وزارة التعليم العالي و البحث العلمي

BADJI MOKHTAR UNIVERSITY- ANNABA

UNIVERSITE BADJI MOKHTAR - ANNABA

جامعة باجي مختار - عنابة

FACULTE DES SCIENCES DE L'INGENIORAT

DEPARTEMENT D'ELECTRONIQUE

Brochure de cours du module :

Technologies du Web

Niveau : Master II

Option : réseaux et télécommunication

Dr. Hafs Toufik

Table des matiéres

Préface 1

CHAPITRE I : Introduction au Web

I.1. Introduction 3

I.2. Définition d'internet 5

I.3. Définition du Web 5

I.4. Termes connexes au Web 5

I.5. Le modèle client- serveur et le protocole HTTP 6

I.5.1. La requête HTTP 7

I.5.2. La réponse HTTP 9

I.5.3. Les codes retour HTTP 10

I.6. Exercice d'application ..... …10

CHAPITRE II : Structure d'un document HTML

II.1. Introduction ……13

II.2. Structure général d'un document HTML ……..……....13

II.3. Syntaxe des balises en HTML ……..…15

II.4. XML et DTD ……...16

II.5. Les principales balises HTML ……...18

II.6. Exercice d'application ……..……..26

II.7. Le langage CSS……...27

II.7.1. Méthodes d'insertion du code CSS…….....28

II.7.2. Les attributs « class » et « id » ……..…….....29

II.7.3. Les styles CSS ……..…….....30

CHAPITRE III : Les langages de script côté client

III.1. Introduction ……35

III.2. Le langage JavaScript ……36

III.2.1. Façons d'insertion de JavaScript en HTML ……..36

III.2.2. Les variables JavaScript ……..……..38

III.2.3. Les conditions JavaScript ……...……...39

III.2.4. Les boucles JavaScript ……..…….....39

III.2.5. Les fonctions JavaScript ……..……..40

III.2.6. Objets, propriétés, méthodes ……..……41

III.2.7. Exercice d'application ……..…….....42

III.3. Introduction à Visual Basic .NET ……..……43

III.3.1. Types de données disponibles dans VB.Net ..……...45

III.3.2. Les variables VB.Net ……..……..45

III.3.3. Les fonctions VB.Net ……..……..47

III.3.4. Les classes et objets VB.Net……..……...48

III.4. Introduction au langage jQuery ……..49

III.4.1. Syntaxe jQuery ……..……...50

III.4.2. Les sélecteurs jQuery ……...……..51

III.4.3. Les méthodes d'événement jQuery ……..……52

CHAPITRE IV: Les langages de script côté serveur

IV.1. Introduction…….54

IV.2. Le langage PHP (Hypertext Preprocessor)……55

IV.2.1. Les variables ……..……...58

IV.2.2. Les opérateurs d'affectation combinée ……..……...59

IV.2.3. Les constantes ……..…….....60

IV.2.4. Les fonctions ……..…….....61

IV.2.5. Les tableaux ……..…….....63

IV.2.6. Les boucles ……..…….....64

IV.2.7.manipulation des bases de données avec MySql……67

IV.2.8.Exercice d'application……71

IV.3. Les frameworks ……..……..72

IV.3.1. ASP .NET ……...73
IV.3.2. JSP……77

Chapitre V : Technologies Web avancées

V.1. Introduction……...82
V.2. AJAX……...82
V.2.1. Principe d'AJAX ……83
V.2.2. Les différentes technologies utilisées par AJAX ……84
V.2.3.Exemple concret d'une application AJAX……..88
V.2.4. Sécurité sous AJAX ……...95
V.3. La plateforme JEE ……96
V.4. Struts ……100
Conclusion générale..…….....107
Bibliographie…….....108

liste des figures

Figure I.1: Organisation internet …….. 4
Figure II.1: Première page Web ……..15
Figure II.2: Emplacement des balises de mise en page ……..22
Figure III.1 : principe des langages coté client ……35
Figure IV.1 : Principe du langage coté serveur ……55
Figure IV.2: Accueil de phpMyAdmin ……68
Figure IV.3: La base test a été créée avec succès ……69
Figure IV.4: Création d'une table ……69
Figure IV.5: Détails d'une table MySQL ……70
Figure IV.6: Principe de base ASP.NET MVC……74
Figure IV.7:Création nouveau projet ASP.NET MVC……75
Figure IV.8:Principe de base du JSP……77
Figure IV.9:Etapes d'une demande de page JSP……...80
Figure V.1. Principe de fonctionnement d'AJAX ……...85
Figure V.2. Exécution d'application AJAX : validation de données de la saisie ……89
Figure V.3. Principe des conteneurs JAVA EE ……100
Figure V.4. Architecture de Struts ……102
Figure V.5. Fonctionnement de Struts ……...103

liste des tableaux

Tableau I.1: Les méthodes http 1.1……8
Tableau I.2 : Les champs http 1.1……...9
Tableau II.1: Les balises mise en forme du texte ……19
Tableau II.2: Les balises des listes ……...21
Tableau II.3: Les balises des tableaux ……24
Tableau II.4: Les balises multimédias ……25
Tableau II.5: Les propriétés CSS du fond de la page……...32
Tableau II.6: Les tailles relatives CSS……..33
Tableau II.7: Les propriétés bordures et ombres CSS ……34
Tableau III.1: les méthodes d'événement jQuery……..53
Tableau IV.1: Les variables prédéfinies PHP……...59
Tableau IV.2:Les variables prédéfinies PHP……60
Tableau IV.3:Quelques constantes prédéfinies PHP……60
Tableau V.1:Status et StatusText……..88

Préface

Le présent manuel est destiné aux étudiants en Master II (Filière : Télécommunications Spécialité : Réseaux et Télécommunications). Il s'intitule Technologies du Web.

L'objectif est de présenter un support d'apprentissage didactique et facile à utiliser afin d'aider les étudiants à aborder sans difficultés le programme officiel de la matière Technologies du Web.

Pour atteindre cet objectif, nous avons procédé à un apprentissage progressif et constructif basé sur la réalisation d'activités permettant une compréhension par l'exemple des concepts proposés.

Le manuel est étayé de nombreux exemples, et exercices construits de manière à ce que l'étudiant acquière le raisonnement nécessaire pour la construction de sites Web et l'utilisation des techniques modernes de collaboration et de communication.

Structuré en cinq chapitres, le manuel mène l'étudiant progressivement dans l'univers de conception et de développement Web.

Le premier chapitre «Introduction au Web » présente un tour d'horizon de l'univers du Web ainsi que les techniques et les outils liés à cet univers.

Le deuxième chapitre «Structure d'un document HTML» est traite l'apprentissage des langages de création des pages Web statiques à savoir les langages HTML et CSS. Dans cette partie, nous avons exposé les concepts fondamentaux qui doivent être présents à l'esprit de tout développeur Web.

Dans le troisième chapitre «Les langages de script côté client», nous avons abordé quelques langages de script côté client comme Javascript, VB script et jQuery.

Le quatrième chapitre «Les langages de script côté serveur», la création de pages Web dynamiques à travers les langages PHP, ASP, JSP et Mysql.

Le dernier chapitre «Technologies Web avancées», illustre les langages Web évolués comme AJAX, JEE et Struts.

A la fin de ce manuel, figurent des références bibliographiques qui apportent un complément d'informations nécessaires pour mieux appréhender les concepts abordés dans les différents chapitres.

Nous restons à la disposition de tous les utilisateurs voulant nous faire parvenir des remarques et suggestions.

Chapitre I

Introduction au Web

I.1.Introduction :

Au cours des dernières décennies, on a assisté à une véritable révolution dans le monde de l'information et l'informatique qui a été engendrée grâce aux progrès offerts par Internet.

Le monde du Web qui fait partie d'internet a permis aux gens de communiquer, rechercher des informations et de créer de nouvelles applications afin de faciliter leurs vies quotidiennes.

Ce chapitre présente d'abord une introduction à Internet et au web en particulier, puis définit le modèle client- serveur et les protocoles utilisés.

I.2.Définition d'Internet :

C'est un réseau qui englobe plusieurs réseaux à la fois. Il est aussi appelé le réseau des réseaux. Internet a pour but d'offrir des services, de faire communiquer les gens, de partager des ressources et de faire la gestion de l'information.

L'idée d'Internet remonte à l'année 1957, en pleine guerre froide. Le gouvernement américain cherche à concevoir un réseau électronique pour échanger des informations militaires. Après des années de recherche, en 1969, le département de la défense américaine dévoile DARPAnet qui devient en 1972

ARPAnet. C'est un réseau qui relie des centres universitaires afin d'échanger des informations qui est considérer comme l'ancêtre d'Internet.

Entre 1991 et 1993, Tim Berners-Lee introduit le protocole HTTP qui a pour but d'afficher les pages Web . Le Web est né et avec lui avec le premier navigateur Web (Mosaic). Depuis 1995, Internet demeure accessible au grand public.

Internet est dirigé par plusieurs entités. Chaque entité s'occupe d'une partie d'internet. Il y a des entités qui s'occupent de l'architecture du réseau, d'autres gèrent le réseau tandis que d'autres s'occupent de la gestion du développement web. La figure suivante montre l'organisation d'internet.

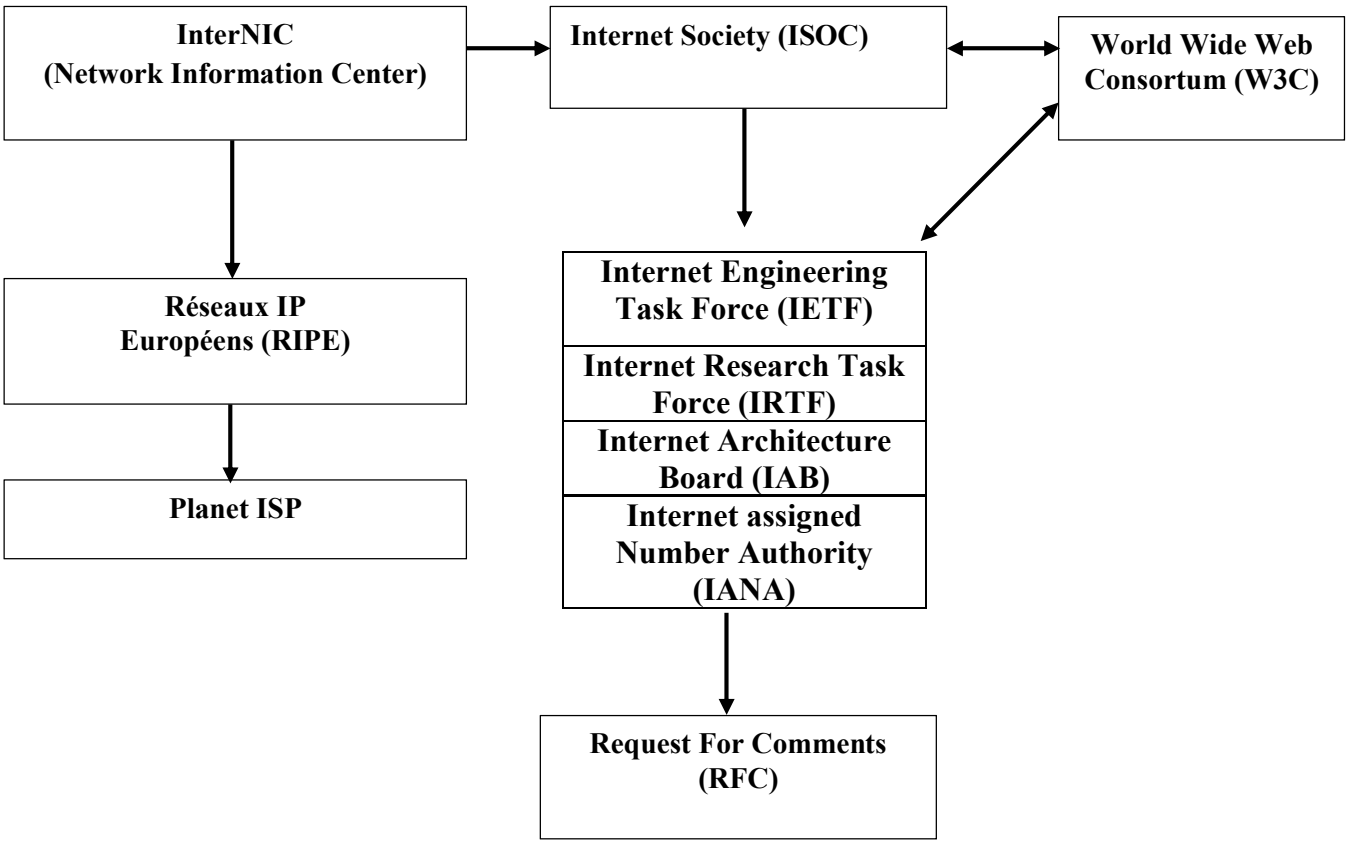


Figure I.1: Organisation internet

I.3.Définition du Web :

C'est l'une des applications d'internet comme le courrier électronique et le partage des fichiers. Le Web est l'appellation courante du *World Wide Web (WWW)* qui est gérer la société *World Wide Web Consortum (W3C)*.

Inventé par *Tim Berners-Lee* et *Robert Cailliau* à la fin des années 1980, le Web est un système basé sur l'utilisation de la notion d'hypertexte. Il permet à ces utilisateurs de consulter des sites et des pages web par le biais d'un navigateur.

I.4.Termes connexes au Web :

Il existe de nombreux termes et expressions rattachés au Web comme surfer sur le Web, serveur Web, navigateur web…. Dans ce qui suit, nous allons définir ces termes en relation avec le Web :

- **Serveur Web :** est un hébergeur de sites Web ainsi que des ressources qui seront proclamées par les utilisateurs d*u* Web. Un serveur Web désigne également le logiciel pour manipuler ces ressources.
- **Site Web :** est un ensemble de pages Web programmées en HTML et relier entre eux avec des hyperliens. Ces sites sont accessibles via une adresse Web et ils doivent être hébergés dans des serveurs Web.
- **Navigateur Web** : est un logiciel conçu pour explorer le contenu du Web. C'est une interface homme-machine qui joue le rôle de traducteur des codes HTML. Il existe autant de navigateurs comme : Internet explorer, Mozilla Firefox, Google chrome….
- **Surfer sur le web :** c'est consulter le contenu Web. Cela signifie l'action de basculement entre les différentes pages Web.
- **La mesure de l'audience** : c'est la mesure du nombre des consultations effectuées sur un site. Cette métrique cherche à comprendre les préférences des utilisateurs, en fonction des ressources les plus utilisées.

- **URL (*Universal/Uniform Resource Locator*)** : c'est une adresse pour repérer un document, un fichier, un ensemble de données. Les noms d'URL sont formés par lettres en minuscule, des chiffres et certains caractères comme **/ . : #** .

  Une URL doit avoir la syntaxe suivante :

  <Protocole> ://<nom serveur>/<chemin>

- **Hébergeur** : c'est une société qui a pour mission de gérer les serveurs qui contiennent les sites Web. Elle s'assure du bon fonctionnement des serveurs 24h/24, 7j/7. En effet, si en panne surgit dans un l'un des serveurs, tous les sites présents sur la machine deviennent inaccessibles.

I.5. Le modèle client- serveur et le protocole HTTP:

Le Web est basé sur le principe réseau appelé le modèle client-serveur. Ce principe repose sur deux entités un client représenté par navigateur Web et le serveur qui contient les ressources. Un serveur est une machine assez puissante qui fournit plusieurs services. D'autre part, le client utilise un programme sur sa machine qui va être demandeur de services, en l'occurrence ce client est un navigateur Web qui va demander des pages Web à un serveur Web. Le dialogue entre le client et le serveur se compose donc de requêtes émises par le client et de réponses données par le serveur. Cet échange s'effectue en quatre étapes :

1. **Connexion :** le client effectue une connexion TCP/IP sur le serveur. Le serveur valide la connexion.
2. **Requête :** le client exprime ses besoins en émettant une requête. Cette requête l'adresse de la ressource demandée (une page Web par exemple).
3. **Réponse :** le serveur expédie le document demandé, ou bien émet un message d'erreur en cas du non disponibilité de la ressource.
4. **Fermeture :** le serveur coupe la connexion.

Cet échange est effectué par le biais d'un protocole destiné uniquement au Web appelé HTTP (HyperText Transfert Protocol). Le protocole HyperText Transfert Protocol est un ensemble de règles qui régit la demande et l'envoi de pages web entre un client et un serveur. HTTP a été défini aussi pour transférer des documents hypermédia établis en HTML. Ces documents sont transmis en binaire afin pour pouvoir émettre des informations différentes (texte, image, son).

HTTP est disponible en trois versions (HTTP 0.9, HTTP 1.0 et HTTP 1.1). La dernière version est compatible avec la plupart des serveurs et navigateurs web.

I.5.1. La requête HTTP

Le client se connecte au serveur et exprime ses besoins (que l'on appelle méthodes), le serveur lui répond et se déconnecte. Cet échange s'appelle une requête.

C'est un protocole en mode texte, généralement utilisé sur une connexion TCP, dédié au transfert de ressource. Pour effectuer une connexion au serveur web de google, il suffit de faire : telnet www.google.fr 80. Ainsi un être humain peut par le biais du protocole "telnet" dialoguer avec un serveur en se servant des noms des commandes ainsi que leurs paramètres. Dans HTTP version 1.1, il existe sept méthodes :

Méthode	Description
GET	Cette méthode permet de demander une ressource (page, document…) sans la modifier.
HEAD	consiste à connaitre des informations sur une ressource
POST	Permets d'envoyer une ressource vers le serveur tel qu'un résultat d'un formulaire ou un fichier. Les informations à envoyer sont contenues dans la requête et non pas dans l'URL.
PUT	Remplace ou ajoute une ressource sur le serveur à condition d'avoir l'autorisation de ce dernier.

DELETE	Suppression d'une ressource depuis un serveur si on possède les droits
TRACE	Permettre d'effectuer un diagnostic de la connexion en demandant au serveur de renvoyer ce qu'il a reçu.
CONNECT	Demande aux auxiliaires de ne pas modifier le contenu des requêtes jusqu'elle arrive à destination du serveur.

Tableau I.1: Les méthodes HTTP 1.1

Les deux méthodes les plus utilisées sont GET et POST. La méthode GET est la méthode la plus simple. La méthode POST permet d'envoyer des informations au serveur dans le corps du message d'une requête HTTP. La méthode HEAD est utilisée essentiellement pour les applications de cache.

En réalité, Il existe peu de serveurs qui tolèrent l'utilisation des méthodes de type PUT et DELETE pour des raisons évidentes de sécurité.

Une requête HTTP se présente sous le format suivant :

- Ligne de commande (Commande, URL, Version de protocole)
- En-tête de requête
- [Ligne vide]
- Corps de requête

La première ligne d'une requête HTTP peut être suivie d'un certain nombre de champs permettant de décrire quelques renseignements concernant le client. Il existe énormément de champs. Dans ce qui suit, nous allons décrire les plus utilisés ou les plus utiles pour la programmation :

Nom du champ	Description
Accept	Type du contenu accepté par le navigateur (par exemple text/html).
Content-Length	Longueur du corps de la requête
Content-Type	Type de contenu du corps de la requête (par exemple text/html).
Date	Date de début de la requête
Forwarded	Utilisé par les proxys entre le navigateur et le serveur
From	Spécification de l'adresse e-mail du client
Referer	URL du lien à partir duquel la requête a été effectuée

If-Modified-Since	Dernière date de réception du contenu
Host	Nom du serveur/domaine de destination
User-Agent	Donne des informations sur le client, comme le nom et la version du navigateur, le système d'exploitation

Tableau I.2: Les champs HTTP 1.1

I.5.2. La réponse HTTP:

Une réponse HTTP est un ensemble de lignes transmis au client par le serveur. Les réponses HTTP présentent le format suivant :

- Ligne de statut (Version, Code-réponse, Texte-réponse)
- En-tête de réponse
- [Ligne vide]
- Corps de réponse

La ligne de statut précise la version du protocole HTTP utilisée ainsi que l'état du traitement de la requête à l'aide d'un code spécifique et d'un texte descriptif. Le code est utilisé par le programme client alors que l'explication permet d'informer l'utilisateur en cas d'erreur.

Des informations supplémentaires sur la réponse, la ressource et le serveur sont formulées par un ensemble de lignes facultatives dans la réponse. Chaque ligne est composée d'un nom qualifiant le type de champ, suivi de deux points (:) ainsi que la valeur correspondante.

Les champs de réponse du protocole HTTP ne sont pas nombreux. Voici les quatre champs les plus utilisés.

Content-Lenght : c'est la longueur du corps de la réponse.

Content-Type : représente le type du document demandé et retourner par la requête.

Date : c'est la date de dernière modification du fichier demandé.

Location : ce champ indique la nouvelle adresse de la ressource demandée.

I.5.3. Les codes retour HTTP:

Dans le protocole HTTP, le serveur doit absolument répondre au client soit on lui fournissant ces ressources ou bien on renvoyant un code d'erreur. Les codes retour sont importants, car ils représentent le statut de la transaction. Le code de réponse est formé de trois chiffres, le premier informe sur la classe du statut et les deux suivants montrent la nature exacte de l'erreur.

La famille des codes **20x** indique que l'opération s'est déroulée avec succès. Le code le plus courant est le code 200 (OK).

La famille des codes de classe **30x** montre que la ressource proclamée a été déplacée, il faut donc introduire un nouveau URL afin d'accéder au contenu de cette ressource.

La famille des codes de classe **40x** intervient lorsque le client a commis une erreur. Il y a deux erreurs très rencontrées : l'erreur '404 Not found' lorsque la ressource demandée n'existe plus, et l'erreur '403 FORBIDDEN' qui indique que le client à saisie un mauvais mot de passe.

La famille des codes de classe **50x** signifie qu'une erreur du côté du serveur est survenue.

