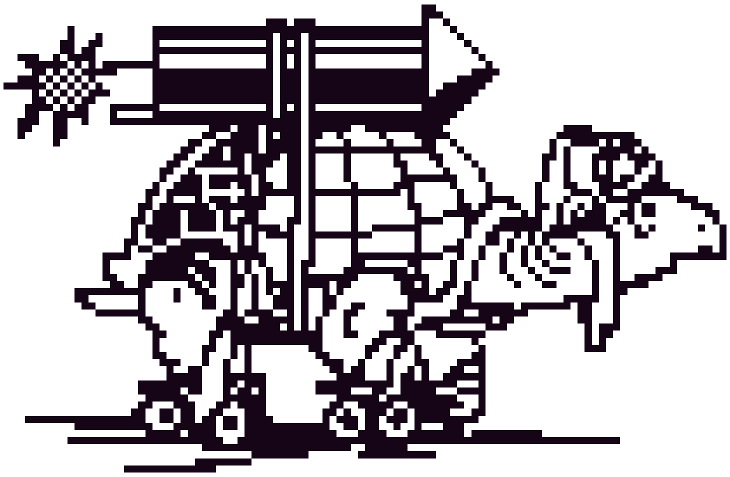
Medias-France

Hassan Makhmara, Fernando Niño, Jean-Luc Boichard

Le Web et ses interfaces



Toulouse, 24 juin 2002

Formation Base de données et interfaces Web

Le but de ce document est de décrire succinctement l'usage et les fonctionnalités des langages utilisés sur le Web pour communiquer avec des bases de données.
Ces descriptions ne sont ni exhaustives ni pérennes, elles représentent une synthèse des connaissances de **Médias-France** en vue d'aider tout développeur à faire des choix pertinents dans ce domaine.

Les connaissances pré-requises sont :
- Un langage de programmation objet
- Utilisation d'un SGBD et du Web
- Beaucoup de calme, de patience, d'obstination (et de Valium ! ! !).

Le contenu de ce document est très inspiré, voire copié à partir de nombreux articles, documents et ouvrages disséminés sur le Net.
En contrepartie, l'usage et la modification de tout ou partie de ce document à des fins personnelles ou commerciales est totalement libre.

L'ensemble de cette documentation à été écrite avec des logiciels libres :
- Système : Linux
- Editeur de texte : Emacs
- Traitement de texte : Latex
- Conversion HTML : Hevea, images : Netpbm
- Figures : tgif, xfig.

Nous en remercions, sans pouvoir les citer, les auteurs et tous ceux qui ont participé à leur rédaction, à leur développement et à leur diffusion.

Toute remarque, suggestion, correction, complément, droit de copie, droit de réponse, ..., pourra être transmise à l'adresse E.mail :

wwwadm@medias.obs-mip.fr

Si un programme est utile, il doit être changé.
Si un programme est inutile, il doit être documenté.

PlusOnAideLeFou PlusIlRit.

Table des matières

I Le Web et ses interfaces **1**

1 Présentation du WWW(World Wide Web) **3**

2 Rendre le Web Dynamique **7**
2.1 Pages dynamiques : définition et fonctionnement 7

II Les langages du Web **11**

3 HTML : Hyper Text Markup Language **13**
3.1 Introduction à HTML 13
3.2 Composantes lexicales de **HTML** 13
3.3 Structure syntaxique de HTML 14
3.4 Classes de balises de HTML 4 14
3.5 Les feuilles de style 17

4 Évolution de HTML **19**
4.1 DHTML 19
4.1.1 DOM 19
4.1.2 Exemples 20
4.2 XHTML 22
4.2.1 pourquoi XHTML 22
4.3 XML : Extensible Markup Language 23
4.3.1 Exemple de déclaration de contenu en XML 24
4.3.2 Visualiser **XML** avec CCS 25
4.3.3 Visualiser **XML** avec **XSL** (eXtensible Stylesheet Language) 25
4.4 APPML : le rêve ou le futur 29

III Interaction Utilisateur coté Client **31**

5 Javascript **33**
5.1 Introduction 33
5.1.1 Limitations de JavaScript 33
5.1.2 Où et comment placer les scripts 33
5.1.3 Variables, fonctions et constantes 34
5.2 Le document : format et contenu 36
5.2.1 La fenêtre 36
5.3 Contrôler le navigateur 38

5.4 Interaction avec des formulaires . . . 40
5.5 Interaction avec l'utilisateur . . . 44
5.6 Cookies . . . 44
5.6.1 A quoi servent les cookies ? . . . 44
5.6.2 Comment stocker des informations ? . . . 44
5.6.3 Comment récupérer les informations ? . . . 45
5.7 Manipulation d'images . . . 46

6 Java : applets 49
6.1 Introduction . . . 49
6.2 Caractéristique de Java . . . 49
6.3 Construire une applet . . . 50

IV Interaction Utilisateur coté Serveur 53

7 Les CGI : Common Gateway Interface. 55
7.1 Les sorties du CGI . . . 55
7.2 Les entrées du CGI . . . 56
7.3 Problèmes de sécurité liés aux programmes CGI - côté serveur . . . 56

8 Les Formulaires en HTML 59
8.1 Introduction . . . 59
8.2 Définir la feuille de formulaire . . . 59
8.2.1 La syntaxe du formulaire . . . 59
8.2.2 Les attributs de la balise FORM . . . 59
8.2.3 Les composants du formulaire . . . 61
8.3 Recourir à un CGI pour traiter le contenu du formulaire . . . 62
8.3.1 Exemple d'utilisation d'un formulaire associé à une BD . . . 63

