Introduction à la programmation web serveur

Découverte et pratique avec PHP

Fabien Givors

d'après les cours de Philippe Renevier Gonin

Université Nice — Sophia-Antipolis

UE: SLZIO012

Année universitaire 2014-2015

Programme de l'option

Cours

- Introduction à la programmation côté serveur avec php
- Éléments de php, passage de paramètre d'une page à une autre
- Notion de session
- Manipulation de fichier
- Les fonctions en php
- Les formulaires
- Introduction à la POO, notion d'architecture
- Génération de données php, recherche
- Contrôle d'accès (redirection)

TP

- Réalisation d'un mini-site (échelonné sur les séances)
- Exercices enseignant les bonnes pratiques (programmation, sécurité)

Calendrier et évaluation

Cours et TP

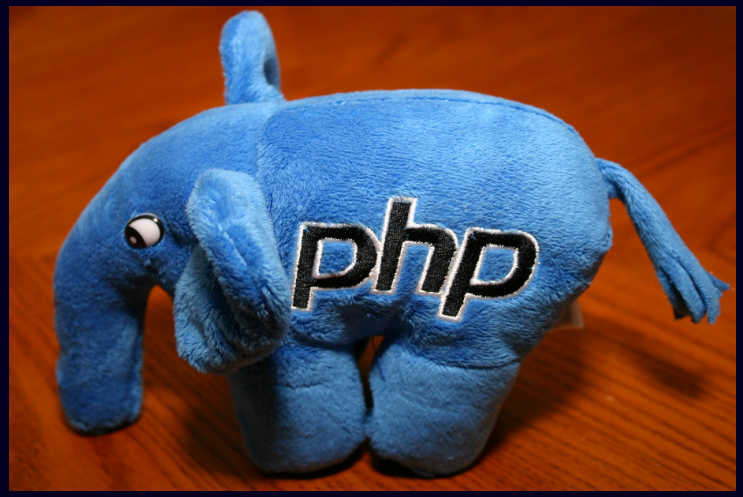
- Début le jeudi 18 septembre
- Cours les jeudi après-midi, tant que possible
- TP les lundi et jeudi après-midi (Groupes de TP à faire en fin de séance)

Évaluation

- 30% : Note de TP
- 30% : Contrôle continu
- 40% : Contrôle final

1. présentation et problématiques

   HTTP, PHP, HTML



Vue d’ensemble

- Le navigateur affiche une page Web.
- Cette page Web est en fait un fichier HTML.
- Elle est transmise au navigateur par le serveur via le protocole HTTP.
- Le HTML transmis est le résultat de l’interprétation d’un fichier PHP.

Notes :

- Le navigateur ne voit jamais le PHP
- Le PHP n’agit que sur le HTML envoyé (pas de dynamisme pendant la consultation d’une page)

PHP : Hypertext Pre Processor

Langage et interpréteur

- Un Pré-processeur pour HTML (HTTP)
- Langage impératif avec couche objet
- Typage dynamique
- Créé par Rasmus Lerdorf en 1994
- Déjà 5 versions majeures (actuelle 5.6.0)
- Interpréteur libre
- http://php.net

Langage exécuté par le serveur

- Le code est situé sur le serveur (ex : fichier index.php).
- Le navigateur fait une requête sur une page php /index.php?num=1.
- Le serveur identifie le fichier de script et ses arguments (en fonction de la requête.)
- Le serveur interprète (exécute) le fichier de script (c'est à dire le code entre balise « <?php » et « ?> »
- Le serveur renvoie le résultat au navigateur.

Exemple de fonctionnement

Le navigateur demande la ressource à l'adresse

http://example.com/index.php?num=1

Requête HTTP :

```
GET /index.php?num=1 HTTP/1.1
Host: example.com
```

Code PHP exécuté :

```
<?php
$numero = $_GET["num"];
?><p>Le <span style="font-weight: bold;">numéro</span>
entré est le <?php echo $numero; ?>.</p>
```

Données (HTML) transmises en retour par HTTP :

```
<p>Le <span style="font-weight: bold;">numéro</span>
entré est le 1.</p>
```

Affichage navigateur : « Le **numéro** entré est le 1. »

