Le langage PHP

Philippe Rigaux – rigaux@cnam.fr
CNAM

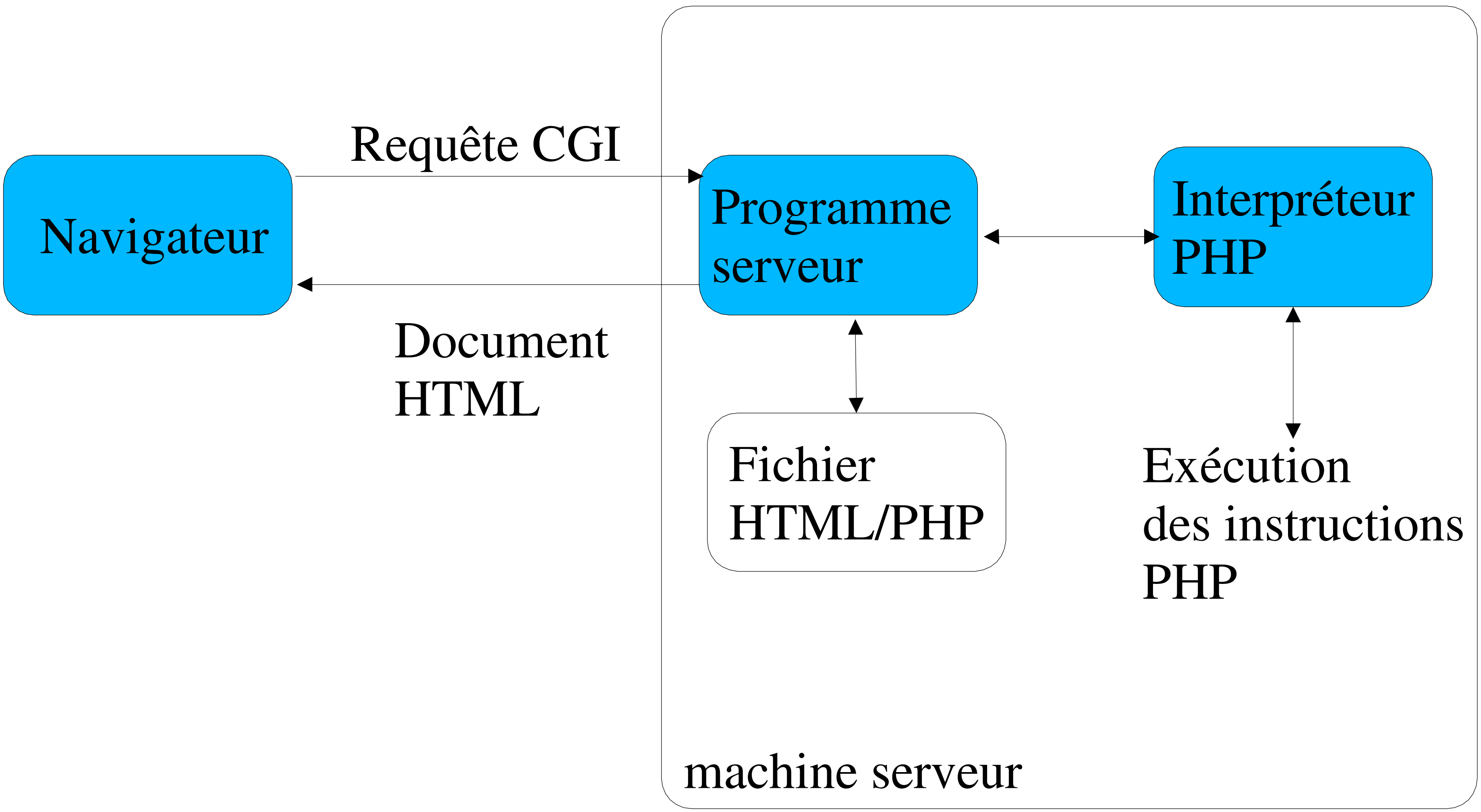
PHP, qu’est–ce que c’est ?