I.6. Exercice d'application :

1- Cocher la méthode appropriée pour chaque version de HTTP dans le tableau suivant :

Méthode / Version HTTP	HTTP 0.9	HTTP 1.0	HTTP 1.1
Get			
Post			
Head			
Options			
Put			
Delete			
Trace			
Connect			

2- Un client demande un document à l'adresse http://www.example.com/index.html, il accepte tous les types de documents en retour, il préfère les documents en français, il utilise un navigateur (browser) compatible Mozilla 4.0 sur un système Windows NT 5.1 (Windows XP) et signale au serveur qu'il faut garder la connexion TCP ouverte à l'issue de la requête (car il a d'autres requêtes à transmettre).

- Rédiger la requête correspondante à cette description.

3- Le serveur (qui utilise le HTTP 1.1) répond avec le code 200 pour indiquer que le document demandé a été trouvé. Le logiciel serveur utilisé est Apache/1.3.27. Le serveur transmet la date actuelle, la date de dernière modification du document (avant 5 min de la date actuelle) et la date d'expiration (après laquelle le document doit être demandé à nouveau) qui est de 60 secondes. Le serveur apprend que le document retourné est de type HTML et le corps du document à une longueur de 1456 octets.

- Rédiger la réponse correspondante à cette description.

4- Définir la faille sécuritaire dans cette réponse et proposer une solution convenable à cette faille.

1-

Méthode / Version HTTP	HTTP 0.9	HTTP 1.0	HTTP 1.1
Get	**X**	**X**	**X**
Post		**X**	**X**
Head		**X**	**X**
Options			**X**
Put			**X**
Delete			**X**
Trace			**X**
Connect			**X**

2-

GET /index.html HTTP/1.1
Host: www.example.com Accept: */*
Accept-Language: fr
User-Agent: Mozilla/4.0 (MSIE 6.0; windows NT 5.1)
Connection: Keep-Alive

3-

HTTP/1.1 200 OK
Date: Mon, 07 Oct 2019 23:48:34 GMT
Server: Apache/1.3.27 (Darwin) PHP/4.3.2 mod_perl/1.26 DAV/1.0.3
Cache-Control: max-age=60
Expires: Mon, 07 Oct 2019 23:49:34 GMT
Last-Modified: Mon, 07 Oct 2019 23:43:34 GMT
Accept-Ranges: bytes
Content-Length: 1456
Content-Type: text/html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>

4- La faille sécuritaire dans cette réponse est la définition des informations du serveur (logiciel, type langage). Pour résoudre ce problème, ces informations doivent être cryptées.

Chapitre II

Structure d'un document HTML

II.1 Introduction

Le développement Web est soumis à des règles bien précises pour l'interprétation et la création des pages et sites Web en s'appuyant sur le modèle client/serveur ainsi que le protocole HTTP. Un navigateur Web peut nous procurer plusieurs possibilités pour créer et modifier les pages Web. Cela est rendu possible en utilisant des langages spécifiques :

- HTML (*HyperText Markup Language*) pour créer l'ossature des pages Web
- CSS (*Cascading Style Sheets*) pour mettre en forme les pages Web
- JavaScript pour développer des applications côté client
- PHP (*Hypertext Preprocessor*) pour développer des applications côté serveur

Les langages HTML et CSS sont incontournables, car ils n'ont pas de concurrent et sont à la base de tout projet de développement web. Ils représentent aussi une base pour comprendre, modifier et résoudre les problèmes d'un site Web.

II.2 Structure générale d'un document HTML:

C'est le langage utilisé pour créer des pages Web statiques qui s'appuie sur la notion d'hypertexte et de balisage.

HTML est un langage simple et incontournable. Le navigateur Web va traduire ce code de part et d'autre.

HTML peut être développé en utilisant un simple logiciel comme le notepad++.

La structure de base d'un document HTML est divisée en trois parties. Chaque partie a son utilité par rapport au résultat final :

- **La partie qui indique le type du document :** Elle commence par la balise < !doctype>. Son rôle essentiel est de définir le langage utilisé pour la création des pages Web.
- **La partie entête :** Cette section donne des informations de type général comme le titre de la page ou le caractère du texte. Cette section peut aussi servir pour l'insertion du code CSS ou JavaScript. Cette section commence par la balise <head>. A noter que les informations contenues dans cette partie ne sont pas affichées sur la page.
- **La partie corps de la page :** C'est la partie visible de toute page Web qui va générer le résultat final de l'exécution d'un document HTML. Cette section commence par la balise <body>. Elle contient toutes les balises responsables d'ajouter et d'éditer des pages Web. Toute balise HTML ouverte doit être absolument fermée selon la syntaxe suivante :

<nom de la balise > ……..< /nom de la balise >

Ouverture de la balise — **Fermeture de la balise**

Il existe quelques balises orphelines qui n'ont pas besoin de balise de fermeture par exemple la balise </br>. On peut les appeler également auto-fermant. Voici un exemple d'un document établi en HTML :

```
<!DOCTYPE html>
<html>
<head>
<title> Cours HTML</title>
</head>
<body>
```

```
<p> Page Web en HTML </p>
</body>
</html>
```

Voici le résultat de l'exécution de ce code avec un navigateur (Mozilla Firefox) :

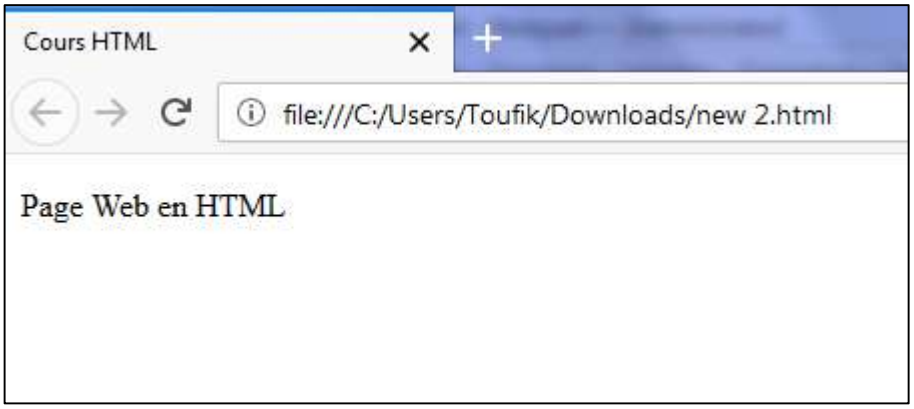


Figure II.1: Première page Web

II.3 Syntaxe des balises en HTML:

Un document HTML est un fichier texte formé par un ensemble de balises. Ces dernières doivent être écrites selon un format bien spécifique afin de décrire correctement la structure du document. Les balises montrent au navigateur comment la façon d'afficher le document. Certaines balises permettent d'ajouter des contenus multimédias comme des sons, des images ou bien des vidéos à côté du texte de la page.

Une balise sous HTML est composée de deux composantes : l'élément et l'attribut selon la syntaxe suivante :

```
<Elément attribut="valeur">      </Elément>
```

Exemple :

```
<a href="http://www.google.com"> Le site de Google </a>
```

L'élément d'une balise représente le nom de la balise qui a une fonctionnalité bien définie. L'attribut représente une caractéristique spéciale ou une option de la tâche principale de la balise qui donne des indications supplémentaires à l'élément. A noter qu'un élément peut avoir plusieurs attributs.

Lors de l'exécution d'un code HTML, le navigateur n'affiche pas les balises telles qu'elles sont écrites. En effet, le navigateur analyse le document et le traduit afin d'afficher la page web convenable au code. Par exemple, si le document contient une balise <video>, le navigateur chargera la vidéo associée et affichera la vidéo à la place de la balise HTML.

II.4 XML et DTD :

La norme XML (*eXtensible Markup Language*) est une manière de description de la construction d'un fichier texte qui permet de stocker des informations en respectant une structure donnée. *Document Type Definition* (en français une Définition de Type de Document) DTD est un ensemble de règles d'écrire une définition de les documents XML. La DTD sert à implémenter un ensemble de règles qui vont construire le document XML.

En réalité, il existe deux types de DTD : les DTD externes et les DTD internes. Les règles des DTD internes sont introduites directement dans le fichier XML alors que les règles des DTD externes sont produites dans un fichier séparé portant l'extension .dtd . Pour définir les règles portant sur les balises, on utilise le mot clef ELEMENT selon la syntaxe suivante :

<!ELEMENT balise (contenu)>

Le but de la norme XML est de définir une structure générale. L'utilisation du XML permet:

- D'échanger des informations entre diverses applications.

- De générer, à partir d'une seule source XML, des documents HTML par exemple possédant différentes caractéristiques selon l'utilisateur final.

Contrairement à HTML, le XML n'est pas un format de présentation. En effet un même fichier XML pourra être utilisé pour donner des informations à un système de traitement, de créer des documents sonores ou bien créer des pages Web en association avec HTML.

Un document XML est constitué d'un prologue au début du document qui sera suivi d'un ensemble de balises.

Le prologue contient une ligne qui commence par **<?xml** afin d'indiquer qu'il s'agit d'un document XML. Cette ligne qui se termine par **?>** permet aussi de préciser la version de XML utilisée ainsi que le type d'encodage utilisé.

XML ne définit aucune balise, c'est à l'utilisateur de les définir ainsi que leurs attributs.

Voici un exemple d'un document en XML :

```
<?xml version="1.0"?>
<annuaire>
   <personne>
      <nom>Hafs</nom>
      <prenom>Toufik</prenom>
   </personne>
   <personne>
      <nom>Mohamedi</nom>
      <prenom>Ahmed</prenom>
   </personne>
</annuaire>
```

Il existe une grande différence entre le HTML et le XML en matière de philosophie de conception. Le XML décrit, structure, échange des données tandis que le HTML ne fait qu'afficher des données. De plus, le XML est extensible et permet de créer ses propres balises en fonction des données traitées. En HTML, les

balises sont prédéfinies. L'association de ces deux langages de programmation à donner lieu à la naissance du XHTML qui est une qu'une adaptation du HTML 4.0 selon la syntaxe du XML.

II.5 Les principales balises HTML :

Dans cette section, on s'intéressera à la description des principales balises utilisées dans HTML. Il existe plus de 140 balises dans HTML version 5. Dans ce qui va suivre, nous allons présenter les balises HTML par catégorie :

- **Balises de formatage de texte :** Ces des balises sont liées essentiellement à la mise en forme du texte des pages Web. On peut citer les balises suivantes :
  - **La balise :** Le langage HTML propose justement la balise pour délimiter les paragraphes.
    Exemple : Voici mon premier site web
     signifie « Début du paragraphe »
     signifie « Fin du paragraphe »
    Cette balise peut être accompagnée par l'attribut (align= "left/right/center/justify ") qui définit l'alignement du texte.
  - **Les balises <h1>à <h6> :** ces balises définies la taille du texte d'une façon décroissante (la balise <h1> définie la taille la plus grande tandis que la balise <h6> définit la taille la plus petite)
    Les balises <h1> à <h6> acceptent l'attribut align = "left" | "center" | "right" .
    Exemple : <h2 align="right">Titre 2 aligné à droite.</h2>

Le tableau suivant résume les balises de mise en forme du texte et leurs descriptions sous HTML :

Balise	Description
<font>	Début d'un texte mis en forme. Cette balise propose les attributs suivants : ***size="Valeur"***, *Valeur* étant un nombre de 1 à 7. Précise la taille du texte. ***face="Valeur"***, *Valeur* étant le nom d'une police. Précise la police de caractère. ***color="Valeur"***, *Valeur* étant une indication de couleur. Précise la couleur du texte.
<blockquote>	Début d'une citation
<sup>	Début d'un texte en exposant. Exemple : texte normal ^{texte en exposant}
<sub>	Début d'un texte en indice. Exemple : texte normal _{texte en indice}
  ou 	Saut de ligne
<hr> ou	Insertion d'une ligne horizontale cette balise propose les attributs suivants : ALIGN=LEFT/RIGHT/CENTER. Pour définir alignement. NOSHADE. Sans ombre SIZE= **"Valeur"**. Précise la taille. WIDTH= ***"Valeur. Précise la largeur.***
<i>	Mettre en italique
	Mettre un peu en valeur votre texte
	Mettre un texte bien en valeur
<mark>	Permets de faire ressortir visuellement une portion de texte
<pre>	Insertion d'un texte préformaté
<big>	Mettre le texte en gras
<!-- voici un commentaire -->	Insertion d'un commentaire

Tableau II.1: Les balises mises en forme du texte

- **Les balises des listes :** Ces des balises liées à l'organisation dans la page web sous forme de liste. Il existe trois types de liste :

- **Liste à puces :** c'est une liste non ordonnée qui a la syntaxe suivante :

```
<ul>
    <li> premier élément</li>
    <li> deuxième élément</li>
</ul>
```

- **Liste numérotée :** c'est une liste non ordonnée qui a la syntaxe suivante :

```
<ol>
    <li> premier élément</li>
    <li> deuxième élément</li>
</ol>
```

Il est possible de choisir le type de puce que nous souhaitons grâce à l'attribut type.

Pour une liste ordonnée (ol), l'attribut type peut avoir les valeurs suivantes :

type="1" => pour une suite numérique

type="a" => pour une suite alphabétique

type="A" => pour une suite alphabétique en majuscule

type="I" => pour une suite numérique en chiffres romains

Pour une liste non-ordonnée (ul), l'attribut type peut avoir les valeurs suivantes :

type="disc" => pour un rond plein

type="circle" => pour un rond vide

type="square" => pour un carré

- **Liste de définitions :** C'est une liste qui comporte des définitions de termes selon la syntaxe suivante :

```
<DL>
<DT>Terme à définir </DT>
```

<DD>La définition du terme </DD>

</DL>

Le tableau suivant résume les balises de création de listes et leurs descriptions sous HTML :

Balise	Description
	Début d'une liste ordonnée.
	Début d'une liste non-ordonnée.
<DL>	Début d'une liste de définition.
	Début d'un item de la liste (une puce sera ajoutée).
<DT>	Création du terme à définir dans une liste de définition
<DD>	Création de la définition dans une liste de définition

Tableau II.2: Les balises des listes

- **Les balises des listes :** Ces balises sont destinées à personnaliser une page web. Dans ce qui va suivre, nous allons présenter quelques balises qui organisent les menus de navigation, les différentes sections au centre et le pied de page. A noter que ces balises existent uniquement dans HTML5 .
  - **La balise <head>** : Cette section donne quelques informations générales sur la page, comme son titre, l'encodage (pour la gestion des caractères spéciaux), etc. Cette section est généralement assez courte. Les informations que l'en-tête contient ne sont pas affichées sur la page, ce sont simplement des informations générales à destination de l'ordinateur. Elles sont cependant très importantes.
  - **La balise <footer> :** C'est une balise qui personnalise le pied de page. On y trouve des informations comme des liens de contact, le nom de l'auteur, les mentions légales, etc.
  - **La balise <nav> :** Cette balise regrouper tous les principaux liens de navigation du site.

- **La balise <section> :** Cette balise sert à regrouper des contenus en fonction de leur thématique. Elle englobe des portions du contenu de la page.
- **La balise <aside> :** cette balise est conçue pour ajouter des informations complémentaires à la page Web. Ces informations sont généralement placées sur le côté de la page.
- **La balise <article> :** Cette balise sert à englober une partie indépendante de la page. Cette partie pourrait ainsi être reprise sur un autre site.

La figure suivante indique l'emplacement de chaque balise de mise en page dans une page Web :

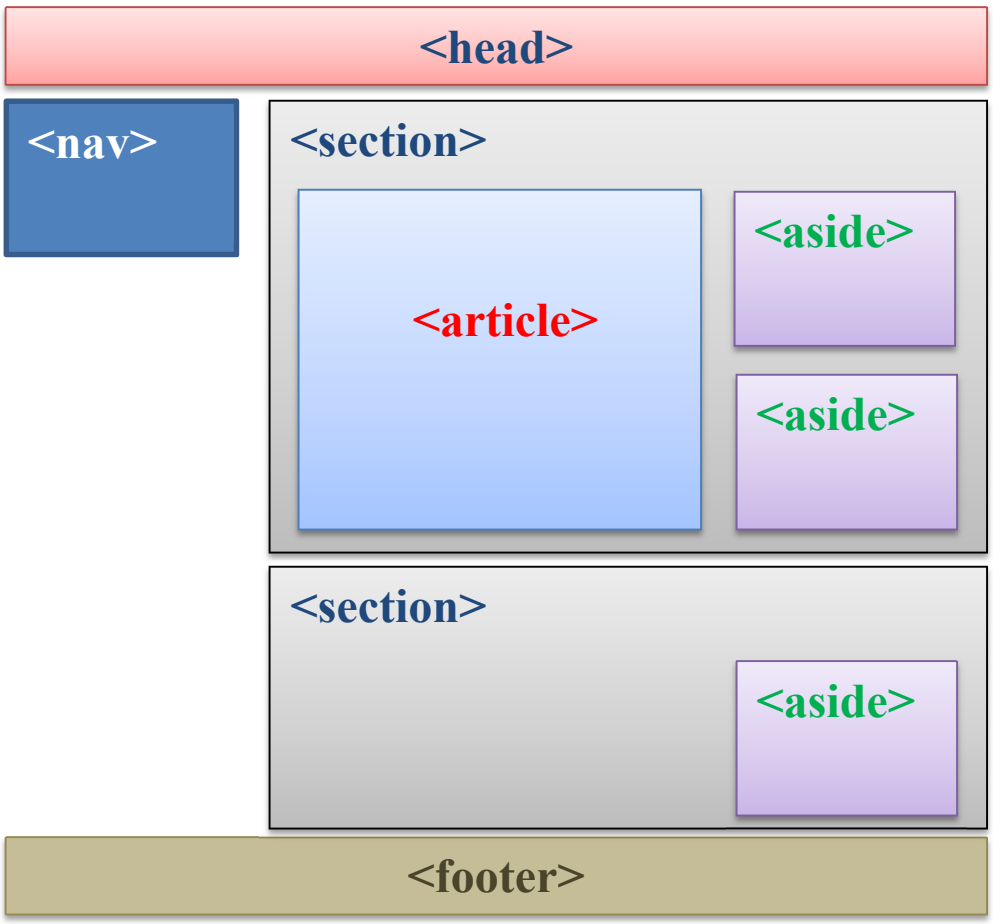


Figure II.2: Emplacement des balises de mise en page

❖ **Les balises de liens hypertextes :** Les liens hypertextes représentent la base de la navigation Web. Les liens permettent aux utilisateurs de basculer d'une page vers une autre ou d'un site vers un autre. Il existe trois façons de créer des liens sous HTML :

- **Les liens dans la même page :** C'est le principe des blogs (site à une seule page) ou des réseaux sociaux. Avec cette notion, on peut naviguer dans la même page en utilisant la notion d'ancre. A travers ces liens, on peut se déplacer le même espace de la page en spécifiant à chaque fois le nom de l'ancre. Cela est effectué avec la balise <a> et l'attribut "href" selon la syntaxe suivante :

  ..

  Pour définir l'emplacement de l'ancre dans la page il suffit d'attribuer à l'élément vers lequel on veut pouvoir pointé un identifiant en utilisant la balise et l'attribut "id" selon la syntaxe suivante :

  ...

- **Les liens vers une autre page :** C'est le basculement vers une page d'un même site selon la syntaxe suivante :

  Lien

  Il est conseillé d'avoir toutes les pages du site dans le même dossier afin d'éviter l'insertion du chemin complet de la page selon :

  Lien

- **Les liens vers un autre site :** C'est le basculement vers une page d'un autre site selon la syntaxe suivante :

  Lien vers Youtube

Par défaut un lien non visité est en bleu et un lien visité est en violet. La couleur de l'état d'un lien est identifiée par l'attribut "link" de la balise <body> selon la syntaxe suivante :

<body link=" couleur">

On peut distinguer trois états pour chaque lien avec trois couleurs associés :

- Les liens hypertextes avant le clic sont en bleu selon :

<body link="blue">

- Les liens hypertextes est pendant le clic sont en jaune selon :

<body alink="yellow">

- Les liens hypertextes visités sont en rouge selon :

<body vlink="red">

❖ **Les balises des tableaux :** Les tableaux sont des éléments essentiels dans le contenu d'une page. Ils permettent de bien présenter les données sur une page Web. Le tableau suivant résume toutes les balises liées aux tableaux :

Balise	Description
<table>	Début d'un tableau. Cette balise propose les attributs suivants : width="Val", Val étant un entier ou un pourcentage. Précise la largeur du tableau cellpadding="Val", Val étant un entier. Précise l'espacement entre le texte d'une cellule et le bord de la cellule cellspacing="Val", Val étant un entier. Précise l'espacement entre 2 cellules border="Val", Val étant un entier. Précise la largeur du trait des bords des cellules.
<tr>	Début d'une ligne du tableau
<td>	Début d'une cellule (ou case) du tableau [td => table data]. Cette balise propose les attributs suivants : width="Val", Val étant un entier ou un pourcentage. Précise la largeur de la cellule align="Val", Val pouvant être left (gauche), right (droite), center (centré) ou justify (justifié). Précise l'alignement horizontal du texte. valign="Val", Val pouvant être top (haut), bottom (bas), ou middle (milieu). Précise l'alignement vertical du texte.
<th>	Début d'une cellule d'intitulé (entête, généralement utilisé pour la première ligne) du tableau [th => table header]. Cette balise propose les mêmes attributs que la balise td (cf ci-dessus).

Tableau II.3: Les balises des tableaux

❖ **Les balises multimédias :** Les fichiers multimédias sont des éléments interactifs dans un site Web.

Ils permettent d'attirer l'attention d'un visiteur par rapport au contenu du site.

