



# ***Technologies Web : Backend***

**Thomas Bourdeaud'huy**

**Mai 2018**



# Plan

- Installation des serveurs apache2 & mysql
- Éléments de php
- Protocole HTTP, redirections, cookies, sessions
  - Cas d'étude sécurisation
- Architecture MVC
  - Framework « TinyMVC »
  - Cas d'étude chat



# Présentation



- 1994 PHP/FI (Rasmus Lerdorf)
  - pour “Personal Home Page Tools/Form Interpreter”
  - Acronyme récursif “Php Hypertext Processor”
- PHP 4.0 : Intégration du Moteur Zend
  - Du nom de l’entreprise qui dirige le projet
- PHP 5 : Nouveau moteur Zend2
  - Support objet complet
- PHP 7 : déc. 2015
- Langage de script spécialisé dans la génération de code HTML
- Nombreuses bibliothèques spécialisées : images, BDD, PDF ...
- Hérite de spécificités syntaxiques et sémantiques du langage C






# Vérifiez votre Serveur !

## test.php

- (W/M)AMP / Easyphp
  - Windows/Mac Apache Mysql Php
- Apache2 sous Linux
- Produire une erreur dans [test.php](#), vérifier qu'elle s'affiche
- Si pb : modifier le fichier de configuration : [php.ini](#)
  - Directives **display\_errors = On** et **error\_reporting = E\_ALL**
  - NB : **phpinfo()** affiche le chemin du fichier php.ini



A stylized logo for PHP (PHP: Hypertext Preprocessor) is located on the left side of the slide. It features a dark purple silhouette of a person in a dynamic, jumping pose. The figure is positioned over two overlapping circles: a pink one on top and a blue one on the bottom. The circles have a watercolor-like texture.

# ***Quelques éléments de php***

# Double interprétation

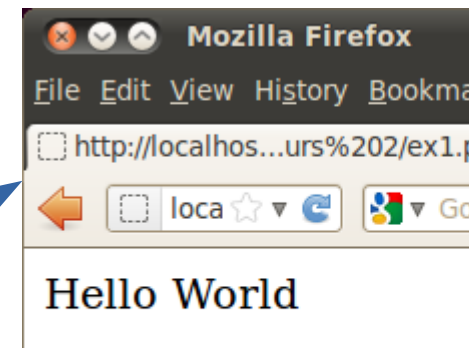
- Le php s'insère directement dans le source d'une page HTML, entre les balises `<?php` et `?>`
- Les pages HTML vues par le client sont le résultat d'une **génération dynamique** par le moteur PHP
- Seul le code PHP est substitué par le moteur PHP

```
ex1_hello.php x
<?xml version="1.0" encoding="UTF-8" ?
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
<html xmlns="http://www.w3.org/1999/xhtml"
<!-- **** H E A D **** -->
<head>
  <meta http-equiv="Content-Type" content="text/html" />
</head>
<!-- **** F I N **** H E A D **** -->
<!-- **** B O D Y **** -->
<body>
  <?php
    echo "Hello\nWorld";
  ?>
</body>
</html>
```

apache

```
6 <head>
7   <meta http-equiv="Content-Type" content="text/html" />
8 </head>
9 <!-- **** F I N **** H E A D **** -->
10
11
12 <!-- **** B O D Y **** -->
13 <body>
14
15 Hello
16 World
17 </body>
18 <!-- **** F I N **** B O D Y **** -->
19
20 </html>
```

firefox





# \$Variables en php

## elements.php

- La meilleure référence : <http://php.net/manual/fr>

```
<html><body> ...
```

```
<?php
```

```
// intégré dans le html; sera interprété par le serveur
```


```
$uneVariable="sa valeur"; // non typée
```

```
echo "Valeur de cette variable : " . $uneVariable;
```

```
?>
```

```
... </body></html>
```





# Tableaux en php

## elements.php

```
<?php
```

```
$unTableau = array(1,2,"trois");    // hétérogène
```

```
$unTableau[] = 4;  // insertion sans indice !
```

```
$tabAssociatif = array("cle"=>"valeur", "cle2"=>2);
```

```
$tabAssociatif["cle3"] = "encore";
```

