Introduction à PHP

```
<?php
echo "Hello world";
?>
```

Georges GARDARIN et. al.

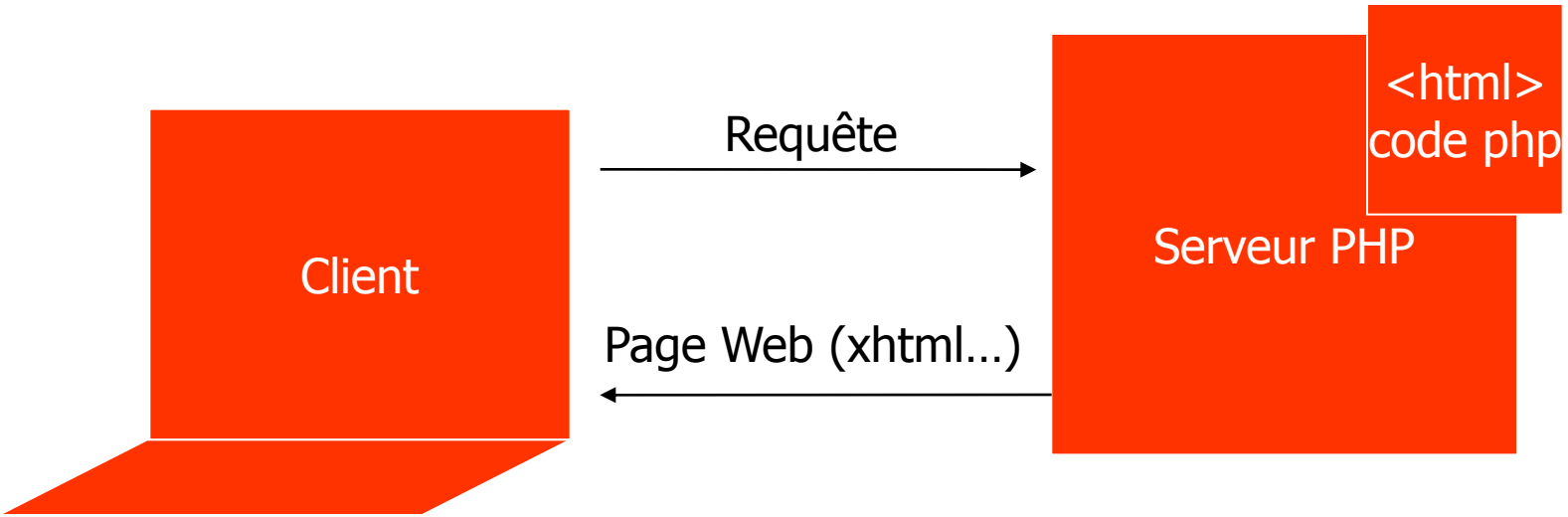
Synthèse pour les étudiants de M1 Stats

Plan

- Introduction
- Types de données
- Programmation de base
- Programmation objet
- Accès aux bases de données
- Accès au Web
  - cURL
  - XML
  - AJAX

1. PHP – Introduction (1)

- *PHP: Hypertext Preprocessor*
- *Code exécuté coté serveur*
- *Langage interprété [compilable]*



PHP – Introduction (2)

- Portabilité et distribution C/S, 3/3, P2P
- Accès aux bases de données facile
- Génération graphique sophistiquée
- Interopérabilité avec XML, parsing xml avec les standards SAX et DOM, Web services…
- Support des protocoles réseau (LDAP, IMAP, SNMP, POP3…)
- Traitement des chaînes de caractères (regex, perl)
- Nombreuses bibliothèques (PEAR, PECL, ...)

