, le Framework génère automatiquement une interface Web de test pour le Web service. En l'absence d'erreur la page suivante sera obtenue :

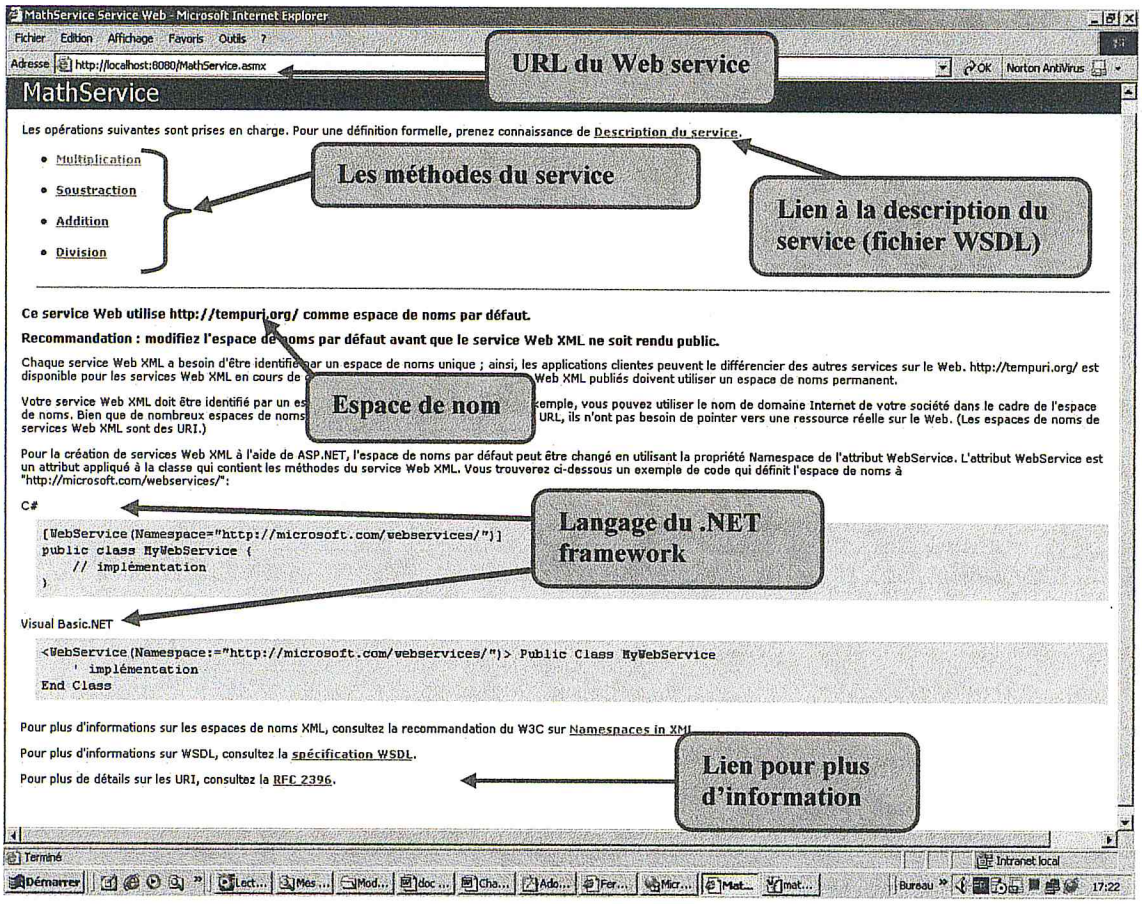


Figure VII - 6 - Test du Web Service dans le .Net Framework.

On peut noter que la page utilise un espace de nom (namespace) par défaut « http://tempuri.org/ » qui lors du déploiement sera changée à l'Url du Web service(Figure VII-6). On peut aussi noter les différentes méthodes de notre Web service à savoir « *addition* », « *soustraction* », « *multiplication* » et « *division* ». Il est possible d'accéder à la description du service (fichier WSDL) en ajoutant « **?WSDL** » à l'URL du service dans la barre d'adresse d'Internet Explorer(Figure VII-7), ou en suivant le lien de la première ligne de la page de test. Ce fichier a été détaillé dans le chapitre décrivant la norme WSDL. (cf.IV-4-)

Figure VII - 7 - Description du Web Service « fichier WSDL »

En choisissant l'une des méthodes on obtient la page de description de celle-ci;ou les Textbox des paramètres de la méthode sont présents avec les protocoles associés «Soap et Http Post»

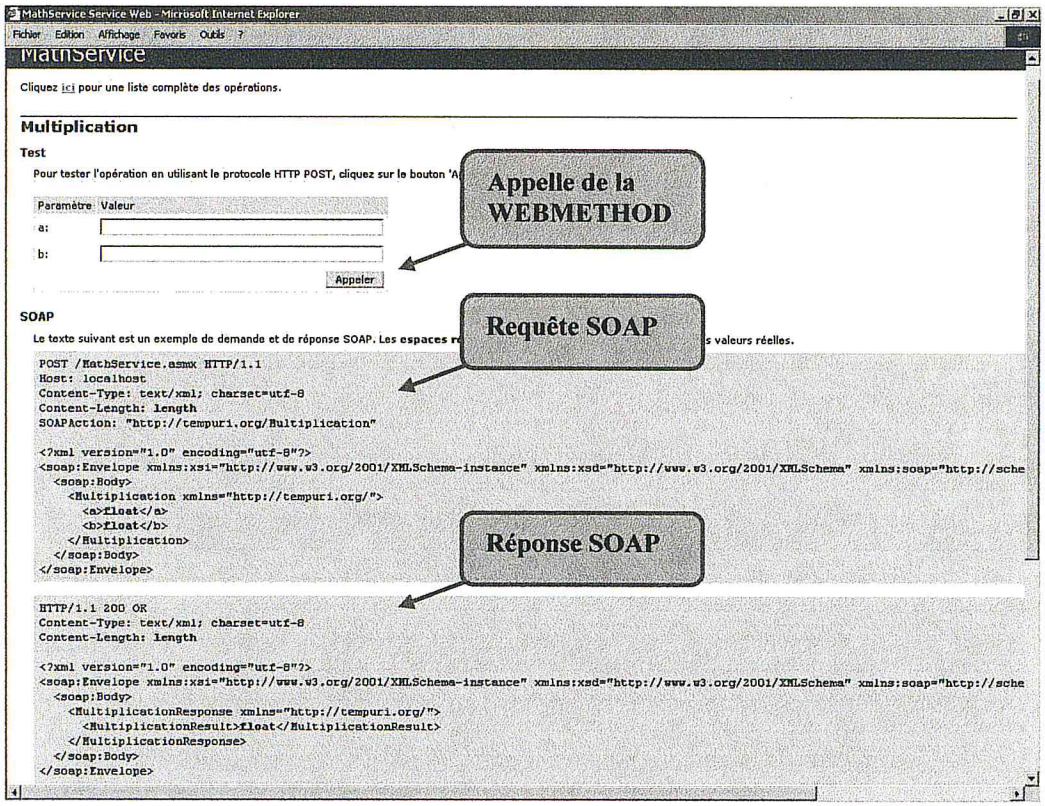


Figure VII - 8 - Protocole SOAP de la méthode Multiplication.

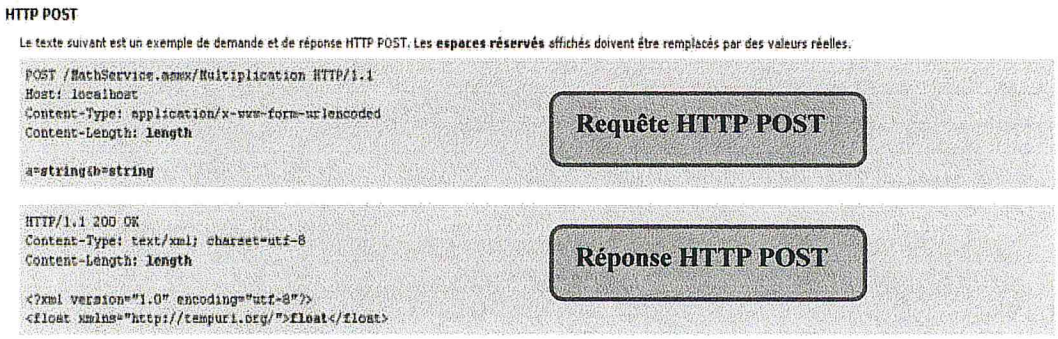


Figure VII - 9 - Protocole HTTP POST de la méthode Multiplication.

Pour invoquer la méthode, il suffit alors de rentrer les données des paramètres « *a* et *b* » et appeler la méthode via le bouton « *appeler* » ; par exemple « *a=5,5* et *b=2* » ce qui donne « *16,5* » comme résultat, qui sera affiché au format XML dans une nouvelle fenêtre.

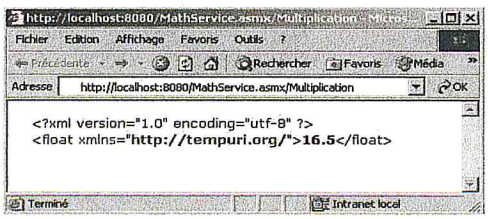


Figure VII - 10 - Résultat après interrogation de la méthode.

Le choix de l'une des méthodes sur la page de test permet d'afficher les différents protocoles permettant l'appel des Web services, chacun avec un message de requête et un message de réponse.

Les méthodes peuvent être décrites et cette description sera affichée sur la page de test et ajoutée au fichier de description WSDL du service et ceci en ajoutant à l'attribut WebMethod cette description en C# et VB.Net respectivement comme suit:

[WebMethod (Description="description de la méthode")]

Il faut aussi changer l'espace de nom avant que le service ne soit rendu public et ceci en C# et VB.Net respectivement comme suit:

[WebService(Namespace="http://AttabaBenSaid.org/webservices/")]

<WebService(Namespace:=" http://AttabaBenSaid.org /webservices/")>

Ceci dit notre Web service est prêt à la consommation par n'importe quelle application, c'est un service assez simple faisant la multiplication, division, addition et soustraction de deux chiffres. Une interface utilisateur ou cliente (Figure VII-11) sera maintenant développée pour illustrer la consommation de ce service.

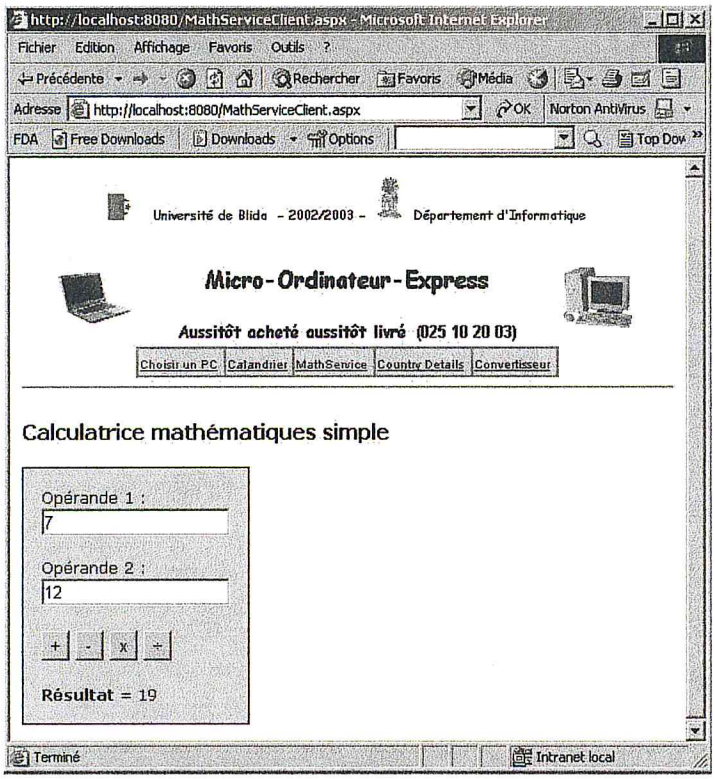


Figure VII - 11- Application Asp.net pour consommer le WebService Mathservice

Notre interface est représentée par deux zones de texte 4 boutons et un message, les valeurs peuvent être saisies dans les zones de texte et le Web service invoqué par les boutons, le résultat pourra être affiché par le contrôle message.

Pour pouvoir appeler ce Web Service à partir d'une application ASP.Net ci-dessus il faut créer ce qu'on appelle une classe Proxy (Figure VII-12).

VII-3-Le Rôle de la Classe Proxy

Comme expliqué précédemment, cette classe est une classe qui sert comme un intermédiaire entre la page Web ASP.NET et le Web Service. (Le diagramme prochain devrait aider à clarifier ceci). La structure .NET contient un programme console appelé "**WSDL.EXE**" (Web Service Description Langue) qui crée une classe Proxy pour un Web service spécifique.

Créer une classe Proxy implique trois étapes:

1. Créez le code source pour la classe qui dépend du WSDL du Web service.
2. Compilez la classe dans un DLL
3. Copiez le DLL au répertoire **\bin**

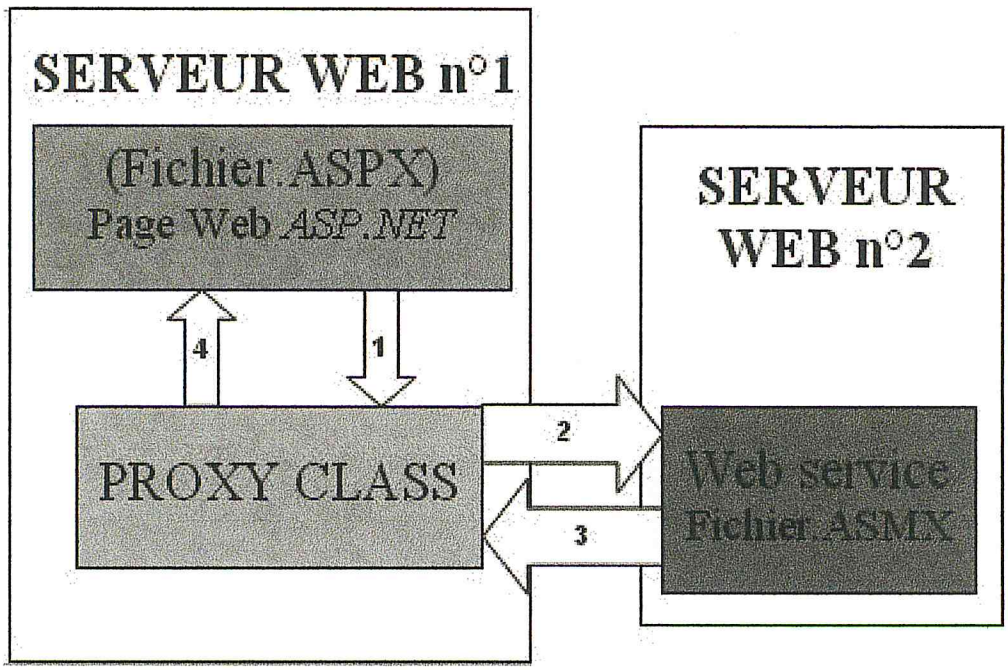


Figure VII - 12 - schémas du rôle de la classe Proxy

Une fois ces trois étapes sont complétées nous pouvons utiliser notre classe Proxy pour accéder aux méthodes du Web service comme si elles étaient des méthodes d'un composant local.

