M1202 - Algorithmique

Vos premiers pas en PHP

David Annebicque

IUT de Troyes

@DavidAnnebicque

Sommaire

Sommaire

Introduction

Objectifs du module :

- Comprendre le fonctionnement d'un site internet dynamique
- Comprendre les échanges clients-serveurs
- Écrire ses premières pages dynamiques
- Comprendre les concepts de boucles, tests conditionnels et fonctions

Organisation

Organisation du module :

- 10 TD
- 6 TP
- 1 évaluation écrite
- 1 évaluation de TP

PHP ?

- Langage de programmation (contrairement au HTML, langage de description)
- Orienté web
- Exécuté par un serveur (contrairement à javascript sur un client)
- Créé en 1995 par Rasmus Lerdorf
- PHP : Personnel Hypertext PreProcessor
- Documentation officielle (et en français) http ://www.php.net
- Versions courantes 5.5, 5.6 et 7.0

PHP, pour faire quoi ?

PHP permet

- De générer (construire) des pages web sur un serveur avant de les envoyer à un client
- De traiter des formulaires web
- D'accéder aux bases de données
- De générer des documents HTML, PDF, Office, Zip, images, ...

Quels types de site ?

Les sites statiques

- Uniquement en HTML et CSS.
- Le contenu est fixe et non modifiable facilement.
- Toute modification implique de devoir modifier les fichiers html.

Les sites dynamiques

- Le résultat est toujours en HTML et CSS .
- Implique aussi du PHP et souvent du MySQL.
- Le contenu est dynamique car il peut changer sans l'intervention du webmaster.

Dans la pratique

La quasi-totalité des sites sont dynamiques

Comment fonctionne un site web

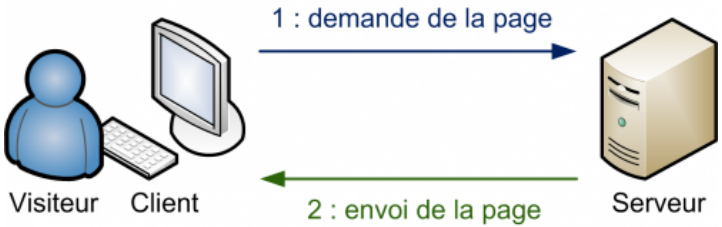
Le client

Les internautes, les visiteurs d'un site. Chaque client consulte un site sur un navigateur web

Le serveur

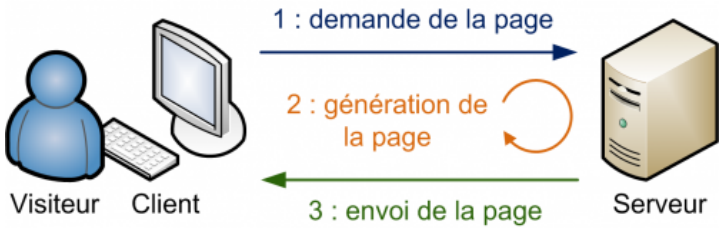
Des ordinateurs puissants qui stockent les sites web. Les serveurs ne sont pas visible de la majorité des clients

Fonctionnement d'un site statique



La page demandée par le client est automatiquement envoyée par le serveur car celle-ci ne peut pas changer

Fonctionnement d'un site dynamique



Un étape supplémentaire est nécessaire. En effet, les pages n'existent pas dans un état visible par les client. Elles doivent être construites en fonction des interactions des clients et des données disponibles (dans une base de données par exemple).
Deux clients différents peuvent donc ne jamais voir la même page (dans le cas d'une recherche par exemple)

Les pré-requis

Vous devez connaître les langages suivants

- HTML
- CSS
- Cela se fera en M1206 (intégration web)

Remarques pour le S1

- Le PHP se trouve TOUJOURS dans une page HTML
- Le PHP sert TOUJOURS à produire du HTML

Hello World

Listing 1 – Premier code PHP

```
1
2  <!DOCTYPE html>
3  <html lang="fr">
4    <head>
5          <title>Ma page avec du PHP</title>
6          <meta charset="utf-8" />
7    </head>
8    <body>
9      <?php
10         echo 'Hello World';
11     ?>
12   </body>
13 </html>
```

Comment faire fonctionner cet exemple

Les logiciels nécessaires

- Un éditeur de texte (notepad++, sublimeText) ou un IDE php (PhpStorm)
- Un serveur PHP (distant : un hebergeur, votre espace personnel), local (WampServer, MAMP, ...)
- Un client FTP (si votre serveur est distant, ex. FileZilla)
- Un navigateur (un client)

Attention à l'encodage !