Programmer en PHP (3)

- Syntaxe proche du C
- Interpréteur souple
- Types de données classiques
- Typage à première affectation
- Conversion automatique de type
- Tableaux dynamiques à sélecteur
- Modèle objet intégré

2. Types de données

- Quatre types de données simples :
  - Integer
    - Entier compris entre -2 147 483 647 et 2 147 483 647.
  - double
    - nombre à virgule flottante compris entre -1.78*10308 et +1.78*10308.
  - String
    - chaîne de caractères de longueur variable 'string' ou "$string".
  - Boolean
    - valeur booléenne true ou false.
- Les variables ne possèdent pas de types de données prédéfinis et changent de type selon leur contenu
  - $a = 1; $a = 1.3 ; $a = ‘ABC’ ;

Les chaînes de caractères

- Entre guillemets simples ou doubles
  - 'Ma chaine $var' : pas interprétée
  - "Ma chaine $var " : interprétée
- Caractères de contrôle d'espace
- Caractères d'échappement \ permettant l'exécution du contrôle
  - \t tabulation
  - \n nouvelle ligne
  - \r retour à la ligne
  - \v tabulation verticale
- Possibilité d'imposer le suivi d'un format

Les expressions régulières

- Patterns de chaines
- Permet les recherches de sous-chaînes
- Permet les extractions et les remplacements
- Deux formes de chaîne : Unix (POSIX) et Perl (PCRE)
- Exemple (PCRE) :
  - extraction des éléments d'un tableau HTML
  - $regexp_code = '#(?s)<TD>([A-Z]+)</TD>#';
  - $regexp_chif = '#(?s)<TD>([-+.0-9]+|NUL1)</TD>#';

  - preg_match_all($regexp_code, $content, $table_code) ;

  - preg_match_all($regexp_chif, $content, $table_chif) ;

Les tableaux

- Tableaux classiques
  - // initialisation
  - $table=array(0,1,2,3,4,5);

  - // parcours simple
  - print 'table a ' . count($table) . " éléments\n";
  - for($i=0;$i<count($table);$i++)
    - print "table[$i]=$table[$i]\n";

Tableaux associatifs

- Aussi appelés dictionnaires
- Tableaux associant une clé à une valeur
- La clé peut être une chaîne de caractère ou un nombre

- // Insertion d'une ligne
  - $dico["cle"] = valeur; exemple : $dico['Pierre'] = 20 ;

- // déclaration d'un tableau
  - $dico=array('Pierre'=>20, 'Paul'=>22, 'Marie'=>18);

- // Accès à un élément
  - $variable = $dico["cle"]; exemple: $age = $dico['Pierre'] ;

3. Opérateurs

- Arithmétiques

+	Addition	5 + 2
-	Soustraction	5 - 2
*	Multiplication	5 * 2
/	Division	5 / 2
%	Modulo	5%2

- De comparaisons
  - Servent à tester une condition
  - Attention, = est l'opérateur d'affectation, pas de test d'égalité !

==	Egalité	$v == 5
<, <=	Infériorité	$v < 5
>, >=	Supériorité	$v >= 7
!=	Différent	$v != 7
%	Modulo	5%2

Opérateurs logiques et sur bits

&& and	et	$c1 && $c2 $c1 and $c2
|| or	ou	$c1 || $c2 $c1 or $c2
xor	ou exclusif	$c1 xor $c2
!	négation	!$c1
? :	Affectation conditionnelle (opérateur ternaire)	$c1 == 0 ? $c2 : $c3

&	et	$c1 & $c2
|	ou	$c1 | $c2
^	ou exclusif	$c1 ^ $c2
~	négation	~$c1
>> <<	Décalage à droite Décalage à gauche	$c1 <<2 $c2>>3

Il existe des ordres de préséance des opérateurs classiques
Il est possible d'utiliser les parenthèse () pour forcer une préséance

Structure de contrôle

```
if (cond) {  instructions 1; }
elseif (cond) {instructions 2; }
…
else(cond) {instructions N; }
```

```
switch (expression) {
case val1:    instruction; break;
case val2:    instruction;  break;
…
default:      instruction;
}
```

```
for(init,cond,evolution) {
        instructions;
   }
```

```
while (condition) {
        instructions;
        [break;]
   }
```

```
do {instructions}
while (condition) ;
```

