Introduction au web et à la gestion de base de données

Laurent Pouilloux

XX janvier 2006

1 Introduction

L'échange d'informations est primordial dans l'évolution des sociétés. A la manière de l'imprimerie, l'informatique en général et Internet en particulier représente une mine de possibilités, que ce soit dans la communication, l'industrie, la recherche ou l'enseignement. Ce cours est là pour vous faire découvrir l'envers du décor. Actuellement, la production de site web ne requiert pas de connaissances informatiques car les solutions toutes faites types blog, Content Management Server (CMS) et wiki permettent de faciliter la publication d'informations. Mais l'administration demande d'appréhender certains concepts de base que nous allons présenter ici. Dans un premier temps, nous rappelerons le fonctionnement d'un ordinateur. Puis nous évoquerons brièvement le principe de fonctionnement d'un serveur web et la composition d'une page. Dans un second temps, nous expliquerons les principes de fonctionnement du langage HyperText Markup Language (HTML) et de l'ajout de Cascading Style Sheet (CSS) pour la mise en page. Enfin, nous introduirons un langage dynamique le PHP pour créer des sites puissants et simples, en utilisant des bases de données MySQL.

2 Rappels Généraux

Nous allons parler des principes généraux liés au monde de l'informatique : comprendre le fonctionnement d'un ordinateur et la logique d'un système d'exploitation. Puis nous évoquerons les réseaux et Internet. Enfin, nous introduirons le concept de page web.

2.1 L'ordinateur

L'ordinateur est un ensemble de composants électroniques qui échangent des informations sous forme de courants électriques. Basiquement, il se décompose en une unité centrale et un ensemble de périphériques. L'unité centrale est le coeur de l'ordinateur. Elle comprend une Carte Mère (qui relie tous les composants), un microprocesseur (qui effectue les calculs), de la mémoire vive (qui sert à stocker les informations temporaires), une Carte réseau (qui permet de communiquer avec un autre ordinateur),une Carte Graphique (qui calcule spécifiquement l'affichage), une Carte Son et une alimentation. Cet ensemble des informations avec les périphériques. La carte son envoie le signal audio aux haut-parleurs, la carte graphique le signal vidéo, la carte mère transfère les informations entre les différents composants. Elle va lire et écrire ce qui est stockés sur un CD-ROM, un disque dur, une clé usb ou sur un site web. Un schéma simplifié est présenté sur la figure 1. Pour de plus amples informations, rendez-vous sur http ://www.commentcamarche.com.

La présentation de la partie hardware étant maintenant terminé, il nous faut expliquer la partie software, c'est-à-dire l'ensemble des programmes qui sont utilisés pour faire fonctionner l'ordinateur. La première partie installée est le système d'exploitation. Windows, Linux ou MacOS permettent de faire fonctionner les périphériques (affichage, son, usb) : ils sont constitués de tâches qui s'exécutent en permanence sans que l'utilisateur n'agissent. On peut le décomposer en deux parties : les tâches non-graphiques et les tâches graphiques (X). On installe ensuite des applications