Pour éviter tout problème vous devez avoir toute votre "chaîne" d'édition qui fonctionne en UTF-8 (sans BOM). Attention au format d'enregistrement de vos fichiers !

Comment faire fonctionner cet exemple

Étapes pour le test de l'exemple précédent

1. Créer un fichier "index.php" avec le code précédent
2. Connectez vous au serveur distant avec votre client FTP. L'adresse du serveur est 195.83.128.55 pour votre compte MMI.
3. Déposer le fichier "index.php" dans le répertoire "public_html" (vous pourrez créer des répertoires par la suite)
4. Ouvrez ce fichier en saisissant son adresse dans votre navigateur web `http://195.83.128.55/~dannebicque/index.php`

Explication du code

Quelques explications

- ligne 8 : <?php : ouverture de la balise php, nécessaire pour écrire du code php
- ligne 9 : echo 'Hello World' ; : L'instruction **echo** permet de dire à PHP d'écrire du code HTML.
- **le ; est obligatoire à chaque instruction PHP**
- ligne 10 : ?> : fermeture du bloc PHP

Quelques remarques

- On travaille **TOUJOURS** en chemin relatif (par rapport au fichier dans lequel on est !)
- On n'utilisera jamais d'espace ou de caractères spéciaux pour nommer les répertoires et les fichiers. Seul le _ est autorisé !
- Les noms des variables PHP respecterons les mêmes règles !
- Un fichier PHP à **TOUJOURS** une extension **.php**, sinon il ne fonctionne pas !

Exercice 1

- Tester votre premier fichier
- Modifier le fichier pour écrire votre nom en plus de "Hello World"
- Que se passe-t'il ?

Exercice 1

- Tester votre premier fichier
- Modifier le fichier pour écrire votre nom en plus de "Hello World"
- Que se passe-t'il ?

Correction

Le texte est collé "Hello WorldDavid". Il n'y a pas de retour chariot ! Il faudrait l'écrire en HTML. Pour cela on ajoute la balise HTML <**br**> dans un **echo**.

Exercice 2

- Corriger le code précédent
- Écrivez 'je m'appelle *xxx*', plutôt que simplement votre nom
- Que se passe-t'il ?

Exercice 2

- Corriger le code précédent
- Écrivez 'je m'appelle *xxx*', plutôt que simplement votre nom
- Que se passe-t'il ?

Correction

Vous avez un message d'erreur à cause des '. Les ' servent à informer PHP du texte à écrire. Pour pouvoir écrire un ' dans un **echo**, il faut utiliser le \et noter \'

Sommaire

Notion de séquence

Quand on écrit plusieurs instructions PHP, le moteur PHP les exécute une par une, en **séquence**, de gauche à droite, et de haut en bas.
L'ordre à donc de l'importance.

Listing 2 – Exemples de séquence

```
1 <?php
2 echo 'Bonjour';
3     echo 'David ';
4     echo 'Je suis enseignant';echo 'en DUT MMI';
5 ?>
```

L'instruction echo

echo permet d'écrire ce qui la suit lorsque le code est interprété (généralement du contenu HTML).
echo peut être suivi (ou composée) de variables, c'est la valeur de ces variables qui seront affichés.

Listing 3 – Exemple

```php
<?php
$variable = 12;
    echo $variable;
?>
```

L'instruction echo

echo peut être suivi d'une chaîne de caractères entre ' ou ".

Listing 4 – Exemple

```php
<?php
$variable = 12;
    echo $variable;
    echo "David";
?>
```

Attention

Nous n'utiliserons que la notation avec un '. Même si l'utilisation des " " peut sembler plus simple et facile, cela posera des problèmes pour générer du HTML valide !

L'instruction echo

echo peut être suivi de toute les combinaison de chaînes de caractères et de variables possible. Il faut dans ce cas les séparer par des points (.). Le . (point) est l'opérateur de **concaténation** ou de collage)

Listing 5 – Exemple

```
1 <?php
2 $moyenneUE1 = 12;
3 $moyenneUE2 = 19;
4 echo 'je␣suis␣David,␣j\'ai␣' . $moyenneUE1 . '␣de␣moyenne
5 en␣UE1␣et␣' . $moyenneUE2.'␣en␣UE2.';
6 ?>
```

Quelques exemples plus complexes

Tous les exemples suivants donnent le même résultat

Listing 6 – Exemple

```
1 <?php
2         echo 'Bonjour␣David';
3 ?>
```

Listing 7 – Exemple

```
1 <?php echo 'Bonjour␣David'; ?>
```

Listing 8 – Exemple

```
1 <?php echo 'Bonjour'; ?>
2 <?php echo '␣David'; ?>
```

Quelques exemples plus complexes

Vous devez prendre soin de gérer vos espaces entre les chaînes et variables, ainsi que les sauts de ligne !

