Développement web S1
– 1. Introduction –

Christophe BLANC

– IUT MMI –
IUT d'Allier
Université Clermont Auvergne
www.christophe-blanc.fr

2021-2022

Notion d'algorithme

- ✔ Un **algorithme** est une **suite d'instructions** qui une fois exécuté correctement conduit à la **résolution d'un problème** en un nombre fini d'étapes.
- ✔ A partir de **données**, les **entrées** de l'algorithme, on va donc parvenir à un **résultat**, la **sortie**.
- ✔ Un algorithme doit contenir uniquement des **instructions compréhensibles** par celui qui devra l'exécuter

Notion d'algorithme

Exemple : recette de cuisine

- ✔ Problème : comment faire de la tarte à la bière ?
- ✔ Résolution :
  - ✔ Etaler de la pate brisée dans un plat
  - ✔ Verser 2l de bière dans la pâte
  - ✔ Passer 1 min au micro-onde
  - ✔ Déguster

La maîtrise de l'algorithmique requiert deux qualités :

- ✔ intuition
- ✔ méthodique et rigoureux : il faut systématiquement se mettre mentalement à la place de la machine qui va exécuter la série d'instructions que vous avez écrite.

Notion d'algorithme

Origine du mot algorithme

- ✔ Le mot algorithme vient du nom du mathématicien perse Abu Abdullah Muhammad ibn Musa al-Khwarizmi (9 ème siècle après Jésus Christ).
- ✔ Son nom a ensuite été latinisé au Moyen Age en "algoritmi" .
- ✔ Origine du mot algorithme ne veut pas dire origine du principe. On écrivait des algorithmes dès l'antiquité comme nous allons le voir sur des exemples.

Notion d'algorithme

Algorithme de calculs d'intérêts (1800 avant J.-C.)

- ✔ Problème : trouver en combien d'années on double un montant soumis à un certain taux annuel.
- ✔ Résolution :
  1. Initialiser le taux t à la valeur voulue, a à 0 et b à 1
  2. Tant que $b < 2$, remplacer b par $b * (1 + t)$ et ajouter 1 à a.
  3. La valeur finale de a est le nombre d'années cherché.

a	b
0	1
1	1.2
2	1.44
3	1.728
4	2.0736
il faut 4 ans	

Notion d'algorithme

Algorithme de multiplication égyptienne (1650 avant J.-C.)

- ✔ Problème : multiplier deux entiers a et b.
- ✔ Résolution :
  1. Décomposer a comme somme de puissances de 2.
  2. Calculer les puissances de 2, 4, 8, etc. de b.
  3. Additionner les puissances de b qui correspondent aux puissances de 2 trouvées dans a.

2^n	$a = 25$	$b = 14$
1	1	14
2	0	28
4	0	56
8	1	112
16	1	224
$a * b = 14 + 112 + 224 = 350$		

Algorithme d'Euclide (300 avant JC)

- ✔ Problème : trouver le PGCD de deux entiers a et b.
- ✔ Résolution :
  1. Si b est non nul, diviser a par b.On note r le reste de cette division euclidienne.
  2. Remplacer a par b et b par r.
  3. Recommencer tant que cela est possible à partir de l'étape 1.
  4. Le PGCD est alors la dernière valeur non nulle de r.

a	b	r
142	38	28
38	28	10
28	10	8
10	8	2
8	2	0
PGCD(142,38)=2		

Notion d'algorithme

- ✔ En informatique, les instructions d'un algorithme seront bien sûr à destination d'un ordinateur.
- ✔ Pour qu'elles soient compréhensibles par ce dernier, elles doivent évidemment être écrites dans ce que l'on appelle un langage de programmation.
- ✔ Celui-ci sera lui même traduit par un compilateur en langage machine ou exécuté par un interpréteur.

Notion d'algorithme

- ✔ Avant l'étape d'écriture du programme, on peut/doit concevoir l'algorithme en lui même de façon non formelle, en version "papier".
  Problème $\rightarrow$ Algorithme $\rightarrow$ Programme
- ✔ L'algorithme contient la réflexion, l'abstraction du programme.
- ✔ Ce dernier pouvant être complexifié par des contraintes techniques.

