# PHP

# PHP : histoire et état des lieux

## Un peu d'histoire

- 1994 : création de PHP/FI (Personal Home Page Tools/Form Interpreter) par Rasmus Lerdof (Canada) pour évaluer la consultation de son CV en ligne.
- 1995 : mise à disposition du code pour tout le monde.
- 1997 : redéveloppement du coeur par 2 étudiants (Andi Gutmans et Zeev Zuraski, Israël) pour donner PHP (PHP:  Hypertext Processor) 3.0.
- 2002 : 8 millions de sites basés sur PHP.
- 2004 : 15 millions de sites basés sur PHP.
- Etc.

## Evolution du langage

- Au départ basé sur le langage Perl, PHP a été réécrit ensuite en langage C.
- Le moteur a été réécrit plusieurs fois.
- D'abords procédural, le langage a évolué a partir de la version 4.0 pour intégrer les technologies objet.

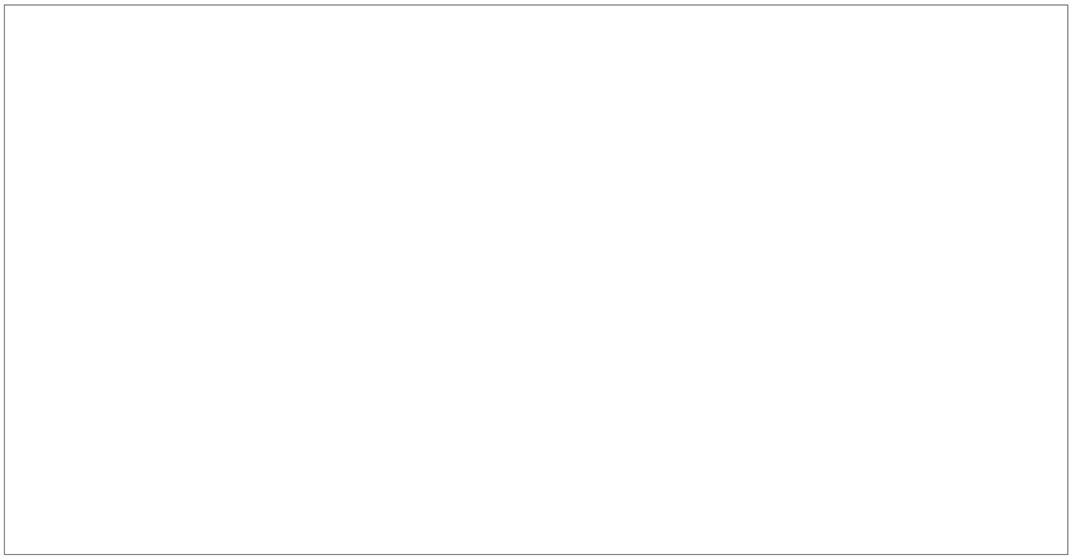
## Principales caractéristiques

- Le moteur PHP s'intègre au serveur web Apache : il prend alors en charge toutes les pages avec une extension .php.
- PHP est un langage interprété : il n'y a pas de phase de compilation et les erreurs sont découvertes au moment de l'exécution.
- La syntaxe est très inspirée du langage C, tout en apportant beaucoup de simplification, notamment dans la gestion des variables.

**Remarque : dans le présent support, on n'aborde pas les caractéristiques objet de PHP ; le code décrit est compatible PHP 3.0 et versions ultérieures.**

# PHP : principe de fonctionnement

# Principe des applications web



- Les applications web fonctionnent grâce à des échanges réseau entre un logiciel client (le navigateur web) et un logiciel serveur (le serveur web).
- Ces échanges sont basés sur le protocole HTTP : le navigateur envoie une requête au serveur web et reçoit du contenu à afficher.
- La navigation sur un site est constituée d'un ensemble de requêtes/réponses qui s'enchaînent.
- Il y a 2 types de requêtes : les requête de type GET (barre d'URL et clic sur un lien) et les requêtes de type POST (validation de formulaire).

