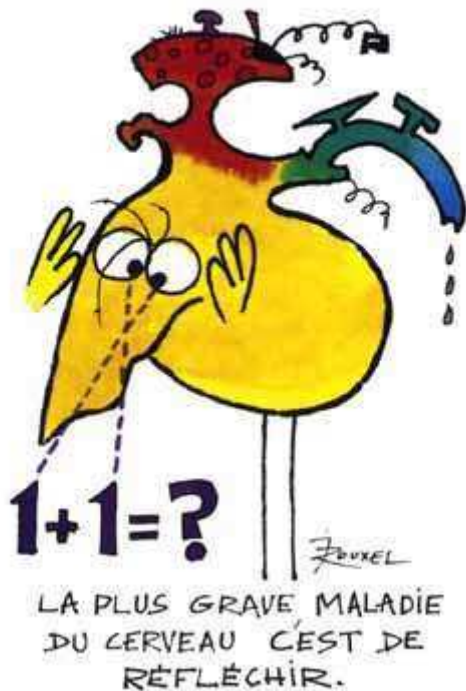
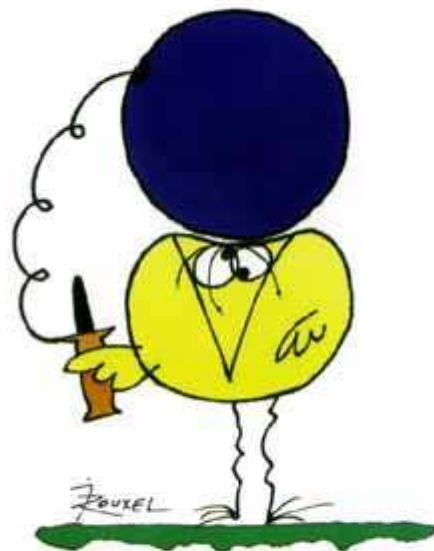


Création de Pages Web Dynamiques

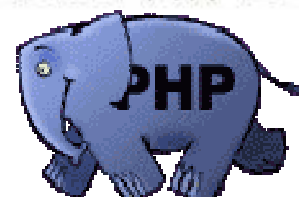
Les devises Shadok



Les devises Shadok



Sommaire :



| | | |
|-------|---|----|
| 1 | Présentation du PHP | 3 |
| 2 | Les différentes méthodes de mise en oeuvre: | 4 |
| 3 | Fonctionnement | 5 |
| 4 | Le langage HTML | 6 |
| 4.1 | Les balises | 7 |
| 4.2 | Les attributs | 7 |
| 4.3 | Première page | 8 |
| 4.4 | Mise en forme du texte : | 8 |
| 4.5 | Les titres | 8 |
| 5 | Les feuilles de styles | 11 |
| 6 | Langage PHP | 15 |
| 6.1 | Les commandes de sortie : | 18 |
| 6.2 | Terminer l'exécution du script | 18 |
| 6.3 | Commenter le code | 18 |
| 6.4 | Les types de variables | 19 |
| 6.4.1 | Variables scalaires | 19 |
| 6.4.2 | Tableaux | 19 |
| 6.5 | OPERATEURS | 21 |
| 6.5.1 | Comparaison | 21 |
| 6.5.2 | Logiques | 21 |
| 6.5.3 | Arithmétiques | 21 |
| 6.5.4 | Affectation | 22 |
| 6.6 | STRUCTURES DE CONTROLE | 22 |
| 6.6.1 | Si : if | 22 |
| 6.6.2 | switch | 22 |
| 6.7 | Boucles | 23 |
| 6.8 | FONCTIONS | 24 |
| 6.8.1 | Paramètres optionnels Les paramètres optionnels sont autorisés : il suffit de leur affecter une valeur par défaut. | 24 |
| 6.8.2 | Passage de paramètres par adresse | 25 |
| 7 | Les variables prédéfinies (variables superglobales) | 26 |
| 7.1 | Format d'une URL | 26 |
| 7.1.1 | Syntaxe générale | 26 |
| 7.1.2 | Syntaxe de la liste des paramètres | 27 |
| 7.1.3 | Syntaxe du couple (nom, valeur) | 27 |
| 7.1.4 | Résumé | 27 |
| 7.1.5 | Exemple concret | 27 |
| 7.2 | Passage de paramètres entre pages | 28 |
| 7.3 | Les éléments du formulaire | 29 |
| 7.3.1 | Les petites zones de texte | 29 |
| 7.3.2 | Une grosse zone de texte: | 29 |
| 7.3.3 | La liste déroulante : | 29 |
| 7.3.4 | Les cases à cocher : | 30 |
| 7.3.5 | Les boutons d'option : | 30 |
| 7.3.6 | Les champs cachés : | 30 |
| 7.4 | Transmettre en utilisant un formulaire | 31 |
| 7.5 | Les sessions et les cookies : | 32 |



1 Présentation du PHP

On doit la première version de PHP à Rasmus Lerdorf qui l'a mise au point pour ses propres besoins en 1994. Cette version était destinée à son usage personnel, d'où le nom (Personal Home Pages). Cette version n'a pas été mise à disposition du public. Vers 1995, une version qui permettait l'exécution de quelques macros fut mise à disposition, elle permettait entre autre de gérer un livre d'or et un compteur de hits.

PHP, est un acronyme récursif, qui signifie "**PHP: Hypertext Preprocessor**" : c'est un langage de script HTML, exécuté côté serveur.

il est **interprété** c'est-à-dire que le code PHP est traduit en **HTML**.

PHP implémente la programmation orientée objet.

En effet, le code PHP **n'est pas visible** en éditant la source d'une de vos pages.

Vous verrez **uniquement du HTML** coté client.

Sa syntaxe est empruntée aux langages C, Java et Perl, et est facile à apprendre.

Le but de ce langage est de permettre aux développeurs web d'écrire des pages dynamiques rapidement.

Coder en PHP demande tout de même une certaine maîtrise du langage HTML car le PHP est obligatoirement mêlé à de l'HTML.

3 utilisations principales :

Langage de script coté serveur. C'est l'utilisation la plus traditionnelle, et aussi le principal objet de PHP. Vous aurez besoin de trois composants pour l'exploiter : un analyseur PHP (CGI ou module serveur), un serveur web et un navigateur web.

Langage de programmation **en ligne de commande**. Vous pouvez écrire des scripts PHP et l'exécuter en ligne de commande, sans l'aide du serveur web et d'un navigateur. Il vous suffit de disposer de l'exécutable PHP. Cette utilisation est idéale pour les scripts qui sont exécutés régulièrement (avec un cron sous Unix ou Linux), ou un Task Scheduler (sous Windows). Ces scripts peuvent aussi être utilisés pour réaliser des opérations sur des fichiers texte.

Ecrire des **applications clientes graphiques**. PHP n'est probablement pas le meilleur langage pour écrire des applications clientes graphiques, mais si vous connaissez bien PHP et que vous souhaitez exploiter des fonctionnalités avancées dans vos applications clientes, vous pouvez utiliser PHP-GTK pour écrire de tels programmes.

PHP c'est avant tout un langage vous proposant plus de **2000 fonctions** des plus diverses, il existe des bibliothèques vous permettant de créer des images, travailler/créer des fichiers .pdf, travailler avec flash, ... (mais toutes **ces bibliothèques ne sont forcément disponibles** sur tous les serveurs donc renseignez vous auprès de votre hébergeur)

En général, PHP est **principalement utilisé** pour aller chercher des informations dans une **base de données MySQL**.

2 Les différentes méthodes de mise en œuvre:

Pour pouvoir lancer des pages PHP localement (sur votre ordinateur), vous devez impérativement disposer d'un **serveur web** (en général Apache) **d'un interpréteur PHP** et d'un **système de gestion de base de données**.

Différentes plate forme :

Plate-forme **LAMP** (Linux Apache Mysql PHP)

Plate-forme **MAMP** (MacOs Apache Mysql PHP)

Plate-forme **WAMP** (Windows Apache Mysql PHP)

L'installation :

- Un par un, en installant séparément **Apache, PHP, MySQL**

- Automatiquement : Le plus simple avec EasyPHP que vous pouvez télécharger à cette adresse :

<http://www.easyphp.org/>.

Dans ce cours, nous utiliserons Easy PHP V5.3.8

EasyPHP est un **WAMP**, une plateforme de développement Web, permettant de faire fonctionner **localement** (sans se connecter à un serveur externe) des scripts PHP.

EasyPHP n'est pas en soi un logiciel, mais un environnement comprenant deux serveurs (un serveur web **Apache** et un serveur de bases de données **MySQL**), un interpréteur de script **PHP**, ainsi qu'une administration SQL **PhpMyAdmin**.

Il dispose d'une interface d'administration permettant de gérer **les alias** (dossiers virtuels disponibles sous Apache), et le démarrage/arrêt des serveurs.

