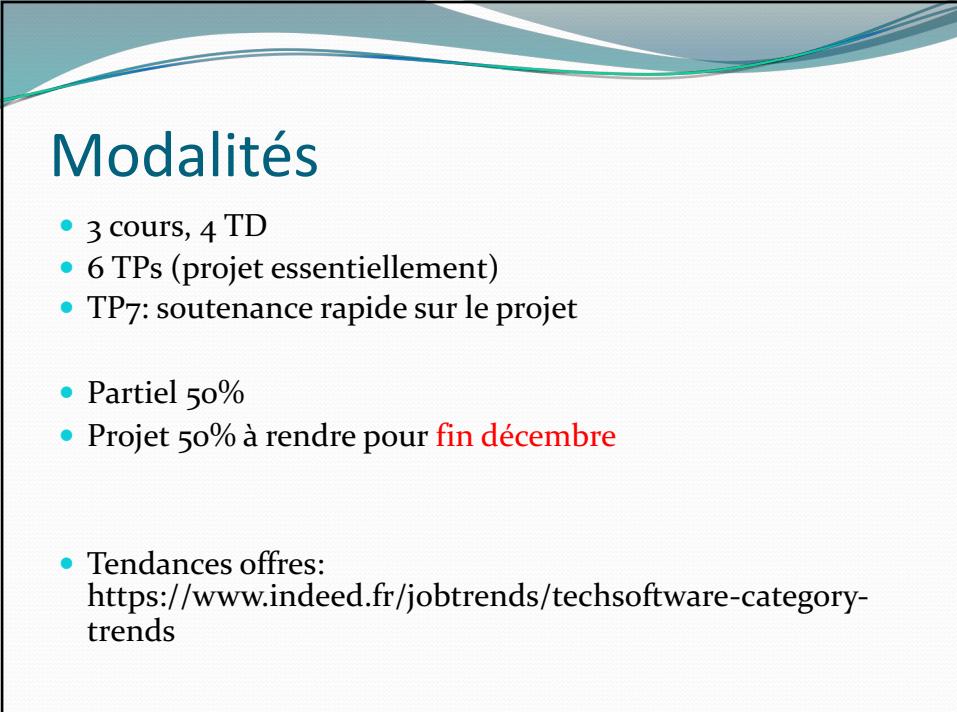




Les CGI Le langage PHP



Modalités

- 3 cours, 4 TD
- 6 TPs (projet essentiellement)
- TP7: soutenance rapide sur le projet
- Partiel 50%
- Projet 50% à rendre pour **fin décembre**
- Tendances offres:
<https://www.indeed.fr/jobtrends/techsoftware-category-trends>

Don't panic !

(Douglas Adams)

Ce qui suit est de la syntaxe, que vous connaissez déjà !

Références

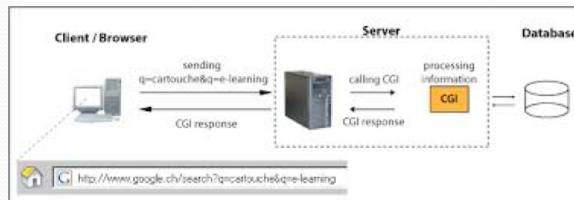
- <http://php.net/docs.php> (doit toujours être ouverte près de vous)
- <https://github.com/vhf/free-programming-books/blob/master/free-programming-books.md#php>
- <http://edu.williamdurand.fr/php-slides>

Introduction aux CGI

CGI (Common Gateway Interface)

Interface permettant d'appeler des exécutables par le protocole http

- Programmes désignés par URL <http://www.hina/cgi-bin/prog.exe>
- Chargement de l'URL provoque l'**exécution** du programme du côté du serveur



Introduction aux CGI

Intérêts:

- Créer des pages html dynamiques (crées à la demande suivant des paramètres (heure, entrées utilisateur, BD, etc.)