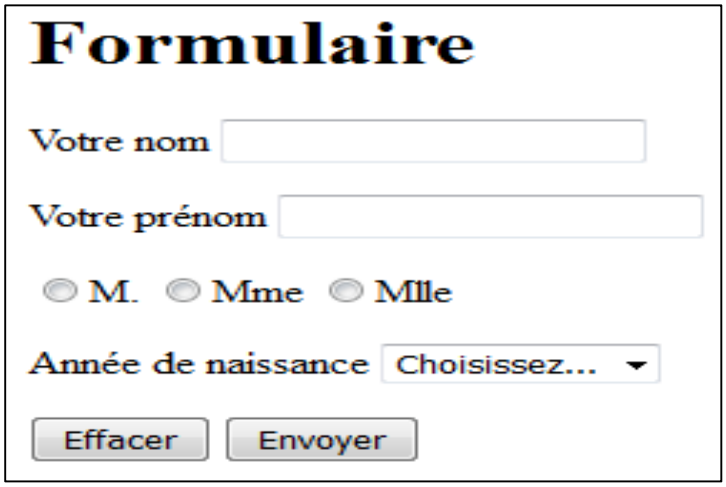
Le tableau suivant résume toutes les balises liées à l'insertion des contenus multimédias :

Balise	Description
<img/>	 Insertion d'une image Cette balise propose les attributs suivants : src="nom de l'image (chemin).l'extension"
<video>	<video src="mon-video.mp4" controls></video> Cette balise propose les attributs suivants : **src :** indique le nom du fichier, son extension et son chemin **contrôlent** : pour afficher les contrôles de base du lecteur vidéo (lecture, pause, contrôle du volume). **autoplay** : spécifier que la vidéo doit commencer instantanément quand la page est chargée. **preload** : charger le contenu du fichier vidéo en même temps que la page Web. **loop** : permets de répéter la vidéo en boucle. **muted** : permet de mettre le volume de la vidéo à 0.
<audio>	Cette balise fonctionne de la même façon que l'élément vidéo, mais il est limité aux formats audio. <audio src="mon-audio.mp3" type="audio/mp3" controls> </audio>

Tableau II.4: Les balises multimédias

II.6 Exercice d'application :

Adapter votre page HTML afin d'y insérer le Formulaire suivant :



Solution:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml: lang="fr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-15" />
<title>Formulaire</title>
</head>
<body>
<h1>Formulaire</h1>
<form action="mailto: toufikhafs@univ-annaba.dz" method="post">
<p>Votre nom <input type="text" name="nom" /> </p>
<p>Votre prénom <input type="text" name="prénom" /></p>
<p><input type="radio" name="titre" value="1" />M.
<input type="radio" name="titre" value="2" />Mme
<input type="radio" name="titre" value="3" />Mlle
```

```
</p>
<p> Année de naissance
<select name="année">
<option value=""> Choisissez...</option>
<option>1985</option>
<option>1986</option>
<option>1987</option>
<option>1988</option>
</select>
</p>
<p><input type="reset" value="Effacer" />
<input type="submit" value="Envoyer" /></p>
</form>
</body>
</html>
```

II.7 Le langage CSS :

CSS (*Cascading Style Sheets*) un langage associé à HTML qui a pour but de mettre en forme les pages HTML. Le CSS décrit comment des éléments de HTML devraient être montrés.

CSS ne peut pas être utilisé d'une façon indépendante. Il doit être absolument associé à HTML afin de définir la façon d'affichage d'une page. CSS est très performant en matière de couleur parce qu'il permet d'utiliser une large palette de couleurs selon les standards de colorisation (RGB, HSV, Y Cb Cr).

II.7.1 Méthodes d'insertion du code CSS :

On peut insérer le code CSS dans HTML de trois façons différentes :

- **Dans les balises HTML :** Cette méthode consiste à insérer les commandes de style dans chaque balise HTML

  Exemple : <div background:#fff; border:1px>

  Cette façon est vivement déconseillée à cause de l'ambigüité qu'elle peut causer à l'intérieur des codes.

- **A l'intérieur de la balise Head :** Cette méthode consiste à insérer tous les styles de la page dans une seule section en se servant de la balise <style> suivie du nom de chaque balise qu'on veut la mettre en forme.

  Exemple :

```
<!DOCTYPE >
<html>
<head>
<style>
div
{
background:#f255
}
</style>
</head>
<body>
<div> boite 1 </div>
</body>
</html>
```

- **Dans un fichier à part :** Cette méthode consiste à écrire le code CSS dans un fichier spécial ayant l'extension**".css"**. Cette méthode est la plus pratique et la

plus recommandé parce qu'elle nous évite de tout mélanger dans un même fichier. Ce fichier CSS est appelé dans la partie <head> selon la syntaxe suivante : <link rel="stylesheet" href="style.css" />

Exemple :

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8" />
      <link rel="stylesheet" href="style.css" />
      <title>fichier CSS</title>
   </head>
   <body>
      <h1>bonjour</h1>
      <p> bienvenue sur mon site </p>
     </body>
     </html>
```

Et voici le fichier CSS externe (style.css) appelé avec la balise <link>. A noter que les styles sont appliquer par balise:

```
h1
{
   color: blue;
}
p
{
   color: red;
}
```

II.7.2 Les attributs « class » et « id » :

Ce sont les deux outils principaux pour la sélection des zones dans une page Web. L'attribut "class" peut être utilisé plusieurs fois dans le document en attribuant

à chaque zone un nom bien déterminé. Cet attribut est inséré dans les balises de la façon suivante :

<nom de la balise class="nom de la zone">

Elle est appelée dans les fichiers CSS de la manière suivante :

```
. nom de la zone
{
   Les styles…

}
```

Il y a une autre façon de définir une zone par l'utilisation de l'attribut "id". Cet attribut est utilisé une seule fois dans le code et possède le même principe que celui de l'attribut "class". Cet attribut est inséré dans les balises de la façon suivante :

<nom de la balise id="nom de la zone">

Elle est appelée dans les fichiers CSS de la manière suivante :

```
# nom de la zone
{
   Les styles…
}
```

II.7.3 Les styles CSS:

CSS est très puissant en matière d'application des styles. Il peut modifier l'apparence d'une page Web en s'appuyant sur le changement des couleurs, des fonts de texte ….etc

- **Les couleurs sous CSS :** Contrairement à HTML, CSS peut définir des millions de couleurs. Il existe trois façons de définir des couleurs sous CSS :

- Soit en introduisant le nom d'une couleur prédéfinie. Il existe 16 couleurs prédéfinies (white, silver, gray, black, red, maroon, lime, green, yellow, olive, blue, navy, fuchsia, purpule, aqua, teal)

  Exemple:

```
h1
{
   color: maroon;
}
```

- Soit en utilisant une combinaison en hexadécimale formée de chiffre et de lettres (de 0 à 9 et de A à F). Ces lettres ou chiffres fonctionnent par pair où une quantité de rouge, une quantité de vert et une quantité de bleu sont mélangées respectivement pour former une couleur .Ainsi, #000000 correspond à la couleur noire et #FFFFFF à la couleur blanche.

  Exemple:

```
p
{
   color: #FFFFFF;
}
```

- Soit en utilisant le standard RGB en définissons une quantité de rouge, de vert et de bleu.

Exemple:

```
p
{
   color: rgb(240,96,204);
}
```

- **Couleur et de fond :** Avec CSS, on peut modifier le fond de la page Web soit avec des couleurs ou bien avec des images. Le tableau résume les propriétés CSS liées à cette opération :

Propriété CSS	Description
color	Couleur du texte
background-color	Couleur de fond
background-image	Image de fond
background-attachment	Fond fixe
background-position	Position du fond
background-repeat	Répétition du fond
opacity	Transparence

Tableau II.5: Les propriétés CSS du fond de la page

- **Format du texte :**

Pour changer la police du texte, on utilise la propriété CSS ***font-family***. On doit insérer le nom de la police comme ceci :

```
balise
{
   font-family: nom de la police
}
```

Voici une liste des noms de polices les plus utilisées : Arial, Arial Black, Comic Sans MS, Courier New, Georgia, Impact, Times New Roman, Trebuchet MS, Verdana.

Pour changer la taille d'un texte, la propriété CSS ***font-size*** est utilisée. Il existe deux manières de définir la taille d'un texte :

- **Définir une taille absolue :** en pixels, en centimètres ou en millimètres. Cette méthode présente une précision élevée, mais il est conseillé de l'utiliser avec modération par risque de définir une taille trop petite pour certains utilisateurs du site Web.

Exemple : font-size: 16px;

- **Définir une taille relative :** cette technique est plus souple et plus adaptable aux préférences de taille des utilisateurs des sites Web. Il y a plusieurs moyens d'indiquer une valeur relative. Le tableau suivant résume toutes les façons de définition des tailles relatives :

Façon	Taille
xx-small	minuscule
x-small	très petit
small	petit
medium	moyen
large	grand
x-large	très grand
xx-large	gigantesque
1em	taille normale
1.3em	grossir le texte
0.8em	réduire le texte

Tableau II.6: Les tailles relatives CSS

- **Les bordures et les ombres CSS:** CSS nous offre une large palette de bordures pour personnaliser une page Web. Il existe plusieurs propriétés CSS qui permettent de modifier l'apparence des bordures. Le tableau suivant regroupe ces propriétés ainsi que leurs descriptions :

Propriété	Description
border-top	bordure du haut
border-bottom	bordure du bas
border-left	bordure de gauche
border-right	bordure de droite
border-width	Épaisseur de bordure
border-color	Couleur de bordure

border-style	Type de bordure : none: pas de bordure (par défaut) solid: un trait simple. dotted: pointillés. dashed: tirets. double: bordure double. groove: en relief . ridge: autre effet relief . inset: effet 3D global enfoncé. outset: effet 3D global surélevé.
border-radius	Bordure arrondie
box-shadow	Ombre d'une boîte
Propriétés des boîtes	
margin-top	Marge en haut
margin-left	Marge à gauche
margin-right	Marge à droite
margin-bottom	Marge en bas
padding-top	Marge intérieure en haut
padding-left	Marge intérieure à gauche
padding-right	Marge intérieure à droite
padding-bottom	Marge intérieure en bas

Tableau II.7: Les propriétés bordures et ombres CSS

Chapitre III

Les langages de script côté client

III.1 Introduction

Les technologies du monde Web offrent la possibilité d'effectuer des traitements de l'information de trois grands axes distincts : la programmation côté serveur, la programmation côté client et les feuilles de style CSS.

Pour améliorer la rapidité de la navigation dans l'espace Web, il est fortement conseillé de limiter au maximum l'utilisation des programmes côté serveur. En effet, chaque utilisation d'un programme côté serveur engendre un envoi d'informations du client vers le serveur, puis l'attente puis l'affichage d'une page reçut du serveur ce qui va couter énormément de temps et de données. Le principe des langages côté client est illustré dans la figure suivante (Javascript comme exemple) :

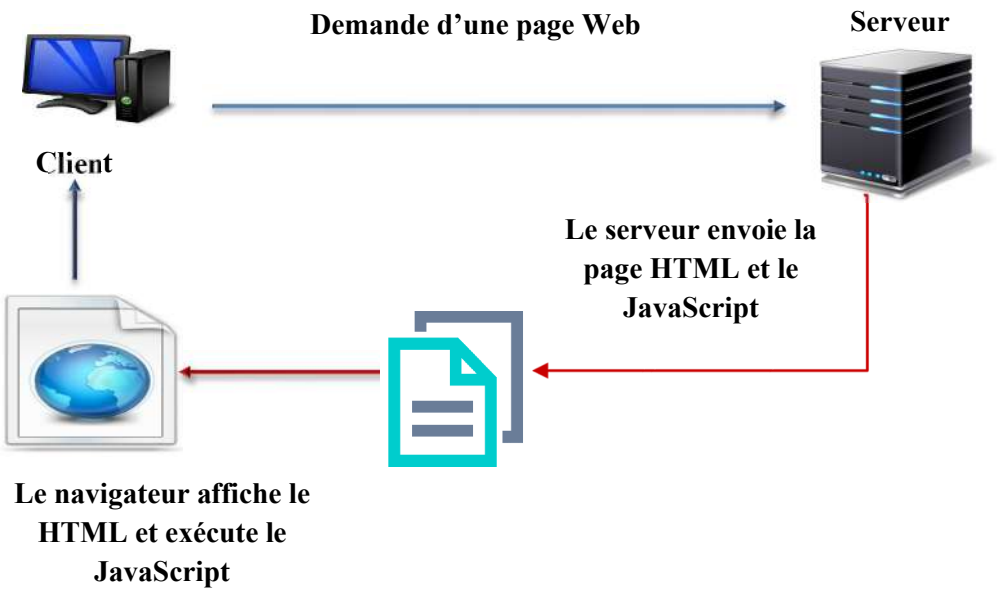


Figure III.1: principe des langages côté client

La programmation côté client peut se faire par le biais de plusieurs langages de programmation. Dans ce chapitre, nous allons illustrer trois langages cotés client à savoir le Java script, le visual basic et le jquery.

III.2 Le langage JavaScript

JavaScript est un langage complémentaire côté client qui est utilisé avec HTML et CSS. JavaScript a été défini pour compléter le manque de l'aspect mathématique dans HTML et CSS. Il est capable de définir plusieurs formules mathématiques qui vont créer des applications interactives dans les pages Web.

JavaScript est dit un langage coté client parce qu'il est utilisé uniquement pour développer des applications cotées client.

Ainsi le langage JavaScript est fortement dépendant du navigateur appelant la page web dans laquelle le script est incorporé, mais en contrepartie il ne nécessite pas de compilateur. Lorsqu'un utilisateur demande une page Web depuis un serveur, ce dernier va lui envoyer la page codée en HTML et CSS ainsi que le code JavaScript qui va avec. A la réception, le client va afficher le code HTML et CSS et exécute le code JavaScript par le biais de son navigateur Web.

Javascript est *case sensitive* (en français sensible à la casse), c'est-à-dire qu'il fait une différence entre un nom de variable contenant ou non des majuscules. Chaque instruction Javascript se termine par un point-virgule (;).

III.2.1 Façons d'insertion de JavaScript en HTML :

JavaScript peut être incorporé avec HTML et CSS de trois façons différentes :

- **Grâce à la balise <SCRIPT> :** On peut insérer un code JavaScript partout dans le document HTML en utilisant la balise <SCRIPT>. Il est important d'avoir une synchronisation entre le code HTML et le code JavaScript. Pour

cela, il est préférable d'insérer le code JavaScript dans la partie <Head> du code HTML. Voici un exemple de cette façon d'insertion :

```
<head>
 <SCRIPT language="Javascript">
 Placez ici le code de votre script
 </SCRIPT>
 </head>
```

- **Dans un fichier séparé :** Il est possible d'insérer le code JavaScript dans un fichier externe Le code à insérer est le suivant :

  ```
  <SCRIPT LANGUAGE="Javascript" SRC="url/fichier.js">
  </SCRIPT>
  ```

  Où url/fichier.js correspond au chemin d'accès au fichier contenant le code en JavaScript.

- **Grâce aux événements :**

  Un événement est une action de l'utilisateur, comme le clic d'un des boutons de la souris. Voici comme insérer un évènement dans le code :

  ```
  <balise eventHandler="code Javascript à insérer>
  ```

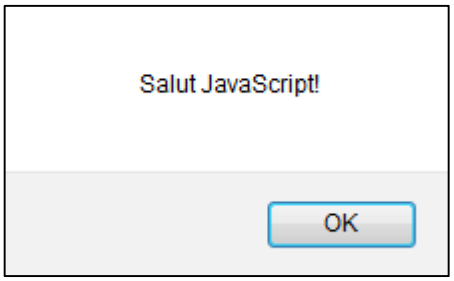
  Où *eventHandler* représente le nom de l'événement.

  Voici un code HTML simple qui contient un code JavaScript, placé à l'aide d'une balise <script> pour afficher une boite de dialogue :

  ```
  <!DOCTYPE html>
  <html>
  <head>
     <title>Salut JavaScript!</title>
  </head>
  ```

```
<body>
    <script>
        alert('Salut JavaScript!');
    </script>
</body>
</html>
```

On obtient le résultat suivant :



III.2.2 Les variables JavaScript :

JavaScript admet tous types de variables, il peut supporter des variables de type entier, de type texte, de type booléen ou bien de type réel. Voici quelques exemples de déclarations de variables sous JavaScript :

```
var number = 5; // type entier
var number = 5.678; // type réelle
var text1 = "Mon premier texte"; // Chaine de caractères
 var isTrue = true; // booléen
```

JavaScript peut aussi faire n'importe quelle opération mathématique par exemple les opérations arithmétiques (addition, multiplication division ou soustraction), les opérations logiques ainsi que les opérations de comparaisons.

III.2.3 Les conditions JavaScript :

JavaScript peut générer des conditions en utilisant « if else ». Cette opération permet à l'utilisateur d'avoir plusieurs cas de figure. Cette condition s'écrit selon la syntaxe suivante :

if (*condition*)
instruction1
else
instruction2

Exemple :

```
var nom = prompt("Entrez un prénom");
if(nom == "toufik")
    alert("C'est moi !");
else if(nom == "ali")
    alert("C'est mon frére  !");
else if(nom == "Fouad")
    alert("C'est mon cousin");
else
    alert("C'est pas un membre de la famille");
```

III.2.4 Les boucles JavaScript : Les boucles permettent de répéter des actions d'un manière simple et rapide. Voici les différentes boucles fournies par JavaScript :

- **La boucle « for » :**

Une boucle « for » répète des instructions jusqu'à ce qu'une condition donnée ne soit plus vérifiée. Dans JavaScript, une boucle « for » s's'écrit comme suit :

```
for ([expression Initiale]; [condition]; [expression Incrément]) {
 //corps de la boucle for
      }
```

- **La boucle « while »:**

  L'instruction while permet de créer une boucle qui s'exécute tant qu'une condition de test est vérifiée. La condition est évaluée avant d'exécuter l'instruction contenue dans la boucle. Cette boucle se présente de la syntaxe suivante :

  ```
  while ( c ) {
  //corps de la boucle while
  }
  ```

- **La boucle « do...while » :**

L'instruction **do...while** permet de répéter un ensemble d'instructions jusqu'à ce qu'une condition donnée ne soit plus vérifiée. Une instruction **do...while** s'utilise de la façon suivante :

```
do {
//corps de la boucle do
} while ( c );
```

III.2.5 Les fonctions JavaScript :

Une fonction Javascript est une suite d'une ou plusieurs instructions ou commandes. Chacune de ces opérations est soit définie par un script soit prédéfinie par le langage de programmation lui-même.

Ainsi, dans l'exemple précédent, la fonction Number (ligne 5) a-t-elle pour effet de recevoir une valeur, ici une suite de caractères (le contenu d'une case de texte), et de retourner le nombre qu'elle contient, s'il y en a un, 0 sinon. Voici un autre exemple de fonction :revenons d'abord sur le type le plus simple d'instruction : l'appel de fonction. Il s'écrit de la façon suivante :

```
nom_de_la_fonction ( liste, de, valeurs );
```

On fera attention au point-virgule, qui termine les instructions. Le saut de ligne les termine également. On sera donc très vigilant à l'indentation.

III.2.6 Objets, propriétés, méthodes:

JavaScript est un langage à objets. Les objets se présentent comme étant des valeurs complexes ayant plusieurs "attributs. Ces derniers sont de deux catégories :

- Les propriétés qui sont soit des valeurs simples (nombre, booléen ou chaîne de caractères) soit d'autres objets.
- Les méthodes qui sont des fonctions qui s'appliquent implicitement à leur objet.

Les propriétés et méthodes sont construites par le nom de l'objet suivi d'un point puis du nom de la propriété ou méthode.

Par exemple, « document.f. » est formé de la propriété f de l'objet document, en l'occurrence il s'agit de son formulaire nommé f. Nous avons également rencontré document.write. Il s'agit de la méthode write de l'objet document.

A présent nous allons expliquer l'expression document.f.A.value :

document : le document courant

document.f : la propriété f de ce document, dans ce cas c'est le formulaire nommé f. en effet, les documents possèdent les propriétés des formulaires HTML qu'ils contiennent .

document.f.A : la propriété A de ce formulaire, comme le contrôle du champ nommé A (une case de texte), en effet, les formulaires ont comme propriétés les contrôles qu'ils contiennent, désignés par les noms de champs.

document.f.A.value : la valeur du champ, c'est une chaîne de caractères correspondant à ce que l'on aura saisi dans la case.

Dans JavaScript, il existe des fonctions globales, appelées globalement et qui renvoient directement leur résultat à l'objet appelant. Voici quelques exemples de ces fonctions :eval(), uneval(), isFinite(), isNaN(), parseFloat(), parseInt(), decodeURI(), decodeURIComponent(), encodeURI(), encodeURIComponent(), escape(),unescape().

III.2.7 Exercice d'application :

Vous avez une page avec un formulaire et un champ. Écrivez un script qui va valider le formulaire. Si le champ est vide, un message d'erreur est affiché à côté du champ et le formulaire n'est pas envoyé. Si le champ contient des données, le formulaire peut être envoyé.

Solution :

Le fichier HTML est :

```
<!DOCTYPE html>
   <html lang="fr"><head>
<script type="text/javascript"src="script.js"></script>
     </head><body>
       <div>
         <form id="myForm" onsubmit="return validateForm();">
           <div>
      <label for="myField">Champ :</label>
             <input type="text" id="myField" size="60">
             <span id="myFieldError"></span>
           </div>
           <div>
           <input type="submit" value="Envoyer">
<input type="submit" value="Envoyer">
```

```
        </div>
      </form>
    </div>
  </body>
</html>
```

Le script JavaScript est :

```
function validateForm() {
  if(document.getElementById('myField').value == '') {
    document.getElementById('myFieldError').innerHTML = 'Le champ ne peut être vide';
    return false;
  } else {
    document.getElementById('myFieldError').innerHTML = '';
    alert('Le formulaire peut être envoyé');
    return true;
  }
}
```

III.3 Introduction à Visual Basic .NET:

Visual Basic .NET (VB.NET) est un langage de programmation informatique orienté objet implémenté sur le .NET Framework. Bien qu'il s'agisse d'une évolution du langage Visual Basic classique, il n'est pas compatible avec VB6, et tout code écrit dans l'ancienne version ne se compile pas sous VB.NET.

Comme tous les autres langages .NET, VB.NET prend totalement en charge les concepts orientés objet. Tout dans VB.NET est un objet, y compris tous les types primitifs (court, entier, long, chaîne, booléenne, etc.) et les types définis par l'utilisateur et les événements. Tous les objets héritent de la classe de base Object.