9 Architecture Serveur 65

10 PHP Hypertext Preprocessor 67
10.1 Que peut faire PHP ? . . . 67
10.2 Fonctionement de PHP . . . 68
10.3 Utilisation de PHP . . . 68

11 ASP Active Server Page 73
11.1 Asp est orienté objet : . . . 73
11.2 Exemple : ajouter un enregistrement à une table . . . 74
11.3 PHP/ASP : comparaisons . . . 75

12 Les Serveurs d'applications dans les Architectures N-Tiers 77
12.1 Introduction . . . 77
12.2 A quoi correspond réellement un serveur d'application ? . . . 78
12.2.1 Le serveur d'application . . . 78
12.2.2 Le serveur d'objet . . . 79

13 Java **81**
13.1 Servlet . . . 81
13.2 Jsp . . . 84

V Exemples d'application **87**

14 Génération dynamique d'images **89**
14.1 Génération de graphiques avec Gnuplot (shell CGI) . . . 89
14.1.1 Format de sortie . . . 89
14.1.2 Commandes de génération d'images . . . 90
14.1.3 Exemples . . . 90
14.2 Génération de graphiques avec JpGraph (PHP) . . . 92
14.2.1 Utilisation . . . 92
14.2.2 Exemples . . . 93
14.3 Génération d'images géographiques avec GMT (PHP/CGI) . . . 96
14.3.1 Options standard . . . 96
14.3.2 Exemples . . . 98

VI Conclusions **103**

VII Annexes **107**

A Liens utiles : **109**
A.1 Conception de pages Html . . . 109
A.2 Perl . . . 109
A.3 PHP . . . 109
A.4 ASP . . . 109
A.5 Les cookies . . . 109
A.6 SQL . . . 110
A.7 Servlet . . . 110
A.8 Applet . . . 110
A.9 Javascript . . . 110
A.10 XML/XSL . . . 110

Première partie

Le Web et ses interfaces

Chapitre 1

Présentation du WWW(World Wide Web)

Le **Web** est un ensemble de sources d'information disponible à travers le réseau Internet. Son but est de relier différents systèmes pour échanger des informations variées telles que du texte, des images mais aussi du son, voire de la vidéo.

Historiquement le **Web** visait à fédérer des moyens plus anciens tels que :

- **Gopher :** méta index de documents organisé de manière hiérarchique
- **WAIS :** (Wide Area Information System) moteur de recherche de type client serveur Internet
- **Archie :** Archivage de fichiers et programmes informatiques.

Initié par le CERN, à Genève en 1989, pour diffuser des notices techniques et scientifiques, il s'agissait d'adapter les fonctions *Hypertext* à un réseau TCP/IP afin de répartir les pages d'un document entre plusieurs machines et de consulter ce document à travers le réseau.

Ce projet se caractérisait par :

- Une architecture décentralisée permettant de déporter sur des sites distants aussi bien la consultation que l'archivage du document.
- L'utilisation systématique de standards existants et l'orientation vers les systèmes ouverts (par opposition à des solutions propriétaires).

Sa popularité s'est décuplée par l'effort du NCSA (*National Center for Computing Application*) qui a produit Mosaic en 1994, l'ancètre des navigateurs actuels.

En quelques années, ce service s'est étendu, et après le désengagement du CERN, le suivi des normes a été repris par un organisme international, le **W3C** (World Wide Web Consortium) piloté par le MIT (*Massachusetts Institute of Technology*, l'INRIA et par l'université de Keio au Japon.

Devant le succès sans cesse croissant du **Web** , ses spécifications évoluent vers la définition d'un standard de terminal multimédia susceptible de supporter des applications client/serveur.

Ce service est basé sur deux standards faisant l'objet de recommandations du W3c :

- un protocole de télécommunications : **HTTP** ;
- un langage : **HTML**

Les autres briques, navigateur, serveurs Web, ... ne sont pas normalisées.

HTML

HTML (Hyper Text Markup Langage) est un langage de description de documents qui est une application d'un langage plus générale **SGML** (Standard Generalised Marked Langage) . Un des ses points forts est de permettre de spécifier des liens vers d'autres pages **HTML** , ne résidant pas nécessairement sur le même ordinateur, ou vers des ressources du système de nature différentes. Ces ressources peuvent être des fichiers images en divers formats, des sons ou des programmes auxquels un bloc de paramètres est transmis.

Cette dernière caractéristique ouvre la voie vers des applications transactionnelles.

Le protocole HTTP

Il est possible de copier des pages **Web** sur son poste de travail, puis d'y accéder en local à travers un navigateur, mais les pages installées de la sorte ne sont accessibles aux autres machines du réseau que si un logiciel spécifique de distribution, ou serveur, a été installé.

Le rôle de ce logiciel serveur est de rendre visible au **Web** une partie des fichiers de la machine. Les deux logiciels, le navigateur et le serveur, interagissent suivant un protocole de télé-communication spécifique : **HTTP** (Hyper Text Transfer Protocol).

Le serveur **HTTP** a pour fonction de traiter les requêtes émises par les navigateurs. La première version de ce protocole était particulièrement simple, et décrivait un échange entre le client et le serveur en quatre phases :

1 Une connexion à l'initiative du client où celui-ci spécifie l'adresse IP et le numéro de port du serveur.

2 Une requête émise par le client constituée d'un mot clé et de caractères ASCII. Cette requête spécifie l' **URL** (*Uniform resource Locator*) ou adresse de la page demandée.