4. Inclusion de code et sortie

- Require ("fichier")
  - Recopie un fichier de code à sa place
  - Exemple : require ("opendb.php")
- Include ("fichier")
  - Idem require mais évalue et exécute le code à chaque rencontre de include – possibilité include_once()
- Exit
  - Permet de quitter une page PHP
- Die ('message')
  - Die écrit le message et termine le programme

5. Les fonctions

- Bloc d'instructions avec paramètres optionnels créé par l'intermédiaire de l'instruction **function** :
  - function nom_fonction([$param_1 [= val_déf], ..., $param_N [= val_déf]])
  - { Bloc d'instructions...
  - [return valeurs...;] }
- Appel de fonction de la forme :
  - $resultat = nom_fonction([arg_1, ..., arg_N]);

Portée des variables et exemple

```
function SurfaceTriangle($largeur, $hauteur)
 { $resultat = ($largeur * $hauteur) / 2;
return $resultat; }

echo SurfaceTriangle(20, 10) .
" cm<sup>2</sup>"; // retourne 100 cm²

echo "($largeur * $hauteur) / 2 = " . $resultat; //
   instruction erronée : variables indéfinies
```

Exemples : fonctions sur variables

- gettype($var)
  - retourne le type de la variable argument
- settype($var)
  - change le type d'une variable, ce qui peut impliquer une conversion
- isset($var) / empty($var)
  - permet de tester si une valeur a été affectée à la variable / ou ne l'a pas été
- unset($var)
  - détruit une variable
- is_int/is_double/is_string
  - Permettent de tester le type d'une variable

Les bibliothèques de fonctions

- Chaînes de caractères
- Expressions régulières
- Gestion de fichiers

- Une grande variété
- Aperçu
  - http://www.guidephp.com/bibliotheque-php.php

6. Exemple de programme d'après Serge Tahé, université d'Angers

```
<?php
// ceci est un commentaire
// variable utilisée sans avoir été déclarée
$nom="dupont";
// un affichage écran
print "nom=$nom\n";
// un tableau avec des éléments de type différent
$tableau=array("un","deux",3,4);
// son nombre d'éléments
$n=count($tableau);
// une boucle
for($i=0;$i<$n;$i++)
print "tableau[$i]=$tableau[$i]\n";
// initialisation de 2 variables avec le contenu d'un tableau
list($chaine1,$chaine2)=array("chaine1","chaine2");
// concaténation des 2 chaînes
$chaine3=$chaine1.$chaine2;
// affichage résultat
print "[$chaine1,$chaine2,$chaine3]\n";
```

Exemple (suite)

```
// utilisation fonction
affiche($chaine1);
// le type d'une variable peut être connu
afficheType($n);
afficheType($chaine1);
afficheType($tableau);
// le type d'une variable peut changer en cours d'exécution
$n="a changé";
afficheType($n);
// une fonction peut rendre un résultat
$res1=f1(4);
print "res1=$res1\n";
// une fonction peut rendre un tableau de valeurs
list($res1,$res2,$res3)=f2();
print "(res1,res2,res3)=[$res1,$res2,$res3]\n";
// on aurait pu récupérer ces valeurs dans un tableau
$t=f2();
for($i=0;$i<count($t);$i++)
print "t[$i]=$t[$i]\n";
```

Exemple (fin)

```
// des tests
for($i=0;$i<count($t);$i++)
// n'affiche que les chaînes
if (getType($t[$i])=="string")
print "t[$i]=$t[$i]\n";
// d'autres tests
for($i=0;$i<count($t);$i++)
// n'affiche que les entiers >10
if (getType($t[$i])=="integer" and $t[$i]>10)
print "t[$i]=$t[$i]\n";
// une boucle while
$t=array(8,5,0,-2,3,4);
$i=0;
$somme=0;
while($i<count($t) and $t[$i]>0){
print "t[$i]=$t[$i]\n";
$somme+=$t[$i]; //$somme=$somme+$t[$i]
$i++; //$i=$i+1
}//while
print "somme=$somme\n";
// fin programme
exit;
?>
```