- **Parcours :**

```
for($i=0;$i<count($unTableau);$i++) echo $unTableau[$i];
```

```
foreach ($unTableau as $nextVal) ...
```

```
foreach ($tabAssociatif as $nextCle => $nextVal) ...
```







# Modularité

## elements.php

Déclaration d'une fonction: `function foo() {...}`

- Valeur par défaut des arguments optionnels : “=”

```
function puissance($x,$exposant=2) {  
    return pow($x, $exposant) ;  
}
```

```
$a =puissance(2);           //4
```

```
$b =puissance(2, 3);        //8
```

Charger une librairie : `include(“chemin_de_la_librairie.php”)`






# ***Le protocole HTTP***



# Notion de Resource : URL


## RFC 3986

- Uniform Resource Locator « Adresse Réticulaire »
  - Ressource : ***plus petite unité d'information adressable***
- Permet au client d'identifier :
  - le **type de protocole** à utiliser
  - l'**adresse précise de la ressource**
- Format : **<protocole>://<machine.domaine>/<chemin/ressource>#<a>?<qs>**
  - RFC 3986 <http://www.w3.org/Addressing/>
  - Insensible à la casse
  - Encodage hexadécimal possible , recommandé pour # % < > @ & ? : / { } ( ) | \ ^ ~ [ ] ` et espace
    - Cf. fonction php **urlencode()**
  - IRI permet d'utiliser tout unicode (RFC 3987)



# Chemins Absolus/Relatifs

## chemins.html



*Bannir le  
double-clic*

- Ne pas confondre
  - URL (<http://>)
    - Le navigateur va effectuer une connexion réseau
  - Chemin local (<c:\>)
    - Le navigateur lit un fichier présent sur le disque dur de l'utilisateur
- Ne pas confondre
  - URL ABSOLUE (<http://...>)
    - Permet de se connecter à un serveur différent
    - **Faute de goût** si on reste sur le même serveur
  - URL RELATIVE (<rep/fichier>) - **À privilégier**
    - Induit une recherche de ressource sur le même serveur



# Protocole HTTP 1.1

## RFC 2616 / 7230

① Demande de connexion Tcp (Port 80)



② Connexion acceptée



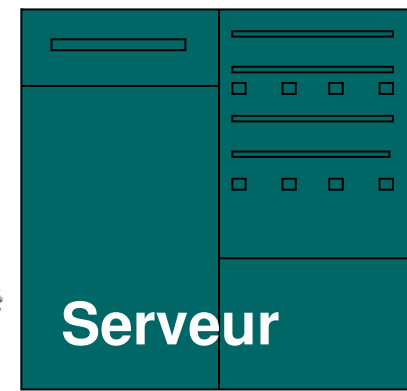
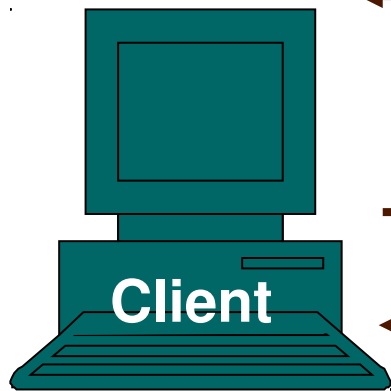
③ Requête HTTP



④ Envoi d'un page html



Fermeture de la connexion



**NB : Il n'y a pas de session permanente entre le client et le serveur !**

**\*.html → aucune modification**  
**\*.php \*.cgi, \*.asp ...**  
**→ interprétation préalable**





# Entêtes de requête

→ Le navigateur envoie plus qu'une simple URL

- Une ligne de requête: comprend trois éléments devant être séparés par un espace :
  - La **méthode**
  - La **ressource**
  - La **version du protocole** utilisée par le client (généralement HTTP/1.0)
- Les champs d'en-tête de la requête: lignes composées du nom de l'en-tête, deux points (:) et de la valeur de l'en-tête
  - Ligne vide
  - Le corps de la requête (cas POST)