Notion d'algorithme

Les ordinateurs ne sont fondamentalement capables de comprendre que quatre catégories d'ordres (en programmation, on n'emploiera pas le terme d'ordre, mais plutôt celui d'instructions). Ces quatre familles d'instructions sont :

- ✔ l'affectation de variables
- ✔ la lecture / écriture
- ✔ les tests
- ✔ les boucles

Un algorithme informatique se ramène donc toujours au bout du compte à la combinaison de ces quatre petites briques de base. Il peut y en avoir quelques unes, quelques dizaines, et jusqu'à plusieurs centaines de milliers dans certains programmes de gestion. Cependant, la taille d'un algorithme ne conditionne pas en soi sa complexité : de longs algorithmes peuvent être finalement assez simples, et de petits très compliqués.

Notion d'algorithme

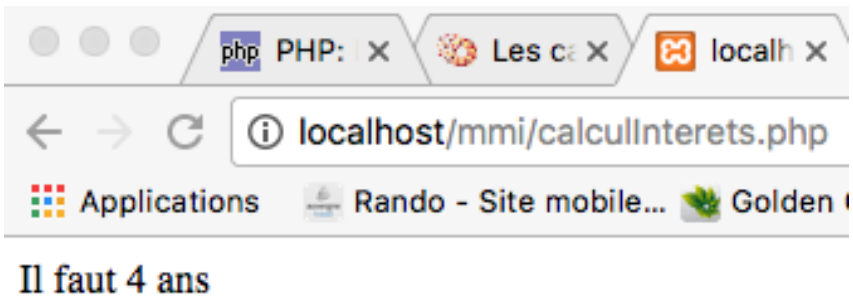
- ✔ Nous n'adopterons pas dans ce cours de syntaxe formalisée des algorithmes, chacun les rédigera pour lui-même dans l'unique but de structurer ses idées.
- ✔ On peut même tout à fait imaginer faire des schémas en guise d'algorithme.

Notion d'algorithme

Exemple : l'algorithme de calculs d'intérêts deviendra le programme suivant en PHP

```php
<?php
$t=0.2;
$a=0;
$b=1;

while ($b < 2)
{
        $b=$b*(1+$t);
        $a=$a+1;
}
echo "Il␣faut␣".$a."␣ans";
?>
```



Bref historique

- ✔ PHP : Personal Home Page
- ✔ Conçu en 1994 par Rasmus Lerdorf pour ces besoins personnels
- ✔ Première publication du code en 1995
- ✔ PHP est utilisé par plus de 80% des sites
- ✔ Actuellement : version 7.3 (décembre 2018)

Qu'est ce que le PHP ?

- ✔ PHP est un langage de script (multi plateforme, libre, gratuit) qui s'exécute côté serveur
- ✔ Le résultat de cette exécution est intégré dans une page HTML qui est envoyée au navigateur. Ce dernier n'a aucune connaissance de l'existence du traitement qui s'est déroulé sur le serveur.
- ✔ Cette technique permet de réaliser des pages Web dynamiques dont le contenu peut être complétement ou partiellement généré au moment de l'appel de la page, grâce à des informations récupérées dans un formulaire ou extraites d'une base de données.

Fonctionnement

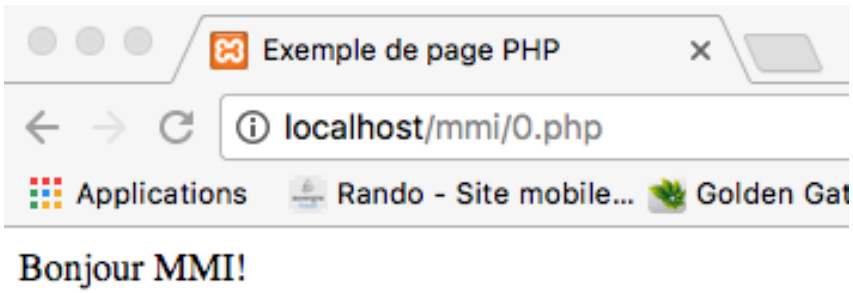
- ✔ Lorsqu'un visiteur demande à consulter une page de site internet, son navigateur envoie une requête au serveur HTTP correspondant.
- ✔ Si la page est identifiée comme un script PHP (généralement grâce à l'extension .php), le serveur appelle l'interprète PHP qui va traiter et générer le code final de la page (constitué généralement d'HTML ou de XHTML, mais aussi souvent de feuilles de style en cascade et de JS).
- ✔ Ce contenu est renvoyé au serveur HTTP, qui l'envoie finalement au client.