Listing 9 – Exemple

```
1 <?php
2         echo 'Bonjour ' . 'David';
3 ?>
```

Listing 10 – Exemple

```
1 <?php echo 'Bonjour' . ' David'; ?>
```

Exercice

Voici un début de code PHP

Listing 11 – Exemple

```php
<?php
        $nom = 'Annebicque';
    $prenom = 'David';
    $age = 32;
?>
```

Écrire le code qui donne le résultat suivant :

Résultat

Bonjour, je suis David Annebicque, j'ai 32 ans.

Générer du HTML

Jusqu'à maintenant nous avons utiliser le PHP pour produire du texte (une fois les fichiers interprétés). C'est donc assez limité comme utilisation.
Le but essentiel du PHP est de générer des pages web (ou des PDF, des images, ...)
Comment faire selon-vous ?

Générer du HTML

Le HTML c'est des balises, et des balises c'est du texte !

Listing 12 – Ecrire du HTML

```
1 <?php
2         echo '<strong>Hello World !</strong>';
3 ?>
```

Que se passe-t'il ?

Générer du HTML

Selon votre navigateur le texte sera en gras (parce que votre navigateur ajoute les balises manquantes). Sinon il sera toujours sans mise en forme, car il manque toujours la structure HTML nécessaire pour l'affichage correct.

Listing 13 – Ecrire du HTML correct

```
1  <!DOCTYPE html>
2  <html lang="fr">
3    <head>
4          <title>Ma page avec du PHP</title>
5          <meta charset="utf-8" />
6    </head>
7    <body>
8      <?php
9          echo '<strong>Hello World</strong>';
10     ?>
11   </body>
12 </html>
```

Générer du HTML

On pourrait écrire le code suivant pour obtenir le même résultat

Listing 14 – Ecrire du HTML correct V2

```
1  <!DOCTYPE html>
2  <html lang="fr">
3    <head>
4          <title>Ma page avec du PHP</title>
5          <meta charset="utf-8" />
6    </head>
7    <body>
8          <strong>
9      <?php
10         echo 'Hello World';
11     ?>
12     </strong>
13   </body>
14 </html>
```

Générer du HTML

Comment écrire des attributs ?

Listing 15 – Ecrire du HTML correct V2

```
1  <!DOCTYPE html>
2  <html lang="fr">
3    <head>
4          <title>Ma page avec du PHP</title>
5          <meta charset="utf-8" />
6    </head>
7    <body>
8
9      <?php
10         echo '<p style="color:red;">Hello World</p>';
11     ?>
12
13   </body>
14 </html>
```

Exercice

Écrire un fichier php qui affiche une image.

Correction

Écrire un fichier php qui affiche une image.

Listing 16 – Afficher une image en php

```
1  <!DOCTYPE html>
2  <html lang="fr">
3    <head>
4          <title>Ma page avec du PHP</title>
5          <meta charset="utf-8" />
6    </head>
7    <body>
8
9      <?php
10         echo '<img src="monimage.jpg" alt="image">';
11     ?>
12
13   </body>
14 </html>
```

Sommaire

Indentation du code

Il est important d'indenter le code HTML mais aussi PHP pour faciliter la lecture et la recherche d'erreurs

Listing 17 – Code non indenté

```
1  <!DOCTYPE html>
2  <html lang="fr">
3  <head>
4  <title>Ma page avec du PHP</title>
5  <meta charset="utf-8" />
6  </head>
7  <body>
8  <?php
9  echo '<img␣src="monimage.jpg"␣alt="image">';
10 ?>
11 </body>
12 </html>
```

Commentaires

- Les commentaires permettent de donner des informations et des explications lors de la lecture du code source PHP.
- Ils ne sont pas interprétés par le moteur
- Il existe des commentaires sur une seule ligne, et des commentaires sur plusieurs lignes

Commentaires

Listing 18 – Commentaires sur une seule ligne

```
1  <?php
2    //j'affiche une image
3    echo '<img src="monimage.jpg" alt="image">';
4  ?>
```

Listing 19 – Commentaires sur plusieurs lignes

```
1  <?php
2  /*
3    cet affichage ne fonctionne pas
4        echo '<img src="monimage.jpg" alt="image">';
5    je commente donc
6  */
7  echo 'ici ca fonctionne';
8  ?>
```

