Technologies Web

Farah Benamara Zitoune
Maître de conférences
IRIT-UPS
benamara@irit.fr

Plan du cours

- Cours 1 : Introduction HTML/CSS
- Cours 2 : Introduction programmation web + javascirpt
- Cours 3 : Introduction PHP
- Cours 4 : Php et Base de données
- Cours 5 : Php session+Cookies
- Si on a le temps : Introduction à Ajax et problèmes de sécurité

Origine de PHP

- Né avec le site de Rasmus Lerdof en 1994: page personnelle qui permettait de conserver une trace de passage des utilisateurs.
- Mise en ligne de Php 1.0
- Aujourd'hui, Php 5.6.6
  - Site officiel de Php France. http://www.phpfrance.com/
  - Tutoriels, documentations, forum de discussions…

PHP ne signifie pas "People Hate Perl!" **Mais** "Personal Home Page is an Hypertext PreProcessor"

Principaux atouts de PHP

- PHP est supporté par le **serveur web Apache**, le plus répandu dans le monde (plus de 70% des serveurs web),
  - Il fonctionne évidemment avec d'autres serveurs web comme IPlanet, IIS....
- PHP permet d'exploiter facilement de très nombreuses bases de données comme :
  - Oracle, MySQL, dBase, Sybase, PostgreSQL, MSQL
- PHP reconnait l'essentiel des protocoles et formats disponibles sur Internet et intranet
  - TCP, HTTP, SMTP, POP, XML, PDF...

Principaux atouts de PHP

- PHP est gratuit et performant tout comme MySQL. Le duo PHP/MySql est particulièrement facile à mettre en place.
- PHP/MySql est très largement documenté car de plus en plus répandu notamment sur les sites professionnels.
- PHP est multi-platesformes : Windows, UNIX, LINUX et MAC OS.

Principaux atouts de PHP

- Le code PHP est fortement inspiré du C et de Perl, ce qui en facilite l'apprentissage.
- Un des gros avantages de PHP sur d'autres langages comme PERL est l'intégration dans la même page du code HTML « brut » et du code PHP.
  - Le code PHP s'imbrique dans le code HTML en étant délimité comme tel.
- Grâce à de nombreuses extensions dynamiques, PHP peut générer des fichiers PDF, s'interfacer avec des serveurs de messagerie, générer des images et graphiques à la volée, ou encore générer des animations flash.

Un programme PHP, c'est…

- Un fichier HTML…
- dans lequel on trouve des tags PHP :

```
<html>
<head><title>TEST</title></head>
<body>
<?php
    echo "Bonjour MasterII Bio Info <br > " ;
    echo "Bienvenu au cours de PHP";
?>
</body>
</html>
```

Exercice 1:

- Écrire cette page. Qu'affiche-elle?

Un programme PHP, c'est…

La syntaxe du PHP est très proche de celle du langage C, à savoir:

- ; à la fin de chaque ligne d'instructions.
- {…} pour encadrer un bloc d'instructions
- les opérateurs de comparaison et d'affectation sont les mêmes (&&, ||, ==, …)
- les symboles des commentaires // et /* … */
- toute une série de mots réservés qui correspondent à des mots-clés du langage.

Exercice 2:

- Écrire cette page. Qu'affiche-t-elle?

- Afficher le code source de cette page sur votre navigateur. Que constatez vous?

```
<html>
<head>
<title>Un exemple de page en php</title>
</head>
<body bgcolor="#FFFFFF">
<font size="3" face="arial">
<?php
echo "Texte généré par PHP. 1er "; // mon premier essai
echo "<br>Date du jour = <font color=\"red\"><b>". date("d/m/y - H:i:s") .
"</b></font>";
# fin du commentaire sur mon premier essai
/* ceci est un commentaire qui s'étale
sur plusieurs lignes... */
?>
</font>
</body>
</html>
```

Concaténation de chaînes de caractère

Déclaration de variables

- PHP est un langage dit de **"typage faible et dynamique“.** Il n’est pas nécessaire de:
  - déclarer au préalable les variables
  - de préciser leur type qui est défini par défaut et de façon transparente par PHP en fonction de leur valeur.