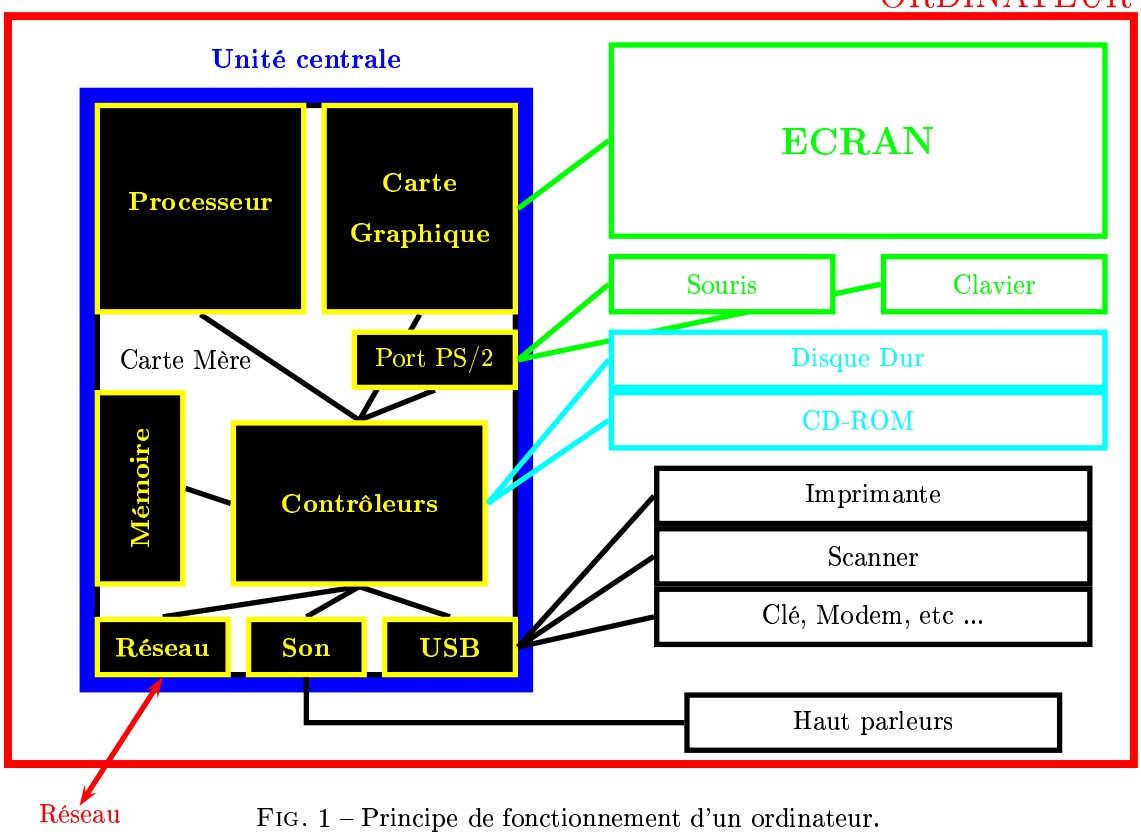


FIG. 1 – Principe de fonctionnement d'un ordinateur.

(Office, Firefox, Gimp, Winamp) qui s'exécutent à la demande de l'utilisateur. Celle-ci utilisent des fichiers d'entrées et des fichiers ou périphériques de sorties.

2.2 Les réseaux et Internet

Les ordinateurs partagent de l'information. Des cassettes à bandes des débuts, on a ensuite utilisé des disquettes puis des CD-ROM et des clés USB. Dans le même temps, on a donné aux ordinateurs la capcité de communiquer directement par l'intermédiare de fils, permettant de travailler ensemble sur la même applications ou d'echanger directement des fichiers. Il s'est formé ainsi des réseaux d'ordinateurs dans de nombreux endroits (laboratoire de recherche, centre de l'armée) qui ont étés ensuite liés pour former une immense toile tout autour du monde : le World Wide Web. Depuis n'importe quel ordinateur connecté au réseau des réseaux, on peut communiquer avec des ordinateurs distants : serveur email, serveur de news, serveur web, ... Nous allons nous intéresser de manière plus détaillée sur le serveur web, qui est important de comprendre avant de se lancer dans la réalisation de site web.

2.2.1 Qu'est qu'un serveur web ?

De nombreux programmes fonctionnent selon le concept client-serveur. Un ordinateur est le serveur : il gère les connections, distribuent les informations, effectuent les calculs. De nombreux ordinateurs clients se connectent sur le serveur pour effectuer des tâches diverses utilisant les fonctionnalités du serveur. Ainsi, l'utilisateur travaillant sur un navigateur, tel Mozilla Firefox, va interroger un serveur web en cliquant sur un lien HTML (voir figure 2). Sur le serveur web sont stockés les différentes pages. l'ordinateur client envoie une requète au serveur ("Je veux me rendre à l'adresse http ://www.acrimed.org") et le serveur envoie en retour la page concernée au format HTML. Celle-ci peut être directement stockée en mémoire au format HTML ou bien être créer par un langage dynamique comme PHP. La première méthode est la plus simple, la seconde

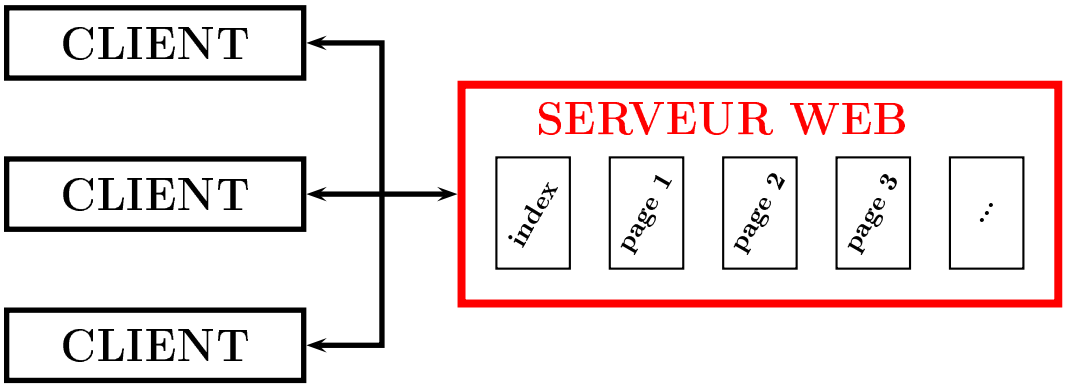


FIG. 2 – Principe de fonctionnement d'un serveur web.

beaucoup plus puissantes puisqu'elle permet d'utiliser des bases de données. Dans ce cours, nous introduirons les bases de types MySQL. Il est à noter qu'avec la page HTML est envoyé un certain nombre de fichiers utilisés par la page : image, javascript, CSS.

Nous disposons maintenant des principes de bases et nous pouvons donc s'intéresser aux langages permettant de créer des pages web.

3 HTML

Le premier langage que nous allons voir est le HTML, pour HyperText Markup Language. Une traduction simple est "langage à balises utilisant l'hypertexte". Dans un premier temps, nous allons détailler le concept de balises et nous nous intéresserons aux liens hypertextes. Dans un second temps, nous parlerons des images, des tableaux et des listes dans le cadre de la création d'une page personnelle. Enfin, nous introduirons le CSS afin de personnaliser le design de son site.

