TOULOUSE

Informatique

iut Midi-Pyrénées

UNIVERSITÉ TOULOUSE III

PROGRAMMATION INTERNET

S2 - PGI

Sommaire

CHAPITRE 1 : ARCHITECTURE CLIENT/SERVEUR

1.0 Introduction

L'architecture **client/serveur** désigne un mode de communication entre plusieurs ordinateurs d'un réseau qui distingue un ou plusieurs postes clients du serveur : chaque logiciel client peut envoyer des requêtes à un serveur. Un serveur peut être spécialisé en serveur d'applications, de fichiers, de terminaux, ou encore de messagerie électronique.

Caractéristiques d'un serveur :

- il est passif (ou esclave) ;
- il est à l'écoute, prêt à répondre aux requêtes envoyées par des clients ;
- dès qu'une requête lui parvient, il la traite et envoie une réponse.

Caractéristiques d'un client :

- il est actif (ou maître) ;
- il envoie des requêtes au serveur ;
- il attend et reçoit les réponses du serveur.

Le client et le serveur doivent bien sûr utiliser le même protocole de communication. Un serveur est généralement capable de servir plusieurs clients simultanément.

Un autre type d'architecture réseau est le poste à poste (ou *peer-to-peer* en anglais), dans lequel chaque ordinateur ou logiciel est à la fois client et serveur.

1.1 Architectures trois tiers et multi-tiers

Les termes « trois tiers » et « multi-tiers » sont abusivement traduis de l'anglais *three tier* et *multi-tier* ou *n-tier*.

L'architecture client/serveur possède deux types d'ordinateurs sur un réseau : les clients et les serveurs, elle possède donc deux niveaux et s'appelle *two-tier* en anglais. Les architectures multi-tiers (ou distribuées) scindent le serveur en plusieurs entités (par exemple, un serveur d'application + un serveur de base de données).

>>> Avantages par rapport aux architectures distribuées

- Toutes les données sont centralisées sur un seul serveur, ce qui simplifie les contrôles de sécurité et la mise à jour des données et des logiciels ;
- les technologies supportant l'architecture client/serveur sont plus matures que les autres.

>>>Inconvénients

- Si trop de clients veulent communiquer avec le serveur au même moment, ce dernier risque de ne pas supporter la charge (alors que les réseaux P2P marchent mieux en ajoutant de nouveau participants) ;
- si le serveur n'est plus disponible, plus aucun des clients ne marche (le réseau P2P continue à marcher, même si plusieurs participants quittent le réseau).

>>>Exemples

- La consultation de pages sur un site internet fonctionne sur une architecture client/serveur. Un internaute connecté au réseau via son ordinateur et un navigateur Web est le client, le serveur est constitué par le ou les ordinateurs contenant les applications qui délivrent les pages demandées. Dans ce cas, c'est le protocole de communication HTTP qui est utilisé.

- Les courriels sont envoyés et reçus par des clients et gérés par un serveur de messagerie. Les protocoles utilisés sont le SMTP, et le POP ou l'IMAP La gestion d'une base de données centralisée sur un serveur peut se faire à partir de plusieurs postes clients qui permettent de visualiser et saisir des données.

- Le système X Window fonctionne sur une architecture client/serveur. En général le client tourne sur la même machine que le serveur mais peut être aussi bien lancé sur un autre ordinateur faisant partie du réseau.

1.2 Protocoles

C'est le protocole de communication HTTP qui, sur le Web, permet de transférer à partir d'un serveur HTTP, un document HTML. Lorsque le serveur HTTP reçoit une demande concernant un document, il est possible qu'il ait à générer une partie du document suivant les indications qu'aura eu soin de lui laisser l'auteur de la page (cf. langages spécialisés web).

Les documents HTML sont identifiés par une URL et sont interprétés par le navigateur Web du visiteur. Grâce à ce dernier, le document HTML apparaît à l'écran ou à l'impression comme l'auteur l'a voulu. Sont ainsi représentés texte, typographie, couleurs, tableaux, images, parfois du son, etc.

1.3 - Page web simple : HTML statique

Le code HTML est le langage de base pour concevoir des pages destinées à être publiées sur le réseau Internet ou intranet. Ce n'est pas un langage de programmation mais un simple outil de description de documents web. Une page en HTML peut contenir un certain nombre d'objets (textes, images, sons, vidéos,...) qui seront affichés ainsi que des instructions de mise en page qui déterminent la façon dont ces objets seront affichés.

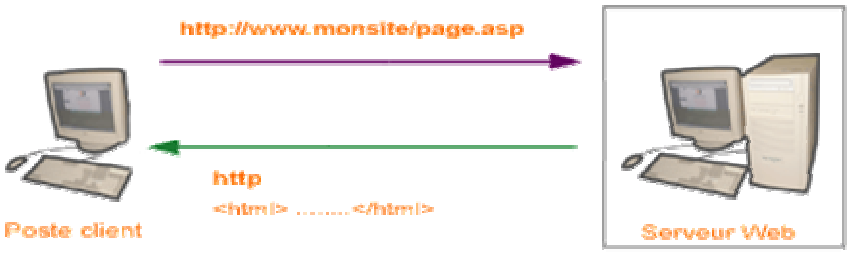
L'exemple suivant utilise une page web utilisant le code HTML

```
<HTML>
        <HEAD>
                <TITLE>exemple 1</TITLE>
        </HEAD>

        <BODY>
                <P>
                Do bee do bee do ...
                </P>
        </BODY>
</HTML>
```

Lorsque le client lance une requête pour accéder à cette page (en cliquant sur un lien ou en tapant l'URL correspondant). Le serveur, quelque part sur le réseau Internet ou Intranet, renvoie la réponse en format HTML.

Pour une requête simple, la procédure est illustrée sur la figure suivante.



Le dialogue entre le serveur et le poste client s'effectue, en réalité, entre un logiciel serveur (voir plus loin) et un programme client que nous appelons tous ***un navigateur***, grâce auquel un utilisateur du web peut demander et consulter très simplement des documents.

Ce dialogue s'effectue selon des règles précises qui constituent ***un protocole.*** Le protocole de web est le HTTP (HyperText Transfert Protocol = protocole de transfert hypertexte). Il est basé sur le protocole TCP/IP et permet de définir le format, le contenu et l'ordre des messages échangés entre le client et le serveur. Comme tous les messages sont échangés sous forme de chaînes de caractères ASCII, le protocole HTTP est largement multi-plateforme.

La communication entre client et serveur web s'effectue de la manière suivante :

le client envoie un message de requête au serveur web. Ce dernier y répond par un message de réponse. Le protocole HTTP organise la structure de ces messages. Lors de l'envoi, une connexion TCP/IP est établie avec le serveur web. Si celui-ci peut fournir la page demandée, le document est retourné, en format HTML, par la même connexion, qui est ensuite refermée.

N.B.

Il est possible de communiquer avec un serveur via d'autres protocoles, comme FTP qui permet d'échanger des fichiers ou SMTP protocole simple de transfert de courrier, POP, IMAP,...

Les pages web statiques sont des documents en HTML invariables préparées à l'avance. Le serveur renvoie ces pages à l'utilisateur mais n'effectue aucune action particulière. Le code source

de la page affichée par le navigateur sur le poste client est identique au code source de la page web installée sur le serveur.

Les avantages *du code HTML statique* :

1. le code HTML est facile à comprendre, à corriger et à produire.
2. tous les navigateurs, en principe !, sont capables de l'afficher correctement !!
3. les requêtes sont traitées rapidement par le serveur en utilisant moins de ressources.

ses inconvénients :

1. il est difficile de faire évoluer les pages web (imaginez la mise à jours des annuaires ou moteurs de recherche s'ils utilisaient que HTML statique !!)
2. manque d'interactivité
3. les contenus ne sont pas personnalisés à la demande du client

Pour ces raisons, le code HTML statique n'est plus à "la mode" et ne peut plus répondre aux exigences liées à la création, l'animation et la mise à jour d'un site web "moderne" (ou plutôt professionnel).

De nombreuses technologies complémentaires ont été développées pour répondre à ces limites. Du coté client (client side) : on trouve Javascript, Jscript, Vbscript, Les feuilles de style en cascade le DHTML, XML, XSL, les applettes Java,...

L'utilisation de scripts coté client ne doit pas laisser penser que l'on se trouve en présence d'un vrai site dynamique.

Exemple 2

Les deux pages ci-dessus, avec et sans langage javascript, s'afficheront da la même façon

```
<HTML>
	<HEAD>
		<TITLE>exemple 1</TITLE>
	</HEAD>

	<BODY>
		<P>
		Do bee do bee do ...
		</P>
	</BODY>
</HTML>
```

Equivalent

```
<HTML>
	<HEAD>
		<TITLE>exemple 1
		</TITLE>
	</HEAD>
	<BODY>
	<P>
	<SCRIPT LANGUAGE="JavaScript">
document.write("Do bee do bee do...");
	</SCRIPT>
		</P>
	</BODY>
</HTML>
```

1.4 -Page dynamique : Script côté serveur

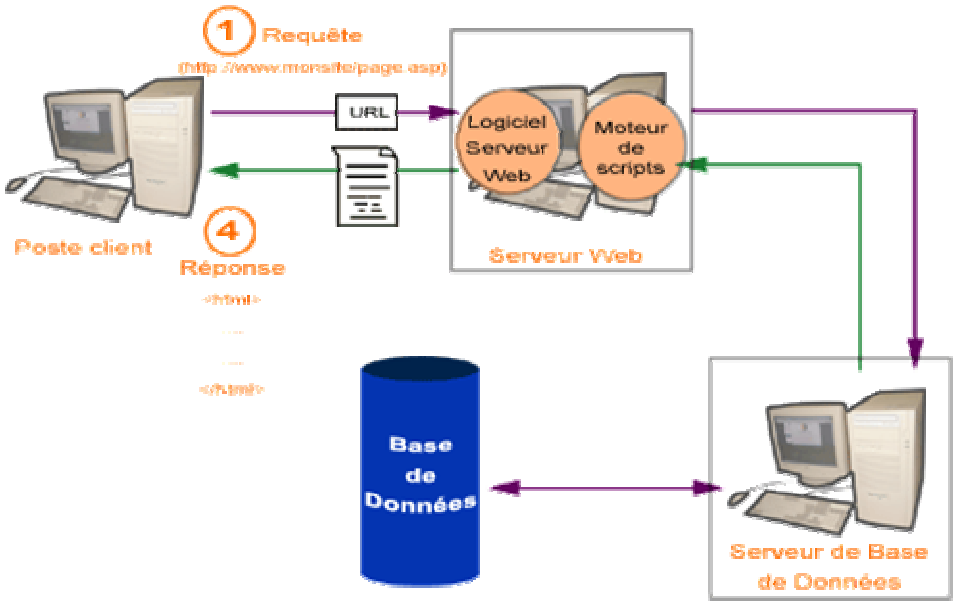
Le contenu des pages web dynamiques sont créés en réponse à une demande bien prise d'un utilisateur du web.

