Cours PHP, By TR@PCOD13N

Site Web : http://trapcodien.free.fr
E-Mail : trapcodien@hotmail.fr

Bonjour à tous, aujourd'hui j'ai décidé de commencer à rédiger un tutoriel sur le langage web PHP.

- Dans cette première partie du tutoriel qui s'intitule '**à la découverte de php**', nous allons découvrir qu'est ce que php, pourquoi l'utiliser, savoir comment ça marche et comment le faire fonctionner sur votre machine.

- La partie II s'intitule '**Codons un peu**' et est en cours de rédaction...

Pré-requis :
- Des bases de html/xhtml (il n'est pas obligatoire de connaitre parfaitement le html, on est pas là pour faire du design)

I - à la découverte de php

1- C'est quoi PHP ? o_Ô

PHP est un langage web **dynamique** très puissant.
Retenez bien ce mot : **dynamique**

2- PHP, c'est bien ?

PHP est un langage merveilleux, après y avoir goûter vous ne pourrez plus vous en passer.
Pourquoi est-il si bien ?
On a dit tout à l'heure que php est un langage **dynamique**.
Pourquoi dynamique ? Je vais tenter de vous l'expliquer.

3- ça veut dire quoi dynamique ?

Contrairement à ce que certains débutants peuvent penser en entendant le mot '**dynamique**', ce langage ne permet pas de créer des animations.
Je vais vous donner un exemple et vous comprendrez vite ce que veut dire 'dynamique'.

Premièrement, nous allons voir qu'est ce qu'un site statique.
Je vous donne un exemple : http://simpp-kode.tuxfamily.org
Ce site n'est composé que de page HTML.
Le contenu ne changera jamais (à moins que le webmaster modifie les fichiers.html)

Voici maintenant un site dynamique : http://cryptor.net
Par exemple lorsque vous allez sur un topic, vous voyez les messages.
Et bien cette page ne restera jamais comme ça, elle changera dés qu'un membre postera un message (bon si personne poste de messages le site restera dynamique quand même on est bien d'accord)
Le fait que le message que vous postez soit enregistrez quelque part et s'affiche par la suite, c'est PHP qui le gère.
Bon PHP il peut pas tout faire tout seul non plus il lui faudra l'aide d'un serveur **mysql**.

Mysql : vous avez sûrement dû en entendre parler, **mysql** est une **base de donnée** (on dit **bdd** pour faire plus court), en anglais on dit **database** (on dit **db** pour faire plus court).

Base de donnée : pour faire simple, il s'agit d'un endroit où l'on peut stocker toutes sortes d'informations, et c'est bien rangé croyez moi ^^
On reparlera plus tard de **mysql**.

[4- Mais comment ça marche php ?

Je vais essayer de vous expliquer la notion de **client/serveur**
Le php est exécuté du **côté serveur**, tandis que le html est exécuté du **côté client**.

Côté client : Il s'agit du **navigateur**, c'est lui qui reçoit les données que le **serveur** lui envoi, les données envoyées sont au format html.
Le **navigateur** ne peut pas exécuter le code php, c'est le **serveur** qui exécute le code php et le met au format html.

Côté serveur : C'est sur le **serveur** que les scripts php sont stockés, c'est ce **serveur** qui va exécuter le code php, il va le retranscrire en html pour l'envoyer au **client**.

Exemple :

```
//Imaginons qu'il s'agit d'un code php.
Code qui vérifie si l'utilisateur est connecté.
Si l'utilisateur est connecté alors on affiche : 'bonjour *nom d'utilisateur*'
si il n'est pas connecté, on affiche : 'bonjour'
```

Donc je vais sur cette page avec mon **navigateur**, on va prendre le cas où on est pas connecté, le **client** fait donc une requête au **serveur** pour que celui-ci envois la page html.
Le **serveur** exécute le code php, le script va donc vérifier si on est connecté, comme ce n'est pas le cas il nous renvois juste : **bonjour**.
Le navigateur va seulement recevoir le texte '**bonjour**'.