3.1 Les Bases

Le langage HTML comprend un ensemble d'instructions logiques que le navigateur interprète. Il permet d'écrire le contenu du site web qui sera affiché par le navigateur de l'ordinateur client. Le principe de ce langage est de délimiter des zones par des balises. On commence un bloc par une balise par <balise> pour l'ouverture et on termine par </balise> pour la fermeture. Il existe aussi des balises simples qui s'écrivent <balise/> (ouvrante-fermante). Des exemples de balises des deux types seront donnés par la suite dans le cours . Les balises possèdent des attributs qui permettent de donner des précisions sur la balise (taille, alignement, etc...)

A ce stade précoce, un point important est à noter : le HTML est un langage dont la véritable norme n'est pas appliqué et dont l'interprétation dépend sensiblement du navigateur employé. Nous recommandons d'utiliser Mozilla Firefox (http ://www.mozilla.org) car il est open source et que c'est le plus proche de la norme (x)HTML. De plus, il est moins vulnérable aux spywares et autres désagréments liés à Internet. Passons maintenant aux choses pratiques : la structure de la page, la mise en forme du texte et les liens.

3.1.1 Structure d'une page

Une page au format HTML comprend deux types d'informations : un corps (<body>), qui comprend les informations qui vont être affichées dans la page ; un entête (<head>) ; qui permet d'ajouter à la pages des informations (titre, affichage, script). Commençons avec une page simple. On démarre par créer une page en utilisant la commande touch, dans le répertoire *public_html*.

Pour cela, on lance un terminal et on execute les commandes suivantes :

```
cd public_html
touch mapage.html
```

On l'ouvre ensuite avec l'éditeur de son choix, par exemple nedit.

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
   <head>
       <title>Bienvenue sur ma page !</title>
       <meta http-equiv="Content-Type" content="text/html;
       charset=iso-8859-1" />
   </head>
   <body>
Bonjour tout le monde !<br/>
Ceci est ma page personnelle
   </body>
</html>
```

Il faut maintenant interpréter le code source HTML. On lance le navigateur (ici Firefox) et on ouvre la fichier *mapage.html*. Nous avons fait notre première page.

La balise <html> (fermée par </html>) est la balise principale qui englobe toute la page HTML. Vient ensuite la balise <head> qui contient les informations d'en-tête de la page web, par exemple la balise <title> pour le titre de la page. On a utilisé également la balise <body> où l'on tape le contenu qui sera affiché par le navigateur. Remarquons au passage que le saut de ligne nécessite la balise inline
. Nous allons maintenant introduire les balises de bases permettant de mettre en forme le texte.

3.1.2 Mise en forme du texte

La mise en forme du texte s'effectue en utilisant des balises spécifiques. On peut par exemple créer des titres (balises <h#></h#>) et des paragraphes ( ). On ajoute dans notre page, entre les balises <body></body>, le code suivant :

```
       <h1>Bonjour tout le monde !</h1>
       <p>
         Ceci est ma page personnelle.
       </p>
       <h2>Mes informations</h2>
       <p>
Ceci est un paragraphe.
       </p>
       <p>
Ceci est un autre paragraphe.
       </p>
```

Nous allons maintenant introduire cinq balises permettant de modifier le texte.

- : pour mettre en italique.
- : pour mettre en gras.
- : pour mettre en exposant.
- : pour mettre en indice.
- <acronym title="Sciences de la Terre de l'Environnement et des Planètes">STEP</acronym> : pour expliquer un sigle.

Vous pouvez les tester en les introduisant dans le texte que vous avez taper auparavant.

3.1.3 Les liens hypertextes

Passons maintenant à l'autre composante du langage : les liens. Il s'agit de balises spécifiques qui permettent de naviguer au sein d'une page ou d'une page à une autre. Dans un premier temps, on va créer une nouvelle page toujours dans le répertoire public_html, nommée *monprojet.html*. On introduit dans le code de *mapage.html* le code suivant :

```
Cliquez <a href="monprojet.html">ici</a> pour accéder à mon projet.
```

La page cible est situé après l'attribut href et le texte affiché est contenu entre les balises <a>. Après avoir actualisé la page, on peut cliquez sur le lien. Le navigateur ouvre alors la page *monprojet.html*, qui est vide. On peut y écrire ce que l'on veut, par exemple créer un lien pour revenir à *mapage.html*.

On peut aussi créer des liens vers des pages qui ne sont pas situés sur même serveur web. On peut alors taper :

```
Accès à un <a href="http://www.acrimed.org">site sur les dérives des médias</a>.
```

Il existe (principalement) deux autres types de liens, le mailto et l'ancre, qui permettent d'envoyer un email ou de créer des points de navigation dans une même page. La syntaxe est la suivante :

```
Pour m'écrire, cliquez <a href="mailto:login@ipgp.jussieu.fr">ici</a>.<br/><br/>
Si vous voulez descendre dans la page,
<a href="mapage.html#bas_de_page">cliquez ici</a>.
```

En bas de la page, on écrit :

```
<a name="bas_de_page">Bas de page</a>.
```

Evidemment, pour voir un effet, il faut que le bas de la page ne soit pas déjà visible. Pour remonter en haut de la page, on utilise la balise #top.

Nous avons maintenant vu les concepts de bases que sont la structure de la page, la mise en forme du texte et le fonctionnement des liens. Dans la suite, nous nous intéresserons à d'autres balises très utiles et à leur utilisation.

3.2 Pour faire des choses pratiques

Un support multimedia comprend des zones de textes, des images, des fichiers attachés. Nous allons expliquer comment introduire une image dans une page ainsi que des fichiers externes, par exemple un fichier pdf. Puis nous nous intéresserons aux tables et à leur utilisation dans la mise en page du site. Enfin, nous créerons un menu en utilisant les listes.

3.2.1 Insérer une image

La première chose à savoir, c'est que HTML gère principalement trois types d'images : JPG, PNG, GIF. Chacune a ses spécificités. JPG est très intéressant pour les photos ; PNG permet de laisser un fond transparent ; GIF autorise les images animées. On créer maintenant un sous-répertoire images/ dans public_html/. Dans un terminal, tapez :