7. Programmation objet en PHP

- Class MaClasse [extends SuperClass]
  - function MaClasse(parametre1, parametre2) {
  - this.attributPublic = parametre1;
  - var attributPrive = parametre2;
  - this.methodePublique = function() { }
  - var methodePrivee = function() { }
  - }
- Instanciation
  - var objetDeMaClasse = new MaClasse("valeur1", "valeur2");
- Appel des méthodes
  - $obj = new Classe(); //Instanciation de classe
  - $obj->methodeInstance(); //Application de méthode

Exemple de classe

- class Livre extends Produit {
  - private $auteur;
  - private $editeur;
  - private $nb_pages;
  - function _construct($auteur, $editeur, $nb_pages) {
    - $this->auteur = $auteur;
    - $this->editeur = $editeur;
    - $this->nb_pages = $nb_pages; }
  - function changer_nb_pages($nb){…}
  - }
- $livre = new Livre(…);
- $livre->changer_nb_pages(150) ;

8. PHP et les bases de données

- Support de nombreuses bases de données
  - DB2, Dbase, Oracle, SqlServer, ODBC, PostgreSQL, Unix DBM…
- Interface spécifique à chaque base
- Couche d'abstraction de bases de données
  - Exemple : PEAR:MDB2
- Principe d'interface similaire
  - Exemple : MySQL

PHP – MySQL

- Ouvrir une connexion au serveur :
  - mysql_connect
- Sélectionner une base de données de travail :
  - mysql_select_db
- Effectuer une requête :
  - mysql_query
- Lire le résultat d'une requête :
  - mysql_result, mysql_fetch_array, ...
- Fermer la connexion :
  - mysql_close

```php
<?php
// Connexion et sélection de la base
$link = mysql_connect('mysql_host', 'mysql_user', 'mysql_password')
    or die('Impossible de se connecter : ' . mysql_error());
echo 'Connected successfully';
mysql_select_db('my_database') or die('Impossible de sélectionner la base de données');

// Exécution des requêtes SQL
$query = 'SELECT * FROM my_table';
$result = mysql_query($query) or die('Échec de la requête : ' . mysql_error());

// Affichage des résultats en HTML
echo "<table>\n";
while ($line = mysql_fetch_array($result, MYSQL_ASSOC)) {
    echo "\t<tr>\n";
    foreach ($line as $col_value) {
        echo "\t\t<td>$col_value</td>\n"; // ...
?>
```

PHP – SGBD : Abstraction

- Exemple de bibliothèque d'abstraction : PEAR::MDB2

```
<?php
require_once('MDB2.php');

$mdb2 =& MDB2::connect('pgsql://usr:pw@localhost/dbnam');
if (PEAR::isError($mdb2)) { die($mdb2->getMessage()); }
// La requête...
$res =& $mdb2->query('SELECT * FROM clients');
if (PEAR::isError($res)) { die($res->getMessage()); }
while (($row = $res->fetchRow())) {
foreach($row as $col) { echo $col; }
}
$mdb2->disconnect();
?>
```

9. PHP – cURL

- Bibliothèque portable
- Communication entre serveur applicatifs
- Interaction transparente via les protocoles courants

PHP – cURL : protocoles

- Support de nombreux protocoles
  - HTTP: authentification, Get, Post, gestion des cookies et des sessions
  - FTP : authentification, liste, envoi et réception de fichiers
  - LDAP : authentification, lecture
  - Autres : Telnet, Gopher, Dict, File
- Chiffrage et certificat SSL
- Connexions via un proxy