## Apport de PHP

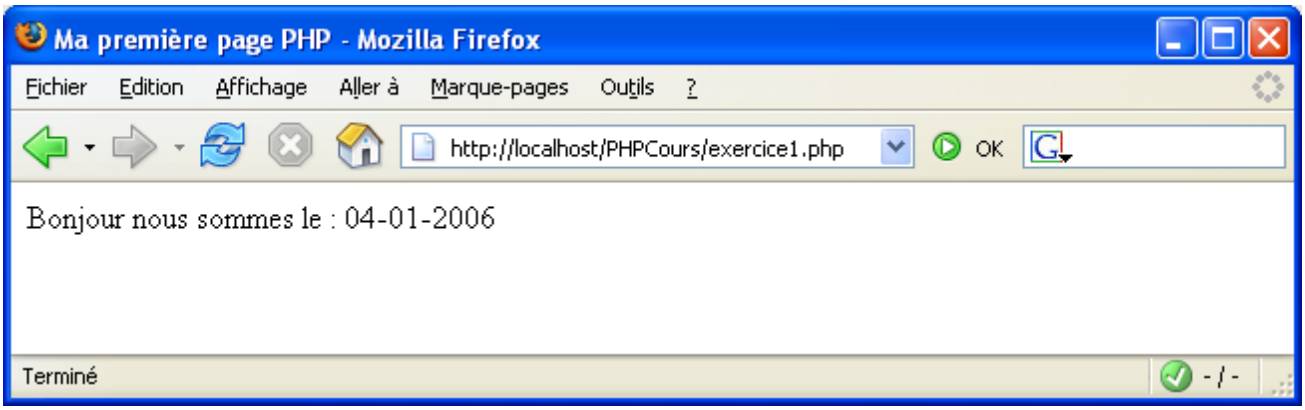
- Les pages HTML sont des pages de contenu statique stockées dans le système de fichier du serveur.
- Les pages PHP contiennent du code HTML, mais aussi du code PHP qui est exécuté sur le serveur au moment de la requête et produit du code HTML ; la page est dite dynamique, car le contenu final HTML renvoyé au navigateur peut changer selon le contexte.
- PHP nécessite l'ajout et la configuration d'un module spécifique dans le serveur web Apache.

# Premier exemple de code (1 / 2)

## Exemple simple

```
<?php echo "<?xml version=\"1.0\" encoding=\"ISO-8859-15\"?
>" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
     "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
    <head>
       <meta http-equiv="Content-Type"
             content="text/html; charset=ISO-8859-15" />
      <meta name="keywords" content="agronomie,INA-PG">
        <meta name="revisit-after" content="30 days" />
        <title>Ma première page PHP</title>
    </head>
    <body>
       <div>
        Bonjour nous sommes le :
        <?php
            $date = date("d-m-Y");
            echo "$date"; // Affichage de la date
        ?>
       </div>
    </body>
</html>
```

## Affichage



# Premier exemple de code (2 / 2)

## Code HTML généré

Si on demande l'affichage du code source dans le navigateur :