Les étapes nécessaires à la création de cette classe sont décrites (Figure VII-12), et dont l'application est comme suit:

- ***Etape 1:*** La page Web ASP.NET instancie la *Page.aspx* de la classe Proxy :
  - *Dim objClass = New ProxyClassName*
  - Et appelle une des méthodes du Web service:
    - *objClass.MethodName(paramList)*

- ***Etape 2:*** La classe Proxy marshals la liste des paramètres et fait une requête HTTP au Web service qui se trouve sur le Serveur Web n°2

- ***Etape 3:*** Le Web service unmarshals les paramètres d'entrée, utilise la méthode, et marshals les paramètres de réponse. Ceux-ci sont renvoyés dans une réponse HTTP.

- ***Etape 4:*** La classe Proxy unmarshals les paramètres de retour et laisse passer le résultat à la page Web ASP.NET. La transaction est complétée!

VII-3-1-Génération de la classe Proxy

Visual Studio.NET aussi bien que WebMatrix peuvent exécuter toutes ces tâches avec un clic de bouton.

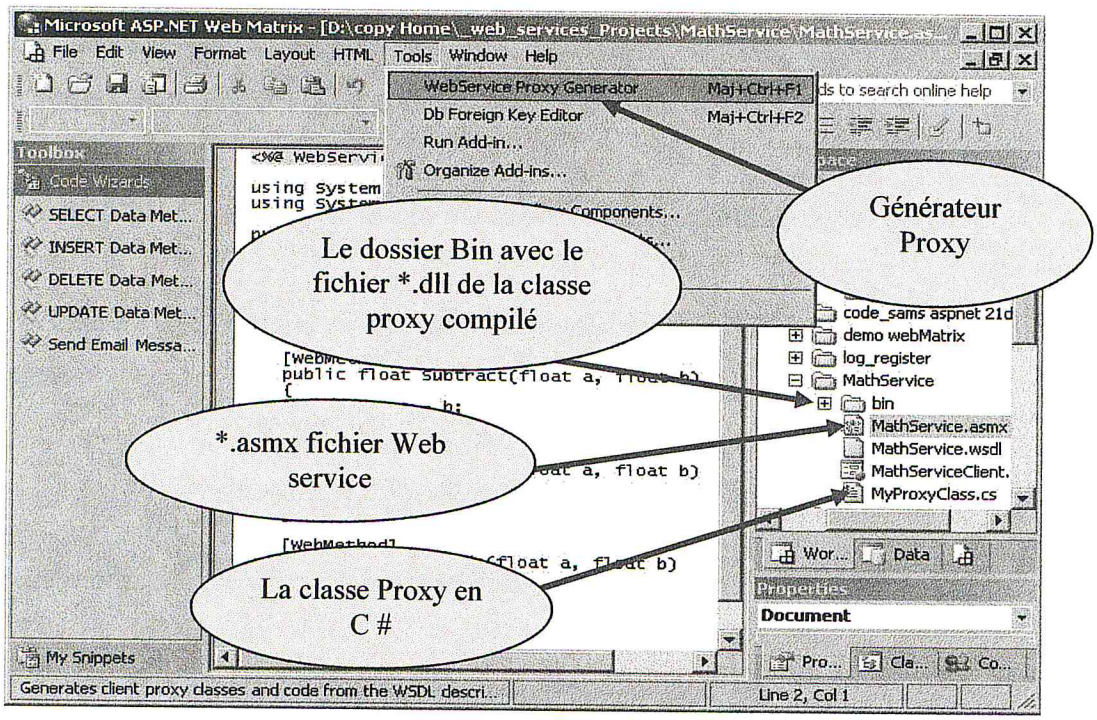


Figure VII - 13 - Génération de la classe Proxy avec WebMatrix.

Les Figures VII-13 et VII-14 montrent quelques détails pour la création de cette classe avec Web Matrix ; sous le menu «**Tools**» il suffit de choisir **Web Service Proxy Generator** pour obtenir la fenêtre de la Figure VII-14 où l'on peut spécifier l'URL du fichier WSDL, les détails de la classe à savoir le **namespace** et le **language** avec le nom de fichier, aussi bien que le fichier où réside le WSDL et le nom du fichier Binaire DLL qui sera créé dans un sous répertoire BIN où réside le fichier WSDL.

VII-4-Application ASP.NET consommant les Web Services

Comme précédemment expliqué, pour créer des Web Services et des applications Asp.net on a besoin de télécharger *ASP.NET WebMatrix Project*. Après avoir effectué les configurations nécessaires il est possible de commencer la création de Web services et de les consommer par des applications ASP.NET.

Notre application est un Web site de **e-commerce** (commerce électronique) assez simple qui permet à des clients de choisir des micro-ordinateurs suivant certaines caractéristiques, les ajouter au panier, vérifier et valider leur commande, et fournir l'adresse de livraison puis confirmer cette commande pour le paiement à domicile. De plus, le site est accompagné par d'autre services tels que : le calcul des conversions monétaires (Dinars Euros et Dollars) et celui du calcul mathématique d'addition, soustraction, multiplication et division.

Notre site se compose de plusieurs pages dont une partie (ou une base de données ainsi que des contrôles ASP.Net sont nécessaire) est considérée comme étant l'application principale, et qui à travers des liens permet d'accéder à la seconde partie qui n'est autre que des pages séparées, la partie principale est constituée des pages suivantes :

- Page d'accueil ou de présentation des produits ou le panier utilise l'un de nos Web Service pour le calcul du montant TTC WebService-1

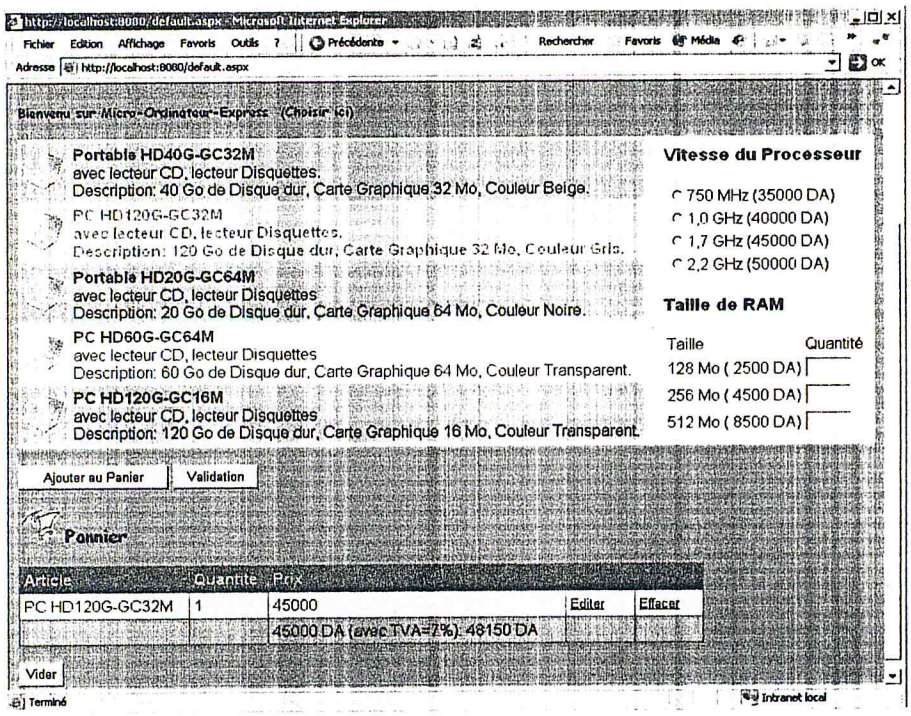


Figure VII - 17 -page de présentation.

- Page de récapitulatif des achats avec détails du client

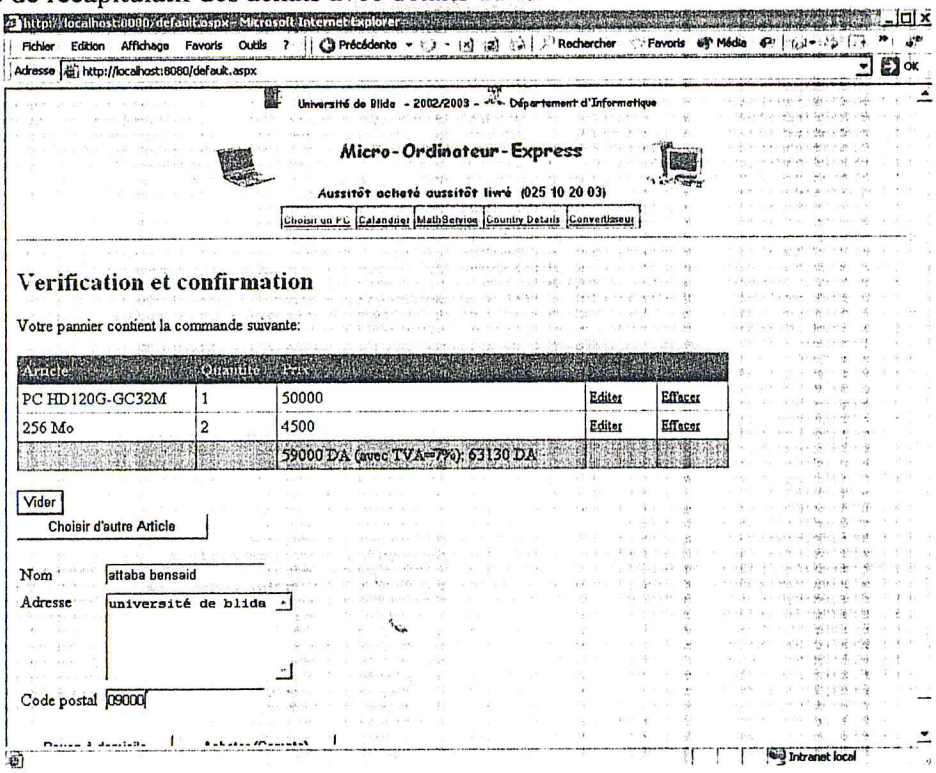


Figure VII - 18 -vérification et de Validation des commandes.

- Page de confirmation des commandes.

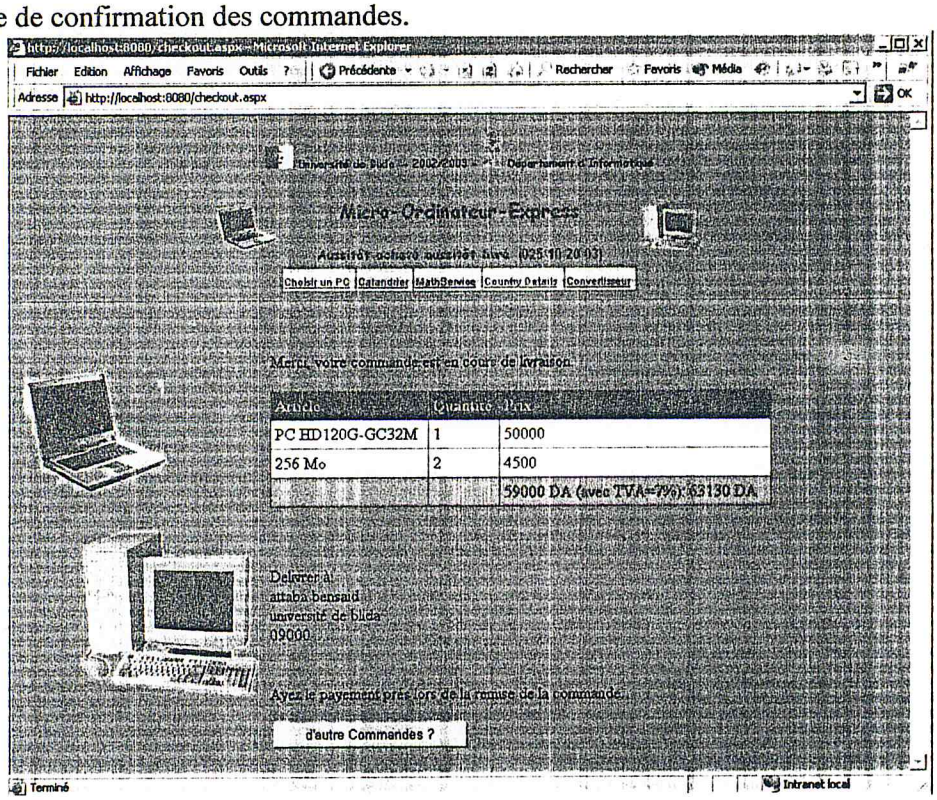


Figure VII - 19 –page de confirmation des commandes.

Et il existe des liens à d'autres pages constituant la seconde partie :

- Page de conversion monétaire WebService-2 et la Page du Web service d'une calculatrice mathématique simple WebService-3

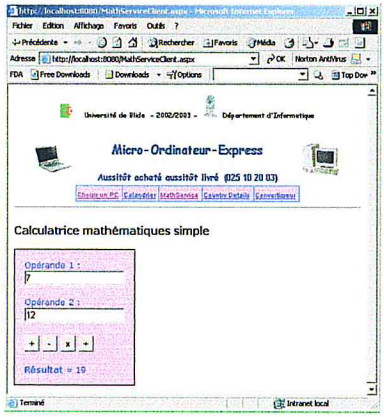


Figure VII-20–WebService calculatrice mathématique.

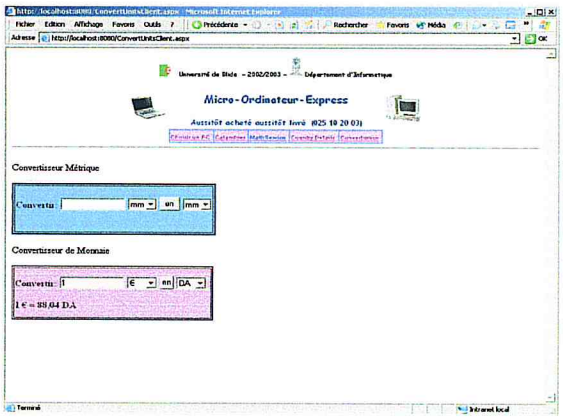


Figure VII-21-WebService Conversions monétaires.