Un prog. CGI peut être écrits dans n'importe quel langage:

- soit compilé (C, C++, Ada, Java, VB, ...)
- soit interprété (Perl, Unix sh, tcl, Python, ...) dans ce cas l'interpréteur doit être lancé

Introduction aux CGI

Etape 1: client → serveur

le client charge une URL `http://www.hina/cgi-bin/prog`

- le navigateur construit une commande HTTP GET
Paramètres séparés par un point d'interrogation
`http://www.hina/cgi-bin/prog?var1=toto&var2=sasa`
- Ou le navigateur construit une commande HTTP POST

Paramètres à la suite des **en-têtes HTTP**, dans un champ appelé *corps de la requête*

⇒ quantité de données envoyées n'est plus limitée

⇒ Mode non connecté (pas d'instance sur le serveur) besoin de cookies et de sessions

Introduction aux CGI

Exemple de Requête / Réponse HTTP:

Requête:

```
GET /my/simple/uri?with-query-string HTTP/1.1 Host: example.org Content-Type:  
text/plain; charset=utf-8 Content-Length: 17 This is a content
```

Réponse:

```
HTTP/1.1 200 OK Content-Type: text/html; charset=utf-8 Content-Length: 76  
<!DOCTYPE HTML> <html> <head> </head> <body> <h1>Hello world !</h1>  
</body> </html>
```

Il y a aussi un entête où l'on trouve plein d'informations: cookies, sessions, etc.

- Status HTTP: <http://httpstatus.es/>

4xx: erreur chez le client

5xx: erreur sur le serveur

Introduction aux CGI

- Paramètres données par un formulaire HTML, méthode GET ou POST ou liens, ou javascript

```
<HTML><BODY>
<P><FORM METHOD=POST(ou GET)
ACTION=http://hina/cgi_bin/prog // ou http://hina/cgi_bin/prog?action=toto
<INPUT TYPE=TEXT NAME="nom" SIZE=10 MAXLENGTH=20 VALUE="">
<P><INPUT TYPE=SUBMIT NAME="go" VALUE="Rechercher">
<INPUT TYPE=RESET NAME="reset" VALUE="Reset">
</BODY></HTML>
```

GET should not change anything, POST is for thing that change the server
(<http://stackoverflow.com/questions/1592418/mixing-get-with-post-is-it-a-bad-practice>)

Langage PHP

Syntaxe, utilisation simple

PHP

- Wikipedia:

- PHP was conceived sometime in the fall of 1994 by [Rasmus Lerdorf](#),
- July 13, 2004, PHP 5
- November 2015, PHP7
- January 2013, PHP was installed on more than 240 million [websites](#) (39% of those sampled)

PHP

Installation:

- Linux
 - `sudo apt-get install php5-common libapache2-mod-php7 php7-cli php7-mysql php7-curl`
- Windows
 - Wamp
- Mac
 - Mamp

Scripts PHP

Programme s'exécutant **côté serveur Web**

Généralement du code **embarqué** dans une page HTML entre les balises <?php et ?>

plusieurs morceaux de code PHP <?php ... ?> peuvent être trouvés dans un même fichier

- les fichiers .php sont stockés sur le serveur
- ils sont désignés par une **URL**, ex:<http://hina/page.php>
- le **chargement** de l'URL provoque l'**exécution côté serveur = CGI**

Scripts PHP

Page html contenant un script PHP

```
<HTML> <BODY>
<H1>Table des factorielles</H1>
<?
for ( $i=1,$fact=1 ; $i<4 ; $i++,$fact*=$i )
{
print "$i! = $fact <BR>";
}
?>
</BODY> </HTML>
```

```
<HTML> <BODY>
<H1>Table des factorielles</H1>
1! = 1 <BR>
2! = 2 <BR>
3! = 6 <BR>
</BODY> </HTML>
```



Retourne la page html

Syntaxe de PHP

Proche de C, Java, et de Perl

- commentaires /* ... */ ou // ... ou # ...
- instructions séparées par des points-virgules
- blocs d'instructions entre accolades