Si l'utilisateur aurait été connecté, le script php aurait réagit différemment et il aurait envoyer au **client** : **bonjour trap** (dans le cas où le nom d'utilisateur est **trap**)

Bon je vais m'arrêter là avec la notion **client/serveur**.

Nous allons donc passer à la suite, l'installation d'un **serveur apache**.

5- Un serveur apache ? C'est quoi ce truc ?

C'est le nom du serveur qui **exécute le code PHP**.
Nous allons installer un logiciel qui inclut un serveur **Apache** ET un serveur **mysql** (nous aurons besoin de **mysql** plus tard)

Le serveur **mysql** : c'est lui qui va **gérer les données enregistrées** (comme le nom des utilisateurs, leur mot de passe, les messages qu'ils auront poster, le nombre de visite sur votre site web, etc, ...)

Bon nous allons passer à l'installation du logiciel.
Je vous est préparé **2 petites vidéos**, la première présente l'installation sur **Windows** et la deuxième l'installation sur **Linux**.

Sur **Windows** nous allons utiliser **EasyPHP,**
Sur **Linux**, nous allons utiliser **LAMPP (XAMPP FOR LINUX)**
Renseignez vous sur ces **plateformes de développement web** si ça vous chante.
Sachez qu'il en existe d'autres, mais je ne vais pas faire un tuto par plateforme.

Installation de EasyPHP (Windows XP) →
http://trapcodien.free.fr/content/videos/INSTALLATION-EasyPHP_win32.swf

Installation de LAMPP (Debian) → http://trapcodien.free.fr/content/videos/INSTALLATION-lampp_debian.swf

II - Codons un peu

1- Les balises PHP

Si vous avez vu l'une des 2 vidéos qui faisait parties de la première partie, vous avez sûrement remarqué que j'ai parler des balises PHP, je considère que vous sachez déjà qu'est ce qu'une balise.

La balise d'ouverture du code php est : <?php

La balise de fermeture du code php est : ?>

Ce qui nous donne ceci :

```
1  <?php
2      // Du code PHP
3  ?>
```

On peut trouver la balise d'ouverture sous cette forme : <?
Mais il est déconseillé de l'utiliser car certains serveurs apache sont configurés pour ne pas

interpréter cette balise, on ne va donc pas l'utiliser au cours de ce tutorial.
J'ai décidé quand même de vous en parler au cas où vous tomberez sur un code source qui emploie cette balise, au moins, vous ne serez pas déboussolé.

2- Afficher du texte avec echo

Je vais vous présenter une instruction qu'on appelle **echo**.

Elle permet d'**afficher du texte**.

Je ne vais pas tourner autour du pot et je vais vous donner le code tout de suite.

```php
<?php
   echo "La fonction echo permet d'afficher du texte !";
?>
```

Bon je pense que vous cela nécessite plus d'informations.

Tout d'abord on doit appeler l'instruction **echo** en l'écrivant tout simplement.

Ensuite nous devons écrire du texte ce texte doit être écrit entre-guillemet et doit être séparés de **echo** par un espace.

N'oubliez surtout pas le **point-virgule** (**;**) à la fin de chaque instructions sinon **apache** vous retournera une vilaine erreur du genre : **Parse Error.**

Enfin ! Vous venez d'écrire votre première ligne de code en **php**.

ça mérite un bien un p'tit test.

écrivez ce code dans un fichier en **.php** et mettez ce fichier dans votre dossier **www** (si vous utilisez **EasyPHP**) ou le dossier **htocs** (si vous utilisez **XAMPP**).

Testez en allant sur la page : **http://127.0.0.1/nomDeVotrePage.php**

Vous vous êtes sûrement dit que ça ne servait à rien et que ça revenait au même d'écrire ce texte dans un fichier**.html** mais vous comprendrez sûrement l'utilité de **echo** plus tard.