Il permet donc d'installer en une seule fois tout le nécessaire au **développement local du PHP**. Par défaut le serveur Apache crée un nom de domaine virtuel (car local) **http://127.0.0.1** ou **http://localhost**.

Ainsi, quand on choisit **"Web local"** dans le menu d'EasyPHP, le navigateur s'ouvre sur cette URL et affiche la page **index.php** de ce site qui correspond en fait au contenu du dossier www d'EasyPHP.

La version 5.3.8 :

Apache 2.2.19, PHP 5.3.8, MySQL 5.5.15, PhpMyAdmin 3.4.3.2, Xdebug

2.1.2

Version 5.3.3(Apache 2.2.16, PHP 5.3.3, MySQL 5.1.49, PHPMyAdmin 3.3.5)

La version 2.0 : Apache 2.2.3, PHP 5.2.0, MySQL 5.0.2, PHPMyAdmin 2.9.1.1, SQLiteManager 1.2.0.

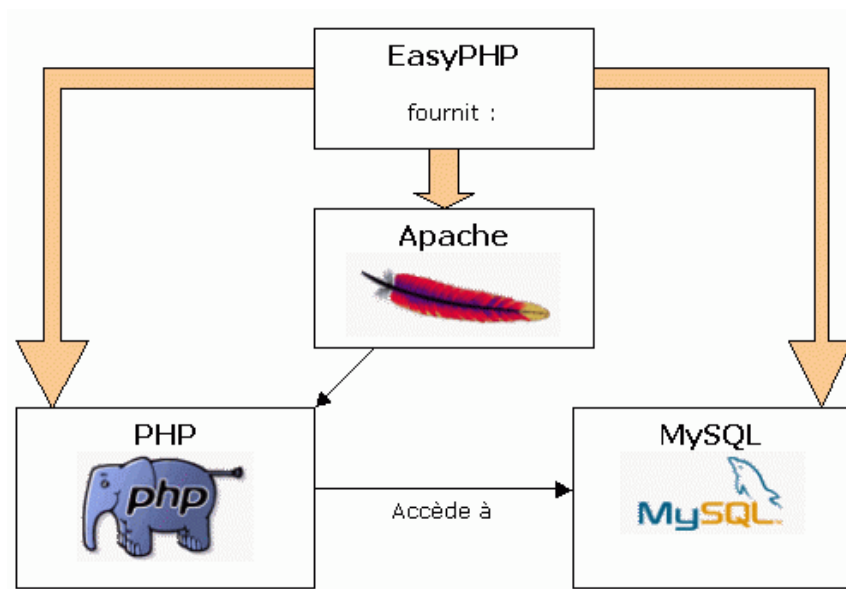
EasyPHP peut être utilisé comme **application portable**, c'est-à-dire lancé sur une clé USB.

Easy PHP comprend :

- Apache
- Apache Manager
- MySQL
- PHP V3
- PHP V4
- PHPMyAdmin
- Supporte le WAP, le XML et différents types d'images

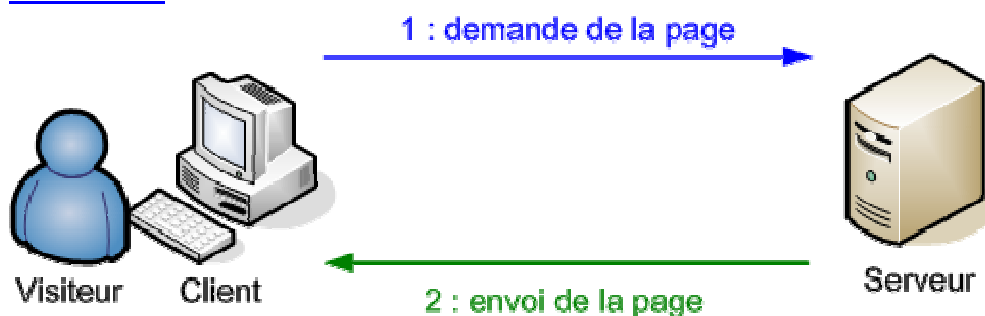


On peut représenter graphiquement :



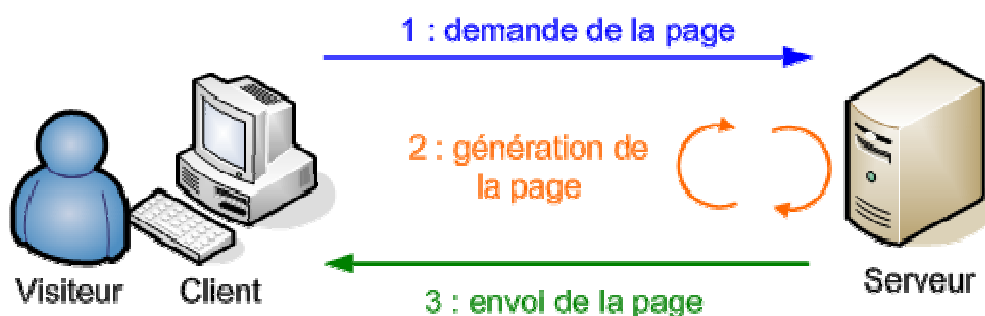
3 Fonctionnement

en HTML :



en PHP :

Il y a une étape qui vient s'ajouter entre les deux : la page **PHP est générée** par le serveur avant l'envoi.



Cela veut dire qu'en fait la page générée peut être à chaque fois **unique**.

Notez que la génération de la page peut prendre du temps (quelques millisecondes en fonction de la taille de la page).

Donc le serveur doit être plus puissant pour pouvoir traiter du PHP qu'un serveur HTML classique.

4 Le langage HTML

Pour créer un site web, on doit indiquer des informations à l'ordinateur. Il ne suffit pas de simplement taper le texte qu'il y aura dans son site, il faut aussi savoir placer ce texte, insérer des images, faire des liens etc...

Pour expliquer à l'ordinateur ce que vous voulez, il va falloir utiliser un langage qu'il comprend.

Les langages **HTML**, **XHTML** et **CSS** servent à créer des sites web, ils ont été créés de manière à être simples à utiliser.

XHTML : c'est l'abréviation de *eXtensible HyperText Markup Language*.
HTML puis en 2000 XHTML et HTML5 en 2009

Ce langage HTML, c'est celui avec lequel vous **créerez le contenu** de votre site web.

Il contient des informations sur la page, le contenu avec les formats : Titre, contenu, images ...

CSS : c'est l'abréviation de *Cascading Style Sheets* ("Feuille de style").

Ce langage sert uniquement **à définir la présentation de la page web**

centré, menu avec un fond rouge...

Grâce au CSS on précisera la chartre graphique des pages.

Grâce à ce langage, nous pourrons créer rapidement et simplement la mise en page d'un site et lui donner une **présentation homogène**.

Pour créer un site le **Bloc-Notes** suffit, mais pour faciliter l'écriture nous utiliserons **Notepad++** qui permet de visualiser le code avec des couleurs / langage utilisé.

Pour consulter des pages il faut un **navigateur**, c'est le programme qui transforme le langage HTML en aspect graphique.

Parmi les navigateurs citons :

- **Internet Explorer**
- **Mozilla Firefox**
- **Chrome Google**
- Opéra
- Netscape
- **Konqueror** (pour Linux)
- Lynx (pour Linux)
- **Apple Safari** (pour Mac)

Les versions les plus utilisées d'Internet Explorer ("IE") sont :

IE 5.5 : livré avec Windows 98

IE 6 : livré avec Windows XP

IE 7 IE 9 ...

Les versions 5.5 et 6 sont incomplètes et pose des problèmes avec le CSS.

L'utilisation de FireFox évite ces problèmes.

Vous pouvez installer Firefox sans désinstaller Internet Explorer.

Les 2 navigateurs peuvent fonctionner en même temps sans souci.

4.1 Les balises

Dans une page HTML en plus du texte, il y a **les balises**.

Une balise commence par "<" et se termine par ">".

Par exemple :

<balise>

Les balises sont **invisibles** pour le visiteur, elles servent **de marqueurs** pour indiquer quelque chose au navigateur.

Il existe 2 types de balises :
- balises par paire
- balises seules.

Les balises existant par paire : ce sont les plus courantes.

Ex : titre **<title>** Bienvenue sur mon site ! **</title>**

La première balise **<title>** indique le début du titre, et elle est refermée plus loin avec **</title>**

Le navigateur comprend que ce qui est entre **<title>** et **</title>** est le titre de votre site web, et que celui-ci est **"Bienvenue sur mon site !"**

Les balises seules :

elles sont un peu plus rares, on sert en général pour insérer un élément dans une page.

Ce type de balise se termine toujours par un **slash "/"**,

Par exemple la balise qui permet d'insérer une image :

**** (image)

4.2 Les attributs

Les attributs sont un moyen de donner des précisions sur une balise.

On peut trouver des attributs sur les deux types de balises (par paire ou seules).