Types primitifs

- boolean, integer, float, string,
- array, object
- resource, null

De base, on ne donne pas de typage à une variable \$var, elle est automatiquement convertie sur le type attendu

Syntaxe de PHP

Variables

- précédées de \$, ex: `$compteur`
fonction bool isset()=> variable définie
- une variable peut changer de type, il suffit de lui affecter une nouvelle valeur Ex: `$compteur="premier";`
- une variable non affectée a une valeur par défaut (pas d'erreur de syntaxe)
- conversion de type automatique `$pi="3.14"; $piplusun=$pi+1;`

Syntaxe de PHP

- Chaînes de caractères

- entre guillemets `$couleur="rouge";`
- substitutions de variables à l'intérieur d'une chaîne
`$figure="carré $couleur foncé"; ! "carré rouge foncé"`
- entre **apostrophes** : sans substitution, **ni** encodage
`$figure='carré $couleur foncé'; ->"carré $couleur foncé"`
- longueur d'une chaîne `strlen($figure)`
- comparaison `==`, `$figure == $couleur`
- concaténation « . » `$figure.$couleur`

nombreuses fonctions de manipulation disponibles=> voir doc

- Variables dynamiques = var. dont l'identificateur est la valeur d'une variable

*ex. : \$var="hello"; \$\$var="world";
echo "\$var \$hello \${\$var}"; ! "hello world world"*

Syntaxe de PHP

- Tableaux

Définition:

```
$array = [
    "1" => "bar",
    "2" => "foo",
    "multi" => ["1" => "boo"]
];
```

Ou `$a=array (...)` à la place des crochets

Parcours:

(`for`, `while`, etc.)

```
foreach ($array as $value) {
    var_dump($value); }
```

Syntaxe de PHP

Tableaux

- pas de déclaration préalable
- accès aux membres à l'aide de crochets `$tab[0]="a";`
- ajout d'un élément en fin de tableau `$tab[]="b";`
- taille d'un tableau `count($tab)`
- comparaison `$tab == $tab2`
- nombreuses fonctions de manipulation disponibles

Tableaux associatifs tableaux dont l'**indice** est une **chaîne**

ex. : `$mois["janvier"] = 1; $mois["février"] = 2;`

Resp: S. SALVA

IUT 1ère année

IUT d'aubiere

Syntaxe de PHP

Opérateurs

- Comparaison == != < > <= >=

`$a <=> $b` : Un entier inférieur, égal ou supérieur à zéro lorsque `$a` est respectivement inférieur, égal, ou supérieur à `$b`. Disponible depuis PHP 7.

- Arithmétiques + - * / % ++ --

- Manipulation de bits & | ^ ~ << >> >>>

^ XOR

~ NOT

>> décalage à droite en conservant le signe

>>> décalage à droite en ajoutant des 0

- Affectations = += -= *= /= <<= >>= >>>= &= ~= != ternaire ?:

- Logiques &&, ||, !

Syntaxe de PHP

- **Conditions:**

- Conditions if (*condition*) { ... } else { ... /* facultatif */ }
- if (*cond*) { ... } elseif (*cond*) { ... } else { ... }
- <?php if (*condition*): ?> html code to run if condition is true
<?php else: ?>html code to run if condition is false <?php endif ?>
- (*expr1*) ? (*expr2*) : (*expr3*) est évaluée à *expr2* si *expr1* est évaluée à **TRUE**, et *expr3* si *expr1* est évaluée à **FALSE**.
- switch (*expression*) {

 case *constante* : ... break;

 ...

 default : ...}

Syntaxe de PHP

- **Boucles**

- for (*initialisation* ; *test* ; *increment*) { ... }
- while (*condition*) { ... }
- do { ... } while (*condition*);
- Pour tableaux et Objets:

 foreach (*array_expression* as \$value){ //commandes }

 foreach (*array_expression* as \$key => \$value){

 //commandes }

Clé de l'élément (nom attribut,
nom colonne)

