Université de Montréal

PHP

Jamel Eddine Jridi

Introduction

- Le contenu de ce cours considère que ces prérequis sont acquis :
  - Une compréhension générale du monde de l'Internet et du World Wide Web
  - Une connaissance de base du langage HTML
  - Une connaissance de base du langage SQL
- PHP: Hypertext Preprocessor (≠ Personal Home Page)
- Langage de script (Perl, Shell, etc.)
- Langage interprété (≠ langage compilé)
- Supporte de nombreux SGBDs
- Langages concurrents pour les sites web dynamiques : JSP, ASP, etc.

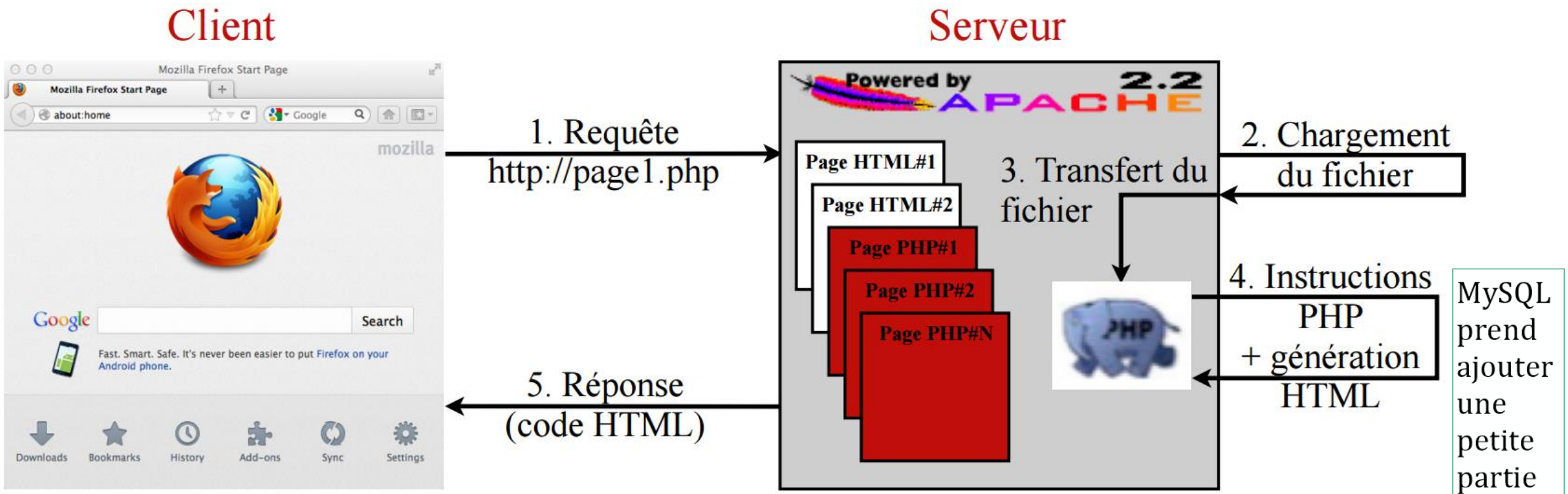
Q de l'examen

Avantages - Limites

- MultiOS (Windows, Linux, Unix, MacOS, etc.)
- Multi plates-formes (Apache, IIS, Netscape, etc.)
- Gratuit et open source (licence GNU GPL)
- Simplicité d’écriture / disponibilité de codes source
- Langage interprété : moins rapide que les langages compiles
- Maintenabilité à grande échelle

Page statique vs. dynamique

- Principe Client Serveur en PHP



comme login avec username et mot de passe et le serveur nous porte la page HTML crée pour notre porfil

Exemple de déclaration

on peut le mettre n'importe où dans le body

- Code placé entre balises <?php ... ?>
- Une instruction se termine toujours par un ;
- Commentaires
  - *//ligne commentée*
  - */* plusieurs lignes commentées */*

```
<?php
   echo 'Hello world';
   //une ligne de commentaire
   /*plusieurs
     lignes de
     commentaires*/
?>
```