Les pages web dynamiques font appel à un langage de programmation. Ce langage est lié à des technologies autorisant la connexion aux bases de données. On note, à titre d'exemple, le CGI, ASP, PHP, IDC/HTX, etc.

La transmission des données, qui s'afficheront par la suite sur le poste client, sont liées à l'exécution d'un certain nombre d'instructions (scripts) sur le serveur et on parle donc des scripts coté serveur.

Comment cela se passe-il ?

Le dialogue entre le poste client et le serveur, dans le cas des scripts coté serveur, est schématisé dans la figure suivante :



Du client vers le serveur

L'utilisateur remplit un formulaire en ligne ou clique sur un lien dynamique. Les données sont envoyées au serveur web. Ce dernier est géré par un logiciel serveur appelé aussi serveur httpd

Quelques logiciels serveurs

- IIS (Internet Information Serveur) est un serveur Web/Ftp/Gopher
- Apache : logiciel libre, donc gratuit et dont le code source est ouvert donc disponible.
- HTTPD, le serveur de Bob Denny
- WebSite
- NCSA httpd
- CERN httpd
- Serveur Web Personnel Server (PWS)
- ...

Le logiciel serveur réagit en transmettant les données à un script en vue de leur traitement, en se connectant, puis en interrogeant une base de données.

Du serveur vers le client :

Les pages web sont assemblées à partir des résultats produits par le serveur de données et renvoyées à l'utilisateur dans une enveloppe http, en format HTML.

Lorsque les données arrivent sur le poste client, le navigateur fait de son mieux pour les afficher.

Le travail le plus important d'écriture de script coté serveur consiste à connecter le serveur web au serveur de données dont l'architecture et le type peuvent être différents. (Si les moyens ne permettent pas d'avoir deux serveurs différents, une seule machine peut être serveur web et serveur de données !).

1.5 - Différentes technologies Web / Bases de données

Il existe plusieurs modèles d'architecture web couplés aux bases de données :

- L'accès aux bases de données via les scripts CGI (Common Gateway Interface)
- l'accès aux bases de données via les API (Application Programming Interface = Interface de programmation d'application)
- l'accès aux bases de données via les Middleware
- ...

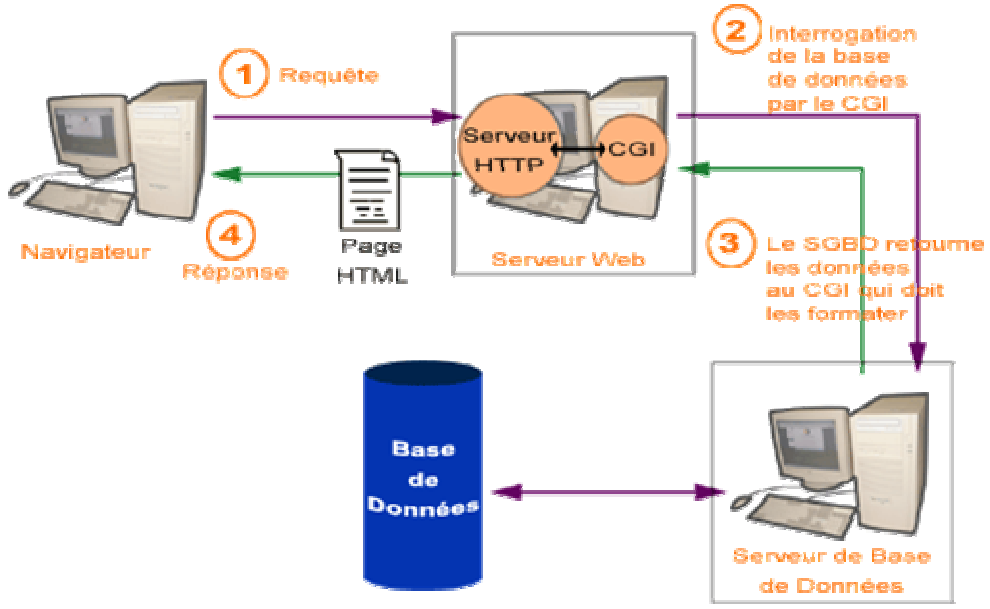
Il y a donc plusieurs approches dont le choix dépend du serveur web et du système d'exploitation utilisés.

>1.5.1 Common Gateway Interface (CGI)

La première technique utilisée est le standard multiplateforme ***CGI*** *(**C**ommon **G**ateway **I**nterface)* : programme écrit en shell, C ou le Perl. Ce dernier reste le langage typique de gestion des requêtes formulées par une page web. Le CGI peut être une application exécutable qui offre d'autres possibilités que la récupération de données. Il permet de tirer partie d'autres fonctionnalités du système d'exploitation.

Le serveur web envoie les informations du navigateur web sous la forme de chaînes de caractères, et l'application CGI renvoie une chaîne de contenant le code HTML de la page à retourner au navigateur.

La figure ci-dessous schématise la transmission des données dans le cas des scripts coté serveur utilisant le script CGI.



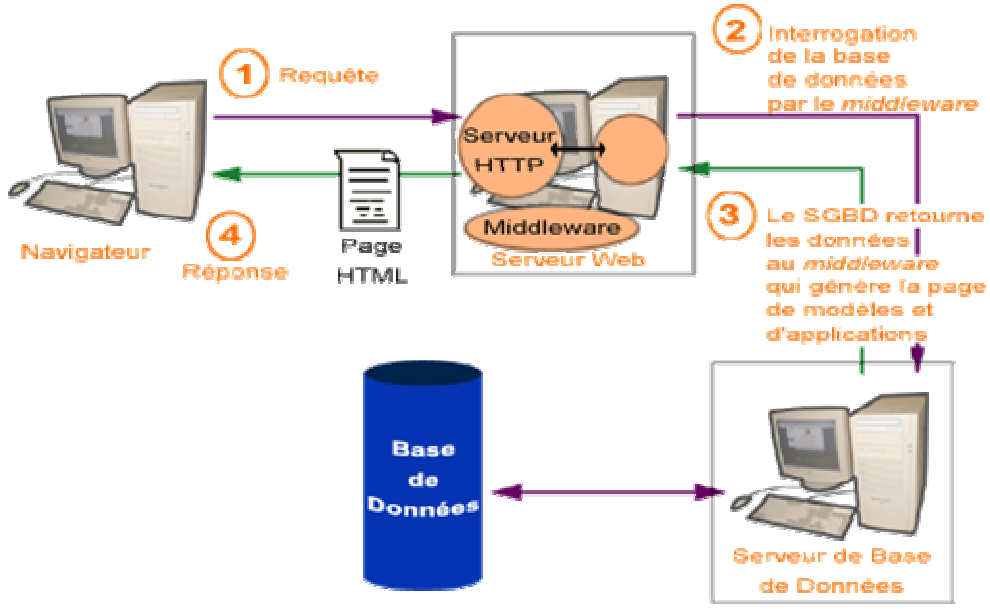
Un programme CGI peut être écrit dans de nombreux langages. La seule condition est en fait, que le langage choisi puisse être exécuté sur une ligne de commande sans faire appel à un autre programme. On utilise sous UNIX les langages PERL, C, C++,shel, Fotran, Pascal, sous windows on utilise le C, le C++, visual Basic et sur le Macintosh, essentiellement AppleScript. Si les scripts sont écrits dans un langage de programmation qui demande à être compilé (C, C++, Fortran, Pascal…), les fichiers sources se trouvent généralement dans le répertoire /cgi-src, mais les fichiers compilés sont dans le répertoire /cgi-bin. Si les scripts sont écrits dans un langage de programmation directement interprétable (PERL, shel UNIX, AppleScript,…), ils doivent se trouver dans le répertoire /cgi-bin. Dans tous les cas (surtout sous UNIX), il faut vérifier que les fichiers possèdent bien les permissions d'exécution.

>1.5.2 Accès aux bases de données via les middleware

Un middleware est un traducteur qui met en relation deux programmes essayant de changer des informations. Utiliser cette technique consiste à développer une couche logiciel entre l'application et le réseau.

Exemple :

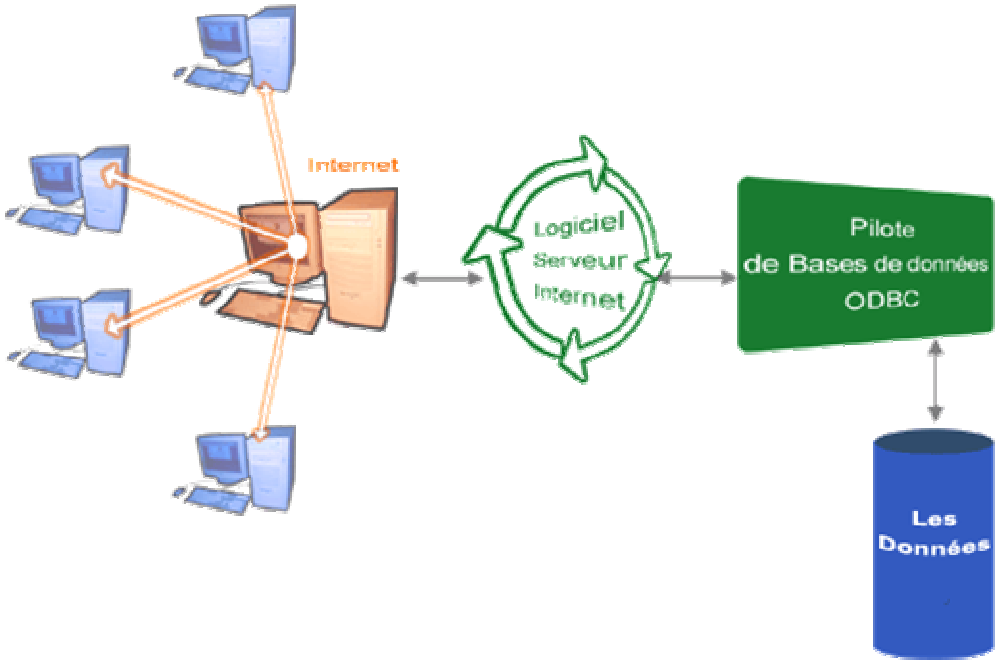
La figure suivante illustre les aspects essentiels d'une communication client/serveur utilisant un middleware



Le SGBD Sybase utilise un middleware, nommé Adaptive Server.