VB.NET est implémenté par le .NET framework de Microsoft. Par conséquent, il a un accès complet à toutes les bibliothèques du .Net Framework. Il est également possible d'exécuter des programmes VB.NET sur Mono, l'alternative open source à .NET, non seulement sous Windows, mais même sous Linux ou Mac OSX.

Les raisons suivantes font de VB.Net un langage professionnel largement utilisé :

- Moderne, à usage général.
- Orienté objet.
- Orienté composants.
- Facile à apprendre.
- Langage structuré.
- Il produit des programmes efficaces.
- Il peut être compilé sur une variété de plateformes informatiques.
- Fais partie de .Net Framework.

VB.Net possède de nombreuses fonctionnalités de programmation solides qui le rendent attachant pour une multitude de programmeurs dans le monde. Citons quelques-unes de ces fonctionnalités :

- Conditions booléennes
- Collecte automatique des déchets
- Bibliothèque standard
- Propriétés et événements
- Délégués et gestion des événements
- Génériques faciles à utiliser
- Indexeurs
- Compilation conditionnelle

Dans la méthodologie de programmation orientée objet, un programme se compose de divers objets qui interagissent les uns avec les autres au moyen d'actions. Les actions qu'un objet peut entreprendre sont appelées méthodes. On dit que les objets du même genre ont le même type ou, plus souvent, sont de la même classe.

Lorsque nous considérons un programme VB.Net, il peut être défini comme une collection d'objets qui communiquent en invoquant les méthodes les uns des autres. Voyons maintenant brièvement ce que signifient les classes, les objets, les méthodes et les variables d'instance.

Objet : Les objets ont des états et des comportements. Un objet est une instance d'une classe.

Classe : Une classe peut être définie comme un modèle qui décrit les comportements états pris en charge par les objets de son type.

Méthodes : Une méthode est essentiellement un comportement. Une classe peut contenir de nombreuses méthodes. C'est dans les méthodes où les données sont manipulées et toutes les actions sont exécutées.

Variables d'instance: Chaque objet a son ensemble unique de variables d'instance. L'état d'un objet est créé par les valeurs affectées à ces variables d'instance.

A présent, nous allons présenter les bases du langage VB.Net

III.3.1 Types de données disponibles dans VB.Net :

VB.Net fournit une large gamme de types de données comme : Booléen décimal, chaîne de caractère, double, entier…

III.3.2 Les variables VB.Net :

L'instruction **Dim** est utilisée pour la déclaration de variables et l'allocation de stockage pour une ou plusieurs variables.

L'instruction **Dim** est utilisée au niveau du module, de la classe, de la structure, de la procédure ou du bloc. La syntaxe de déclaration des variables dans VB.Net est :

[< attributelist >] [accessmodifier] [[Shared] [Shadows] | [Static]]
[ReadOnly] Dim [WithEvents] variablelist

Où :

attributelist : est une liste d'attributs qui s'appliquent à la variable. Optionnel.

accessmodifier : définis les niveaux d'accès des variables. Optionnel.

Shared : déclare une variable partagée, qui n'est associée à aucune instance spécifique d'une classe ou d'une structure, plutôt disponible pour toutes les instances de la classe ou de la structure. Optionnel.

Shadows *:* indiquent que la variable redéclare et masque un élément de nom identique, ou un ensemble d'éléments surchargés, dans une classe de base.Optionnel.

Statique *:* indique que la variable conservera sa valeur, même après la fin de la procédure dans laquelle elle est déclarée. Optionnel.

ReadOnly : signifie que la variable peut être lue, mais pas écrite. Optionnel.

WithEvents : spécifie que la variable est utilisée pour répondre aux événements déclenchés par l'instance affectée à la variable. Optionnel.

Variablelist : fournit la liste des variables déclarées.

Chaque variable de la liste des variables a la syntaxe et les parties suivantes :

variablename[([boundslist])] [As [New] datatype] [= initializer]

Où :

variablename : est le nom de la variable

boundslist : il fournit la liste des limites de chaque dimension d'une variable du tableau.

New : il crée une nouvelle instance de la classe lors de l'exécution de l'instruction Dim.

datatype : il spécifie le type de données de la variable.

Initializer : expression évaluée et affectée à la variable lors de sa création.

Voici quelques exemples de déclarations de variables :

```
Dim StudentID As Integer
Dim StudentName As String
Dim Salary As Double
Dim count1, count2 As Integer
Dim status As Boolean
Dim exitButton As New System.Windows.Forms.Button
Dim lastTime, nextTime As Date
```

III.3.3 Les fonctions VB.Net :

L'instruction Function est utilisée pour déclarer le nom, le paramètre et le corps d'une fonction. La syntaxe de l'instruction Function est

```
[Modifiers] Function FunctionName [(ParameterList)] As ReturnType
  [Statements]
End Function
```

Où :

Modifiers : spécifie le niveau d'accès de la fonction. Les valeurs possibles sont : *Public, Private, Protected, Friend, Protected Friend.*

FunctionName : indique le nom de la fonction

ParameterList : spécifie la liste des paramètres

ReturnType : spécifie le type de données de la variable renvoyée par la fonction

Le code suivant montre une fonction *FindMax* qui prend deux valeurs entières et renvoie la plus grande des deux.

```
Function FindMax(ByVal num1 As Integer, ByVal num2 As Integer) As Integer
  ' local variable declaration */
  Dim result As Integer
  If (num1 > num2) Then
    result = num1
  Else
    result = num2
  End If
  FindMax = result
End Function
```

III.3.4 Les classes et objets VB.Net :

Lorsqu'on définit une classe, on définit un plan directeur pour un type de données. Cela ne définit en fait aucune donnée, mais cela définit ce que signifie le nom de la classe, c'est-à-dire en quoi consistera un objet de la classe et quelles opérations peuvent être effectuées sur un tel objet.

Les objets sont des instances d'une classe. Les méthodes et variables qui constituent une classe sont appelées membres de la classe.

Une classe commence par ***Class*** suivi du nom de et le corps de la classe. Elle se termine par l'instruction ***End Class***. Voici la forme générale d'une classe :

[<attributelist>] [accessmodifier] [Shadows] [MustInherit | NotInheritable]
[Partial] _ Class name [(Of typelist)]
    [Inherits classname]
    [Implements interfacenames]
    [statements]
  End Class

Où :

Attributelist : est une liste d'attributs qui s'appliquent à la classe.

Accessmodifier : définis les niveaux d'accès de la classe.

Shadows : indiquent que la variable redéclare et masque un élément de nom identique, ou un ensemble d'éléments surchargés, dans une classe de base.

MustInherit : spécifie que la classe ne peut être utilisée qu'en tant que classe de base et qu'on ne peut pas créer un objet directement à partir d'elle.

NotInheritable : spécifie que la classe ne peut pas être utilisée comme classe de base.

Partial : indique une définition partielle de la classe.

Inherits : spécifie la classe de base dont il hérite.

Implements : spécifie les interfaces dont la classe hérite.

III.4 Introduction au langage jQuery :

jQuery est une bibliothèque JavaScript rapide et concise créée par John Resig en 2006 avec une belle devise : Écrivez moins, faites plus. jQuery simplifie la traversée des documents HTML, la gestion des événements, l'animation et les interactions Ajax pour un développement Web rapide. jQuery est une boîte à outils JavaScript conçue pour simplifier diverses tâches en écrivant moins de code.

Il existe deux façons d'utiliser jQuery :

Installation locale : On peut télécharger la bibliothèque jQuery sur notre machine locale et l'inclure dans votre code HTML.

Exemple :

```
<html>
  <head>
    <title>Exemple jQuery </title>
    <script type = "text/javascript" src = "/jquery/jquery-2.1.3.min.js">
    </script>
    <script type = "text/javascript">
      $(document).ready(function() {
        document.write("Bonjour!");
      });
    </script>
  </head>
```

```
  <body>
    <h1>Bonjour</h1>
  </body>
</html>
```

Version basée sur CDN : On peut aussi inclure la bibliothèque jQuery dans votre code HTML directement depuis Content Delivery Network (CDN).

```
<html>
  <head>
    <title> Exemple jQuery </title>
    <script type = "text/javascript"
      src = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
    </script>
          <script type = "text/javascript">
      $(document).ready(function() {
        document.write("Bonjour!");
      });
    </script>
  </head>
  <body>
    <h1> Bonjour </h1>
  </body>
</html>
```

jQuery est un *framework* construit à l'aide de fonctionnalités JavaScript. Ainsi, on peut utiliser toutes les fonctions et toutes les capacités disponibles en JavaScript. Dans ce qui suit, on va expliquer la plupart des concepts de base les plus fréquemment utilisés dans jQuery.

III.4.1 Syntaxe jQuery :

La syntaxe jQuery est conçue sur mesure pour sélectionner des éléments HTML et effectuer des actions sur les éléments.

La syntaxe de base est: $ (sélecteur).action ()

Un signe $ pour définir / accéder à jQuery

Un (sélecteur) pour "interroger (ou trouver)" des éléments HTML

Une action () jQuery à effectuer sur le ou les éléments.

Exemples:

$ (this) .hide () - masque l'élément courant.

$ ("p"). hide () - masque tous les éléments <p>.

$ (". test"). hide () - masque tous les éléments avec class = "test".

$ ("# test"). hide () - masque l'élément avec id = "test".

III.4.2 Les sélecteurs jQuery :

Les sélecteurs jQuery permettent de sélectionner et de manipuler des éléments HTML.

Les sélecteurs jQuery sont utilisés pour sélectionner des éléments HTML en fonction de leur nom, id, classes, types, attributs, valeurs d'attributs et bien plus encore. Il est basé sur les sélecteurs CSS existants, et en plus, il a ses propres sélecteurs personnalisés.

Tous les sélecteurs de jQuery commencent par le signe dollar et les parenthèses : $ ().

Le sélecteur d'élément jQuery sélectionne les éléments en fonction de leur nom. On peut sélectionner tous les éléments <p> sur une page de la façon suivante :

$ ("p")

Exemple :

Le script suivant a pour objectif de masquer tous les éléments <p> lorsqu'un utilisateur clique sur un bouton:

```
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide();
  });
});
```

III.4.3 Les méthodes d'événement jQuery:

Toutes les actions des différents visiteurs auxquelles une page Web peut répondre sont appelées événements. Un événement représente le moment précis où quelque chose se passe.

Exemples:

- Déplacer une souris sur un élément
- Sélection d'un bouton radio
- Cliquer sur un élément

Dans jQuery, la plupart des événements DOM ont une méthode jQuery équivalente. Pour affecter un événement de clic à tous les paragraphes d'une page, on doit procéder comme suit:

```
$("p").click();
```

L'étape suivante consiste à définir ce qui doit se produire lorsque l'événement se déclenche. Vous devez transmettre une fonction à l'événement :

```
$("p").click(function(){
  // l'action });
```

Le tableau suivant résume les méthodes d'événement jQuery couramment utilisées :

La méthode d'événement jQuery	***Signification***
$(document).ready()	permet d'exécuter une fonction lorsque le document est entièrement chargé
click()	attache une fonction de gestionnaire d'événements à un élément HTML. La fonction est exécutée lorsque l'utilisateur clique sur l'élément HTML.
dblclick()	attache une fonction de gestionnaire d'événements à un élément HTML. La fonction est exécutée lorsque l'utilisateur clique deux fois sur l'élément HTML.
mouseenter()	attache une fonction de gestionnaire d'événements à un élément HTML. La fonction est exécutée lorsque le pointeur de la souris entre dans l'élément HTML:
focus()	attache une fonction de gestionnaire d'événements à un champ de formulaire HTML. La fonction est exécutée lorsque le champ du formulaire obtient le focus.
blur()	attache une fonction de gestionnaire d'événements à un champ de formulaire HTML. La fonction est exécutée lorsque le champ du formulaire perd le focus.
on()	attache un ou plusieurs gestionnaires d'événements pour les éléments sélectionnés.

Tableau III.1: les méthodes d'événement jQuery

Chapitre IV

Les langages de script côté serveur

IV.1 Introduction

En plus des langages client que nous avons illustrés dans les chapitres précédents (HTML, CSS et JavaScript), il existe plusieurs langages de programmation destinés uniquement aux serveurs.

Les langages serveur sont des langages produits et gérés par les serveurs eux même. En conséquence, les clients n'ont pas à accéder à ces codes mais ils auront uniquement le résultat de l'exécution de ces codes.

Les langages client définissent la façon d'affichage des sites web par contre les langages serveur décrivent le comportement du site Web. Par exemple, avec un langage serveur, on peut définir un menu qui ne doit s'afficher que si l'utilisateur a créé un compte sur mon site.

Les langages serveur sont nombreux comme : PHP, Java, Python, Ruby, C#... Dans ce qui suit, nous allons illustrer un langage serveur en occurrence PHP ainsi que deux *frameworks* ASP et JSP. On va aussi évoquer quelques notions concernant la manipulation des bases de données.

On peut résumer l'interaction entre le client et le serveur comme ceci :

1. Le client demande une page au serveur.
2. Le serveur génère la page en utilisant le langage serveur comme le PHP.

3. Le serveur envoie la page proclamée (sous forme de code HTML et CSS) .

4. Le navigateur du client affiche les codes HTML,CSS, JavaScript et exécute le fichier exécutable du code PHP. Voici un schéma qui résume ces étapes :

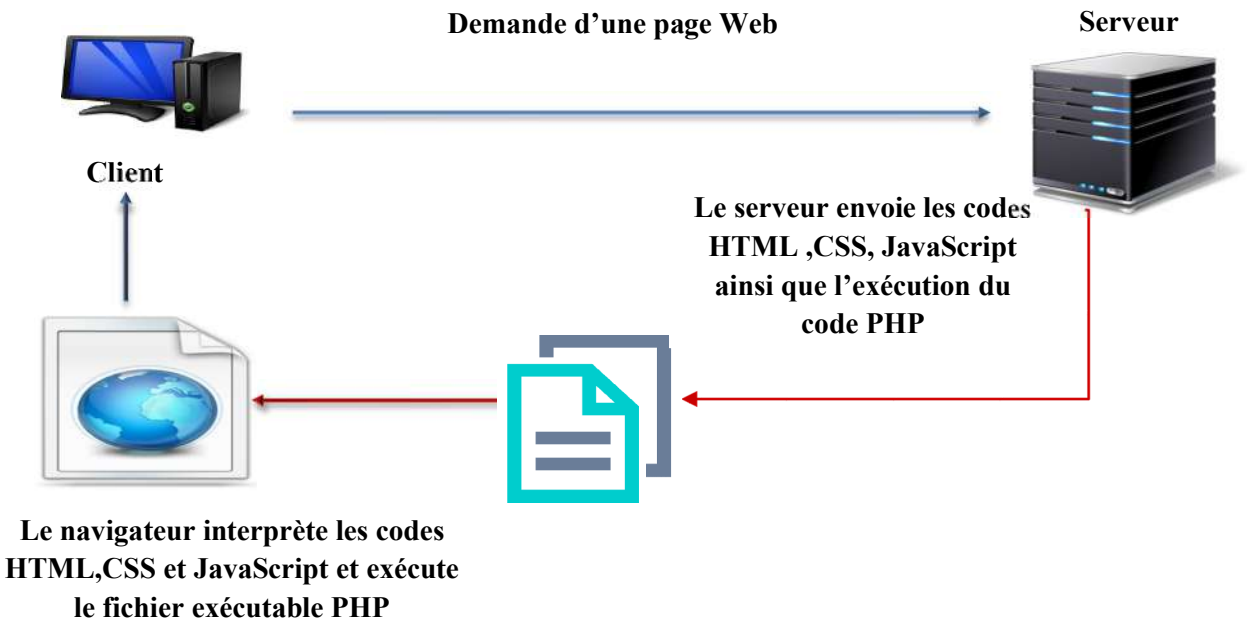


Figure IV.1. Principe du langage coté serveur

IV.2 Le langage PHP (*Hypertext Preprocessor*)

PHP est un langage que seuls les serveurs comprennent et qui permet de rendre votre site dynamique. C'est PHP qui génère la page web qui sera exécutée côté client. PHP est sensible à la case et il s'appuie sur des fonctions prédéfinies pour effectuer ces tâches. Pour pouvoir utiliser du PHP dans HTML, on va devoir introduire une nouvelle balise qui commence par **<?php** et se termine par **?>**.

Voici une balise PHP vide :

<?php ?>

À l'intérieur, on écrira donc du code source PHP :

<?php /* Le code PHP se met ici */ ?>

On peut sans problème écrire la balise PHP sur plusieurs lignes. En fait, c'est même indispensable car la plupart du temps le code PHP fera plusieurs lignes. Cela donnera quelque chose comme :

```
<?php
/* Le code PHP se met ici
Et ici
Et encore ici */
?>
```

Il existe d'autres balises pour utiliser du PHP, par exemple<? ?>,<% %>, etc. Ne soyez donc pas étonnés si vous en voyez. Néanmoins,<?php ?>est la forme la plus correcte, vous apprendrez donc à vous servir de cette balise et non pas des autres.

La balise PHP que nous venons de découvrir s'insère au milieu du code HTML de la façon suivante:

```
<!DOCTYPE html>
<html>
   <head>
      <title>Document  PHP</title>
      <meta charset="utf-8" />
   </head>
   <body>
      <h2>Page de test</h2>

      <p>
         Cette page contient du code HTML avec des balises PHP.<br />
         <?php /* Insérer du code PHP ici */ ?>
         Voici quelques petits tests :
      </p>

      <ul>
      <li style="color: blue;">Texte en bleu</li>
      <li style="color: red;">Texte en rouge</li>
      <li style="color: green;">Texte en vert</li>
```

```
        </ul>

        <?php
        /* Encore du PHP
        Toujours du PHP */
        ?>
    </body>
</html>
```

Une balise PHP peut être placée n'importe où dans le code HTML et pas seulement dans le corps de la page. Voici un exemple ou le code PHP est placé dans l'entête du code HTML :

```
<!DOCTYPE html>
<html>
    <head>
        <title>Ceci est une page de test <?php /* Code PHP */ ?></title>
        <meta charset="utf-8" />
    </head>
```

On peut même insérer une balise PHP au milieu d'une balise HTML. L'exemple suivant explique cette opération :

```
<!DOCTYPE html>
<html>
    <head>
        <title>Ceci est une page de test</title>
        <meta <?php /* Code PHP */ ?> charset="utf-8" />
    </head>
```

Dans ce qui va suivre, nous allons faire un tour d'horizon des principales fonctionnalités de PHP.

IV.2.1. Les variables

PHP admet tous types variable (entiers, flottants, chaînes de caractères, tableaux, booléens, objets, ressource ou NULL).Dans PHP, chaque variable commence toujours par le caractère dollar ($) suivi du nom de la variable. Ces noms de variable sont soumis à un ensemble de règles comme suit :

- Le nom doit commencer par un caractère alphabétique ou bien un caractère de soulignement (_) suivi des mêmes caractères plus des chiffres. La longueur du nom n'est soumise à aucune limitation.
- La déclaration des variables n'est pas obligatoire en début de script. On peut créer des variables n'importe où, à condition de les créer avant de les utiliser.
- L'initialisation des variables n'est pas obligatoire
- Les noms des variables sont sensibles à la casse (majuscule et minuscule).

Voici un exemple de variable valide sous PHP :

```
$var
$_var
$var2
$T1
$_123
```

L'affectation d'une valeur à une variable consiste à déterminer le type de la variable. Cette affectation peut se faire de deux façons : soit par valeur ou par référence.

L'affectation par valeur s'effectue par l'opérateur =, soit après la création de la variable, soit en même temps. Voici quelques exemples d'affectation par valeurs :

```
$var=17;
$var="Annaba";
$var=5*3+6/7-81%9; //PHP évalue l'expression puis affecte le résultat
$var=php_connect($a,$b,$c); //la fonction retourne une ressource
$var=isset($var&&($var==9)); //la fonction retourne une valeur booléenne
```

L'affectation par référence, est aussi effectuée au moyen de l'opérateur =, l'opérande de droite est une variable qui doit être précédée du caractère **&**. Voici un exemple de ce type d'affectation :

$var2 = &$var1;

PHP est aussi doté d'un grand nombre de variables prédéfinies. Le tableau suivant résume ceux les plus utilisées :

Variable prédéfinie	Signification
$GLOBALS	Contiens le nom et la valeur de toutes les variables globales du script
$_COOKIE	Contiens le nom et la valeur des cookies enregistrés sur le poste client
$_ENV	Contiens le nom et la valeur des variables d'environnement
$_FILES	Contiens le nom des fichiers téléchargés à partir du poste client
$_GET	Contient le nom et la valeur des données issues d'un formulaire envoyé par la méthode GET
$_POST	Contiens le nom et la valeur des données issues d'un formulaire envoyé par la méthode POST
$_REQUEST	Contiens l'ensemble des variables superglobales $_GET, $_POST, $_COOKIE et $_FILES.
$_SERVER	Contiens les informations liées au serveur web, tel le contenu des en-têtes HTTP ou le nom du script en cours d'exécution
$_SESSION	Contiens l'ensemble des noms des variables de session et leurs valeurs.

Tableau IV.1: Les variables prédéfinies PHP.

IV.2.2. Les opérateurs d'affectation combinée

Dans PHP, il existe plusieurs opérateurs d'affectation combinée. Ces opérateurs réalisent deux opérations à la fois. Le tableau suivant décrit l'ensemble de ces opérateurs.

L'opération	Définition
+ =	Addition puis affectation
- =	Soustraction puis affectation
* =	Multiplication puis affectation
** =	Puissance puis affectation
/=	Division puis affectation
% =	Modulo puis affectation
. =	Concaténation puis affectation

Tableau IV.2: Les variables prédéfinies PHP.