- Page de calendrier
- Page de détails des pays du monde (monnaie, code, nom du pays)

VII-4-1-Conception du site

Pour la création de ce site Web on est donc passé par les étapes suivantes :

1- création de la couche de données DATALAYER en C# ou VB.Net à partir d'un fichier XML et de procédures SQL.

2- Création de deux Contrôles utilisateurs, un de type **Barre de Navigation** qui apparaîtra sur toutes les pages du site Web avec différents hyperliens, ainsi que la création du **Panier** qui accueillera les commandes.

3- Création de la **Page de Présentation** pour la sélection des micro-ordinateurs où différents Contrôles ASP.NET seront utilisés.

4- Création de la **Page de Vérification et Validation des Commandes** où les détails du client peuvent être entrés pour la livraison des commandes et le paiement.

5- Création de la page où l'administrateur du site peut consulter les commandes des clients.

6- Eventuellement des pages de connexion sécurisées (Login-Logout-Inscription)

Le client accède à la première page fait ses achats les ajoute au panier (bouton « **ajouter au panier** »), les valide avec bouton « **Valider** » et sera redirigé par la suite vers la pages de confirmation des commandes où il donnera ses détails. Lorsque le client effectue ses commandes il peut accéder à d'autres services disponibles à travers des liens à d'autre pages telle que celles d'autres Web services. Le plan du site se résume par le diagramme suivant :

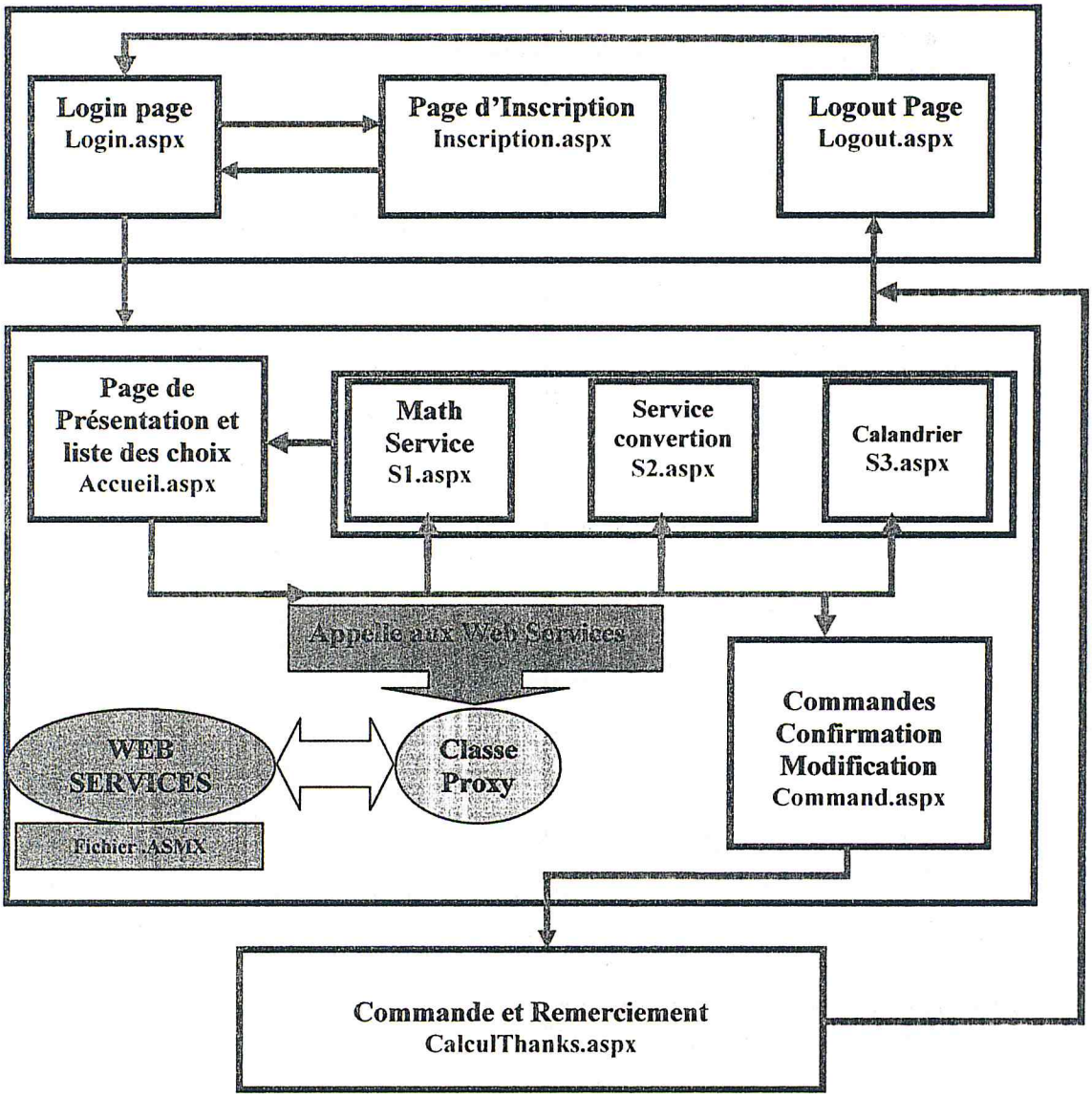


Figure VII - 22 - Plan de l'application ASP.NET et appel des Web Services.

VII-4-2- Gestion des données

A travers cet exemple on a pu utiliser les deux cas possible de gestion des données de ADO.NET, l'un par l'utilisation de fichier XML et l'autre par l'utilisation de base de données SQL. La première nous a permis de mettre en valeur le standard XML dans l'affichage des données en .Net utilisant un DataSet sous forme d'une couche de données Datalayer, par contre la base SQL nous a été utile pour le stockage des données des clients à savoir commandes et détails des clients aussi bien que leur noms d'utilisateurs et mots de passe.

VII-4-2-1-Création de la couche de données DataLayer

La création de la couche de données est faite par l'intermédiaire de deux fichiers : un fichier XML qui contient les données (figure VII-23), et d'une Classe qui charge ces données et effectue les connexions nécessaires à la base de données.

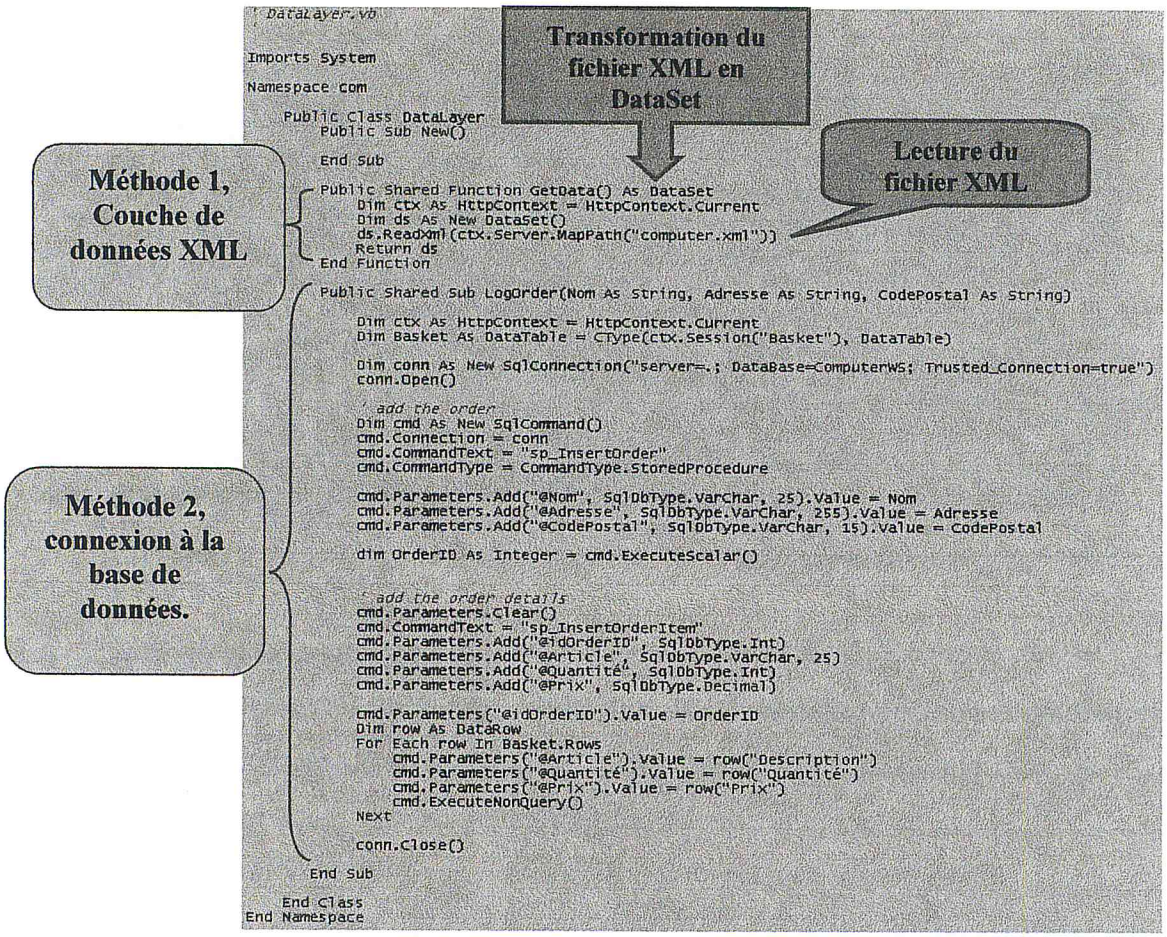
Pour utiliser les données à partir de ce fichier XML on est donc passé par la création d'une classe nommée ***DataLayer*** avec comme espace de nom ***com*** en langage VB.NET, il est possible de créer cette classe par l'insertion du code de **INSERT Data Method** généré automatiquement et modifier le code suivant notre cas de figure.

Ainsi dans cette classe on utilise deux méthodes assez simples. La première méthode charge simplement et retourne le fichier XML comme un DataSet. La seconde accède à la base de données et ajoute les détails de la commande.

Après la création de cette classe « **DataLayer.vb** » en VB.Net par exemple dans notre cas, on passe à l'étape de compilation, malheureusement WebMatrix ne peut effectuer la compilation, donc elle est faite manuellement en passant par le mode console.

vbc /debug /nologo /t:library /out:bin/DataLayer.dll r:System.dll /r:System.Xml.dll /r:System.Web.dll r:System.Data.dll Datalayer.vb

Un fichier DLL « **DataLayer.dll** » est alors crée dans le répertoire **/bin**




```
' DataLayer.vb

Imports System

Namespace com

    Public Class DataLayer
        Public Sub New()

        End Sub

        Public Shared Function GetData() As DataSet
            Dim ctx As HttpContext = HttpContext.Current
            Dim ds As New Dataset()
            ds.ReadXml(ctx.server.MapPath("computer.xml"))
            Return ds
        End Function

        Public Shared Sub LogOrder(Nom As String, Adresse As String, CodePostal As String)

            Dim ctx As HttpContext = HttpContext.Current
            Dim Basket As DataTable = CType(ctx.Session("Basket"), DataTable)

            Dim conn As New SqlConnection("server=.; DataBase=ComputerWS; Trusted_Connection=true")
            conn.Open()

            ' add the order
            Dim cmd As New SqlCommand()
            cmd.Connection = conn
            cmd.CommandText = "sp_InsertOrder"
            cmd.CommandType = CommandType.StoredProcedure

            cmd.Parameters.Add("@Nom", SqlDbType.VarChar, 25).Value = Nom
            cmd.Parameters.Add("@Adresse", SqlDbType.VarChar, 255).Value = Adresse
            cmd.Parameters.Add("@CodePostal", SqlDbType.VarChar, 15).Value = CodePostal

            dim OrderID As Integer = cmd.ExecuteScalar()


            ' add the order details
            cmd.Parameters.Clear()
            cmd.CommandText = "sp_InsertOrderItem"
            cmd.Parameters.Add("@idOrderID", SqlDbType.Int)
            cmd.Parameters.Add("@Article", SqlDbType.VarChar, 25)
            cmd.Parameters.Add("@Quantité", SqlDbType.Int)
            cmd.Parameters.Add("@Prix", SqlDbType.Decimal)

            cmd.Parameters("@idOrderID").Value = OrderID
            Dim row As DataRow
            For Each row In Basket.Rows
                cmd.Parameters("@Article").Value = row("Description")
                cmd.Parameters("@Quantité").Value = row("Quantité")
                cmd.Parameters("@Prix").Value = row("Prix")
                cmd.ExecuteNonQuery()
            Next

            conn.Close()

        End Sub

    End Class
End Namespace
```

Figure VII - 23- La couche de données (DataLayer.vb) liée au fichier XML et la base de données

Apres avoir sélectionné l'option de fichier ASCX des fichiers proposés par WebMatrix, le premier contrôle sera constitué de 4 images, 2 tableaux, 5 hyperliens, et du texte.

Le deuxième contrôle utilisateur est le **Panier**, ce contrôle nécessite quelques fonctionnalités avec les bases de données, donc on ne peut utiliser les **templates** des contrôles Asp.Net. Mais par contre, on peut utiliser ceux du **Editable Data Grid** de la section **Data** de WebMatrix.

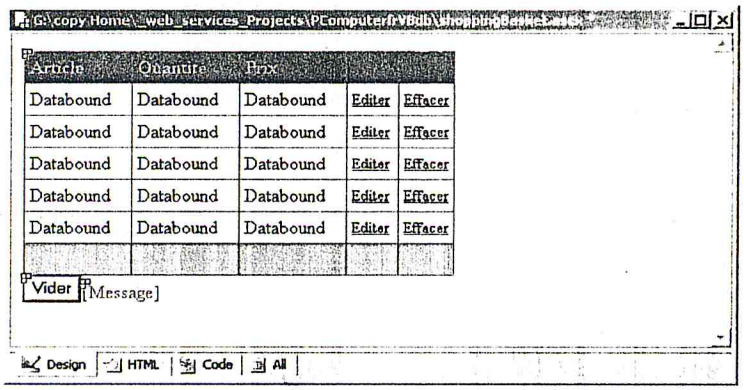


Figure VII - 29 –Contrôle DataGrid du Panier

Un code assez long sera généré mais seul quelques modifications sont nécessaires pour rendre ce code compatible avec notre base de données, les détails d'un tel travail ne sont pas nécessaires étant donné que le code est long et notre but est d'illustrer la consommation des Web services et non la création de page Asp.Net (Figure VII -30).

Une partie du code du panier fait appel au Web service MontantTTC.