>*1.5.3 Accès aux bases de données via les API*

Cette technologie est plus récente, il existe deux formes d'API (Application Programming Interface) auxquelles se conforment les principales bases de données : NSAPI de Netscape et ISAPI (Internet Server Application Programming Interface). Ce mode d'accès est très utilisé, actuellement, mais il reste très lié aux types de serveurs http et aux bases de données. En effet, les ISAPI sont une interface propre à Microsoft. Si on utilise IIS (Internet Information Serveur) de windows NT ou un autre serveur compatible. ISAPI permet d'utiliser un ensemble de fonctions utilisables depuis la plupart des langages de programmation. Ces fonctions permettent de récupérer les informations en provenance du navigateur et de lui transmettre en retour des pages générées dynamiquement. ISAPI se connecte aux bases de données en utilisant une connexion spéciale.



Cette connexion a besoin d'une combinaison matériel-logiciel entre la base de données et le monde extérieur. Elle utilise une couche de traduction adéquate permettant les mouvements de données entre Internet et la base de données.

Pour mettre en place physiquement cette connexion, la machine serveur doit disposer d'un logiciel de connexion à l'Internet qui supporte la connexion ODBC (IIS, website, PWS, ...).

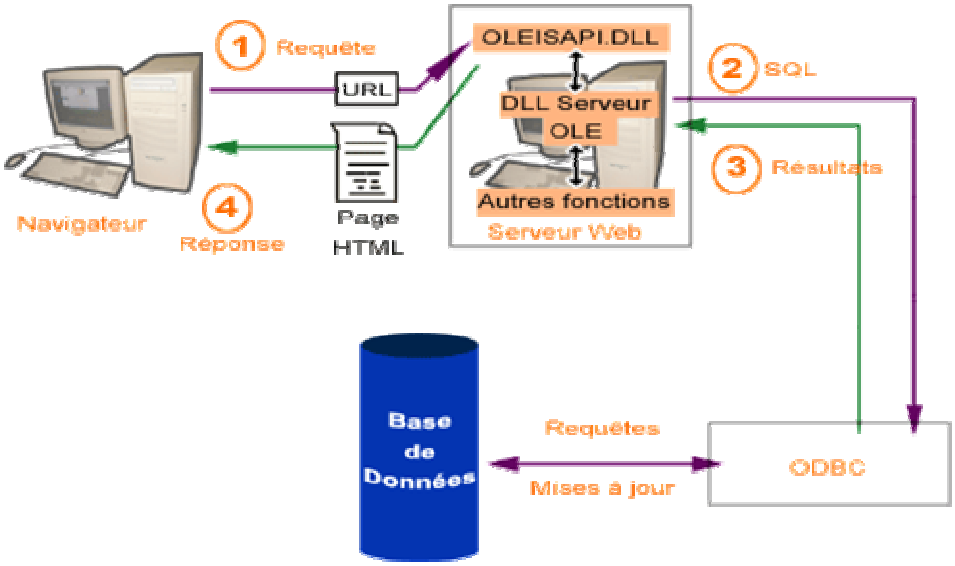
>1.5.4 Couche ODBC

Open **D**ata**b**ase **C**onnectivity est une interface standard d'accès à des bases de données hétérogènes (sql Server, access, oracle, DB2, dbase,…) .

La communication s'effectue via un pilote (driver) qui effectue le lien entre un moteur de script et le SGBD.

Il est possible de se connecter directement à l'API d'Internet Information Serveur en créant une bibliothèque de liens dynamiques (DLL – Dynamic Link Library) au standard Windows. Cependant, on peut utiliser la technologie ActiveX (OLE), référencée sous le nom de OLEISAPI (API OLE d'Internet Server), ce qui nécessite la création d'un serveur OLE Automation. Un serveur OLE Automation est essentiellement une collection d'objets programmables en vue d'exécuter certaines tâches. On peut utiliser Visual Basic pour créer des DLL de type serveur OLE.

La figure ci-dessous schématise la transmission des données dans le cas des scripts coté serveur utilisant l'application OLEISAPI.



OLEISAPI a ouvert la création de pages dynamiques au moyen d'une DLL ActiveX compilée. Toutefois, la technique de transmission de données et de mise en oeuvre à laquelle elle a recours n'est performante et efficace que pour des taches mineures et les opérations effectuées sur les intranets (contrairement aux sites Internet à volumes élevés). Par ailleurs, là aussi, chaque modification apportée à la page ou marquage exige une nouvelle compilation de la DLL.

>>> Conclusion :

Qu'est ce qui détermine le choix d'une solution d'interfaçage entre un serveur web et un système de gestion de bases de données ?

la réalisation et l'exploitation d'un site web professionnel impliquent le couplage de compétences de deux acteurs principaux :

- L'administrateur système qui s'occupe de la configuration, de la gestion et de la maintenance de serveurs et de systèmes d'exploitation. Il veillera au bon fonctionnement du serveur web en fonction de la surcharge du processeur, il travaillera directement avec les webmaster pour alléger le site et faciliter son chargement.
- l'administrateur de base de données qui prend en charge la conception et, l'organisation et l'optimisation des bases de données déclarer sur le système.

le choix de la technologie de l'interfaçage web/SGBD dépendra de ces acteurs qui tiendront compte de :

- contexte d'exploitation
- contraintes techniques liées aux matériel, architecture et la plate-forme
- la taille de site, la nature des données
- la maintenance, l'évolution technologique, et l'accès rapide aux données

Perl fut le premier langage de script côté serveur. Depuis, d'autres ont été mis au point : **ASP** (**A**ctive **S**erver **P**age), **IDC** (**I**nternet **D**atabase **C**onnector, **PHP** (**P**ersonnel **H**ome **P**age),…

Il existe également des langages conçus pour certains types d'utilisateurs : par exemple, **TCL** facilite les calculs mathématiques complexes dans le domaine des sciences.

CHAPITRE 2. HTML

L'***Hypertext Markup Language***, généralement abrégé **HTML**, est le langage informatique créé et utilisé pour écrire les pages Web. HTML permet en particulier d'insérer des hyperliens dans du texte, donc de créer de l'hypertexte, d'où le nom du langage.

Techniquement, HTML est une application du Standard Generalized Markup Language (SGML). Le développement de HTML a été interrompu en 1999 au profit de celui du XHTML, qui est une application de l'Extensible Markup Language (XML). Le W3C a cependant relancé le développement HTML en 2007, suite notamment aux demandes des fabricants de navigateurs.

L'anglais *Hypertext Markup Language* est rarement traduit littéralement en **langage de balisage d'hypertexte**. On utilise généralement l'abréviation HTML, parfois même en répétant le mot « langage » dans **langage HTML**. *Hypertext* est parfois écrit *HyperText* pour marquer le *T* de l'abréviation HTML.

Tel qu'il a été pensé à ses origines par Tim Berners-Lee, le HTML ne sert pas à décrire le rendu visuel des pages Web (contrairement à la publication assistée par ordinateur), mais plutôt le sens des différentes parties du texte : titre, liste, passage important, citation, etc. Cette séparation du fond et de la forme n'a pas toujours été respectée au cours du développement du langage, comme en témoigne par exemple le balisage de style de texte, qui permet d'indiquer notamment la police de caractères souhaitée pour l'affichage, sa taille, ou sa couleur.

2.1 Les balises : principes

Le HTML utilise ce que l'on nomme des balises afin de structurer les informations et pour transformer votre code source en document correct affiché à l'écran. Connaître toutes les balises par coeur n'est heureusement pas nécessaire mais il faut en avoir bien compris le principe pour pratiquer le HTML. On a de plus tendance à utiliser un nombre de balises plus restreint qu'il n'y paraît.

Petits rappels : une balise est composée d'un mot clef entouré des symboles « < » et « > ». Par exemple <html> est une balise.

Les balises viennent en général par deux.

- Une ouvrante dont la syntaxe est décrite juste au dessus.
- Une fermante qui s'écrit comme l'ouvrante sauf que l'on fait précéder de mot clef par le caractère « / ».

Certaines balises sont dites vides, c'est-à-dire qu'elles ne contiennent pas d'autres éléments. Ces balises ne possèdent donc pas de balises de fermeture. Pour indiquer qu'il n'y a pas de balise de fin, en XHTML on ajoute le caractère « / » à la fin de la balise.

Lorsqu'une balise n'est pas vide, vous pouvez mettre différentes choses entre la balise d'ouverture et la balise de fermeture comme du texte ou d'autres balises.

Il est interdit de « croiser les balises » c'est-à-dire qu'il n'est pas permis de fermer une balise alors qu'une autre, ouverte après elle, n'est pas encore fermée. Il faut toujours faire attention à bien les emboîter.

Un grand nombre de balises prennent des attributs. Ils servent à paramétrer finement la sémantique d'une balise (par exemple en indiquant la cible d'un lien). Après le mot-clef, il suffit de mettre le nom de l'attribut suivi du symbole = et d'une valeur à donner à l'attribut, placée entre guillemets.

Ainsi dans la balise Lien vers IUT A :

- le mot-clef est « a »
- elle contient un attribut nommé « href » qui a pour valeur « http://www.iut-tlse3.fr ».

Les attributs sont toujours de la forme nom_de_l'attribut="valeur".

Il peut y avoir un nombre illimité d'attributs dans une balise. Bien sûr, comme pour les balises, il est inutile de connaître par coeur tous les attributs et toutes les valeurs.

2.2 Éléments de HTML

La version 4 de HTML décrit 91 éléments et 188 attributs. Certains attributs sont propres à un élément, d'autres s'appliquent à toute une série d'éléments et quelques attributs à tous les éléments. En suivant la spécification de HTML 4, les fonctionnalités implémentées par HTML peuvent être réparties ainsi :

Structure globale du document
: Au plus haut niveau, un document HTML est séparé entre un entête et un corps. L'entête contient les informations sur le document, notamment son titre. Le corps contient ce qui est affiché.

Informations sur la langue
: Il est possible d'indiquer la langue de n'importe quelle partie du document et de gérer le mélange de texte s'écrivant de gauche à droite avec du texte de droite à gauche.

Marquage sémantique
: HTML permet de différencier des contenus spécifiques tels que les citations d'oeuvres externes, les extraits de code informatique, des passages en emphase ou des sigles. Certains éléments HTML historiques ont cependant un sens mal déterminé, et sont en pratique très rarement employés (différenciation entre les éléments de *variable* et d'*exemple* de valeur dans un code informatique, par exemple, ou encore instance d'un terme défini dans le contexte).

Listes
: HTML différencie des *listes non ordonnées* et des *listes ordonnées*, selon que l'ordre formel du contenu dans le code est en soit ou non une information. Des *listes de définition* existent également, mais sans que leur champ d'application ne soit exactement déterminé

Tables
: Cette fonctionnalité a été développée pour permettre la présentation de données tabulaires mais a été immédiatement exploitée pour ses puissantes capacités de mise en page.