PHP et HTML

- Intégration des balises PHP au sein du code HTML...
- ...ou des balises HTML au sein du code PHP
- Les balises HTML s'appliquent au code PHP

```
<html>
  <body>
    <?php
      echo 'Du texte PHP';
    ?>
    Du texte HTML
    <font size="3">
      <?php echo 'Encore du texte <font size="4">PHP</font>'; ?>
    </font>
  </body>
</html>
```

JS peut fonctionner bien sur Windows mais pour fonctionner sur LINUX il exige d'avoir un changement sur structure mais PHP a l'avantage de protabilité

Variable en PHP

- Définies sous la forme $nom_variable
  - Eviter les caractères spéciaux (accents, espaces, etc.)
  - Toujours utiliser des minuscules (garantit la portabilité sur différents systèmes d'exploitation)
- Les variables ne sont pas typées explicitement
  - Mais possibilité de transtypage
- Théoriquement, pas besoin d'initialiser les variables
- Affectation : $nom_variable = valeur ;

Types de données

- Integer (entier) - Transtypage (integer)
  - `$var = 5;`
- Double (réels à virgule flottante) - Transtypage (float), (double), (real)
  - `$var = 1.5;`
- String (chaîne de caractères) - Transtypage (string)
  - `$var = "bonjour"; $var = 'bonjour';`
- Array (tableau - cf. plus loin)
- Boolean (booléen) - Transtypage (bool), (boolean)
  - `$var = true;`
- Objects (programmation orientée objet - cf. plus loin)

Opérateurs sur les chaînes

- Un seul opérateur : concaténation “.”

```php
<?php
  $a = "Hello ";
  $b = $a."world!"; //$b contient "Hello world!"
?>
```

- Chaîne encadrée de simples ou doubles quotes
- Doubles quotes : remplacement des variables par leur valeur

```php
<?php
  $var = 'newbie';
  echo 'Hello $var'; //affiche "Hello $var"
  echo "Hello $var"; //affiche "Hello newbie"
?>
```

Opérateurs arithmétiques

$a + $b	Addition	Somme de $a et $b
$a - $b	Soustraction	Reste de la différence de $b et $a
$a * $b	Multiplication	Produit de $a par $b
$a / $b	Division	Dividende de $a par $b
$a % $b	Modulo	Reste de la division entière de $a par $b

Opérateurs logiques

$a and $b $a && $b	Et	Vrai si $a ET $b sont vrais
$a or $b $a || $b	Ou	Vrai si $a OU $b sont vrais, ou les deux
$a xor $b	Ou exclusif	Vrai si $a OU $b est vrai, mais pas les deux
!$a	Négation	Vrai si $a est faux

Opérateurs de comparaison

$a == $b	Egal	Vrai si $a est égal à $b
$a != $b	Différent	Vrai si $a est différent de $b
$a < $b	Inférieur	Vrai si $a est strictement inférieur à $b
$a > $b	Supérieur	Vrai si $a est strictement supérieur à $b
$a <= $b	Inférieur ou égal	Vrai si $a est inférieur ou égal à $b
$a >= $b	Supérieur ou égal	Vrai si $a est supérieur ou égal à $b
$a === $b	Identique	Vrai si $a est égal à $b, et de même type

Structure de contrôle

- Code placé entre accolades {}
  - Si une seule instruction apparaît, accolades inutiles
- Contrôles classiques
  - Test (*if...elseif/else if...else, switch...case*)
  - Boucles (*while, for, do...while*)
- Contrôles spécifiques à un type de variable
  - Parcours de tableaux (*foreach*)

Condition if … elseif … else

```php
<?php
  if ($expr1) {
    echo "$expr1 est vrai";
  }
  elseif ($expr2) {
    echo "$expr2 est vrai";
  }
  ...
  else {
    echo "tout est faux";
  }
?>
```

```php
<?php
  $i = 100;
  if ($i >= 0 && $i < 200) {
    echo $i.' est compris entre 0 et 199';
  }
  elseif ($i >= 200 && $i < 500) {
    echo $i.' est compris entre 200 et 499';
  }
  else {
    echo $i.' est supérieur à 499';
  }
?>
```