Outils : plateformes

- ✔ Plateforme : système d'exploitation + serveur HTPP + langage serveur + SGBD.
- ✔ Pour PHP : WAMP (Windows Apache MySQL PHP), MAMP (Mac OS Apache MySQL PHP), LAMP (Linux Apache MySQL PHP)
- ✔ Une plateforme WAMP s'installe généralement par le biais d'un seul logiciel qui intègre Apache, MySQL, PHP : par exemple EasyPHP

Exemple : Hello World !

```
<HTML>
<HEAD>
<TITLE>Exemple de page PHP</TITLE>
</HEAD>
<BODY>
<?php
echo "Bonjour␣MMI!";
?>
</BODY>
</HTML>
```



TD1 : à tester

Le code PHP est inclus dans une page HTML (extension .php) à l'intérieur de balises (aussi appelées tags). Les instructions PHP se situent entre les balises :

- ✔ `<?php ...?>`
- ✔ `<?...?>`
- ✔ `<script language="php"> ... </script>`

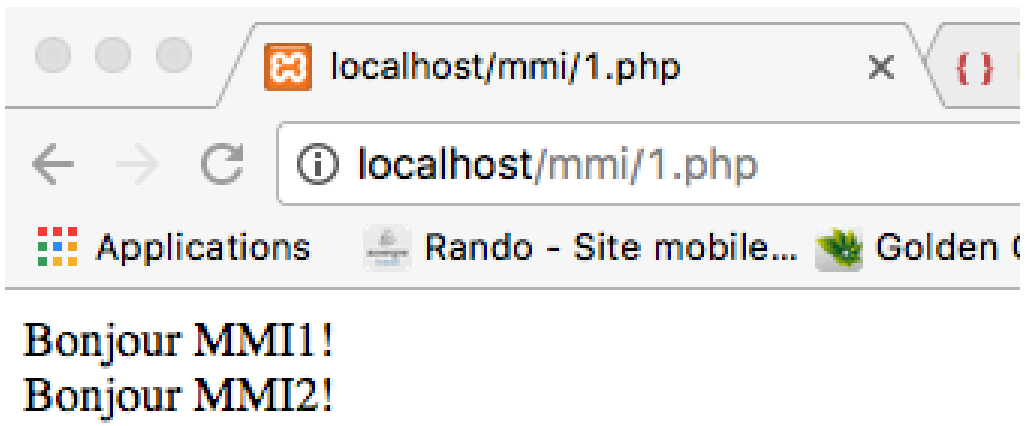
  : obsolète
- ✔ `<% ... %>`

  : obsolète

Structure de base d'une page PHP

Les balises (tags) PHP

```
<HTML>
<?php
echo "Bonjour␣MMI1!";
?>
</BR>
<?php
echo "Bonjour␣MMI2!";
?>
</HTML>
```



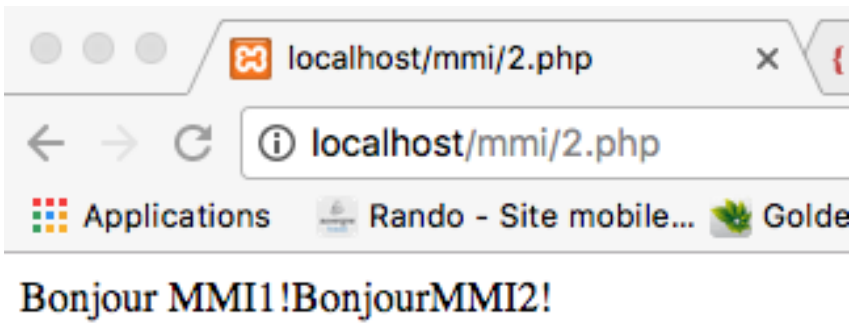
TD1 : à tester

Elle permet d'afficher une ou plusieurs chaînes de caractères et donc d'inclure du texte dans la page HTML envoyée au navigateur.
Syntaxe :

```
echo (chaine1);
echo chaine1,...,chainen;
```

La première syntaxe n'accepte qu'un paramètre alors que la deuxième en accepte plusieurs.

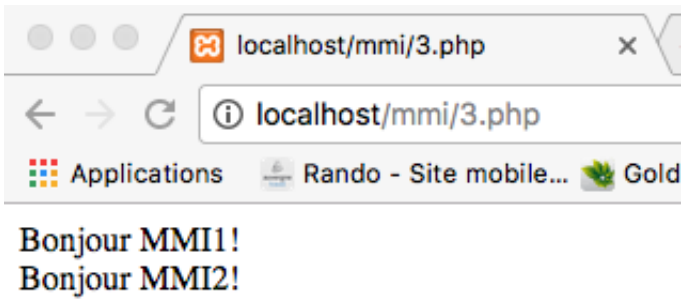
```
<HTML>
<?php
echo "Bonjour␣MMI1!";
echo "Bonjour", "MMI2!";
?>
</HTML>
```



Il n'y a pas de saut de ligne automatique. En cas de besoin, il est donc nécessaire d'insérer la balise HTML

qui provoque un saut de ligne.