Hyperliens
: La fonctionnalité première de HTML.

Inclusion d'images, d'applets et d'objets divers

À l'origine HTML permettait seulement de donner des hyperliens sur les médias externes. L'invention d'éléments spécialisés pour le multimédia a permis l'inclusion automatique d'image, de musique, de vidéo, etc. dans les pages Web.

Style de la présentation
: Chaque élément, voire tout le document, peut se voir appliquer des styles. Les styles sont définis dans le document ou proviennent de feuilles de style en cascade (CSS) externes.

Marquage de présentation du texte
: Développé avant la généralisation de CSS pour fournir rapidement des fonctionnalités aux graphistes. D'usage désormais officiellement déconseillé pour la plus grande partie.

Cadres
: Aussi connu sous le nom de *frames*, une fonctionnalité souvent décriée qui permet d'afficher plusieurs documents HTML dans une même fenêtre.

Formulaire pour l'insertion interactive de données
: L'invention qui a permis l'apparition du commerce en ligne sur le Web.

Scripts
: Permet d'associer des morceaux de programmes aux actions des utilisateurs sur le document. Les langages utilisés sont généralement JavaScript et VBScript.

EXEMPLE

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>

 <head>
  <title>Titre affiché dans la barre de titre du navigateur</title>
 </head>

 <body>
  <!-- C'est ici que vous mettrez votre contenu  -->
 </body>

</html>
```

2.3 La définition de type de document

Il existe plusieurs version du HTML et avec ça plusieurs variantes. Cette première balise donc est la *déclaration de type document* (appelée couramment *doctype*). Vous avez ci-dessous une liste des Doctype les plus utilisés.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
   "http://www.w3.org/TR/html4/frameset.dtd"
```

2.4 Les balises Principales

Un document HTML est entièrement compris dans une balise html. Même si le navigateur s'y attend, vous êtes poli et vous lui dites que vous commencez votre document HTML, puis que vous le terminez. Ainsi la balise <html> sera toujours la toute première après le doctype, et la balise </html> la toute dernière.

À l'intérieur on trouve deux parties principales : un en-tête et un corps, placé respectivement entre les balises head et body. L'en-tête est constitué de déclarations générales concernant la page HTML, destinées au navigateur, aux moteurs de recherche etc. Le corps contient le document lui-même : ce qui sera affiché par le navigateur dans la fenêtre de rendu. Cette partie ne contient aucun élément obligatoire.

La balise <head>

La balise <head> délimite l'en-tête de la page dont on vient de parler. On y trouve des informations qui ne seront pas affichées directement dans la zone de rendu du navigateur. Par exemple le titre de la page, le lien vers la feuille de style, etc... L'en-tête des documents HTML est l'objet du chapitre *L'en-tête*.

La balise <title>

L'en-tête contient un élément obligatoire : title qui indique le titre de la page. C'est le titre qui s'affiche en suite en haut de la fenêtre du navigateur.

Essayez de mettre un titre pertinent et différent pour chaque page, qui permette d'identifier le site et la page en elle-même. Par exemple, "Sommaire" est un très mauvais choix. Quand votre page se retrouvera dans les favoris de quelqu'un, cette personne sera incapable de savoir de quelle page il s'agit rien qu'en regardant le nom. Préférez des choses du style "Accueil - www.ladressedemonsite.com".

La balise <body>

Tout le corps de notre document est dans la partie body. Elle comprend donc le texte, les liens, la référence des images et tout ce qu'un auteur peut vouloir mettre dans un document HTML.

Les commentaires

Les commentaires sont du texte écrit dans le code HTML qui n'est pas visible dans le rendu de la page. Les commentaires jouent habituellement le rôle de notes pour expliquer ce qui a été fait dans la page, ou bien tout simplement pour indiquer des modifications à faire ultérieurement. Ils sont biens sûr facultatifs, mais ils peuvent vous être utiles !

Un commentaire commence par les caractères <!-- et se termine par les caractères -->.

Pratiquement n'importe quelle chaîne de caractères peut être placée à l'intérieur d'un commentaire : du texte, des balises, etc

2.5 Un exemple à faire

Creation d'une première page.

Pour cela, ouvrez votre éditeur de texte préféré — nous parlons bien d'*éditeur* de texte et pas de *traitement* de texte. Sous Microsoft Windows, le Bloc-Note (Notepad) fait très bien l'affaire. Prenez le texte ci-dessous, copiez-le, et collez-le dans la page vide (ou bien tapez-le).

Exemple

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html>

 <head>
  <title>Premier essai</title>
 </head>

 <body>
  Bonjour le monde.
 </body>

</html>
```

Puis, utilisez la fonction **Fichier | Enregistrer sous**, et sauvez-le dans le fichier bonjour.html (avec Notepad, il faut choisir « **Tous les fichiers** » dans le menu déroulant **Type de fichier**). Si vous double-cliquez dessus depuis l'explorateur de disque (Explorateur Windows, Finder…), cela ouvre votre navigateur Internet par défaut, et affiche :

« Premier essai » dans la barre de titre ;
« Bonjour le monde. » dans la fenêtre principale.

Réessayez maintenant avec le texte suivant :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"> <html> <head> <title> Premier essai </title>
</head> <body> Bonjour le monde. </body> </html>
```

(le même texte mais sur une seule ligne, sans mise en forme) — une fois le fichier sauvé, il suffit d'appuyer sur le bouton « rafraîchissement/recharger » du navigateur pour voir la différence. On remarque qu'il n'y en a aucune.

Aucune différence en ce qui concerne le rendu, mais le code source est lui bien moins lisible. Donc moins facilement modifiable, augmentable, corrigeable. Il convient donc de prendre des « bonnes habitudes de programmation ».

2.6 Bonnes habitudes de programmation

Quelques conseils

- aérer son code :
  - si vous revenez à la ligne, cela correspond à une espace sur le rendu ; n'hésitez donc pas à revenir à la ligne ;
  - si vous mettez plusieurs espace, cela est interprété comme un seul espace, on peut donc jouer sur la « mise en forme » du code pour se repérer, et par exemple mettre un ou plusieurs espace en début de ligne (indentation) ; en général, quand un texte est entre une balise d'ouverture et une balise de fermeture, on décale les différentes lignes de 2 ou 3 espaces par rapport à ce qui précède ;
- mettez des commentaires pour pouvoir vous repérer ;
- utilisez un éditeur de texte avec
  - gestion des indentations : il suffit d'appuyer sur la touche de tabulation pour créer un décalage, et le décalage est appliqué automatiquement aux lignes suivantes ;
  - « coloration syntaxique » : les caractères spéciaux et balises sont reconnus et mis en couleur, ce qui facilite la lecture du code.

Voici par exemple un commentaire permettant de se repérer facilement (l'exemple suivant se trouverait au sein d'un code, donc notamment après les balises <!DOCTYPE …> <html> … <body> et avant le </body> </html> final).

```
…
<!--
*******************
* premiere partie *
*******************
-->

<h1> Première partie </h1>
…
```

Le mauvais exemple

Prenez votre traitement de texte favoris. Créez un document vide, et tapez simplement « Bonjour le monde. », puis enregistrez le fichier sous la forme d'un fichier HTML : menu **Fichier | Enregistrer sous**, et choisissez l'option **Page Web (*.htm, *.html)** dans le menu déroulant **Type de fichier**. Appelez ce fichier bonjour1.html.

Ouvrez maintenant ce fichier depuis l'éditeur de texte. Vous voyez que le fichier contient un nombre beaucoup plus important de lignes. Certaines de ces lignes peuvent contenir des informations personnelles, que vous aurez renseignées lorsque vous avez installé votre système d'exploitation, et que vous ne désirez peut-être pas voir figurer sur Internet… Et selon le logiciel (et sa version), vous aurez du code plus ou moins « propre » : dans certains cas, pour un texte long, il redéfinit à chaque paragraphe la police utilisée… Essayez de sauvegarder ainsi au format HTML un texte que vous avez déjà tapé auparavant et constatez les dégâts.

Vous remarquez aussi que vous n'avez pas pu définir le titre s'affichant dans la barre de titre.

S'il est simple de générer du code HTML, simple dans le sens « en peu d'opérations et sans connaissance particulières » (« en un clic »), il faut se méfier du résultat, même si le rendu est correct.

2.7 Avec quoi écrire un document HTML ?

Comme indiqué plus haut, il existe des éditeurs HTML plus développés, allant de l'amélioration de la présentation du code (exemple : les balises sont distinguées du texte par une couleur spécifique) à l'éditeur WYSIWYG (*« What You See Is What You Get »*, littéralement « ce que vous voyez est ce que vous obtenez », c'est-à-dire que vous voyez directement le résultat apporté par les modifications que vous entreprenez).

Vous devrez tout de même garder en tête la notion (récente) d'encodage des caractères, et faire la différence entre les principaux types (utf-8, ISO-8859-1…), et la nécessité pour votre éditeur de texte de reconnaître et respecter cet encodage, sous peine de voir afficher de drôles de caractères à la place des accents…

Voici une sélection (à compléter) d'éditeurs libres de qualité :

- Notepad++ Coloration syntaxique paramétrable, ouverture simultanée de plusieurs sources, support d'une quarantaine de langages, reconnaissance de l'encodage, macro, plugiciels…
- Bluefish
- JEdit
- Quanta Plus
- NVU. Ce dernier serait plutôt un éditeur WYSIWYG mais permet d'éditer directement la source d'une page.

Pour choisir, le mieux est de tester. Quelques éléments à prendre en compte pour faire un choix :

- Le logiciel permet-il la coloration syntaxique ? Quels langages sont supportés (php, css, html, javascript ?) ?
- Peut-on ouvrir plusieurs fichiers dans différents onglets ?
- Peut-on visionner simplement le résultat ? (par exemple avec une touche *voir cette page dans le navigateur*)
- Est-ce que les encodages de caractères sont bien gérés ?
- Y'a-t-il une autocomplétion ? (quand vous écrivez une balise : le logiciel écrit directement la balise fermante)
- L'indentation est elle facilement modifiable ? Notamment, est-ce que le logiciel comporte une fonction permettant de déplacer tout un bloc de ligne vers la gauche ou vers la droite ?

2.8 Programmation HTML L'en-tête

L'**en-tête** est la partie du fichier HTML comprise entre les balises <head>…</head>. Cette partie est située juste après le *doctype* et la balise d'ouverture <html>.