Itération avec do … while

```php
<?php
  do {
    //Traitement
  } while (expression);
?>
```

```php
<?php
  $i = 0;
  $max = 10;
  do {
    echo "$i est inférieur à $max";
    $i++;
  } while ($i < $max);
  echo "$i est égal à $max";
?>
```

Itération avec while

```php
<?php
  while (expression) {
    //Traitement
  }
?>
```

```php
<?php
  $i = 0;
  $max = 10;
  while ($i < $max) {
    echo "$i est inférieur à $max";
    $i++;
  }
  echo "$i est égal à $max";
?>
```

Itération avec for

```php
<?php
  for (cond init; cond sortie; iter) {
    //Traitement
  }
?>
```

```php
<?php
  $max = 10;
  for ($i = 0; $i < $max; $i++) {
    echo "$i est inférieur à $max";
  }
  echo "$i est égal à $max";
?>
```

Condition switch

```php
<?php
  switch ($foo) {
    case condition1 :
      //Traitement condition1
      break;
    case condition2 :
      //Traitement condition2
      break;
    ...
    default :
      //Traitement par défaut
  }
?>
```

```php
<?php
  $i = 1;
  switch ($i) {
    case 0 :
      echo 'La variable i vaut 0';
      break;
    case 1 :
      echo 'La variable i vaut 1';
      break;
    default :
      echo "La variable i n'appartient pas à [0-1]";
  } ?>
```

Fonctions spécifiques

- Déclaration

```php
<?php
  function nomMethode($params) {
    //Traitement de la fonction
  }

  function nomFonction($params) {
    //Traitement de la fonction
    return ($resultat);
  }
?>
```

- Appel

```php
<?php
  nomMethode($params);
  $resultat = nomFonction($params);
?>
```

```php
<?php
  function sayHello($prenom) {
    echo "Bonjour $prenom";
  }

  function additionner ($i, $j) {
    return ($i + $j);
  }
?>
```

```php
<?php
  //Affiche "Bonjour Dupont"
  sayHello("Dupont");

  $res = additionner (2, 3);
  echo $res;
?>
```

Fonction « include »

- Permet d'inclure et d'exécuter un fichier B lors de l'exécution d'un fichier A
- A utiliser pour les portions de code ou motifs répétitifs
- Renvoi une erreur de type *warning* si le fichier B est introuvable

Fichier A

```
<html>
  <body>
    <?php
      include('fichierB.php');
    ?>
  </body>
</html>
```

Fichier B

```
<?php
  // code exécuté lors de
  // l'appel au fichier A;
?>
```

c un apple serveur

Fonction « require »

- Présente les mêmes fonctionnalités que la fonction *include()*
- Mais génère une erreur de niveau *fatal* : l'exécution du script s'arrête si le fichier à inclure est introuvable

```php
<?php
  //Stoppera l'exécution du script
  //si le fichier est introuvable
  require('verif.php');

  /*
  Code exécuté si le fichier est
  présent
  */
?>
```

Les tableaux en PHP

- Les tableaux classiques (à index numérique)

```php
<?php
  //Tableau à index numéroté
  $tab = array ('valeur0', 'valeur1', 'valeur2', ...);

  //Accès à chacune des valeurs
  $val0 = $tab[0];
  $val1 = $tab[1];
?>
```

- Les tableaux associatifs (à index associatif)

```php
<?php
  //Tableau associatif
  $tab = array ('index1'=>'valeur1', 'index2'=>'valeur2', ...);

  //Accès à chacune des valeurs
  $val1 = $tab['index1'];
  $val2 = $tab['index2'];
?>
```

Parcours des tableaux

- La fonction *foreach()*

```php
<?php
  //Cas du tableau à indexe numéroté
  $tab = array ('valeur0', 'valeur1', 'valeur2');
  foreach ($tab as $val){
    echo $val; //Retourne valeur0, puis valeur1, puis valeur2
  }

  //Cas du tableau associatif
  $tab = array ('index1'=>'valeur1', 'index2'=>'valeur2');
  foreach ($tab as $val){
    echo $val; //Retourne valeur1, puis valeur2
  }
  //Consultation des indexes et valeurs
  foreach ($tab as $index=>$val){
    echo "$index a pour valeur $val";
  }
?>
```