Si vous programmez ne serait-ce qu'un minimum avec un autre langage vous devez sûrement deviner à quoi cette instruction va nous servir plus tard.

Sachez aussi que vous pouvez mettre votre texte entre 2 **apostrophes** (').

Mais attention, si vous écrivez un code de ce genre.

```php
<?php
   echo 'L'aventure continue sur trapcodien.free.fr';
?>
```

Je vous ai dit tout à l'heure qu'il fallait placer son texte entre **apostrophes**.

Si vous regardez bien, ce n'est même pas du texte qui est entre **apostrophes** mais une seul lettre : la lettre **L.**

Effectivement, la lettre **L** de "**L'aventure**" est entre **apostrophes**, le reste du texte est derrière et n'est pas entre **apostrophes**.

Le serveur **apache** va donc générer une erreur de syntaxe.

ça vaut la même chose pour les **guillemets** :

```php
<?php
    echo "L'instruction "echo" permet d'afficher du texte";
?>
```

Vous vous posez sûrement une question du genre : mais comment on fait alors ? c'est galère moi j'arrête le **PHP**.

Surtout ne faites pas de bêtises et **lisez la suite :**

3- Le caractère d'échappement

Le caractère d'échappement est la solution à notre problème.

Il ressemble à ça : \

Il s'agit effectivement de l'**anti-slash**, aussi appelé **backslash**.

Pour échapper un caractère dans une chaîne il suffit de le faire précéder d'un **anti-slash.**

Voyons tout de suite un exemple.

```php
<?php
    echo 'L\'aventure continue sur trapcodien.free.fr';
?>
```

Voilà, c'est pas plus compliqué que ça, on échappe le caractère à l'aide de **\ (** ceux qui ont fait de C n'ont pas dû se sentir perdu **)**

On peut aussi échapper le guillemet.

```php
<?php
    echo "L'instruction \"echo\" permet d'afficher du texte";
?>
```

Notez qu'ici, on utilise un apostrophe sans avoir besoin de l'échapper, c'est normal puisque notre texte est entre guillemets.

Certains d'entre vous ont dû se poser tout de même une question : Comment on fait pour afficher un **anti-slash** dans son texte, puisque ce caractère est un caractère spéciale ?

Et bien il suffit tout simplement d'échapper le **backslash** avec un backslash, on va donc le doublé.

Exemple :

```php
<?php
    echo "Ici, j'ai envie d'afficher un \\ !";
?>
```

N'hésitez pas à essayer vos scripts avec votre serveur apache, c'est avec la pratique qu'on apprends.

4- Les commentaires

Nous allons maintenant parler des **commentaires**.
Comme son nom l'indique, ceux-ci sont utilisés pour **commenter** votre script.
Les commentaires sont du textes écrit et visible lorsque vous lisez votre source.
Ils ne sont pas exécuté par apache, c'est juste du texte pour vous y retrouver dans votre code.

Ils sont utilisés pour :

- Vous aider à cous retrouver dans votre code lorsque celui-ci est particulièrement long
- Donner des indications à la personne qui lire votre script

Bon je vais maintenant vous expliquer comment insérer des commentaires dans votre source.

1. Les commentaires sur une ligne. (//)

```php
<?php
    //Ceci est un commentaire, cette ligne ne sera pas exécutée par apache.
    echo "ça c'est du texte !"; // ça aussi c'est un commentaire.
?>
```

Vous pouvez les placés à la fin d'une instruction PHP, où sur une ligne vierge. Exemple :
Pour placer un commentaire il suffit de placer le texte que vous voulez précédé de 2 Slashs (/)

2. Les commentaires multi-lignes (/* */)

Un petit exemple qui va vous aider à comprendre :

```php
<?php
    /* Ceci est la balise ouvrante des commentaires multi-lignes
    Je peux continuer à rédiger mon commentaire
    Là aussi
    etc...
    ..............
    En dessous il s'agit de la balise fermante des commentaires multi-lignes
    */
    echo "là j'affiche du texte";
    /* Je ré-ouvre une balise de commentaire multi-lignes
    Il ne faut surtout pas écrire après la fermeture de la balise multi-ligne
    */
?>
```