- Un langage pour la programmation du serveur
- Permet d’inclure des parties programmées dans du HTML
- Très proche du langage C
- Très riche en fonctions, notamment pour l’accès aux bases de données

Petit exemple PHP

```
J’ai reçu les données suivantes, venant du formulaire
<B><?php echo $monNom; ?></B> affiché par le navigateur
<B><?php  echo $HTTP_USER_AGENT ?></B> :
<P>
<TABLE BORDER=3>
<?php
  echo "<TR> <TD>Titre : <TD> $titre \n";
  echo "<TR> <TD>Année : <TD> $annee \n";
  echo "<TR> <TD>Genre : <TD> $genre \n";
  echo "<TR> <TD>Pays  : <TD> $pays \n";
  echo "<TR> <TD>Réalisateur  : <TD> $realisateur \n";
  echo "<TR> <TD>Résumé  : <TD> $resume\n";
?>
</TABLE>
```

Comment ça marche



Syntaxe de PHP

- Commentaires
  - /* ... */ ou //... ou # ...
- Variables
  - Un '$', puis (au moins) une lettre, puis tout ce qu'on veut.
  - Attention : distinction minuscules/majuscules
- Constantes
  - define (PI, 3.14116)

Déclaration de variables, types

- Pas de déclaration de variable en PHP !
  - L'apparition d'un nouveau symbole préfixé par '$' entraîne la création d'une variable

    Ex : `$maVariable = 1;`
  - La valeur initiale est "" ou 0 selon le contexte
- Le typage est dynamique
  - Une même variable peut être utilisée pour stocker plusieurs types de données
  - Des conversions sont appliquées quand c'est nécessaire.

Types scalaires

- Les entiers

```
$i = 1;
```

- Les flottants

```
$i = 1.12;
```

- Les chaînes de caractère

```
$i = "Je suis une chaîne";
```

- Booléen

```
$i = true;
```

  - NB: 0 <=> false, tout le reste vaut true.

A propos des chaînes de caractères

- Apostrophes simples : mise en page quelconque.

```
$c = ’Je suis une chaîne
                de caractères’;
```

- Apostrophes doubles : possibilité d’inclure des variables

```
$nom = "Phileas Fogg";
$c = "Je m’appelle $nom";
```

Production de HTML

- On écrit avec `echo` ou `printf`
- Des chaînes et/ou des variables
- En utilisant la concaténation, '.'
-
```
$nom = "Fogg";
$nb = 3;
echo "Je m'appelle $nom"
     . "et je prends le métro "
     . $nb . " fois par jour";
```

Tableaux

- **Tableaux indicés** :
  - liste de valeurs indicées par 0, 1, 2, ...

```
$tab[0] = 1;
$tab[1] = "une chaîne";
```

- **Tableaux associatifs** :
  - liste de valeurs référencées par des clés

```
$MES["Vertigo"]="Hitchcock";
$MES ["Gladiator"]="Scott";
```

Initialisation d'un tableau

- Avec `array` :
  - Indicé:
    ```
    $tab = array("e1","el2",120);
    ```
  - Associatif:
    ```
    $mes=array("Vertigo"=>"Hitchcock",
             "Gladiator" => "Scott");
    ```
- Implicitement (utile pour les formulaires)
  - `$tab[] = "el1";`
  - `$tab[] = "el2";`
  - `$tab[] = 120;`

Parcours d'un tableau

- Tableau indicé : en incrémentant l'indice
  - ```
    for ($i=0; $i < count($tab); $i++)
     echo "Element $i =" . $tab[$i];
    ```
- Tableau associatif : avec des fonctions
  - Un curseur sur le tableau est déplacé avec *next* ou *prev*
  - Les fonctions *key* et *current* renvoient les données du curseur.
  - ```
    do{
      echo "Clé= key($mes).El.= current($mes)"
     } while (next($mes));
    ```

Expressions

- Expression = tout ce qui a une valeur.
  - En PHP, comme en C, presques toutes les instructions ont une valeur.
    - `10;`
    - `$i = 10;`
    - `$i;`
  - L'affectation a elle–même une valeur, donc on peut écrire
    - `$j = $i = 10;`

Opérateurs

- Opérateurs arithmétiques: +, –, *, /
  - `$i = 20; $i = ($i + 2) * 34;`
- Comparaisons : ==, <, >, <=, >=
  - `$i == 20; $j == $j - 1;`
- Opérateurs logiques: && (and) || (or) ! (not)
  - `($i == 20 && $j = j-1)`
- Concaténation de chaînes
  - `$c1 = "Phileas"; $c2 = "Fogg";`
  - `$nom = $c1 . " " . $c2;`

Facilités syntaxiques

- Expressions équivalentes:
  - `$i = $i + 20;`
  - `$i += $20;`
  - Valable pour tous les opérateurs
  - `$nom = "Phileas ";`
  - `$nom .= "Fogg";`
- Incrémentations
  - Au lieu de `$i=$i+1`, on écrit `$i++` ou `++$i;`
  - Valeur de l'expression ?