```
mkdir images
```

Passons maintenant au code HTML. La balise est de type inline, comme
, et s'écrit :

```
<img src="images/image.png" title="commentaire de l'image"/>
```

L'attribut src correspond à la source (on lui indique le chemin relatif de l'image) et l'attribut title permet de faire apparaitre une description quand la souris reste dessus. Il est conseillé de mettre toutes les images dans un même répertoire. Evidemment, une image peut servir de lien, simplement en encadrant la balise <img /> par des balises de liens (<a>).

3.2.2 Liens vers des fichiers

On peut include dans un site web des fichiers de toutes sortes. Pour cela, on les place dans un répertoire spécifiques, par exemple public_html/files, et on écrit dans le fichier *mapage.html* :

```
<a href="files/exemple.pdf">Téléchargez le pdf</a>.
```

En cliquant sur le lien, le navigateur propose de l'ouvrir ou de l'enregistrer sur le disque. Ce principe fonctionne pour de nombreux types de fichiers, comme les vidéos ou les musiques. Cela permettra par exemple mettre en ligne votre cv. Il vaut mieux utiliser des formats utilisables partout, comme le pdf, et petit, comme le mp3. Attention cependant à ne pas utiliser des ressources sous copyright sans autorisation.

3.2.3 Créer un tableau

Nous allons maintenant introduire une méthode simple pour créer des sites webs lisibles sur tous les navigateurs et bénéficiant d'une mise en page agréables. Pour cela, on va créer des tables. Une table s'ouvre avec une balise <table> et se ferme avec une balise </table>. Les lignes sont délimitées par les balises <tr> et </tr>. Dans chaque ligne, on introduit des cellules par les balises <td> et </td>. Voici une exemple simple de table, à mettre entre les balises <body></body> :

```
<table border="1" cellspacing="0" cellpadding="0" width="100%">
<tr>
    <td align="center" colspan="2">
    <a href="mapage.html">Bienvenue sur ma page</a>
    l'attribut colspan permet de fusionner des
    cellules horizontalement. Pour les lignes, c'est
    rowspan
    </td>
</tr>
<tr>
    <td valign="top" width="120px">
    valign="top" permet de coller le texte en haut et
    width="120px" permet  definir la largeur de la colonne.
    </td>
    <td align="justify">
    sk,lmsfqdjmql;vm,m ls,dffm: mll,sfdm,x cml,smdf :!klzsùlcx
    lskmjfds mksdfksjdflmsjdf smdffkjsdf sdksdfl jlskdfjlsdkfj
    </td>
</tr>
</table>
```

Détaillons un peu cet exemple. La balise <table> possède ici quatre attributs : **border**, qui donne l'épaisseur de la bordure des cellules (0 pour ne pas en avoir) ; **cellspacing**, pour l'espacement entre les cellules ; **cellpadding**, pour la position du texte par rapport aux bordures ; **width**. Pour les balises <td>, on a ajouté différents attributs qui sont détaillés directement dans la table.

Evidemment, il existe une quantité d'attributs possibles dont nous ne parlerons pas ici mais que vous pourrez trouver dans les sites référencés à la fin. Nous allons conserver cette structure de base et créer une page web personnalisée.

3.2.4 Créer un menu

Un site web contient généralement plusieurs pages. Par exemple, on peut avoir une page d'accueil, en général *mapage.html*, une page sur le curriculum vitae, ici *cv.html*, une page sur le projet

informatique, ici *monprojet.html*, et une page de liens, *liens.html*. Dans un premier temps, nous allons créer les fichiers :

```
touch cv.html
touch liens.html
```

Le principe d'un menu est de créer des liens vers les différentes pages. Une bonne manière de le faire est d'utiliser des listes. Dans la case 2ème ligne - 1ère colonne, on tape entre les balises <td></td> le code suivant :

```
<h3>Menu</h3>
<ul>
<li><a href="mypage.html">Accueil</a>
<li><a href="cv.html">Mon CV</a></li>
<li><a href="monprojet.html">Mon Projet</a></li>
<li><a href="liens.html">Mes liens</a></li>
</ul>
```

Nous avons maintenant une page de base, avec un menu permettant de naviguer entre les pages. Mais tout ceci reste très moche, et nécessite d'être un peu enjolivé.

3.3 CSS et mise en page

Nous allons maintenant introduire un autre langage adapté à la mise en forme de notre page web : le CSS. Après en avoir expliqué le principe, nous allons créer un site web personnalisé.

3.3.1 Qu'est-ce que le CSS ?

Le CSS est un langage très important pour la présentation du site web. Le principe est simple : on ajoute à n'importe quelle balise un atribut class et on caractérise la classe concernée dans un fichier .css. Pour l'activer, on ajoute dans le <head></head> un appel vers le fichier

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
   <head>
       <title>Exemple d'utilisation de CSS externe</title>
       <meta http-equiv="Content-Type" content="text/html;
       charset=iso-8859-1" />
       <link rel="stylesheet" media="screen" type="text/css"
       title="essai" href="essai.css" />
   </head>
   <body>
       <h3> Titre en rouge et en Verdana </h3>
       <p>Cette page comporte une feuille de style externe. C'est la
       meilleure méthode à utiliser quand on fait du CSS.</p>
       <p class="bleu">
       je voudrais écrire ce paragraphe en bleu, donc j'ajoute
       l'attribut class="bleu" à ma balise et je vais ensuite editer
       le fichier css que j'ai appelé dans le header (design.css)
       </p>
   </body>
</html>
```