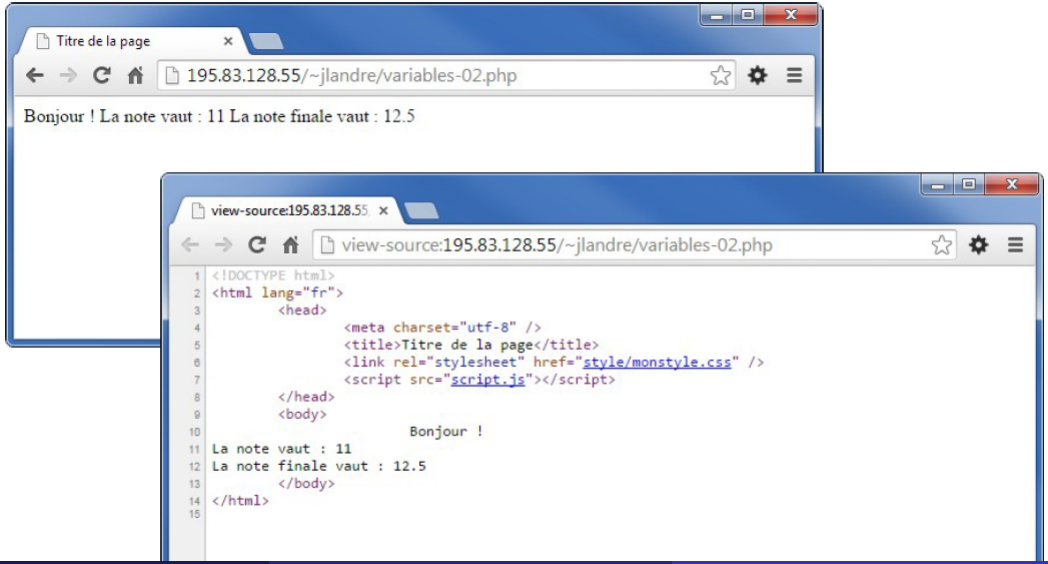
Mise en page du code généré

Si vous regardé le code généré (afficher source de la page dans votre navigateur), il est souvent difficile à lire.
Il est possible de dire à PHP d'écrire ce code un peu plus "proprement". Pour sauter des lignes dans le code HTML généré, il faut utiliser le caractère spécial \n (newline) dans des chaînes délimitées par des guillemets

Listing 20 – Code généré propre

```
1        <?php
2        echo 'Bonjour␣!'."\n";
3        $note=11;
4        echo 'La␣note␣vaut␣:␣';
5        echo $note."\n";
6        $bonus=1.5;
7        $note_totale=$note+$bonus;
8        echo 'La␣note␣finale␣vaut␣:␣'.$note_totale."\n";
9        ?>
```

Mise en page du code généré



Sommaire

Variables

- Une variable est un emplacement mémoire possédant un nom qui sert à stocker des informations
- Une variable possède un type
  - Entier, réel, booléen, chaîne de caractères, ...
  - Tableau, arbre, pile, file, date, ...

Important

- En PHP, les variables commencent **TOUJOURS** par un $

Nommage des variables

- Le nom d'une variable commence par un caractère non numérique
- Le nom d'une variable doit correspondre à ce qu'elle contient (pour faciliter la lecture)
- Les caractères spéciaux sont **interdits** dans le nom des variables
- Les espaces sont interdits dans le nom des variables, tout comme les "."
- Il est recommandé d'utiliser une notation "camelCase"

camelCase

La première lettre du nom d'une variable est toujours en minuscule, puis la première lettre de chaque mot composant le nom de la variable est en majuscule. Exemple : *$maVariable, $unLongNomDeVariable*.

Affectations

- Pour stocker quelque chose dans une variable PHP on utilise une affectation.
- L'opérateur d'affectation est le = (égal)

Listing 21 – Affectations

```
1        <?php
2        $maVariable = 10;
3        $somme = 10 + 15;
4        $moyenne = $somme / 2;
5        ?>
```

Opérations sur les variables

Listing 22 – Exemples

```php
<?php
$a = 1;
$b = 3;
$c = $a + $b; // addition
$d = $a - $b; // soustraction
$e = $a * $b; // multiplication
$f = $a / $b; // division
$g = $a % $b; // modulo (reste de la division)

?>
```

Opérations sur les variables

Listing 23 – Exemples

```
1        <?php
2        $var = 5;
3        $var++; //incrementation de 1, $var= $var + 1;
4        $var--; //decrementation de 1, $var= $var - 1;
5        $var += 2; //$var = $var + 2;
6        $var -= 3; //$var = $var - 3;
7        $var *= 4; //$var = $var * 4;
8        $var /= 2; //$var = $var / 2;
9        ?>
```

Exercice

Considérons le code php suivant

Listing 24 – Exemples

```php
<?php
$monTexte = 'Bienvenue !'; // chaine
$motDePasseUser = '123toto456'; // chaine du mot de passe
$monNombreEntier = -10; // entier
$monNombreReel = 3.1415; // reel
$monBooleen = true; // booleen
?>
```

Qu'affiche ce code ?