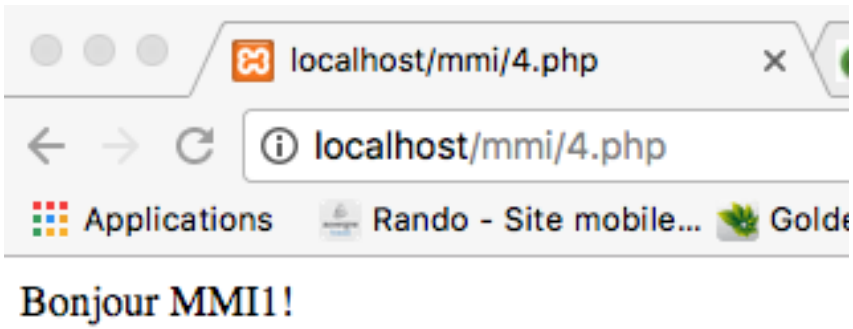
```
<HTML>
<?php
echo "Bonjour␣MMI1!";
?>
</BR>
<?php
echo "Bonjour␣", "MMI2!";
?>
</HTML>
```



TD1 : à tester

Le texte passé en paramètre à la fonction echo peut être écrit sur plusieurs lignes dans le source mais il est affiché sur une seule dans le résultat :

```
<HTML>
<?php
echo "Bonjour
MMI1!";
?>
</HTML>
```

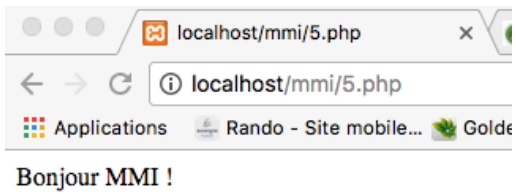


Structure de base d'une page PHP

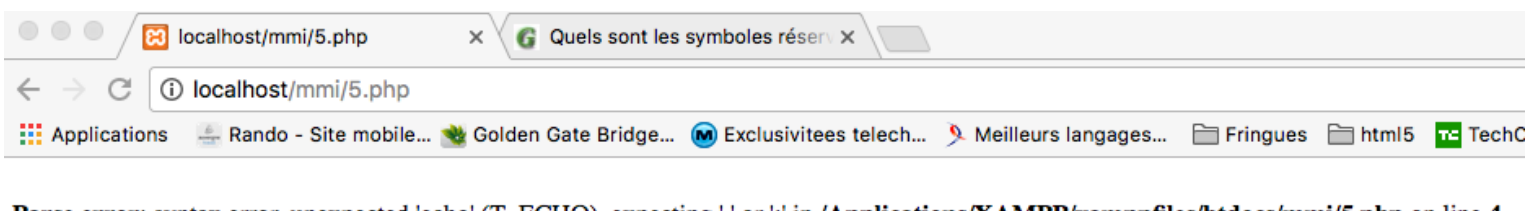
Le séparateur d'instructions

En PHP, toutes les instructions doivent se terminer par un point-virgule.

```
<HTML>
<?php
echo "Bonjour␣";
echo "MMI␣!";
?>
</HTML>
```




```
<HTML>
<?php
echo "Bonjour␣"
echo "MMI␣!";
?>
</HTML>
```



- ✔ Commentaire sur une seule ligne :

```
<HTML>
<?php
// Commentaire sur une seule ligne
?>
</HTML>
```

- ✔ Commentaire sur plusieurs lignes :

```
<HTML>
<?php
/* Commentaire sur
plusieurs lignes*/
?>
</HTML>
```

- ✔ Non visible par l'utilisateur (au contraire des commentaires html)

Structure de base d'une page PHP

Les commentaires

```
<html>
        <head>
                <title>Ma page</title>
        </head>
        <body>
                <!-- Commentaire HTML-->
                <?php
                // Ceci est un commentaire
                /* Cela aussi! */
                ?>
        </body>
</html>
```

Résultat retourné au client :

```
<html>
        <head>
                <title>Ma page</title>
        </head>
        <body>
                <!-- Commentaire HTML-->
        </body>
</html>
```