PHP – cURL Fonctionnement

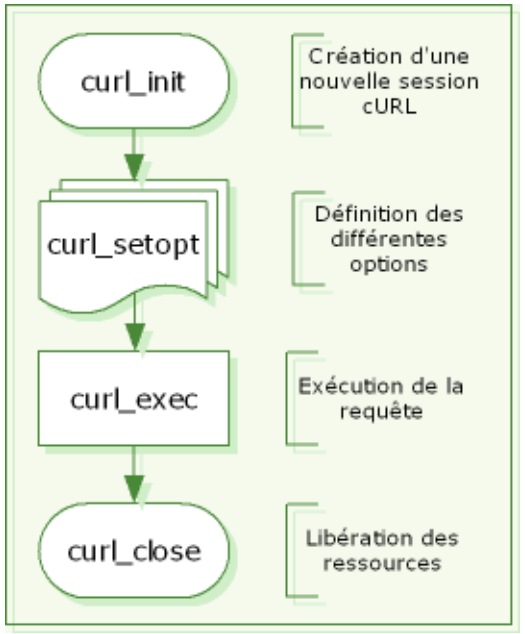
```
<?php
/** Ce code récupère le flux ATOM de Google News
et le retourne directement au navigateur **/

# Initialisation
$ch = curl_init("http://news.google.fr/"
."nwshp?hl=fr&tab=wn&output=atom");

# Exécution
curl_exec($ch);

# Fermeture de la session cURL
curl_close($ch);

# C'est tout !
?>
```



PHP – cURL

- cURL est utilisable dans d'autres langages
- cURL fait bien plus !
- Exemple :
  - récupération d'un flux ATOM dans un fichier temporaire et upload sur un serveur FTP distant

```php
<?php

define('TMP_FILE', "tmp_atom.txt");

# Initialisation ... vide !
$ch = curl_init();
# Configuration de l'URL et du retour
curl_setopt($ch, CURLOPT_URL,
"http://news.google.fr/nwshp?hl=fr&tab=wn&output=atom");
# Création du fichier de destination et ouverture rw
$fp = fopen(TMP_FILE, "w+");
# Lien : mettre le contenu ici
curl_setopt($ch, CURLOPT_FILE, $fp);
# Exécution
curl_exec($ch);
if(curl_errno($ch) != 0)  { echo "Erreur lors du téléchargement\n"; die();
}
# Fermeture de la session cURL
curl_close($ch);
```

```
# Seek au début du fichier de destination
fseek($fp, 0);
# Et c'est reparti !
$ch_ftp = curl_init();
# L'emplacement FTP de destination
curl_setopt($ch_ftp, CURLOPT_URL,
"ftp://ftp_login:password@ftp.exemple.fr/leFluxAtom");
# Indique à cURL qu'on uploade
curl_setopt($ch_ftp, CURLOPT_UPLOAD, 1);
# Transfert en ASCII
curl_setopt($ch_ftp, CURLOPT_TRANSFERTEXT, 1);
# Passage du pointeur
curl_setopt($ch_ftp, CURLOPT_INFILE, $fp);
# .. et la taille du fichier
curl_setopt($ch_ftp, CURLOPT_INFILESIZE, filesize(TMP_FILE));
# ET HOP !
curl_exec ($ch_ftp);
if(curl_errno($ch_ftp) != 0)  { echo "Erreur lors de l'upload\n"; die(); }
curl_close ($ch_ftp);
echo "Terminé avec succès !\n"
?>
```

10. PHP - XML

- eXtensible Markup Language : langage de balisage extensible
- Assurer l'intéropabilité entre les SIs

```
<article>
    <title> Extensible Markup Language </title>
    <para>
          <acronym> XML </acronym>
          « langage de balisage extensible »
    </para>

</article>
```

PHP - XML

- Parsing d’un document XML
- Séparation présentation / contenu
- Analyse syntaxique
- Librairie *expat*

PHP - XML

- Deux types d’approche :
  - L’approche hiérarchique : DOM
  - L’approche événementielle : SAX

```
<debut> Bienvenue </debut>
```

Start Element : debut
Start CDATA section, value : Bienvenue
Close Element : debut

PHP - XML

- Créer l'analyseur : *xml_create_parser()*
- Association aux 7 gestionnaires :
  - *xml_set_element_handler()*
  - *xml_set_character_data_handler()*
  - *xml_set_external_entity_ref_handler()*
  - *xml_set_unparsed_entity_decl_handler()*
  - *xml_set_processing_instruction_handler()*
  - *xml_set_notation_decl_handler()*
  - *xml_set_default_handler()*