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
     "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
    <head>
       <meta http-equiv="Content-Type"
             content="text/html; charset=ISO-8859-15" />
      <meta name="keywords" content="agronomie,INA-PG">
        <meta name="revisit-after" content="30 days" />
        <title>Ma première page PHP</title>
    </head>
    <body>
       <div>
        Bonjour nous sommes le :
        O4-01-2006
       </div>
    </body>
</html>
```

- Le navigateur reçoit et interprète toujours du code HTML car le code PHP est évalué au niveau du serveur.
- Contrairement à une page HTML, on ne peut donc pas tester une page PHP directement en l'ouvrant depuis le système de fichier sans passer par une connexion HTTP sur un serveur de type Apache.

# Notions de base

## Code PHP au sein d'une page HTML

- Le code PHP est inséré directement au sein d'une page contenant du HTML.
- Le code PHP doit cependant être contenu dans des balises spéciales : <?php ... ?> ou <? ... ?>.
- Du code HTML est produit grâce à l'instruction *echo*.
- On peut mettre plusieurs instructions au sein d'une même balise <? ?>.
- Une instruction est toujours terminée par un ;.
- <? echo $variable; ?> peut être condensé en <?= $variable ?>.
- Pour être reconnu par l'interpréteur PHP, le fichier doit porter l'extension .php.

## Syntaxe de base

- L'instruction echo permet de générer de l'affichage HTML au sein d'une page.
- Les chaînes de caractères sont écrites entre "" : on doit faire précéder un " d'un \ pour l'inclure au sein d'une chaîne.
- Les noms de variables commencent toujours par le caractère $.
- Les variables n'ont pas besoin d'être déclarées, ni d'être typées ; elles sont créées implicitement lors de la première utilisation.
- Le langage fournit un certain nombre de fonction prédéfinies ; lors d'un appel de fonction, les paramètres sont passés entre ( ) et séparés par des ,.
- Les commentaires sont écrits en commençant par // sur une ligne ou entre /* et */ sur plusieurs lignes.
- Les accolades {} permettent de délimiter des blocs d'instructions.

Remarque : lorsqu'on respecte la norme XHTML, les caractères <? de la première ligne sont interprétés comme une balise PHP. Il faut donc obligatoirement générer cette première ligne avec une instruction PHP echo.

TD P1

# Structures de contrôle

PHP propose toutes les structures de contrôle classiques.

## Tests conditionnels

```
<?
    if (is_numeric($texte)) {
        echo "La variable contient un nombre";
    }
    else {
        echo "La variable contient du texte";
    }
?>
```

On peut enchaîner des tests avec l'instruction *elseif*.
On peut utiliser l'instruction *switch* lorsqu'on a beaucoup de conditions.

## Schémas itératifs

```
<?
    for ( $i = 1; $i <= 10 ; $i++)
    {
        echo $i . ' ';
    }

    $i = 1;
    while ($i<=10) {
        echo $i . ' ';
        $i++;
    }

    foreach(range(1, 10) as $i)
    {
?>
        <?= $i ?>
<?
    }
?>
```

TD P2

# Interactions avec l'utilisateur (1 / 2)

## Principe général

Dans une application web, l'interaction avec l'utilisateur est essentiellement fondée sur la validation de formulaire HTML, qui provoque l'envoie de données vers le serveur et l'affichage de nouvelles informations en provenance de celui-ci.
Des technologies telles que Javascript permettent la gestion d'événements, mais celle-ci reste locale au navigateur et donc limitée.

## Formulaire : côté client

Soit un fichier formulaire.php :

```
...
<body>
    <div>
        <form action="validation_formulaire.php"
            method="post">
            <p><label for="nom">Nom :</label>
            <input type="text" size="30" name="nom" /></p>
            <p><label for="prenom">Prénom :</label>
            <input type="text" size="20" name="prenom" /></p>
            <p><input type="submit" name="btAction"
                value="OK" />
            <input type="reset" name="btAnnuler"
                value="Annuler" /></p>
        </form>
    </div>
</body>
...
```

Il faut préciser dans la balise form le nom du fichier PHP qui traitera les données, ainsi que la méthode de transfert de ces données (POST conseillé).

Il est important de donner un nom à chaque élément du formulaire : ce nom permet ensuite de retrouver les valeurs associées saisies par l'utilisateur.

# Interactions avec l'utilisateur (2 / 2)

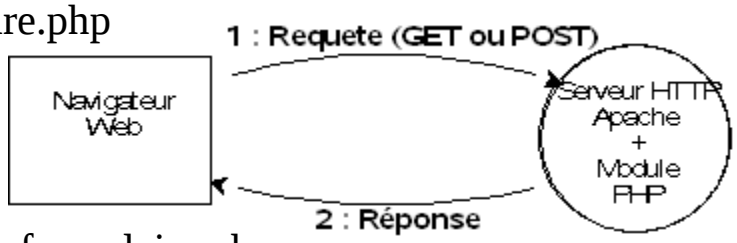
## Formulaire : côté serveur

Le formulaire précédent est traité par validation_formulaire.php :