TD1 : retrouver ce résultat

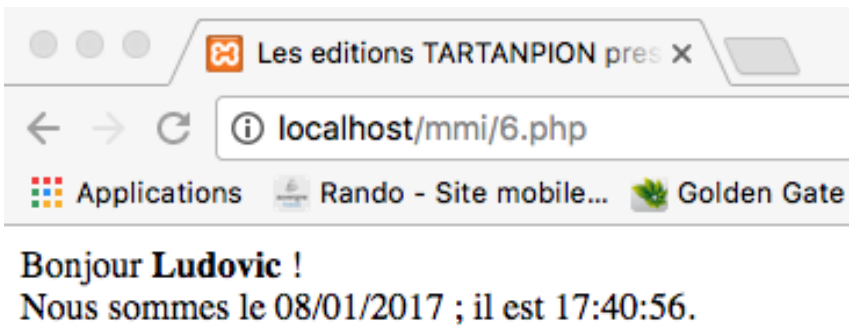
Les différentes approches pour mixer du PHP et de l'HTML reposent sur deux principes très simples :

- ✔ La page peut contenir une ou plusieurs inclusion(s) de code PHP.
- ✔ Le code PHP génère du "texte" qui est intégré dans la page HTML envoyée au navigateur. Tout "texte" compréhensible par le navigateur peut donc être généré par le code PHP : du texte simple, du code HTML, du code JavaScript...

Les exemples qui suivent utilisent des variables et des fonctions PHP (récupération de la date et l'heure). Ces notions sont présentées plus en détail dans le prochain chapitre.

Structure de base d'une page PHP
Mixer du PHP et de l'HTML

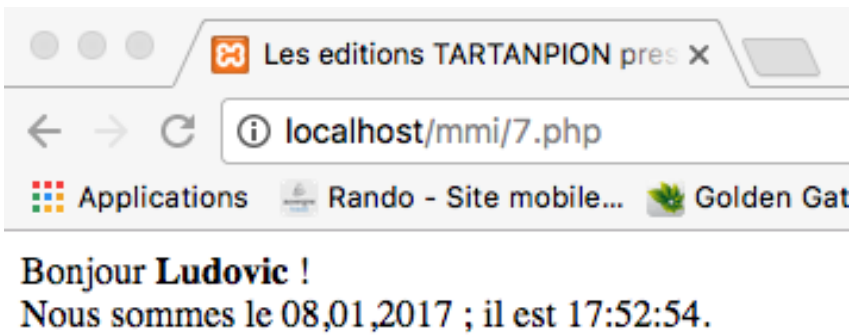
```
<?php
//declaration de variables utilisees plus loin
//cette section ne genere pas de sortie dans la page HTML
$nom="Ludovic";
$titre = "Les␣editions␣TARTANPION␣presentent...";
$aujourdhui = date("d/m/Y");
$heure = date("H:i:s");
?>
<html>
        <head>
                <title><?php echo $titre;?></title>
        </head>
        <body>
                <?php
                        echo "Bonjour␣<B>$nom</B>␣!␣</BR>";
                        echo "Nous␣sommes␣le␣$aujourdhui␣;␣il␣est␣$heure.";
                ?>
        </body>
</html>
```



TD1 : analyser le source de la page dans le navigateur

Structure de base d'une page PHP
Mixer du PHP et de l'HTML

```php
<?php
//declaration de variables utilisees plus loin
//cette section ne genere pas de sortie dans la page HTML
$nom="Ludovic";
$titre = "Les editions TARTANPION presentent...";
$aujourdhui = date("d,m,Y");
$heure = date("H:i:s");
echo "<html>";
echo "<head>";
echo "<title>$titre</title>";
echo "</head>";
echo "<body>";
echo "Bonjour <B>$nom</B> ! </BR>";
echo "Nous sommes le $aujourdhui ; il est $heure.";
echo "</body>";
echo "</html>";
?>
```



TD1 : analyser le source de la page dans le navigateur

PHP propose deux fonctions particulièrement utiles pour obtenir des informations sur la configuration : phpversion et phpinfo.
Exemple 1 :

```
<?php
echo phpversion();
?>
```

Exemple 2 :

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

TD1 : à tester

Règle de nommage

Toute entité PHP nommée (variable, constante, fonction,...) doit avoir un nom qui respecte les règles suivantes :

- ✔ commencer par une lettre ou un souligné
- ✔ suivi de lettres, chiffres ou soulignés

Ici, une lettre peut être a-z, A-Z, et les octets de 127 à 255 (code ascii). Les caractères accentués sont donc autorisés, mais pas les caractères du type #$%& qui ont une signification spéciale dans le langage PHP. $*this* est une variable spéciale qui ne peut pas être assignée.