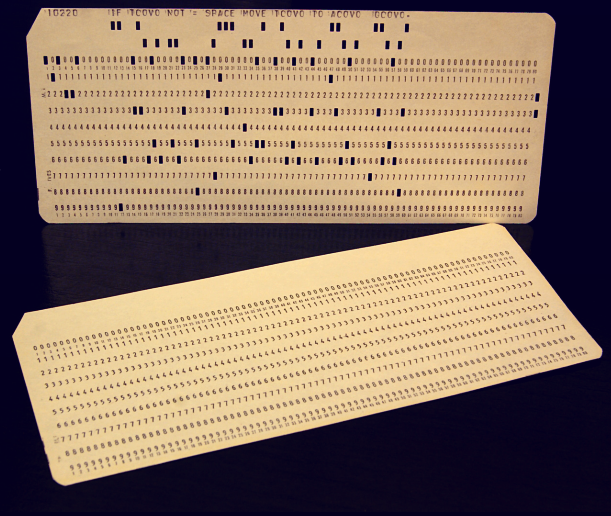
Intérêt

Un unique script PHP peut :

- analyser et traîter les données fournies avant de les afficher;
- afficher un très grand nombre de pages similaires mais différentes;
- effectuer des opérations privilégiées côté serveur :
  - envoi de fichiers de données,
  - édition de données,
  - interraction avec des bases de données,
  - etc.
- etc.

2. programmer en php

```
echo ”Hello World!”;
```



Éléments syntaxiques — Exemple de code 1

```php
<?php
/* compteCars compte le nombre d'occurrences du
   caractère $car dans la chaîne $chaine. */
function compteCars($chaine, $car)
{
  $compteur=0;
  for($i=0; $i<strlen($chaine); $i++)
  { // On itère sur chaque caractère de $chaine via $i
    if($chaine{$i} == $car) // On a trouvé un $car
    {
      $compteur++;
    }
  }
  return $compteur;
}
$phrase="abracadabra";
$nba=compteCars($phrase,"a");
echo "Il y a $nba occurrence(s) de 'a' dans « ".$phrase." »\n";
?>
```

Éléments syntaxiques — Exemple de code 1

```php
<?php
/* compteCars compte le nombre d'occurrences du
   caractère $car dans la chaîne $chaine. */
function compteCars($chaine, $car)
{
  $compteur=0;
  for($i=0; $i<strlen($chaine); $i++)
  {
    // On teste chaque caractère de $chaine via $i
    if($chaine{$i} == $car) // On a trouvé un $car
    {
      $compteur++;
    }
  }
  return $compteur;
}
$phrase="abracadabra";
$nba=compteCars($phrase,"a");
echo "Il y a $nba occurrence(s) de 'a' dans « ".$phrase." »\n";
?>
```

PHP Tags

Début d'une portion de code PHP :

```
<?php
```

Fin d'une portion de code PHP :

```
?>
```

Autant de portions de code que vous voulez dans un fichier

Éléments syntaxiques — Exemple de code 1

```php
<?php
/* compteCars compte le nombre d'occurrences du
   caractère $car dans la chaîne $chaine. */
function compteCars($chaine, $car)
{
  $compteur=0; // Initialisation du compteur
  for($i=0;$i<strlen($chaine);$i++) // On itère sur chaque caractère de $chaine via $i
  {
    if($chaine{$i} == $car) // On a trouvé un $car
      { // On itère sur chaque caractère de $chaine via $i
        $compteur++;
      }
  }
  return $compteur;
}
$phrase="abracadabra";
$nba=compteCars($phrase,"a");
echo "Il y a $nba occurrence(s) de 'a' dans « ".$phrase." »\n";
?>
```

Commentaires

Commentaire multi-ligne entre /* et */

Commentaire mono-ligne après //

Un commentaire doit être informatif et ne pas répéter le code.

Éléments syntaxiques — Exemple de code 1

```php
<?php
/* compteCars compte le nombre d'occurrences du
   caractère $car dans la chaîne $chaine. */
function compteCars($chaine, $car)
{
  $c
  fo
  {
      $compteur++;
    }
  }
  return $compteur;
}
$phrase="abracadabra";
$nba=compteCars($phrase,"a");
echo "Il y a $nba occurrence(s) de 'a' dans « ".$phrase." »\n";
?>
```

Langage impératif

- Suite d'instructions
- Une instruction se termine par un point-virgule : ;
- Un bloc d'instructions s'écrit entre accolades : { ... }
- Nombreuses fonctions internes et mots-clés : echo, strlen, ...
- *À apprendre sur le tas.*