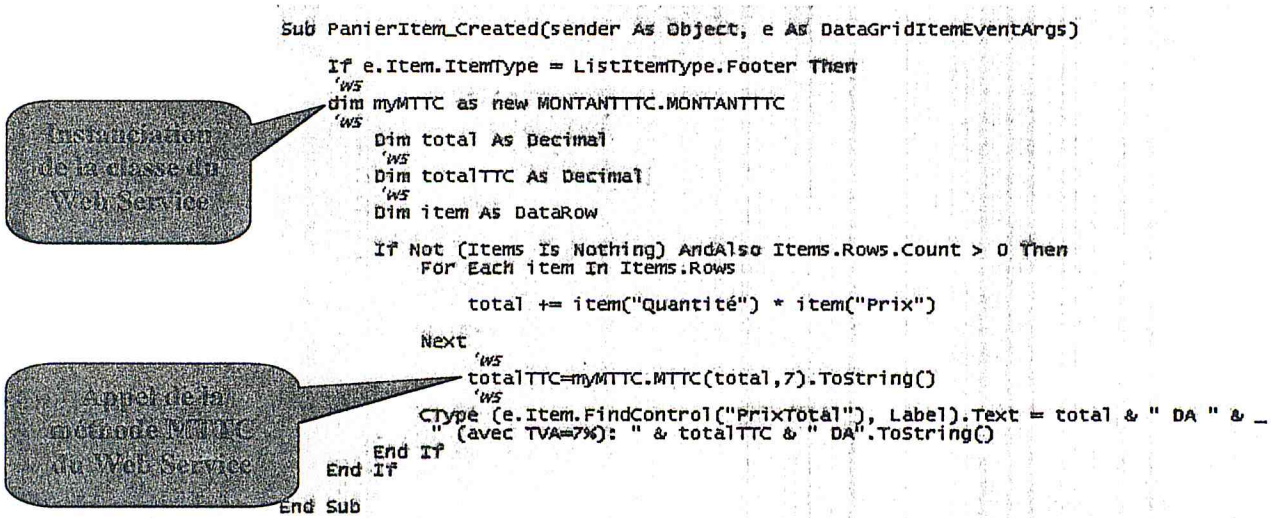


Figure VII - 30-- code de consommation du Web Service Montant TTC

Après avoir effectuer le choix approprié, et avoir valider ce choix en cliquant sur le bouton « **Ajouter au panier** » sous ce dernier, apparaît le panier affichant le prix de chaque article et la somme totale de la commande en valeur sans et avec taxes (TVA), il faut noter que les prix

en dinars sont approximatifs. Le calcul de la somme avec TVA passe par l'appel d'un Web service nommé MOTANTTTC (Figure VII-1) dont la description a précédé.

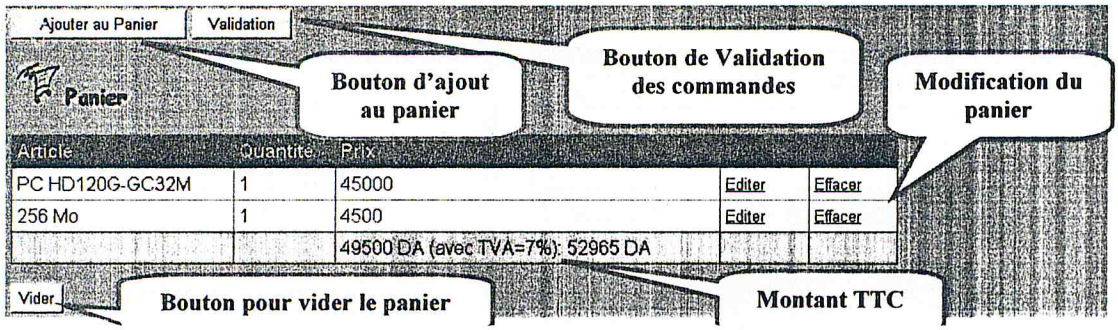


Figure VII - 31 – Eléments du Panier.

Le client a par la suite, la possibilité de modifier sa commande sur le panier (les colonne **Editer** et **Effacer**) ou de la confirmer et passer à la page de vérification des commandes, avec le bouton « **Valider** », où il peut entrer par la suite les détails d'adresse, code postal et de nom pour la livraison. Cette partie de la page de présentation est constituée de trois boutons, et du contrôle ASP.NET du panier à partir d'un **Editable Data Grid**, le code de ce contrôle nous permet de sauvegarder, modifier et mettre à jours le panier. Peu de modifications sont nécessaires pour modifier le « **Template** » fourni par WebMatrix, seuls trois événements sont modifiables sur le code : le **DataGrid_Update**, **DataGrid_Delete** et finalement **AddNew_Click**. Après confirmation des commandes en cliquant sur le bouton « **Payez à domicile** » (Figure VII -32), la dernière page fait le récapitulatif de la commande avec un panier qui ne peut être modifié (en mettant les deux dernière colonnes -**Editer** et **Effacer**- invisible) (Figure VII -19), et l'adresse en dessous de ce dernier.

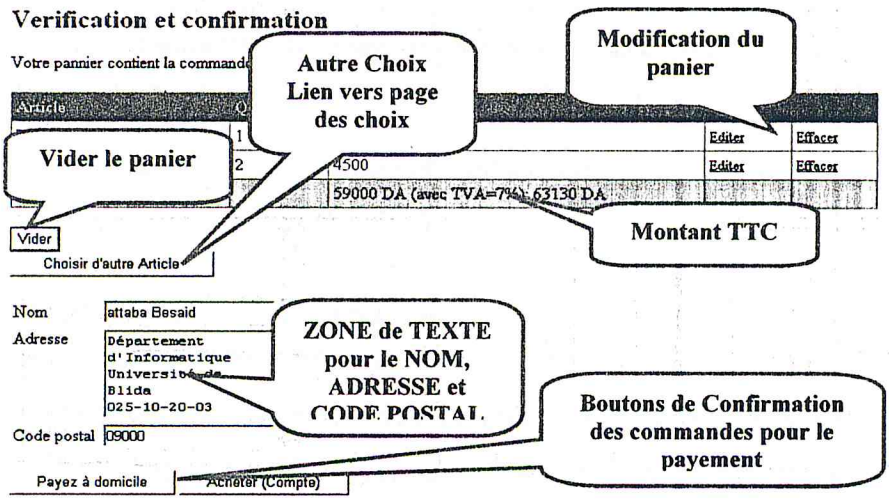


Figure VII - 32-.page de vérification et de confirmation

Sur le Web Service : On utilise, dans l'authentification avec le protocole SOAP, les en-têtes d'un message SOAP pour pouvoir stocker le nom d'utilisateur et son mot de passe. Pour créer un en-tête SOAP, il va falloir créer une classe dans le WebService qui dérive de « SoapHeader » avec deux champs publics :

- UserName
- UserPassword

Puis il faut créer une instance de cette classe dans la classe du WebService. Sans oublier qu'il faut tout d'abord déclarer le namespace « System.Web.Services.Protocols » :

Voici à présent le code de la classe « AHeader » qui va représenter l'en-tête de nos messages :

```
public class AHeader : SoapHeader
public string UserName;
public string UserPassword;
}
```

Nous allons par la suite créer, dans la classe représentant le WebService, une instance de la classe « AHeader » :

```
public AHeader SHeader;
```

Il faut indiquer que le message correspondant à la méthode qui contient un en-tête pour chaque WebMethod du WebService,. Cela va être fait à partir de l'attribut « SoapHeader ».

```
[SoapHeader("SHeader")]
```

Sur la page de test du WebService en cliquant sur la méthode, il est possible de vérifier si l'en-tête de la méthode contient bien les deux champs « UserName » et « UserPassword ».

Sur l'Application Cliente : Il faut à présent indiquer à l'application cliente de créer un en-tête SOAP du fait que l'appel de la méthode cause une erreur.

Pour ajouter un en-tête, nous allons créer une instance de la classe « SoapHeader » qui est déclarée dans la classe Proxy du WebService. Il faut ensuite entrer les valeurs de « UserName » et « UserPassword ».

```
//Création d'une instance de la classe CalcService
localhost.CalcService myWebService = new localhost.CalcService();
//Création de l'en-tête
localhost.AHeader myHeader = new localhost.AHeader();
myHeader.UserName = "UserNameTest";
myHeader.UserPassword = "UserPasswordTest";
//On remplit la propriété AHeaderValue
//par le header(en-tête) que nous venons de créer
myWebService.AHeaderValue = myHeader;
//Appel des méthodes du WebService
```

Le code ci-dessus va permettre de créer un en-tête en attribuant la valeur « UserNameTest » à « UserName » et « UserPasswordTest » à « UserPassword ».

Méthode « Add » : le code de la WebMethod « Add » du WebService « MathService », WebService qui a été réalisé dans les sections précédentes est modifié comme suit:

```
public class MathService : System.Web.Services.WebService
{
        public AHeader SHeader;
        [WebMethod]
        [SoapHeader("SHeader")]
        public float Add(int valeur1,int valeur2)
        {
                if(SHeader.UserName == "UserNameTest" && SHeader.UserPassword
=="UserPasswordTest")
                {
                        return a + b;
                }
                else
                {
                        return 0;
                }
        }
}
```

Dans cet exemple, la méthode « Add » va retourner la somme des deux arguments si le champ de l'en-tête « UserName » est égal à « UserNameTest » et si le champ « UserPassword » est égale à « UserPasswordTest ». Dans le cas contraire, la méthode « Add » retourne la valeur « 0 ».

Dans ce cas, le nom de l'utilisateur et son mot de passe sont stockés dans le code mais dans un cas réel, il y faut créer une méthode qui permettrait de vérifier les informations de l'en-tête à partir d'une base de données.Seul le protocole SOAP permettra d'accéder au WebService Lors de la création des authentifications personnalisées avec des en-têtes, les autres protocoles d'accès ne sont plus disponibles.

VII-6- Conclusion

Dans ce chapitre nous avons vu en premier lieu comment créer des Web services en .NET par l'utilisation de la directive @WebService et l'attribut WebMethod et bien sûr avec de la programmation orientée objet en C# et VB.Net. Par la suite nous avons exploré comment tester ces même Web services sur le .net Framework, les méthodes mises en jeux ainsi que les protocoles associés.

Pour pouvoir consommer ces Web services à travers Internet par une application cliente, il fallait passer par la création d'une classe, appelée classe Proxy, pour chaque Web service qui jouera lors des transactions le rôle d'intermédiaire entre l'application et le Web service. Avec

Conclusions

Conclusions Générale

Les Web Services sont la nouvelle génération de technologies d'applications distribuées qui concours vers la standardisation. Elle permet l'indépendance de plateformes, de langages de programmation, ainsi que tout logiciel ou matériel. Elle bénéficie des systèmes distribués et de l'infrastructure du Web.

La partie étude a permis de découvrir le socle de fonctionnement des Web Services, qui repose essentiellement sur les standards : XML, SOAP et WSDL. Ces derniers permettent de créer des Web Services et de les décrire, d'établir une communication, et de représenter les données échangées entres Web Services.

L'étude des plateformes a permis de trouver une implémentation plus évoluée de Web Services. La création des Web Services nécessite la classe Proxy pour le client; cette phase d'implémentation demande un travail de développement très laborieux, donc la plateforme .NET choisie a facilité cette tâche.

L'architecture client serveur a permis de réaliser un scénario de communication entre serveurs de Web Services et applications clientes au sein de la plateforme .NET. Nous avons étudié le Framework .NET et ses différents composants, ASP.NET a permis de créer des pages dynamiques. Avec l'outil WebMatrix qui représente l'interface graphique du Framework .NET nous avons créé, testé et consommée des Web Services de façon assez simple. En .NET un Web Service est un fichier portant l'extension « ASMX » désigné par la directive « @ WebService ». Créer un Web Service revient à créer une classe qui dérive de la classe « WebService » et de l'espace de nom « System.Web.Services ».

Le Web Services peut contenir plusieurs méthodes publiques à l'aide de l'attribut « WebMethod ». Ce dernier est défini pour chacune des méthodes. Il permet de rendre les méthodes accessibles sur le Web, donc la communication avec celles-ci est rendue possible.

Ainsi ces méthodes représentent l'infrastructure principale de messagerie pour les Web Services, d'où l'utilité de messages SOAP permettant d'accéder aux méthodes et d'établir un dialogue entre clients et Web Services. Ces derniers précisent le type de messages acceptés à l'aide d'opérations définies dans le document WSDL correspondant. Pour permettre l'accès au Web Service par n'importe qu'elle application écrite en n'importe quelle langage supporté par la plateforme .NET, la création de la Classe PROXY est une étape importante. Cette dernière constitue la passerelle entre les applications clientes et le Web service.

Un site de e-commerce de vente de matériels informatiques, a été créé. C'est une application ASP.NET, pour consommer et illustrer les échanges Application/Web Services. Tout est intégré dans la plateforme, c'est le plus grand avantage tiré par celle-ci. Nous avons effectué un échange local avec des Web Services et des applications d'un serveur local CASSINI intégré à WebMatrix.

Pour les travaux futurs, reste à déployer les Web Services sur Internet pour être consommés par des applications hétérogènes, appartenant à d'autres technologies. Par exemple créer une application cliente dans la plateforme J2EE pour consommer des Web Services de la plateforme .NET ou l'inverse. Ainsi, le référencement des Web Services dans l'annuaire UDDI pour être retrouvés facilement sur Internet et essayer de les invoquer à partir d'une application ASP.NET ou autre application qui soit hétérogène. Ce qui va illustrer de façon plus claire l'utilité des Web Services dans le domaine des applications distribuées hétérogènes.

La sécurité des Web Services est très importante lors de leur déploiement sur le Web chaque compagnie ou entreprise qui veut rendre publique ses services doit assurer une meilleure sécurité, car l'intérêt majeur de chaque société est de bénéficier des services qu'elle offre donc les protéger contre les intrusions extérieures, ou bien elle ne veut pas que ses Web Services soient accessibles à tout le monde. La plateforme NET propose toute une méthode de programmation pour réaliser la sécurité avec son outil Visual Studio .NET.