3 Une réponse du serveur qui expédie l'information réclamée, ainsi qu'un code de retour.

4 Une dé-connexion à l'initiative du serveur en fin de transaction ou si le poste client n'a pas expédié d'acquittement au bout d'un certain délai.

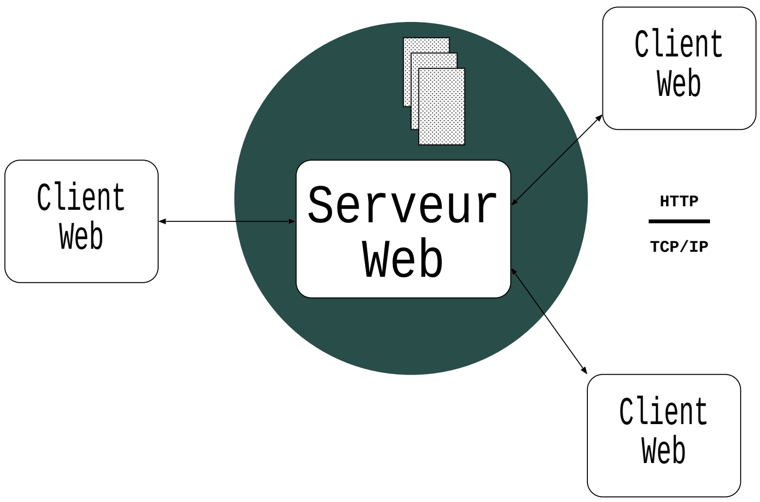


FIG. 1.1 – Client/Serveur Web

essayer :

```
telnet medias.obs-mip.fr 80        //port 80: port http
get index.html HTTP/1.0            // méthode Url niv. du prot.
```

HTTP apparaît comme un protocole client/serveur très dissymétrique, puisque les documents qui constituent la part la plus importante de l'information ne circulent que du serveur vers le client.

La réponse du serveur est constituée de l'en-tête indiquant, entre autres informations, le format du document codé sous format MIME [1] ,et du corps du document qui n'est pas nécessairement codé en **HTML** .

Le serveur n'a, sur la plupart des sites, pas à interpréter le document, son exploitation est sous la seule responsabilité du logiciel de navigation qui prend les options appropriées pour le traiter.

Un tel protocole est sans état (idempotent), le serveur n'a pas à garder en mémoire d'information sur le poste client. Comme chaque page est obtenue par une transaction indépendante, il est facile de répartir un texte entre plusieurs sites.

La simplicité de ces échanges évite l'implémentation d'un système complexe de report d'erreurs et de reprise, mais toute notion de session, entre deux transactions est, par essence, inexistante.

A la différence d'autres protocoles, commençant par une phase d'identification, il n'est pas nécessaire d'entrer un login et un mot de passe pour accéder à un site. Il est toutefois possible de restreindre l'accès d'une partie d'un site, par mot de passe ou par listes d'adresses Internet, mais ces options ne sont pas activées par défaut.

Ce protocole n'est pas sécurisé, pour palier aux problèmes de sécurité deux solutions existent qui permettent de crypter les données transitant sur le réseau :
- **HTTPS/SSL** **HTTP** sur SSL
- **SHTTP** Secure Hypertexte Transfer Protocol

[1] MIME (Multi-Purpose Internet Mail Extensions) est une extension au protocole Internet e-mail qui permet d'utiliser ce protocole pour échanger différentes sortes de fichiers de données : audio, video, images, programmes ..., ainsi que de pur fichiers textes

Chapitre 2

Rendre le Web Dynamique

Les sites web utilisant les pages dynamiques et les bases de données règnent aujourd'hui en maître sur Internet. Il n'est plus imaginable de développer un site Web sans faire appel à ces technologies et aux possibilités d'interactivité et de personnalisation qu'elles permettent.

2.1 Pages dynamiques : définition et fonctionnement

Des pages statiques vers les pages dynamiques

L'explosion de l'Internet, vers 1995, a vu la multiplication des sites dits *cartes de visite*. Ces sites se contentaient, pour la plupart, de transposer au mieux sur le Web les présentations classiques, sur papier, de l'entreprise. Ce type de site, construit en **HTML** pur, a pour défaut majeur de n'offrir qu'une et une seule présentation, sans aucune possibilité de personnalisation et avec une interactivité limiteé au strict minimum. Cette limitation est due à la nature même du **HTML** , langage de présentation des donneés dans lequel le contenu et sa présentation sont indissociables. C'est pour cela que ces pages sont dites statiques. Si l'on veut modifier le *design* d'un site ainsi réalisé, il faut changer une à une toutes les pages qui le composent, avec un éditeur **HTML** , comme **Frontpage, Dreamweaver ou Golive**.

Si cette technique peut suffire pour des sites **Web** de petite taille, au contenu permanent ou rarement mis à jour, elle est par contre peu adapteé aux sites dont les informations font l'objet de modifications fréquentes, comme un catalogue de produits, un journal d'information ou, de manière générale toute application *e -business*.

Pour créer des documents dynamiques (qui s'assemblent ou s'automodifient en fonction des requêtes du client), on peut recourir aux techniques suivantes :

– les documents **HTML** sont finalisés sur le serveur Web, avec les données de la requête transmise par le client.