Par exemple : la balise seule ****. Il faut indiquer quelle image.

Ici, l'attribut nom est **"src"**, et il a pour valeur **"chemin/toto.jpg"**.

Cela indique que l'image que l'on veut insérer s'appelle **"toto.jpg"** et se trouve dans le répertoire

Chemin dans le répertoire où se trouve la page.

alt = "portrait de toto"

Première page

```
1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4 <!-- informations pour le navigateur -->
5 <title> Bienvenue sur mon site </title>
6 <meta charset="utf-8" />
7 <!-- Section non affichable -->
8 </head>
9 <body>
10 <!-- Corps de la page web -->
11 <p>Structure d'une page web en HTML5</p>
12 </body>
13 </html>
```

Note : On peut ajouter des commentaires dans une page HTML .

Ex : **<!-- c'est ici que l'on écrit les commentaires -->**

4.3 Mise en forme du texte :

En XHTML, les choses sont plutôt simples : ça fonctionne en paragraphes.

Chaque paragraphe se trouve entre les balises **<p>** et **</p>**

Pour passer une ligne

soit on crée un nouveau paragraphe

soit on insère une balise : passage de ligne **
**. A l'intérieur du paragraphe !

4.4 Les titres

Rédiger du texte c'est bien, mais donner un titre à votre texte c'est encore mieux.

En XHTML, on a le droit d'utiliser **6 niveaux** de titres différents.

(un peu comme avec WORD)

Il y a 6 balises de titre différentes que l'on peut utiliser :

- **<h1> </h1>** : signifie "titre très important".
 - En général, on s'en sert pour afficher le titre de la page en haut.
- **<h2> </h2>** : signifie "titre important".
 - Utilisez-les par exemple pour organiser vos paragraphes et leur donner un titre.
- **<h3> </h3>** : c'est un titre un peu moins important (un "sous-titre")
- **<h4> </h4>** : titre encore moins important.
- **<h5> </h5>** : titre pas important.
- **<h6> </h6>** : titre vraiment pas important du tout.

Si, à l'intérieur de vos paragraphes, il y a du texte à faire ressortir, HTML met à votre disposition des balises : ***mise en forme par défaut dans les navigateurs***

```
<p>
<strong> Texte en gras </strong>
<em> Texte en italique </em>
<mark> Texte surligné </mark>
</p>
```

Texte en gras Texte en italique Texte surligné

Il est également possible de
Mettre en exposant

2^{ème} → 2^{ème}

Mettre en indice

x_n → x_n

Pour insérer une image, on doit utiliser la balise

Elle peut prendre plusieurs attributs, et 2 d'entre eux sont indispensables :

- **src** : il permet d'indiquer où se trouve l'image que l'on veut insérer.
 - **src="images/cours.jpeg"**
- **alt** : cela signifie "texte alternatif". On doit **toujours** indiquer un texte alternatif à l'image, c'est-à-dire un court texte qui décrit ce que contient l'image. Ce texte sera affiché à la place de l'image si celle-ci ne peut pas être téléchargée :
 - **alt="C'est mon Cours"**

Si vous ne le faites pas, votre page ne sera plus une page HTML "valide" et votre site sera détecté comme non conforme aux normes.

4.5 L'encodage

Cette balise indique l'encodage utilisé dans votre fichier .html.

L'encodage indique la façon dont le fichier est enregistré. C'est lui qui détermine comment les caractères spéciaux vont s'afficher (accents, idéogrammes chinois et japonais, symboles arabes, etc.).

Il y a plusieurs techniques d'encodage utilisées en fonction des langues :

ISO-8859-1, OEM 775, Windows-1253...

Seul : UTF-8 devrait être utilisé. Cette méthode d'encodage permet d'afficher sans aucun problème pratiquement tous les symboles dans toutes les langues.

Note :

Sous Notepad++, allez dans le menu **Encodage > Encoder en UTF-8 (sans BOM)** pour que votre fichier soit enregistré en UTF-8 dès le début.

Pour ne pas avoir à le faire pour chaque nouveau fichier, aller dans le menu **Paramétrage > Préférences, onglet Nouveau document/Dossier**. Sélectionnez UTF-8 sans BOM dans la liste.

La structure la plus simple d'une page web en HTML5 sera composée d'au minimum 4 Balises :

- La balise indiquant le [doctype](#)
- La balise `<html>` en tout début et `</html>` en tout fin de document
- Les balises `<head>` et `</head>` renfermant des informations utiles au navigateur mais non affichées
- Les balises `<body>` et `</body>` qui comme leur nom l'indique encadrent le corps de votre page

Lorsque vous codez un page, il est important d'utiliser des commentaires.
Ces commentaires ne seront ni visibles sur le navigateur ni interprétés par celui-ci.
Ils ne serviront qu'à vous repérer sur votre code.

Les commentaires se situent entre les signes `<!--` et `-->`

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8" />
    <!-- Section non affichable -->
    <!-- informations pour le navigateur -->
  </head>
  <body>
    <!-- Corps de la page web -->
    <p>Structure d'une page web en HTML5</p>
  </body>
</html>
```

Balises complémentaires :

<header> Informations d'introduction ou de présentation d'une page.

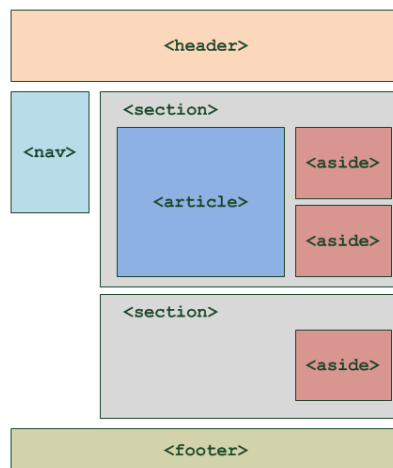
<nav> Éléments de navigations (menu).

<section> portion au centre de la page.

<article> Détermine une identité à part entière de la page : billet de blog, un commentaire...

<aside> : Qui correspond à une zone secondaire non liée au contenu principal de la page

<footer> Le pied car chaque section peut avoir un footer y compris la page entière



Les listes

- **** délimite toute la liste ;
- **** délimite un élément de la liste (une puce).
- Une liste ordonnée ****

Les liens :

La balise <a> permet de créer un lien vers un site ou une page

Pour créer un lien vers un site, exemple google.fr :

```
<a href="http://www.google.fr">Site de google</a>
```

Pour ouvrir dans une nouvelle page il faut ajouter `target="_blank"`

Pour créer un lien vers une page P4.html située dans le même répertoire que notre page courante :

```
<a href="P4.html">page 4 en html </a>
```

Pour une page située ailleurs que dans le répertoire de la page d'appel, il faut ajouter le chemin.

Pour télécharger un fichier : (qui sera dans le même répertoire que la page)

```
<a href="Document1.pdf">Télécharger le fichier</a>
```

Envoyer un mail :

```
<a href="mailto:c_hanrion@orange.fr">Envoyez-moi un e-mail </a>
```

Lien vers ancrage dans la page :

```
<a href="#Niveau1">Vers paragraphe 1 </a>
<a href="#Niveau2">Vers paragraphe 2 </a>
```

```
<h1> Zone à atteindre </h1>
<h2 id="Niveau1"> Paragraphe 1 </h2>
<p><br/><br/><br/><br/><br/><br/></p>
<h2 id="Niveau2"> Paragraphe 2 </h2>
```

Les figures :

```
<figure>
  
  <figcaption>Le logo de Loritz</figcaption>
</figure>
```

-

Préfixes :

- -webkit- pour Chrome 10 et suivants ainsi que pour Safari 5.1 et suivants,
- -moz- pour Firefox 3.6, et suivants,
- -ms- pour IE 10 et suivants,
- -o- pour Opera 11.10, et suivants.
-

Dégradé de couleur

background-image: linear-gradient(right top, #D60F0F 0%, #FFDD00 100%)

Couleur avec transparence rgba

rgba(255,15,15,0.5) .

La 4^{ème} valeur est comprise entre 0 et 1

0 : transparent

1 : opaque

Couleur RGB + la propriété : opacity: 0.6;

5 Les feuilles de styles

On parle de "Feuille de style" car on écrit le code CSS dans un fichier à part avec l'extension

« **.CSS** »

C'est un fichier dans lequel on décrit l'apparence que notre site doit avoir : la couleur et la police du texte, la taille des titres, la position des menus, la couleur ou l'image de fond etc...

Pour que vos pages HTML y fassent référence il faut ajouter une ligne en entête :

```
<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="Design clair" href="styles/clair.css" />
</head>
```

Titre du
style

Nom et
chemin