- En PHP, tous les noms de variable commencent obligatoirement par le signe **$**

Déclaration de variables

- **Remarques sur les noms de variables :**
  - ils sont obligatoirement des chaînes de caractères non numériques et non réservées
  - les caractères de ponctuation (. , ; ...) et le caractère - ne sont pas acceptés
  - éviter les caractères accentués
  - bannir les espaces
  - le premier caractère ne peut être un chiffre
  - préférer les noms explicites
  - **PHP est sensible à la casse !**
    - $nbre est différent de $Nbre

Déclaration de variables

- Les types de données standards en PHP sont :
  - Integer (entier)
    - $var = 12;
  - Double (réels représentés par des virgules flottantes)
    - $var = 3.1415957
  - String (chaînes de caractères)
    - $var = "bienvenue"
  - Array (tableaux)
  - Object (programmation orientée objet)

Déclaration de variables

- $var = "0"; //$var est une chaîne de caractères (code ASCII 48)
- $var++; //$var est la chaîne de caractères "1" (code ASCII 49)
- $var = $var + 1; //$var est un entier et vaut 2
- $var += 1.3; //$var est de type "double" et vaut 3.3
- $var = 2 + "8"; //$var est de type "integer" et vaut 10

Fonctions utiles pour les variables

Fonctions	Résultats	Exemples
Gettype($Var)	détermine le type de données de la variable ("integer", "double", "string", "array", "object", "class" ou "unknown type")	$x = 2.3; echo (gettype($x)); // double
Settype($var)	Définit explicitement le type de la variable	$x = 2.3; settype ($x, "integer"); echo $x; // $x=2
isset ($nom_var)	sert à savoir si une variable possède une valeur (true ou false)	$x = 2.3; $val = isset ($x) ; // true

Fonctions utiles pour les variables

Fonctions	Résultats	Exemples
unset ($nom_var)	détruit une variable	unset ($x); /* isset($x) donne false*/
empty ($nom_var)	renvoie true si la variable est vide ou vaut 0, sinon revoie false	$x = 2.3; $vide = empty ($x) ; // false

Fonctions utiles pour les variables

Fonctions	Résultats	Exemples
is_int()*, *is_integer()*, *is_long()	VRAI si le type est entier	$x = 2.3; $entier = is_int ($x); // false
is_double()*, *is_float()*, *is_real()	VRAI si le type est double	$x = 2.3; $double = is_real ($x); // true
is_string()*, *is_array()*, *is_object()	VRAI si le type est string, array ou object	$x = "2.3"; $str = is_string ($x) ; // true

Fonctions utiles pour les variables. Quelques exemples

```
<?php
$a_bool = TRUE;  // un booléen
$a_str  = 'foo';  // une chaîne de caractères
$a_str2 = 'foo';  // une chaîne de caractères
$an_int  = 12;    // un entier

echo gettype($a_bool); // affiche "boolean"
echo gettype($a_str);  // affiche "string"

// Ceci est un entier, on l'incrémente de 4
if (is_int($an_int)) {
   $an_int += 4;
}

// Si $bool est une chaîne, l'afficher
// (Ne pas imprimer n'importe quoi).
if (is_string($a_bool)) {
   echo ‘’Chaîne : ‘’ . $a_bool;
}
?>
```

Les constantes

- Les constantes internes
  - TRUE et FALSE valent 1 et 0
  - PHP_VERSION numéro de version de l'analyseur PHP
    - exemple : echo PHP_VERSION; // 4.0.5
  - PHP_OS donne le système d'exploitation côté serveur
    - exemple : echo PHP_OS; // WINNT
- Déclaration de constantes
  - define ("Formation", "Master BioInfo");
  - define ("PI", 3.14);
  - define ("font", "<FONT size=4 color=#FF0000><B>")
  - defined ("PI") // la constante PI est-elle définie? TRUE
  - defined ("Pi") // FALSE
  - echo ("<HR>" . font . Formation);

Opérateurs et expressions

Opérateurs arithmétiques

```
$op1 = 5;
$op2 = 2;
```

Opérateur	Signification	Exemple	résultat ou valeur $op1
+	addition	`$op1 + $op2`	7
-	soustraction	`$op1 - $op2`	3
*	multiplication	`$op1 * $op2`	10
/	division	`$op1 / $op2`	2.5
%	modulo	`$op1 % $op2`	1
-	négatif	`$op1 = -$op2`	-2