On crée le fichier essai.css et on entre le code suivant :

```
h3{
    color:red;
    font-family:Verdana;
}
.bleu{
    color:blue;
}
```

De cette manière, tous les titres <h3> seront écrit en rouge et en police Verdana. Les balises auquelles on aura ajouté l'attribut class="bleu" auront du texte écrit en bleu (à noter, le . devant bleu dans le fichier css qui permet de créer une nouvelle classe).

3.3.2 Créer un design personnalisé

Nous allons maintenant appliqué ce principe pour notre page et créer le design de son choix. Pour cela, nous allons créer une nouvelle page nommée *template.html* en introduisant les attributs de balises class.

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
   <head>
       <title>Bienvenue sur ma page !</title>
       <meta http-equiv="Content-Type" content="text/html;
       charset=iso-8859-1" />
       <link rel="stylesheet" media="screen" type="text/css"
       title="essai" href="monstyle.css" />
   </head>
   <body>
    <table class="table_struct">
    <tr>
        <td class="header" colspan="2">
        <a href="mapage.html">Bienvenue sur ma page</a>
        </td>
    </tr>
    <tr>
        <td class="menu">
<h3>Menu</h3>
        <ul>
        <li><a href="mypage.html">Accueil</a>
        <li><a href="cv.html">Mon CV</a></li>
        <li><a href="monprojet.html">Mon Projet</a></li>
        <li><a href="liens.html">Mes liens</a></li>
        </ul>
<br/>
        </td>
        <td class="content">
&npsp;
        </td>
    </tr>
    <tr>
    <td class="footer" colspan="2"><a href="mailto:login@ipgp.jussieu.fr</td>
    </tr>
    </table>
    </body>
</html>
```

Nous avons crée 4 classes pour chacune de nos cases : header représente la bannière, menu représente le menu, content représente le contenu et footer représente le pied de page. On va attribuer des caractéristiques à chacune d'entre elles dans le fichier monstyle.css.

```
body{
        margin:0px;
        text-align:center;
}
.table_struct{
        width:800px;
}
.header{
        height:120px;
        text-align:center;
        background:url('images/header.png');
        vertical-align:top;
        padding-top:10px;
        padding-left:160px;
}
.header a{
        font-size:40px;
        font-family:Impact;
        color:black;
        text-decoration:none;
}
.menu{
        width:150px;
        border:4px dashed #021680;
        background-color:#d4dbff;
        text-align:center;
}
.menu h3{
        color:#021680;
}
.menu ul{
        text-align:left;
}
.menu a{
        color:#021680;
        text-decoration:none;
}
.content{
        padding:5px;
        text-align:justify;
}
```

Vous pouvez tester les différents attributs pour comprendre leur fonctionnement et créer ainsi votre page personnalisé. Pour l'exemple, l'attribut background permet de mettre une image dans le fond.

Nous avons créé notre template, nous allons maintenant apprendre à créer des pages de manière dynamique en utilisant PHP et MySQL.

4 PHP et MySQL

Un site web fonctionne avec une quantité d'informations qui est constante sur toutes les pages. Quant le site est grand, il est intéressant d'utiliser des fichiers spécifiques pour les parties communes et un langage permettant de créer des pages HTML : le PHP. A la différence d'un serveur HTML, les pages envoyées au client ne sont pas stockés mais sont créées à chaque fois. Le client envoie une demande en cliquant sur un lien vers une page PHP, le serveur compile la page en utilisant des informations issues du serveur ou d'une base de données MuSQL ou encore données par le client. Elle est ensuite envoyée par le serveur au format HTML (voir figure 3).

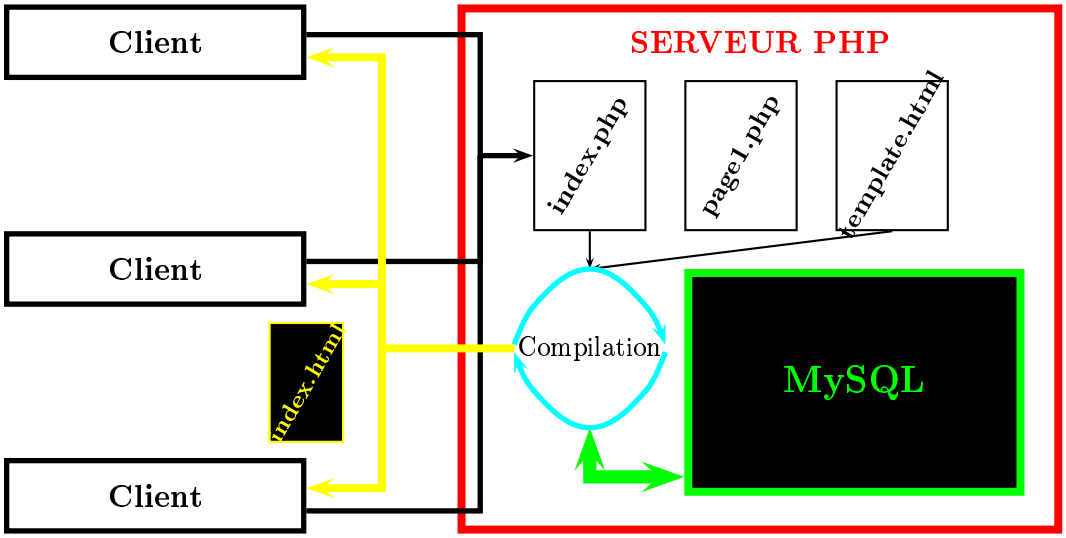


FIG. 3 – Principe de fonctionnement d'un serveur PHP.

Pour faire fonctionner un site contenant du PHP, il faut utiliser des serveurs particuliers. Sous Linux on utilise :

- Apache : permet au serveur de distribuer des pages web.
- PHP : plugin pour apache
- MySQL : permet d'utiliser des bases de données.

Il existe aussi des solutions simples sous Windows, comme EasyPHP. Le serveur que l'on va utiliser est celui de l'Institut de Physique du Globe. Passons maintenant au langage à proprement parler. Nous allons effectuer quelques rappels d'algorythmique.

4.1 PHP : un langage de programmation pour le web

Le PHP est un langage compilé qui contient les différentes composantes d'un langage de programation. Nous allons brièvement passer en revue les différents éléments. Pour l'utiliser, on insère dans une page le code en utilisant la balise <?, on choisit des instructions à effectuer puis on referme avec ?>.