```
<title>Bienvenue sur mon site !</title>
<!-- codage en HTML 5 -->
<link rel="stylesheet" href="styles/S2.css" />
```

On aura donc 2 fichiers :

clair.css ou **S2.css** → feuille de styles

Nom.html → page Html

Dans un CSS, on trouve 3 éléments différents :

Des noms de balises : on écrit les noms des balises dont on veut modifier l'apparence. Par exemple, si je veux modifier l'apparence de tous les titres <h1>, je dois écrire **h1**

Des propriétés CSS : les "effets de style" de la page sont rangés dans des propriétés. Il y a par exemple la propriété *color* qui permet d'indiquer la couleur du texte, *font-size* qui permet d'indiquer la taille du texte etc etc...

Les valeurs : à chaque propriété CSS on doit indiquer une valeur. Par exemple, pour la couleur, il faut indiquer le nom de la couleur. Pour la taille, il faut indiquer quelle taille on veut.

Schématiquement, une feuille de style CSS ressemble à ça :

| | |
|---|--|
| balise1 { propriete: valeur; } | p { /*Tout le texte des paragraphes en vert taille 16*/ color: green; /*ou rgb(0,255,0)*/ font-size: 16px; } |
| balise2 { propriete: valeur; propriete: valeur; } | body { /* corps de page */ background-color : yellow; } h1 h2 { /*titres cumulés*/ color : blue } |

on peut **imbriquer** les balises :

```
h3 em /* => tous les em situés à l'intérieur d'un h3 */  
{  
  color: red;  
}
```

Exemple d'attributs :

La couleur : soit un **nom de couleur** ("black", "red"...), soit une valeur **hexadécimale** (#FF0000), soit une **valeur rgb** (rgb(198, 212, 37))

Le type de bordure : Votre bordure peut être un simple trait, ou des pointillés, ou encore des tirets etc... Voici les différentes valeurs disponibles :

none : pas de bordure (par défaut)

solid : un trait simple.

dotted : pointillés.

dashed : tirets.

double : bordure double.

groove : en relief.

ridge : effet 3D.

inset : autre effet 3D (on a l'impression que le bloc forme un creux).

outset : encore un autre effet 3D (on a l'impression que le bloc est surélevé).

marge : padding, margin (côté du bloc)

Le plus gros avantage est que pour modifier l'apparence de toutes les pages il suffit de modifier la page CSS utilisée.

Pour créer des pages HTML il existe des logiciels permettant de créer via une interface graphique les pages et de générer du code. (DREAMWEAVER, KOMPOZER par exemple)

Les sélecteurs de classe

Les sélecteurs de classe vous permettent d'appliquer le même style à un groupe d'éléments en leur donnant à tous le même attribut de `class`. En CSS3

```
1 .class { ... }
```

Un point (.) précède le nom de la classe. Il est permis d'utiliser l'attribut de classe même sur de multiples éléments sur une page.

```
1 .rouge {  
2     color : red;  
3 }  
  
1 <p class="rouge">Ce texte sera affiché en rouge</p>  
2 <p>Ce deuxième paragraphe ne le sera PAS</p>
```

Les sélecteurs d'ID (identifiant)

Les sélecteurs d'ID sont semblables à des sélecteurs de `class` mais ils sont utilisés pour cibler un élément UNIQUE sur le document. En CSS3 :

```
1 #identifiant { ... }
```

Un symbole diese (#) précède le nom de l'identifiant. Les ID sont utilisés une fois par page et devrait idéalement être réservés à des éléments significatifs.

```
1 #rouge {  
2     color : red;  
3 }  
  
1 <p id="rouge">Ce texte sera affiché en rouge</p>  
2 <p>Ce deuxième paragraphe ne le sera PAS</p>
```

Combiner plusieurs sélecteurs

Le CSS3 a la capacité de combiner les sélecteurs et d'hériter des styles. Cela vous permet de commencer avec un sélecteur générique et de continuer à travailler de façon plus spécifique. En outre, vous pouvez combiner des sélecteurs différents pour être le plus précis possible

```
1 <p id="rouge">Le mot en gras est en <strong>ROUGE</strong> </p>.  
2 <p>Le mot en gras est en <strong class="vert">Vert</strong> </p>.  
3 <p>Le mot en gras est en <strong>Bleu</strong> </p>.  
  
1 strong {  
2     color : blue ;  
3 }  
4 #rouge strong {  
5     color : red ;  
6 }  
7 strong.vert {  
8     color : green ;  
9 }
```

Image de fond :

```
body {  
  background-color: #FFCC66;  
  background-image: url("papillon.gif");  
  background-repeat: no-repeat;  
  background-attachment: fixed;  
  background-position: 50% 50%;  
}
```

Couleur du fond

imag

unique

Fixée : ne défile

Position : défaut = top left
center = 50%
bottom = 100%

Taille du texte :

```
#texte1
```

```
{  
  font-size: 18px;  
}
```

Taille en pixels

```
#texte2
```

```
{  
  font-size: small;  
}
```

Taille prédéfinie : de xx-small à xx-large

```
#texte3
```

```
{  
  font-size: 1.3em;  
}
```

Proportionnelle à la Taille par défaut

Police :

Liste des polices possibles, le navigateur utilisera la première trouvée.

```
#texte4
```

```
{  
  font-family: "Arial Black", Arial, Verdana;  
}
```

font-style: italic; (ou normal)

font-weight: bold; (gras)

text-decoration: underline; (souligné, line-through : barré, blink : clignotant)

text-align: center; (left, right, justify) ne fonctionne qu'avec les balises block

```
.imageflot  
{  
  float: left;  
}  
.dessous  
{  
  clear: both;  
}
```

```
<p></p>  
<p>texte affiché à côté de l'image flottante.</p>  
<p class="dessous">texte affiché sous l'image flottante.</p>
```

Image et texte flottants

```

.boite
{
box-shadow: 6px 6px 6px black;
}
.texte-ombre
{
text-shadow: 2px 2px 4px black;
}

```

Si vous ajoutez l'attribut **inset** l'ombre sera à l'intérieur du cadre

Réaction aux actions - passage de la souris : **hover**

```

p: hover /* Quand on survol un paragraphe */
{}
a: active /* Quand on clique sur le lien */
{
background-color: #FF0000;
}
a: focus /* Quand on sélectionne le lien */
{
background-color: #00FF00;
}

```

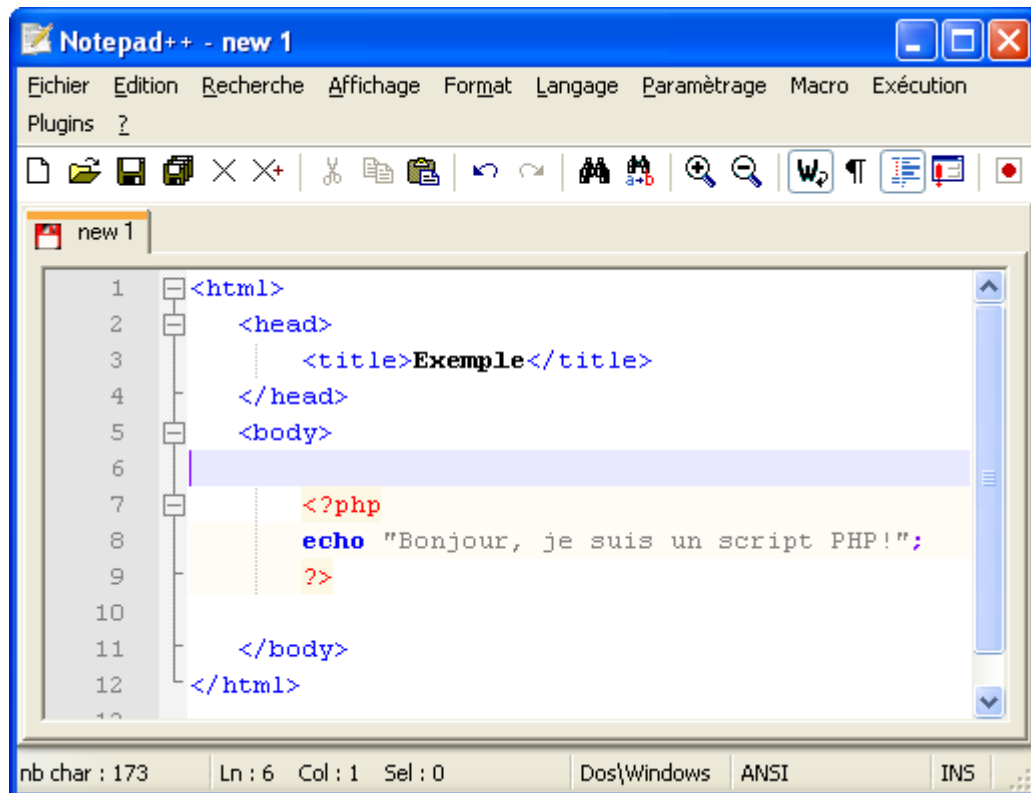
Validation du codage de votre site : <http://validator.w3.org>.

6 Langage PHP

Il est à noter la différence avec les autres scripts CGI écrits dans d'autres langages tels que le Perl ou le C . Au lieu d'écrire un programme avec de nombreuses lignes de commandes afin d'afficher une page HTML, vous écrivez une page HTML avec du code inclus à l'intérieur afin de réaliser une action précise (dans ce cas là, afficher du texte).

Le code PHP est inclus entre une **balise de début** et une **balise de fin** qui permettent au serveur web de passer en "mode PHP".

<?php
?>



interprété :

```
<html>
<head>
<title>Exemple</title>
</head>
<body>
<p>Bonjour, je suis un script PHP ! </p>
</body>
</html>
```

Dans Navigateur :

Bonjour, je suis un script PHP !

Si on veut avoir les informations sur le serveur PHP installé il suffit d'insérer la commande **phpinfo()** dans une page et lors de l'ouverture de la page la fonction affichera l'ensemble des infos.