5- Les variables

Nous arrivons à un chapitre clef de ce cours PHP, nous allons parler des variables qui est une chose qu'on utilise très fréquemment en php (Je dirais même tout le temps)

Pour expliquer avec des mots simples :

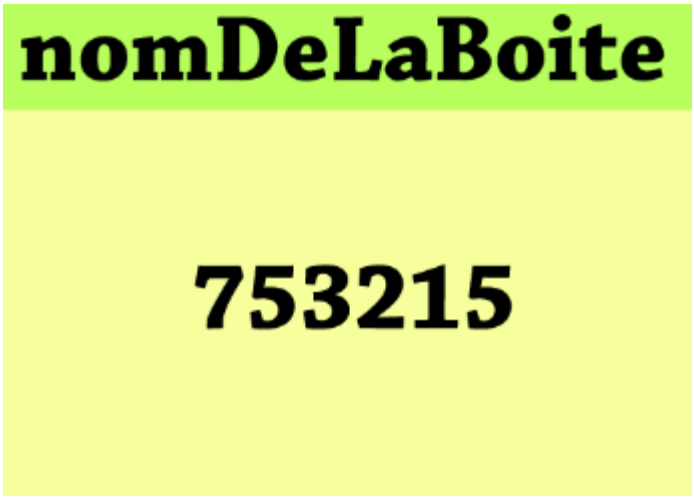
Nous allons comparer une variable avec une boîte qui possède une étiquette.

Cette étiquette est le nom de la boite.

La boite peut contenir un nombre ou du texte.

Donc si vous avez bien suivit : Une variable possède un nom et peut contenir un nombre ou du texte. (elle ne peut pas contenir plusieurs nombres, il faudrait utiliser les tableaux qui est un ensemble de boite, nous verrons ça plus tard)

Je suis gentils, je vous ai fait un petit schéma :



Sachez que le nom d'une variable ne doit pas contenir d'espace ou de caractère spéciaux.

Vous avez le droit à toutes les lettres de l'alphabet, tous les chiffres, le caractère underscore (_) ou le tiret (-)

Si j'en ai oublier, veuillez me contacter (voir en haut du PDF)

Voilà donc là nous avons une variable avec comme nom 'nomDeLaBoite' qui contient le nombre '753215'

Voyons voir maintenant ce que ça donne en code.

```php
<?php
    $nomDeLaBoite = 753215; // Pas de guillemet pour un nombre
    $nomDeLaSecondeBoite = "Bonjour"; // Ici, c'est pareil que l'instruction echo, on peut utiliser les guillemets,
                                      // les apostrophes, le caractère d'échappement, etc.
?>
```

Pour **déclarer** une variable en PHP, on lui donne donc un nom en prenant soin que celui-ci soit précédé du symbole dollars (**$**)

Après l'avoir nommé, on va lui attribuée une valeur avec =

Ensuite lisez les commentaires c'est facile à comprendre.

Pensez à donner des noms explicites à vos variables.

Pour afficher le contenu d'une variable, il suffit d'utiliser la commande echo.

```php
<?php
    $nomDeLaBoite = 753215;           // Pas de guillemet pour un nombre
    $nomDeLaSecondeBoite = "Bonjour"; // Ici, c'est pareil que l'instruction echo, on peut utiliser les guillemets,
                                      // les apostrophes, le caractère d'échappement, etc.
    echo $nomDeLaSecondeBoite;        // Pas besoin de guillemet ou d'apostrophe pour afficher le contenu de la variable
?>
```

6- La concaténation

Ne fuyez pas à la vue de ce mot qui à l'air compliquer alors que ce n'est pas si compliquer que ça.
Voyons voir ce que raconte wikipedia à propos de la concaténation :

Concaténation

Cet article est une ébauche concernant l'informatique et les mathématiques.
Vous pouvez partager vos connaissances en l'améliorant (comment ?) selon les recommandations des projets correspondants.