L'en-tête contient des informations sur la page elle-même (« méta-données ») : titre, auteur, description du contenu de la page, mots-clefs, où feuille(s) de style à utiliser…

Nous utilisons ici le XHTML. Pour le HTML, enlevez simplement les barres de fraction à la fin des balises isolées, par exemple

XHTML	HTML
<meta name="*propriété*" content="*attributs*" />	<meta name="*propriété*" content="*attributs*">

>>> Principaux éléments

Titre

Le titre est compris entre les balises <title>…</title>.
Habituellement, le titre de la page est affiché dans la barre de titre du navigateur (tout en haut), et lorsque le navigateur gère les onglets, dans l'étiquette des onglets.
Cet élément est obligatoire.

Encodage

Si le fichier contient des caractères répondant à la norme ISO-8859-1 « Latin-1 », on mettra la balise

```
<meta
  http-equiv="content-type"
  content="text/html; charset=ISO-8859-1" />
```

S'il s'agit de la norme ISO-8859-15 « Latin-9 » :

```
<meta
  http-equiv="content-type"
  content="text/html; charset=ISO-8859-15" />
```

Cette information est recommandée.

Auteur

Le nom de l'auteur s'indique avec la balise suivante :

```
<meta name = "author"
  content = "nom de l'auteur" />
```

Description de la page

La description du contenu de la page s'indique avec la balise suivante :

```
<meta name = "description"
  content = "phrase de description" />
```

Mots-clefs

Les mots-clefs servent à référencer la page. Cependant, de nombreux moteurs de recherche n'ont plus recours aux mots-clefs car des auteurs utilisaient des mots-clefs sans rapport avec le contenu afin d'augmenter la fréquentation de leur page (cas notamment de nombreux sites pornographiques). Les mots-clefs s'indiquent avec la balise suivante :

```
<meta name = "keywords"
  content = "liste de mots-clefs" />
```

Feuille de style

Lorsque la page utilise une feuille de style située dans un autre fichier, on utilise la balise

```
<link rel= "stylesheet"
  type = "text/css"
  href="nom_du_fichier.css"/>
```

On peut aussi écrire la feuille de style (code CSS) directement dans l'en-tête, entre les balises <style type="text/css">…</style>.

>>> Balise <meta>

La balise <meta> a été utilisée plusieurs fois ci-dessus. Vous avez remarqué que sa syntaxe générale était :

<meta name="*propriété*" content="*attributs*" />

Dans certains cas, on peut utiliser http-equiv à la place de name. Dans ce cas-là, lorsque le fichier est retiré par le protocole HTTP (c'est-à-dire en gros lorsqu'il est lu *via* le Web et non pas en local), peut s'en servir pour créer un en-tête suivant les spécifications du RFC 822.

2.9 Programmation HTML : Caractères Spéciaux

>>> Définition

HTML utilise un jeu d'**entités** pour incorporer des **caractères spéciaux** dans le document. Plus simplement, vous tapez une sorte de mot spécial (oui c'est ça, un mot magique), et miraculeusement il se transforme en un joli caractère quand vous affichez la page dans un navigateur. Ces entités ont toutes la même tête : elles sont précédées d'une *esperluette* « & » et suivies d'un *point-virgule* « ; ».

Il est possible de recourir à deux types d'entités :

- les entités numériques composées d'un nombre précédé du caractère *croisillon* # (souvent appelé à tort « dièse »[1]) entre l'*esperluette* et le *point-virgule* ;
- les entités caractères composées d'une chaîne de caractères entre l'*esperluette* et le *point-virgule*.

Ainsi il est possible d'écrire le signe euro (€) de deux manières :

- € qui en est l'entité numérique ;
- € qui en est l'entité caractère.

On peut aussi taper l'entité numérique en hexadécimal, en mettant un « x » entre le croisillon et le nombre. Par exemple, Ä est la même chose que Ä, c'est-à-dire « Ä ».

>>> liste des entités

Quelques exemples d'entités

Caractère spécial	Entité caractère	Entité numérique	Mnémotechnique
Lettres spéciales (diacririquées, ligaturées, non romaines)			
É	É	É	E (accent) aigu *(acute)*
Ò	Ò	Ò	O (accent) grave
Â	Â	Â	A (accent) *circ*onflexe
Ã	Ã	Ã	A tildé
Ä	Ä	Ä	A *umlaut* (inflexion allemande, marquée par un tréma)
Å	Å	Å	*ring* = anneau

Æ	Æ	Æ	A et E ligaturés (E dans l'A)
æ	æ	æ	a et e ligaturés (e dans l'a)
Œ	Œ	Œ	O et E ligaturés (E dans l'A)
œ	œ	œ	o et e ligaturés (e dans l'o)
Ç	Ç	Ç	C cédille
Γ	Γ	Ɖ	gamma capitale (lettre grecque)
γ	γ	γ	gamma minuscule (lettre grecque)
Caractères typographiques			
—	—	—	*M dash* (tiret de la largeur d'un M, tiret long, tiret d'incise, tiret cadratin)
–	–	–	*N dash* (tiret de la largeur d'un N, tiret moyen, tiret d'intervalle, tiret demi-cadratin)
·	·	·	*middle dot* (point centré, virgule géorgienne)
espace insécable[2]			*non breakable space*
espace fine[3]			*thin space*
«	«	«	*left angle quote* (guillemet angulé gauche, ouvrant)
»	»	»	*right angle quote* (guillemet angulé droit, fermant)
•	•	•	*bullet* (puce)
…	…	…	*horizontal ellipse* (points de suspension)
Symboles commerciaux et mathématiques			
™	™	™	*trade mark* (marque commerciale)
®	®	®	*registred* (marque déposée)
©	©	©	*copyright* (tous droits réservés)
€	€	€	
²	²	²	*superior 2* (2 en exposant, carré)
½	½	½	fraction 1/2
±	±	±	*plus minus* (plus ou moins)
←	←	←	*left arrow* ; voir aussi → (*right* →), ↑ (*up* ↑), ↓ (*down* ↓), ↔ (*horizontal* ↔), ↵ (*carriage return arrow*, « retour chariot » ↵)
□	⇒	⇒	*right big arrow* ; voir aussi ⇒ (*right* □), ⇑ (*up* ⇑), ⇓ (*down* ⇓), ⇔ (*horizontal* □)
×	×	×	*times* = fois
⋅	⋅	⋅	*scalar multiplication dot* (signe de multiplication scalaire)
÷	÷	÷	*divide* = diviser
√	√	√	radical (racine carrée)
‰	‰	‰	*per* (« pour » en latin) mille
□	∝	∝	proportionnel

≃		&#8771;	
¦	&brvbar;	&#166;	*broken vertical bar* (barre verticale interrompue, tube)

Exemple

```
<em>Dungeons &amp;amp; Dragons<em>&amp;trade;
est le premier jeu de rôle de l'histoire&amp;nbsp;;
c'est aussi le plus joué.
```

donne

Dungeons & Dragons™ est le premier jeu de rôle de l'histoire ; c'est aussi le plus joué.

2.10 Programmation HTML Titres et paragraphes

> Les paragraphes

>> La balise p

La balise p sert à créer un paragraphe. Habituellement, il est isolé par défaut du reste du texte par une petite espace verticale avant et après (typographie anglaise[1]), mais vous pouvez changer ça avec les fameux CSS. De plus, le texte va automatiquement à la ligne à la fin d'un paragraphe.

Exemple

```
<body>

   <p>Voici un paragraphe.</p>
   <p>En voici un deuxième.</p>

</body>
```

ce qui donne

Voici un paragraphe.

En voici un deuxième.

>> Retours à la ligne

Les retours de ligne sont interprétés comme une espace. Pour placer un retour de ligne au sein d'un paragraphe, on utilise la balise
[2] *(break)*.

L'utilisation de plusieurs balises <br /> successives est à proscrire ; il s'agit d'un retour de ligne et pas d'un saut de ligne. Rappelez-vous que le HTML concerne la description de contenu et pas la mise en forme. Pour définir une espace vertical entre deux paragraphe, il faut avoir recours au CSS ; par exemple, pour un paragraphe ponctuel, on pourra utiliser

```
<p style="margin-top:20px">
…
<p>
```

pour avoir une espace de 20 pixels (cet espacement ne change pas si l'on modifie la taille de la police), ou bien

```
<p style="margin-top:2em">
…
<p>
```

pour avoir une espace de 2 fois la largeur du caractère M (cet espacement est proportionnel à la taille de la police).

>> Ligne de séparation

Il est possible de mettre une ligne de séparation entre deux paragraphes, avec la balise <hr />[3] *(horizontal ruler)*.

>> Les titres : balises H

Les titres et le sous titres sont indiqués par la balise h1 à h6 (*heading* — six niveaux de titres sont possibles).

Le niveau 1 est le niveau le plus haut dans la hiérarchie du document, suivi du niveau 2, etc.

Exemple

```
<body>

 <h1>Un titre de niveau 1 (un gros titre)</h1>
  <p>Un paragraphe.</p>

  <h2>Un titre de niveau 2 (un sous titre)</h2>
   <p>Un paragraphe.</p>

   <h3>Un titre de niveau 3 (un sous-sous titre)</h3>
    <p>Etc.</p>

</body>
```

ce qui donne quelquechose comme

Un titre de niveau 1 (un gros titre)

Un paragraphe.

Un titre de niveau 2 (un sous titre)

Un paragraphe.

Un titre de niveau 3 (un sous-sous titre)

Etc.

2.11 Programmation HTML Style de texte

Styles de texte les plus courants

Style	Balise	Mnémotechnique	Rendu par défaut
mise en emphase	…	emphase	italique
mise en emphase forte	…	« fort »	gras
citation d'une référence	<cite>…</cite>	…	italiques
citation courte dans le texte	<q>…</q>	*quote* (citation)	entre guillemets
citation à part	<blockquote>…</blockquote>	« bloc de citation »	marge à gauche plus grande
code source	<code>…</code>	…	police à chasse fixe avec empattement (type Courier)
texte préformaté (par exemple pour aligner avec des espaces, ou faire des dessins ASCII)	<pre>…</pre>	*preformed* (préformatté)	police à chasse fixe, en général avec empattement (type Courier)

Certains ont tendance à utiliser les balises … pour l'italique et … pour le gras *(bold)* à la place de em et strong. Dans l'absolu, cela revient à faire de la mise en forme et non de la description de texte : c'est déconseillé. Dans la pratique, si cela ne change pas grand chose pour un rendu graphique (sauf si la feuille de style ou le rendu du navigateur en décide autrement), cela pose un problème d'accessibilité : s'il possible pur un lecteur d'interpréter une mise en emphase (par exemple en changeant le ton de la voix), il est en revanche impossible de rendre une variation de mise en forme.