```
<body>
    <?php phpinfo(); ?>
</body>
```

6.1 Les commandes de sortie :

Exemple :

```
<?php
echo "\ "PHP\ " ";    → "PHP"
?>
```

Affichera dans votre navigateur le texte suivant : "PHP". Mais vous auriez pu faire de même avec la fonction print() :

```
<?php
print( "\ "PHP\ " " );
?>
```

PHP possède aussi un but intéressant c'est de pouvoir mêler aisément PHP et HTML :

Exemple :

Affichage de "bonjour" en gras :

```
<?php
echo "<strong>bonjour</strong>";
?>
```

code HTML généré : bonjour

Affichage d'une image :

```
<?php
echo "<img src=\"image.gif\"/>";
?>
```


commentaires

```
<?php
echo "J'habite a NANCY.<br/>"; // cette ligne indique où j'habite
/*
plusieurs lignes de commentaires
*/
// la ligne suivante indique mon âge
echo "J'ai 22 ans.";
?><br/>
```

6.2 Terminer l'exécution du script

on utilise **exit;**

6.3 Commenter le code

Commentaire sur une ligne: // ou #

Commentaire sur plusieurs lignes: /* ... */

6.4 Les types de variables

PHP supporte les huit types basiques suivants :

PHP supporte quatre types scalaires :

- booléen
- entier
- nombre à virgule flottante
- chaîne de caractères

PHP supporte deux types composés :

- tableau
- objet

PHP supporte deux types spéciaux :

- ressource
- NULL**

Les variables sont précédées du signe \$, quelque soit leur type.

Leur déclaration est optionnelle.

Il n'y a pas de type pointeur.

Les variables PHP sont à **typage faible**. C'est PHP qui décide de son type lors de l'affectation.

La portée des variables est la même que pour le langage C. (limitée au bloc où elles sont déclarées)

6.4.1 Variables scalaires

Les variables scalaires sont de 3 types : chaîne, entiers, réels.

Exemple :

```
$a = 15 ;
```

```
$b= " coucou " ;
```

```
$c=1689.126 ;
```

Un nom de variable (ici a, b et c) peut contenir des chiffres, des lettres ou le caractère _ **mais jamais** de caractères spéciaux comme : **é, è, !, ., ;, ° ...**

6.4.2 Tableaux

Déclaration

```
$fruits = array();
```

Affectation

```
$fruits[0]= "pomme";
```

```
$fruits[1]= "banane";
```

```
$fruits[] .= "orange"; // equivaut a $fruits[2]= "orange"
```

ou

```
$fruits= array( "pomme", "banane", "orange" );
```

Fonctions relatives

sizeof()

Renvoie le nombre d'éléments d'un tableau.

```
$nb_elements= sizeof( $tablo );
```

is_array()

Renvoie `true` si la variable est de type tableau, `false` sinon.

Reset()

`reset($tablo)` place le pointeur interne sur le premier élément du tableau: Chaque variable tableau possède un pointeur interne repérant l'élément courant.

End()

`end($tablo)` place le pointeur interne du tableau sur le dernier élément du tableau.

Current()

renvoie l'élément courant du tableau.

Next()

déplace le pointeur vers l'élément suivant, et renvoie cet élément. Renvoie `false` s'il n'existe pas `prev()`

déplace le pointeur vers l'élément précédent, et renvoie cet élément. Renvoie `false` s'il n'existe pas

each()

`$a=each($tablo)` renvoie l'index et la valeur courante dans un tableau à 2 éléments;

`$a[0]` contient l'index, `$a[1]` la valeur.

List()

`list($scalar1, $scalr2, ...)` construit un tableau temporaire à partir des variables scalaires passées en argument.

Key()

`key($tablo)` renvoie l'index de l'élément courant du tableau.

sort, rsort, usort, uasort

Ce sont différentes fonctions de tri de tableau.

- `sort` trie par valeurs croissantes : `$tablo_trie = sort($tablo);`

- `rsort` par valeurs décroissantes

- `usort` et `uasort` permettent au programmeur d'implémenter lui-même la fonction de tri utilisée.

Exemple d'affichage d'un tableau défini par `$tableau` :

```
while (list ($cle, $valeur) = each ($tableau))  
{  
    echo "$cle => $valeur<br\>\n";  
}
```

Tableaux associatifs

Un tableau associatif est un tableau dont l'index est une **chaîne de caractère** au lieu d'un nombre. On parle aussi de " hash array " ou " hash ". Il se déclare comme un tableau traditionnel, la distinction se fait lors de l'affectation.

Déclaration

```
$calories= array(); // comme un tableau
```

Affectation

Affectons un nombre de calories moyen aux fruits :

```
$calories["pommes"]= 80;  
$calories["banane"]= 130;  
$calories["litchie"]= 30;
```

Fonctions relatives

isset

Pour tester l'existence d'un élément, on utilise la fonction `isset()` :

```
if(isset($calories["pommes"]))
{
    echo "une pomme contient ". $calories["pommes"] . " calories\n";
} // une pomme contient 80 calories
else
{
    echo "pas de calories definies pour la pomme\n";
}
```

asort, arsort, ksort, ksort

Ces fonctions de tri conservent la relation entre l'index et la valeur, généralement le cas dans un tableau associatif.

asort trie par valeurs croissantes, **arsort** par valeurs décroissantes

ksort trie par index (key) croissantes

6.5 OPERATEURS

PHP dispose des opérateurs classiques du C ainsi que d'autres inspirés du Perl.

6.5.1 Comparaison

| | |
|--------------------|--|
| \$a==\$b | Egal - Résultat vrai si \$a est égal à \$b |
| \$a!=\$b | Différent - Résultat vrai si \$a est différent de \$b |
| \$a<\$b | Inférieur - Résultat vrai si \$a est strictement inférieur à \$b |
| \$a>\$b | Supérieur - Résultat vrai si \$a est strictement supérieur à \$b |
| \$a<=\$b | Inf ou égal - Résultat vrai si \$a est inférieur ou égal à \$b |
| \$a>=\$b | Sup ou égal - Résultat vrai si \$a est supérieur ou égal à \$b |

6.5.2 Logiques

Les opérateurs logiques sont utilisés dans les tests, par ex. dans un

```
if( condition )           if (($a>$b) &&($c >$b) || ($c<$d))
                           if (($a>$b) AND ($c>$b) OR ($c<$d))
```

| | | |
|-------------------|-------------|------------------------|
| && | et | |
| | ou | <i>Alt GR 6 2 fois</i> |
| xor | ou exclusif | |
| ! | negation | |

Note: les opérateurs **and**, **or** , **not** sont également disponibles et font la même chose.

6.5.3 Arithmétiques

| | |
|----------------|---|
| \$a+\$b | Addition Somme de \$a et \$b. |
| \$a-\$b | Soustraction Reste de la différence de \$b et \$a. |
| \$a*\$b | Multiplication Produit de \$a par \$b. |
| \$a/\$b | Division Dividende de \$a par \$b. |
| \$a%\$b | Modulo Reste de la division entière de \$a par \$b. |

l'opérateur / renvoie un entier si les 2 opérandes sont des entiers, sinon il renvoie un flottant.

6.5.4 Affectation

= affectation
+= addition puis affectation
-= soustraction puis affectation
*= multiplication puis affectation
/= division puis affectation
%= modulo puis affectation
.= concaténation puis affectation d'une chaîne
`$res .= $chaîne ; $res .= " "`

Exemple de concaténation d'une chaîne :

```
<?php
$Var = "ODERIE";
$var = "Agathe";
$chaîne= "Votre nom est" ;
echo $chaîne ." " . $var ." " . $Var ;
?>
```

Votre nom est Agathe ODERIE

6.6 STRUCTURES DE CONTROLE

6.6.1 Si : if

```
<?php
...
if ($password == "1234") {
    echo "Mot de passe valide !";
} else {
    echo "Mot de passe incorrect !";
}
...
?>
```

Dans le cas de plusieurs tests successifs portant sur une même variable, on utilisera plutôt le test *switch*.

Note : Si le corps du test ne comporte qu'une instruction, les accolades {} sont optionnelles.

6.6.2 switch

Le **switch** n'a pas d'équivalent en Perl. Il est équivalent au *SELECT CASE* en Basic. Il permet de confronter une variable à plusieurs valeurs prédéfinies. Il permet un code plus compact et lisible qu'un test *if-elseif-elseif...*

La valeur de *[variable]* est comparée successivement à chaque **case**. S'il y a égalité, le bloc d'instruction est exécuté.

Il ne faut pas omettre le **break** en fin de bloc, sans quoi le reste du switch est exécuté.

Le handler **default** permet de définir

```
<?php
...
switch ($var) {
    case 1:
        echo "La valeur de var est 1";
        break;
    case 2:
        echo "La valeur de var est 2";
        break;
    case 3:
        echo "La valeur de var est 3";
        break;
    default :
        echo "La valeur de var est indéfinie";
}
...
?>
```

des instructions à effectuer par défaut, c'est à dire si aucun **case** n'a " fonctionné ".