```
...
<body>
    <div>
        <?
            if (empty($_POST["nom"])
                or empty($_POST["prenom"])) {
                echo "Vous devez saisir
                    un nom et un prénom !";
            }
            else {
                $nom = $_POST["nom"];
                $prenom = $_POST["prenom"];
                echo "Bonjour $prenom $nom";
            }
        ?>
    </div>
</body>
```

## Enchaînement

formulaire.php




validation_formulaire.php (résultat HTML)

Interprétation de validation_formulaire.php

1. Le navigateur rassemble les informations saisies dans le formulaire (nom et prénom) et les envoie par une méthode POST au serveur.
2. Le serveur interprèle validation_formulaire.php avec les données reçues et génère un flux HTML résultat renvoyé au navigateur.

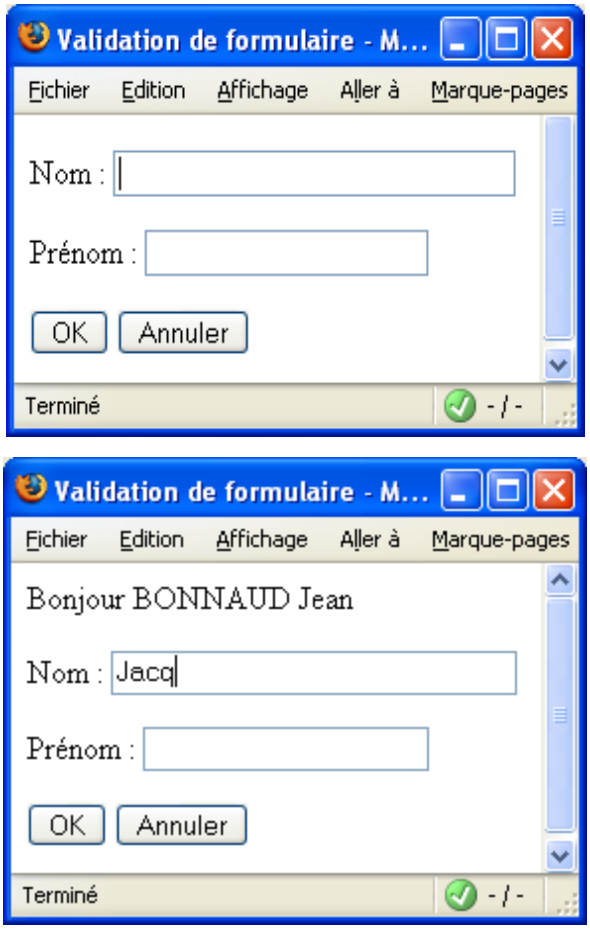
# Formulaire validé par lui-même (1 / 2)

## Principe

Dans l'exemple précédent, lorsque l'utilisateur s'est trompé ou s'il veut faire une nouvelle saisie, il soit revenir en arrière en utilisant la touche back du navigateur.
Il est souvent plus judicieux d'afficher le formulaire et le résultat sur une même page :

- lors du premier accès, la méthode GET est employée ;
- lors  de la validation, la méthode POST est employée.

## Exemple



# Formulaire validé par lui-même (2 / 2)