Syntaxe de PHP

Fonctions:

Avant PHP 7 : pas de typage des arguments, ni de la valeur de retour
 PHP7: **possibilité** de typer arguments et retour (voir doc)

*Ex: function factorielle(int \$n) :int { //?int => null est accepté (php7.1)
 if (\$n<2) return 1; else return \$n*factorielle(\$n-1);}*

- Valeurs par défaut possibles pour les arguments

*Ex: function racine(float \$x,int \$degre=2) :float{
 return pow(\$x,1/\$degre); # pow(): fct puissance prédéfinie}*

Invocation : racine(1,...) ou racine(1) \Rightarrow degré=2

Type de retour null => :void

Syntaxe de PHP

Fonctions:

Passage de paramètre par référence
 la fonction travaille **sur les paramètres** d'appel (ici \$x)

*\$x = 10;
 \$y = double(&\$x); \Rightarrow y=20 x=20*

Syntaxe de PHP

- Objets en PHP**

On peut programmer en object en php tout comme tout autre langage (def de private, abstract, final, static, clone, implements,...)

Ex:

classes de connexions aux bases de données, de vérification de formulaire, de configuration de site,

Syntaxe de PHP

- Objet en PHP**

```
<? class ExempleMethodes {
public $propriete1;
private $propriete2;
protected $propriete3;

function __construct(){ //constructeur
    $this->propriete1=1;
    $this->propriete2=2;
    $this->propriete3=3; }

public function methode1(){
    echo 'méthode publique';
    $this->propriete2++; // autorisé car référencé au sein de la classe
    $this->propriete3++; // autorisé car référencé au sein de la classe
    $this->methode2(); // autorisé car référencé au sein de la classe
    $this->methode3(); // autorisé car référencé au sein de la classe }
```

Syntaxe de PHP

• Objet en PHP

```
protected function methode2(){
    echo 'méthode protégée'; }

private function methode3(){
    echo 'méthode privée'; }
}

//utilisation
$obj = new ExempleMethodes();

$obj->methode1();//autorisé
echo $obj->propriete1; // autorisé
echo $obj->propriete2; // non autorisé
echo $obj->propriete3; // non autorisé
$obj->methode2();//non autorisé
$obj->methode3();//non autorisé

?>
```

Syntaxe de PHP

• Objet en PHP

Definition de constantes:

```
const VALUE = 123;
// PHP 5.6+
const SENTENCE = 'The value of VALUE is ' . self::VALUE;
const ARRAY_OF_VALUES = ['a', 'b'];
```

[When to use self vs \\$this?](http://stackoverflow.com/questions/151969/when-to-use-self-vs-this) (<http://stackoverflow.com/questions/151969/when-to-use-self-vs-this>)

⇒ Use \$this to refer to the current object.

⇒ Use self to refer to the current class.

In other words, use \$this->member for non-static members, use self::\$member for static members.

Syntaxe de PHP

Objet en PHP

Méthodes statiques:

```
class Foo
{
    public static $my_static = 'foo';

    public function staticValue() :string {
        return self::$my_static;
    }
}

echo Foo::$my_static . "\n";
```

Syntaxe de PHP

Objet en PHP

Héritage: (<http://php.net/manual/fr/language.oop5.inheritance.php>)

```
class Bar extends Foo
```

Mais aussi des interfaces:

(<http://php.net/manual/fr/language.oop5.interfaces.php>)

interface Footable (puis utilisation par implements),
 (ca vous rappelle quelque chose?)