```
switch($prenom){
    case "julien";
    echo "bonjour $prenom ! vous etes un garçon";
    break;
    case "patricia" ;
    echo "bonjour $prenom ! vous etes une fille";
    break ;
    default ;
    echo "Bonjour $prenom !" ;
    break ;
}
```

6.7 Boucles

En PHP, on dispose des structures de boucle similaires au C.

L'instruction **break** permet de sortir d'une boucle à tout moment.

L'instruction **continue** permet de revenir au début de la boucle.

For

```
<?php
...
for ($i=0; $i<5; $i++) { //tout est sur la même ligne
    echo "itération numéro ".$i."<br>";
}
...
?>
```

while

```
<?php
...
$i = 0;
while ($i < 5)
{
    echo "itération numéro ".$i."<br>";
    $i++;
}
...
?>
```

do .. while

La condition de sortie est située en fin de boucle. Ainsi la boucle est parcourue une fois au minimum.

```
<?php
$fp= fopen( "monfichier.txt" );
do
{
    $ligne = fgets( $fp, 1024 );
    echo " $ligne " ;
}
while(!feof($fp));
?>
```

6.8 FONCTIONS

Il n'y a **pas de distinction fonctions / procédures** en PHP.

Les fonctions PHP prennent de 0 à n paramètres. Ces paramètres peuvent être de type quelconque.

Note : Il faut **déclarer** la fonction en **amont** de son utilisation, contrairement au C.

Dans le cas contraire, PHP sort une erreur du type

Call to unsupported or undefined function (fonction) in (file) on line (number)

On ne peut pas déclarer le prototype d'une fonction.

Les fonctions peuvent ou non renvoyer un résultat. Dans ce cas, on utilise l'instruction **return**. La variable retournée peut être de type quelconque. Elle est transmise par copie.

Exemple d'une fonction retournant la somme de 2 nombres passés en paramètres :

```
<?php
...
function somme($a,$b)
{
    $resultat=$a+$b ;
    return $resultat ;
}
...
$total=somme(45,12) ; // $total vaut 45+12
...
?>
```

```
$couleur= " bleu " ; // variable globale
function mafonc()
{
    global $couleur;
    ...
}
```

Par défaut, les variables globales (accessibles par tout le script) ne sont pas connues à l'intérieur du corps d'une fonction. On peut cependant y accéder à l'aide du mot-clé **global**.

6.8.1 Paramètres optionnels

Les paramètres optionnels sont autorisés : il suffit de leur affecter une valeur par défaut.

```
<?php
...
function myfunc( $param1 = "inconnu", $param2="" )
{
    echo "param1=$param1 param2=$param2\n";
}
myfunc("toto","titi"); // affiche "param1=toto param2=titi"
myfunc("toto"); // affiche "param1=toto param2="
myfunc(); // affiche "param1=inconnu param2="
...
?>
```


6.8.2 Passage de paramètres par adresse analogue au passage par référence

Par défaut, les paramètres sont transmis par **copie**, c'est à dire que la fonction possède une copie locale de la variable envoyée.

On peut cependant passer un paramètre par adresse en le précédant de **&**.

Cela permet de modifier ce paramètre dans la fonction.

```
<?php
...
function refparam( $param )
{
    $param .= "...modifié"; // .= concatenation de chaine
}

$variable= "le parametre";
refparam( &$variable );
echo $variable; // affiche "le parametre...modifié"
...
?>
```

Exemple de fonctions toutes faites :

gestion de la date et heure

```
<?
// Enregistrons les informations de date dans des variables

$jour = date("d");
$mois = date("m");
$annee = date("Y");

$heure = date("H");
$minute = date("i");

// Maintenant on peut afficher ce qu'on a recueilli
echo "Bonjour ! Nous sommes le $jour/$mois/$annee et il est $heure h $minute.";
?>
```

```
echo "Bonjour ! Nous sommes le ".$jour." / ".$mois." / ".$annee. "
<br/>et il est " . $heure. " h " . $minute."<br/>" ;
```

Adresse pour la liste et description des fonctions :

<http://www.php.net>

<http://fr2.php.net/manual/fr/funcref.php>

7 Les variables prédéfinies

\$_GET : donne les valeurs des informations indiquées dans l'url.

\$_POST : informations issues d'un formulaire.

\$_SERVER : ce sont des valeurs utiles que nous donne le serveur.

\$_SERVER['PHP_SELF'] : donne le chemin de la page que vous êtes en train d'exécuter, par rapport à la racine de votre site web.

\$_SERVER['HTTP_REFERER'] : donne l'url de la page qui a amené le visiteur sur la page courante.

\$_SERVER['REMOTE_ADDR'] : Elle nous donne l'adresse IP du client.

\$_ENV : variables d'environnement, données par le système d'exploitation (Linux)

\$_SESSION : variables de session.

\$_COOKIE : valeurs des cookies enregistrés sur l'ordinateur du visiteur.

\$_FILES : utilisée pour l'envoi des fichiers sur le serveur à partir d'un formulaire.

7.1 Format d'une URL

Les URL (Uniform Resource Location) sont les chemins des ressources sur internet. Ils sont uniques et permettent d'accéder à n'importe quel document.

7.1.1 Syntaxe générale

<schéma>://<utilisateur>:<mot de passe>@<machine>
:<port>/<chemin>?<paramètres>#<fragment>

| Champ | Description |
|------------------------|--|
| <i>schéma</i> | protocole utilisé (http, ftp, https ...) |
| <i>machine</i> | nom de domaine ou adresse IP du serveur |
| <i>utilisateur</i> | nom d'utilisateur pour certains protocoles nécessitant une |
| identification | (ftp, telnet, ...) |
| <i>mot de passe</i> | mot de passe pour l'authentification |
| <i>port</i> | n° de port sur lequel se connecter, (80 pour http) |
| <i>chemin</i> | chemin complet de la ressource |
| <i>paramètres</i> | liste des noms de paramètres et de leurs valeurs |
| <i>fragment</i> | identifiant d'un lien interne à une page HTML |
| (créé par) | |

7.1.2 Syntaxe de la liste des paramètres

Le champs <paramètres> ci-haut peut être décomposé:

<couple (nom,valeur)>&<couple (nom,valeur)>&...

7.1.3 Syntaxe du couple (nom, valeur)

Le champs <couple (nom,valeur)> peut être décomposé: nom=valeur

7.1.4 Résumé

Le caractère ? indique que la suite de l'URL sont des paramètres et ne font pas partie du nom de fichier. Le caractère = sépare un nom de paramètre et sa valeur transmise.

Le caractère & sépare deux couples (nom, valeur).

Pour faire face au cas général d'un paramètre dont la valeur contient des caractères interdits, on utilisera les fonctions de codage.

7.1.5 Exemple concret

Version imprimable

Dans cet exemple on transmet deux variables au script index.php : \$imprim de valeur "yes" et \$user_id de valeur "75".

Le Symbole & remplacé par &

*http://www.monsite.com/infos.php? jour=6&mois=11&a
mp;annee=2007&titre=Dates*

Tous les & seront transformés en symboles & par le navigateur Internet du visiteur.

Ici, 4 variables seront créées.

- \$_GET[' jour '] = 6 ;
- \$_GET[' mois '] = 11 ;
- \$_GET[' annee '] = 2007 ;
- \$_GET[' titre '] = "Dates" ;

echo "Param recus : ".\$_GET['titre']." Et ".\$_GET['jour']. ;

7.2 Passage de paramètres entre pages

2 pages à créer ► 1 pour l'appel ► l'autre pour la cible

Exemple de la page d'appel : appel.php