Opérateurs et expressions

Opérateurs de comparaison.

```
$op1 = 5;
$op2 = 2;
```

Opérateur	Signification	Exemple
==	égalité (attention ! il faut deux "égals")	`$op1 == $op2`
<	strictement inférieur	`$op1 < $op2`
>	strictement supérieur	`$op1 > $op2`
<=	inférieur ou égal	`$op1 <= $op2`
>=	supérieur ou égal	`$op1 >= $op2`
!= ou <>	différent	`$op1 != $op2`

Opérateurs et expressions

Utilisation de raccourcis pour les opérateurs.

```
$op1 = 5;
$op2 = 2;
```

Opérateur	Signification	Exemple	Equivalent	Résultat
+=	addition	`$op1 += 7;`	`$optl = $optl + 7;`	12
-=	soustraction	`$op1 -= 7;`	`$optl = $optl - 7;`	-2
*=	multiplication	`$op1 *= 7;`	`$optl = $optl * 7;`	35
/=	division	`$op1 /= 7;`	`$optl = $optl / 7;`	0.7141
%=	modulo	`$op1 %= 7;`	`$optl = $optl % 7;`	5
.=	concaténation	`$op1 .= 7;`	`$optl = $optl."7";`	57

Opérateurs et expressions

- Opérateurs logiques.

opérateurs	ET	OU inclusif	OU exclusif	NON
syntaxe	"&&" ou "and"	"||" ou "or"	xor	!

- Opérateur d'affectation: =

Opérateurs et expressions

- Opérateur de concaténation sur les chaînes de caractères: ” . ”
  - $ a= "Hello";
  - $b=$a . "World! "; //Affiche Hello Word!

- Vous pouvez spécifier une chaîne en l’encadrant entre de simples ou des doubles quottes
  - $ val= ‘Pengo’
  - echo ‘Hello $val’ //affiche Hello $val
  - echo "Hello $val " //affiche Hello Pengo

Opérateurs et expressions

Quelques fonctions de traitement de chaînes de caractères

Fonction	Signification	Exemple
strlen()	renvoie le nombre de caractères contenus dans la chaîne	`$nbcar = strlen($var);`
substr()	extrait un nombre de caractères d'une chaîne à partir d'une position	`$extrait = substr ($var,$pos,$nb);`
trim()	supprime les espaces avant et après la chaîne	`$var = trim($var);`
addslashes()	despécialise les caractères apostrophe ('), double apostrophe (") et le slashe (\) pour les bases de données.	`$varbd = addslashes ($var);`
explode()	renvoie dans un tableau les valeurs comprises entre un séparateur	`$tableau = explode ($delimiteur,$var);`

Structures de contrôles

```
if (expression conditionnelle) {
  instruction(s) à exécuter si la condition est vraie
}
else {
  instruction(s) à exécuter si la condition est fausse
}
```

```
if (!isset($ var)))
{
  echo ‘La variable $var n'a pas ete declaree’;
}
```

Structures de contrôles

if (expression conditionnelle 1) {
  instruction(s) à exécuter si la condition 1 est vraie
}
elseif (expression conditionnelle 2) {
  instruction(s) à exécuter si la condition 2 est vraie
}
else (expression conditionnelle) {
  instruction(s) à exécuter si toutes les valeurs des expressions précédentes sont fausses
}

Structures de contrôles

```
switch ($variable) {
  case valeur1 :
              instruction(s) 1;
              break;
  case valeur2 :
            instruction(s) 2;
             break;
  ...
  case valeurn :
             instruction(s) n;
             break;
  default :
          instruction(s) par défaut si aucune valeur ne correspond à la valeur de la

        variable
        break;
}
```

Structures de contrôles

```
while (expression à valider)
{
  instructions à exécuter;
 }
```

```
do
{
  instructions à exécuter;
}
while (expression à valider);
```

```
for (initialisation; valeur maximale; incrément du compteur)
 {
  instructions;
}
```

Les tableaux

- Type de données très important en PHP
  - Car les fonctions qui retourne plusieurs valeurs le font généralement sous forme de tableaux
    - Cas des fonctions liées aux bases de données

- On distingue deux types de tableaux en PHP:
  - Les tableaux classiques: les indices sont numériques
    - $tab[0]=1;
    - $tab[1]=4;
    - $tab[2]=6;
    - Équivalent à: $tab=array(1,4,6);

Les tableaux

- On distingue deux types de tableaux en PHP:
  - Les tableaux associatifs: les indices sont des chaînes de caractères qui représente des valeurs associées qui sert de clés d'accès.
    - $ville[‘’toulouse’’]=31000;
    - $ville[‘’paris’’]=75000;
    - $ville[‘’marseille’’]=13000;