```
<? // code Php à insérer
   // code ligne 2
   // ...
?>
```

La commande // permet d'insérer un commentaire. La première commande que nous allons voir permet d'afficher du texte. On utilise pour cela la commande echo. Par exemple,

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
    <head>
        <title>La première instruction : echo</title>
        <meta http-equiv="Content-Type" content="text/html;
         charset=iso-8859-1" />
    </head>
    <body>
        <h2>Affichage de texte avec PHP</h2>

        <p>
            Cette ligne a été écrite entièrement en (x)HTML.<br />
            <? echo "Celle-ci a été écrite entièrement en PHP."; ?>
        </p>
    </body>
</html>
```

Cette commande sert à afficher n'importe quel texte, y compris du html, des variables ou autre. Nous verrons cela plus loin.
Remarque 1 : à la fin d'une instruction PHP, il faut ABSOLUMENT mettre un ; sinon la page affichera un PARSE ERROR.
Remarque 2 : pour mettre des guillemets, il faut écrire \" au lieu de " pour que PHP comprenne que ce n'est pas la fin de la chaine de caractères.

paragraphPremier bilan : Le PHP n'est pas executé par le client. Celui effectue un requète sur une page, par exemple *index.php*, qui s'execute sur le serveur et qui renvoie la page produite au client. Nous pouvons maintenant passer aux composantes classiques de la programmation

4.2 Composantes classiques de tous langages de programation

Comme tout langage de programmation, PHP comprend des variables, des fonctions, des boucles et des conditions. Nous allons présenter succintement la syntaxe. ?

4.2.1 Les variables

Une variable est une information stockée en mémoire temporairement. Elle est caractérisée par son nom et possède une valeur. A la différence du FORTRAN, il n'est pas nécessaire de déclarer les variables et le type est automatiquement déterminé lors de l'affectation. Si le type est string, on met des guillemets ; pour un nombre, on met le nombre seul ; pour un boulean, on met true ou false seul.

```
   <h2>Affectation et affichage d'une variableavec PHP</h2>

   <? $toto=17;?>
   <p>
   <? echo $toto;
   echo "La variable toto vaut".$toto.". Intéressant, non ?"; ?>
    </p>
    </body>
</html>
```

L'affectation s'effectue à l'aide d'un signe égal et on l'affiche en utilisant echo. On peut aussi l'afficher au milieu d'une chaine de caractères par concaténation (signe .).
En PHP, on a les possibilités de calcul classique (+,-,*,/) et on peut donc effectuer ce que l'on veut :

```
<?
$nombre = 2 + 4; // $nombre prend la valeur 6
$nombre = 5 - 1; // $nombre prend la valeur 4
$nombre = 3 * 5; // $nombre prend la valeur 15
$nombre = 10 / 2; // $nombre prend la valeur 5

// Allez on rajoute un peu de difficulté
$nombre = 3 * $nombre + 1; // $nombre prend la valeur 16
$nombre = (1 + 2) * 2; // $nombre prend la valeur 6
?>
```

Tout cela est parfaitement classique et donc nous ne nous y attarderons pas. Passons maintenant à une spécificité PHP : la transmission de variables dans un lien. Par exemple, on va écrire en HTML :

```
<a href="index.php?jour=27&amp;mois=07&amp;annee=05">Infos du 27/07/05</a>
```

& sert à mettre un &. Cette modification du lien revient à créer des variables spéciales, de type GET qui peuvent être appelées dans la page cible par les commandes suivantes :

```
<?
  echo $_GET['jour'];
  echo $_GET['mois'];
?>
```

Il existe un autre moyen de transmettre des variables, en utilisant des formulaires.