# Entêtes de réponse

→ Ce qui est affiché par le navigateur n'est pas tout ce qui a été reçu

- Une ligne de statut :
  - La **version du protocole** utilisé
  - Le **code de statut**
  - La **signification du code**
- Les champs d'en-tête de la réponse : lignes composées du nom de l'en-tête, deux points (:) et la valeur de l'en-tête
- Ligne vide
- Le corps de la réponse:
  - Document demandé codé en (X)HTML
  - Autre chose



# Exemples http.php

- Envoi d'une requête vers <http.php>
- Envoi d'une requête vers <http.php?excel>
  - Le serveur renvoie une entête **Content-Type** différente
- Envoi d'une requête avec entête **Accept-Encoding:gzip**
  - Le serveur renvoie une entête **Content-Encoding**
- Inspecter les interactions entre votre navigateur et des serveurs :
  - Console de développement / onglet “Réseau”
  - **telnet** sur le port 80 sous Linux







# Lire/Envoyer des entêtes en php : entetes.html

- Lire :
  - Tableau associatif `$_SERVER`
- Ecrire :
  - Fonction `header()`
    - A écrire AVANT tout HTML !
    - Sinon : « Erreur : *headers already sent* »... KEZAKO?
- NB : Bufferisation possible
  - Configuration dans `php.ini` : `output_buffering`





# Formulaires

*Chaîne de requête :  
« querystring »*

- Un moyen d'envoyer des informations au serveur
  - 1) Ces informations sont collectées par le navigateur
    - Chaque information porte un nom : attribut **"name"**
    - Et une valeur : définie par l'utilisateur ou prédéfinie : attribut **"value"**
  - 2) Le navigateur prépare la **chaîne de requête** :
    - Format : **cle1=valeur1&cle2=valeur2**
  - 3) Il l'envoie au serveur à l'aide d'une requête HTTP
    - L'adresse de la page destination est précisée dans l'attribut **"action"**
    - Le type de requête est précisé dans l'attribut **"method"** : GET ou POST





# Exemple : identification.html

```
<form action="identification.php" method="GET" />  
<label for="pseudo">Pseudo : </label>  
<input type="text" id="pseudo" name="login" />  
<br />  
<label for="mdp">Mdp :</label>  
<input type="password" id="mdp" name="passe" />  
<br />  
<input type="submit" value="Envoyer" />  
</form>
```

*Tester POST*





# Tous les champs

## forms/form2.html

```
<input type="" name ="nom" />
```

type= radio | checkbox | text | password | image | reset | submit | hidden | file

```
<select name = nom2>
```

```
    <option value = "val1">
```

```
    <option value = "valn">
```

```
</select>
```

```
<texarea name = nom3 rows = nb_lignes cols = nb_colonnes>
```

texte par défaut de la zone

```
</textarea>
```



# Envoyer des données par GET ou POST en HTTP

```
telnet moodle.ec-lille.fr 80
Trying 193.48.25.144...
Connected to moodle.ec-lille.fr.
Escape character is '^['.
```

*Les données sont dans l'URL*

**GET** /index.php?champ\_texte=12345... HTTP/1.0

...

```
User-Agent : Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:19.0)
Gecko/20100101 Firefox/19.0
```

*ligne vide = fin des entêtes HTTP de requête*

**POST** /index.php HTTP/1.0

...

```
User-Agent : Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:19.0)
Gecko/20100101 Firefox/19.0
Content-type: application/x-www-form-urlencoded
Content-length: 17
```

*ligne vide = fin des entêtes HTTP de requête*

champ\_texte=12345...