Les tableaux

- Les tableaux multidimensionnelles
  - Chaque élément est lui-même un tableau : array(array(…), array(…), array(…))
  - Il est ainsi possible de créer des tableaux à 2, 3, 4 … dimensions

- Déclaration
  - $dept_iut[0] [0]= "uf info";
  - $dept_iut[0] [1]= "uf eco";
  - $dept_iut[1] [0]= "uf pv";
  - $dept_iut[1] [1]= "uf pa";
- ou
  - $dept_iut = array (array ("uf info", "uf eco"), array ("uf pv", "uf pa"));

Quelques fonctions utiles pour les tableaux

Fonctions	Résultats	Exemples
count($var)	compte le nombre d'éléments affectés	$dept = array ("des","dpa"); $nb = count($dept); // = 2
current($var)	détermine la valeur de l'élément en cours	$dept[5]="formco"; $val =current($dept); //="formco"
reset($var)	déplace le pointeur sur le 1er élément	$reset ($dept); // = "des"

Quelques fonctions utiles pour les tableaux

Fonctions	Résultats	Exemples

List ($indice,$val)
each ($var_tableau)
.
	permettent de parcourir les éléments même si les indices ne sont pas consécutifs	while (list($indice, $val) =each($dept))
		{ echo ("$indice - $val"); // "0 - des" puis "1 - dpa"
		}

sort($var)	trie les éléments selon un ordre numérique ou alphabétique et réaffecte les indices du tableau	$dept=array ("des","dpa","dbe");
		sort($dept);
		while (list($indice, $val) =each($dept))
		{ echo ("$indice - $val"); // "0 - dbe" puis "1 - des“ // puis "2 - dpa"
		}

Les fonctions

Pour définir une fonction, on utilise le mot clé: **function**

```
Function($val1, …, $valn)
{
  Instructions;
 }
```

```
// fonction de calcul de la superficie d'un disque
function calcul_superficie($rayon)
 {
  $sup = $rayon * $rayon * 3.14159;
  return $sup;    // résulat retourné
}
```

Portée des variables

- Locale
  - Une variable déclarée dans une fonction aura une portée limitée à cette seule fonction.
  - Portée par défaut.
- Globale
  - Variable déclarée tout au début du script, en dehors et avant toute fonction.
  - Afin de distinguer une variable locale qui porterait le même nom qu'une variable globale, il faut la déclarer à l'aide du mot-clé global.
- Variables statiques
  - Implicitement, les variables locales d'une fonction sont réinitialisées à chaque appel conserver leurs valeurs précédentes,
  - Il faut les déclarer par le mot-clé static

Portée des variables

```
<?
$formation=   " Master BioInfo " ;      /* var  connue de toutes les
                                          fonctions.*/

function signer()
{ global $formation;   /* $formation est la variable globale
                          précédente*/
   $titre="Responsable formation";
   $titre = $titre . $formation;  /*Responsable formation Master BioInfo*/
 }
 ?>
```

Portée des variables

```
function nb_visite ($name) {
{
  static $liste_nom = "";
  static $compteur = 0;
  $liste_nom = $liste_nom . "$name - " ;
  $compteur++;
  echo ($liste_nom . $compteur . "<BR>");
 }
```

```
nb_visite("Claude"); // Claude – 1
nb_visite("Jean"); //Claude – Jean - 2
```

Inclusion de fichiers

- Lorsqu'un même script doit être utilisé dans des pages HTML différentes, afin de faciliter sa maintenance, il est conseillé d'écrire ce script dans un fichier texte séparé (suffixé par exemple .PHP).

- PHP possède 2 fonctions qui concernent l'inclusion de fichiers externes :
  - `require("fichier.php") ;`
    - inclut le fichier fichier.php
    - évaluation en pre-processing
  - `include("fichier.php") ;`
    - inclut le fichier fichier.php
    - évaluation à chaque fois

Exercices

1. Ecrire un programme PHP qui somme les éléments d’un tableau sans fonction.
2. Refaire le même programme qu'en 1 mais cette fois-ci en utilisant une fonction.
3. Refaire le même programme qu'en 2 mais cette fois-ci en utilisant une fonction définie dans un fichier externe.

Utilisation des formulaires en PHP

La procédure de soumission d'un formulaire est contrôlée par 2 attributs de la balise <FORM> : METHOD et ACTION.