4.2.2 Les fonctions

Pour effectuer des instructions répétitives on peut utiliser des fonctions. La déclaration est très simple :

```
<?
  function DireBonjour($nom)
  {
  echo "Bonjour $nom !<br />";
  }
?>
```

et l'utilisation de même :

```
<?
  DireBonjour("Gerard Majax")
?>
```

De la même manière on peut écrire des fonctions de calcul :

```
<?
// Ci-dessous, la fonction qui calcule le volume du cône
 function VolumeCone($rayon, $hauteur)
 {
 $volume = $rayon * $rayon * 3.14 * $hauteur * (1/3); // calcul du volume
 return $volume; // indique la valeur à renvoyer, ici le volume
 }
```

```
 $volume = VolumeCone(3, 1);
 echo "Le volume d'un cône de rayon 3 et de hauteur 1 est de ".$volume;
?>
```

Bien sûr, PHP est doté de nombreuses fonctions très utiles comme par exemple la fonction date() :

```
<?
$annee = date("Y");
echo "$annee";
?>
```

Vous trouverez des fonctions sur les sites référencés plus bas.

4.2.3 Les conditions

Les conditions permettent d'effectuer des instructions différentes selon les cas. La première chose à connaitre sont les symboles des conditions :

Symbole	Signification
==	Est égal à
>	Est supérieur à
<	Est inférieur à
>=	Est supérieur ou égal à
<=	Est inférieur ou égal à
!=	Est différent de

La première structure est if ... then ... else ... En voici un exemple :

```
<?
 $age = 8;
 if ($age <= 12) // SI l'âge est inférieur ou égal à 12
 {
 echo "Salut gamin ! Bienvenue sur le site !<br />";
 $autorisation_entrer = "Oui";
 }
 else // SINON
 {
 echo "Ceci est un site pour enfants, vous êtes trop vieux
 pour pouvoir entrer. Au revoir !<br />";
 $autorisation_entrer = "Non";
 }
?>
```

Evidemment on peut faire des conditions multiples en utilisant les mots-clés AND ou OR.

```
<?
 if ($test="oui" OR $test2 == "oui")
 {
 echo "Bonjour !;
 }
 else
 {
 echo "Au revoir";
 }
?>
```

Un structure conditionnnelle très pratique est **switch($variable)** : elle permet de traiter différents cas de manière beaucoup plus simple qu'un succession de **if then else**. un exemple simple :

```
<?
 switch($variable);
 {
 case 1:
 echo "toto";
 break;
 case 2:
 echo "titi";
 break;
 case 3:
 echo "tata";
 break;
 default:
 echo $variable;
 }
?>
```

4.2.4 Les boucles

Les boucles permettent de répéter des instructions le nombre de fois que l'on veut. Nous allons voir deux types de boucles : while et for. La première permet de répéter autant de fois qu'il le faut une série d'instruction. Pour la seconde, on fixe à l'avance le nombre de répétition. Voici un exemple pour chacune :

```
<?
// la boucle while
while($continuer_boucle == "oui")
{
// instructions à répéter
}
// la boucle for
for ($i=1; $i<10, $i=$i+1)
{
echo "Ceci est la ligne ".$i;
}
?>
```

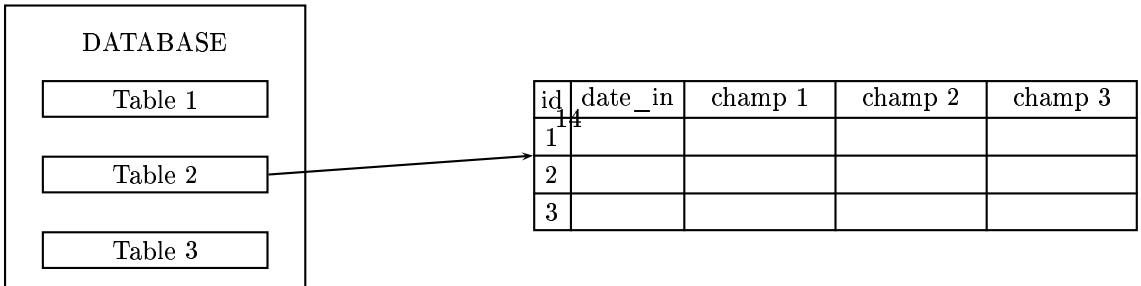
Tout cela est encore une fois très classique donc ne devrait pas poser de problème.

4.2.5 Les tableaux

4.2.6 Les formulaires

4.3 MySQL : un gestionnaire Open source de Base de données

Voici maintenant une partie très importante de PHP : l'intéraction avec les bases de données. Une base de données est un système qui enregistre et classe des informations. Elle est constituée de différentes tables, un peu à la manière d'une armoire (la base) avec des dossiers (les tables).