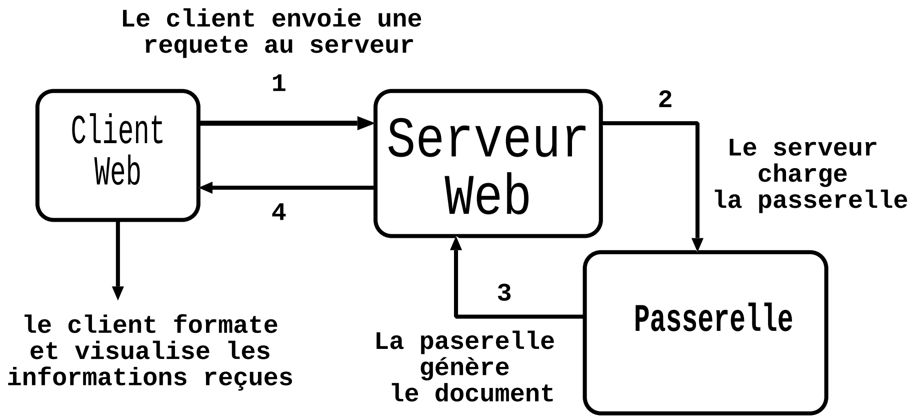


Fig. 2.1 – Création de page dynamique coté serveur

– Les documents HTML sont finalisés chez le client au moyen de scripts ou programmes embarqués dans le document **HTML** .

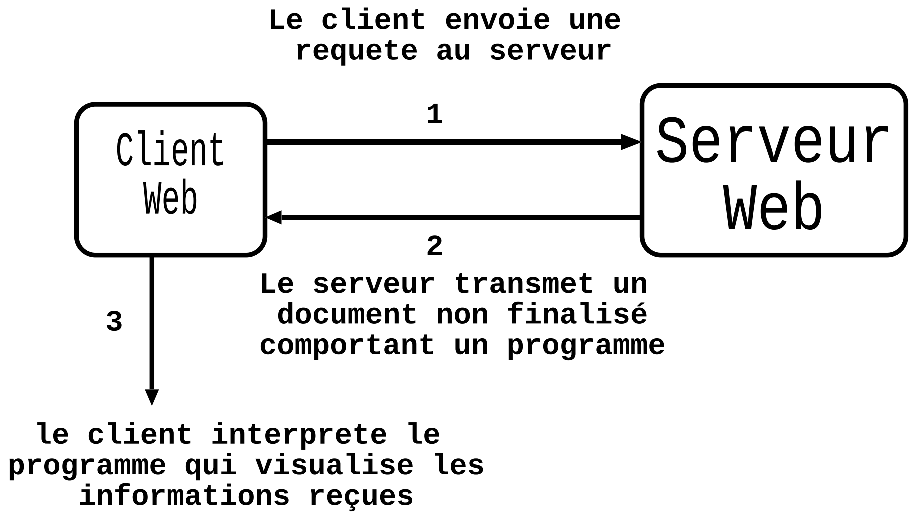


Fig. 2.2 – Création de page dynamique coté client

Avantages et inconvénients des deux solutions

Dynamisme coté client :

Ce type de mécanisme est essentiellement utilisé pour répondre aux besoins
- d'animation des pages **HTML** (menu déroulant, boutons sensitifs....)
- de validation locale des données de formulaire

Les problèmes rencontrés sont :
- portabilité médiocre liée à l'absence de standardisation des langages de script, (VbScript, JavaScript, Java) peu ou mal reconnu par l'ensemble de navigateurs.
- insuffisance des langages de script (Vbscript, JavaScript) telles que :
  - absence de primitives graphiques
  - absence de liaison avec le serveur

- gestion de la sécurité coté client difficile à maîtriser

Dynamisme coté serveur :
Ce type de mécanisme est essentiellement utilisé pour répondre aux besoins
- de liaison avec des applicatifs existants (SGBD, SIG....)
- de création de pages HTML à la volée portable sur l'ensemble des navigateurs (**XSL**)

Les problèmes rencontrés sont :
- augmentation de la charge du serveur
- trous de sécurité potentiel coté serveur
- réponse inefficace aux évènements du poste client

L'approche mixte est souvent la plus utilisée pour pallier aux problèmes rencontrés :
- gestion des évènements locaux coté clients en utilisant une intersection commune des langages de scripts ou des applets Java
- liaison avec les applicatifs distants coté serveur à l'aide de langage de scripts puissants (Vbscript, ASP, PERL,PHP....), de servlet java, ou de programmes exécutables(CGI).....

Deuxième partie

Les langages du Web

Chapitre 3

HTML : Hyper Text Markup Language

3.1 Introduction à HTML

Un fichier **HTML** est un fichier texte contenant des *tags* ou balises.

** Ce texte est gras **

Ces *tags* permettent au navigateur Web de présenter ou structurer le document transmis, ainsi que de matérialiser les liens vers d'autres documents WEB.

```
<body>
  Ceci est ma première page. <b>Ce texte est en gras</b>
  mais il en existe une <a href="page2.html"> deuxième</a>
</body>
```

Les tags , non reconnus par le navigateur, sont ignorés.

Un fichier **HTML** peut être créé par un simple éditeur de texte.

Bien que normalisé par le **W3C,** de nombreuses et fréquentes évolutions,
- HTML 2 (Standard IETF, 1996)
- HTML 3.2 (jan 97)
- HTML 4.01 (déc 99)
- XHTML (2001)

ainsi que des additifs propriétaires perturbent la diffusion de cette norme et grèvent l'avenir de tout document écrit dans ce format.