*Les données sont dans le corps de la requête, après les entêtes*



# Récupérer les données des formulaires en php

- Tableaux `$_GET`, `$_POST`
  - Tableaux superglobaux : disponibles dans toutes les portées :
  - [https://www.w3schools.com/php/php\\_superglobals.asp](https://www.w3schools.com/php/php_superglobals.asp)
- `$_REQUEST`
  - un mix de `$_POST`, `$_GET` et `$_COOKIES`





# Never Trust User Input

- L'utilisateur oublie d'entrer une valeur, de cocher un champ, il s'amuse avec la chaîne de requête... Attention !
- Solution : **tester l'existence** des cases manipulées

```
if (isset($_GET["cle"]) && !empty($_GET["cle"])) {  
    // On peut se servir de $_GET["cle"]  
} else {  
    // Il faut définir soi-même la valeur ou traiter l'erreur  
}
```

- Cf. `valider(nom,[Source])` dans `maLibUtils.php`





# GET ou POST ?

- GET

- Variables visibles dans l'URL
- Limitation de la taille (dépend de l'OS )
- Simplicité d'extraction des données
- Persistance des données (notamment dans les favoris)
- Données dans le fichier log

- POST

- les variables ne sont pas visibles dans l'URL
- les variables sont envoyées dans le corps de la requête HTTP (après les entêtes)
- Pas de limitation de la taille
- Possibilité d'upload (attribut de form : `enctype=multipart/form-data`)







# GET ou POST ?

- Pour une meilleure performance, opter pour des champs courts transmis via GET
  - Permet aussi un débogage plus simple !
  - GET est plus simple à l'utilisation surtout dans le cas de langages non écrits pour le WEB
- Si les champs sont nombreux ou longs, opter pour POST
  - Pour la sécurité, utiliser POST (données non capturables directement dans l'URL)



# Principales Méthodes HTTP

## ... vers REST - RFC 2616

- GET : récupérer une ressource
- HEAD : récupérer uniquement les méta-informations associées à une ressource, pas la ressource elle-même
- POST : annoter une ressource
- PUT : créer une ressource
- DELETE : demander à supprimer la ressource
- PUT ou POST in REST ?  
<http://stackoverflow.com/questions/630453/put-vs-post-in-rest>





# Redirections

- Entête de réponse HTTP : "Location:<url>"
- Déclenche une réaction de la part du navigateur :
  - Il va émettre une nouvelle requête vers l'url indiquée
  - Et donc afficher une autre page
    - l'entête **REFERER** contiendra l'adresse de la page initiale





# Redirections : en php

## redirection\_form.php

- Fonction `header("Location:<url>")`
- Attention :
  - `header` n'arrête pas l'interprétation du script !
  - A vous de le faire avec `die("msg")`
- Renvoyer un message à l'occasion d'une redirection : chaîne de requête GET !
  - Utiliser `urlencode()`





# Cookies - RFC 2109

“Etat persistant de données coté client”

- Fichier texte créé lors de la première visite du client
- Envoyé **à l'initiative du SERVEUR**
- Renvoyé par le client à chaque nouvelle requête vers cette destination, tant que la date d'expiration n'est pas échue
- Entête de réponse HTTP : **Set-Cookie**
  - Spécifie le domaine, un sous-chemin éventuel, la date d'expiration
- Entête de requête HTTP : **Cookie**
  - Quelques contraintes : nombre total limité à 300 ; taille maximale 4 ko ; au maximum 20 cookies par domaine




# Cookies : en php

## cookies.php

- Envoyer un cookie
  - fonction `setcookie(nom,valeur,[timestamp_expiration] ...)`
  - Sans date d'expiration, un cookie est supprimé dès la fermeture du navigateur
    - à écrire AVANT tout HTML :
      - Pourquoi ? Envoie un cookie d'identification !
      - Sinon ? « *Error : headers already sent* »
- Récupérer des cookies envoyés par client
  - `$_COOKIE["nomCookie"]`





# Cas d'étude

## "rester connecté"

- Améliorer le formulaire d'identification pour permettre à l'utilisateur de rester connecté, même plusieurs jours après son identification
- Si le client a coché la case "rester connecté", on lui envoie un cookie avec ses identifiants
  - Le navigateur renverra les identifiants du client tant qu'il possèdera le cookie
  - Le serveur s'en servira pour l'identifier
- Comment faire du **retargeting** ?





