

# PHP

Laurent Réveillère

Enseirb  
Département Télécommunications

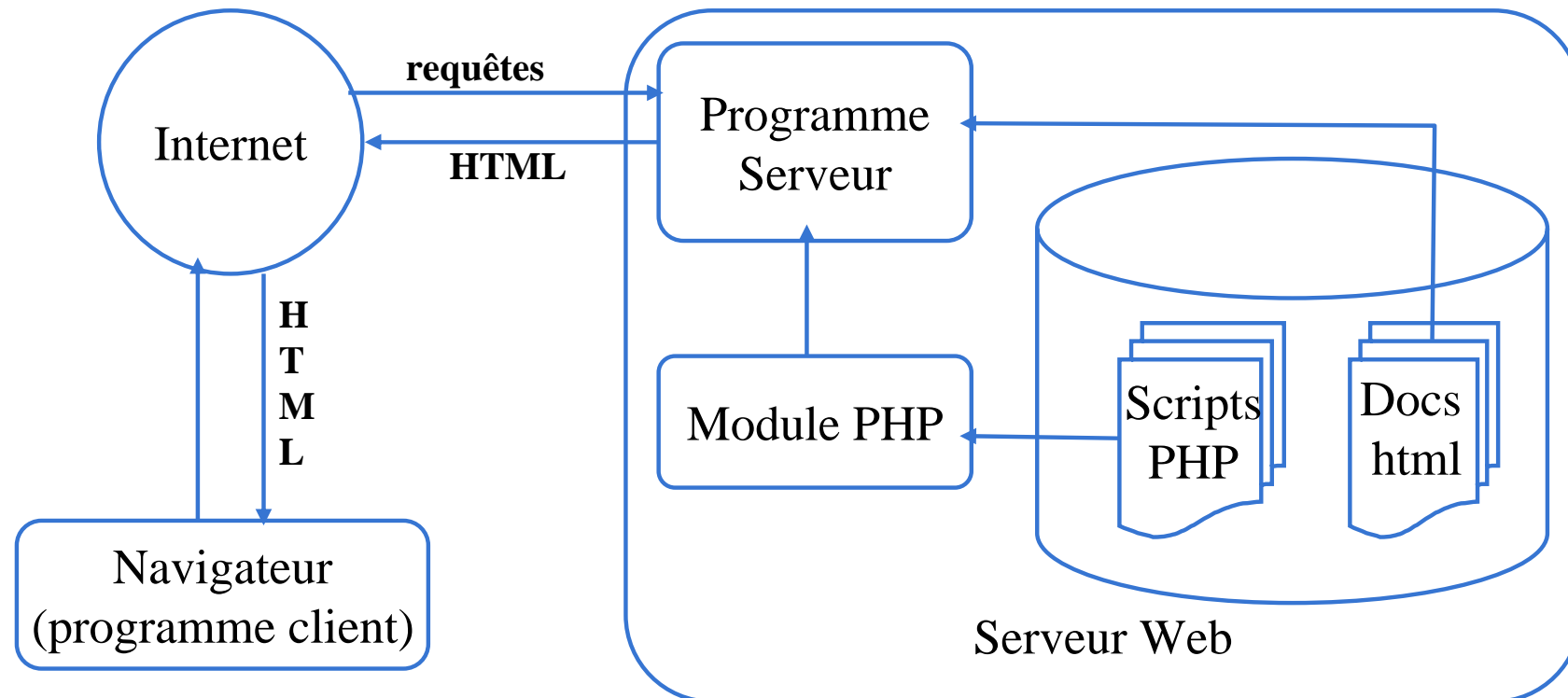
`Laurent.Reveillere@enseirb.fr`  
`http://www.enseirb.fr/~reveille`

# Introduction

---

- ◆ PHP : Hypertext Preprocessor
- ◆ PHP est un langage de programmation
  - Interprété (exécuté côté serveur)
  - Intègre des éléments de POO
  - Syntaxiquement entre C et Perl
- ◆ Code stocké sur le serveur
  - ➔ Jamais visible directement par le client