Références

1. *Introduction au World Wide Web par Brnard MARTIN "Technique de l'ingénieur traité Informatique H 2 908 ".*
2. *Dictionnaire encyclopédique bilingue de la micro-informatique Microsoft press.*
3. *Understanding Web ServicesArthur Ryman Senior Technical Staff Member and Solution Architect, IBM XML and Web Service.Development Environment, IBM Toronto Lab December 2000.The Dawn of the Service Web.*
4. *Services Web avec SOAP, WSDL, UDDI, ebXM Jean-Marie Chauvet, Eyrolles 2002.*
5. *Description du langage WSDL Carlos C Tapang Infotects MSDN Library* http// :.www.Microsoft.com.
6. *XML pourr l'entreprise Didier Girard ; Tanguy Crusson infos@improve.fr Version provisoire 0.91* http://www.application-servers.com/livresblancs/xml/
7. *L'architecture .NET comparée à J2EE ;.Au coeur des WebServices .NET et J2EE ;Auteur: Sami JABER* http://www Nicolas Fonrose/Guillaume Renaud.htm
8. *www.corbucci.com\essi\alain.htm*
9. webmaster@capescience.com *hit.The Problem with J2EE Amrai O'Toole, CEO, Cape Clear Software.*
10. C. L. / www.dotNET-fr.org , Le *Projet Saturn, alias ASP .NET WebMatrix*, http://www.dotnet-fr.org/sections.php3?op=viewarticle&artid=27 17juin 2002.
11. ADDA Arnaud, *Asp.Net et WebMatrix*, http://www.supinfo-projects.com/2003/asp.net%5Fwebmatrix/ 24/04/2003.
12. MSDN LIBRARY, *ASP.NET Web Matrix (Reloaded!)* http://www.microsoft.com/france/msdn/technologies/technos/asp/info/info.asp?mar=/france/msdn/technologies/technos/asp/info/20021706_aspnet-webmatrix.html#aperçu 16 juin 2003.
13. Austermühl F, *Access97 Toutes les réponses à vos question ! 100% Efficace*, MicroApplication, Düsseldorf : 1997.
14. Jaber S. http://www.dotnetguru.org/articles/DotNetFree/SharpDevWebMatrix/DotNetFree.htm , juin 2002.
15. Petillon T, *Les Cahiers du Programmeur ASP.NET*, Eyrolles, 2003.
16. http://fr.gotdotnet.com/quickstart/aspplus/
17. http://www.microsoft.com/france/asp.net/default.asp
18. Bobier J-F, *Microsoft .NET : Architecture et Services*, mémoire : Option Ingénierie Du Logiciel, École Nationale Supérieure des Télécommunications : Paris : 2001.

Bibliographies

Les livres

Auteurs, *titre du livre*, édition, lieu édition : éditeur, année.

1 Bouzeghoub M, Gardarin G et Valduriez P, *Les Objets*, Paris : edition revue et augmentée, Eyrolles, 2000.

2 Thau D, *Java Script par la pratique*, 1ère Ed, Paris : Eyrolles, 2001.

3 Edwards J, *Client/Serveur à 3 niveaux en pratique*, 1ère Ed, Paris : Vuibert Informatique, 2000.

4 Courtin J, Kowarski I et Arsac J, *Initiation à l'algorithmique et aux Structure de données – « 1- Programmation structurée et structures de données élémentaires »*, 1ère Ed, Paris : Dunod Informatique, 1989.

5 Courtin J et Kowarski I, *Initiation à l'algorithmique et aux Structure de données – « 2- récursivité et structures de données avancées »*, 1ère Ed, Paris : Dunod Informatique, 1989.

6 Tanenbaum A, Henrnandez J A et Joly R, *Reseaux*, 3ème Ed, Paris : Dunod, 1999.

7 Charton E et Pavie O, *Crée un Site Internet*, 1ère Ed, Marsat : CampusPress France, 2000.

8 Young M J, *XML step by step*, 2ème Ed, USA : Microsoft Press, 2002.

9 Engst A C, Low C S et Simon M A, *c'est dans la boite ! INTERNET pour Windows*, 1ère Ed, Paris : Simon & Schuster Macmillan, 1995.

10 Guyot J et Vial C, *Arbres Tables Algorithmes*, 1ère Ed, Batna : Chihab-Eyrolles, 1994.

11 Jacoud R, *Série Schaum -Mathématiques pour Informaticiens- cours et problemes*, 1ère Ed, Paris : McGraw Hill, 1985.

12 Mirecourt A et Saumont P Y, *Java 2 Edition 2001 SDK 1.3*, 1ère Ed, Paris : Osman Eyrolles Multimedia, 2001.

13 Bouzeghoub M, Gardarin G et Valduriez P, *Objets « Concept – Langages - Base de Données - Méthodes - Interfaces »*, 1ère Ed, Paris : Eyrolles, 199.

14 Vauquier D, *Développement Orienté Objets « Principes, Processus, Procédés »*, 1ère Ed, Paris : Eyrolles, 199.

15 Delannoy C, *Programmer en Language C++*, 4èmé Ed, Paris : Eyrolles, 1998.

16 aa, *Access PC pour les* Malins, 1ère Ed, Paris : ellipes, 1996.

17 Beaud M, *L'art de la thèse*, 1ère Ed, Paris : La Decouverte, 1986.

18 Inmon W H, *le développement des application Client/Serveur*, 1ère Ed, Paris : Mason, 1993.

19 Rizzo J & Poole L, *Le Petit Livre des Réseaux*, 1ère Ed, 1ère Ed, Paris : Campus Press France, 1999.

20 *Dictionnaires encyclopédique bilingue de la micro-informatique*, Paris : Microsoft Press, 1999.

21 Hubbard J R, *Programming with Java – Schaum's outlines series*, 1ère Ed, Londres: McGraw Hill, 2001.

22 Diallo N M, *mise en place d'une architecture organisationnelle, applicative et technique pour la gestion documentaire*, these de E.F.T.G 2002-2003

23 Hughes J M, *C++ la Maîtrise par l'Exemple*, 1ère Ed, Paris : Sybex, 1990.

Annexes

Annexe A

Le concept de base du Web :

Le Web est construit autour d'une **architecture client / serveur** qui possède les éléments suivants :

-Un client Web : c'est un processus qui exécute un programme pour permettre à l'utilisateur de soumettre des requêtes à un serveur Web et d'en visualiser le résultat.

-Un serveur Web (HTTPD : Hypertext Transfer Protocol Daemon) : ou démon c'est un processus qui a pour rôle de répondre à des requêtes émises par les clients qui exécutent un browser.

-Un protocole de communication : il s'agit de HTTP entre un client et un serveur Web c'est un protocole question-réponse, ce protocole possède plusieurs méthodes parmi elles :

la méthode GET qui compose la requête du client pour obtenir un document désigné par l'URI (Uniform Resource Identifier), ou exécuter un script dont le résultat sera transmis au client,ou bien d'effectuer des recherches.

La méthode POST qui permet d'annoter un document, déposer un message dans un newsgroup (forum), ajouter un élément dans une base de donnée…etc.

-Un mécanisme d'exécution le CGI (Common Gateway Interface) : c'est le mécanisme qui permet au client de faire exécuter un programme appelé **script CGI** sur le serveur, ce sont des scripts ou bien des programmes exécutables, que l'administrateur(Webmaster) du serveur Web doit autoriser .

-les techniques de cache et de Proxy : le cache c'est pour les clients (browser), le Proxy (ou mandataire) c'est pour les serveurs,ces techniques permettent d'améliorer les performances entre le client et le serveur,car le mandataire joue le rôle d'intermédiaire entre le client et le serveur Web ;en recevant la requête du client et la satisfaire si le document est dans son cache sinon la transmettre au serveur.le Proxy agit pour l'ensemble des clients connectés au même réseau ,les clients de type « navigateur » ont l'option de spécifier leur mandataire par son adresse TCP /IP (Transmission Control Protocol/Internet Protocol)

-la notion d'hypertexte et hypermédia :

Un <u>document hypertexte</u> est un fichier texte contenant des liens vers d'autres parties du document ou d'autres documents.

Un <u>lien hypertexte</u> (hyperlien) se compose d'une ancre et l'adresse du document désigné.l'ancre est un mot ou ensemble de mots, images…et qui par un simple clic de la souris permet de charger les pages correspondantes

Un <u>document hypermédia</u> est un document hypertexte qui peut référencer d'autres documents hypertextes ou des documents contenant du son, images ou vidéo.

Un document hypermédia est créé interprété par <u>les langages standards du Web</u> il s'agit de HTML (Hhypertext Mmarkup Llanguage), XML (eXtensible Markup Langage) et SMIL (Synchronized Multimedia Integration Langage).

-la notion d'URL (Uniform Resource Locator) :c'est une référence non ambiguë contenue dans l'hyperlien, dans le réseau Internet c'est l'extension qui exprime le chemin utilisé pour désigner le fichier demandé.

-La notion de formats de données : lors d'un dialogue entre le client et le serveur, le client donne la liste des types qu'il peut traiter, l'extension du nom de fichier permet au serveur de déduire le type MIME (Multipurpose Internet Mail Extensions se) du fichier (se sont des fichiers e-mail), (.htm ou .html) pour des document html, (.doc) pour des documents Word.

Annexe B

Les standards des Web Services

Type d'utilisation	Nom du standard	Description
Identification et Localisation des Web Services	**UDDI** (Universal Description Discovery and Interoperability)	permet aux web services de s'invoquer dynamiquement et permet de rechercher les services dont les coordonnées sont inconnues.
	WS_Inspection (Web Services_**Inspection**)	agrége les pointeurs vers les web services d'une entreprise souhaite exposer, puis interroge le document web résultant de l'agrégation,à la différence d'UDDI il ne permet pas de rechercher des services dont on ne connaîtrait pas les coordonnées.
	eb_XML (electronic business_**XML**)	-spécifie les règles de conception pour le processus BtoB, il inclut ses propres spécifications d'inscription et d'annuaires. (BtoB:Business to Business, c'est le commerce électronique d'entreprise à entreprise par exemple les transactions interentreprises).
	ADS(Advertissement and Discovery of Services protocol):)	fournit un monde de description des web services, à la manière des Meta données et permet aux annuaires (comme UDDI) d'effectuer des référencements automatiques.
	WSIL (Web Services Inpoint Langage)	décrit la qualité du service (temps de réponse) associée aux composants Web Services pour faciliter leur mode de distribution.
Exposer les fonctions des Web Services	**WSDL**(WebServicesDescriptionLangage)	Le langage de description des services offerts.
	SOAP(SimpleObjectAccessProtocol)	le protocole de transport des données.
	DIME (Direct Internet Message Encapsulation)	enrichi le mode de dialogue entre composants en assurant l'encapsulation dans un message (SOAP) de divers types de données (audio, vidéo, texte).
Echanger de messages entre Web	**SWAT** (SoapWith Attachement vehicule)	sert à véhiculer les documents multimédias sous forme de fichier attachés aux messages

Services		SOAP une méthode qui allége le travail de l'analyseur lors de la réception, que la méthode d'encapsulation ne permet pas.
	XMLprotocol	Nom de la prochaine version de SOAP qui comporte les nouvelles fonctions transactionnelles
Les protocoles de transactions	**SOAP-RP (SOAP-**Routing Protocol)	il utilise toute une variété de protocoles réseau (tcp, http..) et permet de les combiner ; contrairement à SOAP qui se limite à une seule couche de transport de bout en bout.
	WS-transaction (Web Services **transaction)**	c'est un langage qui garanti l'intégrité des transactions exécutées via les web services.
	SAML	couvre les problématiques d'authentification et l'habilitation des transactions SOAP.
	BTP (Business Transaction Protocol)	un protocole qui gère l'intégrité des transactions longues et complexes.
Les protocoles de sécurité :	**XACML (XAccessControlMark**upLangage)	définit un schéma décrivant les politiques de droit d'accès à des objets XML.
	WS-Security (Web Services-**Security)**	offre les fonctions d'accès et de chiffrement des échanges.

Annexe C

Les standards XML

Recommandation

[illegible]	Date	[illegible]
XML, DTD	Février 1998	La norme définissant XML et les DTD (*Document Type Definition*)
XML Namespaces	Janvier 1999	Convention pour l'attribution de noms et pour le groupage des balises Eyrolles.
XPath v.1.0	Novembre 1999	Expressions en XML de fragments de documents XML
XSLT v1.0	Novembre 1999	Expressions en XML de transformation et d'opérations sur des documents XML Chauvet.
XHTML v1.0	Janvier 2000	Reformulation en XML de HTML v. 4.0
XML Schema	Mai 2001	Expressions en XML des structures de données
XLink v1.0	Juin 2001	Expressions en XML de liens (complexes) entre fragments de documents XML
SMIL, SVG, RDF, MathML	1999-2001	Vocabulaires « verticaux » XML pour le multimédia, le graphisme vectoriel, les structures de données et les expressions mathématiques
XSL v1.0	Octobre 2001	Feuilles de style XML
XML Information Set	Octobre 2001	Ensemble de définitions permettant à d'autres WMDL. spécifications de se référer à l'information contenue dans un document XML

Il existe plusieurs éditeurs de texte pour XML tel que :

-XML Writer.

-XMLSPY 5 Enterprise Edition.

-Microsoft XML Notepad.

-XML CooKTop.

-Peter's XML Editor

Annexe D

XML et Unicode :

UCS, pour *Universal Character Set.* Chaque caractère étant codé sur quatre octets, le codage normatif est désigné par le sigle UCS-4.Quatre octets permettent de coder 256^4=4 294 967 296 valeurs différentes

Le basic Multilingual Plane : Les 65 536 premières positions de l'UCS (0000 à FFFF) forment le *Basic Multilingual Plane* (BMP). Le codage normalisé UCS-2 permet de représenter les caractères du BMP sur deux octets .le BMP contient la définition des caractères latins, grecs, arabes, hébreux, etc,et suffit en pratique à couvrir la quasi-totalité des besoins des utilisateurs européens, américains ou japonais.

Les codages de transformation : les deux codages canoniques sont UCS-4 qui permet de représenter la totalité de l'UCS, et UCS-2 qui suffit pour coder le BMP. Trois autres codages des caractères, dits « codages de transformations », ou **UTF**, ont été définis :

UTF-16 : Etablit le pont entre UCS-2 et UCS-4, inclure certains caractères de UCS-4 dans une chaîne codée en UCS-2.Toutes les applications XML doivent supporter le codage UTF-16.Un fichier de caractères UTF-16 doit commencer par un octet de signature, à savoir le caractère, #xFEFF « zero width no-break space » qui permet au processeur XML de déterminer si le fichier qui suit est codé en UTF-8 ou en UTF-16.

UTF-8 : UTF-8 code les caractères sur une séquence d'octet de longueur variable (de 1 à 6). Sa qualité principale est que les flots de caractères ASCII, qui représentent sans doute 80 à 90% des transmissions de données sur Internet, sont codés en utilisant seulement un octet
par caractère.Les caractères ASCII (00 à 7E) se voient directement attribuer les mêmes valeurs en UTF-8.donc *UTF-8 est le codage par défaut utilisé par les applications XML.*

UTF-7 : conçu pour permettre la transmission de caractères Unicode par les serveurs de courrier électronique Internet, qui dans leur version traditionnelle ne supportent que les caractères codés sur 7 bits.