# Cookies & Sessions

## session.php

- Les cookies permettent au serveur de se souvenir du client, pour lui attribuer certaines ressources
- Par exemple, un espace de stockage de données spécifique : les variables de session
- Création d'une session : `session_start()`
- à écrire AVANT tout HTML
  - Pourquoi ? Envoie un cookie d'identification !
  - Sinon ? « **Error : headers already sent** »
- Une fois la session créée, on manipule le tableau `$_SESSION` en lecture/écriture







# Vie des sessions

- En l'absence de requête de la part du client pendant **24 minutes** (cf. `php.ini` : `session.gc_maxlifetime`), le serveur oublie l'identifiant de session du client
  - Le client devra se reconnecter
- Les cookies d'identification de session n'ont **pas de date d'expiration**, ils sont donc stockés dans **l'environnement du navigateur** et pas sur le disque dur du client
  - Si le navigateur est fermé, le cookie disparaît, le client devra se reconnecter



# Vie des sessions

- Les cookies sont partagés entre ***toutes les fenêtres du navigateur !***
  - => Pour se connecter avec deux utilisateurs différents, utiliser deux navigateurs différents
- Le serveur peut supprimer les données de session à tout moment
  - Utilisation de `session_destroy()`





# Cas d'étude "sécurisation"

- Tester les identifiants d'un utilisateur
- Rediriger vers la page de menu en cas de succès
  - Après avoir sauvegardé une variable de SESSION attestant que l'utilisateur est parvenu à s'identifier
- Rediriger vers la page de connexion en cas d'échec
  - Eviter aussi les intrus qui arrivent directement dans la page de menu
- Offrir un menu de déconnexion manuelle





# ***Php avancé***

- **Templates**
- **Bases de données**



# Retour sur les librairies

- `include("chemin_de_la_librairie.php")`
  - `include_once("chemin")` : ne sera inclus qu'une fois
- Le code de ces fichiers est **recopié** à l'endroit où la fonction est appelée
  - Il est aussi **exécuté**
  - Les fonctions qu'il définit deviennent disponibles !
- Très pratique pour la production de templates, l'internationalisation de sites...





# Exemple : “templates”

## templates.php

- Un fichier contient des définitions du vocabulaire dans la langue choisie, sous forme de tableau associatif :  
[dictionnaire.php](#)
- Des templates utilisent ces mots de vocabulaire dans une structure XHTML
  - [presentation\\_fr.php](#)
  - [presentation\\_en.php](#)
- Le fichier principal inclut le vocabulaire et la structure :  
[templates.php](#)





# Créer une base de données

- Serveur de Bdd : **mysql**
- Utilisation de **phpMyAdmin**
  - Interface Web de gestion
- Fonctionnalités :
  - Import/export
  - Test de requêtes
  - Sécurisation





# Se connecter à mysql en php

## Ancienne méthode...

- Connection au serveur

```
mysql_connect("localhost", "root", "");
```

- Sélection de la base de données

```
mysql_select_db("mabase");
```

- Exécution d'une requête

```
$res = mysql_query("SELECT login FROM ... ");
```

- Parcours des résultats

```
while ($enregistrement = mysql_fetch_array($res)) {  
    echo $enregistrement["login"];  
}
```





# Librairie maLibSQL.pdo.php

- Fichier `config.php`
  - Le **SEUL** fichier où l'on paramètre l'accès à la base de données
- `SQLUpdate($sql)`
  - Renvoie le nb d'enregistrements affectés
- `SQLInsert($sql)`
  - Renvoie l'identifiant (clé primaire/numAuto) de l'enregistrement inséré
- `SQLGetChamp($sql)`
  - Renvoie directement la valeur du champ recherché
- `SQLSelect($sql)`
  - Renvoie faux si aucun résultat



# parcoursRs(\$result)

## bdd.php

- [maLibSQL.pdo.php](#)
  - Fonction **parcoursRs()**
  - Transformation  
*ressource mySQL*  
*<=> tableau associatif*

```
$sql= "select login from... ";  
$res = SQLSelect($sql);  
if ($res)  
{  
    $tab = parcoursRs($res);  
    foreach ($tab as $enrg)  
    {  
        echo $enrg["login"];  
    }  
}
```