Éléments syntaxiques — Exemple de code 1

```php
<?php
/* compteCars compte le nombre d'occurrences du
   caractère $car dans la chaîne $chaine */
function compteCars($chaine, $car)
{
  $compteur=0;
  for($i=0; $i<strlen($chaine); $i++)
  { // On itère sur chaque caractère de $chaine via $i
    if($chaine[$i] == $car) // On a trouvé un $car
    {
      $compteur++;
    }
  }
  return $compteur;
}
$phrase="abracadabra";
$nba=compteCars($phrase,"a");
echo "Il y a $nba occurrence(s) de 'a' dans « ".$phrase." »\n";
?>
```

Variables

Les noms de variables commencent toujours par $.

Déclaration et initialisation avec =

```
$compteur=0;
```

Portée : une variable n'est pas accessible en dehors du bloc { ... } dans lequel elle est déclarée.

Réutilisation de la valeur d'une variable

```
$phrase="abracadabra";
$nba=compteCars($phrase,"a");
```

Éléments syntaxiques — Exemple de code 1

```php
<?php
/* compteCars compte le nombre d'occurrences du
   caractère $car dans la chaîne $chaine. */
function compteCars($chaine, $car)
{
  $compteur=0;
  for($i=0; $i<strlen($chaine); $i++)
  {
    if($chaine{$i} == $car) // On a trouvé un $car
      $compteur++;
  }
  return $compteur;
}
$phrase="abracadabra";
$nba=compteCars($phrase,"a");
echo "Il y a $nba occurrence(s) de 'a' dans « ".$phrase." »\n";
?>
```

Tests et comparaisons

Vérifier si deux termes sont égaux avec `==`

Vérifier si un terme est plus petit qu'un autre avec `<`

Un test renvoie un booléen true ou false.

Éléments syntaxiques — Exemple de code 1

```php
<?php
/* c
   c
func
{
   $c
   fo
   {
      re
}
$phrase="abracadabra";
$nba=compteCars($phrase,"a");
echo "Il y a $nba occurrence(s) de 'a' dans « ".$phrase." »\n";
?>
```

Fonction

Permettent d'organiser et factoriser le code (éviter les répétitions).

Déclaration avec le nom et les arguments de la fonction :

```php
function compteCars($chaine, $car)
```

La définition suit ensuite entre { ... }.

La valeur calculée par la fonction est renvoyée par return

```php
return $compteur;
```

La fonction est appelée par son nom et avec ses arguments.

```php
$nba=compteCars($phrase,"a");
```

La valeur de retour est récupérée en la stockant dans une variable.

Éléments syntaxiques — Les chaînes de caractères

Définition et affectation :

- Texte entre guillemet : ”ceci est une chaîne”
- Affectation à une variable : $machaine = ”ornythorinque”;
- Substitution de variable : ”Mon animal préféré est l’$machaine”
- Accéder au $i + 1$ème caractère : $machaine{1} (== ”r”)
- Concaténation : ”B”.”A” (== ”BA”)
- Comparaisons lexicographiques : <, <=, ==, >=, >, !=

Opérations :

- Affichage : echo ”chaîne de caractère”;
- Calcul de la longueur : $l = strlen(”chaîne de caractère”);

Caractères d’échappement :

- ’\n’ : saut de ligne
- ’\t’ : tabulation
- ’\$’ : symbole « $ »
- ’\”’ : symbole « ” »

Éléments syntaxiques — Les entiers et les flottants

Définition et affectation :

- Nombre avec ou sans virgule : `$monnum = 42;`
- Comparaisons : `<, <=, ==, >=, >, !=`
- Type forçable (cast) : `(double) 4`
- Type forçable (cast) : `(int) 0.0`

Opérations :

- Affichage : `echo 42;`
- Arithmétique : `$a=2**3+2*(10/0.5)-6;`
- Et aussi :
  - modulo : `10%4 (==2)`
  - parties entières : `ceil(0.4) (==1), floor(0.4) (==0)`
  - etc.

Éléments syntaxiques — Variables

Variables globales

- accessibles et modifiables depuis toute la page
- accessibles depuis les fonctions et autres pages via le mot clé global.

Éléments syntaxiques — Variables

Variables globales

- accessibles et modifiables depuis toute la page
- accessibles depuis les fonctions et autres pages via le mot clé global.

> **Tip**
> À éviter à tout prix.
> Considéré comme une très mauvaise pratique.