- L'attribut METHOD détermine la manière dont les données sont transmises au serveur. Il peut prendre les valeurs GET et POST.
  - POST demande à l'explorateur d'empaqueter toutes les données et de les transmettre au serveur. Les données ne sont pas visibles par l'internaute au moment de l'envoi
  - GET, quant à lui, envoie les données en tant que partie intégrale de l'URL pour la page cible. De ce fait, les données sont visibles dans l'URL de la page cible. Elles se présentent ainsi : http://www.ups-tlse.fr/nouv_page.php?lst_valeur=Dupont&txt_nom=azer_

Utilisation des formulaires en PHP

- L’attribut ACTION spécifie la page cible pour les données soumises. Cet attribut n'est pas obligatoire. Dans ce cas, la page en cours est re-exécutée.

Un exemple de formulaire: méthode POST

Quand on clique ici: la page TraiteFormulaire.php est appelée

Le fichier formulaire.html

```
<form method=“post" action="TraiteFormulaire.php">
Votre adresse e-mail<input type="text" name="mail" value="adresse e-mail"
size="30" maxlength="55">
<input type="checkbox" NAME="mailing" value="oui" checked> abonnement
gratuit à notre-planete.info
<br> Description de votre site<br>
<textarea name="description" rows="3" cols="60"></textarea><br>
Type d'échange souhaité :
<select name="demande">
<option selected="selected" value="echange de liens">Echange de
liens</option>
<option value="partenariat">Partenariat</option>
</select>
Dans quelle rubrique souhaitez-vous être présent ?<br>
<input type="radio" name="rubrique"
value="environnement">Environnement / Ecologie</input><br>
<input type="radio" name="rubrique"
value="geographie">Géographie</input><br>
<input type="radio" name="rubrique">Photos</input>
<input type="submit" value="valider">
</form>
```

Récupérer les valeurs d'un formulaire

TraiteFormulaire.php

```
$mail = $_POST["mail"];
echo "votre adresse e-mail est : $mail<br />";
$mailing = (isset($_POST["mailing"])? "oui" : "non");
echo "Avez-vous souhaite vous inscrire sur la lettre d'informations ?
$mailing<br />";
$description = htmlentities($_POST["description"],ENT_QUOTES);
echo "et sa description est la suivante : <br />".nl2br($description)."<br />";
$rubrique = $_POST["rubrique"]; echo "pour la rubrique ".$rubrique;
```

Affichage sur le navigateur

votre adresse e-mail est : toto@netalya.com
Avez-vous souhaité vous inscrire sur la lettre d'informations ? oui
et sa description est la suivante :
Site sur les insectes tropicaux.
Découvrez les plus impressionnants insectes visibles sur notre planète en toute saison : l'été comme l'hiver.
pour la rubrique photos

Récupérer les valeurs d'un formulaire

➔Dans l'exemple précédent:

➔ *htmlentities()* transforme certains caractères du texte entré par l'utilisateur dans la zone de texte en d'autres qui ne permettront pas l'exécution de code HTML

➔ La fonction *nl2br()* permet de conserver les sauts de ligne effectués dans la zone de texte

Mixer avec du Javascirpt

- Vérification côté client
  - Write a script **`checkParam(…)`**
  - Event OnSubmit to add as a form parameter
  - Action is executed iff OnSubmit returns true

```
<form action="...php" method="POST"
  OnSubmit="return(checkParam(.))">
```

Les formulaires: méthode GET

- Les variables ne transitent pas toujours via un formulaire mais bien souvent par l'URL via la méthode GET.
  - Les variables et les valeurs qu'elles prennent sont déclarées directement dans l'URL c'est à dire via la balise de lien HTML <a href="URL">.
  - Les variables sont ensuite exploitables sur la page cible en PHP.
- Déclarer une variable visible dans la page cible
  - Placer à la fin de l'URL un ? suivi d'un couple nom=valeur en séparant les varaibles par un & lorsqu'il y en a plus d'une: *url.php?variable1=valeur1&variable2=valeur2&variablen=valeurn* avec une limite fixée à 255 caractères.
  - Dans le cas d'une seule variable il n'est pas utile de mettre de &,: *url.php?variable1=valeur1*

Les formulaires: méthode GET

```
echo 'Choisissez une action<br>';
echo '1. <a href="dossier.php?page=2&n=15">lire la suite du
dossier</a><br />';
echo '2. <a href="dossier.php?fin=1">terminer la lecture</a>';
```