```
...
<body>
   <div>
        <?
             if ($_SERVER["REQUEST_METHOD"] == "POST") {
                // on est appelé par la méthode POST, donc
                // en validation du formulaire
                if (empty($_POST["nom"])
                    or empty($_POST["prenom"])) {
                    echo "Vous devez saisir un nom
                            et un prénom !";
                }
                else {
                    $nom = $_POST["nom"];
                    $prenom = $_POST["prenom"];
                    echo "Bonjour $prenom $nom";
                }
             }
        ?>
        <form action="toutenun.php" method="post">
            <p><label for="nom">Nom :</label>
            <input type="text" size="30" name="nom" />
            </p>
            <p><label for="prenom">Prénom :</label>
            <input type="text" size="20" name="prenom" />
            </p>
            <p><input type="submit" name="btAction"
                value="OK" />
            <input type="reset" name="btAnnuler"
                value="Annuler" />
            </p>
        </form>
   </div>
</body>
...
```

TD P3

# Tableaux

## Exemples précédents

Dans les précédents exemples de code, on a :

- `$_SERVER["REQUEST_METHOD"]`
- `$_POST["nom"]`
- `$_POST["prenom"]`

$_SERVER et $_POST sont en fait des tableaux associatifs pré-remplis.

## Tableaux associatifs

Un tableau associatif permet de faire correspondre des valeurs à des clés :

```
<?php
   // Création d'un tableau de 3 éléments
   $tableau = array(1=>"rouge", 2=>"bleu", 3=>"jaune");

   // Ajout d'un élément à la fin du tableau
   $tableau[] = "vert";

   // Modification du 2ème élément
   $tableau[2] = "vert";

   // Affichage du 3ème élement
   echo $tableau[3];

   // Ajout d'un élément associé à "rien"
   $tableau["rien"] = "transparent";

   // affichage de tout tableau :
   // Array ( [1] => rouge [2] => vert ... )
   print_r($tableau);

   // Parcours du tableau
   foreach ($tableau as $index => $couleur) {
       echo "<br /> $index => $couleur";
   }

   // Effacement du 2ème élément
   unset($tableau[2]);

   // Effacement de tout le tableau
   unset($tableau);
?>
```

TD P4

# Fonctions

## Quelques fonctions prédéfinies

| | |
|---|---|
| Chaînes de caractères | **strlen($chaine)** : longueur d'une chaîne<br>**strpos($chaine, $car)** : position d'un caractère<br>**strtolower($chaine)** : conversion minuscules<br>**strtoupper($chaine)** : conversion majuscules |
| Dates | **getdate()** : date et heure<br>**gettimeofday()** : heure actuelle<br>**date($format, $date)** : formatage |
| Tableaux | **count($tableau)** : nombre d'éléments<br>**sort($tableau)** : tri |

## Fonctions définies par l'utilisateur

PHP permet de créer des fonctions :

```
<?php
    function factorielle($nombre) {
        $resultat = 1;
        for ( $i = 2; $i <= $nombre ; $i++) {
            $resultat = $resultat * $i;
        }
        return $resultat;
    }
?>
```

On peut définir des fonctions dans des fichiers .php et les utiliser dans d'autres fichiers grâce à l'instruction include :