Éléments syntaxiques — Variables

Variables globales

- accessibles et modifiables depuis toute la page
- accessibles depuis les fonctions et autres pages via le mot clé global.

Variables locales

- variables définies à l'intérieur d'une fonction ;
- arguments d'entrée d'une fonction ;
- accessibles uniquement depuis cette fonction.

Éléments syntaxiques — Variables

Variables globales

- accessibles et modifiables depuis toute la page
- accessibles depuis les fonctions et autres pages via le mot clé global.

Variables locales

- variables définies à l'intérieur d'une fonction ;
- arguments d'entrée d'une fonction ;
- accessibles uniquement depuis cette fonction.

```
function compteCars($chaine, $car)
{
  $compteur=0;
```

...

```
}
```

Éléments syntaxiques — Tableaux et dictionnaires

- Initialiser un tableau vide : `$t = array()`
- Initialiser un tableau simple : `$t = array(42, "meuh", -1667)`
- Initialiser un tableau avec étiquettes : `$t = array("num" => 10, "nom" => "frites")`
- Ajouter un élément à la fin d'un tableau : `$t[] = "bla"`
- Accéder au $i + 1$-ème élément : `$t[0]`
- Accéder à l'élément d'étiquette *num* : `$t["num"]`
- Vérifier si un tableau a un élément d'étiquette *num* : `has_key`
- Compter le nombre d'éléments d'un tableau : `len($t)`

Éléments syntaxiques — Tableaux et dictionnaires

Exemple

```php
<?php
$formules = array(11.90, 18.90, 22.50);
$specialites = array();
$specialites[] = array(
  "nom"  => "Salade niçoise",
  "prix" => 6.50
);
$specialites[] = array(
  "nom"  => "Socca",
  "prix" => 3.00
);
$specialites[] = array(
  "nom"  => "Pissaladière",
  "prix" => 5.50
);

  $formules[0] = 15.90;
  $specialites[1]["prix"] = 4.50;
```

Éléments syntaxiques — Structures de contrôle

- if

```php
if ($saison == "été")
{
  // Les prix de certains produits augmentent en été
  $formules[0] = 15.90;
  $specialites[1]["prix"] = 4.50;
}
```

- if ... else
- for
- while
- foreach

Éléments syntaxiques — Structures de contrôle

- if
- if ... else

```php
if ($saison == "été")
{
  // Les prix de certains produits augmentent en été
  $formules[0] = 15.90;
  $specialites[1]["prix"] = 4.50;
}
else
{
}
```

- for
- while
- foreach

Éléments syntaxiques — Structures de contrôle

- if
- if … else
- for

```
<h2>Formules</h2>
<ol>
<?php
for($i=0; $i<count($formules); $i++)
{
  echo "\t<li>Formule à ".$formules[$i]." euros.</li>\n";
}

?>
</ol>
```

- while
- foreach

Éléments syntaxiques — Structures de contrôle

- if
- if … else
- for
- while

  Plus tard.
- foreach

Éléments syntaxiques — Structures de contrôle

- if
- if … else
- for
- while
- foreach

```
<h2>Spécialités</h2>
<ul>
<?php
foreach($specialites as $specialite)
{
   echo "\t<li>La ".$specialite["nom"]." ";
   echo "est à ".$specialite["prix"]." euros.</li>\n";
}
```

Éléments syntaxiques — Opérations logiques

Notion de booléen :

- $a < $b : Vrai si $a est inférieur à $b
- true : Vrai
- false : Faux
- 0 : Faux
- non-0 integer : Faux

Éléments syntaxiques — Opérations logiques

Notion de booléen :

- $a < $b : Vrai si $a est inférieur à $b
- true : Vrai
- false : Faux
- 0 : Faux
- non-0 integer : Faux

$a === $b : Vrai si $a et $b sont le même objet

vs

(0 == false) est vrai.

(0 === false) est faux.

Éléments syntaxiques — Opérations logiques

Notion de booléen :

- $a < $b : Vrai si $a est inférieur à $b
- true : Vrai
- false : Faux
- 0 : Faux
- non-0 integer : Faux

Opérations booléennes :

- ! $a $\leftrightarrow$ non $a
- $a && $b $\leftrightarrow$ $a et $b
- $a || $b $\leftrightarrow$ $a ou $b
- $a xor $b $\leftrightarrow$ ($a et non $b) ou ($b et non $a)