Exemple

```
<p>
  Selon Boileau :
  <blockquote>
    Vingt fois sur le métier
    remettez votre ouvrage
  </blockquote>
  bref, <em>ne vous découragez pas !</em>
  <strong>Soyez patient !</strong>
</p>
```

ce qui donne

Selon Boileau :

Vingt fois sur le métier remettez votre ouvrage

bref, *ne vous découragez pas !* **Soyez patient !**

Voici quelques autres styles de texte :

Autres styles de texte			
Style	**Balise**	**Mnémotechnique**	**Rendu par défaut**
saisie clavier	<kbd>…</kbd>	*keyboard* (clavier)	*idem* <code>
abréviation	<abbr>…</abbr>	*abbreviation*	souligné par un trait pointillé
acronyme	<acronym>…</acronym>	…	*idem* abbr

Note

En français, un acronyme est lexicalisé, c'est-à-dire prononcé comme un mot (comme laser, ovni), alors que dans un sigle, les lettres sont prononcées séparément (SNCF). En anglais, langue de référence du HTML, la définition est plus floue : certains distinguent les termes *initialism* (sigle) et *acronym* (sigle lexicalisé, même sens qu'en français), alors que pour d'autre *acronym* désigne un sigle qu'il soit lexicalisé ou non (voir l'article du Wikipédia anglophone *Acronym*). Dans la pratique, il n'y a pas de différence dans le traitement de <abbr> et de <acronym>, cette dernière disparaît d'ailleurs du XHTML 2.

Même lorsqu'ils ne modifient pas la mise en forme, les balises abbr et acronym sont utiles pour les analyseurs syntaxiques *(parsers)* et les correcteurs d'orthographe. Ils sont surtout utiles avec le paramètre title (titre), qui permet d'expliciter l'abréviation ; en général, le titre s'affiche dans une info-bulle.

Exemple

```
Les pages Web sont écrites en <acronym title="Hypertext Markup Language">HTML</acronym>.
```

Donne

Les pages Web sont écrites en HTML

>>> Paramètres

Pour les balises de citation (<q> et <blockquote>), on peut indiquer la source, avec le paramètre cite suivi de l'adresse réticulaire.

Exemple

```
Selon le site Alsacréations,
<q cite="http://css.alsacreations.com/Bases-et-indispensables/Quelle-est-la-difference-entre-un-div-et-un-calque">
  cet abus de langage
  est malheureusement demeuré très ancré
  et induit de nombreux amalgames.
</q>>
```

>>> Balises de mise en forme

Avant le CSS, la modification de la police était déjà une préoccupation, le W3C a donc créé des balises permettant ces modifications — donc de la mise en forme —, on peut donc toujours trouver de telles balise dans du code ou des ouvrages. Les balises sont donc données à titre d'information, mais elles devraient être proscrites au profit du CSS.

Autres styles de texte		
Style	**Balise**	**Mnémotechnique**

affichage écran (*idem* code source)	<tt>…</tt>	*teletype* (terminal)
italique	…	*italic*
gras	…	*bold* (gras)
grandes lettres	<big>…</big>	(grand)
petites lettres	…	(petit)
texte barré	<strike>…</strike> ou <s>…</s>	(barré)
texte souligné	<u>…</u>	*underligned* (souligné)

On peut également modifier :

- la taille de la police : …, ou *taille* est un nombre absolu en unité arbitraire (la taille normale est 3), ou un nombre relatif (+1 pour la taille courante augmentée de 1, -2 pour la taille courante diminuée de 2) ;
- la nature de la police : ….

On peut combiner les deux, par exemple …

Ces balises sont maintenant déconseillées. On peut utiliser em et strong pour l'italique et le gras, et le CSS pour le reste :

- lettres plus grandes de 10 % : …
- lettres plus petites de 10 % : …
- souligner : …
- barrer : …
- type de police : <span style="font-family:"Times New Roman" ">…

mais il vaut mieux définir des styles dans un fichier à part et faire appel à ces styles (ce qui simplifie la maintenance des fichiers), par exemple, mettre dans le fichier CSS :

```
.grand {font-size:1.1em}
.petit {font-size:0.9em}
.souligner {text-decoration:underline}
.barrer {text-decoration:linethrough}
```

ce qui s'exploite dans le fichier HTML de la manière suivante :

```
<p>
 Du texte en <span class="grand">grandes</span>
 ou <span class="petit">petites</span> lettres,
 <span class="souligner">souligné</span>
 ou <span class="barrer">barré</span>.
</p>
```

2.12 Programmation HTML Liens

Un lien (également appelé « lien hypertexte », d'où le nom d'HTML) est un mot ou groupe de mots permettant, après avoir cliqué dessus, d'ouvrir un fichier. Si ce fichier est une page HTML ou un fichier texte (.txt ou autre), il s'affichera dans le navigateur à la place ou en plus de la page actuelle. S'il s'agit d'un autre type de fichiers, le navigateur demandera généralement de sélectionner une application pour ouvrir le fichier ou d'enregistrer le fichier. Il est possible que l'utilisateur ait installé une extension *(plug-in)* pour pouvoir lire certains types de fichiers.

Note : certains formats sont devenus si courants que la plupart des navigateurs les supportent sans que l'on ne doive installer d'extension. C'est le cas *notamment* des formats d'images les plus courants (.bmp, .gif, .jpg, .png, .tif, etc). Cela dépend évidemment du navigateur utilisé et surtout de sa version. Les plus récents supportent généralement aussi XML.

Note : certains navigateurs sont livrés avec les extensions gérant les formats les plus courants, comme Java ou Flash.

Note : l'installation de certains logiciels entraîne le support de certains types de fichiers sur certains navigateurs. C'est le cas notamment d'Adobe Reader 6.0.

>> Balise

Pour créer un lien en html, on utilise les balises <a> et pour encadrer le contenu du lien (qui peut être du texte, une image, etc). La balise <a> dispose de trois attributs possibles :

- href : permet d'indiquer l'adresse du lien ;
- name : permet de définir une ancre ;
- target : permet de définir la cible du lien .

```
<a href="http://www.irit.fr">Lien vers IRIT</a>
<a name="ancre_1">Définition d'une ancre</a>
<a href="http://www.irit.fr" target="_blank">Lien ouvrant IRIT dans une nouvelle fenêtre</a>
```

>>Les adresses réticulaires

Les adresses réticulaires (URL, pour *uniform ressource locator*)[1] indiquent où aller chercher les ressources ; par ressource, on entend autre page Web, mais aussi autre type de fichier : image, son…

Si le fichier à aller chercher se trouve dans le même répertoire (dossier) que la page Web que l'on écrit, on se contente de donner le nom du fichier. On a ici deux type d'adresses : les adresses relatives et les adresses absolues.

>> Liens absolus

Un lien absolu indique l'adresse complète du fichier lié (de http:// à l'extension ou à l'extension du domaine). Il est généralement utilisé pour afficher une page d'un autre site.

Exemple :

```
<a href="http://www.iut-tlse3.fr">Une formation vraiment géniale</a>
```

L'adresse absolue peut être utilisée lorsque l'objet se trouve sur son propre site, à condition que l'on soit sûr qu'il ne bougera pas ; par exemple, toutes les images sont mises dans un répertoire /images qui ne sera jamais déplacé.

L'utilisation de l'adresse absolue est obligatoire lorsque l'on fait un lien vers un objet situé sur un autre site Web.

>> Liens relatifs

Un lien relatif indique l'adresse du fichier **par rapport** à la page actuelle. Il est vivement conseillé de l'utiliser le plus souvent dans son site web, car cela permet de changer d'hébergeur sans devoir rééditer chaque lien.

Si le fichier se trouve dans le même répertoire que la page courante, il suffit d'indiquer le nom du fichier (avec son extension). Si le fichier se trouve dans un sous-répertoire du répertoire contenant la page courante, il suffit d'indiquer le nom du répertoire, suivi d'une barre oblique « / » et du nom du fichier.

Exemples :

Lien vers le fichier « exemple.html » se trouvant dans le même répertoire que la page actuelle

Lien vers le fichier « exemple.html » se trouvant dans le répertoire « repertoire_exemple » qui se trouve dans le même répertoire que la page actuelle

Si le fichier se trouve dans le répertoire parent du répertoire actuel, ou dans un sous-répertoire du répertoire parent, il faut « remonter » à l'aide de deux points « .. ».

Exemple :

Lien vers le fichier « exemple.html » se trouvant dans le répertoire parent du répertoire dans lequel se trouve la page actuelle

L'utilisation de l'adresse relative est intéressante si l'objet visé est situé dans une branche de l'arborescence qui restera toujours solidaire. Si l'on déplace la branche (dnc la page Web courante *et* l'objet), l'adresse relative reste valable.

Par contre, si la page Web change de répertoire et pas l'objet cible, l'adresse relative devient erronée.

>> Liens vers une ancre

Une ancre permet d'afficher une page web à partir d'un certain point (très utile dans les longues pages web). Ainsi, ce lien affiche le chapitre « Liens » du livre « HTML » à partir de la section « Liens vers une ancre ».
Pour créer une ancre, il faut utiliser la balise <a> avec l'attribut name, comme ceci (Il n'est pas nécessaire d'inclure du texte entre les deux balises) :

Lorsque l'ancre est vide, elle ne doit cependant pas être écrite sous la forme <a name="section_5" />, formellement valide en XML, mais déconseillée par les normes XHTML et XML pour des raisons de compatibilité avec différents navigateurs.

Pour créer un lien vers une ancre, il suffit de rajouter un croisillon (#) suivi du nom de l'ancre après l'adresse (absolue ou relative) de la page. Si l'ancre se trouve sur la page actuelle, il ne faut pas noter l'adresse de la page.

Exemples :

Lien vers l'ancre n°1 de la page "exemple.html"
Lien vers l'ancre n°1 de la page actuelle

>> Comment ouvrir un lien dans une nouvelle fenêtre ?

L'attribut target permet de définir la cible du lien, c'est-à-dire l'endroit où sera ouvert le fichier du lien. Pour ouvrir le fichier dans une nouvelle fenêtre, assigner la valeur _blank à l'attribut target. Exemple :

Lien ouvrant Wikilivres dans une nouvelle fenêtre

>> Autres types de liens

Nous avons vu que l'attribut href peut contenir autre chose que l'adresse d'une page Web. S'il contient un nom de fichier autre que HTML, cela télécharge le fichier, ou éventuellement cela ouvre le fichier avec un programme dédié (par exemple Adobe Acrobat Reader pur un fichier PDF).

Mais l'attribut href peut faire référence à des ressources disponibles par un autre protocole que le HTTP :

- transfert de fichier (protocole FTP) : par exemple href="ftp://ftp.gnu.org/".
- forum usenet (protocole NNTP) : par exemple href="news:fr.comp.infosystemes.www.auteurs".
- adresse de courriel (protocole SMTP) : par exemple href="mailto:jeveuxduspam@monfournisseur.fr".
- adresse de messagerie instantanée (protocole XMPP/Jabber) : href="xmpp:jeveuxduspim@monserveurjabber.fr".