Syntaxe de PHP

- Namespaces:**

(<http://php.net/manual/fr/language.namespaces.php>)

- But: créer des packages (comme en Java) ou l'on peut avoir plusieurs classes différentes de même nom (prévenir collision de nom)
- namespace Vendor\Model; ou
- namespace MyNamespace { // ... }
- \ est le namespace global
- Inclure un namespace: **use** \Vendor\exempleNamespace
- (à utiliser après avoir un peu d'expérience ☺)

Syntaxe de PHP

- Gestion des exceptions**

PHP gère les exceptions par les classiques *throw*, *try{} catch() {} finally() {}*

Exemple:

```
<? class dvpExempleException {
public function genererException($probleme) {
    if(!is_int($probleme)) {
        throw new Exception ("L'argument -$probleme- n'est pas numérique");
    }
}

$Obj = new dvpExempleException();
try { $Obj->genererException('chaîne au lieu d\'un numérique'); }
catch (Exception $exception)
{ echo $exception->getMessage().'.'.$exception->getLine().'.'.$exception->getFile(); }
?>
```

Syntaxe de PHP

•Gestion des exceptions PHP7

PHP7 gère des **Exceptions** et des **Error**

(<https://trowski.com/2015/06/24/throwable-exceptions-and-errors-in-php7/>)

Error : pour fatal error qui n'était pas récupérable avant:

```
interface Throwable
    |- Exception implements Throwable
        |- ...
    |- Error implements Throwable
        |- TypeError extends Error
        |- ParseError extends Error
        |- ArithmeticError extends Error
            |- DivisionByZeroError extends Error
        |- AssertionException extends Error
```

Resp: S. SALVA

IUT 1ère année

IUT d'aubiere

Syntaxe de PHP

Mais aussi:

- Méthodes magiques (<http://php.net/manual/en/language.oop5.magic.php>):
 - Commencent par __
 - Exemple __construct, __toString()
- Fonction anonymes (lambda)

Resp: S. SALVA

IUT 1ère année

IUT d'aubiere

Syntaxe de PHP

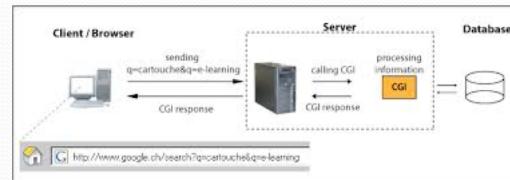
print("bonjour")	affiche bonjour
date("d-m-Y")	donne la date. Il existe plusieurs formats. Ici la fonction donne le jour-le mois-l'année
"."	permet de concaténer deux chaînes de caractères ou chaîne et une variable
strpos(chaine,ch)	retourne la position d'un caractère dans une chaîne si ch existe Autrement retourne ""
strtolower(chaine ch)	retourne une chaîne contenant les caractères de ch en minuscule
substr(chaine ch , debut, fin)	retourne une chaîne contenant les caractères de debut à fin de ch
phpinfo()	retourne de nombreuses informations concernant le serveur
empty()	retourne la valeur FALSE si la variable var est affectée ou bien a une valeur différente de 0
isset()	renvoie TRUE si la variable var est définie
exit()	quitte un processus php
password_hash()	chiffrage
password_verify()	vérification chiffrage

CGI en PHP

Structure générale d'un script PHP

1. Récupération des paramètres en entrées (url, entête http)
2. Vérification des paramètres (isset(), empty(), filter_var(), exp régulières)
3. Exécution de tâches (accès au bases de données, accès à un autre serveur,...)
4. Envoi d'une page HTML vers le navigateur client (page web dynamique)

Récupérer le contenu de champs de formulaires



Au niveau Serveur, en PHP:

Paramètres type GET: `$_GET`
Paramètres type POST: `$_POST`
 Mais aussi une variable globale `$_REQUEST`

Récupérer le contenu de champs de formulaires

```
<?FORM METHOD=POST
ACTION=http://hina/cgi_bin/prog?action="first">
<INPUT TYPE=TEXT NAME="plat">
<INPUT TYPE=TEXT NAME= " description" >
<INPUT TYPE=TEXT NAME= "prix" >
<P><INPUT TYPE=SUBMIT NAME= " act "
VALUE= " Action">
```

```
<?php
echo $_GET['action'] affiche first
echo $_POST['plat']; affiche truffade
```