Exercice

Considérons le code php suivant

Listing 25 – Exemples

```php
<?php
$monTexte = 'Bienvenue !'; // chaine
$motDePasseUser = '123toto456'; // chaine du mot de passe
$monNombreEntier = -10; // entier
$monNombreReel = 3.1415; // reel
$monBooleen = true; // booleen
?>
```

Qu'affiche ce code ?

Correction

Rien bien sûr ! Il n'y a aucun **echo**, donc aucun affichage de la part de PHP !

Sommaire

Ruptures de séquence

- Règle : les programmes sont exécutés en lisant les instructions de gauche à droite et de haut en bas : **c'est la séquence d'instructions**
- Dans certains cas, il est utile et/ou obligatoire de ne pas suivre cette règle
- Il y a donc des instructions qui induisent une rupture de cette séquence, ce sont les ruptures de séquence
- Il y en existe trois en PHP :
  - Les tests conditionnels (ou conditions)
  - Les boucles
  - Les appels de fonction

Tests conditionnels

- Ils permettent d'effectuer des instructions si une certaine condition est remplie (est vraie) et/ou d'autres instructions lorsque la condition n'est pas remplie (est fausse)
- Il en existe trois en php
  - L'instruction if (si)
  - L'instruction switch/case (sélecteur à choix multiples)
  - Les conditions condensées (ou notation ternaire)

L'instruction if

L'instruction if permet de tester une condition et d'effectuer un bloc d'instructions si cette condition est **vraie**

Listing 26 – Exemples

```php
<?php
    $age=19;
    if ($age>=18)
    {
        echo '<p>';
        echo 'Vous etes majeur !';
        echo '</p>'."\n";
    }
?>
```

Bloc de séquence

Bloc d'instruction

Notez la présence des { } qui encadrent les instructions qui suivent le if (ce qui sera executé en cas de test vrai).
C'est ce qu'on appelle un bloc d'instruction. Toute séquence pourrait se trouver dans un bloc d'instruction.
Par contre, toute séquence qui est dans une rupture de séquence (après un test (if), une boucle, une fonction, ...) **DOIT** se trouver dans un bloc de séquence et entre { et }

L'instruction if

Listing 27 – Exemples

```
1  <?php
2      $age=19;
3      echo '<p>Regardons␣votre␣age␣:</p>'."\n";
4      if ($age>=18)
5      {
6          echo '<p>Vous␣etes␣majeur␣!</p>'."\n";
7      } else
8      {
9          echo '<p>';
10         echo 'Vous␣etes␣mineur␣!';
11         echo '</p>'."\n";
12     }
13     echo '<p>Bienvenue␣en␣MMI␣!</p>'."\n";
14 ?>
```

Mise en page du code généré

Regardons votre age :

Vous êtes majeur !

Bienvenue en MMI !

Qu'est ce qu'une condition

- Une condition est une expression qui peut prendre uniquement deux valeurs **booléennes** : vrai (**true**) ou faux (**false**)
- Exemples de condition :
  - L'age est supérieur ou égal à 18
  - La taille est inférieure à 1,80 m
  - La surface est supérieure ou égale à 100 m^2
  - L'age est égal à 18 et la ville de naissance est Troyes
- Pour toutes ces expressions, il n'y a que **deux choix possible**s : vrai ou faux, ce sont donc des conditions

Opérateurs de comparaison

Pour tester les valeurs des variables, on utilise les opérateurs de comparaison suivants :

Egal	==
Différent	!=
Supérieur strictement	>
Supérieur ou égal	>=
Inférieur strictement	<
Inférieur ou égal	<=

Pour lier entre elles plusieurs conditions, on utilise les opérateur et (AND), ou (OR) et non(NOT)

Et (and)	& &
Ou (Or)	||
Non (not)	!