Structures de contrôle : tests

- ```
  if (expression)
  {
      // bloc si expression != 0
  }
  else
  {
     // bloc si expression = 0
  }
  ```
- Rappel: `expression` = tout ce qui produit une valeur !
- Les if–then–else peuvent s'imbriquer.
  ```

Structures de contrôle : switch

```
switch (expression)
{
 case val1:
    // bloc si expression==val1
    break;
 case val2:
    // bloc si expression==val1
    break;
 ...
 default:
    // tout a échoué
    break;
}
```

Boucles : do et while

- ```
  while (expression)
  {
     // bloc tant que expression!=0
  }
  ```
- A un moment donné, expression doit devenir 0 !
- ```
  do
  {
     // bloc tant que expression!=0
  } while (expression)
  ```
- NB : le bloc s’exécute au moins une fois

Boucles : for

- ```
  $i = 0;
  while ($i < 5)
  {
     // bloc
     $i++;
  }
  ```
- Peut être remplacé par :
- `for ($i=0; $i<5; $i++) {bloc};`
- NB : on peut faire des choses **très** compliquées.
  ```

Instructions `break` et `continue`

- L’instruction `break` permet de sortir du corps d’une boucle.
- L’instruction `continue` force le passage à l’itération suivante.
-
```
$i = 0;
while (1)
{
   if (++$i == 5) break;
   if ($i == 3) continue;
   // instructions
}
```

Fonctions: passage des arguments

```
function Addition ($i, $j)
{
  // NB :$i et  $j sont des variables locales
  $somme =  $i + $j;
  $i = 2; $j = 3;
  return $somme;
}

 $i = 1; $j = 1;
 // Appel de la fonction, passage par valeur
 $k = Addition ($i, $j);
 // $i et  $j valent toujours 1 !
```

Fonctions: passage par adresse

- En préfixant par ’&’, on passe une variable par adresse.
- Introduit un effet de bord si la fonction modifie ses arguments : très mauvais

```
$i = 1; $j = 1;

// Appel de la fonction, passage par valeur
$k = Addition (&$i, &$j);

// $i et  $j valent 2 et 3 !
```

Passage par adresse avec déclaration

```
function Addition ($i, $j, &$somme)
{
  $somme =  $i + $j;
  $i = 2; $j = 3;
}

 $i = 1; $j = 1;

 // Appel de la fonction, passage par adresse de $k
 Addition ($i, $j, $k);

 // $i et  $j valent toujours 1, $k vaut 2
```

Valeurs par défaut

- Il est possible de donner une valeur par défaut aux arguments: très pratique
  - ```
    function Connexion ($nom="rigaux",
                        $passe="mdp",
                        $serveur="cortes")
    { // instructions }
    ...
    // Connexion à cortes
    Connexion ();

    // Connexion de Amman à cortes
    Connexion ("amann","mdp2");

    // Connexion à cartier
    Connexion ("rigaux","mdp","cartier")
    ```
    ```

Fonctions et variables

- Bon principe : une fonction n'accède pas aux variables du programme appelant, et réciproquement
  - **Variables automatiques** : apparaissent et disparaissent avec la fonction => très bon
  - **Variables statiques** : locales à la fonction, mais persistantes entre deux appels => bon
  - **Variables globales** : visibles partout => **TRES MAUVAIS.**
- => Se limiter aux variables automatiques

Intégration avec les formulaires

- On récupère, sous forme de variable PHP, les champs d'un formulaire
  - Pour les valeurs scalaires –> un variable
  - Pour les valeurs « multiples » –> un tableau
  - Pour les fichiers –> un nom de fichier temporaire

Passage de valeurs multiples

```
<FORM ACTION="http://cartier/ITCE/EXEMPLES/ExPHP2.php"
        METHOD=POST">

  Titre : <INPUT TYPE=TEXT SIZE=20 NAME="titre">
  <TABLE><TR><TH>Prénom</TH><TH>Nom</TH></TR>
    <TR><TD><INPUT TYPE=TEXT SIZE=30 NAME="prenom[]">
      <TD><INPUT TYPE=TEXT SIZE=30 NAME="nom[]">