Recherche dans un tableau

- La fonction *array_key_exists()*

```php
<?php
  //Cas du tableau associatif
  $tab = array ('index1'=>'valeur1', 'index2'=>'valeur2');
  if array_key_exits('nom_index', $tab){
    echo "L'index <i>nom_index</i> existe dans le tableau";
  }
?>
```

- La fonction *in_array()*

```php
<?php
  //Cas du tableau associatif
  $tab = array ('index1'=>'valeur1', 'index2'=>'valeur2', 'etc');
  if in_array("valeur1", $tab){
    echo "La valeur <i>valeur1</i> existe dans le tableau";
  }
?>
```

Rappel : Formulaire en HTML

- Permettent un "dialogue" avec l'internaute
- L'internaute saisit des informations en remplissant des champs ou en cliquant sur des boutons
- Un bouton de soumission valide le formulaire et l'envoie à un URL
- Balise *Form* : deux attributs obligatoires
  - *Method* indique sous quelle forme seront envoyées les données saisies
  - *Action* indique l'URL de réception des données saisies

Contenu de la balise *form*

- Une ou plusieurs balises Input
  - Différents types : *text* (champ de saisie), *checkbox* (case à cocher), *hidden* (champ invisible - pour le passage de variables PHP par exemple), *file* (fichier), *password* (texte invisible), *radio* (choix unique entre plusieurs options), *reset* (bouton de remise à zéro), *submit* (bouton de soumission), etc.
- Une ou plusieurs balises Textarea
  - Zone de saisie plus vaste que la balise *Input*
- Une ou plusieurs balises Select
  - Choix multiples entre plusieurs options

Transmission par formulaire

- Quand un formulaire est rempli et envoyé, le contenu des champs saisis est transféré à l'URL de destination sous forme de variables en utilisant les méthodes GET ou POST

```
<html>
  <body>
    <!-- envoi d'un formulaire avec la méthode POST -->
    <form action="destination.php" method="post">
      ....
    </form>
    <!-- envoi d'un formulaire avec la méthode GET -->
    <form action="destination.php" method="get">
      ....
    </form>
  </body>
</html>
```

Récupération du formulaire

- Méthode *GET* : les paramètres apparaissent dans la barre d'adresse du navigateur
- Méthode *POST* : les paramètres sont invisibles
- Récupération des valeurs des paramètres : dans les variables prédéfinies $_POST ou $_GET

```php
<?php
  //dans le cas d'un envoi des paramètres en POST
  $var1 = $_POST['nom_champ'];
  //dans le cas d'un envoi des paramètres en GET
  $var1 = $_GET['nom_champ'];
?>
```

Exemple

- Le fichier *formulaire.html*

```
<html>
  <body>
    <form action="traitement.php" method="post">
      Nom : <input type="text" name="nom"><br />
      Prenom : <input type="text" name="prenom"><br />
      <input type="submit" value="OK">
    </form>
  </body>
</html>
```

- Le fichier *traitement.php*

```
<?php
  //Récupération des paramètres du formulaire
  $nom = $_POST['nom'];
  $prenom = $_POST['prenom'];
  //Affichage des paramètres
  echo "Bonjour $nom $prenom !";
?>
```

$_post est un tableau associatif
'nom'=> 'ift'
'prenom' => '3225'

bonne exemple pour examen final

Transmission par hyperlien

http://www.iro.umontreal.ca/~lapalme/ift3225/Dynamique/
http://www.iro.umontreal.ca/~lapalme/ift3225/Dynamique/echo.php?source

- Les hyperliens peuvent être utilisés pour faire passer des paramètres ou variables d'une page source vers une page destination

```
<a href="destination.php?var1=contenu1&var2=contenu2&...">
  mon lien avec des paramètres
</a>
```

- Récupération des paramètres dans la page destination : avec la variable prédéfinie *$_GET*, ou directement avec l'appel à $var1, $var2, ...

```
<?php
  $variable1 = $_GET['var1'];
  $variable2 = $_GET['var2'];
?>
```