Le terme **concaténation** (substantif féminin), du latin *cum* ("avec") et *catena* ("chaîne, liaison"), désigne l'action de mettre bout à bout au moins deux chaînes.

Pour bien comprendre le concept, je vais vous donner un petit exercice.

Imaginons qu'on a der variables définit comme cela :

```php
<?php
    $bonjour_partie1 = "bon";
    $bonjour_partie2 = "jour";
?>
```

Le but est d'afficher "bonjour" en utilisant les 2 variables et en utilisant une seul fois l'instruction echo. Ne lisez pas plus loin si vous voulez trouvez tout seul.

Vous avez dû sûrement essayer pleins de trucs dans ce genre :

```php
<?php
    $bonjour_partie1 = "bon";
    $bonjour_partie2 = "jour";
    echo $bonjour_partie1$bonjour_partie2;
    // Ou encore
    echo $bonjour_partie1 $bonjour_partie2;
?>
```

La solution est simple, il suffit de séparer le nom des variables par un point (.)
Les espaces ne rentrent pas en compte.

Voici la solution :

```php
<?php
    $bonjour_partie1 = "bon";
    $bonjour_partie2 = "jour";
    echo $bonjour_partie1.$bonjour_partie2;
    // Ou encore
    echo $bonjour_partie1 . $bonjour_partie2;
?>
```

Avant de finir ce chapitre sur la concaténation, on va refaire un petit exercice.

```php
<?php
    $bonjour_partie1 = "bon";
?>
```

Nous disposons maintenant que de la première partie de 'bonjour'
Le but ici est encore d'afficher 'bonjour' avec UNE seul instruction **echo**.
N'oubliez pas le point qui va séparer la chaine de caractère et la variable.

Voici la solution :

```php
<?php
    $bonjour_partie1 = "bon";
    echo $bonjour_partie1 . "jour";
?>
```

Pour l'instant vous ne voyez peut être pas l'utilité d'une variable mais vous allez bientôt savoir à quoi ça sert réellement.

Un dernier petit script pour la route :

```php
<?php
    $pseudo = "trapcodien";
    echo "Bienvenue " . $pseudo . " !";
?>
```

Passons au dernier chapitre de la partie II qui sera très court.

7- Les booléens

Et oui, encore du vocabulaire.
Un booléen est une variable particulière, vous verrez à quoi ça nous sert lorsque nous verrons les conditions. (partie III)

Article wikipedia :

Booléen

Un **booléen** en logique et en programmation informatique est un type de variable à deux états. Les variables de ce type sont ainsi soit à l'état *vrai* soit à l'état *faux* (en anglais *true* et *false*).

Généralement les *conditions* sont de type booléen, car elles nécessitent une réponse binaire du type *oui* ou *non*.

bouton enfoncé : une condition booléenne
lumière allumée : une variable booléenne

Si *bouton enfoncé*
alors *lumière allumée* = vrai
sinon *lumière allumée* = faux

Certains langages utilisent le bit pour représenter des booléens : ainsi un *0* représentera la valeur *faux* et un *1* représentera la valeur *vrai* (ou l'inverse, selon les conventions). D'autres langages préfèreront utiliser l'anglais TRUE (vrai) et FALSE (faux) traditionnellement en majuscules.

Voici un code qui résume l'article :

```php
<?php
    $vrai = true;
    $faux = false;
    // Il est inutile d'afficher la valeur du booléen
    // Vous pouvez tout de même essayer, ça vous entrainera.
?>
```

Nous voici donc à la fin de la partie II.
Cela a vous a peut être parut un peu lourd, mais si vous n'avez fait que survolez cette partie, vous verrez que vous y reviendrais car c'est vraiment des bases très importants à avoir pour la suite.

Dans la partie III, On va faire des choses beaucoup plus amusante ^^.

III - Un code plus dynamique

...EN COURS DE REDACTION...