Un cours d'apprentissage sur le langage peut être trouvé sur
http ://www.w3schools.com/html/

3.2 Composantes lexicales de HTML

- Caractères de données : ascii augmentés de codes spéciaux :

```
&amp;  (&)
&gt;   (>)
&lt;   (<)
&#39;  (')
&eacute; (é)
 ....
```

- Tags (en français : balises, marqueurs, étiquettes) :
  <tag-ouvrant ... > </tag-fermant>
  Les tags fermants ne sont pas tous obligatoires (ie <LI>). L'imbrication anarchique est souvent tolérée (<P> Lisez la <B>FAQ</P></B>)
- Des noms, limités à 72 caractères
  <body>, <Body>, <BODY> : insensibilité à la casse
- Des attributs ou arguments :
  <IMG **SRC="logo.jpg" alt="Notre Logo"**> bien que normalisé par le W3C, plusieurs attributs ont été ajoutés pour des besoins spécifiques des constructeurs, ou sont mal ou pas interprétés par les navigateurs.
- Des commentaires :
  ```
  <$! simple ... > ou <$!-- multi-ligne -->
  ```

3.3 Structure syntaxique de HTML

Un document **HTML** est composé de trois parties :

- Un(des) identificateur(s) de version/namespace du langage
  ```
  <$! DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN"$>
  ```
- **L'en-tête et ses méta-informations**
  ```
  <HTML>
    <HEAD>
      <meta name="author" content="Jean-Luc Boichard">
      <meta name="keywords" content="HTML, CSS">
      <TITLE>Ma homepage</TITLE>
   </HEAD>
  ```
  Les meta attributs de ce document identifient l'auteur et les mots clés attribués au document, ce sont ces attributs qui sont analysés par les moteurs de recherche, et qui de ce fait permettent d'être plus ou moins bien référencés.
  Pour référencer gratuitement ses pages dans plusieurs moteurs à la fois. il exite des meta-référenceurs :
  ```
  http://www.multimania.com/stoi/: Le Référenceur Gratuit
  http://home.cbhouse.fr/carpe/: Référencement gratuit
  etc...
  ```
- **Le corps du document**
  Seule cette partie est obligatoire et affichée par le navigateur
  ```
   <BODY>
     <H1>Bienvenue sur ma page personnelle </H1>
     Test sur<BR>deux lignes
   </BODY>
  </HTML>
  ```

3.4 Classes de balises de HTML 4

Les titres :

```
<H1> à <H6> (ordre d'importance décroissant)
```

Éléments de structure :

```
Paragraphe: <P>
Groupage: <DIV>, <SPAN>
Aussi: <ADDRESS>, <PRE>, <BLOCKQUOTE>
```

Éléments de liste :

```
Liste ordonnée (numéro) <OL> ou non (puces) <UL> et ses items <LI>
 <UL>
   <LI>  ... première ligne
   <LI>  ... seconde ligne
   ...
 </UL>

Liste de définition <DL> <DT> <DD>
 <DL>
   <DT>  ... 1er terme à définir
   <DD>  ... definition du premier terme
   <DT>  ... 2eme terme à définir
   <DD>  ... définition du second terme
       ...
 </DL>
```

Ancres (pour les liens) :

Une ancre peut référencer toute ressource accessible du Web : une page **HTML** , une image, un fichier son, une animation, ...

```
ie:  <a href="http://www.w3schools.com/">Visit W3Schools!</a>
```

La syntaxe de base est :

```
<a href=URL>nom de la page</a>
```

La structure d'un URL est :

```
typeDeRessource : information suppléméntaire
```

Le type de ressource peut être :

- **HTTP** (i.e. la ressource usuelle d'une page **Web**),
- MAILTO,
- FILE,
- NEWS,
- GOPHER,
- TELNET,
- FTP.

L'information suppléméntaire peut être , par exemple pour **HTTP** , l'adresse du fichier **HTML** sur un ordinateur.

L'URL peut être absolue (http ://medias.obs-mip.fr/escompte/) ou relative : par exemple dans le fichier index.html pour faire un lien vers le fichier cv.html qui se trouve dans le même répertoire, il suffit d'écrire :

```
<a href=cv.html>mon CV</a>
```

Pour créer des liens internes au document visualisé, il faut définir à l'endroit ciblé, une ancre grâce à la balise :

```
<a name=paragraphe>
```

et de spécifier un lien avec une balise **href** un peu modifiée :

```
<a href=#paragraphe>Paragraphe</a>
<a href=index.html#paragraphe>Paragraphe</a>
```

Il est possible de faire des liens vers des images (ie : zoom), des .doc Word, des fichiers sons... Si le navigateur ne peut exécuter le fichier, il proposera de le télé-charger (par exemple un fichier compressé). Pour faire un lien vers une adresse e-mail, l'attribut est modifié :

```
<a href="mailto:boichard@medias.cnes.fr">écrivez-moi</a>
```

Quand le visiteur active le lien, son client de messagerie s'ouvre pour générer un nouveau message avec le champ destinataire renseigné.

Tables :

```
<TABLE> <TR> <TD>

<TABLE BORDER=1>
  <TR><TD>This is the first cell<TD>This is the second cell
  <TR><TD>This is the third cell<TD>This is the fourth cell
</TABLE>
```

Alignement, fontes :

```
<HR> <FONT ...>
```

Objets, Applet, Images :

```
<IMG SRC="...">
<OBJECT ...>
```

Ces balises interprétées par le navigateur déclenchent des requêtes supplémentaires.