IV.2.3. Les constantes :

Pour définir des constantes personnalisées sous PHP , la fonction define() est utilisée de la manière suivante :

boolean define(string nom_cte, divers valeur_cte, boolean casse)

Voici un exemple de création de la constante PI avec PHP :

```
<?php
define("PI",3.1415926535,TRUE);
? >
```

PHP dispose d'un ensemble de constantes prédéfinies utilisables dans tous les scripts. Le tableau suivant cite quelques constantes prédéfinies :

Constante prédéfinie	Définition
PHP_VERSION	Version de PHP installée sur le serveur
PHP_OS	Nom du système d'exploitation du serveur
DEFAULT_INCLUDE_PATH	Chemin d'accès aux fichiers par défaut
__FILE__	Nom du fichier en cours d'exécution
__LINE__	Numéro de la ligne en cours d'exécution

Tableau IV.3: Quelques constantes prédéfinies PHP.

IV.2.4. Les Fonctions :

PHP propose des fonctions prédéfinies que l'on peut utiliser immédiatement. Pour autant, on peut aussi créer nos propres fonctions selon nos besoins. On peut se servir des fonctions PHP prédéfinies pour créer nos propres fonctions. Le code suivant montre un exemple de fonctions PHP prédéfinies :

```
<?php
echo phpversion();
?>
```

Une fonction doit commencer obligatoirement par le mot clé 'function'. Une fonction peut ne pas retourner une valeur. Voici un exemple de fonction affichant le message "bonjour ":

```
<?php
function salut()
 {
  echo "Bonjour tout le monde !";
 }
?>
```

L'exemple précédent est une fonction qui ne contient aucun paramètre. Une fonction peut avoir un ou plusieurs paramètres. Ces paramètres sont généralement des variables et peuvent être de types différents. Après la définition du nom de la fonction, on doit déclarer tous les paramètres de la fonction en les séparant par des virgules.

L'exemple suivant calcule la division de deux nombres. On met les deux nombres en paramètres. Le résultat de cette fonction est la division des deux nombres. Le mot clé 'return' permet de retourner le résultat.

```
<?php
function division($nombre1, $nombre2)
{
  $resultat=$nombre1/$nombre2;
  return $resultat;
}
?>
```

Pour appeler une fonction, il suffit d'indiquer son nom et d'introduire les valeurs des paramètres. Voici comment appeler la fonction division de l'exemple précédent:

```
$valeur=division(100,50);
echo $valeur;
//ou encore
$numerateur=100;
$denominateur=50;
$quotien= division($numerateur, $denominateur);
echo $quotien;
?>
```

IV.2.5. Les tableaux :

Un tableau sous PHP peut être de deux types, soit un tableau numéroté ou bien tableau associatif.

Pour créer un tableau numéroté en PHP, on utilise la fonction PHP prédéfinie ***array***. Voici un exemple qui montre comment créer ce type de tableau :

```
<?php
$prenoms = array ('Toufik', 'Mourad', 'Mahmoud', 'Ali', 'Brahim');
?>
```

Dans ce type de tableau, l'ordre a une grande importance. Le premier élément (Toufik) aura le n°0, ensuite Mourad le n°1, etc. Cet exemple donne le résultat suivant à l'exécution :

0	Toufik
1	Mourad
2	Mahmoud
3	Ali
4	Brahim

Le tableau précédent peut être créé d'une façon manuelle (case par case) :

```
<?php
$prenoms[0] = 'Toufik';
$prenoms[1] = 'Mourad';
$prenoms[2] = 'Mahmoud';
$prenoms[3] = 'Ali';
$prenoms[2] = 'Brahim';
?>
```

Les numéros du tableau peuvent être affectés automatiquement par PHP en laissant les crochets vides :

```
<?php
$prenoms[] = 'Toufik';
$prenoms[] = 'Mourad';
$prenoms[] = 'Mahmoud';
?>
```

Pour afficher un élément, on doit indiquer sa position entre crochets après le terme $prenoms. Pour afficher « Mourad », on doit donc écrire :

```
<?php
echo $prenoms[1];
?>
```

IV.2.6. Les boucles :

Les boucles sont des éléments indispensables dans un langage de programmation. Elles permettent de répéter plusieurs fois une même opération, tant qu'une condition est remplie ou bien jusqu'à ce qu'elle soit remplie. Comme dans tous les langages de programmation, PHP gère les structures de boucle **for** et **while** mais aussi les déclinaisons **foreach** et **do**.

- **La boucle For :**

En PHP, la boucle **for** s'utilise avec la syntaxe suivante :

```
<?php
for (initialisation) {
   Condition à vérifier ;
}
?>
```

Voici un exemple qui affiche 10 fois la phrase ‘Bienvenu au cours technologies web ' et affiche aussi les chiffre de 5 à 10:

```
<?php
for ($i=0; $i<10; $i++) {
   echo ' Bienvenu au cours technologies web ';
}
```

```
for ($i=5; $i<11; $i++) {
    echo $i;
}
?>
```

- **La boucle While :**

Une boucle **while** s'écrit de la façon suivante:

```
<?php
while ($condition) {
    // instructions
}
?>
```

L'exemple d'affichage des chiffres de 5 à 10 peut alors s'écrire comme suit :

```
<?php
$i = 5;
while ($i < 11) {
    echo $i;
    $i++;
}
?>
```

- **La boucle Foreach :**

Cette boucle est utilisable pour parcourir un tableau (ou les membres d'un objet). Il est ainsi possible d'afficher l'ensemble des éléments d'un tableau et les clés associées de la façon suivante:

```
<?php
foreach ($tableau as $cle => $valeur) {
    echo $cle.' - '.$valeur.'<br />'."\n";
```

```
}
?>
```

Si l'on n'a pas besoin de connaitre les clés, le code devient :

```
<?php
foreach ($tableau as $valeur) {
   echo $valeur.'<br />'."\n";
}
?>
```

- **La boucle Do ... while :**

L'instruction **do** est associée au terme **while**. Elle est similaire à la boucle **while** à ceci près que les instructions sont exécutées au moins une fois, la condition n'étant testée pour la première fois qu'après exécution des instructions. Voici l'exemple d'affichage des chiffres de 5 à 10 vu précédemment établi avec cette boucle :

```
<?php
$i = 5;
do {
   echo $i;
   $i++;
} while ($i<11);
?>
```

IV.2.7. manipulation des bases de données avec MySql:

Une base de données un ensemble de données stockées dans des fichiers particuliers et d'une façon hiérarchique. Elle nous permettre de stocker une grande quantité de données que nous pouvons par la suite les manipuler.

Le langage utilisé est le SQL (*Structured Query Language*) qui est un langage de requêtes qui permettre d'accéder et de manipuler les bases de données. Pour faire ces opérations, on utilise un système de gestion de bases de données comme le MySQL qui va nous permettre d'envoyer des requêtes SQL. A la manière des expressions régulières, nous allons pouvoir l'utiliser avec le PHP.

Il existe deux façons pour effectuer des requêtes SQL par le biais de MySQL: soit dans nos fichiers de code PHP, soit en passant par l'interface *phpMyAdmin* qui est un logiciel également codé en PHP.

Voici un exemple qui illustre la connexion, l'exécution d'une requête, la lecture d'informations obtenues et la déconnexion d'une base de données MySQL.

```
<?php
// Connexion et sélection de la base
$link = mysql_connect('mysql_host', 'mysql_user', 'mysql_password')
   or die('Impossible de se connecter : ' . mysql_error());
echo 'Connected successfully';
mysql_select_db('my_database') or die('Impossible de sélectionner la base
de données');

// Exécution des requêtes SQL
$query = 'SELECT * FROM my_table';
$result = mysql_query($query) or die('Échec de la requête : ' .
mysql_error());

// Affichage des résultats en HTML
echo "<table>\n";
while ($line = mysql_fetch_array($result, MYSQL_ASSOC)) {
   echo "\t<tr>\n";
   foreach ($line as $col_value) {
```

```
            echo "\t\t<td>$col_value</td>\n";
        }
        echo "\t</tr>\n";
    }
    echo "</table>\n";
    // Libération des résultats
    mysql_free_result($result);
    // Fermeture de la connexion
    mysql_close($link);
    ?>
```

La deuxième façon consiste à utiliser *phpMyAdmin* afin d'effectuer des actions manuelles directes sur nos bases de données. L'accueil de *phpMyAdmin* ressemble à la figure suivante :

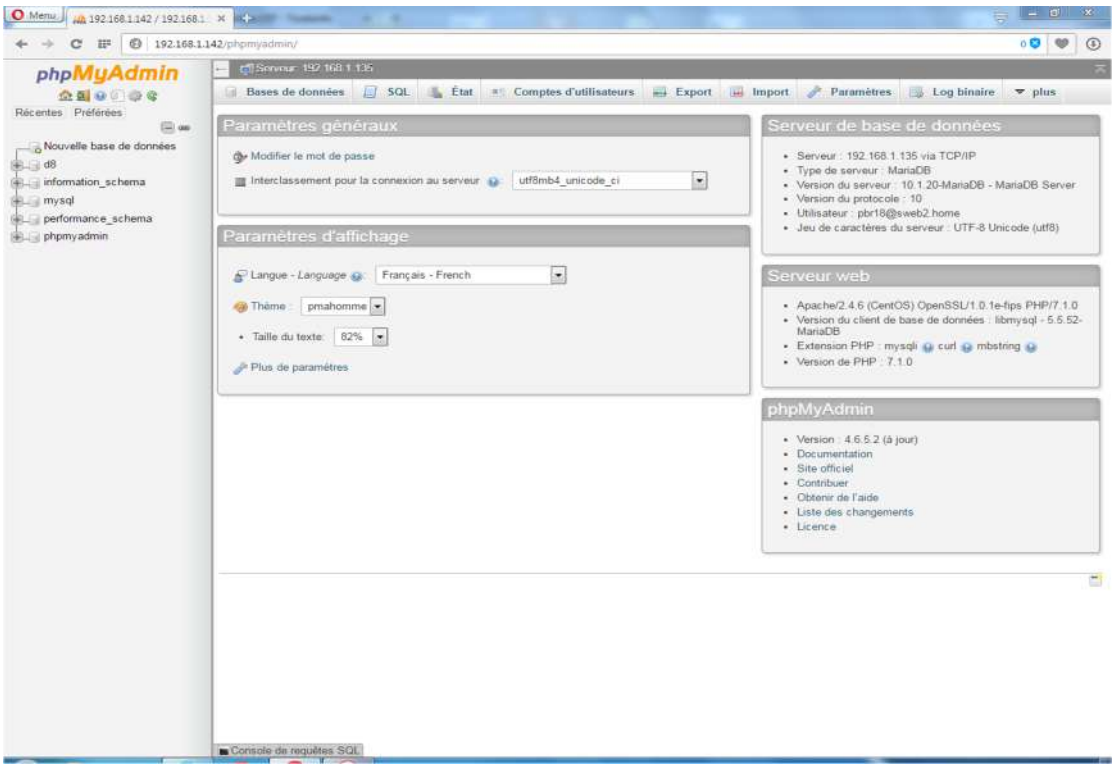


Figure IV.2. Accueil de *phpMyAdmin*

On peut remarquer deux zones importantes :

- **La liste des bases de données :** c'est la liste de nos bases de données déjà créée.
- **Créer une base de données :** pour créer une nouvelle base de données, on doit préciser un nom dans le champ du formulaire à droite

A présent, nous allons créer une nouvelle base de données appelée test. La figure suivante nous indique que la base de données a bien été créée.

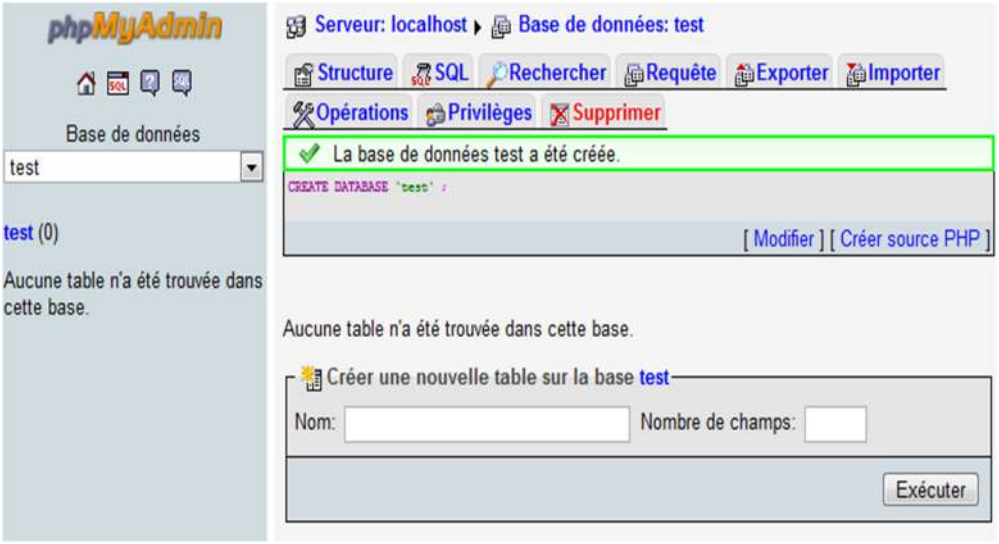


Figure IV.3. La base test a été créée avec succès.

Dans cette base de données, aucune table n'est créée. Pour en créer une table, on va se diriger vers le champ « Créer une nouvelle table sur la base test », comme vous le montre la figure suivante :

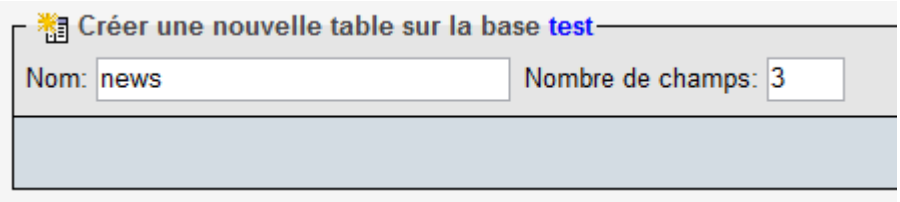


Figure IV.4. Création d'une table

L'étape suivante consiste à indiquer le nom des champs ainsi que les données qu'ils peuvent contenir. Pour cette table, on va créer les trois champs : id, titre et contenu. La figure suivante montre la création d'une table MySQL :

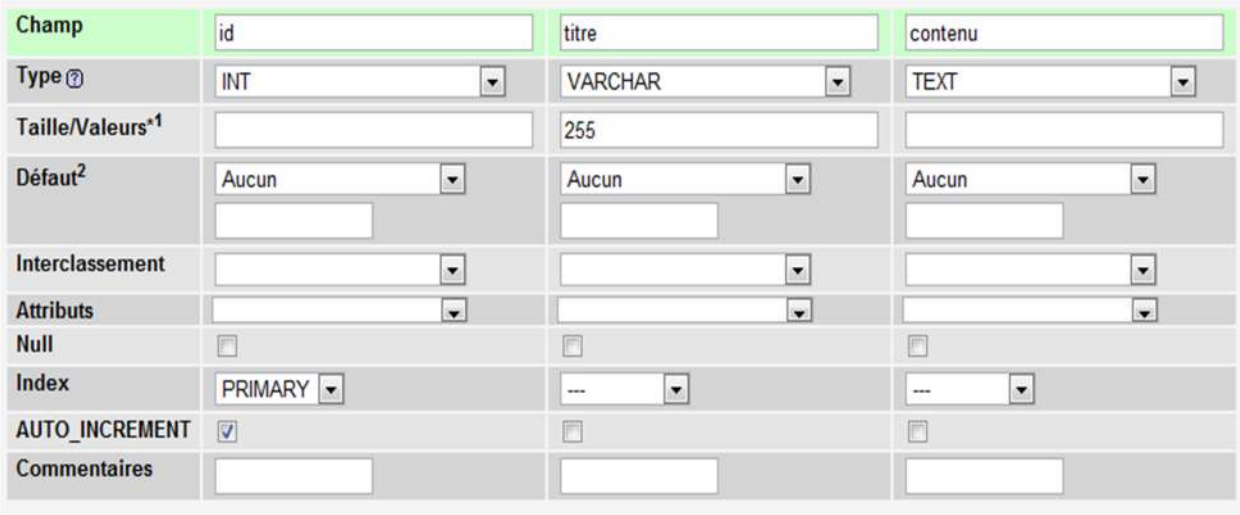


Figure IV.5. Détails d'une table MySQL

Chaque colonne représente un champ. Nous avons demandé trois champs, il y a donc trois colonnes. *phpMyAdmin* nous exige le remplissage de plusieurs champs,mais réellement il n'est pas nécessaire de tout remplir. Les sections les plus significatives sont :

- **Champ :** permets de définir le nom du champ
- **Type :** le type de données que va stocker le champ (nombre entier, texte, date…)
- **Taille/Valeurs :** permets d'indiquer la taille maximale du champ
- **Index :** active l'indexation du champ.
- **AUTO_INCREMENT :** permets au champ de s'incrémenter tout seul à chaque nouvelle entrée.

Contrairement à PHP, MySQL propose énormément de types de données. Voici quelques catégories de données sous MySQL :

- **NUMERIC :** c'est la catégorie des nombres.

- **DATE and TIME :** c'est les dates et les heures.
- **STRING :** c'est la catégorie des chaînes de caractères.
- **SPATIAL :** cela concerne les bases de données spatiales.

En fait, *phpMyAdmin* propose une liste les quatre types de données les plus courants :

- **INT :** nombre entier
- **VARCHAR :** texte court (entre 1 et 255 caractères) **.**
- **TEXT :** long texte.
- **DATE :** date (jour, mois, année)**.**

IV.2.8. Exercice d'application :

1. Écrivez un tableau multidimensionnel associatif dont les clés sont des noms de personne et les valeurs des tableaux indicés contenant le prénom, la ville de résidence et l'âge de la personne.
2. Créer un formulaire permettant à l'utilisateur de saisir le pseudonyme à rechercher afin de faciliter la saisie pour l'utilisateur.

Solution :

1.
```
<?php
$tab=array("BILEL"=>array("toufik","annaba",67),"MOHAMED"=>
array("SAMIR","Alger",35),"salim"=>array("hacene","oran",45));
print_r($tab);
?>
```

2.
```
<h1> Veuillez saisir un pseudonyme : </h1>
<form method="get" action="psd.php">
<p> Pseudonyme : <input type="text" name="pseudo" value="<?php
if(isset($_GET['pseudo']))
echo $_GET['pseudo'];?>"/>
```

```
<input type="submit" value="Recherche" /> </p>
</form>
    <?php
    $personnes = array(...);
    $trouve= false ;
    if( ( isset($_GET['pseudo']) ) and ( trim($_GET['pseudo']) !='') ) {
    $pseudo = $_GET['pseudo'];
    foreach($personnes as $p => $info)
    if($p==$pseudo) {
    echo '<p> Bonjour ' . $info['prenom'] . ' ' . $info['nom'] .'. Vous avez '.
    $info['age'] . ' et vous habitez ' . $info['ville'] . '. </p>'."\n";
    $trouve=true; }
    }
    if($trouve== false ) {
    echo '<p> Désolé, votre pseudo n\'apparaît pas dans la liste! </p>'."\n";
    ?>
```

IV.3. Les *frameworks*:

En plus de ces langages de programmation, ils existent des *frameworks* qui ont pour but de faciliter la création de sites web dans ces langages.

Les frameworks sont des boîtes à outils additionnelles au langage qui se révèlent de plus en plus nécessaire de jour en. Ils ont le rôle de facilité la création d'un site web en donnant un coup de main au langage de programmation et en conséquence au créateur u site Web.

Les frameworks se rapportent à des langages. On peut citer :

- Pour PHP : Symfony, Zend...

- Pour Java : Java EE (ou J2EE), JSP
- Pour Python : Django
- Pour Ruby : Ruby on Rails
- Pour C# : ASP .NET

Dans ce qui va suivre, on va introduire deux de ces *frameworks* en occurrence ASP .NET et JSP.

IV.3.1. ASP .NET:

ASP.NET est une plateforme de développement Web créé par Microsoft qui permet la réalisation d'applications web.

ASP.NET se compose en deux grandes parties, à savoir : **ASP.NET *WebForms*** et ***ASP.NET MVC***. *WebForms* et *MVC* sont deux logiques différentes de développement au-dessus du *framework* ASP.NET.

ASP.NET WebForms a été créé en 2002 par Microsoft afin que les développeurs d'applications Windows puissent facilement créer des applications web à partir de leurs connaissances de l'environnement de développement Windows.

Ainsi, ASP.NET *WebForms* dispose d'un ensemble de mécanismes permettant d'abstraire le plus possible le modèle web afin de rendre les développeurs dans les conditions de développement d'une application Windows.

En conséquence, les applications ASP.NET *WebForms* se rapprochent d'un modèle événementiel. Chaque composant d'une page peut réagir à une action de l'utilisateur du site Web. Le modèle ASP.NET WebForms permet aussi de conserver l'état d'une page Web à l'opposé du protocole HTTP.