# Architecture d'une session PHP



# PHP et HTML

- ◆ Au plus court :

```
<? Instructions ?>
```

- ◆ Au plus lisible :

```
<?php instructions ?>
```

- ◆ À la mode « XML » :

```
<script language="php">  
    instructions  
</script>
```

- ◆ À la mode « MicroSoft »

```
<% instructions %>
```

# Exemple

fact.php

```
<html>
  <body>
    <h1>Table des factorielles</h1>
    <script language="php">
      $fact = 1; // Valeur de 0!
      for ($i=1; $i<=9; $i++) {
        $fact *= $i;
        echo ($i . "! = " . $fact . "<br>");
      }
    </script>
  </body>
</html>
```



# Commentaires

- ◆ Mettre en commentaire une ligne (à la C++)

```
// Tous les caractères qui suivent sont ignorés
```

- ◆ Mettre en commentaire une ligne (à la shell)

```
# Tous les caractères qui suivent sont ignorés
```

- ◆ Mettre en commentaire plusieurs lignes (à la C)

```
/* Toutes les lignes comprises entre ces repères  
sont ignorées par l'interpréteur de code */
```

- ◆ Ne pas confondre

- les balises de commentaires html (<!-- -->)
- les caractères de commentaires PHP (//, /\* \*/, #)

# Notion de variables

- ◆ Affectation d'une valeur à un identificateur

```
$x = 2.5;
```

- ◆ Nom des variables sensible à la casse `$x`  $\neq$  `$X`
- ◆ Certaines variables sont prédéfinies
  - Considérées comme des variables globales
  - Il est possible d'obtenir la liste avec `phpinfo()`
- ◆ Typage faible
  - une variable peut à tout moment changer de type
  - le type est déterminé au moment de l'affectation

# Portée des variables

- ◆ Portée dépendant de l'endroit où la variable est déclarée
  - Extérieur de toute fonction et de tout bloc
    - ➔ variable globale (accessible partout sauf au sein des fonctions)
  - Intérieur d'un bloc ou d'une fonction
    - ➔ variable locale (seulement dans le bloc)
- ◆ Accès à une variable globale dans une fonction
  - Utilisation du mot clé `global` (`global $i;`)
  - Utilisation du tableau `$GLOBALS`

# Notion de constantes

- ◆ Variable dont la valeur ne peut être modifiée
- ◆ Définition à l'aide de la fonction `define()`

```
define ("auteur", "Laurent Réveillère");  
Print ($auteur);
```

- ◆ Peuvent être utilisées partout où une variable peut l'être aussi
- ◆ Ignore la notion de portée (accessibles partout)
- ◆ Certaines constantes sont créées automatiquement par PHP (`M_PI`, `__LINE__`, ...)

# Principaux types de données

- ◆ Entiers

- ◆ Nombre décimaux

- ◆ Chaîne de caractères

- suite de caractères encadrée par des guillemets simples ( ' ) ou doubles ( " )

- caractère spéciaux

\\ caractère contre-oblique

\n retour à la ligne

\\" guillemets doubles

\t tabulation

\r retour chariot

\x00-xFF caractère hexa

- ◆ Booléen

- true **ou** false

# Types de données (suite)

## ◆ Tableaux

- Collection de valeurs associées à des
  - » Index (entier)
  - » Clés (chaîne de caractères)
- Exemple

```
$Tab[0] = 15;  
$Tab[1] = "mai";
```

≡

```
$Tab[] = 15;  
$Tab[] = "mai";
```

```
$Tab[0][0] = "15";  
$Tab[0][1] = "mai";  
$Tab[1][0] = "Jeudi";  
$Tab[1][1] = 2;
```

# Tableaux associatifs

- ◆ Indices sous forme de chaîne de caractères
  - Tableau associatif ou table de hachage
- ◆ Exemple

```
$Etudiant["Promo"] = 1;  
$Etudiant["Rang"]  = 24;  
$Etudiant["Nom"]   = "Toto";
```

# Opérateurs

- ◆ Comparaison `== != < > <= >=`
- ◆ Arithmétiques `+ - * / % ++ --`
- ◆ Logiques `&& and || or xor !`
- ◆ Manipulation de bits `& | ^ ~ << >>`
- ◆ Affectations `= += -= *= /= <<= >>= &= ~= |= .=`
- ◆ Sur les chaînes `.`
- ◆ Sur les variables `$ &`