PHP - XML

- *xml_set_element_handler()* :
  - `function ouverture ($parser, $name, $attrs)`
  - `function fermeture ($parser, $name)`
  - `xml_set_element_handler($xml_parseur, "ouverture", "fermeture");`
- *xml_set_character_data_handler()* :
  - `function texte ($parser, $data_text)`
  - `xml_set_character_data_handler($xml_parseur, "texte");`
- *xml_set_default_handler()* :
  - `function defaut ()`
  - `xml_set_default_handler($xml_parseur,"defaut");`

```php
<?php
/** Ce code récupère le flux ATOM de Google News et récupère les titres des articles **/
# Fonctions de retour (callback)
include('lib.sax.php');

$ch = curl_init("http://news.google.fr/nwshp?hl=fr&tab=wn&output=atom");
# Nous voulons récupérer le contenu dans une variable
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
# Exécution, contenu dans la variable $atomik
$atomik=curl_exec($ch);

if(curl_errno($ch) != 0)  { echo "Erreur lors de la récupération du flux\n"; die(); }
curl_close($ch);
```

```
# Création du parser
$sax=xml_parser_create();
/** Définition des fonctions de retour (callback) **/
# 1. Les tags
xml_set_element_handler($sax, 'openTag', 'closeTag');
# 2. Le contenu
xml_set_default_handler($sax, 'parseContent');

# Go go parsing
xml_parse($sax, $atomik, true); // premier et dernier morceau
echo xml_error_string(xml_get_error_code($sax));
?>
```

```php
<?php

/** Fonctions de callback **/

$printContent=false; // est ce qu'on doit afficher le contenu des balises ?

function openTag($parser, $name, $attribs)
{
    global $printContent;
    // traitement en fonction du tag ouvert
    switch(strtolower($name))
    {
        // ceci étant un exemple, on va écrire directement
        // et ne traiter qu'un échantillon de tags
        case 'title':
            echo '<h2>'; $printContent=true; break;
        case 'subtitle':
            // ... résultat : http://chipo.oxileo.net/~francois/sax.php
    }
}
```

```
function closeTag($parser, $name)
{
    global $printContent;
    switch(strtolower($name))
    {
        case 'title':
            echo '</h2>'; $printContent=false; break;
        case 'subtitle':
            // ...
    }
}

function parseContent($parser, $data)
{
    global $printContent;
    if($printContent) echo $data;
}

?>
```

11. PHP - AJAX

- **AJAX** est un acronyme signifiant Asynchronous JavaScript and XML
- Permet d'appeler des données depuis un client Web sur un serveur en asynchrone
- AJAX nécessite une architecture client/serveur Web
- Composants conformes aux standards du W3C.

PHP - AJAX

- AJAX n'est pas une technologie mais l'utilisation conjointe d'un ensemble de technologies
  - **HTML** (ou XHTML) pour la structure sémantique des informations ;
  - **CSS** ou **XSL** pour la présentation des informations ;
  - **DOM** et **Javascript** pour afficher et interagir dynamiquement avec l'information présentée ;
  - l'objet **XMLHttpRequest** pour échanger et manipuler les données de manière asynchrone avec le serveur Web.
  - **XML** pour le transfert de données

PHP – AJAX : fonctionnement

- Récupérer uniquement les données nécessaires via HTTP XMLHttpRequest
- Requêtes peuvent être « asynchrones »
- Applications plus réactives, la quantité de données échangées entre le navigateur et le serveur HTTP étant fortement réduite
- Chargement de la première page peut être pénalisé si l'application utilise une bibliothèque AJAX volumineuse

12. Conclusion

- Langage pour développer des pages Web dynamiques
- Très complet, nombreuses bibliothèques, facile …
- Possibilités de développements multi-serveurs, web services, pair à pair, etc.
- Références et compléments :
  - http://www.laltruiste.com/
  - http://php.developpez.com/cours/
  - http://www.expreg.com/ereg.php
  - http://www.manuelphp.com/
  - http://g-rossolini.developpez.com/tutoriels/php/cours/
  - http://www.php-mysql-tutorial.com/