# Quelques mots sur ...

## PDO : Php Data Objects

- Extension définissant l'interface pour accéder à une base de données depuis php
- Orientée Objet
- Nombreux pilotes de SGBD pour PDO
  - Facilite la migration d'une application vers un autre SGBD
- Requêtes préparées pour éviter les injections SQL





# Quelques mots sur ... Injections SQL

- **NEVER TRUST USER INPUT**
- Lutter contre les injections SQL
  - Utiliser **addslashes** : protège les caractères spéciaux à l'aide de '\'
  - **maLibUtils.php** : **proteger(\$str)**
  - Nécessite d'encadrer TOUS les champs SQL provenant de l'utilisateur par des apostrophes





# Un thème de CLOK ?

## Sécurité du Web

- La fonction `protéger()` est insuffisante...
- A vous de le prouver !





# Quelques mots sur ...

## CRUD

- Les 4 opérations de base pour la gestion d'une base de données :
- Create : créer (INSERT INTO ...)
- Read : lire (SELECT ...)
- Update : mettre à jour (UPDATE ...)
- Delete : supprimer (DELETE ...)





# ***Vers un Framework MVC***

**TinyMVC**



# Organisation Générale

- Principe de séparation des responsabilités
- Découpage **technique** pour la réalisation des interfaces
  - Un même besoin **fonctionnel** donne lieu au développement dans **chacune** des couches
  - On **contraint** le développeur pour garantir la qualité du code
- M – Modèle
- V – Vues
- C – Contrôleur







# TinyMVC

- Trois fichiers php et des templates génèrent des morceaux d'interface web (éléments de menu, formulaires, etc.)
- Utilise systématiquement des requêtes de type GET pour faciliter le débogage des pages
- Pourquoi réécrire un framework MVC ?
  - zend, cakephp, symfony2 ...
  - On ne fait pas de la magie !



# Couche Modèle

- Des fonctions « métier » d'accès à la base de données, en lecture/écriture
  - Permet une abstraction de la base de données
    - Vers les ORM...



# TinyMVC : couche Modèle

- Fichier `modele.php`
  - Inclut `maLibSQL.pdo.php` (qui inclut `config.php`)
- Les fonctions de lecture renvoient des tableaux associatifs
  - Appels à `parcoursRs()`





# Vues

- Les vues représentent l'IHM de l'application
- Elles utilisent des appels aux fonctions de la couche modèle pour récupérer les données à afficher
- Elles invoquent le contrôleur pour réaliser les traitements





# TinyMVC : Vues

- Fichier [index.php](#)
  - Inclut [modele.php](#)
- Un paramètre “**view**” devra être systématiquement passé en paramètre
  - Selon les valeurs du paramètre view, il faudra parfois également passer d'autres paramètres.
- Inclut le template correspondant au paramètre “**view**” reçu
- Tous les formulaires présents dans les templates doivent renvoyer leurs données vers le contrôleur
  - en lui fournissant un paramètre “**action**”






# Contrôleur

- Page destinataire des soumissions de formulaire
  - Elle exécute des appels aux fonctions de la page modele.php pour mettre à jour la base de données
  - Elle choisit la vue à afficher en retour par le navigateur



# TinyMVC : contrôleur

- Fichier `controleur.php`
  - Inclut `modele.php`
- Teste la valeur d'un paramètre “**action**” qui caractérise le traitement à réaliser
  - Appelle des fonctions de `modele.php` pour réaliser ce traitement
- Redirige vers `index.php` en indiquant la vue à afficher (paramètre “**view**”)



# Cas d'étude : "messagerie instantanée"

- Des utilisateurs peuvent se connecter
  - Le serveur enregistre leur identité en variables de session
- Une fois connectés, ils ont accès à la liste des conversations
- Une fois la conversation choisie, ils accèdent aux messages de la conversation
  - Postés par des utilisateurs non blacklistés
- Ils peuvent poster de nouveaux messages
  - Sauf si la conversation est archivée