# Instructions

## ◆ Conditions

```
if (condition) { ... }  
if (condition) { ... } else { ... }  
if (condition1) { ... } elseif (condition2) { ... }  
condition ? ... : ...  
switch (expression) {  
    case constante : ... break;  
    default: ...  
}
```

## ◆ Boucles

```
for ( initialisation ; test ; increment ) { ... }  
while (condition) { ... }  
do { ... } while (condition)  
foreach (array as key=>value) { ... }
```

# Instructions (suite)

## ◆ Déroulements

`break`

**Interruption**

`for, case, while, do-while`

`continue`

**passage à l'itération suivante**

`for, while, do-while`

`exit`

**Interruption de l'exécution du code**

# Fonctions

- ◆ Définition identique aux fonctions en C mais
  - pas de typage des arguments
  - pas de typage de la valeur de retour

```
function fact ($n) {  
    if ($n < 2)  
        return 1;  
    else return $n * fact ($n-1);  
}
```

# Fonctions (suite)

## ◆ Arguments avec valeurs par défaut

```
function printColored ($text, $color="red") {  
    print ("<div style=\"color: $color\">$text</div>");  
}  
  
printColored ("Un texte en rouge");  
print ("<br>\n");  
  
printColored ("Un texte en bleu", "blue");  
print ("<br>\n");
```

# Fonctions (suite)

- ◆ Nombre variable d'arguments
  - Depuis PHP4!

```
<?php function makeListe () { ?>
<ol>
<?php for ($i=0; $i < func_num_args(); $i++) {
    print ("<li>" . func_get_arg($i));
} ?>
</ol>
<?php } ?>
<?php makeListe ("PHP", "c'est puissant!",
"non?") ?>
```

# Fonctions standards

- ◆ PHP met à notre disposition un très grand nombre de fonctions : <http://www.php.net/>
- ◆ On peut également trouver des bibliothèques de fonctions utilitaires sur des sites comme <http://www.asp-php.net/asphp2/fr/>
- ◆ <http://www.phpdebutant.com/>
- ◆ Utilisez-les!

# Bibliothèques (1)

---

- ◆ Bases de données
- ◆ Vérification d'orthographe (aspell)
- ◆ Précision arbitraire (bcmath)
- ◆ Calendrier
- ◆ Date/heure
- ◆ dbm (Berkeley data base manager)
- ◆ Système de fichier
- ◆ Chargement dynamique de fonctions
- ◆ Exécution de programmes externes

# Bibliothèques (2)

---

- ◆ HTTP
- ◆ Manipulation d'images (gd)
- ◆ IMAP (Internet Mail Access Protocol)
- ◆ LDAP (Lightweight Directory Access Protocol)
- ◆ Courrier électronique
- ◆ Mathématiques
- ◆ Réseau (socket, ...)
- ◆ PDF (Portable Document Format)

# Bibliothèques (3)

---

- ◆ Expressions régulières
- ◆ Sémaphores, mémoire partagée
- ◆ SNMP (Simple Network Management Protocol)
- ◆ Chaînes de caractères
- ◆ Cryptage, codage
- ◆ Compression, décompression
- ◆ XML (eXtended Markup Language)
- ◆ Paiement électronique

# Inclusion de fichiers

- ◆ **require** ("fichier.inc") ;
  - Inclusion du fichier `fichier.inc`
  - Évaluation lors du pre-processing
    - » Instruction remplacée par le contenu du fichier
    - » Même dans un if
  
- ◆ **include** ("fichier.inc") ;
  - Inclusion du fichier `fichier.inc`
  - Évaluation lors de l'exécution
    - » Pas si dans une branche morte

# Ouverture d'un fichier

- ◆ Plusieurs modes d'ouverture existent :
  - lecture, écriture, lecture/écriture,...
- ◆ Fonction open

```
int fopen("nom_du_fichier", "mode")
```

- Si le nom commence par "http" réalise une connexion HTTP
- Retourne un pointeur de fichier ou FALSE en cas d'échec.
- Modes possibles : 'r', 'w', 'a', 'r+', 'w+', 'a+'

# Lecture dans un fichier

## ◆ Lecture d'au plus une ligne

```
string fgets (int fp, int length)
```

- Retourne la chaîne lue jusqu'à la longueur `length - 1` octets, ou bien la fin du fichier, ou encore un retour chariot (le premier des trois qui sera rencontré).