Exemples de conditions

Listing 28 – Exemples

```php
<?php
if ($note>=10) { ... }
if ($note<10) { ... }
if !($note>=10) { ... }
if ($age>=18) { ... }
if ($age<18 && $ville=="Troyes") { ... }
if ($age>6 && $age<12) { ... }
if ($ville=="Troyes" || $ville=="Romilly") { ... }
if ($age>=18 && ($ville=="Troyes" || $ville=="Romilly")){ ... }
?>
```

Clause Else

L'instruction if peut aussi appeler un bloc d'instructions si la condition n'est pas vraie à l'aide de l'instruction **sinon** (else) :

Listing 29 – Exemples

```
1  <?php
2  $age=19;
3  if ($age>=18)
4  {
5          echo '<p>';
6          echo 'Vous etes majeur !' ;
7          echo '</p>'."\n";
8  } else
9  {
10         echo '<p>';
11         echo 'Vous etes mineur !' ;
12         echo '</p>'."\n";
13 }
14 ?>
```

Clause Elseif

On peut aussi enchaîner plusieurs conditions avec l'instruction elseif (sinon si) :

Listing 30 – Exemples

```
1  <?php
2  if ($ville=="Troyes")
3  {
4          echo '<p>Vous␣etes␣troyen␣!</p>'."\n";
5  } elseif ($ville=="Romilly")
6  {
7          echo '<p>Vous␣etes␣romillon␣!</p>'."\n";
8  } elseif ($ville=="Bar␣sur␣AUbe")
9  {
10         echo '<p>Vous␣etes␣baralbin␣!</p>'."\n";
11 } else
12 {
13         echo '<p>'Vous habitez ailleurs !</p>'."\n";
14 }
15 ?>
```

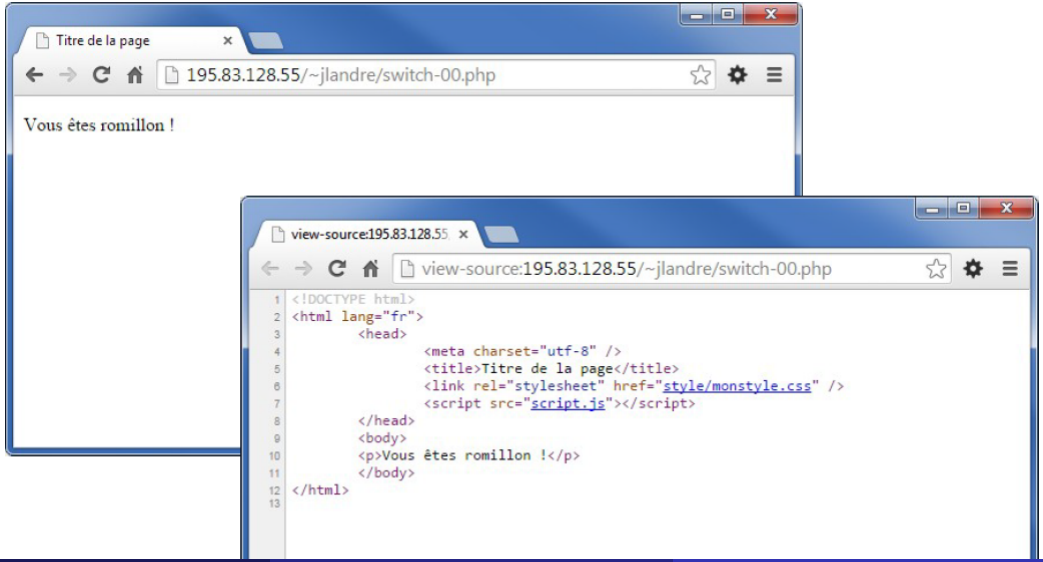
Instruction Switch

L'instruction switch/case permet de traiter plusieurs conditions sous une forme réduite :

Listing 31 – Exemples

```php
<?php
switch ($ville)
{
        case "Troyes":
                echo '<p>Vous etes troyen !</p>'."\n";
                break;
        case "Romilly":
                echo '<p>Vous etes romillon !</p>'."\n";
                break;
        case "Bar-sur-Aube":
                echo '<p>Vous etes baralbin !</p>'."\n";
                break;
        default:
                echo '<p>Vous habitez ailleurs !</p>'."\n";
}
```

Résultat


Titre de la page
195.83.128.55/~jlandre/switch-00.php
Vous êtes romillon !
view-source:195.83.128.55
view-source:195.83.128.55/~jlandre/switch-00.php
<!DOCTYPE html>
<html lang="fr">
<head>
<meta charset="utf-8" />
<title>Titre de la page</title>
<link rel="stylesheet" href="style/monstyle.css" />
<script src="script.js"></script>
</head>
<body>
Vous êtes romillon !
</body>
</html>