ASP.NET MVC (*Model View Controller*) a été créé en 2009. ASP.NET MVC offre un cadre de réalisation des applications web et un ensemble de lignes directrices qui vont guider l'utilisateur dans la réalisation d'une application web. ASP.NET MVC propose plus de maîtrise et de liberté sur HTML en permettant d'utiliser des bibliothèques Javascript externes. **V**oici un schéma illustratif du principe de base de ASP.NET MVC :

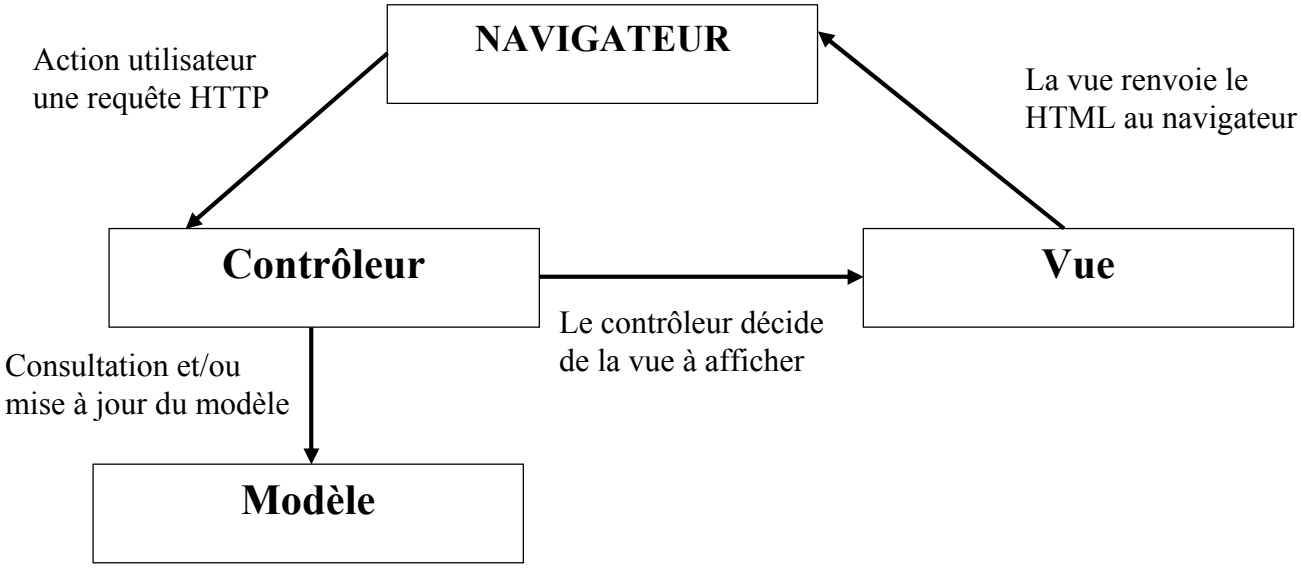


Figure IV.6. Principe de base ASP.NET MVC

ASP.NET MVC se base trois composants principaux : modèle, vue et contrôleur.

Modèle : les modèles reflètent nos objets, et sont des moyens de passer des données entre les contrôleurs et les vues.

Vue : les vues sont les pages qui rendent et montrent les données modèles à l'utilisateur. Des vues d'ASP.NET MVC sont typiquement écrites selon la syntaxe *Razor*.

Contrôleur : les contrôleurs traitent des demandes entrantes d'un client HTTP, et renvoient habituellement un ou plusieurs modèles à une vue appropriée.

Nous allons maintenant détailler ce schéma par l'illustration d'un exemple pratique sous l'environnement de développement Visual Studio de MICROSOFT. La première étape consiste à créer un nouveau projet sur une fenêtre qui ressemble à la figure suivante :

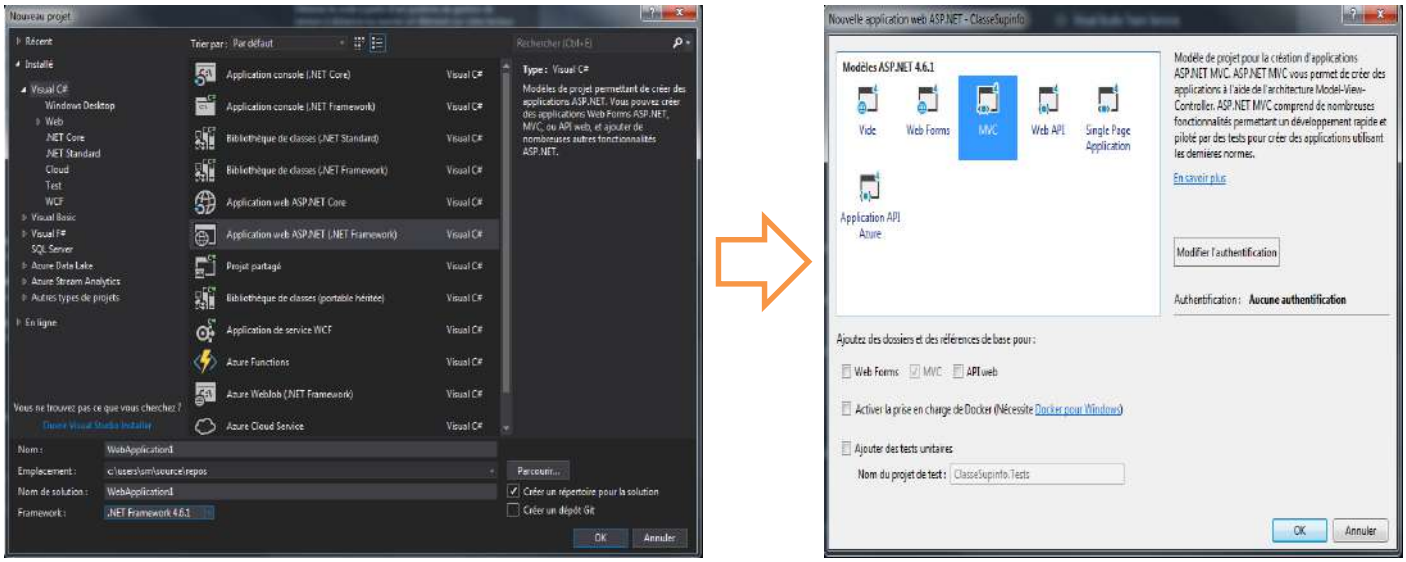


Figure IV.7. Création nouveau projet ASP.NET MVC

Pour cette application, nous allons créer une page qui permettra de lister les informations d'une personne. Il nous faudra donc un modèle avec les informations des personnes, un contrôleur pour instancier nos données et créer des actions, puis une vue qui permettra d'afficher ces informations.

Tout d'abord, nous allons créer le modèle, créez donc une nouvelle classe à l'intérieur de notre répertoire "Models" nommé Personne, et insérez-y ce code :

```
public class Person
{
public string Surname { get; set; }
public string FirstName { get; set; }
public string Patronymic { get; set; }
public DateTime BirthDate { get; set; }
}
```

Puis, on doit ajouter un nouveau contrôleur dans l'onglet "Controllers" :

```
public class HomeController : Controller

{
//Action Method
public ActionResult Index()
{
// Initialize model
Person person = new Person
{
Surname = "Person_SURNAME",
FirstName = "Person_FIRSTNAME",
Patronymic = "Person_PATRONYMIC",
BirthDate = new DateTime(1990, 1, 1)
};

}
}
```

Enfin, on va ajouter la vue à /Views/Home/ nommé Index.cshtml selon le code suivant :

```
@* Model de vue Person *@
<h2>Bonjour @Model.FirstName !</h2>
<div>
<h5>Details :</h5>
<div>
@Html.LabelFor(m => m.Surname)
@Html.DisplayFor(m => m.Surname)
</div>
<div>
@Html.LabelFor(m => m.FirstName)
@Html.DisplayFor(m => m.FirstName)
</div>
<div>
```

```
@Html.LabelFor(m => m.Patronymic)
@Html.DisplayFor(m => m.Patronymic)
</div>
<div>
@Html.LabelFor(m => m.BirthDate)
@Html.DisplayFor(m => m.BirthDate)
</div>
</div>
```

IV.3.2. JSP :

JavaServer Pages (JSP) est une technologie standard de Java qui permet d'écrire des pages dynamiques pour les applications Web de Java. JSP est un langage script côté serveur. Le code est exécuté par le serveur web. Les pages JSP sont des fichiers qui ont l'extension .jsp . Les fichiers JSP contiennent un mélange entre le code HTML ainsi que du code imbriqué en Java. Ce mélange les pages JSP permet de séparer la présentation générale de la page (en HTML) des aspects procéduraux ce qui va offrir une grande souplesse dans la création de sites web.

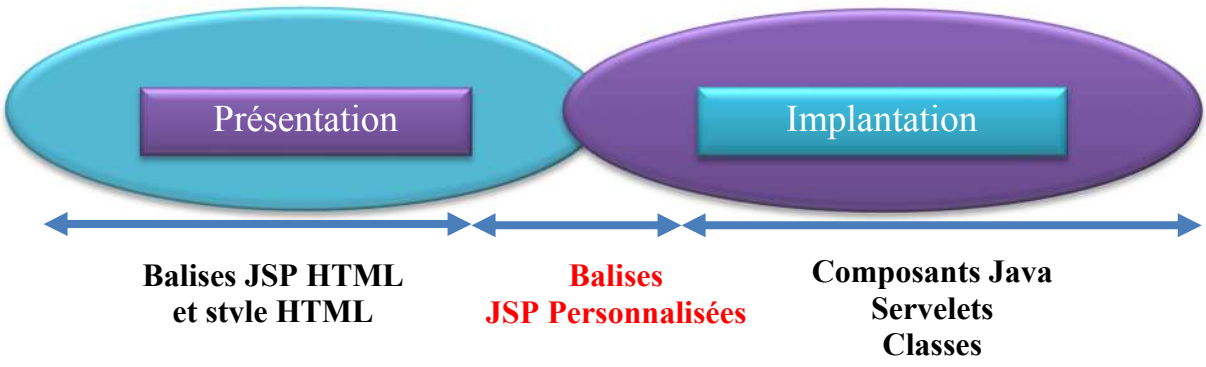


Figure IV.8. Principe de base du JSP.

JSP est construit sur les spécifications de Java Servlet. Un servlet est un composant qui étend les fonctionnalités d'un serveur web de manière portable et efficace. Ces deux technologies fonctionnent typiquement ensemble, et plus particulièrement dans les applications Web les plus anciennes de Java.

JSP présente énormément d'avantages par rapport aux autres langages de développement Web. JSP profite de toutes les modalités offertes par Java pour le développement et les déploiements d'applications Web. En effet Java est un langage orienté objet fortement typé, permettant l'encapsulation, le traitement des exceptions et la gestion automatique de la mémoire. En effet, JSP utilise l'ensemble de la plateforme JAVA sous-jacente, par conséquent, les JSP peuvent directement tirer avantage de toutes les API Java. A présent, on va faire un tour d'horizon sur la syntaxe de base de JSP. Voici un premier exemple d'un code JSP insérer dans une page HTML :

```
<html>
<body>
<% String visiteur = request.getParameter("name");
if (visitor == null) visitor = " World"; %>
Hello, <%= visitor %>! </body> </html>
```

Ce code représente une page JSP qui génère dynamiquement un contenu. Il déclare une variable Java de type String appelée **visiteur** et essaie ensuite de l'initialiser à partir de la requête HTTP courante. Si aucune valeur n'est présente pour cette variable dans la requête, une valeur par défaut lui est attribuée. Une expression JSP est ensuite utilisée pour insérer la valeur de cette variable dans la sortie HTML de la page. JSP propose trois catégories de balises :

- **Les balises de directives**

Ces balises consistent à transmettre des informations de traitement spécifiques à une page au conteneur de JSP. Ces balises auront la forme syntaxique suivante :

<%@ page attribut1="valeur1" attribut2="valeur2" %>

- **Les éléments script**

Ces balises sont conçues pour intégrer les instructions de programmation, écrites dans le langage script destiné pour la page Web. Ces instructions doivent être exécutées chaque fois que la page est traitée pour une requête donnée. On peut citer :

- **Les déclarations :** Les déclarations sont utilisées pour définir des variables et des méthodes spécifiques à la page JSP.

```
<%%! String Chaine = "hello"; int num = 10;
public void exemple() {
// instructions;
}
%>
```

- **Les expressions**

Ces balises consistent à ajouter des variables et des méthodes à une page JSP. L'élément ***expression*** des JSP est conçu pour la génération des sorties. Sa syntaxe est :

```
<%= expression %>
```

- **Les sciptlets**

Les scriptlets sont des instructions d'un langage de script quelconque qui, comme dans le cas des déclarations, sont uniquement évaluées pour leur effet de bord. Cependant elles n'ajoutent pas automatiquement des contenus aux sorties des pages JSP. La syntaxe est :

```
<% scriptlet %>
```

- **Les balises d'actions**

Elles permettent de réaliser plusieurs comportements différents. Elles peuvent transférer le contrôle entre les pages, spécifier des applets et interagir avec des composants JavaBean côté serveur.

➢ **Exécution des JSP :**

Lorsque la page est demandée par un utilisateur avec le protocole HTTP, le serveur web HTTP passer sa requête à un moteur JSP qui va interpréter le page, compiler le code et générer la réponse. De son côté, l'utilisateur ne verra pas apparaitre du code Java dans la page Web reçu. La figure suivante illustre les étapes d'une demande de page JSP:

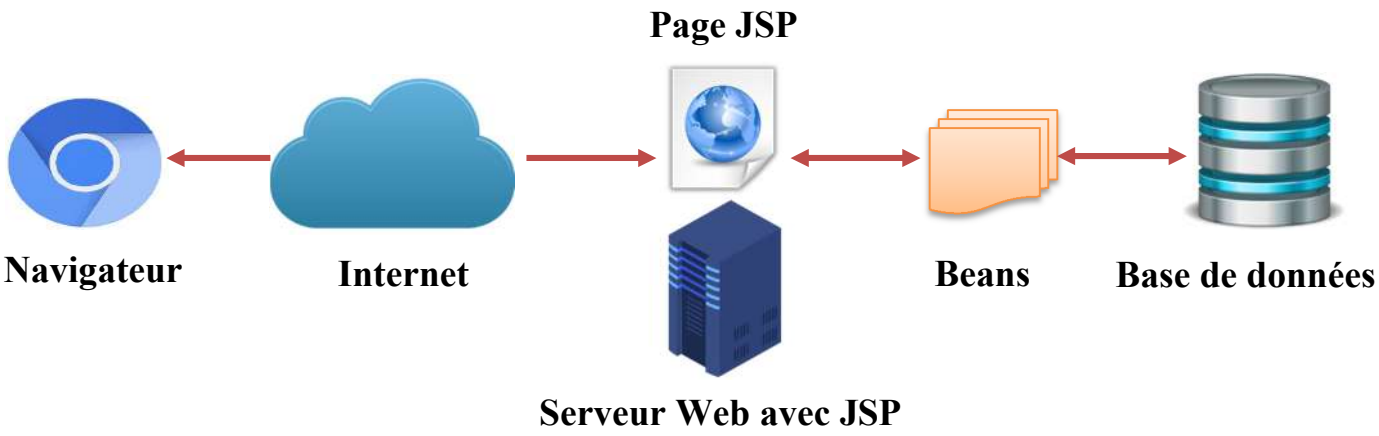


Figure IV.9. Etapes d'une demande de page JSP

JSP définit la notion de *beans* associé à une page. Un *bean* est un objet Java simple qui se comporte comme une variable que l'on peut utiliser partout de cette page. Un *bean* possède les propriétés suivantes :

- Il doit implémenter en série
- Il doit posséder un constructeur vide
- Il doit exposer des propriétés, sous forme de paires *getters / setters* .

Le *bean* peut être de deux type : soit un *bean* existant, ou bien créé dans le cadre même de la page.

Voici un exemple d'un *bean* existant :

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC  "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
```

```
<jsp:useBean id="marin" beanName="marin" scope="request"
          type="org.paumard.cours.model.Marin"/>

<html>
  <head>
     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
     <title>JSP Page</title>
   </head>
   <body>
     <p>Using <%=marin%> ; </p>
   </body>
</html>
```

Ce *bean* utilise le tag <jsp:userBean> pour être déclaré . Cette opération permet l'association d'un nom à un *bean*. Ce nom sera ensuite utilisé dans la page, comme champ normal.

Et voici un deuxième exemple qui présente la création d'un nouveau *bean* de type Marin associé au nom de champ marin :

```
<jsp:useBean id="marin" scope="page" class="org.paumard.cours.model.Marin">
<jsp:setProperty name="marin" property="nom"    value="Surcouf"/>
<jsp:setProperty name="marin" property="prenom" param="prenom"/>
<jsp:setProperty name="marin" property="age"    param="age"/>
</jsp:useBean>
```

Chapitre V

Technologies Web avancées

V.1 Introduction

Le web 2.0 est un terme émergeant qui désigne les sites web modernes qui possèdent une interface à partir de laquelle les utilisateurs peuvent s'échanger entre eux et participer à l'enrichissement du site Web (création ou la modification du contenu) en utilisant du contenu multimédia (textes, photos, vidéos ...).

Le web 2.0 a permis entre autres le développement des blogs personnels, des flux RSS (*Really Simple Syndication*), des plateformes de partage de photos et de vidéos ainsi que la création des réseaux sociaux : Facebook, Twitter, instagram, Pinterest…

En conséquence, de nouvelles technologies Web sont développées afin de parvenir à satisfaire ces nouveaux sites Web en matière de rapidité et de performances. Dans ce chapitre, nous allons mettre l'accent sur trois de ces nouvelles technologies à savoir Ajax, JEE et Struts .

V.2 AJAX :

AJAX (*Asynchronous JavaScript And XML*) n'est pas un langage de programmation en lui-même, mais c'est une combinaison de plusieurs de technologies existantes depuis plusieurs années.

AJAX permet l'utilisation des fonctionnalités Web novatrices et utiles, qui ont connu un grand succès tel que Google Maps ou writely, etc.

Avec Ajax, on a la possibilité de créer des applications Web qui ressemble aux applications Windows. L'avantage essentiel d'Ajax réside dans une plus grande réactivité de l'interface par rapport au Web classique.

AJAX admet aux pages Web d'être mises à jour d'une façon asynchrone en effectuant des transactions des données avec un serveur Web sans que le client aperçoive cet échange. Cela implique que cette technologie est capable de rafraîchir des parties d'une page Web, sans recharger la page entière.

V.2.1. Principe d'AJAX :

Pour bien comprendre le principe de fonctionnement d'AJAX, nous allons comparer l'échange classique entre le client (représenté par son navigateur) et le serveur d'une part et le dialogue par le biais d'AJAX d'autre part.

L'échange classique entre le navigateur du client et le serveur se fait de la manière suivante : lorsque l'utilisateur qui est sur une page Web effectue une manipulation, le navigateur renvoie une requête avec la référence d'une page web au serveur. A son tour, le serveur va calculer et renvoie les résultats, sous forme d'une nouvelle page web au navigateur. Dans ce cas, chaque requête engendrer par une manipulation du client entraîne ainsi l'affichage d'une nouvelle page web. Le client est contraint à attendre la réponse du serveur avant de pouvoir effectuer de nouvelles manipulations. Cela représente une perte nette en matière de temps et de données.

Avec l'utilisation d'AJAX, la procédure est plus simple. AJAX intègre dans chaque page web un script écrit en JavaScript qui sera exécuté par le navigateur à chaque manipulation effectuée par l'utilisateur. Les requêtes sont envoyées en arrière-plan au serveur qui va rafraîchir le contenu de page actuelle sur le navigateur en fonction du résultat qu'il a obtenu. Cela implique une modification partielle de la page Web ce qui va éviter la retransmission des données et l'affichage d'une

nouvelle page complète à chaque manipulation. La figure suivante schématise cette comparaison :

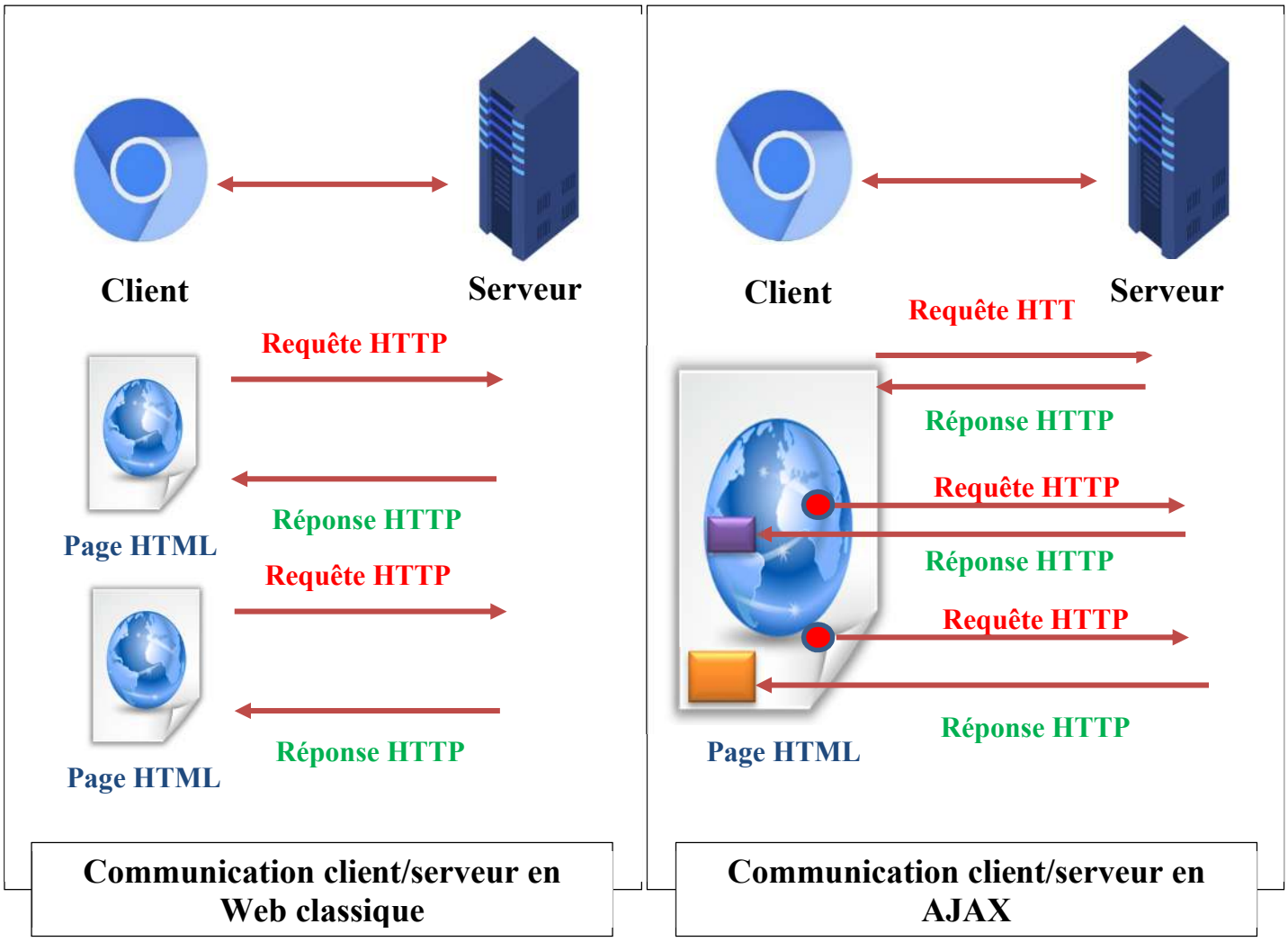


Figure V.1. Principe de fonctionnement d'AJAX

V.2.2. Les différentes technologies utilisées par AJAX :

Comme on a évoqué précédemment, AJAX est un ensemble de nouvelles technologies associé afin de favoriser le transfert de données entre le serveur et le navigateur web du client d'une façon performante et efficace. En conséquence, il est très importance de bien structurer les données des transactions. Voici les technologies couramment utilisées par AJAX :

- **HTML et CSS :**

Le HTML et le CSS servent pour la présentation de base des pages. Ces deux langages permettent une présentation standardisée selon les normes W3C des pages web.

- **JSON (*JavaScript Object Notation*) :**

L'autre technologie la plus couramment utilisée en AJAX est le JSON (*JavaScript Object Notation*), qui permet d'effectuer une segmentation des données dans un objet JavaScript, ce qui est très favorable pour économiser les données ce qui permet de petits transferts de données segmentées. Voici un exemple d'une représentation d'une liste de membres et leurs informations :