## ◆ Lecture de la totalité du fichier

```
array file (string filename [, int use_include_path])
```

- Retourne le fichier dans un tableau.
- Chaque élément du tableau correspond à une ligne du fichier, et les retour-chariots sont placés en fin de ligne.

# Écriture dans un fichier

- ◆ `int fputs ( int fp, string str [, int length])`
  - Écrit le contenu de la chaîne `string` dans le fichier pointé par `fp`. Si la longueur `length` est fournie, l'écriture s'arrêtera après `length` octets, ou à la fin de la chaîne (le premier des deux).
- ◆ `int fwrite ( int fp, string string [, int length])`
  - Même chose que `fwrite` sauf que fonctionne également pour des fichiers binaires

# Fermeture d'un fichier

- ◆ Tout fichier ouvert doit être fermé
- ◆ Fonction `fopen`

```
bool fclose (resource fp)
```

- ferme le fichier désigné par `fp`
- Retourne `TRUE` en cas de succès, et `FALSE` en cas d'échec.
- Le pointeur de fichier doit être valide, et avoir été correctement ouvert par la fonction `fopen()`

# Tests sur les fichiers

- ◆ Est-ce que le fichier est un répertoire?

```
bool is_dir (string nom_fichier)
```

- ◆ Est-ce que le fichier est un lien symbolique?

```
bool is_link (string nom_fichier)
```

- ◆ Est-ce que le fichier est ni un répertoire, ni un lien symbolique?

```
bool is_file (string nom_fichier)
```

# Exemple

```
<?php
$fp=fopen ("fichier.txt", "r");
// Vérifier si l'opération est un succès
if(! $fp) {
    print ("Impossible d'ouvrir fichier.txt");
    exit;
} else {
    $lineNumber = 1;
    while (!feof($fp)) {
        $lineNumber++;
        $ligne = fgets($fp, 256); // lit une ligne
        echo "$lineNumber: $ligne<br>";
    }
    fclose($fp);
}
?>
```

# Formulaires HTML (rappels)

- ◆ Exemple :

```
<form action="prog.php"  
method="post">  
    <input type="submit" name="go">  
</form>
```

- ◆ Action : URL cible

- ◆ Method : dans la pratique GET ou POST

# Gestion de formulaires en PHP

- ◆ PHP transforme tous les champs en variables d'environnement (globales)
  - Également accessible dans les tableaux associatifs `$HTTP_POST_VARS` et `$HTTP_GET_VARS` : suivant la méthode d'envoi GET ou POST
- ◆ Attention:
  - Il ne faut pas avoir deux zones de formulaires avec le même nom sur la même page
    - ➔ utilisation de tableaux

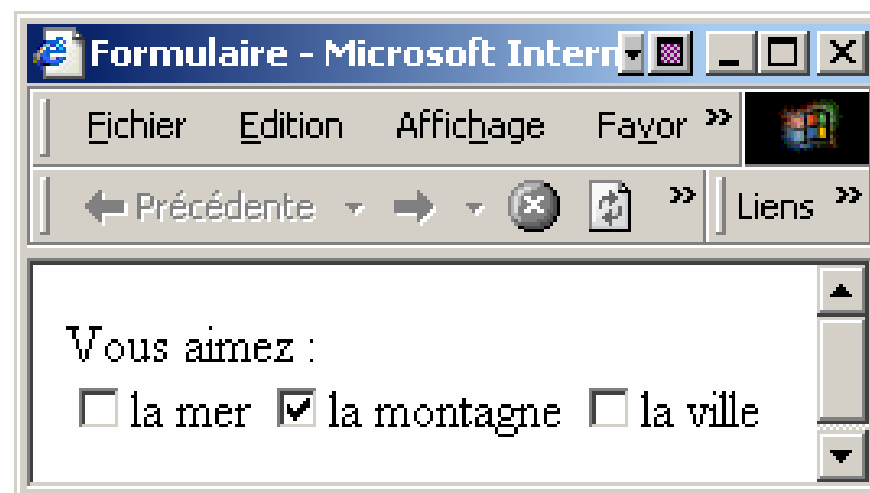
# Exemple

```
<html>
<head> <title>Changer la couleur de fond</title> </head>
<?php
// Blanc comme couleur par default
if (!isset($inputColor)) {
    $inputColor = "FFFFFF";
}
echo "<body style=\"background-color: $inputColor\" ?>
<h1>Choix de la couleur de fond</h1>
<?php
echo "<form action=\"\$PHP_SELF\" method=\"post\">";
echo "<input name=\"inputColor\" value=\"\$inputColor\">"; ?>
<input type="submit" name="envoyer" value="Essayer!">
</form>
</body>
</html>
```