- La fonction `isset()` permet de vérifier si la variable est déclarée.
- La fonction `empty()` vérifie si la variable est vide

The screenshot shows a Netscape browser window titled "Netscape: formulaire automatique". The menu bar includes File, Edit, View, Go, Communicator, Help, Back, Forward, Reload, Home, Search, Netscape, Print, and Stop. Below the menu is a form with three text input fields: "plat", "description", and "prix", each with a placeholder value. Below the fields is a submit button labeled "Action". The status bar at the bottom shows "100%" and various icons.

Attention, toujours contrôler les champs !

Validation, filtrage

Toujours valider et / ou filter les données entrées:

- De champs de formulaires
- De cookies
- De session, etc.

Utiliser Filter_var (<http://php.net/manual/fr/function.filter-var.php>)

```
mixed filter_var ( mixed $variable [, int $filter = FILTER_DEFAULT [, mixed $options
]] )
```

```
filter_var('bob@example.com', FILTER_VALIDATE_EMAIL)
```

Filtres de validation, de nettoyage (filtre et renvoi un élément nettoyé)

Filtres pour string, boolean, integer, mail, URL, etc, etc (voir doc)

Certains filtres inclues des méthodes PHP connues comme addslashes(),
HTMLEntities(), etc.

Validation, filtrage (à lire plus tard)

Pour faire des filtres avancés, utilisez des expressions régulières:

<http://php.net/manual/en/reference.pcre.pattern.syntax.php>

Rappel sur les expressions régulières:

[] Les crochets définissent une liste de caractères autorisés
 () Les parenthèses définissent un élément composé de l'expression régulière qu'elle contient
 {} Les accolades lorsqu'elles contiennent un ou plusieurs chiffres séparés par des virgules représentent le nombre de fois que l'élément précédent les accolades peut se reproduire
 (par exemple $p\{3,5\}$ correspond à ppp , $pppp$ ou $ppppp$)
 - Un moins entre deux caractères dans une liste représente un intervalle (par exemple $[a-d]$ représente $\{abcd\}$)
 . Le caractère point représente un caractère unique*Le caractère astérisque indique la répétition indéterminée de l'élément le précédent
 ? Le caractère "point d'interrogation indique la présence éventuelle de l'élément le précédent
 | Occurrence de l'élément situé à gauche de cet opérateur ou de celui situé à droite ($lard|cochon$)
 ^ Placé en début d'expression il signifie "chaîne commençant par .. ", Utilisé à l'intérieur d'une liste il signifie "ne contenant pas les caractères suivants..."
 \$ Placé en fin d'expression il signifie "chaîne finissant par .. "

Validation, filtrage (à lire plus tard)

[:classe:]

Les classes de caractères sont celles définies par UNIX.

- [:alnum:]**caractères alphanumériques (équivalent à [A-Za-z0-9])
- [:alpha:]**caractères alphabétiques ([A-Za-z])
- [:blank:]**caractères blanc (espace, tabulation)
- [:ctrl:]**caractères de contrôle (les premiers du code ASCII)
- [:digit:]**chiffre (/0-9/)
- [:graph:]**caractère d'imprimerie (qui fait une marque sur l'écran en quelque sorte)
- [:print:]**caractère imprimable (qui passe à l'imprimante ... tout sauf les caractères de contrôle)
- [:punct:]**caractère de ponctuation
- [:space:]**caractère d'espacement
- [:upper:]**caractère majuscule
- [:xdigit:]**caractère hexadécimal

Validation, filtrage

Pour manipuler des expressions régulières:

Utiliser les méthodes PCRE :
 (méthodes ereg dépréciées depuis PHP7!)