```
<p>
    Notez que cette page ne contient que du HTML.<br />
    Voici 3 liens vers la page cible.php, avec des variables
    aux valeurs différentes :
</p>
-
<p>
    <a href="cible.php?nom=KESKIDI&prenom=Aimé">Lien vers
    cible.php?nom=KESKIDI&prenom=Aimé</a><br />
    <a href="cible.php?nom=AIMARD&prenom=Jean">Lien vers
    cible.php?nom=AIMARD&prenom=Jean</a><br />
    <a href="cible.php?nom=MIKAO&prenom=Emma">Lien vers
    cible.php?nom=MIKAO&prenom=Emma</a>
</p>
```

Nom du fichier
du script php
cible

Exemple de la page cible : cible.php

<body> en html
echo " <body> " ; // en php

```
<p>Bonjour !</p>
<p>Votre nom est <?php echo $_GET['nom']; ?>
, et votre prénom est <?php echo $_GET['prenom']; ?>.</p>
<p>Ca fait <?php echo $_GET['prenom']; ?>
<?php echo $_GET['nom']; ?>.</p>
<p>Faites un autre essai, <a href="appel.php">cliquez ici</a>
pour revenir à appel.php</p>
```

Nom du fichier
du script php
d'appel

`$_GET['nom']` : un élément d'un tableau associatif

7.3 Les éléments du formulaire

7.3.1 Les edit box (petite zone de saisie)

Une zone de texte ressemble à cela :

En HTML, on l'insère tout simplement avec la balise : `<input type="text" />`

Pour les mots de passe, vous pouvez utiliser `type="password"`, ce qui aura pour effet de cacher le texte saisi par le visiteur.
il y a 2 attributs à ajouter :

name : c'est le nom de la zone de texte.

```
<input type="password" name="Mdp" />
```

```
$_POST['Mdp']
```

Achtung à la casse !

Par exemple : `<input type="text" name="Pseudo" />`

Dans cible.php une variable `$_POST['Pseudo']` sera utilisable

value : c'est ce que contient la zone de texte au départ. Par défaut, la zone de texte est vide. Mais il peut être très pratique de pré-remplir le champ

```
<input type="text" name="pseudo" value="Marcel54" />
```

Les différents types :

`"tel", "email", "number", "url", "range", "date",`

7.3.2 Une grosse zone de texte:

```
<textarea name="message" rows="8" cols="45">
Votre message ici.
</textarea>
```

le texte par défaut est ici écrit entre le `<textarea>` et le `</textarea>`.

7.3.3 La liste déroulante :

On utilise le code HTML suivant :

```
echo " <select name=\"choix\" required > " ;
```

```
<select name="choix">
    <option value="ch1">Choix 1</option>
    <option value="choix2">Choix 2</option>
    <option value="choix3">Choix 3</option>
    <option value="choix4">Choix 4</option>
</select>
```

Ici, une variable `$_POST['choix']` sera créée, et elle contiendra le choix qu'a fait l'utilisateur. S'il a choisi "Choix 3", la variable `$_POST['choix']` sera égale au **value** correspondant, c'est-à-dire "choix3".

Valeur par défaut :

```
<option value="choix3" selected="selected">Choix 3</option>
```

Permet de définir un groupe :

```
<optgroup label="label1">
</optgroup>
```

7.3.4 Les cases à cocher :

```
<input type="checkbox" name="case" /> Ma case à cocher
```

- Si la case est cochée, alors `$_POST['case']` aura pour valeur "on".
- Si elle n'est pas cochée, alors `$_POST['case']` ne contiendra rien (NULL).

Si vous voulez que la case soit cochée par défaut, il faudra lui rajouter l'attribut

checked="checked". Par exemple :

```
<input type="checkbox" name="case" checked="checked" />
```

7.3.5 Les boutons d'option :

Les boutons d'option fonctionnent par groupes de 2 minimum.

Aimez-vous l'informatique ?

```
<input type="radio" name="info" value="oui" checked="checked" /> Oui
<input type="radio" name="info" value="non" /> Non
```

Dans la page cible, une variable `$_POST['info']` sera créée.

Elle aura la valeur du bouton d'option choisi par le visiteur.

Si on aime l'informatique, alors on aura `$_POST['info'] = 'oui'`.

Penser à remplir l'attribut **"value"** du bouton d'option car c'est lui qui va déterminer la valeur de la variable.

7.3.6 Les champs cachés :

C'est du code dans un formulaire invisible pour le visiteur, qui va créer une variable avec une valeur.

```
<input type="hidden" name="pseudo" value="Marcel54" />
```

A l'écran, vous ne verrez rien.

Mais dans la page cible, une variable `$_POST['pseudo']` sera créée (correspondant à *name*), et elle aura la valeur **"Marcel54"** (correspondant à *value*).

Encadrement d'une partie du formulaire

```
<fieldset>
```

```
    <legend>Rubrique1</legend> <!-- Titre du fieldset -->
```

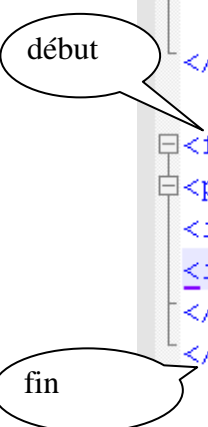
```
</fieldset>
```

autofocus curseur sur la zone

required champ obligatoire

7.4 Transmettre en utilisant un formulaire

Exemple de page d'appelForm :



```
<p>
    Cette page, elle aussi, ne contient que du HTML.<br />
    Veuillez taper votre prénom :
</p>
<form action="cibleForm.php" method="post">
<p>
    <input type="text" name="prenom" />
    <input type="submit" value="Valider" />
</p>
</form>
```

The diagram shows a code editor window with the above HTML code. A speech bubble labeled 'début' points to the first paragraph tag, and another labeled 'fin' points to the closing form tag.

Exemple de page cibleForm :

```
<p>Bonjour !</p>

<p> Tu t'appelles comment <? echo $_POST['prenom']; ?> !</p>

<p>Si tu veux changer de prénom, <a href="appelForm.php">
clique ici</a> pour revenir à appelForm.php</p>
```

La page d'appel est un formulaire : "action" indique la page à afficher (*cibleForm.php*)

```
<input type="text" name="prenom" />
```

Ici le nom de la zone de texte est "prenom" .

Dans la page *cibleForm.php*, une variable `$_POST['prenom']` sera créée, elle aura pour valeur ce que vous aurez entré dans la zone de texte.(tableau associatif)

Règles :

Quand on récupère les valeurs d'un formulaire, on utilise le préfixe `$_POST['xxxx']`.

Quand on récupère les valeurs depuis l'adresse, on utilise le préfixe `$_GET['xxxx']`

7.5 Les sessions et les cookies :

Lorsque vous créez un site web, vous avez rapidement besoin de stocker et d'afficher des informations sur vos utilisateurs pour les distinguer ou leur attribuer des droits d'utilisation.

Le problème principal des langages comme PHP, c'est que les variables n'ont qu'une durée de vie limitée à celle du script qui les appelle (portée des variables).

Pour conserver des données de page en page, il faut alors les passer par la méthode **GET** ou **POST**

Il existe deux moyens de stockage d'informations sous la forme d'un fichier enregistré sur le disque

Le **cookie** qui sera côté client.

La **session** qui sera côté serveur.

Il suffit d'enregistrer les variables dans le fichier depuis le script 1 pour les lire ensuite dans le script 2.

Le problème majeur du **cookie** c'est que le client a le pouvoir de le **refuser**. Votre application risque donc de ne pas pouvoir fonctionner.

Il y a aussi des risques de sécurité : **l'usurpation d'identité**, car ce fichier peut être recopié facilement sur un autre ordinateur, **la manipulation**, le cookie n'étant qu'un simple fichier texte il est facile de modifier son contenu.

7.5.1 Les sessions

La **session** n'aura donc pas cet inconvénient puisque tout est géré sur le **serveur de l'application** auquel le client n'a pas d'accès direct.

Vous n'avez plus à vous soucier du passage des informations de page en page. Vos URL gagnent ainsi en clarté et vos formulaires HTML sont allégés.

Les **données sont stockées** dans la session.

Attention le client reste **maître** du déroulement de l'application. Seule une action de sa part permet l'envoi des informations au serveur pour la génération de la page suivante par **GET** ou **POST**.

La session et le cookie ne permettent le stockage que de variables de **type primitif**. Ce sont les chaînes de caractères, les nombres (entier, réel) mais également les tableaux.

La **durée de vie** d'une session est définie au niveau serveur. Par défaut sous **EasyPHP**, elle est paramétrée à **180 minutes**. Cela veut dire que si votre client laisse son navigateur ouvert sur votre application pendant 3 heures et 1 seconde sans exécuter d'action (GET ou POST), sa session sera détruite automatiquement par le serveur.

Il vous appartient de paramétrer cette variable en fonction de vos besoins. Certains hébergeurs ne permettent pas de modifier cette durée.

Une session en PHP repose sur un identifiant (une suite de chiffres et de lettres), pour la sécurité des informations contenues on peut coupler la session avec **un cookie ou en utilisant l'adresse IP** de l'utilisateur.

Pour utiliser les sessions, PHP vous propose un éventail de commandes :

| | |
|--|--|
| <code>session_cache_expire</code> | Obtenir la configuration du cache expiré |
| <code>session_cache_limiter</code> | Lecture/écriture pour le limiteur de cache |
| <code>session_decode</code> | Décode les données de session |
| <code>session_destroy</code> | Détruit une session |
| <code>session_encode</code> | Encode les données de session |
| <code>session_get_cookie_params</code> | Lit la configuration du cookie de session |

| | |
|--|--|
| <code>session_id</code> | Lecture/écriture pour l'identifiant de la session courante |
| <code>session_module_name</code> | Lecture/écriture pour le module de session courant |
| <code>session_name</code> | Lecture/écriture pour le nom de la session |
| <code>session_readonly</code> | Initialise une session en mode lecture |
| <code>session_save_path</code> | Lecture/écriture pour le chemin de sauvegarde des sessions |
| <code>session_set_cookie_params</code> | Modifie les paramètres du cookie de session |
| <code>session_set_save_handler</code> | Configure les fonctions de stockage de sessions |
| <code>session_start</code> | Initialise une session |
| <code>session_unset</code> | Détruit toutes les variables de session |
| <code>session_write_close</code> | Ecriture de données et fermeture de la session |

Demander au compilateur PHP de démarrer une session pour le client.