Les cadres :

Ils permettent de séparer la fenêtre du navigateur en plusieurs :

```
<FRAMESET> <FRAME>
 <html>
   <head>
     <title>Exemple de frames</title>
   </head>
                                                     // deux colonnes
   <frameset cols="108,*" framespacing="0" border="0">    // une de 108pixels
    <frame src="nav.html" name="bar"  marginheight="2" scrolling="auto" noresize>
    <frame name="win" src="main.html" marginwidth="5" >
```

```
  </frameset>

  <noframes>
    <meta http-equiv="Refresh" content="0;URL=main.html">
  </noframes>
</html>
```

Les formulaires :

```
<FORM> <BUTTON> <INPUT>....

 <form name="input" action="html_form_action.asp"  method="get">
      Username:
     <input type="text" name="user">
     <input type="submit" value="Submit">
 </form>
```

Les formulaires sont utilisés pour permettre à l'utilisateur de fournir ses entrées ; ils seront détaillés chapitre 8 en page 59

Les scripts :

```
<script type="text/javascript">
   document.write("Hello World!")
</script>
```

Ce script interprété par le client, produit un texte célèbre, notez que vous devez spécifier le langage de script.

3.5 Les feuilles de style

Beaucoup de *tags* **HTML** mélangent structure du document et présentation :

```
<EM> vs <I>,      Emphasis traduit par fonte italique
<CODE> vs <TT>    code source traduit par typewriter font
```

Un premier pas, pour améliorer la présentation fut l'introduction de styles "*internes*", dans l'en-tête et les éléments structurant.)

```
<P style="font-size: 12pt; color: fuchsia">

<head>
 <style>
   h1 {color: red}
   h3 {color: blue}
 </style>
</head>
```

Ensuite, **HTML** s'est doté des feuilles de style (Style sheets), permettant de mieux séparer structure et présentation. La définition de la présentation étant sortie des documents **HTML** par la création de classe de présentation :

On crée des fichiers **.css** (cascading style sheet), regroupant des styles définis.

```
.espace { background-image: url(/mir/interface/images/mur.png) }
.ebtitre { font-size: 13pt; font-weight: bold }
.etitre { font-size: 12pt; font-weight: bold }
.einfo { font-size: 11pt; font-weight: normal; font-style: italic }

A:link { text-decoration: none; color: #0000ff }
A:visited { text-decoration: none; color: #0000ff }
A:active { text-decoration: none; color: #0000ff }
```

On inclut ces fichiers de présentation dans l'en-tête du document et leur utilisation dans le corps du document **HTML**

```
<HEAD>
   <LINK href="mystyle.css" rel="stylesheet"  type="text/css">
</HEAD>
<BODY CLASS="espace">
 ....
<FONT CLASS="einfo">Nous vous souhaitons une bonne découverte ...</FONT>
```

Pour transformer le *look* d'un site il suffit alors de modifier les fichiers de présentation.

Chapitre 4

Évolution de HTML

4.1 DHTML

Abréviation pour Dynamic HTML, signifiant une combinaison de **HTML** 4.0, Style Sheets and JavaScript.

A son sujet le W3C dit :

> "Dynamic HTML is a term used by some vendors to describe the combination of HTML, style sheets and scripts that allows documents to be animated."

DHTML est la manière de créer des pages **HTML** dynamiques par l'utilisation d'un langage de script(coté client) pour transformer les styles, le comportement et le contenu du document. Aucun langage de script coté client n'étant normé, ce produit doit être testé sur l'ensemble des navigateurs du marché, il est surtout compatible Microsoft (IE).

4.1.1 DOM

HTML 4.0 a créé le **Document Object Model**, qui permet d'accéder aux éléments du document et de son environnement sous forme d'un objet structuré. Cet objet cartographie l'ensemble des sous éléments du document, permettant de ce fait d'y accéder

```
<form name="boxes">            // création d'un identifier de nom boxes
   Enter first name: <input type="text" name="fName"> <br>
   Enter last name: <input type="text" name="lName">
</form
```

Pour référencer dans un script le premier champ texte du formulaire ci-dessus :

```
window.document.boxes.fName. (ou simplement document.boxes.fName).
    ou
document.forms[0].elements[0]
```

La manipulation des objets ainsi cartographié se fait à l'aide d'un langage de script (ie :JavaScript).

```
<html>
  <body>
    <h1 id="header">My header</h1>   // création d'un identifier de nom header
    <script type="text/javascript">  // modification des attributs du header(H1)
      document.header.style.color="red"       //  par un script javaScript
```

```
    </script>
  </body>
</html>
```

Les objets cartographiés ne sont pas standardisés suivant les navigateurs par exemple un paragraphe n'est pas un objet du document sous netscape alors qu'il l'est sous IE,

Les différents objets de DOM

Objet		**Explication**
window	méthodes	L'objet fenêtre est la racine du DOM
	propriétés	elle fournie de l'information
		sur la fenêtre et ses cadres
document	méthodes	L'objet document représente le
	propriétés	document HTML, il donne l'accès
	collections	aux éléments HTML.
navigator	propriétés	L'objet navigator fournit l' information
		sur le client Web de l'utilisateur.
event	propriétés	L'objet event fournit l' information
	collections	sur les évènements qui se produisent dans la fenêtre.

Cette structure est variable suivant les navigateurs pour IE, la hiérarchie débute a l'objet `fenêtre`, pour Netscape `Fenêtre` et `Navigator` sont au même niveau. Comme `fenêtre` est optionnel, en référençant `navigator` , comme dans l'exemple ci-dessous au lieu de `window.navigator` on obtient un code portable sur IE et NN.