<http://php.net/manual/fr/book.pcre.php>:

preg_match — Perform a regular expression match
preg_filter — Recherche et remplace avec une expression rationnelle
preg_grep — Retourne un tableau avec les résultats de la recherche

```
$email = "test@example.org";
$expression = "/^[_a-z0-9]+(\.[_a-z0-9-]+)*@[a-z0-9-]+\(\.[a-z0-9-]+\)*(\.\[a-z\]{2,3})$/";
if (preg_match($expression, $email)) {
    echo "Email format is correct!";
} else {
    echo "Email format is NOT correct!";
}
```

Utilisation des cookies

- PHP supporte les cookies de manière transparente.
- Les cookies sont un mécanisme d'enregistrement d'informations sur le client, et de lecture de ces informations.
- Ce système permet d'authentifier et de suivre les visiteurs. Vous pouvez envoyer un cookie avec la commande **`setcookie()`**.
- Les Cookies font parties de l'entête HTTP, ce qui impose que **`setcookie()`** soit appelé avant tout affichage sur le client.

Utilisation des cookies

Générer un cookie: `setcookie(nom du cookie, valeur, temps d'expiration)`
 Le temps d'expiration indique le temps de vie du cookie en secondes depuis 1970.

exemple: `setcookie("sasa", "est une vilaine", time() + 365 * 24 * 3600);` crée

Récupérer la valeur d'un cookie:

A chaque fois, PHP recherche si un fichier contenant des cookies pour le site visité existe sur la machine du visiteur.

Pour chaque cookie trouvé, une variable contenant la valeur du cookie est automatiquement instanciée.

Donc, d'après l'exemple précédent, une variable `$_COOKIE["sasa"]`; contenant "est une vilaine" est directement obtenue.

=> Cookies à valider

Utilisation des cookies (à lire plus tard)

- Les cookies peuvent être des tableaux

```
<?php  
setcookie("cookie[three]", "cookiethree" );  
setcookie("cookie[two]", "cookietwo" );  
setcookie("cookie[one]", "cookieone" );  
  
// Après avoir rechargé la page :  
if (isset($_COOKIE['cookie'])) {  
    foreach ($_COOKIE['cookie'] as $name => $value) {  
        echo "$name : $value <br />\n";  
    }  
}  
?>
```

Affichage: three : cookiethree two : cookietwo one : cookieone

Utilisation des cookies

Effacer un cookie

Il suffit d'utiliser setcookie avec une date passée:
`setcookie ("TestCookie", "", time() - 3600);`

Les suivis de sessions

- Protocole HTTP = protocole Internet déconnecté, différent de Telnet, Ftp, ...
- Certaines applications Web (e-commerce : caddie) ont besoin de maintenir une "mémoire" entre deux requêtes
- Concept de *suivi de sessions* => simuler une connection

Les suivis de sessions

Suivi de sessions = Mémoire de ce que fait l'utilisateur d'une page à l'autre

Consiste au transfert de données générées par une requête vers les requêtes suivantes

4 méthodes

- 1) utilisation des cookies (déjà vu)
- 2) réécriture d'URL : passage de paramètres &var=toto&var2=titi&...
- 3) utilisation des champs de formulaire "hidden"
- 4) utilisation des sessions PHP

Les suivis de sessions (à lire plus tard)

Réécriture d'URL

Fonctionne exactement comme un formulaire de type GET, sauf que c'est le programmeur qui crée l'url

•**Principe :**

Ajouter dans l'url appelant un identifiant pour la session, puis des données
`href="http://localhost/prog.php?uid=itey&nom=sasa">`

•**Limitations :**

données volumineuses, caractères autorisés, longueur URL, données visibles (sécurité)

Les suivis de sessions (à lire plus tard)

Utilisation des champs de formulaire "hidden "

Principe :

On cache les données de session dans des champs de formulaire de type "hidden"

Dans ce cas, ils sont présents dans la page html en cours mais ne sont pas visibles

```
<INPUT TYPE="HIDDEN" NAME="uid" VALUE="itey">
<INPUT TYPE="HIDDEN" NAME=" nom" VALUE="sasa">
```