```
<?php
session_start();
?>
```

L'identifiant de session peut être affiché par la commande `session_id()`.

Vous pouvez également gérer vous-même ce nom de session en utilisant `session_name()` avant le démarrage de la session.

La session est perdue définitivement pour l'utilisateur :

Au delà de la durée de vie paramétrée par `session.cache_expire`.

A la **fermeture** de son navigateur.

Si la commande `session_destroy()` est appelée.

Exemple 1^{ère} page :

```
<?php
session_start();
?>
<html>
<body>
<form method="POST" action="page2.php">
Entrez votre nom : <input type="TEXT" name="nom">
<input type="SUBMIT" value="OK">
</form>
</body>
</html>
```

Page 2.php

```
<?php
session_start();
$nom = $_POST['nom'];
$_SESSION['nom'] = $nom;
?>
<html>
<body>
Bienvenue sur ce site <b><?php echo $nom; ?></b>.
<br />
Regardons ce qui se passe sur la
<a href="page3.php">page</a> suivante.<br />
</body>
</html>
```

Page3.php

```
<?php
    session_start();
    if ( isset ( $_SESSION['nom'] ) ) {
        $nom = $_SESSION['nom']; }
    else {
        $nom = "mais votre nom a été effacé"; }
    ?>
<html>
<body>
    Vous êtes toujours parmi nous
    <b><?php echo $nom; ?></b>.<br />
    Effacement de votre nom en cliquant
    <a href="page4.php">ici</a>.<br />
    Effacement de votre session en cliquant
    <a href="page5.php">ici</a>.<br />
</body>
</html>
```

page4.php (permet l'effacement d'une variable session)

```
<?php
    session_start();
    ?>
<html>
<body>
    <?php
        unset ( $_SESSION['nom'] );
        if ( isset ( $_SESSION['nom'] ) ) {
            $resultat = "La suppression a échouée ."; }
        else {
            $resultat = "Votre nom a été effacé."; }
        echo $resultat;
    ?>
    <br />
    Repartons en <a href="page3.php">arrière</a>.<br />
</body>
</html>
```

Page5.php (détruit une session)

```
<?php
    session_start();
    session_destroy();
    ?>
<html>
<body>
    Votre session a été détruite.
</body>
</html>
```

7.5.2 Les cookies

\$_COOKIE

fichier texte de maximum 4 Ko stocké chez le client

accessible par php

ou envoyé volontairement

fonction à placer en tout premier du code php (à cause de la génération de l'entête html)

SET_COOKIE(name,value,expire)

Exemple :

Création :

```
<?php
Set_Cookie('LeMien','Cours de PHP Master1',time()+24*3600) ;
?>
```

Utilisation :

```
<?php
if (isset($_COOKIE['LeMien']))
{ echo htmlentities ($_COOKIE['LeMien'],ENT_QUOTES,) ;
  }
?>
```

Détruire un cookie :

```
<?php
/*
 * Les deux exemples suivants sont équivalents
 */
setcookie('cookie_name'); // exemple 1

setcookie('cookie_name', '', 1); // exemple 2
?>
```

Sécurité :

Chiffrer le contenu en utilisant un contrôle de type Md5.

Quatrième argument ,'/rep1'

Le cookie ne pourra être lu que par un script se trouvant dans le répertoire rep1 ou dans un de ses sous-répertoires.

7.5.2.1 Description

bool **setcookie** (string \$name [, string \$value [, int \$expire [, string \$path [, string \$domain [, bool \$secure [, bool \$httponly]]]]]])

setcookie() définit un cookie qui sera envoyé avec le reste des en-têtes.

Comme pour les autres en-têtes, les cookies doivent être envoyés *avant* tout autre sortie (c'est une restriction du protocole HTTP, pas de PHP).

Cela vous impose d'appeler cette fonction avant toute balise `<html>` ou `<head>`.

Une fois que le cookie a été placé, il est accessible dans les variables globales `$_COOKIE` ou bien `$HTTP_COOKIE_VARS` arrays.

Notez que les superglobales telles que `$_COOKIE` sont disponibles en PHP depuis la version 4.1.0. Les valeurs de cookies existent aussi dans la variable `$_REQUEST`.

[Liste de paramètres](#)

Tous les arguments sauf *name* (nom) sont optionnels. Si seul le nom est présent, le cookie portant ce nom sera supprimé du navigateur de l'internaute. Vous pouvez aussi utiliser une chaîne vide comme valeur, pour ignorer un argument. Comme l'argument *expire* est un entier, il ne peut pas être ignoré avec une chaîne vide, vous devez utiliser le zéro pour cela (0).

name

Le nom du cookie.

value

La valeur du cookie. Cette valeur est stocké sur l'ordinateur du client ; ne stocker pas d'informations importantes. Le paramètre *name* est le 'cookienam', cette valeur est retrouvé en utilisant `$_COOKIE['cookienam']`.

expire

Le temps après lequel le cookie expire. C'est un timestamp Unix, donc, ce sera un nombre de secondes depuis l'époque Unix (1 Janvier 1970). En d'autres mots, vous devriez fixer cette valeur à l'aide de la fonction `time()` et en y ajoutant le nombre de secondes après lequel on veut que le cookie expire. Vous pouvez utiliser aussi `mktime()`. `time()+60*60*24*30` fera expirer le cookie dans 30 jours. Si vous ne spécifiez pas ce paramètre ou s'il vaut 0, le cookie expirera à la fin de la session (lorsque le navigateur sera fermé).

Note: Vous pourrez noter que le paramètre *expire* prend un timestamp unique, et non pas la date au format Jour, JJ-Mois-AAAA HH:MM:SS GMT, car PHP fait la conversion en interne.

Le paramètre *expire* est comparé avec le temps du client qui peut être différent de celui du serveur.

path

Le chemin sur le serveur sur lequel le cookie sera disponible. Si la valeur est '/', le cookie sera disponible sur l'ensemble du domaine *domain* . Si la valeur est '/foo/', le cookie sera uniquement disponible dans le répertoire /foo/ ainsi que tous ces sous-répertoires comme /foo/bar/ du domaine *domain* . La valeur par défaut est le répertoire courant où le cookie a été défini.

domain

Le domaine où le cookie est disponible. Pour rendre le cookie disponible sur tous les sous-domaines de MonSite.com, vous devez mettre la valeur '*MonSite.com*'. Le point (.) n'est pas requis mais est nécessaire pour la compatibilité avec encore plus de navigateurs.

secure

Indique si le cookie doit uniquement être transmis à travers une connexion sécurisée HTTPS depuis le client. Lorsqu'il est positionné à **TRUE**, le cookie ne sera positionné uniquement si la connexion sécurisée existe. La valeur par défaut est **FALSE**. Côté serveur, c'est au développeur d'envoyer ce genre de cookie uniquement sur les connexions sécurisées (e.g. en utilisant la variable `$_SERVER["HTTPS"]`).

httponly

Lorsque ce paramètre vaut **TRUE**, le cookie ne sera accessible que par le protocole HTTP. Cela signifie que le cookie ne sera pas accessible via des langages de scripts, comme Javascript. Cette configuration permet de limiter les attaques via XSS (bien qu'elle ne soit pas supportée par tous les navigateurs). Ajouté en PHP 5.2.0. **TRUE** ou **FALSE**

Valeurs de retour

Si quelque chose a été envoyé avant l'appel à cette fonction, **setcookie()** échouera et retournera **FALSE**. Si **setcookie()** réussit, elle retournera **TRUE**. Cela n'indique pas si le client accepte ou pas le cookie.