Déclarations de codage : toutes les applications doivent être capable de traiter les documents encodés en UTF-8 ou UTF-16. En pratique de nombreux utilisateurs emploient des codages de caractères spécifique à leur langue, tels que ISO-8859-1 pour le français, ISO-8859-7 pour le Grec ou EUC-JP pour le japonais. Les applications XML peuvent être capable de traiter de tels codages, en sus de UTF-8 et UTF-16.Pour signaler à l'application qu'un tel codage de caractères est utilisé, le créateur d'un document doit insérer une déclaration XML dans son document, fournissant une indication de codage :< ? xml version =1.0 encoding = ISO-8859-1 ?>

Annexe E

Exemples de Web Services

L'exemple d'une station de Météo qui veut revendre ses services. Elle veut par exemple partager une méthode avec une entreprise tierce : cette méthode permettrai de retrouver la température d'une région en entrant le Code Postal de celle-ci. Rien de plus simple avec les Web Services ! La Station de Météo devra créer un Web Service avec une WebMethod qui retournera un entier (ou un float) suivant le Code Postal (donc un entier) qui sera transmis à la WebMethod. Ensuite, si la station de Météo veut faire payer son Service, elle devra bien sûr protéger l'accès de la Méthode.

Le scénario de l'agence de voyages. L'agence propose en effet à ses clients un produit, le voyage, qui est en fait un bouquet de services : réservation et émission des billets d'avions, réservation des hôtels, des billets de train, des voitures de location, assurances, etc. Les données nécessaires à l'élaboration d'un voyage personnalisé proviennent de fournisseurs différents (compagnies aériennes, chaînes hôtelières, etc.). Filtrées, choisies, adaptées aux exigences du consommateur et aux tarifications négociées entre les différents partenaires tarifs eux-mêmes soumis à des mises à jour elles servent ensuite de base à la facturation. L'application Web consultée directement par le client de l'agence de voyages résulte de la mise en œuvre de services Web (recherche dans des catalogues et des tarifs, réservation, fidélisation...). Dans le scénario B2B (Business To Business), les mêmes messages circulent mais, cette fois, entre serveurs plutôt qu'entre client et serveur. Agrégation et transformation de l'information sont les tâches que les Web services automatisent complètement. Une application Web résulte alors de l'assemblage de services Web, certains développés en interne et d'autres fournis par des partenaires externes. Aux deux extrémités de cette échelle, on trouve le tout interne (intégration d'application d'entreprises ou EAI) et le tout externe (portail). Pour les applications Web, développer, c'est assembler et déployer, c'est publier.

Le cas simple d'une entreprise qui veut mettre en place un système de messagerie au sein de son Intranet, voir Internet par la suite. Elle pourra donc opter pour la technologie des Web Services en développant le Serveur de Messagerie en Web Service .NET. L'avantage de cette solution est que l'application cliente pourra être développée sous n'importe quel langage. L'entreprise pourra même développer plusieurs clients certains marchant sous Unix, d'autres sous Windows mais utilisant le même serveur de messagerie, c'est-à-dire le Web Service qui se charge de la réception et de la transmission des messages. Nous pourrons donc très bien avoir un client Windows qui communique avec un client Unix voir même un client Pocket PC !

Annexe F

La méthode d'attribut d'éléments en wsdl

Pour montrer la différence entre les attributs d'élément et les attributs de type en particulier lorsque un type est associé à un élément <part> ; l'attribut de type a l'avantage de permettre la description d'une partie pouvant prendre en charge plusieurs types (comme un Variant), ce qui est impossible avec un attribut d'élément. Le document suivant illustre cette possibilité :

```
<?xml version="1.0" encoding="UTF-8" ?>
<definitions … >
1)  <types>
 <schema targetNamespace="someNamespace" xmlns:typens="someNamespace">

1.1) <xsd:complexType name="PERSON">
 <xsd:sequence>
 <xsd:element name="firstName" type="xsd:string"/>
 <xsd:element name="lastName" type="xsd:string"/>
 <xsd:element name="ageInYears" type="xsd:int"/>
 <xsd:element name="weightInLbs" type="xsd:float"/>
 <xsd:element name="heightInInches" type="xsd:float"/>
 </xsd:sequence>
 </xsd:complexType>

1.2) <xsd:complexType name="femalePerson">
 <xsd:complexContent>
 <xsd:extension base="typens:PERSON" >
1.2.1)<xsd:element name="favoriteLipstick" type="xsd:string" />
 </xsd:extension>
 </xsd:complexContent>
 </xsd:complexType>

1.3)<xsd:complexType name="malePerson">
 <xsd:complexContent>
 <xsd:extension base="typens:PERSON" >
1.3.1)<xsd:element name="favoriteShavingLotion" type="xsd:string" />
 </xsd:extension>
 </xsd:complexContent>
 </xsd:complexType>

1.4)<xsd:complexType name="maleOrFemalePerson">
1.4.1)<xsd:choice>
 <xsd:element name="fArg" type="typens:femalePerson" >
 <xsd:element name="mArg" type="typens:malePerson" />
 </xsd:choice>
 </xsd:complexType>

 </schema>
</types>

2)<message name="addPerson">
2.1) <part name="person" type="typens:maleOrFemalePerson"/>
</message>

<message name="addPersonResponse">
 <part name="result" type="xsd:int"/>
</message>
```

```
</definitions>
```

Description

1.1) déclaration de type complexe « **PERSON** » et plus loin l'utilisation de la dérivation par extension de ce même type, les éléments :

1.2)"femalePerson" : type dérivé possède un élément supplémentaire 1.2.1) **"favoriteLipstick".**

1.3)"malePerson" : type dérivé possède un elément supplémentaire 1.3.1) **"favoriteShavingLotion".**

1.4)"maleOrFemalePerson"un type complexe unique pour 1.2) et 1.3) ; il contient : 1.4.1) la construction **<choice>** qui permet à 1.4) de combiner les deus types dérivés1.2) et 1.3)

2) l'elément **<message>** de "addPerson", le type combiné est référencé par l'élément 2.1) <part> de "person". Cet élément <part> ou paramètre peut ensuite être soit "femalePerson" ou "malePerson".

La référence se fait par l'attribut **type** voir 2.1),la différence marquée par une référence d'élément (cf. IV-2 ; 4.1) ; 4.2))

Annexe G
Comparaison J2EE/.NET

[illegible]	[illegible]	[illegible]
Langage C#	Langage Java	C# et Java ont des mécanismes équivalents (ramasse-miettes, héritage simple, typage sûr…) et héritent de la syntaxe de C++. Plus moderne, C# innove avec les attributs qui permettent de décrire le comportement (i.e. déploiement) du code dans le source. Avantage ou désavantage selon les écoles, C# offre une souplesse et une expressivité plus grande que Java, très puriste quant au paradigme orienté-objet.
Common Language Runtime (CLR)	Java Virtual Machine (JVM)	La JVM est un interpréteur de *bytecode*, le CLR compile le code IL systématiquement en code natif. Le CLR est conçu pour accepter un grand nombre de langages et permet grâce aux assemblages la création facile de composants partagés. Java ne peut pour l'instant le faire qu'à l'aide de CORBA, qui n'est pas autant intégré dans J2EE.
Base Class Library	Java 2 Platform API	Quoique très comparable dans les classes de base fournies, .NET se démarque de Java par son excellent support intégré de XML et de SOAP.
- Active Server Pages (ASP.NET) - WebForms	Java Server Pages (JSP)	ASP.NET et JSP sont toutes deux écrites en langage de haut niveau (Java pour JSP, les langages du CLS pour ASP.NET) et sont compilées (en *bytecode* JVM pour Java, en IL puis code natif pour ASP.NET). Les WebForms apportent une couche d'abstraction par rapport au langage cible de présentation (HTML, WML…) dont JSP ne dispose pas.
- Services Web - ADO.NET	- Enterprise JavaBeans - JDBC - (Apache-SOAP)	A l'aide de HTTP et XML, les services Web et ADO.NET permettent de mettre en ligne des services, de partager et mettre à jour des bases de données simplement et indépendamment de l'architecture cible. L'architecture J2EE est moins ouverte de par sa dépendance envers Java (RMI, messagerie Java). Un pont CORBA existe, mais n'est pas directement utilisable. Le modèle de données de JDBC est connecté à la différence d'ADO.NET. Apache-SOAP apporte les services Web à J2EE, mais n'est pas intégré à la plate-forme pour l'instant. La version 1.4 de J2EE prévoit un kit de développement pour les services Web[25].
WinForms	Java Swing	Bien que fortement orientés réseau, .NET et J2EE permettent le développement d'applications graphiques classiques. Swing et WinForms sont orientés-objets en respectant le paradigme Modèle-Vue-Contrôleur (MVC). Après longue maturation (échec de AWT), Swing fonctionne de manière presque similaire sur toutes les plates-formes cibles. WinForms est pour l'instant restreint à Windows uniquement.

Visual Studio.NET	- VisualAge for Java - Sun Forte for Java - ...	Visual Studio.NET est simple d'emploi et très puissant, son intégration fine avec IIS permet des cycles de développements et de tests rapides. Les débutants peuvent écrire des services avec peu de connaissances initiales. Les outils de développement Java sont pléthore et dépendent de l'intégration avec un serveur d'application spécifique pour le déploiement (WebSphere pour VisualAge par exemple). Même dans ce cas, les EJB demeurent des entités complexes à manipuler.

Caractéristiques particulières

[illegible]	[illegible]	[illegible]
Composants réutilisables et partagés, avec informations de déploiement	Assemblages, SOAP et DCOM – autodescription à l'aide des attributs.	Entreprise JavaBeans, RMI et CORBA – descripteurs des EJB séparés et souvent dépendants du serveur d'application.
Support des terminaux mobiles	.NET Mobile SDK - limité à Windows CE pour l'instant.	Java 2 Micro Edition (J2ME).
Développement multi-langage	Plus de 20 langages annoncés.	Langage Java uniquement – quelques tentatives de support de Python, Eiffel ou Cobol mais l'intégration est limitée.
Architectures supportées	Windows – le CLR et les soumissions à l'ECMA permettent un portage par des tiers.	Nombreuses (Windows, Solaris, AIX, Linux, Amiga...) – J2EE reste la propriété de Sun.
Disponibilité	15% maintenant, 50% dans 6 mois et le reste dans 2 ou 3 ans.	90% maintenant, le reste dans 6 mois.

Annexe H

Configuration matériel ET logiciel nécessaire recommandée

Pour offrir des performances optimales, le .NET Framework possède la configuration minimale et recommandée ci-après pour les applications clientes/serveur.

Configuration requise du système d'exploitation

Le .NET Framework est pris en charge sur les plates-formes suivantes.

Scénario	Système d'exploitation
Client	Microsoft Windows 98, 98 Deuxième Édition, Millennium, NT4.0 Workstation avec Service Pack 6.0, 2000 Professionnel, XP Édition familiale et Professionnel, Server 2003
Serveur	Microsoft Windows 2000 Professional avec Service Pack 2.0, MS Windows® 2000 Server avec Service Pack 2.0, MS Windows® 2000 Advanced Server avec Service Pack 2.0 MS Windows® 2000 Datacenter Server avec Service Pack 2.0 MS Windows® XP Professionnel Famille MS Windows® Server 2003

Remarque Sur tous ces systèmes, Microsoft Internet Explorer 5.01 ou version ultérieure et Microsoft Windows Installer 2.0 ou version ultérieure sont également requis.

Configuration logicielle supplémentaire requise

Pour utiliser des fonctionnalités supplémentaires telles que les services ASP.NET, COM+ et le fournisseur de données SQL Server .NET, vous avez besoin des logiciels supplémentaires suivants.

Scénario	Fonctionnalité	Logiciels requis
Client	Fournisseur de données SQL Server .NET	Microsoft Data Access Components (MDAC) 2.6 ou ultérieur
	Accès aux informations de gestion du système	Windows Management Instrumentation (WMI) (installé avec le système d'exploitation sous Windows 2000, Windows Millennium Edition et Windows XP)
	Services COM+	Windows 2000 Service Pack 2.0
Serveur	Fournisseur de données SQL Server .NET	Microsoft Data Access Components (MDAC) 2.7
	ASP .NET	Microsoft Internet Information Services (IIS) 5.0

Configuration matérielle requise

La configuration matérielle requise est résumée dans le tableau suivant :

Scénario	Processeur requis	Processeur recommandé	RAM requise	RAM recommandée
Client (Windows Forms et Windows Services)	Pentium 90 MHz	Pentium 90 MHz ou supérieur	32 Mo	96 Mo ou supérieur
Serveur	Pentium 133 MHz	Pentium 133 MHz ou supérieur	128 Mo	256 Mo ou supérieur

Glossaire

A

ADO.NET - Suite de technologies d'accès aux données incluses dans les bibliothèques de classes .NET Framework qui fournissent l'accès à des données relationnelles et au langage XML. ADO.NET se compose de classes qui constituent le DataSet (telles que des tables, des lignes, des colonnes, des relations, etc.), de fournisseurs de données .NET Framework et de définitions de types personnalisées (telles que les types SqlTypes pour SQL Server).

Application Web ASP.NET (ASP.NET Web application)-Application qui traite des demandes HTTP (demandes Web) et s'exécute sur ASP.NET. Une application Web ASP.NET peut contenir des pages ASP.NET, des services Web XML, des gestionnaires HTTP et des modules HTTP.

ASP.NET (ASP.NET)-Ensemble de technologies contenues dans Microsoft .NET Framework destinées à créer des applications Web et des services Web XML. Les pages ASP.NET s'exécutent sur le serveur et génèrent un balisage (tel que HTML, WML ou XML) qui est envoyé à un navigateur de bureau ou mobile. Les pages ASP.NET utilisent un modèle de programmation compilé, piloté par des événements, qui améliore les performances et permet de séparer l'interface utilisateur et la logique de l'application. Les fichiers de pages ASP.NET et de services Web XML créés à l'aide de ASP.NET contiennent une logique côté serveur (plutôt que côté client) écrite en Visual Basic .NET, C# .NET ou tout autre langage compatible .NET. Les applications Web et les services Web XML tirent parti des fonctionnalités du Common Language Runtime, tel que la sécurité de type, l'héritage, l'interopérabilité des langages et la sécurité intégrée.

Assembly (assembly)-Collection d'un ou de plusieurs fichiers dont les versions sont gérées et qui sont déployés comme une unité. Un assembly constitue le bloc de construction principal d'une application .NET Framework. Tous les types et les ressources managées sont contenus dans un assembly et marqués comme uniquement accessibles à l'intérieur de l'assembly ou comme accessibles à partir du code figurant dans d'autres assemblys. Les assemblys jouent également un rôle clé dans la sécurité. Le système de sécurité d'accès au code utilise des informations relatives à l'assembly pour déterminer le jeu d'autorisations octroyé au code contenu dans l'assembly. Voir aussi : assembly privé, assembly partagé.