Limitations :

idem la "réécriture d'URL" sauf pour la sécurité
(utilisation de formulaires de type POST)

Les suivis de sessions

Principe :

Un objet "session" peut être associé avec *chaque requête*. Il va servir de "container" pour des informations persistantes

- Démarrer une session ou récupérer une session:

```
session_start(); // pleins d'options voir doc.
```

- Ajouter une variable à la session: Création d'une variable color contenant 'vert'

```
$_SESSION['color'] = 'vert';
```

- Utiliser les variables de sessions (**mais besoin de mieux valider**):

```
If isset($_SESSION['color']) echo $_SESSION['color'];
```

- Retirer une variable d'une session

```
unset($_SESSION['color']);
```

Les suivis de sessions

`session_id`

Lit et/ou modifie l'identifiant courant de session

`session.name string`

Spécifie le nom de la session, qui sera utilisé comme nom de cookie.
Il ne doit contenir que des caractères alphanumérique.
Par défaut, c'est PHPSESSID

Chargement des fichiers, des classes

PHP ne gère pas le chargement de fichiers ou de classes, besoin de le faire à la main avec:

- `Include('fichier.php')` -> inclusion du code de fichier.php
- `require ('fichier.php')` -> idem mais erreur fatale si le fichier n'existe pas
- `require_once('fichier.php')` -> idem vérifie si le fichier a déjà été inclus, et si c'est le cas, ne l'inclut pas une deuxième fois.
- UTILISER `require_once` avec les classes: `require_once('class.twitter.php');`
- Oui c'est lourd, mais il y a aussi l'autoloading (voir plus loin)



Historique: Nouveautés de PHP 5

Principalement :

SQLite: Un SGBD embarqué

SimpleXML: Un nouveau parseur XML très efficace et très simple

Un nouveau modèle POO: Le modèle objet complètement remanié,
l'ancien restant correctement interprété par php

Historique: nouveautés de PHP 7

<http://php.net/manual/fr/migration70.new-features.php>
<http://php.net/manual/fr/migration71.new-features.php>
<http://php.net/manual/fr/migration72.new-features.php>

Principalement :

Performances

Nouveaux opérateurs <=>, ?? (\$a ?? 0-> \$a si non null ou 0)

Amélioration du moteur d'exception Throwable ->Error ->Exception

Import multiple de namespaces

Des anciennes fonctionnalités dépréciées (ereg, mysql_, etc.)

Classes anonymes, etc.

Historique: nouveautés de PHP 7

Scalar Type Declarations

spécifier le type que l'on attend dans la signature d'une fonction ou d'une méthode.

2 modes:

Dans le premier mode les paramètres sont convertis dans le type attendu.

Deuxième mode qui pour être utilisé doit être déclaré au tout début du fichier.

Dans ce mode, les paramètres ne sont pas convertis et un "Fatal Error" est déclenché

Return Types

On peut maintenant spécifier le type de retour d'une fonction ou d'une méthode.

Function myfunction(array \$list, int \$nb): bool

Ou ?int ?bool,void, object, iterator, callable, etc.

trucs et astuces

Faire des redirections en php:

```
<?
header("Location: http://www.site.com");
header("Location: ./page.htm");
header("Location: /pages/page.htm");
?>
```

Attention: rien ne doit être écrit sur le navigateur avant (pas même un saut de ligne)

Redirection par la balise meta (html)

```
<meta http-equiv="Refresh" content="20;URL=page2.html"> appelle page2 au bout de
20 secondes
```

trucs et astuces

Envoyer un mail:

```
mail(<adresse du destinataire>,<titre du mail>,<corps du message>);
Modifier php.ini ?
```

```
<?php
$destinataire = "sebastien.salva@uca.fr";
echo "Ce script envoie un mail à $destinataire";
mail($destinataire, "email", « message");
?>
```