2.13 Programmation HTML -Images

Une image est insérée avec la balise img en spécifiant avec l'attribut src (source) le chemin de l'image à inclure (sous la forme d'un adresse réticulaire, URL) et avec l'attribut alt l'éventuel texte de remplacement de l'image :

HTML

<img src="mon_image.png" alt="texte si l'image n'est pas affichée">

La balise img peut avoir divers attributs facultatifs. Seuls les attributs src et alt sont obligatoires.

- align : alignement de l'image (peut prendre les valeurs top, bottom, middle, left ou right) ;
- alt : texte de remplacement (affiché si le navigateur ne peut afficher l'image) ; cet attribut est obligatoire, il permet notamment à un malvoyant de comprendre l'image qu'il ne peut voir (son logiciel lit le texte à voix haute ou le transforme en braille) ;
- title : titre de l'image (s'affiche en info bulle dans le navigateur) ;
- border : largeur de la bordure de l'image (valeur exprimée en pixels) ;
- height : hauteur de l'image (si aucune valeur spécifiée, l'image garde sa hauteur normale) ;
- width : largeur de l'image (si aucune valeur spécifiée, l'image garde sa largeur normale).

>> L'attribut SRC

L'attribut src contient donc l'adresse à laquelle on va chercher l'image (voir le chapitre Liens). On peut utiliser une adresse absolue ou relative.

Il faut cependant proscrire l'inclusion des objets (images, sons, vidéo) situés sur d'autres sites, ce que l'on appelle des « liens à chaud » (hot links). En effet, cela génère du traffic sur le serveur cible au lieu de son propre serveur, ce que le serveur cible peut considérer comme du « vol de bande passante ». Par ailleurs, le fait que l'on utilise l'objet original n'implique pas un respect du droit d'auteur : certes on n'a pas fait de copie de l'objet, mais on utilise l'objet, et cela doit se faire avec l'accord de l'auteur et/ou de ses ayants droit.

Donc, si par exemple notre page web est http://www.monsite.com/mapage.html et que l'on veut inclure l'image http://www.autresite.com/images/img1.png, les deux solutions acceptables sont :

- demander l'autorisation au webmaster du site http://www.autresite.com/, importer l'image sur son propre site et l'exploiter en local, en affichant l'origine de l'image et les conditions d'utilisation imposées par le site d'origine ;

  par exemple, si l'on place l'image en http://www.monsite.com/image/img1.png, on peut écrire

```
<img src="/image/img1.png" … /> <br />
Reproduit avec l'aimable autorisation de
<a href="http://www.autresite.com/">Autresite.com</a>,
tous droits réservés
```

- ne pas inclure l'image mais faire le liens vers cette image, ou mieux vers la page Web contenant cette image ; par exemple

```
Voir l'illustration sur l'article dédié
du site <a href="http://www.autresite.com/lapage.html">Autresite.com</a>
```

>> L'attribut alt

Il est nécessaire de donner, pour chaque image, un texte de remplacement pour les navigateurs ne supportant pas l'affichage d'images ou pour des raisons d'accessibilité des pages web (cf normes W3C). Le texte de remplacement est spécifié par l'attribut alt :

Lorsqu'une image ne véhicule pas d'information, ce texte doit être vide (n'oubliez pas que le alt est lu ou rendu en braille pour un malvoyant), l'attribut est alors présent sous la forme :

>> L'attribut title

Il est possible d'attribuer un titre à chaque image en plus de l'information alternative (alt). L'attribut title doit contenir un information optionnelle sur l'image, ou reproduire l'attribut alt: Les navigateurs affichent cette information dans un info bulle :

Remarque

L'attribut alt et title peuvent être cumulés.

Les attributs height et width

Si l'on omet ces attributs, l'image s'affiche en grandeur réelle (pixel pour pixel).

Si l'on indique une des dimensions (la largeur ou la hauteur), l'autre dimension est calculée pour que l'image s'affiche avec ses proportions originales.

Le fait d'indiquer les deux dimensions peut faciliter l'affichage : les images sont chargées après le texte, le navigateur peut ainsi réserver la place nécessaire, sinon, il doit décaler le texte au fur et à mesure que les images sont chargées. Il peut donc être intéressant d'indiquer les dimensions même si l'on utilise les dimensions originales de l'image. Cependant, l'auteur doit bien s'assurer que les proportions (rapport largeur/hauteur) sont respectées, sous peine que l'image s'affiche déformée.

2.14 Programmation HTML Listes

Les listes permettent d'ordonner une énumération. Même si on les rencontre rarement, elles ont cet avantage qu'elles ont été crées spécialement pour faciliter une opération qu'effectue souvent tout créateur de page web et qui consiste à dresser une liste d'éléments.

>> Types de liste

Il existe deux listes dont la structure est similaire : une tenant compte de l'ordre et pas l'autre et une autre structure par groupe de deux éléments.

>> Listes non numérotées

Les listes non numérotées sont introduites par la balise ul (pour *unordered list*, « liste non ordonnée ») et chaque élément de la liste par li (*list item*, « élément de liste ») :

```
<ul>
   <li>premier élément de la liste ;</li>
   <li>deuxième élément ;</li>
```

```
   <li>etc.</li>
</ul>
```

Les balises fermantes pour les éléments li ne sont pas obligatoires. Ce code donnera quelque chose comme :

- premier élément de la liste ;
- deuxième élément ;
- etc.

>> Listes numérotées

Les listes numérotées fonctionnent sur le même principe que les listes non numérotées. La seule différence est qu'on utilise la balise ol (*ordered list*, « liste ordonée ») au lieu de la balise ul. Les éléments restent encadrés par les balises li.

Exemple :

```
<ol>
   <li>premier élément de la liste ;</li>
   <li>deuxième élément ;</li>
   <li>etc.</li>
</ol>
```

Ce code donnera :

1. premier élément de la liste ;
2. deuxième élément ;
3. etc.

Listes de descriptions

Contrairement aux autres types de listes, les listes de description, dl *(definition list)*, n'utilisent pas la balise li pour les éléments de la liste, mais les sépare en deux parties : le terme, introduit par dt *(definition term)* et sa description introduite par dd *(definition description)*. Par exemple :

```
<dl>
   <dt>C</dt>      <dd>Un langage efficace</dd>
   <dt>Java</dt>   <dd>Un langage portable</dd>
   <dt>Pascal</dt> <dd>Un langage pédagogique</dd>
</dl>
```

Ce code donnera :

C
        Un langage efficace
Java

Un langage portable

Pascal

Un langage pédagogique

2.15 Programmation HTML Tableaux

>> Balises de bases

La déclaration d'un tableau se fait grâce à la balise table.

- La balise tr (**t**able **r**ow) permet de définir une ligne, une rangée dans un tableau. Cette balise encadre les balises td de cellules.
- La balise td (**t**able **d**ata) permet de définir une cellule de donnée dans une ligne. Elle sera contenue obligatoirement dans la balise tr.
- La balise th (**t**able **h**eader) permet de définir des cellules comme étant des entêtes de lignes ou de colonnes. Généralement, le texte sera mis en emphase suivant les styles par défaut du navigateur ou de l'utilisateur.

Exemple élémentaire

```
<table border="1" >

  <caption>Table simple</caption>

  <tr>          <!-- ligne 1 -->
    <th> a1 </th> <!-- case 1 -->
    <th> a2 </th> <!-- case 2 -->
  </tr>

  <tr>          <!-- ligne 2 -->
    <td> b1 </td> <!-- case 1 -->
    <td> b2 </td> <!-- case 2 -->
  </tr>

</table>
```

donne

Table simple

a1	a2
b1	b2

>> Balises de groupement

Les balises de groupement permettent comme leur nom l'indique de grouper des cellules entre elles suivant leur nature. Il est possible de créer des groupes de lignes ou de colonnes. La définition correcte de ces groupes permet entre autres une définition plus aisée des styles.

>>> Lignes

- La balise thead permet de grouper les lignes d'entête. Elle est généralement utilisée conjoitement à la balise th.

- La balise tbody permet de grouper les lignes du "corps" c'est-à-dire de la partie principale du tableau.

- La balise tfoot permet à l'instar des deux précédentes de grouper les lignes du pied de tableau. Elle est par exemple utilisée lorsque, dans le cas d'un tableau très long, les titres sont reproduits à la fin dans le but d'obtenir une meilleur lisibilité.

>>> Colonnes

Les balises col colgroup permettent de définir un groupe de colonnes dans le but d'appliquer un style ou une taille précise à une ou plusieurs colonnes. Elles s'utilisent de cette façon :

```
<colgroup [attributs] >
   <col [attributs] />
   <col [attributs] />
</colgroup>
```

>> Fusionner des cellules

Il est possible de fusionner des cellules entre elles en utilisant des attributs de la balise td.

La fusion de plusieurs cellules d'une même ligne, c'est-à-dire de plusieurs colonnes, se fait grâce à l'option colspan et la fusion de plusieurs cellules appartenant à une même colonne, c'est-à-dire de plusieurs lignes, avec l'option rowspan. Ces deux attributs prennent la valeur correspondant aux nombres de cellules que l'on souhaite fusionner. Par exemple pour fusionner trois cellules d'une colonne :

```
<table>
   <tr>
      <td colspan="3">triple cellule</td>
   </tr>
   <tr>
      <td>cellule simple</td>
      <td>deuxième cellule simple</td>
      <td>troisième cellule simple</td>
   </tr>
</table>
```

Et voilà le résultat :

triple cellule		
cellule simple	deuxième cellule simple	troisième cellule simple

Pour fusionner trois cellules d'une colonne, c'est un peu plus complexe toujours à cause de la construction linéaire des tableaux.

```
<table>
   <tr>
<!-- La première cellule des trois prochaines lignes seront fusionnées avec la suivante -->
      <td rowspan="3">cellule 1</td>
      <td>cellule simple</td>
      <td>cellule simple</td>
```

```
        <td>cellule simple</td>
    </tr>
    <tr>
<!-- La première cellule de cette ligne étant fusionnée avec la précédente,
elle ne doit pas être définie -->
        <td>cellule simple</td>
        <td>cellule simple</td>
        <td>cellule simple</td>
    </tr>
    <tr>
<!-- La première cellule de cette ligne étant fusionnée avec la précédente,
elle ne doit pas être définie -->
        <td>cellule simple</td>
        <td>cellule simple</td>
        <td>cellule simple</td>
    </tr>
</table>
```

cellule 1	cellule simple	cellule simple	cellule simple
	cellule simple	cellule simple	cellule simple
	cellule simple	cellule simple	cellule simple

2.16. Programmation HTML Formulaires

Un formulaire permet de faire entrer des données à un utilisateur sur une page web ; données qui seront ensuite traitées par un script serveur ou client. Un formulaire est composée de un ou plusieurs éléments d'entrée englobés par la balise <form>. Par exemple :

```
<form method="post">
    <p>Entrez votre nom : <input type="text" name="nom" /></p>
    <p>Entrez votre prénom : <input type="text" name="prenom" /></p>
</form>
```

Attention : le HTML seul ne permet pas de récupérer (pour un humain) ou traiter les données du formulaire. Pour cela, il faut utiliser un langage de script, comme PHP. Par souci de compréhension, nous supposons dans nos exemples qu'un script récupère les données.