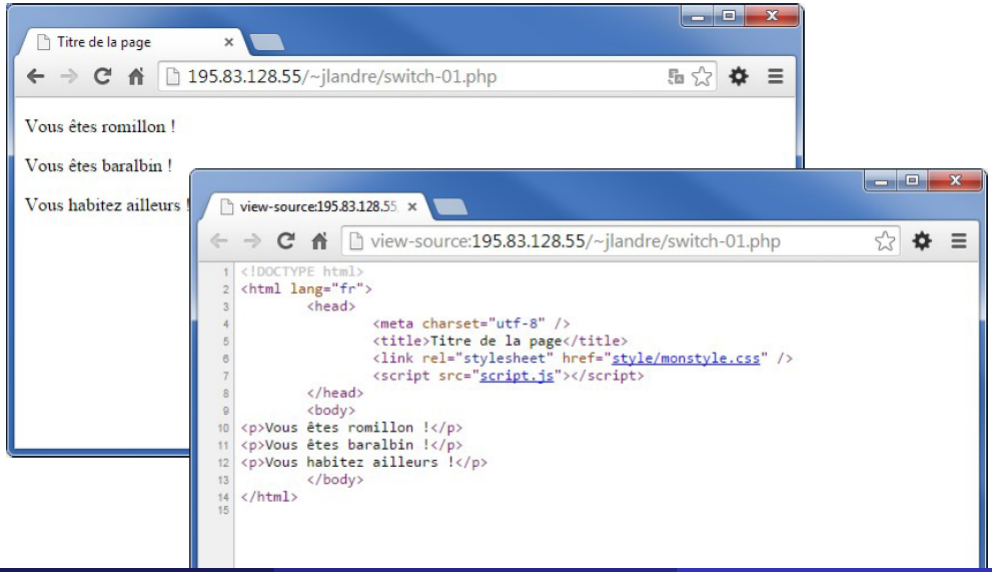
Instruction Switch

L'instruction break permet de sortir du switch/case. Si on l'oublie, les instructions suivantes vont s'exécuter.

Listing 32 – Exemples

```
1  <?php
2  $ville = "Romilly";
3  switch ($ville)
4  {
5          case "Troyes":
6                  echo '<p>Vous␣etes␣troyen␣!</p>'."\n";
7          case "Romilly":
8                  echo '<p>Vous␣etes␣romillon␣!</p>'."\n";
9          case "Bar-sur-Aube":
10                 echo '<p>Vous␣etes␣baralbin␣!</p>'."\n";
11         default:
12                 echo '<p>Vous␣habitez␣ailleurs␣!</p>'."\n";
13 }
14 ?>
```

Résultat sans break



Instruction Switch

Dans certains cas, il est utile de ne pas mettre de break quand plusieurs choix conduisent au même résultat

Instruction Switch

Listing 33 – Exemple

```php
<?php
$ville="Sens";
switch ($ville) {
        case "Troyes":
        case "Romilly":
        case "Bar-sur-Aube":
                echo '<p>Vous etes aubois !</p>'."\n";
                break;
        case "Sens":
        case "Auxerre":
                echo '<p>Vous etes icaunais !</p>'."\n";
                break;
        default:
                echo '<p>Vous habitez ailleurs !</p>'."\n";
}
?>
```

Notation ternaire

Il est possible de condenser l'écriture d'un test sur une seule ligne avec l'instruction **(condition) ?valeursivraie :valeursifausse ;**

Listing 34 – Exemple

```
1 <?php
2 $majeur=($age>=18)?true:false;
3 ?>
```

Exercice

On doit payer 10 % d'impôts si on gagne moins de 1000 € / mois et 25 % d'impôts à partir de 1000 € / mois :

Exercice

On doit payer 10 % d'impôts si on gagne moins de 1000 € / mois et 25 % d'impôts à partir de 1000 € / mois :

Correction

```
$impot=($salaire>=1000)?25/100:10/100; // ou alors
$impot=($salaire<1000)?10/100:25/100;
```

Les Boucles

- Les boucles permettent de répéter plusieurs fois un même bloc d'instructions (rupture de séquence)
- Il existe 3 boucles en PHP :
  - La boucle for ... (pour)
  - La boucle while ... (tant que)
  - La boucle do ... while ... (jusqu'à)

Boucle FOR

Définition

La boucle for permet de répéter un bloc d'instructions quand on connaît a priori le nombre d'itérations (nombre de boucles) à effectuer.