```
{
   Member1: {
      posts: 1702,
      inscription: '28/03/2020'
   },
   Member2: {
      posts: 2012,
      inscription: '04/06/2019'
    }
}
```

Similairement au XML, on reçoit ce code sous forme de chaîne de caractères. Cependant, il est nécessaire d'introduire l'objet nommé JSON afin de garantir le déclenchement du parseur. Cet objet possède deux méthodes :

- La première méthode est **parse()** qui consiste à prendre en charge les paramètres de la chaîne de caractères afin de les analyser et donner le résultat sous forme d'objet JSON.

- La deuxième méthode est **stringify()**. Cette méthode fait l'inverse de la première. En effet, elle consiste à récupérer les paramètres d'un objet JSON et de renvoyer son équivalent sous forme de chaîne de caractères. L'exemple suivant montre l'utilisation de ces deux méthodes :

```
var obj = {
index: 'contenu'
   },
string;
string = JSON.stringify(obj);

alert(typeof string + ' : ' + string); // Affiche : « string : {"index":"contenu"} »
obj = JSON.parse(string);
alert(typeof obj + ' : ' + obj); // Affiche : « object : [object Object] »
```

➢ **Le Dom et XMLHttpRequest :**

Le DOM (*Document Object Model*), une autre technologie utilisée par AJAX, permet d'accéder et de modifier directement des éléments dans une page HTML. L'objet XMLHttpRequest est objet JavaScript qui permet d'effectuer des requêtes de l'utilisateur vers le serveur en coulisse. II est supporté par tous les navigateurs modernes. Il communique avec le serveur par les méthodes GET/POST du protocole HTTP.

Pour créer un nouvel objet XMLHttpRequest sous Internet Explorer, il faut tout d'abord créer un ActiveX de la façon suivante :

xhr = new ActiveXObject("Microsoft.XMLHTTP")

ou bien *xhr = new ActiveXObject("Msxml2.XMLHTTP")*

Pour les autres types de navigateurs, l'objet XMLHttpRequest est incorporé par défaut : *xhr = new XMLHttpRequest();*

L'objet XMLHttpRequest a les méthodes et les propriétés suivantes :

- **Open ("method","URL"[,asyncFlag[, "userName"[,"password"]]]) :** cette propriété sert à préparer une requête en indiquant la méthode, l'URL, le drapeau de synchronisation, le nom d'utilisateur et le mot de passe.
- **Send (contenu) :** Lancer une requête avec un envoi de données.
- **ResponseText :** Réponse du serveur sous forme d'une chaîne de caractères.
- **ResponseXML :** Réponse du serveur sous forme d'un objet DOM.
- **Status :** code numérique de réponse du serveur HTTP.
- **StatusText :** message accompagnant le code de réponse. Les codes possibles pour le statut de l'objet et leurs StatusText sont résumés dans le tableau suivant :

Status	StatusText
0	non initialisé
1	Ouverture. La méthode open() a été appelée avec succès.
2	Envoyé. La méthode send() a été appelée avec succès.
3	En train de recevoir. Des données sont en train d'être transférées, mais le transfert n'est pas terminé.
4	Terminé. Les données sont chargées.

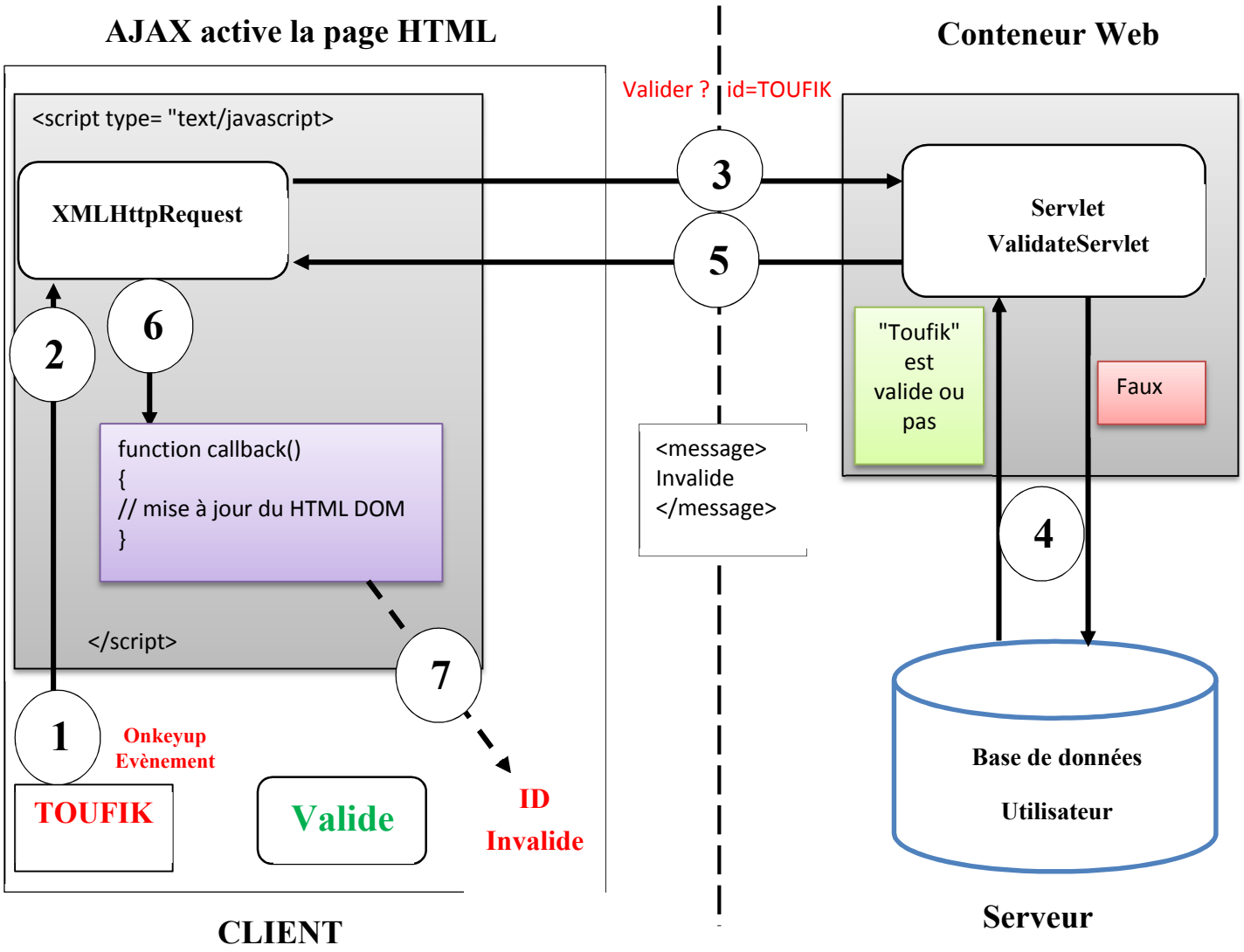
Tableau V.1. Status et StatusText

- **Le JavaScript :**

JavaScript est la technologie la plus indispensable dans AJAX. C'est grâce au code de JavaScript que le DOM et le XMLHttpRequest récupèrent les données du navigateur et du serveur.

V.2.3. Exemple concret d'une application AJAX :

Dans cette section, nous allons présenter les étapes d'exécution d'une application AJAX à travers l'illustration des interactions durant la validation de la saisie d'un formulaire. La figure suivante résume ces étapes :



1- Un événement client est déclenché 2- Création d'un objet XMLHttpRequest 3- L'objet XMLHttpRequest est configuré 4- L'objet XMLHttpRequest déclenche une requête asynchrone	5- La servlet ValidateServlet renvoie un document XML qui contient le résultat 6- La fonction callback () function est appelé par l'objet XMLHttpRequest afin de traiter le résultat 7- Le DOM HTML de la page est mis à jour.

Figure V.2. Exécution d'application AJAX : validation de données de la saisie

Dans cet exemple, un utilisateur tape des caractères dans un champ de données d'un formulaire d'entrée. Cela est désigné comme le numéro 1 dans la Figure précédente. Cela déclenchera un événement «onkeyup». Le gestionnaire d'événements «onkeyup» est ensuite appelé par l'objet XMLHttpRequest qui est créé et configuré. Cela est noté comme numéro 2 dans notre figure.

L'objet XMLHttpRequest envoie ensuite une requête HTTP avec ce que l'utilisateur a tapé sur le serveur. C'est le numéro 3 sur la figure. Le serveur effectue la validation des données et renvoie son résultat. Ces étapes sont indiquées par les numéros 4 et 5 dans la figure. La fonction *callback function* côté client est alors appelée. Cette fonction modifie ensuite un élément HTML dans la page pour refléter les données reçues du serveur. Nous allons détailler chaque étape de ce processus :

- **Etape 1 : Un événement client est déclenché :**

Dans cette étape, une fonction javascript est appelée lors de l'émission de l'événement. Voici un exemple de l'exécution de cette étape :

```
<input type="text"
        size="20"
         id="userid"
        name="id"
      onkeyup="validateUserId();">
```

Dans ce code, la fonction validateUserId() peut etre considérer comme un écouteur du déclenchement de l'événement *onkeyup* sur le champ *input* dont l'attribut id vaut "userid".

- **Etape 2 : Création d'un objet XMLHttpRequest :**

Voici un exemple de création de l'objet XMLHttpRequest :

```
var req;
function initRequest() {
```

```
    if (window.XMLHttpRequest) {
        req = new XMLHttpRequest();
    } else if (window.ActiveXObject) {
        isIE = true;
        req = new ActiveXObject("Microsoft.XMLHTTP");
    }
}
function validateUserId() {
    initRequest();
    req.onreadystatechange = processRequest;
    if (!target) target = document.getElementById("userid");
    var url = "validate?id=" + escape(target.value);
    req.open("GET", url, true);
    req.send(null);
}
```

Dans cet exemple, le gestionnaire d'événements validateUserId () crée un objet JavaScript XMLHttpRequest via la fonction initRequest (). Comme on a déjà indiqué précédemment, la façon dont un objet XMLHttpRequest est créé pour Microsoft Internet Explorer est différente de la façon dont il est créé pour les navigateurs de type Mozilla. Pour les navigateurs de type Mozilla, on utilise «new XMLHttpRequest ()» tout comme la façon dont on crée un nouvel objet Java tandis que pour Microsoft Internet Explorer, doit créer un objet ActiveX. C'est l'une des incompatibilités de navigateur qui existent toujours.

- **Etape 3 : L'objet XMLHttpRequest est configuré :**

  Voici un exemple de configuration de l'objet XMLHttpRequest :

```
var req;
function initRequest() {
    if (window.XMLHttpRequest) {
        req = new XMLHttpRequest();
    } else if (window.ActiveXObject) {
        isIE = true;
        req = new ActiveXObject("Microsoft.XMLHTTP");
```

```
    }
}

function validateUserId() {
    initRequest();
    req.onreadystatechange = processRequest; // callback function
    if (!target) target = document.getElementById("userid");
    var url = "validate?id=" + escape(target.value);
    req.open("GET", url, true);
    req.send(null);
}
```

Une fois l'objet XMLHttpRequest créé, la prochaine étape consiste à définir la propriété «onreadystatechange» de l'objet XMLHttpRequest sur la fonction *callback*. Cette fonction est appelée de manière asynchrone chaque fois qu'un changement d'état se produit sur l'objet XMLHttpRequest. Dans cet exemple, le nom de la fonction *callback* est «processRequest».

- **Etape 4 : L'objet XMLHttpRequest déclenche une requête asynchrone :**

Voici un exemple de déclenchement d'une requête asynchrone par l'objet XMLHttpRequest. Le servlet appelé vaut ici validate?id=TOUFIK :

```
function initRequest() {
    if (window.XMLHttpRequest) {
        req = new XMLHttpRequest();
    } else if (window.ActiveXObject) {
        isIE = true;
        req = new ActiveXObject("Microsoft.XMLHTTP");
    }
}
function validateUserId() {
    initRequest();
    req.onreadystatechange = processRequest;
```

```
    if (!target) target = document.getElementById("userid");
    var url = "validate?id=" + escape(target.value);
    req.open("GET", url, true);
    req.send(null);
}
```

La méthode «open» de l'objet XMLHttpRequest est appelée. La méthode «open» nécessite trois paramètres : la méthode HTTP (GET ou POST), l'URL du composant côté serveur avec lequel l'objet va interagir et un booléen indiquant si l'appel sera ou non effectué de manière asynchrone. Dans cet exemple, et dans la plupart des cas, il est défini sur "true", ce qui signifie que l'interaction requête / réponse HTTP se produit de manière asynchrone.

- **Etape 5 : Le servlet ValidateServlet renvoie un document XML qui contient le résultat :**

Voici un exemple du renvoie un document XML par la servlet ValidateServlet :

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws IOException, ServletException {
    String targetId = request.getParameter("id");
    if ((targetId != null) && !accounts.containsKey(targetId.trim())) {
        response.setContentType("text/xml");
        response.setHeader("Cache-Control", "no-cache");
        response.getWriter().write("<valid>true</valid>");
      } else {
        response.setContentType("text/xml");
        response.setHeader("Cache-Control", "no-cache");
        response.getWriter().write("<valid>false</valid>");
      }  }
```

La réponse HTTP est créée. Tout d'abord, le Content-Type est défini sur text/ xml. Puis, Cache-Control est défini sur no-cache afin d'empêcher les navigateurs d'utiliser les réponses de mise en cache locale. Enfin, il renvoie un fragment XML «vrai» ou «faux».

- **Etape 6 : La fonction callback() function est appelé par l'objet XMLHttpRequest afin de traiter le résultat :**

L'objet XMLHttpRequest a été configuré pour appeler la fonction callback (). Cette fonction est nommée processRequest () dans l'exemple suivant.

```
function processRequest() {
   if (req.readyState == 4) {
      if (req.status == 200) {
         var message = ...;
```

A présent, le contrôle revient au client. Nous avons vu à l'étape 3, que la fonction processRequest est définie comme une fonction *callback* chaque fois qu'il y a un changement d'état dans le champ readyState de l'objet XMLHttpRequest.

La combinaison de deux éléments (la valeur readyState de XMLHttpRequest est 4 et la valeur d'état 200) indique que les données du serveur ont été reçues avec succès par le client.

- **Etape 7 : Le DOM HTML de la page est mis à jour :**

Une fois les données reçues avec succès, le navigateur est alors prêt à afficher un message pertinent.

L'emplacement où on souhaite afficher un message est maintenant représenté par l'élément dans la page. Dans notre code Javascript, on obtient une référence à n'importe quel élément d'une page à l'aide de l'API DOM. Et la méthode recommandée à utiliser est *getElementById* et on passe l'id en tant que paramètre.

Dans l'exemple suivant, l'id de l'élément est défini comme «userIdMessage». Une fois qu'on a une référence à un élément, on peut faire à peu près tout ce qu'on veut faire, par exemple, changer les valeurs des attributs d'élément, le style d'élément où ajouter, supprimer, modifier les éléments enfants. Dans cet exemple, nous allons ajouter un nœud de texte en tant qu'élément enfant à l'élément «userIdMessage» :

```
<script type="text/javascript">
    function setMessageUsingDOM(message) {
        var userMessageElement =
    document.getElementById("userIdMessage");
        var messageText;
        if (message == "false") {
            userMessageElement.style.color = "red";
            messageText = "Invalid User Id";
        } else {
            userMessageElement.style.color = "green";
            messageText = "Valid User Id";
        }
    var messageBody = document.createTextNode(messageText);
        // si l'élément messageBody existe déjà : remplacer sinon ajouter un
        // nouvel élément
        if (userMessageElement.childNodes[0]) {
            userMessageElement.replaceChild(messageBody,
            userMessageElement.childNodes[0]);
        } else {
            userMessageElement.appendChild(messageBody);
        }
```

```
}
</script>
<body>
   <div id="userIdMessage"></div>
</body>
```

Dans ce code, on obtient une référence à l'élément en utilisant la méthode «getElementById» de l'objet document. Et puis, en fonction des données renvoyées par le serveur (soit "false" ou "true") la variable "messageText" est définie sur la chaîne de texte "Invalid User Id" ou "Valid User Id".

Un nouveau nœud de texte est ensuite créé à l'aide de la variable "messageText". Il est ensuite ajouté en tant que nœud enfant à l'élément "userIdMessage". Le navigateur affiche alors immédiatement le texte sur la page.

Ainsi, dans cette page, la mise à jour partielle se produit uniquement avec l'élément userIdMessage.

V.2.4. Sécurité sous AJAX:

Quant à la sécurité des applications Web AJAX, elles utilisent les mêmes schémas de sécurité que les applications Web classiques. En d'autres termes, tout comme les applications Web classiques, on peut spécifier nos exigences d'authentification, d'autorisation et de protection des données dans le fichier web.xml ou écrire le code qui effectue ces vérifications de sécurité.

Et toutes les requêtes HTTP provenant du client, que ce soit à partir d'une application Web régulière ou d'applications AJAX, sont limitées par les schémas de sécurité côté serveur. Il existe quelque différence entre les applications AJAX et les applications régulières sont du côté client. D'abord, le code JavaScript est visible pour un pirate, qui pourrait l'utiliser pour inférer des faiblesses côté serveur. De plus, le code JavaScript est exécuté sur le client. Et un code JavaScript mal intentionné

peut compromettre le client. Dans le même esprit, les applications basées sur AJAX sont soumises aux mêmes menaces de sécurité que les applications Web classiques. Par exemple, ils sont soumis à de telles menaces tels que les scripts interdites ou les failles d'injection.

V.3 La plateforme JEE :

Java EE pour *Java Enterprise Edition* (anciennement J2EE) est un ensemble de technologies destinées sur le développement d'applications d'entreprise à grande échelle. JEE est aussi une plateforme qui facilite le développement de ces applications en apportant un environnement d'exécution ainsi que des composants sous forme d'API.

La plupart des plateformes de développement des applications web y compris Java EE utilise le modèle MVC (décrit dans le chapitre précédent) qui découpe littéralement l'application en couches distinctes appelées Modèle – Vue – Contrôleur. Ce modèle impacte forcément l'organisation du code parce que chaque couche a une fonction bien précise.

Avec la plateforme JEE, chaque élément du modèle MVC possède un nom spécifique. Le Contrôleur porte le nom de Servlet. Le modèle est défini par des objets Java ou des *JavaBeans*. Il peut aussi communiquer avec les bases de données afin de stocker des informations. La Vue quant à elle est gérée par les pages JSP qui sont des pages qui font appel au code HTML ainsi qu'un code spécifique comme JAVA. Cette Vue est ensuite retournée au client par le biais du Contrôleur.

➢ Le servlet JEE :

Un servlet en java EE est une classe Java qui possède la faculté de traiter les requêtes et de personnaliser les réponses, c'est-à-dire qu'elle est capable de recevoir les requêtes HTTP envoyées depuis Client web et de lui renvoyer une réponse HTTP. Un servlet HTTP doit toujours hériter de la classe HttpServlet qui est en effet

une classe abstraite qui fournit des méthodes qui doivent être redéfinir dans la classe héritière. Parmi ces méthodes on y retrouve entre autres : doGet()doPost(),doHead()

- **Les JavaBeans :**

Les *Javabeans* sont un modèle de composants du langage Java. Ce sont des. Ces modèles représentent généralement des données issues du monde réel. Les principaux concepts mis en jeu pour un Bean sont les suivants :

- **Propriétés :** les propriétés permettent de paramétrer le bean, en y stockant des données.
- **La sérialisation :** Ce mécanisme permet une persistance de donnée voir de l'application elle-même.
- **La réutilisation :** un bean est un composant conçu pour être réutilisable.
- **L'introspection :** un bean est conçu pour être paramétrable de manière dynamique.

Ainsi tout objet conforme à ces quelques règles peut être appelé Bean. Il est également important de noter qu'un JavaBean n'est pas un EJB (Entreprise Java Bean).

Un bean doit également avoir la structure suivante :

- Dois être une classe publique
- Dois avoir au moins un constructeur par défaut, public et sans paramètres.
- Peut implémenter l'interface Serializable, il devient ainsi persistant et son état peut être sauvegardé ;
- Ne dois pas avoir de champs publics ;
- Peut définir des propriétés (des champs non publics), qui doivent être accessibles via des méthodes publiques getter et setter, suivant des règles de nommage.