>> La balise <form>

<form> permet de regrouper plusieurs entrées sous un seul nom, ce qui permettra de les traiter ensemble. Exemple :

```
<form name="formulaire_1" method="post" action="traitement.php">
    <p>Entrez votre nom : <input type="text" name="nom" /></p>
    <input type="submit" value="Envoyer 1" />
</form>
<form name="formulaire_2" method="post" action="traitement.php">
    </p>Entrez votre prénom : <input type="text" name="prenom" /></p>
    <input type="submit" value="Envoyer 2">
</form>
```

Ici, si vous cliquez sur « Envoyer 2 », les données saisies dans « nom » ne seront pas récupérées car le bouton « Envoyer 2 » ne porte que sur le formulaire dans lequel il se trouve (en l'occurence « formulaire_2 »).

>> La balise <input>

<input> est la balise le plus utilisée dans les formulaire. Elle permet par exemple de demander à l'utilisateur de la page des informations textuelles (par exemple son nom), un choix entre plusieurs options ou même de sélectionner un fichier à envoyer.

>>> L'attribut name

Cet attribut sert à retrouver un objet (ici la balise <input>) afin de l'exploiter en javascript. Il est également utilisé lors de l'envoi du formulaire vers un serveur afin d'extraire les données saisies par l'utilisateur.

>>> L'attribut id

Comme l'attribut name, id sert à nommer un objet dans une page web. Cependant, contrairement à l'attribut name, tous les id doivent être uniques dans toute la page. Ils servent également à la balise <label> que nous présenterons plus loin.

>>> L'attribut type

Le role de la balise est déterminée via l'attribut type. Voici une liste des possibilités qui s'offrent à vous :

text
: C'est un champ de saisie de texte classique.

password
: Ce type permet d'entrer un mot de passe. Le texte saisi est remplacé par des étoiles (le caractère « * »). Attention, avoir un champ caché à l'utilisateur ne signifie pas que personne pourra le lire car le mot de passe est transmis « en clair » sur le réseau.

checkbox
: C'est une simple case à cocher permettant à l'utilisateur de valider ou non une option.

radio
: C'est une case d'option. Elle n'est jamais utilisée seule car, en groupe, elle permet à l'utilisateur de choisir une option parmi plusieurs.

file
: Ce type permet à l'utilisateur de choisir un fichier afin qu'il soit envoyé vers un serveur.

submit
: C'est un simple bouton permettant d'envoyer le formulaire.

reset
: C'est un bouton permettant de remettre le formulaire dans son état initial.

hidden
: C'est un champ caché qui permet à l'auteur du formulaire de faire passer des informations au serveur qui n'ont pas à être modifiées par l'utilisateur. Attention, même si l'utilisateur ne peu pas voir directement ces champs, leur valeur est aisément modifiable. Il ne faut donc JAMAIS faire confiance aux données venant de ces champs lors du traitement du formulaire.

name

cette attribut permet d'inserer une variable dans le texte.

>>> L'attribut value

Cet attribut permet de spécifier la valeur par défaut d'un champ.

Exemple :

```
<form name="formulaire" method="post">
  <p>Un champ texte : <input type="text" name="nom" id="nom"/> </p>
  <p>Un mot de passe : <input type="password" name="pass" id="pass" /> </p>
  <p><input type="checkbox" name="choix_simple" id="choix_simple" /> Une case à cocher </p>
  <p><input type="radio" name="choix_multiple" value="choix1"/> Premier choix </p>
  <p><input type="radio" name="choix_multiple" value="choix2"/> Deuxième choix </p>
  <p><input type="radio" name="choix_multiple" value="choix3"/> Troisième choix </p>
  <p>Un fichier à mettre sur le serveur :<input type="file" name="fichier" id="fichier"/> </p>
  <p>Un champ que l'utilisateur ne pourra pas modifier ni même voir : <input type="hidden"
name="champ_cache" id="champ_cache" value="coucou" /> </p>
  <p><input type="submit" /><input type="reset" /></p>
</form>
```

>> La balise <label>

L'habitude courante lors du développement d'un formulaire web consiste à placer le texte associé au champs d'entrée à côté de ces champs sans préciser explicitement qu'ils sont associés. Ce n'est pas un gros problème pour l'utiliateur moyen car en visualisant la page, il associe directement ces deux informations et remplit aisément le formulaire. Par contre, pour un utilisateur ayant un handicap, cette association peut ne pas être aussi facile. La balise <label> permet donc d'associer une légende à un champ d'entrée. Ainsi, tous les utilisateurs pourront utiliser votre formulaire sans problème.

>>> L'attribut for

Cet attribut sert à spécifier le champ d'entrée dont le label est la légende.

Exemple :

```
<form name="formulaire" method="post">
  <p><label for="nom">Un champ texte : </label> <input type="text" name="nom" id="nom"/> </p>
  <p><input type="submit" /></p>
</form>
```

>> La balise <fieldset>

Cette balise sert à regrouper plusieurs champs qui ont un rapport entre eux comme part exemple un groupe de cases d'options. Le but de cette balise est le même que celui de <label> car elle permet d'indiquer explicitement la fonction d'un groupe de champs.

>> La balise <legend>

On doit placer cette balise dans un bloc de fieldset. Elle permet de donner une légende au groupe de champs.

Exemple :

```
<fieldset>
 <legend>
  Voici une liste de case à cocher
 </legend>
 <p>
  <input type="radio" name="choix_multiple" id="choix_multiple_1" value="choix1" />
  <label for="choix_multiple_1">Premier choix </label>
 </p>
 <p>
  <input type="radio" name="choix_multiple" id="choix_multiple_2" value="choix2" />
  <label for="choix_multiple_2">Deuxieme choix </label>
 </p>
 <p>
  <input type="radio" name="choix_multiple" id="choix_multiple_3" value="choix3" />
  <label for="choix_multiple_3">Troisieme choix </label>
 </p>
</fieldset>
```

2.17 Programmation HTML Cadres

Un cadre (frame en anglais) est une section de page web contenant elle-même une autre page web. La technique des cadres permet de faire évoluer plusieurs pages web simultanément. Par exemple, un cadre peut contenir le menu d'un site, et un autre son contenu (ce qui évite de devoir insérer un menu sur chaque page du site). Le principal inconvénient des cadres est qu'ils brisent la sémantique des données. Pour cette raison, leur utilisation n'est plus encouragée.

> Définition d'un jeu de cadres : la balise frameset

>> Les attributs cols et rows

Ces attributs servent à déterminer la disposition et les dimensions des cadres : cols pour les colonnes, rows pour les lignes (ou, plus précisément, une séparation verticale ou horizontale).

Imbrication de cadres

Comme vous l'aurez remarqué, la balise frameset ne peut diviser qu'à l'horizontale **ou** à la verticale. Pour combiner les deux, il est possible d'imbriquer les balises frameset. Ce résultat s'obtient en définissant un jeu de cadres (toujours avec la balise frameset donc) dans un cadre.

>> La balise noframe

Certains navigateurs très anciens ne savent pas comment interpréter les cadres. La balise noframe permet d'indiquer à ces navigateurs comment produire la page web d'une manière alternative. Cette balise est ignorée par les navigateurs sachant interpréter les cadres.

> Définition d'un cadre : la balise frame

>> L'attribut src

L'attribut src contient l'adresse (relative ou absolue) de la page à afficher dans le cadre.

>> L'attribut name

L'attribut name permet de donner un nom à un cadre. Ceci permet par exemple d'identifier précisément quel cadre mettre à jour lorsque l'on suit un lien hypertexte.

>> L'attribut longdesc

L'attribut longdesc s'adresse particulièrement aux non-voyants qui utilisent une interface vocale pour "lire" les pages web. Certains logiciels dédiés ont parfois des difficultés à rendre les contenus des cadres dans un ordre pertinent. L'attribut permet d'effectuer un lien vers une description longue des cadres et de leur intérêt. Il s'agit généralement d'une page web séparée contenant un texte explicatif.

>> Les marges : les attributs marginwidth, marginheight et frameborder

L'attribut marginheight accepte une valeur (pixels ou pourcentage) correspondant à une marge par rapport aux bords verticaux (supérieur et inférieur) du cadre. L'attribut marginwidth correspond aux marges horizontales (droite et gauche). L'attribut frameborder accepte les valeurs 1 ou 0 (ou bien TRUE et FALSE) et détermine si le cadre est délimité par une bordure ou non.

>> Les bordures : les attributs border, bordercolor et frameborder

>> Les attributs noresize et scrolling

noresize détermine si l'internaute peut ou non changer la taille du cadre.

scrolling détermine la présence de barres de défilement (toujours, automatique ou jamais).

> Cadres uniques : la balise iframe

Iframes ou fenêtres intégrées dans une page Web

L'intéret est d'ajouter dans une page propre une autre page. Cette méthode est utilisée entre autre pour afficher des bannières de publicité.

>> Les attributs d'iframe

Exemple de code pour la balise Iframe <iframe name="lesteph" src="http://www.lesteph.ch/lesteph/tutoriaux/tuto-2003/webmaster/tutoriaux/html/html-frames.html" width="468" height="60" marginwidth="0" marginheight="0" hspace="0" vspace="0" frameborder="0" scrolling="no"></iframe>

>> Les valeurs de l'attribut target

target est un attribut de la balise a, la balise servant à inclure des liens hypertexte. Il détermine la *cible* (signification de "target" en anglais) du lien, c'est-à-dire où ce lien doit être ouvert, en se basant sur l'attribut name des cadres présents.

target peut prendre différentes valeurs :

nom_du_cadre
: ouvre le lien dans le cadre dont l'attribut name a la valeur "nom_du_cadre".

_self
: le lien s'ouvre dans le même cadre

_parent
: le lien s'ouvre dans le cadre parent du cadre de la page. C'est à dire à la place de la page contenant le <frameset> qui détermine le cadre de la page ou se trouve le lien.

_top
: le lien s'ouvre dans le fenêtre courante (annule donc tous les cadres)

_blank
: ouvre le lien dans une nouvelle fenêtre