Un objet Dom à des méthodes :

```
var text = "This is a new page"
document.open()
document.write(text)
document.close()
```

4.1.2 Exemples

utilisation de DOM en Javascript

```
html>
<body>
 <script type="text/javascript">
 var x = navigator                            // navigateur
 document.write("CodeName=" + x.appCodeName)
 document.write("<br />")
 document.write("MinorVersion=" + x.appMinorVersion)
 document.write("<br />")
 document.write("Name=" + x.appName)
 document.write("<br />")
 document.write("Version=" + x.appVersion)
 document.write("<br />")
```

```
 document.write("BrowserLanguage=" + x.browserLanguage)
 document.write("<br />")
 document.write("SystemLanguage=" + x.systemLanguage)
 document.write("<br />")
 document.write("UserLanguage=" + x.userLanguage)
 </script>
</body>
</html>
```

En savoir plus :

http ://www.slis.indiana.edu/hrosenba/www/Demo/javascript2/DOM.html

4.2 XHTML

4.2.1 pourquoi XHTML

XHTML signifie **EXtensible HyperText Markup Language** son but est de remplacer **HTML** qui devient de plus en plus difficile à maintenir, a évoluer et à porter sur des navigateurs limités (PDA, téléphone) du fait :

- de la tolérance de ce langage :

```
<html>
 <head>
    <title>texte mal formé</title>     //pas de terminaison du header
  <body>
    <IT><B>Bad HTML</IT></B>           // désynchronisation des balises
  </body>
```

- d'un nombre pré défini de balise qui limite la structuration du document, et les possibilités de recherche sur celui-ci.
- de la mauvaise séparation du contenu (données textuelles) et de la présentation, ce qui empêche tout traitement ou recherche de contenu par les applications, ou des traitements spécifiques du contenu en vertu du client.

XHTML a été défini par le W3C en Janvier 2000, il est presque identique à **HTML** 4.01, mais c'est une version plus stricte et plus propre. Toutefois la majorité des navigateurs et produits connexes, comme pour **HTML** 4.01, ne sont pas encore prêts à le traiter de manière exhaustive.

- Les éléments **XHTML** doivent être correctement imbriqués

```
<b><i>This text is bold and italic</b></i>        //mauvais
```

- Les documents **XHTML** doivent être bien formés

```
<!DOCTYPE ...>
<html>
  <head> ... </head>                             // bon
  <body> ... </body>
</html>
```

  La déclaration DOCTYPE définie le type de document :

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

  La **DTD** (Document Type Définitions ou définition du type de document) décrit avec précision et rigueur, la syntaxe et la grammaire des *tags* **XHTML** , il en existe trois (STRICT, TRANSITIONAL, FRAMESET)

- Les noms des *tags* doivent être en minuscule, les attributs doivent être entre guillemets, et posséder un argument (clé='valeur').

```
<body>  <p>This is a paragraph</p>    </body>      // bon
<option name="nom" selected>                       // mauvais
<option name="nom" selected="selected">            // bon
```

- Tous les éléments **XHTML** doivent être terminés, éléments vides compris

```
<br />  ou  <br></br>)
```

XHTML est du **HTML** définit comme une application **XML : eXtensible Markup Language**. C'est une version transitoire entre le **HTML** 4.01 et le **XML** 1.0, HTML 5.0 ne devrait pas voir le jour ?.

Pour favoriser la transition **HTML** -> **XHTML** il existe des convertisseurs :
- effectuer la conversion de **HTML** vers **XHTML**
  `ie: http://www.w3.org/People/Raggett/tidy/`
- valider le code **XHTML**
  `ie: http://validator.w3.org/`
- enfin pour tester votre code afin de vous assurer qu'il est bien conforme aux normes XHTML, utiliser le logiciel gratuit "xmllint" qui vient avec la librairie libxml. Pour l'utiliser, entrer cette commande :

```
xmllint --valid --noout document.html
```

Convertir ses pages **HTML** 4.01 en **XHTML** 1.0 n'entraîne aucun problème de compatibilité notable avec les navigateurs actuels, et il convient bien sûr à tous dans la mesure ou ces pages seront conformes aux prochaines recommandations. Aujourd'hui **XHTML** est la meilleure façon de suivre efficacement le développement des standards Web .

4.3 XML : Extensible Markup Language

Officiellement né le 10 Février 1998 sous l'égide du W3C, **XML** s'est dan un premier temps cantonné à la gestion documentaire son influence s'étend maintenant à l'ensemble du système d'information.

Comme **HTML** , **XML** est un langage de balises, toutefois celles-ci n'indiquent pas une action à entreprendre sur le texte en vue de son affichage. Les balises **XML** servent à séparer et à structurer l'information pour en définir le contenu, c'est à dire décrire des données.

Ces balises ne sont pas pré-définies, il est nécessaire de créer ses propres balises ou d'utiliser des définitions pré-définies (DTD, shéma), pour définir le contenu d'un document et favoriser les recherches sur celui-ci.

Un document bien formé doit être conforme à la syntaxe **XML** , un document valide doit être conforme à sa **DTD** (descripteur d'éléments)

En fait **XML** ne fait rien, il faut comme pour **HTML** un langage interprété pour le manipuler ou formatter son affichage, on doit utiliser des programmes interpréteur comme **XSL**, (extensible style langage), ou encore Javascript. Côté serveur, les composants ASP, PHP, VBSCRIPT ou encore JSP permettent de manipuler les documents **XML** .