Amorçage – bootstrap ; startup - Synonyme de démarrage. - Processus de mise en service d'un ordinateur : un programme est lu en ROM, des tests sont ensuite réalisés sur le système (mémoire, clavier, lecteurs, etc.) ; le système d'exploitation est alors chargé. Le terme anglais « bootstrap » vient de l'expression des cow-boys to pull one self up by one's own bootstraps qui signifie « se soulever au moyen de ses propres crochets de bottes ». Synonymes : amorçage et initialisation.

Analyseur de protocole - protocol parser - Dispositif qui vérifie que l'information transmise est conforme au protocole utilisé. L'analyseur de protocole teste l'information et non les signaux de transmission.

Application : application ou application program - Sous Windows, synonyme de programme.

API : Application Programming Interface -interface de programmation d'applications- Ensemble de routines utilisées par un programme pour demander et exploiter les services d'un autre programme ou d'une bibliothèque de fonctions. L'API de Windows (User ou User 32) fournit au programme un moyen de communiquer avec le système.

Architecture client/serveur :- client/server architecture - Organisation employée sur des réseaux locaux pour utiliser l'intelligence distribuée afin d'exploiter à la fois le serveur et les stations de travail

comme des dispositifs intelligents, programmables, et tirer parti de la puissance de chacun. Cela se fait en divisant le fonctionnement d'une application en deux composants distincts : un client (ou frontal), et un serveur.

ASP -Active Server Pages - Spécification de pages Web dynamiques, qui contiennent du code Visual Basic ou Jscript. Lorsque le navigateur demande une page ASP, elle est générée en fonction des paramètres qu'elle contient, et envoyée sous forme de page HTML au client. Ceci permet de construire des sites Web évolués, en utilisant des outils de programmation courants. Voir aussi CGI.

B

Bibliothèque de classes .NET Framework (.NET Framework class library)-Bibliothèque de classes, d'interfaces et de types valeur incluse dans le Kit de développement .NET Framework SDK de Microsoft, conforme CLS. Cette bibliothèque, qui permet d'accéder aux fonctionnalités du système, est le fondement des applications, composants et contrôles du .NET Framework.

C

chemin virtuel – virtual path - Séquence de noms utilisée pour localiser un fichier et qui possède la même forme qu'un nom de fichier dans le système de fichiers, mais qui n'est pas nécessairement la séquence réelle de noms de répertoires sous lequel se trouve le fichier.

CGI-Common Gateway Interface - interface de passerelle commune- Interface standard pour le développement d'applications serveur comme HTTP. Cette interface s'adresse aux programmeurs qui créent des applications ou des scripts exécutés sur un serveur Web. Ces scripts peuvent générer du texte, des pages web ou d'autres types de données à la volée, par exemple en réponse à une entrée de l'utilisateur ou à l'extraction d'informations d'une base de données.

Client -1) En programmation orientée objet, membre d'une classe (groupe) qui utilise les services d'une autre classe à laquelle il n'est pas lié. Voir héritage. 2) Programme ou tâche qui fait appel à un service fourni par un autre programme, par exemple un traitement de texte qui appelle une routine de tri d'un autre programme. Le processus client utilise le service requis sans avoir à connaître le fonctionnement de l'autre programme.3) Sur un réseau local ou sur l'Internet, ordinateur qui accède à des ressources réseau partagées fournies par un autre ordinateur, appelé serveur. Voir aussi architecture client/serveur.

C# - C Sharp -Nouveau langage de programmation conçu pour générer des applications d'entreprise s'exécutant dans le .NET Framework. Évolution de C et de C++, C# est orienté objet et possède la sécurisation des types. Comme il est compilé comme du code managé, il bénéficie des services du Common Language Runtime, tels que l'interopérabilité des langages, la sécurité améliorée et l'opération garbage collection.

Code managé (managed code)-Code exécuté par l'environnement du Common Language Runtime plutôt que directement par le système d'exploitation. Les applications de code managé bénéficient des services Common Language Runtime tels que les opérations garbage collection automatiques, la vérification du type au moment de l'exécution et la prise en charge de la sécurité, etc. Grâce à ces services, les applications de code managé ont un comportement uniforme, indépendant de la plate-forme (et du langage). Voir aussi : code non managé.

Code natif (native code)-Code qui a été compilé en code machine spécifique au processeur.

Code non managé (unmanaged code)-Code directement exécuté par le système d'exploitation, hors de l'environnement du Common Language Runtime. Le code non managé doit fournir ses propres opérations garbage collection, la vérification du type, la prise en charge de la sécurité, etc., contrairement au code managé, qui reçoit ces services du Common Language Runtime. Voir aussi : code managé.

COM Interop (COM interop)-Service permettant aux objets .NET Framework de communiquer avec des objets COM.

Common Language Runtime (common language runtime)-Moteur situé au cœur de l'exécution du code managé. Le Runtime fournit du code managé avec des services tels que l'intégration interlangage, la sécurité d'accès du code, la gestion de la durée de vie des objets ainsi que la prise en charge du débogage et des profils.

Common Language Specification (CLS) (Common Language Specification (CLS)) Sous-ensemble de fonctionnalités de langage prises en charge par le Common Language Runtime, comprenant des fonctionnalités communes à plusieurs langages de programmation orientés objet. Les composants et outils conformes CLS sont ainsi assurés de fonctionner avec les autres composants et outils conformes CLS. Voir aussi : conforme CLS.

Compilation JIT (JIT compilation) Just-In-Time -Compilation convertissant le langage MSIL (Microsoft Intermediate Language) en code machine au moment où le code est requis au moment de l'exécution. Voir aussi : MSIL

Conforme CLS (CLS-compliant)-Code exposant publiquement les fonctionnalités de langue (et elles seules) figurant dans la Common Language Specification. La conformité CLS peut s'appliquer aux classes, interfaces, composants et outils. Voir aussi : Common Language Specification (CLS).

COM - Component Object Model - Spécification développée par Microsoft pour créer des composants logiciels qui peuvent être assemblés dans des programmes ou pour ajouter des fonctionnalités à des programmes existants qui fonctionnent sur des plates-formes Microsoft Windows. Les composants COM peuvent être écrits dans différents langages, bien que la plupart le soient en C++; ils peuvent être dissociés pendant l'exécution d'un programme sans qu'il faille le recompiler. COM est à la base des spécifications OLE.

CORBA - Common Object Request Broker Architecture - Spécification, développée par l'Object Management Group en 1992, grâce à laquelle des parties de programmes (objets) communiquent avec d'autres objets dans d'autres programmes, même si les deux programmes sont écrits dans différents langages de programmation et sont exécutés sur différentes plates-formes. Un programme effectue sa requête pour des objets au travers de l'ORB, ce qui lui permet de ne pas avoir à connaître la structure du programme d'où l'objet est originaire. CORBA est conçu pour fonctionner dans des environnements orientés objets.

D

DCOM -Distributed Component Object Model (Distributed COM) - modèle d'objet composant distribué- Version de la spécification Microsoft du modèle objet COM qui stipule comment les composants communiquent sur des réseaux Windows. Il permet la distribution de différents composants d'une même application sur plusieurs ordinateurs en réseau, au moyen d'une application distribuée.

DLL - Dynamic Link Library - bibliothèque de liaisons dynamiques - Sous Microsoft Windows et OS/2, une bibliothèque est dite dynamique dans la mesure où les applications utilisent ses procédures pendant le fonctionnement lorsqu'elles sont nécessaires et non durant la compilation. Elle est mise à jour indépendamment des applications qui utilisent ses procédures. Voir aussi analyseur de protocole.

E

Espace de noms (namespace)_Schéma d'affectation de noms logique pour le regroupement des types associés. Le .NET Framework utilise un schéma d'affectation de noms hiérarchique pour le regroupement de types en catégories logiques de fonctionnalités associées, telles que la technologie ASP.NET ou les fonctionnalités distantes. Les outils de design peuvent utiliser des espaces de noms

pour permettre aux développeurs de naviguer et de référencer plus facilement les types dans leur code. Un seul assembly peut contenir des types dont les noms hiérarchiques ont des racines d'espace de noms différentes et une racine d'espace de noms logique peut s'étendre sur plusieurs assemblys. Dans le .NET Framework, un espace de noms est un espace d'affectation de noms logique au moment du design, tandis qu'un assembly établit la portée du nom des types au moment de l'exécution. Voir aussi : assembly.

F

Fichier code-behind (code-behind file)-Fichier de code contenant la classe de page qui implémente la logique de programme d'une application Web Forms ou Web Forms Mobile ASP.NET

Firewall -pare-feu - Système de sécurité destiné à protéger un réseau contre les intrusions de pirates provenant d'un autre réseau, et à interdire l'accès à ses données. Un pare-feu peut ainsi protéger un service d'une entreprise (comme la direction financière) vis-à-vis des autres services, ou toute l'entreprise vis-à-vis de l'extérieur. Synonymes: coupe-feu et garde-barrière.

FTP -File Transfer Protocol -Protocole utilisé pour copier des fichiers depuis et vers des ordinateurs distants sur un réseau TCP/IP, tel que l'Internet

Framework - Cadre-En programmation orientée objet, structure de conception de base réutilisable, qui consiste en classes abstraites et concrètes, et qui est utilisé dans les applications de construction.

G

GC- opération Garbage Collection Processus de traçage transitif de tous les pointeurs vers les objets utilisés activement pour trouver tous les objets à référencer, puis de réorganisation en vue de réutiliser tout tas non trouvé au cours du traçage. Le garbage collector (également appelé ramasse-miettes) du Common Language Runtime compacte également la mémoire utilisée pour réduire l'espace de travail nécessaire au tas. Voir aussi : tas.

GIF - Graphics Interchange Format - Format standard, avec JPEG, des fichiers images sur le Web. Le format de fichiers GIF est très répandu car il peut utiliser une méthode de compression. Pour réduire la taille des fichiers.

Gopher Utilitaire Internet pour trouver des informations textuelles et les présenter à l'utilisateur.

GUI- Graphical User Interface **-interface graphique** -visual interface - Interface qui produit, sur un écran d'ordinateur, une représentation imagée des fichiers, des données, des équipements et des programmes stockés sur l'ordinateur.

H

HTML -Hypertext Markup Language - langage de balisage hypertexte,- Langage utilisé pour présenter les pages Web. HTML est un système de structuration d'un document par des marqueurs interprétables par les navigateurs du Web.Voir aussi SGML.

HTTP - Hypertext Transfer Protocol - protocole de transfert hypertexte.- Norme World Wide Web pour transférer des données entre des serveurs Web et des clients. Les URL des fichiers de serveurs Web commencent par la chaîne http://. Voir aussi architecture client/serveur.

I

Interface de ligne de commande -command-line interface - Type d'interface, entre le système d'exploitation et l'utilisateur, où l'utilisateur tape les commandes à l'aide d'un langage de commande

spécial. Les systèmes à interface de ligne de commande, par exemple MS-DOS, sont souvent considérés comme plus difficiles à manipuler que les systèmes avec interface graphique ; mais ils sont généralement programmables, ce qui leur donne une souplesse que ne possèdent pas les systèmes graphiques. Voir aussi GUI, interface graphique

J

J2EE- Java 2 Entreprise Edition- plateforme de développement basé sur le langage Java.

JIT- Just-In-Time- Avant d'être exécuté, le langage MSIL doit être converti par un compilateur JIT.NET Framework en code natif, un code spécifique au processeur qui s'exécute sur la même architecture d'ordinateur que le compilateur JIT La compilation JIT tient compte du fait qu'une partie du code ne sera peut-être jamais appelée au moment de l'exécution. Au lieu de consacrer du temps et des ressources mémoire à la conversion de toutes les instructions MSIL d'un fichier exécutable portable (PE) en code natif, elle les convertit au fur et à mesure des besoins au moment de l'exécution et stocke le code natif obtenu afin qu'il soit accessible pour les appels ultérieurs.

JPEG - Joint Photographic Experts Group - 1) Norme ISO/ITU de stockage d'images sous forme compressée, utilisée couramment pour compresser des images photographiques. 2) Nom donné à un graphique stocké dans un fichier au format JPEG.

L

Legacy – héritage en programmation orientée objet

M

Mainframe ou mainframe computer - macro-ordinateur. Ordinateur qui traite les opérations de plusieurs centaines d'utilisateurs. Sa capacité de stockage est généralement très importante.il traite les applications monolithique de l'ancienne génération.

Mapper - to map - Effectuer un mappage ou traduire une valeur en une autre. Par référence aux systèmes de mémoire virtuelle, un ordinateur peut mapper une adresse virtuelle en adresse matérielle. Synonyme : transcrire

Métadonnée - Meta data ou metadata - Donnée qui concerne un ensemble de données. Par exemple, le titre, le sujet, l'auteur et la taille d'un fichier constituent les métadonnées du fichier.

Middleware - logiciel médian- 1) Logiciel intermédiaire entre deux ou plusieurs types de logiciels conçu pour traduire les informations échangées entre eux. Il couvre un large spectre de logiciels et se place généralement entre une application et un système d'exploitation, y compris de réseau, ou un système de gestion de base de données. CORBA ainsi que des programmes de contrôle de réseau ou des programmes de gestion boursière sont des exemples de middleware. 2) Logiciel qui fournit une interface commune de programmation d'applications (API). Les applications écrites avec cette API fonctionnent sur les mêmes systèmes que le middleware. ODBC est un exemple de middleware qui comporte une interface de programmation d'application pour de nombreux types de bases de données. 3) Outil de développement qui permet à l'utilisateur de créer des programmes simples en sélectionnant des services existants et en les liant avec un langage de script.

MIME Multipurpose Internet Mail Extensions - extensions polyvalentes de courrier Internet, Norme qui élargit le protocole SMTP et permet à des données (vidéo, son, fichiers binaires) d'être transmises par messagerie Internet sans qu'il soit nécessaire de les convertir au préalable au format ASCII. Ceci se réalise par l'emploi des types MIME qui décrivent le contenu d'un document. Une application conforme à MIME qui envoie un fichier (par exemple un programme e-mail) attribue un type MIME à ce fichier. L'application de destination doit aussi respecter la norme MIME ; elle se reporte à une liste de documents normalisés organisés en types et sous-types MIME pour interpréter le contenu du fichier

MSIL- Microsoft Intermediate Language **Microsoft Intermediate Language (MSIL) (Microsoft intermediate language (MSIL))**-Langage utilisé comme sortie d'un certain nombre de compilateurs et comme entrée d'un compilateur JIT. Le Common Language Runtime comprend un compilateur JIT pour convertir le langage MSIL en code natif. Voir aussi : compilation

N

Navigateur - browser ou explorer - Logiciel client qui affiche des pages au format HTML et permet l'accès au World Wide Web.