```
<?php
    include "fonctions.php";
    echo factorielle(5);
?>
```

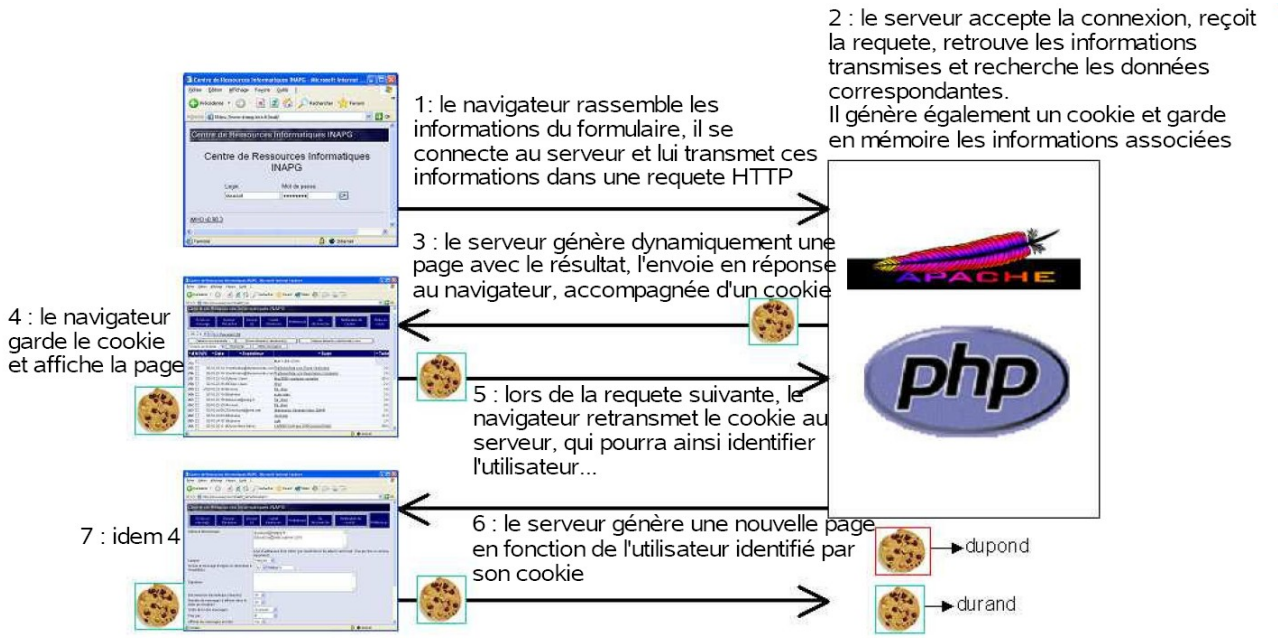
TD P5

# Suivi de la navigation : notion de session

Par nature, le dialogue entre un navigateur web et un serveur web est déconnecté, c'est-à-dire que la connexion n'est pas maintenue entre les deux lors de la navigation. Pour chaque chargement de page, les étapes se déroulent de la manière suivante :

1. Le navigateur initialise une connexion réseau avec le serveur web ;
2. Le navigateur transmet l'URL de la page demandée ;
3. Le serveur retrouve la page et transmet son contenu au navigateur ;
4. La connexion est rompue.

Cependant, dans la plupart des sites dynamiques on a besoin de "suivre" un utilisateur pendant sa navigation (par exemple : webmail, remplissage d'un chariot virtuel sur un site de commerce électronique).
Le mécanisme des sessions permet de suivre un utilisateur par l'intermédiaire de cookies selon le schéma suivant :



# Projet PHP : gestion de session

PHP permet d'initialiser la gestion d'une session, et d'y associer des informations différentes pour chaque utilisateur :

- Pour utiliser la session dans une page :
  `session_start()`
- Conserver une information dans la session :
  `$_SESSION["nom_information"] = $valeur`
- Retrouver une information de la session :
  `$valeur = $_SESSION["nom_information"]`
- Arrêter la gestion des sessions :
  `session_unset()`
  `session_destroy()`

Les sessions permettent de conserver des informations entre plusieurs pages PHP pour la navigation d'un même utilisateur sur le site.

En effet, une variable d'un fichier PHP n'existe par contre que le temps d'exécution de ce fichier, c'est-à-dire que le temps de génération de la page web correspondante.

Typiquement, après identification d'un utilisateur :

- on place l'identifiant de l'utilisateur connecté dans la session ;
- au début de chaque fichier PHP, on vérifie que l'identifiant de l'utilisateur stocké dans la session n'est pas vide ;
- si l'identifiant est vide, alors on met dans la session un message d'erreur, et on redirige l'utilisateur vers la page de connexion dans laquelle on affiche le message d'erreur ;
- on détruit la session lorsque l'utilisateur décide de se déconnecter.

Les sessions sont associées à un timeout, c'est-à-dire qu'elles expirent automatiquement au bout d'un certain temps d'inactivité de l'utilisateur.

# Ressources

## Installation

http://www.easyphp.org
http://www.eclipse.org
http://www.phpeclipse.de

## Développement

http://www.php.net/manual/fr/
http://www.phpfrance.org
http://www.phpindex.com
http://www.commentcamarche.net/php/phpintro.php3
http://php.developpez.com