Dossier.php

```
if (isset($_GET["fin"])) {
  echo 'Vous avez interrompu votre lecture. A bientôt';
  exit();
}
else {
  $page = $_GET["page"];
  $nombrepages = $_GET["n"];
  echo 'Dossier sur les abeilles - page '.$page.' sur '.$nombrepages;
}
```

Les entrées/sorties de fichiers

- **fopen** permet d'ouvrir un fichier en lecture/ecriture/ajout en fin de fichier
  - $fichier=fopen("fich.txt ", " r "); //ouvre en lecture
  - $fichier=fopen(" fich.txt ", " w "); //ouvre en écriture
  - $fichier=fopen(" fich.txt ", " a ") ; //ouvre en mode ajout

- **fgets** permet de lire un certain nombre de caractères dans un fichier ouvert en lecture
  - $chaine=fgets($fichier, 1024); //lit au max 1024 caractères

- **feof** permet de tester si on est arrivé à la fin du fichier
  - If(foef($fichier)) print("fin du fichier atteinte ");

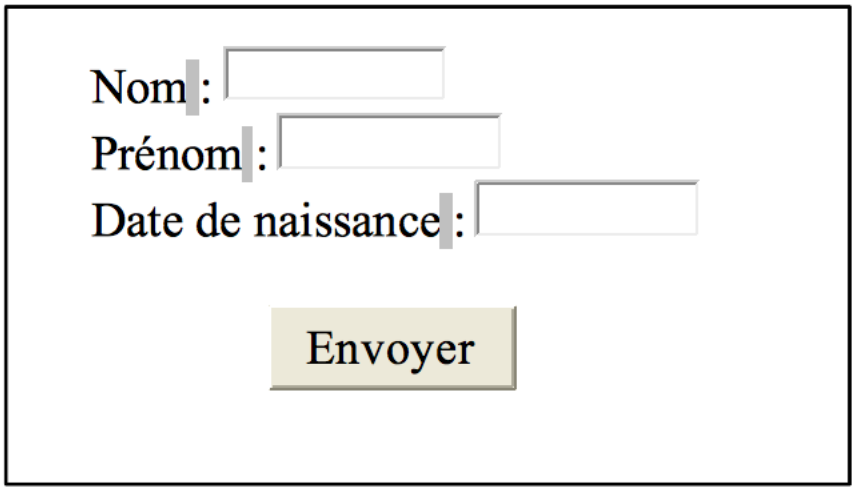
Les entrées/sorties de fichiers

- **fputs** permet d'écrire une chaîne de caractères dans un fichier
  - Fputs($fichier, $chaine);
- **fclose** ferme le fichier
  - Fclose($fichier); //A ne pas oublier
- **file_exists** teste l'existence d'un fichier
  - if (file_exists($fichier)) echo ‘’le fichier existe “;

Gestion du nb de visites : Le code...

```
<?php function calc(){
$file = fopen("counter.txt", "r");          //we read
$counter = fread($file , filesize("counter.txt")) ;
fclose($file) ;
$file = fopen ("counter.txt" , "w");
$counter = $counter + 1 ;                   //we update
fwrite($file , $counter ) ;                 //we write
fclose($file) ;
return $counter; }
  ?></head>
<body>
hello.<br>
You are the <?php  echo calc() ?> visitor. //we display
</body>
```

Ecrire une page HTML affichant le formulaire suivant :



Ecrire un programme php qui récupère les données de ce formulaire, calcule l'âge, vérifie qu'il est raisonnable (compris entre 1 et 120) et affiche une page de ce type :

Si l'âge est correct	Si l'âge est incorrect
Bonjour : **Jacques Durand** Vous êtes né en **1980** Nous sommes en **2015** Vous avez donc environ **35 ans** La page a été visitée **10 fois** depuis sa création	Bonjour : **Jacques Durand** Vous êtes né en **2020** Nous sommes en **2015** Votre âge est incorrect. La page a été visitée **11 fois** depuis sa création

Pour programmer cette page en php, utiliser les fonctions de gestion de fichiers pour gérer le nombre de visiteurs et la fonction date qui permet d'obtenir la date courante.
Par exemple,

- $annee=date("Y") ; //revoie simplement l'année actuelle
- $date_compete=date("d/m/y") ; //revoie une chaîne de caractères de type : 12/12/2005.