Une table est un ensemble d'entrée repérée par une id. Un champ très utile à créer est celui correspondant à la date d'introduction dans la base (**date_in**). Les autres champs peuvent être nom, type ou n'importe quoi d'autre.
PHP va intéragir avec MySQL : il existe beaucoup de fonctions prédéfinies pour le faire. La première fonction permet de se connecter à la base de données. Il s'agit de **mysql_connect()**. Cette fonction a besoin de trois paramètres : **hostname**, c'est-à-dire l'IP de l'ordinateur où MySQL est installé ; **login**, c'est l'identifiant servant à se connecter au serveur ; **password**, c'est le mot de passe de connexion au serveur. Une fois connecté au serveur, il faut sélectionner la base de données qui correspond au site web. Pour cela, on utilise la fonction **mysql_select_db("db_name")**, avec en argument le nom de la base donnée. Dernière chose, à la fin de la page, il faut se déconnecter de la base : **mysql_close()**, sans argument.

```
<?
mysql_connect("hostname","login","password");
mysql_select_db("db_name");

// contenu de la page
// on peut jouer avec la base de données, faire tout et n'importe quoi

mysql\_close();
?>
```

Une fois connecté à la base de données, on va lui envoyer des requètes à exécuter. On crée une variable, $query, qui va être envoyé à la base grâce à la fonction **mysql_query** et le résultat sera retourné dans une variable.

```
<?
$query='SELECT * FROM users';
$result=mysql_query{$query}
?>
```

4.3.1 Requètes MySQL de base

Dans le cadre du projet informatique, vous aurez une base correspondant à votre projet. Une fois connecté et votre base sélectionnez, vous allez devoir créer des tables.

```
<?
$query="CREATE TABLE 'table1' (
        'id' MEDIUMINT NOT NULL AUTO_INCREMENT,
        'champ1' TEXT NOT NULL,
         index('id')
        )";
mysql_query{$query} // et la table sera créer
?>
```

Pour créer une table, il faut lui donner un nom (ici, table1) et créer les différent champs. Chaque champ est caratérisé par un nom (entre cote), un type (MEDIUMINT, TEXT, BLOB pour les fichiers, DATE, ...) et des caractéristiques. De plus, on va choisir un des champs qui servira d'index (ici, id).
Prenons une table, users, possédant 5 champs (id, date_in, login, password, statut) et 20 entrées. Pour accéder aux informations de la table on utilise la requète SELECT :

```
<?
$query="SELECT * FROM users";
$result=mysql_query{$query}; // on aura tous mis
//dans la variables result et
// on fait ensuite un boucle pour afficher les entrées:
while ($donnees=mysql_fetch_array($result)) \\ ce qui signifie
\\ va chercher dans le tableau et rempli la variables donnes
{
echo $donnees['login']." a été enregistré le ".$donnees['date_in']."
Il s'agit d'un ".$donnees['statut'].".<br/>";
}
?>
```

Cette série d'instructions va écrire à l'écran autant de lignes que d'entrées dans la table. Bien sûr, si la table est énorme, cela risque de prendre du temps et tout le contenu n'est pas forcément nécessaire. Il existe donc des critères de sélection. Tout d'abord, on peut sélectionner seulement les champs qui nous intéressent :

```
<?
$query="SELECT login FROM users";
$result=mysql_query{$query}; // on aura mis tous les logins dans la variables result
// On fait ensuite un boucle pour afficher les logins
echo "liste des noms";
while ($donnees=mysql_fetch_array($result))
{
echo $donnees['login'];
}
?>
```

On peut aussi choisir des entrées possédant des champs en communs en utilisant le critère WHERE. Par exemple :

```
<?
$query="SELECT * FROM users WHERE statut='Etudiant'";
$result=mysql_query{$query};
echo "liste des étudiants";
while ($donnees=mysql_fetch_array($result))
{
echo $donnees['login']." a été inscrit le ".$donnees['date_in'];
}
?>
```

Il existe d'autres critères de sélection,(ORDER BY, LIMIT) que vous pouvez trouver une nouvelle fois sur les sites référencés.
Passons maintenant à l'insertion de données. On va utiliser une autre requète : INSERT INTO.

```
<?
$query="INSERT INTO users VALUES('','date('Y/m/d')','toto','pass','Moniteur')";
mysql_query{$query} // on aura tous mis dans la variables result
?>
```

On choisit la table dans laquelle insérer les données et on donne TOUTES les valeurs de l'entrée. On execute la requète toujours en utilisant **mysql_query** et la table est modifiée.
Il est possible de mettre à jour des données en utilisant la requete UPDATE :

```
<?
$query="UPDATE users SET password='new_pass' WHERE login='toto'";
mysql_query($query);
?>
```

et on a changé le mot de passe pour toto. Si cette entrée ne sert plus à rien, on peut la détruire avec la requete DELETE :

```
<?
$query="DELETE FROM users WHERE login='toto'";
mysql_query($query);
?>
```

et l'entrée toto a été supprimée !

5 Quelques sites à voir et des conseils divers

5.1 Sites utiles

- allhtml
  Un site très complet où l'on peut trouver toutes les balises, des menus, des morceaux de code.
  http ://www.allhtml.com/
- Le site du Zér0
  Un site simple pour tout apprendre. Le style est particulier, mais le contenu est bon : c'est un véritable cours en ligne.
  url : http ://www.siteduzero.com/
- php débutant
  Un bon portail pour débuter en PHP
  url : http ://www.phpdebutant.org

Il existe sur Internet une masse énorme de site pour vous aider, il suffit juste de savoir utiliser Google (http ://www.google.fr).

5.2 Conseils

- Indentez votre code : cela facilite grandement le débuggage.
- Structurez votre site : utilisez des fichiers communs pour la configuration, la connexion ou le template.
- Utilisez des variables globales.
- Les autres conseils, ce sera quand vous aurez commencé.

6 Conclusion

Table des matières