# Faibles XSS

- Injection de javascript dans les bases de données
- Faible de sécurité :
  - Le javascript peut insérer de nouveaux éléments non désirés dans la page
  - Il peut servir à “écouter” les événements déclenchés par l'utilisateur dans la page
    - e.g. appuis sur les touches du clavier lors de la saisie d'un mot de passe
- On s'en prémunit avec **htmlspecialchars()**





# Limites du chat 1.0

- Comment afficher à l'utilisateur les nouveaux messages régulièrement ?
  - Recharger la page
- Sans lui faire perdre ce qu'il était en train d'écrire ?
  - Utiliser des requêtes asynchrones : Web 2.0





# ***Annexes***



## <select multiple>

- Plusieurs valeurs associées au même nom
  - `nom=val1&nom=val2`
  - Le serveur ne verra que le dernier...
- Utiliser un nom comportant des crochets : `[]`
  - `<select multiple name="nom[]" > ...`
- Tester avec `is_array`





# Fichiers

- Lire un fichier et récupérer son contenu dans un tableau de lignes :
  - `file($chemin)`
- Opérations élémentaires
  - `file_put_contents`, `file_get_contents`
- Ouvrir des fichiers distants :
  - `fopen($url)`, `get_meta_tags($url)`
- Cf. <http://php.net/manual/fr>





# Variables globales

- En php une variable non déclarée dans une fonction est considérée **locale**
  - Contrairement au javascript !
- Pour utiliser une variable globale dans une fonction, il faut la déclarer en utilisant **global**
- Cf. [maLibSQL.pdo.php](#)
- ```
function SQLSelect($sql)
{
    global $BDD_host;
    global $BDD_base;
    global $BDD_user;
    global $BDD_password;
    ...
}
```






# Passage par référence

```
function foo(&$var)
{
    $var++;
}
```

- `$a=5;`
- `foo ($a);` // \$a vaut maintenant 6





# Exécuter une commande shell en php - shell.php

- `string shell_exec ( string $cmd )`
- `string exec ( string $command [, array &$output [, int &$return_var ]] )`







***Code  
Couleur***



# Culturel/Approfondissement

- A ne pas connaître intégralement par coeur
- Donc, le reste... est à maîtriser parfaitement !
- Pour anticiper les problématiques que vous rencontrerez en stage ou dans d'autres cours
- Pour avoir de la conversation à table ou en soirée...





# Exemple ou Exercice

- Brancher le cerveau et le navigateur
- Expérimenter en prenant le temps...



# Clock

- Centrale Lille “Open Knowledge”
- Des thèmes de veille à approfondir...



# ***Installer & Configurer apache2 (sous Linux)***

- **Paquets**
  - **Configuration**
- **Fichiers Journaux**



# Paquets

- apache2
- libapache2-mod-php
  - Handler /etc/apache2/mods-enabled
- (mysql-server, phpmyadmin)
  - Bases de données





# apache2

- `apt-get install apache2`
- Vérification par le navigateur : `http://localhost`
  - It works !
- Répertoire de publication par défaut ?
  - `/var/www/html`
- Configuration ?
  - `/etc/apache2`





# Premiers tests

- Créer un répertoire dans `/var/www/html`
- Un fichier `hello.html`







# Répertoire virtuel

- Préparation de la configuration dans `sites-available`
- Activation de la configuration par un lien dans `sites-enabled`
- À la main : liens symboliques avec `ln -s`
- Avec les commandes `a2ensite` / `a2dissite`





# Répertoire Virtuel

Alias /repertoire "/repertoire/de/ton/choix"

<Directory "/repertoire/de/ton/choix">

Options Indexes

Require all granted

</Directory>





# Exemple

- Créer un répertoire virtuel sur le Bureau
  - ex : `/home/pi/Desktop/www`
- Configurer, activer et redémarrer le serveur
  - `vi /etc/apache2/sites-available/010-virt.conf`
  - `a2ensite 010-virt`
  - `systemctl reload apache2`
- Tester