    <TR><TD><INPUT TYPE=TEXT SIZE=30 NAME="prenom[]">
      <TD><INPUT TYPE=TEXT SIZE=30 NAME="nom[]">

    <TR><TD><INPUT TYPE=TEXT SIZE=30 NAME="prenom[]">
      <TD><INPUT TYPE=TEXT SIZE=30 NAME="nom[]">
  </TABLE>

  <H1>Votre choix</H1>
  <INPUT TYPE=SUBMIT VALUE='Valider'>
  <INPUT TYPE=RESET VALUE='Annuler'>

</FORM>
```

Que se passe–t–il quand on valide ?

- Le programme client transmet au programme serveur les infos sous la forme:
`titre=t&prenom[]=p1&nom[]=n1&prenom[]=p2&nom[]=n2&...`
- Pour le script PHP, cela donne:
  - ```
    $titre = t;
    $prenom[] = p1,
    $nom[] = n1;
    $prenom[] = p2,
    $nom[] = n2;
    ...
    ```
- => on crée deux tableaux PHP !
    ```

Ce que fait le script

- Une boucle sur les tableaux permet de récupérer les valeurs.

```
<?php

  echo "Pour le film $titre j'ai reçu les acteurs
        suivants :<P>\n";

  $nbActeurs = count($prenom);

  for ($i=0; $i < $nbActeurs; $i++)
   echo "<B> $prenom[$i] $nom[$i]<B><BR>\n";
?>
```

- => A utiliser aussi pour les champs SELECT multiples.

Transmission de fichiers

```
<FORM ACTION="http://cartier/ITCE/EXEMPLES/ExPHP3.php"
        METHOD=POST" ENCTYPE= "multipart/form-data">

<INPUT TYPE="hidden" name="MAX_FILE_SIZE" value="100000">
Titre : <INPUT TYPE=TEXT SIZE=20 NAME="titre">

  <P>

  Affiche : <INPUT TYPE=FILE SIZE=20 NAME='affiche'>

  <H1>Votre choix</H1>
  <INPUT TYPE=SUBMIT VALUE='Valider'>
  <INPUT TYPE=RESET VALUE='Annuler'>

</FORM>
```

Transmission de fichiers

- Le fichier est transmis sur le serveur avec quelques variables qui le décrivent:

```
<?php
 echo "Pour le film $titre j’ai reçu le fichier suivant :<P>\n";
 echo "<B>Nom du fichier client :<B>$affiche_name<BR>\n";
 echo "<B>Nom du fichier sur le serveur :<B>$affiche<BR>\n";
 echo "<B>Taille du fichier :<B>$affiche_size<BR>\n";
 echo "<B>Type du fichier :<B>$affiche_type<BR>\n";

 // Copie du fichier temporaire
 copy($affiche, "./MonAffiche");
?>
```

Intégration avec HTML

- Question : vaut–il mieux mettre du PHP dans du HTML ou l'inverse ??
  - Pour HTML : on génère le site avec un bon éditeur, et on met un peu de PHP dedans
    - Mais ... le code est ILLISIBLE
  - Pour PHP : on programme des utilitaires pour produire du HTML
    - Le code est lisible ... pour les informaticiens.

Fonctions produisant du code

```
// Fonctions produisant des conteneurs HTML
function Ancre ($url, $libelle, $classe="")
{
 return "<A HREF='$url' CLASS='$classe'>$libelle</A>\n";
}

function Image ($fichier, $largeur=–1, $hauteur=–1, $bordure=0)
{
 if ($largeur != –1) $attrLargeur = " WIDTH  = '$largeur %' ";
 if ($hauteur != –1) $attrHauteur = " HEIGHT = '$hauteur %' ";

 return "<IMG SRC='$fichier'" .  $attrLargeur
        . $attrHauteur . " BORDER='$bordure'>\n";
}
```

Création de module

- Un module = un ensemble de fonctions prenant en charge une partie de l'application
- Exemple: module de production de tableaux:
  - TblDebut –> produit la balise <TABLE>
  - TblFin –> produit la balise </TABLE>
  - TblLigne, TblCellule –> pour écrire la table