.NET Framework (.NET Framework)-Plate-forme de création, de déploiement et d'exécution de services et d'applications Web XML. Elle offre un environnement multilingue ultra-productif à base de standards intégrant des investissements existants à la prochaine génération d'applications et de services, de même que la possibilité de résoudre les enjeux du déploiement et du fonctionnement des applications Internet. Le .NET Framework comprend trois parties : le Common Language Runtime, un ensemble hiérarchique de bibliothèques de classes unifiées et une version comportant des composants ASP appelée ASP.NET. Voir aussi : ASP.NET, Common Language Runtime, bibliothèque de classes .NET Framework.

News -Nom générique donné à l'ensemble des forums de UseNet, espace de l'Internet réservé aux échanges thématiques ; groupes de discussion ou forum.

O

ODBC - Open Database Connectivity - connectivité de base de données ouverte- Interface dans la structure WOSA de Microsoft qui permet aux applications Windows d'accéder à une base de données quelconque en réseau grâce une interface d'interrogation commune.

OLE -Object Linking and Embedding - liaison et incorporation d'objets - Technologie qui permet de transférer et de partager des informations entre applications.

OMA - Object Management Architecture - Définition créée par l'OMG pour le traitement distribué orienté objet. L'OMA inclut CORBA.

OMG -Object Management Group - Organisation internationale, fondée en 1989, qui valide des normes ouvertes relatives aux applications orientées objet. Elle définit également l'architecture OMA, modèle objet normalisé pour les environnements distribués.

ORB - Object Request Broker - courtier de requête d'objet, - Dans les applications client/serveur, interface à travers laquelle un client réalise une requête sur un objet. L'ORB dirige la requête vers le serveur qui contient l'objet, puis renvoie les valeurs résultantes au client. Voir aussi CORBA.

Modèle OSI - Open Systems Interconnection model (OSI model) - Architecture en couches (plans) qui standardise des niveaux de service et des types d'interaction pour des ordinateurs échangeant des informations dans un réseau de communications.

P

Page ASP.NET (ASP.NET page)-Composant d'une application ASP.NET.

Page code-behind (code-behind page)-Voir la définition de fichier code-behind.

Passerelle -Gateway - Dispositif de connexion de réseaux qui utilise différents protocoles de communication, de manière à ce que les informations puissent passer des uns aux autres. Une

passerelle transfère les informations et les convertit en une forme compatible avec les protocoles utilisés par le réseau de réception.

Port:-1) Circuit d'entrée ou de sortie des données. On parle des ports du microprocesseur (ils reçoivent ou envoient des bits de données). 2) Connecteur d'un ordinateur auquel se branche l'adaptateur d'un matériel, par exemple une imprimante, un disque dur, un modem ou une souris. C'est un emplacement utilisé pour transférer des données dans et hors d'un périphérique. Le port peut faire référence à une carte qui connecte un serveur à un réseau, à un port série 232, à un port TCP/IP

Processus métier:c'est un enchaînement d'activités et leur séquencement, il est interne à une entreprise et une seule.

Protocole de communication -communication protocol - Ensemble de règles ou de normes destinées à permettre aux ordinateurs de se connecter entre eux pour échanger des informations de la manière la plus fiable possible. Le protocole généralement accepté pour normaliser les communications informatiques est un ensemble de directives à sept niveaux, connu sous le nom de modèle OSI.

R

Réseau -(computer) network - Ensemble d'ordinateurs et de périphériques connectés par une liaison spécialisée ou commutée qui assurent entre eux une communication locale ou distante (de voix, vidéo, données, etc.) et facilitent l'échange d'informations entre des utilisateurs ayant des intérêts communs. Un réseau peut se limiter à un réseau local, constitué de quelques ordinateurs, imprimantes et autres périphériques, ou peut comporter de nombreux ordinateurs distribués sur une vaste zone géographique.

Routage -routing-Synonyme d'acheminement. Acheminement -routing - Détermination d'un chemin à travers les noeuds d'un réseau pour transmettre des données.

RPC - Remote Procedure Call - appel de procédure à distance- Méthode de transmission de messages qui permet à une application de faire appel à des services disponibles sur un autre ordinateur du réseau, qui exécute alors la tâche demandée et renvoie le résultat au programme d'origine.

S

Sérialisation (serialization)-Processus de conversion des informations d'état sur un objet en une forme susceptible d'être stockée ou transportée. Pendant la sérialisation, un objet enregistre son état actuel dans le stockage temporaire ou persistant. Par la suite, l'objet peut être recréé par la lecture, ou de la sérialisation, de l'état de l'objet dans le stockage.

Services d'accès distant (remoting)-Processus de communication entre plusieurs processus du système d'exploitation, indépendamment de savoir s'ils se trouvent sur le même ordinateur. Le système .NET Remoting est une architecture simplifiant la communication entre des objets situés dans des domaines d'application différents, qu'ils se trouvent ou non sur le même ordinateur, et dans des contextes différents, qu'ils se trouvent ou non dans le même domaine d'application. Voir aussi : domaine d'application, contexte.

Service Web XML c'est un composant comprenant des fonctionnalités d'entreprise exposées via des protocoles Internet. Un service Web XML permet aux applications d'échanger des informations entre des applications Web en utilisant des normes telles que la messagerie XML et HTTP pour transférer les données à travers des pares-feux. Les services Web XML ne sont liés ni à une technologie de composant spécifique ni à une convention d'appel d'objet particulière. En conséquence, les programmes peuvent accéder aux services Web XML, quels que soient le langage dans lequel ils sont écrits, le modèle de composant utilisé et le système d'exploitation sur lequel ils s'exécutent

SGML - Standard Generalized Markup Language - Norme de gestion des informations adoptée par l'ISO en 1986. Il permet de fournir des documents et une plate-forme, indépendants d'une application, qui conservent la mise en forme, l'indexation et les informations liées. SGML offre un mécanisme analogue à une grammaire pour que les utilisateurs définissent la structure de leurs documents et les marqueurs qu'ils utiliseront pour indiquer la structure des documents individuels.

Serveur -(network) server - 1) Dans un réseau local, ordinateur qui exécute le logiciel d'administration, qui contrôle l'accès au réseau et à ses ressources 2) Ordinateur ou programme qui répond aux commandes d'un client. 3) Sur l'Internet, ordinateur qui contient des sites Web ou des groupes de discussions. Dans les applications OLE, programme qui crée l'objet, à la demande du client. Voir aussi architecture client/serveur.

Serveur HTTP - HTTP server ou Web server - 1) Toute machine sur laquelle fonctionne un programme serveur HTTP. 2) Logiciel serveur qui utilise HTTP pour fournir des documents HTML.Synonyme : serveur Web.

Service -1) Processus capable de fournir des réponses à une requête ou à une interrogation sur un ordinateur distant. Un service de gestion de fichiers distant, par exemple, peut effacer ou transmettre un fichier en réponse à la demande d'un ordinateur client. Comme l'envoi d'une page HTLM pour un serveur HTTP qui fait fonctionner un service HTTP. 2) En programmation, programme ou routine qui fournit un support à d'autres programmes, en particulier à des programmes de bas niveau (proches du matériel).

Serveur Proxy - Proxy Server - Logiciel serveur qui s'exécute sur un ordinateur hôte. Il gère le trafic Internet vers et depuis un réseau local et peut offrir d'autres fonctionnalités comme le cache de documents et le contrôle d'accès. Un serveur proxy améliore l'accès aux données car il peut fournir des données qui sont fréquemment demandées, comme une page Web très fréquentée ; il peut aussi filtrer et rejeter les demandes qui ne conviennent pas à l'administrateur, comme des demandes d'accès non autorisé à des fichiers propriétaires.

services réseau -network services- Dans l'environnement Windows, extensions du système d'exploitation qui permettent de réaliser des fonctions de réseau, telles que l'impression en réseau et le partage de fichiers.

SMTP - Simple Mail Transfer Protocol - protocole simple de transfert de courrier- Protocole TCP/IP utilisé pour envoyer des messages entre plusieurs ordinateurs en réseau. Ce protocole est utilisé sur l'Internet pour router le courrier électronique. Voir aussi protocole de communication, POP 2.

SOAP -Simple Object Access Protocol-Protocole simple basé sur XML, consistant à échanger des informations de type structurées sur le Web. Le protocole ne contient aucune sémantique d'application ou de transport, ce qui le rend hautement modulaire et extensible.

Standard-Norme-Directive technique de jure préconisée par une organisation non commerciale ou gouvernementale, utilisée pour établir une uniformité dans le développement de matériel ou de logiciels. La norme est le résultat d'un processus formel, basé sur des spécifications rédigées par un groupe coopératif ou un comité après une étude approfondie des méthodes existantes, des approches, des tendances et des développements technologiques. La norme proposée est ensuite ratifiée ou approuvée par une organisation reconnue. Les normes de ce type sont nombreuses : par exemple, jeu de caractères ASCII, langages de programmation standard ANSI, tels que le Fortran et le C.

T

Tas (heap) -Portion de mémoire réservée à un programme en vue de l'utilisation du stockage temporaire des structures de données dont l'existence ou la taille ne peut être déterminée pendant le fonctionnement du programme.

TCP/IP -Transmission Control Protocol/Internet Protocol - protocole de contrôle de transmission / protocole Internet,- Protocole développé par le Département de la Défense américain et destiné aux communications entre ordinateurs. Il est inclus dans le système UNIX et est devenu le standard de facto pour la transmission des données sur les réseaux, y compris l'Internet. Voir aussi UDP.

U

UDDI Universal Description, Discovery and Integration Structure indépendante d'une plate-forme fonctionnant comme un répertoire (identique à un annuaire téléphonique) qui offre un moyen de trouver et d'inscrire des services Web sur Internet. La spécification UDDI exige trois éléments : les pages blanches, qui fournissent des informations sur les contacts professionnels ; les pages jaunes, qui classent les services Web en catégories (par exemple, services d'autorisation des cartes de crédit) ; les pages vertes, qui fournissent des informations techniques détaillées sur des services individuels. UDDI contient également un registre opérationnel, qui est actuellement disponible.

UDP -User Datagram Protocol -Protocole sans connexion dans TCP/IP, qui correspond à la couche Transport du modèle OSI. UDP convertit en paquets les messages de données produits par une application pour les transmettre via IP, mais ne vérifie pas la bonne réception des messages. UDP est donc plus efficace que TCP et il est utilisé à de nombreuses fins,la fiabilité dépend de l'application qui génère le message.

URI-Uniform Resource Identifier -Chiffre ou nom qui identifie de manière unique un élément ou un attribut. Les URI incluent les URN (Uniform Resource Name) et les URL (Uniform Resource Locator).Les URI constituent un modèle général pour rechercher des ressources sur Internet ; ils se concentrent plus sur la ressource que sur l'emplacement. En théorie, un URI pourrait trouver la copie la plus proche d'un document en miroir ou localiser un document déplacé d'un site à un autre.Même si les URN sont supposées devenir plus courantes dans le futur, lorsque XML est abordé actuellement, les URI sont des URL dans presque tous les cas.

URL - Uniform Resource Locator, - localisateur de site uniforme- Adresse d'une ressource sur l'Internet utilisée par un navigateur Web pour la trouver. La première partie de l'URL identifie le type de serveur ou le protocole de transfert ; elle est suivie du nom du serveur où se trouve la ressource et éventuellement du chemin d'accès, comme un nom de document, sur ce serveur. Voir aussi chemin virtuel, FTP, HTML, HTTP.

URN - Uniform Resource Name- nom de ressource uniforme- Schéma qui permet d'identifier de façon unique par leur nom les ressources qui peuvent être disponibles sur l'Internet, indépendamment de leur emplacement. Les spécifications du format des URN sont toujours en cours de développement par l'IETF(Internet Engineering Task Force) Elles comportent toutes les URI ayant les combinaisons urn:, fpi:, et path, c'est-à-dire celles qui ne sont pas des URL.

W

WAIS - Wide Area Information Server - serveur d'informations à grande distance- Système UNIX de recherche et de récupération de documents sur l'Internet. Il peut être utilisé pour rechercher dans plus de 400 bibliothèques.

WAP (WAP)-Protocole standard destiné à fournir des communications Internet et des services de téléphonie avancés sur les téléphones, les récepteurs de radiomessagerie, les assistants numériques personnels (PDA) et d'autres terminaux sans fil.

W3C-World Wide Web Consortium-Organisme formé d'institutions commerciales et universitaires, qui supervise et favorise le développement de normes pour tout ce qui concerne le World Wide Web.

Web – abrév de World Wide Web. Toile (mondiale) (Québec) - Ensemble des documents hypertexte reliés qui résident sur les serveurs HTTP du monde entier. Les documents du World Wide Web, appelés pages Web, sont écrits en langage HTML et identifiés par des URL qui spécifient la machine et le nom du chemin qui permet d'accéder au fichier. Les pages sont transmises de noeud en noeud à l'utilisateur sous le protocole HTTP.

Web Forms est une fonctionnalité d'ASP.NET permettant de créer l'interface utilisateur des applications Web Les pages Web Forms sont basées sur la technologie ASP.NET de Microsoft dans laquelle le code exécuté de façon dynamique sur le serveur génère la sortie des pages Web sur le navigateur ou le périphérique client. Rendent automatiquement le HTML conforme au navigateur pour assurer la prise en charge de fonctionnalités telles que les styles, la présentation basée sur Microsoft .NET Framework.

WebMethod L'ajout de cet attribut à une méthode dans un service Web XML créé à l'aide d'ASP.NET permet l'appel de cette méthode à partir de clients Web distants. Cette classe ne peut pas être héritée.

Windows Forms –c'est la nouvelle plate-forme de développement d'applications Microsoft Windows basée sur le .NET Framework. Elle met à disposition un ensemble complet de classes extensibles et orientées objet, avec lequel le développent des applications Windows élaborées est fait. En outre, Windows Forms peut faire office d'interface utilisateur locale dans une solution distribuée multicouche. Comme tous les objets de .NET Framework, les formulaires sont des instances de classes Les formulaires peuvent être des fenêtres d'interface multidocument (Multiple Document Interface, MDI) Windows standard, des boîtes de dialogue ou des surfaces servant à l'affichage de routines graphiques.

WOSA -Windows Open System Architecture - Ensemble d'API Microsoft conçues pour permettre à des applications Windows de différents éditeurs de communiquer sur un réseau. Les interfaces de WOSA comportent ODBC, MAPI, TAPI, Winsock et RPC.

WSDL-Web Services Description Language langage contractuel basé sur XML décrivant les services réseau offerts par un serveur. Voir aussi : langage XML

X

XML-Extensible Markup Language -Sous-ensemble du fichier SGML (Standard Generalized Markup Language) optimisé en vue de sa remise sur le Web. XML offre une méthode uniforme, de description et d'échange des données structurées, indépendante des applications ou des fournisseurs.