IE5 et Nestcape 6 sont toutefois capable de visualiser des documents **XML** , sans toutefois assurer une couche de présentation par défaut. (comment interpréter une balise TABLE ? ?) .

```
<mobilier>
  <table>500F</table>
  <chaise>100F</chaise>
</mobilier>
```

XML est un ensemble de normes et d'outils, indépendant de toutes plates-formes, pour transmettre et traiter de l' information. Son essor rapide dans le monde du **Web** laisse croire qu' il sera de plus en plus utilisé pour décrire et transmettre les données , **HTML** se résumant à la couche de présentation.

4.3.1 Exemple de déclaration de contenu en XML

```
<Catalogue>
 <prod id="33-657" media="paper"></prod>
 <Texte>
   <PC> Word
     <COMPAGNIE>Microsoft</COMPAGNIE>
     <COMPATIBLE>Win95/Win98</COMPATIBLE>
     <PRIX>Gratuit</PRIX>
   </PC>
   <PC> QuarkXpress
     <COMPAGNIE>Quark</COMPAGNIE>
     <COMPATIBLE>Win95/Win98</COMPATIBLE>
     <PRIX>600F</PRIX>
   </PC>
   <MAC>
     . . .
   </MAC>
  </Texte>
  <Graphisme>
    . . .
  </Graphisme>
</Catalogue>
```

Remarques :

- Une structure hiérarchique lie, les éléments entre eux : Le Catalogue est la racine du document, les logiciels ses fils.

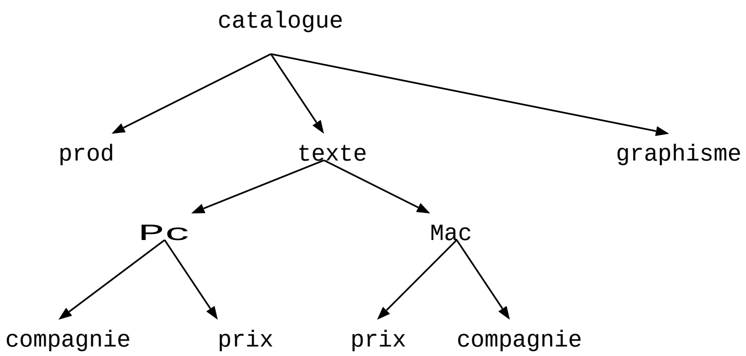


FIG. 4.1 – Structure objet et hiérarchique d'un document Xml

- les éléments sont typés : un élément peut contenir des éléments (Catalogue), de la donnée (PRIX), les deux (PC), ou rien (PROD) ;
- un élément peut avoir des attributs (PROD).

4.3.2 Visualiser XML avec CCS

soit le fichier CCS de style : catalog.css

```
Catalogue
{
  background-color: #ffffff;
  width: 100%;
}
Texte, Graphisme
{
  display: block;
  margin-bottom: 30pt;
  margin-left: 0;
}
PC,MAC
{
  color: #FF0000;
  font-size: 20pt;
}
COMPAGNIE, COMPATIBLE, PRIX
{
  color: #0000FF;
  font-size: 20pt;
}
```

L'ajout du tag suivant :

```
<?xml-stylesheet type="text/css" href="cd_catalog.css"?>
```

au fichier XML devrait permettre aux navigateurs de traiter sa visualisation. Cette fonctionnalité des navigateurs est toutefois très hypothétique, IE5 et IE6 sont toutefois capables de visualiser **XML** avec **XSL** (eXtensible Stylesheet Language)

4.3.3 Visualiser XML avec XSL (eXtensible Stylesheet Language)

XSL n'est pas simplement une représentation des feuilles de styles (CCS) en **XML** . C'est un langage qui peut transformer des documents **XML** en **XHTML** , un langage qui peut trier et filtrer les données **XML** , un langage qui peut formatter des documents pour des médias différents, bref un véritable langage de programmation.

XSL est constitué par trois produits :
- **XSLT** : c'est la partie la plus importante de **XSL** , c'est un langage pour transformer des documents **XML** en d'autres documents **XML**
- **XPath** : un langage utilisé par **XSLT** pour définir les parties d'un document qui sont associées à des *templates*, les parties non associées ne seront pas modifiées
- **XSL Formatting Objects** : un vocabulaire pour formatter les documents **XML**

XSL est un standard du W3C (Octobre 2001) , actuellement seul IE6 supporte pleinement la recommandation W3C XSL.

exemple :

soit le document **XML** : cdcatalog.xml qui définit les données

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>

  <catalog>
     <cd>
       <title>Empire Burlesque</title>
       <artist>Bob Dylan</artist>
       <country>USA</country>
       <company>Columbia</company>
       <price>10.90</price>
        <year>1985</year>
      </cd>
    ...
  </catalog>
```

et son document **XSL** associé cdcatalog.xsl qui définit la présentation :

```
<?xml version="1.0" encoding="ISO-8859-1"?>

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

      <xsl:template match="/">
        <html>
          <body>
           <h2>My CD Collection</h2>
           <table border="1">
             <tr bgcolor="#9acd32">
               <th align="left">Title</th>
               <th align="left">Artist</th>
             </tr>
             <xsl:for-each select="catalog/cd">
              <tr>
                <td><xsl:value-of select="title"/></td>
                <td><xsl:value-of select="artist"/></td>
              </tr>
             </xsl:for-each>
            </table>
          </body>
        </html>
     </xsl:template>

   </xsl:stylesheet>
```

Noter :