Listing 35 – Syntaxe

```
1  <?php
2          echo '<p>Punition</p>'."\n";
3          for ($i=0; $i<100; $i++)
4          {
5                  echo '<p>Je␣dois␣apprendre␣mon␣cours␣!</p>'."\n";
6          }
7  ?>
```

Boucle FOR

Définition

$i est la variable de boucle, c'est elle qui détermine le nombre d'itérations
Trois éléments dans les parenthèses :

- Initialisation de la variable de boucle
- Condition qui doit être vraie pour exécuter le bloc
- Modification de la variable de boucle

Boucle FOR

Listing 36 – Exemples

```php
<?php
        echo '<p>Liste␣des␣entiers␣de␣0␣->␣10</p>'."\n";
        for ($i=0; $i<=10; $i++)
        {
                echo '<p>'.$i.'</p>'."\n";
        }

        echo '<p>Liste␣des␣entiers␣pairs␣de␣0␣->␣10</p>'."\n";
        for ($i=0; $i<=10; $i=$i+2)
        {
                echo '<p>'.$i.'</p>'."\n";
        }
?>
```

Boucle FOR

Listing 37 – Exemples

```
1  <?php
2      echo '<p>Liste␣des␣entiers␣impairs␣de␣11␣->␣1</p>'."\n";
3          for ($i=11; $i>1; $i=$i-2)
4          {
5                  echo '<p>'.$i.'</p>'."\n";
6          }
7  ?>
```

Exercice

Ecrire une boucle qui fait la somme des nombres de 1 à 100, et qui affiche le résultat.

Correction

Ecrire une boucle qui fait la somme des nombres de 1 à 100 et qui affiche le résultat.

Listing 38 – Correction

```
1  <?php
2          echo '<p>Somme␣des␣100␣premiers␣entiers</p>'."\n";
3          $somme=0;
4          for ($i=0; $i<=100; $i++)
5          {
6                  $somme=$somme+$i;
7          }
8          echo '<p>La␣somme␣vaut␣:'.$somme.'</p>'."\n";
9  ?>
```

Boucle While

Définition

La boucle while permet de répéter un bloc d'instructions quand on ne connaît pas a priori le nombre d'itérations (nombre de boucles).

Listing 39 – Syntaxe

```php
<?php
        echo '<p>Punition</p>'."\n";
        $i=0;
        while ($i<10)
        {
                echo '<p>Je dois apprendre mon cours !</p>'."\n";
                $i++;
        }
?>
```

Boucle While

Définition

$i est la variable de boucle, c'est elle qui détermine le nombre d'itérations
Trois éléments :

- Initialisation de la variable de boucle $i=0 ; (on peut mettre autre chose que 0)
- Condition qui doit être vraie pour exécuter le bloc ($i<10)
- Modification de la variable de boucle $i++ ; (on peut agire différement sur $i)

Attention ! ! ! !

Attention aux boucles infinies (c'est à dire dont la condition ne serait jamais vérifiée !)

Listing 40 – Exemple de boucle infinie

```
1 <?php
2         echo '<p>Punition</p>'."\n";
3         $i=0;
4         while ($i<10)
5         {
6                 echo '<p>Je dois apprendre mon cours !</p>'."\n";
7         }
8 ?>
```

Je n'incrémente jamais le $i, donc $i vaut toujours 0, et il est par conséquent toujours inférieur à 10.

Exercice

Ecrire une boucle qui affiche tous les nombres impairs (à partir de 2), inférieurs à 20.

Correction

Ecrire une boucle qui affiche tous les nombres pairs (à partir de 2), inférieurs à 20.

Listing 41 – Correction

```php
<?php
        echo '<p>Nombres pairs</p>'."\n";
        $i=2;
        while ($i<20)
        {
                echo $i."\n";
        $i = $i+2;
        }
?>
```

Boucle While

Définition

La boucle do ... while ... permet de répéter un bloc d'instructions quand on ne connaît pas a priori le nombre d'itérations (nombre de boucles) au moins une fois.

Listing 42 – Syntaxe

```
1  <?php
2          echo '<p>Punition</p>'."\n".'<p>';
3          $i=0;
4          do
5          {
6                  echo 'Je dois apprendre mon cours ! '."\n";
7                  $i++;
8          } while ($i<10);
9          echo '</p>'."\n";
10 ?>
```

Boucle Do ... While

Définition

$i est la variable de boucle, c'est elle qui détermine le nombre d'itérations
Trois éléments :

- Initialisation de la variable de boucle $i=0 ; (on peut mettre autre chose que 0)
- Condition qui doit être vraie pour exécuter le bloc ($i<10)
- Modification de la variable de boucle $i++ ; (on peut agir différement sur $i)