# Gestion des cases à cocher

## ◆ Rappel

```
Vous aimez : <br />
<input type="checkbox" name="aime" value="mer">
la mer</input>
<input type="checkbox" name="aime" value="montagne" checked="checked">
la montagne</input>
<input type="checkbox" name="aime" value="ville">
la ville</input>
```



# Gestion des cases à cocher

- ◆ Utilisation de tableau pour le nom de la zone du formulaire

```
Vous aimez : <br />
<input type="checkbox" name="aime[]" value="mer">
la mer</input>
<input type="checkbox" name="aime[]" value="montagne" checked="checked">
la montagne</input>
<input type="checkbox" name="aime[]" value="ville">
la ville</input>
```

```
<ul>
<?php
foreach ($aime as $elem) {
    echo "<li> $elem </li>"
} ?>
</ul>
```

# Interfaçage avec un SGBD

- ◆ PHP permet un interfaçage avec un grand nombre de bases de données

Adabas D	dBase
filePro	Hyperwave
Informix	InterBase
mSQL	MS SQL Server
MySQL	Oracle 7
Oracle 8	SyBase
PostgreSQL	Solid

- ◆ Si une base n'est pas directement supportée, il est possible d'utiliser un pilote ODBC
- ◆ La communication avec les bases de données se fait à l'aide de requêtes SQL

# Accès aux bases de données

- ◆ Méthodes très similaires quelque soit le SGBD interrogé.
- ◆ En général, il faut suivre la séquence :
  1. Connexion
  2. Requête
  3. Exploitation des résultats
  4. Fermeture de la connexion
- ◆ Exemples avec MySQL

# Connexion à un serveur

- ◆ Paramètres de connexion
  - Variables Nom ou adresse IP du serveur
  - Port d'écoute du serveur
  - Nom d'un utilisateur répertorié
  - Mot de passe de cet utilisateur
- ◆ Peuvent être stockés dans des variables car non visibles pas le client (interprétation du code!)
- ◆ Fonction de connexion

```
int mysql_connect([server [:port], name, password])
```

# Exemple

```
<html>
<body>
<h1>Connexion à une base de données</h1>
<?php
$host = "mysql.enseirb.fr";
$user = "telecom";
$pswd = "mocenelet";
$base = "intervenants";

$connexion = mysql_connect ($host, $user, $pswd);
if ($connexion) {
    print "<p>Connexion établie !</p>";
} else {
    print "<p>Erreur lors de la connexion ...</p>";
    exit;
} ?>
</body></html>
```

# Sélection d'une base

- ◆ Équivalent de `USE base` sous *mysql*
- ◆ Permet de se placer dans une base
- ◆ Fonction de sélection

```
bool mysql_select_db(base [, connexion])
```

- ◆ Exemple

```
<?php  
    mysql_select_db($base, $connexion);  
?>
```

# Et aussi ...

## ◆ Exécution d'une requête

```
int mysql_query ( requete, [connexion] )
```

– Renvoie un identificateur de résultat

- » Vrai si la requête est valide
- » Dans le cas d'une clause SELECT, MySQL retourne un identifiant de résultat à utiliser ultérieurement dans les opérations de consultation

## ◆ Fermeture d'une connexion

```
bool mysql_close ( [connexion] )
```

# Exemple : e-boutique

Sources: reveille/www/cours/web/exemples/PHP/eboutique/