➢ **Les composants JEE**

La spécification Java EE définie les composants Java EE suivants :

- **Les composants Clients exécutés côté client :** Ces composants peuvent être soit des clients web ou bien des applications clientes. Les clients web sont composés de deux parties : les pages web dynamiques qui contenant différents types de langage de balise (HTML, CSS, XML, etc.) d'un côté, et d'un autre coté une d'application cliente qui s'exécute sur un ordinateur client et permet aux clients de faire la gestion des tâches nécessitant une interface utilisateur plus riche que celle fournie par un langage de balisage.
- **Les composants Web exécutés côté serveur (Moteur web) :** Les composants Web Java EE sont des servlets créés à l'aide de la technologie JSP. A rappeler que les servlets sont des classes de langage de programmation Java qui traitent dynamiquement les requêtes et construisent des réponses.
- **Les composants métier exécuté côté serveur (Moteur d'application) :** c'est la logique qui consiste à résoudre ou répondre aux besoins d'un domaine d'activité particulier tel qu'une banque.

➢ **Les conteneurs JAVA EE :**

Le conteneur JEE est un environnement d'exécution destiné à la génération des composants applicatifs. Les conteneurs JEE, on doit fournir :

- **Composants applicatifs :** Les servlets, les JSP, les EJB(Entreprise JavaBean) qui représente les composants de l'application.
- **Descripteur de déploiement :** C'est un fichier XML décrivant les composants applicatifs.

L'architecture d'un conteneur est composée de quatre parties :

- **Les interfaces des composants :** c'est un ensemble d'interfaces spécifiées par le conteneur qui est implémenté par les composants applicatifs.
- **API des services du conteneur :** c'est un service supplémentaire fourni par le conteneur.
- **Services déclaratifs :** ces des services introduits par le conteneur sur vos applications. La figure suivante résume ce principe :

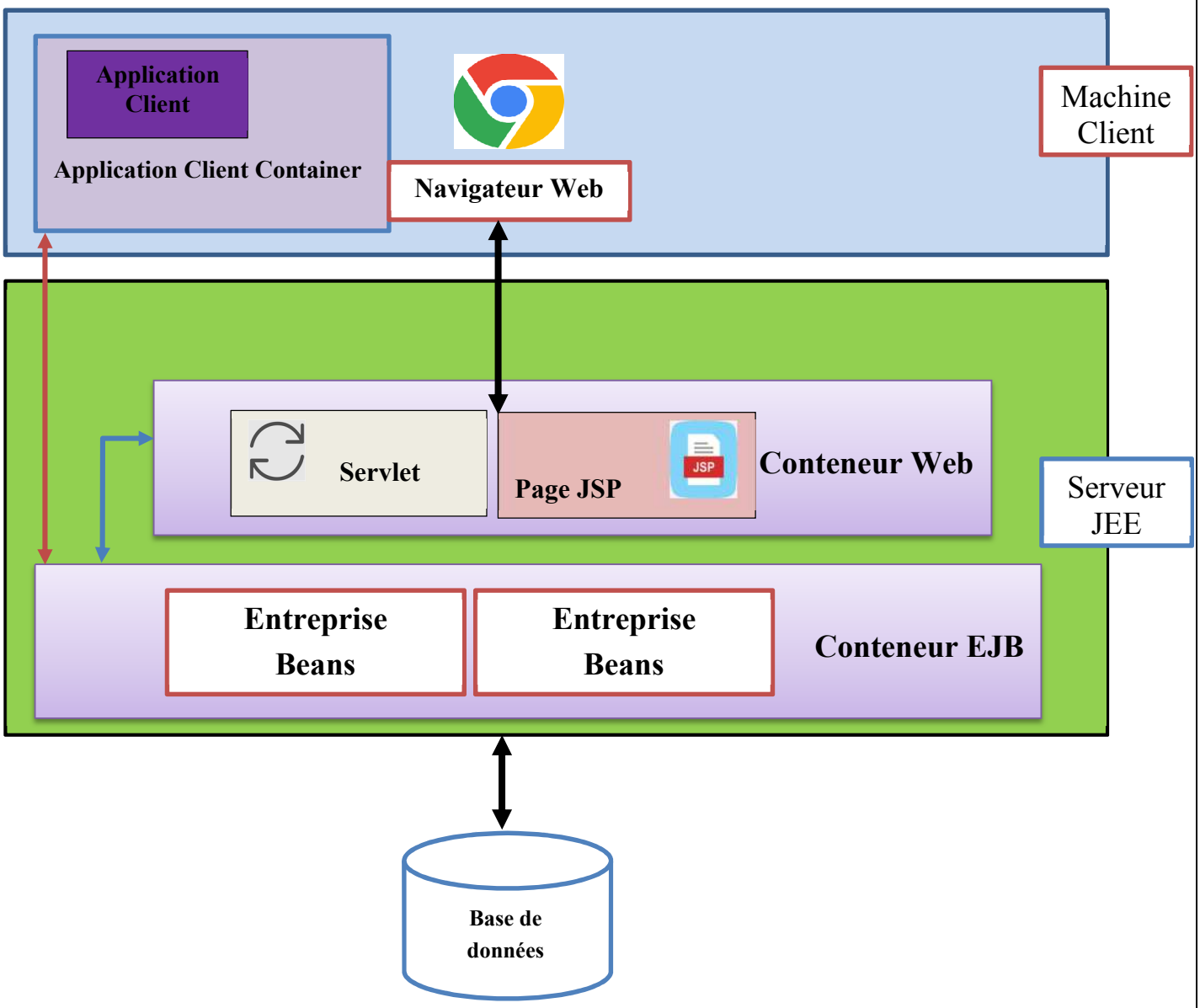


Figure V.3. Principe des conteneurs JAVA EE

➢ **Les API JEE :**

Les composants API (Application Programming Interface) spécifiaient les infrastructures de gestion de nos applications dans la plateforme JAVA EE. En effet cette plateforme offre un environnement d'exécution et un ensemble de services accessibles à travers des APIs afin de concevoir nos applications. Voici un ensemble d'extensions Java standard que chaque plate-forme JEE doit prendre en charge :

- **JNDI (*Java Naming and Directory Interface*):** permets de localiser et d'utiliser les ressources.
- **JDBC (*Java Database Connectivity*):** est une API qui permet aux programmes JAVA de se connecter et d'interagir avec les bases de données .
- **JMS (*Java Message Service*):** c'est une API permettant aux développeurs de construire des applications professionnelles.
- **JTA (*Java Transaction*):** API qui définit des interfaces standards entre un gestionnaire de transactions et les éléments impliqués en occurrence : l'application, le gestionnaire de ressource et le serveur.
- **JSP (*Java ServerPage*) :** permettant de simplifier la génération de pages web.
- **Servlet :** un servlet est un composant côté serveur, écrit en Java, qui a pour but de fournir une trame générale pour l'implémentation des transactions requête-réponse.
- **EJB (*Entreprise Java Bean*) :** c'est un environnement d'exécution qui fournit des services (Sécurité, Communications, Cycle de vie...).
- **Authentification :** JEE fournit des services d'authentification en se basant sur les concepts d'utilisateur, de domaines et de groupes.

V.4 Struts :

Struts est un cadre élégant et extensible pour créer des applications Web Java prêtes pour l'entreprise. Struts est un *framework* de la plateforme JEE. Le cadre du

Struts est conçu pour rationaliser le cycle de développement complet de la création au déploiement en passant par la maintenance des applications. Il existe deux versions de Struts : Struts1 et Struts2. Struts2 n'est pas seulement la prochaine version de Struts 1, mais c'est une réécriture complète de l'architecture Struts.

Struts s'appuie en réalité sur le modèle MVC, on distingue donc un contrôleur qui va guider les requêtes reçues du client vers des sous-contrôleurs qui ont pour tâche d'appliquer le traitement correspondant :

- **Le modèle** : Struts ne définit aucune implémentation concernant le modèle. Le concepteur de l'application Web doit donc définir lui-même le modèle approprié.
- **Les vues** : les vues Struts sont principalement des pages JSP. Pour cela, Struts fournit plusieurs bibliothèques de 'taglibs' permettant de réaliser les pages JSP rapidement.
- **Le contrôleur** : Struts définit un contrôleur principal (représenté par la classe ActionServlet) et des sous contrôleurs (correspondant aux classes Action). Le schéma suivant montre l'architecture Struts selon le modèle MVC :

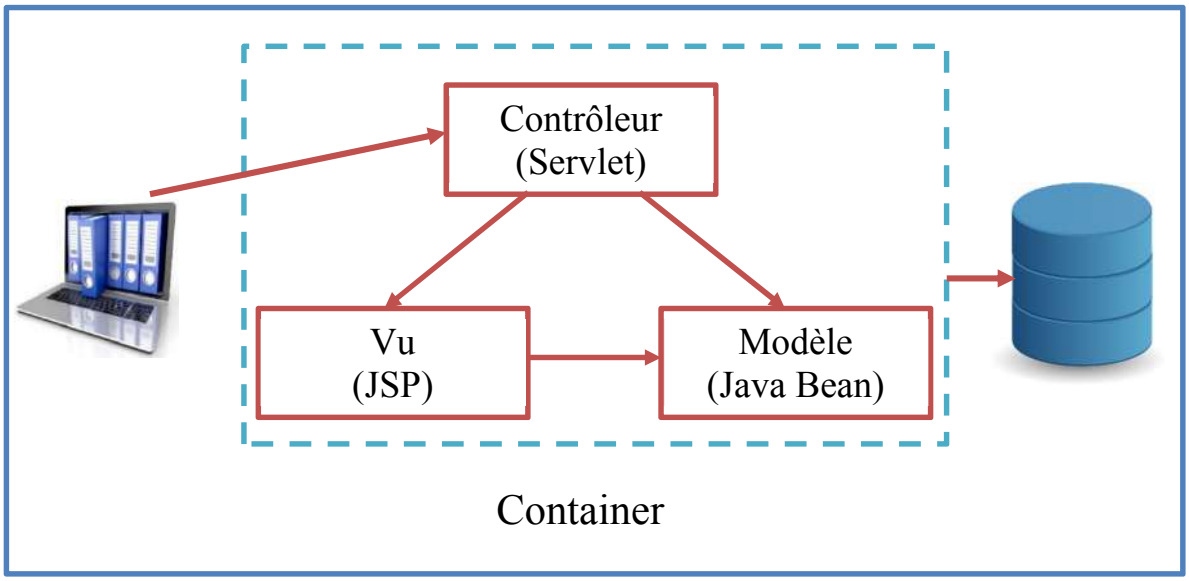


Figure V.4. Architecture de Struts

Struts est composé des éléments suivants :

- **ActionServlet** : Il représente le contrôleur principal. Cet élément est considéré comme servlet et reçoit en conséquence les requêtes du client qu'il renvoie ensuite vers les sous-contrôleurs.
- **Action** : il s'agit des sous-contrôleurs. Ils effectuent le traitement correspondant à une requête précise.
- **ActionForm** : Ils représentent les containers pour les pages JSP. Ils peuvent être utilisés par les Actions lors du traitement.
- **Le fichier de configuration 'Struts-config.xml'** : C'est le composant principal de Struts parce qu'il permet d'effectuer le lien entre tous les composants précédents de Struts. Nous reviendrons plus en détail sur ces composants un peu plus tard dans cette section. Le schéma ci-dessous représente la structure de Struts et son fonctionnement :

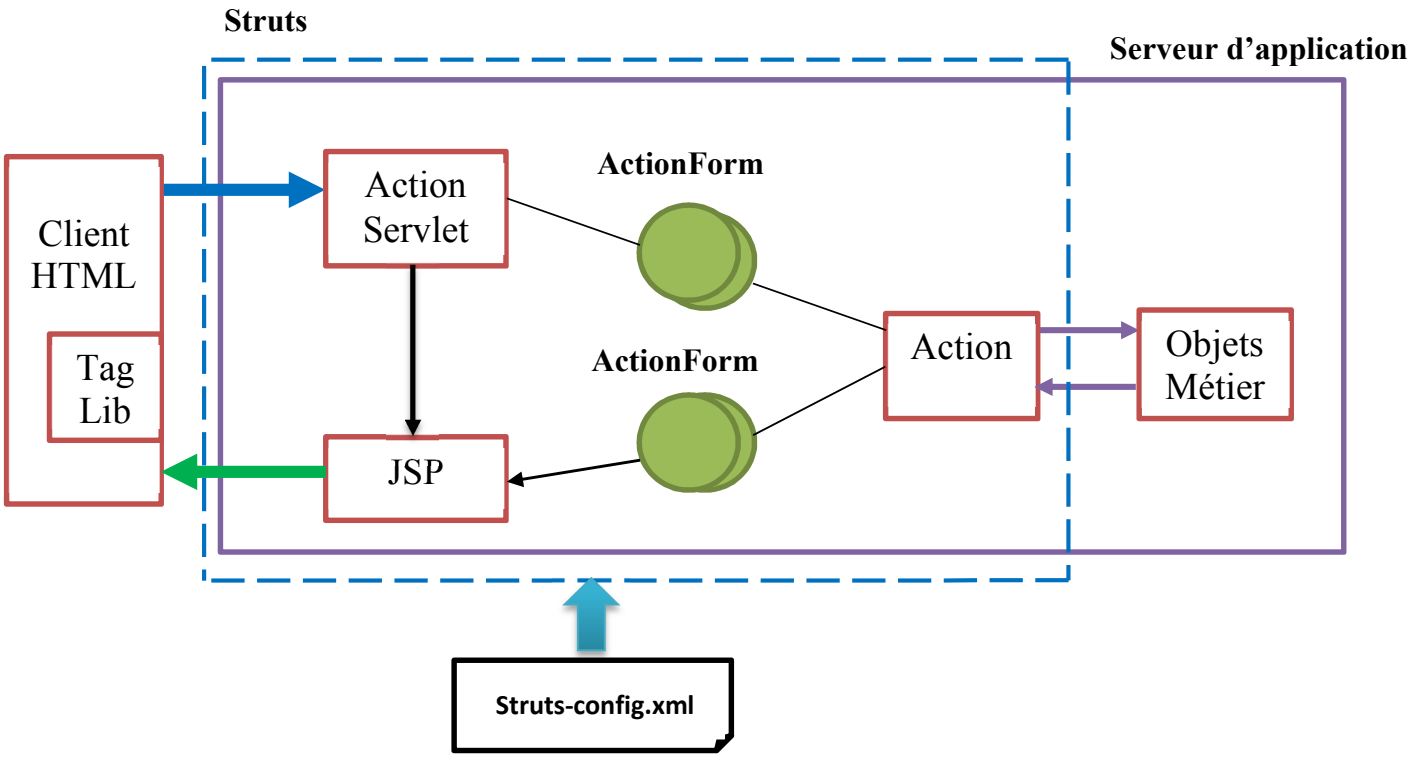


Figure V.5. Fonctionnement de Struts

Sur le schéma ci-dessus, le client envoie une requête à l'ActionServlet. l'ActionServlet aiguille la requête vers l'Action appropriée grâce au fichier de configuration "Struts-config.xml". L'Action applique ensuite le traitement correspondant. En cas de besoin, l'Action fait appel aux ActionForm nécessaires et effectue les opérations utiles sur le modèle. L'action renvoie ensuite le résultat du traitement (réussite, échec…). A partir de cette valeur, l'ActionForm est alors capable de déterminer le résultat à renvoyer au client (redirection vers une autre page JSP…).

A présent, nous allons nous intéresser aux principaux composants de Struts.

- **Le contrôleur principal : l'ActionServlet :**

Comme son nom l'indique, il s'agit d'un servlet. Il représente aussi le contrôleur principal de Struts qui a pour tâche de recevoir toutes les requêtes du client et de répartir ensuite ces requêtes vers les sous-contrôleurs à l'aide des informations contenues dans le fichier de configuration "Struts-config.xml".

Une application Struts contient généralement une seule implémentation de la classe ActionServlet. C'est alors le servlet principal de notre application, et comme pour la plupart d'application WEB, ce servlet doit être déclarée dans le fichier "web.xml" de la façon suivante :

```
<web-app>
  <servlet>
    <servlet-name>action</servlet-name>
    <servlet-
class>org.apache.struts.action.ActionServlet</servlet-class>
    ...
  </servlet>
  <servlet-mapping>
```

```
  <servlet-name>action</servlet-name>
  <url-pattern>*.do</url-pattern>
 </servlet-mapping>
 ...
</web-app>
```

La balise *servlet mapping* précise que toutes requêtes se terminant par ".do" devront être traitées par le servlet action. Cette extension est ajoutée d'une façon automatique lorsqu'on utilise les bibliothèques propres à Struts.

L'ActionServlet sert ensuite de "Factory" pour les sous-contrôleurs Action puisqu'elle effectue une duplication des Actions en fonction des requêtes reçues et par rapport au ficher de configuration "Struts-config.xml".

Dans la plupart des cas, ce mécanisme est automatique et visible pour l'utilisateur qui aura juste à implémenter les sous-contrôleurs et renseigner le fichier de configuration de Struts.

- **Les sous-contrôleurs : les Actions :**

La classe Action permet de définir un traitement spécifique. Cette classe définit une méthode appelée "execute" qui est appelée par le contrôleur principal d'une façon automatique lorsque le traitement doit être effectué.

Chaque Action créée doit être déclarée dans le fichier de configuration XML. Voici un exemple pour un la classe MyAction qui étend d'Action :

```
<action
 path="/actionPerso"
 input="/index.jsp"
 name="myForm"
```

```
type="myPackage.MyAction">
<forward name="action_failed" path="/index.jsp" />
<forward name="action_ok" path="/autre.jsp" />
</action>
```

L'attribut "path" représente le chemin d'accès de la requête, il est utilisé par les pages JSP pour désigner l'action à appeler. L'attribut 'input' spécifie la page à retourner en cas d'erreur. L'attribut "name" est le nom de l'ActionForm utilisé pour effectuer l'action.

Les éléments "forward" reprennent les différentes valeurs pouvant être retournées par l'action et définissent la redirection associée. Ces éléments sont utilisés par l'ActionServlet afin de renvoyer un résultat au client. Il existe des attributs additionnels en relation avec les autres composants de Struts. Ainsi, l'attribut "validate" (pouvant prendre les valeurs logiques) permet de forcer ou non l'appel à la méthode "validate" de l'ActionForm lié à l'Action.

- **Les containers : les ActionForm :**

Les ActionForm servent comme containers pour les pages JSP, ils fonctionnent comme un bean ce qui les rend conformes au standard Java Bean.

Les ActionForm permettent de stocker les différentes valeurs saisies dans une page JSP à l'aide des accesseurs. Comme pour les Actions, on doit étendre cette classe afin de créer ses propres containers.

Comme la majeure partie des composants Struts, les ActionForm doivent être déclarés dans le fichier de configuration XML de la façon suivante :

```
<form-beans>
 <form-bean name="myForm" type="myPackage.MyForm">
  <form-property name="membre1" type="java.lang.String" />
```

```
<form-property name="membre2" type="java.lang.String" />
</form-bean>
</form-beans>
```

Les ActionForm définissent aussi une méthode "validate" qui permet de traiter les erreurs.

- **Le fichier de configuration XML :**

Ce fichier représente le descripteur de déploiement, il permet d'effectuer le lien entre les différents composants de Struts. Il permet de décrire les différents éléments de Struts dont : les ActionForm, les Actions les ressources, les redirections…

Ce fichier de configuration est inclus dans le fichier "web.xml" de l'application à l'aide des lignes suivantes (celles-ci se situent à l'intérieur des balises 'servlet' avec la déclaration du servlet principale) :

```
<init-param>
  <param-name>config</param-name>
  <param-value>/WEB-INF/struts-config.xml</param-value>
</init-param>
```

Conclusion générale

Nous avons désormais fait le tour des notions à connaitre de plusieurs langages liés au développement Web. Vous connaissez désormais le rôle de chacun de ces langages et savez les manipuler.

Désormais, vous êtes capables d'utiliser ces langages pour donner le plus de sens à vos sites Web et de perfectionner vos contenus afin de plaire à vos visiteurs.

Ce cours vous a permis entre autres de connaitre et comprendre un ensemble de concepts et de connaissances du monde du Web afin de vous permettre de bien assimiler le fonctionnement des différents langages entre eux et l'utilité de chacun dans un site web, vous donnant par la même une vue d'ensemble pour aborder vos projets futurs.

A présent, il vous est demandé de pratiquer ces langages le plus possible et de créer vos propres sites Web parce qu'il est vraiment essentiel d'être confronté aux difficultés pour apprend véritablement à coder.

Bibliographie

- "XHTML et CSS: cours et exercices", Jean Engels, éditions eyrolles, 2006.
- "CSS 2 : pratique du design web", R. GoetteR , 2005, 324 pages.
- "Design web : utiliser les standards, CSS et XHTML", J. zeldman, 2006, 444 pages
- "Maîtrise des CSS ",Andy Budd, Simon Collison et Cameron Moll, éditions perason, 2009, 308 pages.
- "Bien Developper Pour Le Web 2.0", C.Porteneuve , éditions eyrolles, 2006.
- "Développez en Ajax", M. Plasse, éditions eyrolles, 2006, 314 pages.
- "PHP 5 avancé", E. dasPet et C. PieRRe de GeyeR., éditions eyrolles, N°12004, 3e édition 2006, 804 pages.
- "Best practices PHP 5", éditions eyrolles , G. Ponçon, N°11676, 2005, 480 pages.
- "PHP 5 (et XML) (Les Cahiers du programmeur)", S. Mariel, éditions eyrolles , N°11234, 2004, 290 pages.
- "Les Cahiers du programmeur PHP 5, PHP objet et XML", Stéphane MARIEL, éditions eyrolles ,N°11234, 2004.
- "Les Cahiers du programmeur ASP.NET ", Thomas PETILLON, éditions eyrolles, N°11210, 2003.
- "JavaScript : l'essentiel du code et des commandes", Christian Wenz, éditions perason, 2009, 280 pages.
- "JSP professionnel, Avec sept études de cas combinant JavaServer Pages, JDBC, JNDI, EJB, XML, XSLT et WML", K. Avedal, éditions eyrolles, N°9247, 2001, 950 pages.
- "De VB6 à VB.NET", D. Appleman, éditions eyrolles, N°11037, 2002, 500 pages.
- "Jakarta Struts", J. Goodwill, éditions eyrolles, N°11231, 2003, 354 pages.