HELHa
Haute École Louvain en Hainaut

Syllabus

Catégorie Économique

Informatique de gestion

1

ECIG1B10IGDE2b
Technologie Internet
1IG

Année académique 2014 - 2015

2

PHP- formulaires

1. Transmettre des données de page en page
   - 1.1 Transmettre des données par l'URL
   - 1.2 Transmettre des données par les formulaires
2. Les champs d'un formulaire
   - 2.1 les zones de texte
   - 2.2 les grandes zones de texte
   - 2.3 les cases à cocher (et groupes)
   - 2.4 les boutons d'option
   - 2.5 les champs cachés
   - 2.6 les listes déroulantes
3. Boutons d'envoi multiples
4. Réinitialisation d'un formulaire
5. Exemple de formulaire
6. Récupération des données d'un formulaire
7. Gestion des boutons multiples
8. les failles de sécurité
9. codage des formulaires

PHP- formulaires

1. Transmettre des données de page en page

Un des aspects intéressants de PHP, c'est qu'on peut se transmettre des variables de page en page.

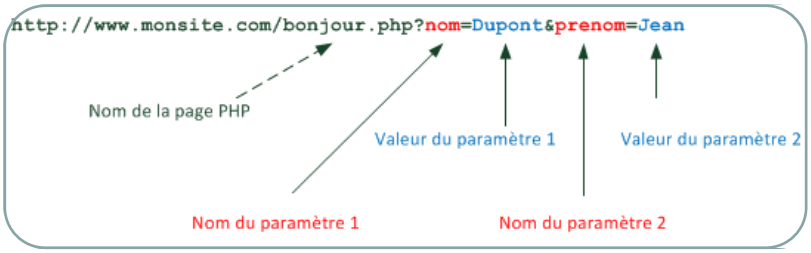
C'est pratique, par exemple pour transmettre le nom du visiteur.

En effet, les variables sont détruites une fois que la page PHP est générée. Alors comment récupérer leur valeur dans une autre page ?

PHP- formulaires

1.1 Transmettre des données par l'URL

URL = Uniform Resource Locator



Le point d'interrogation sépare le nom de la page PHP des paramètres. Ensuite, ces derniers s'enchaînent selon la forme nom=valeur et sont séparés les uns des autres par le symbole &.

La seule limite est la longueur de l'URL. En général il n'est pas conseillé de dépasser les 256 caractères.

PHP- formulaires

1.1 Transmettre des données par l'URL

On insère donc dans "index.php" la ligne suivante

```
1 <a href="bonjour.php?nom=Dupont&amp;prenom=Jean">Dis-moi bonjour !</a>
```

Rem : le "&" est remplacé par "&" (validation W3C)
Tous les & seront transformés en symboles & par le navigateur du visiteur.

Le **World Wide Web Consortium**, abrégé par le sigle **W3C**, est un organisme de normalisation chargé de promouvoir la compatibilité des technologies du World Wide Web telles que HTML

PHP- formulaires

1.1 Transmettre des données par l'URL

Nous allons récupérer les paramètres en PHP dans la page " bonjour.php " par l'intermédiaire du tableau associatif $_GET.

Nom	Valeur
$_GET['nom']	Dupont
$_GET['prenom']	Jean

```php
<p>Bonjour <?php echo $_GET['prenom'] . ' ' . $_GET['nom']; ?> !</p>
```

Rem : les visiteurs peuvent trafiquer les URL.
On ne peut donc avoir confiance dans les paramètres passés.

PHP- formulaires

1.1 Transmettre des données par l'URL

Il faut donc contrôler ces paramètres.

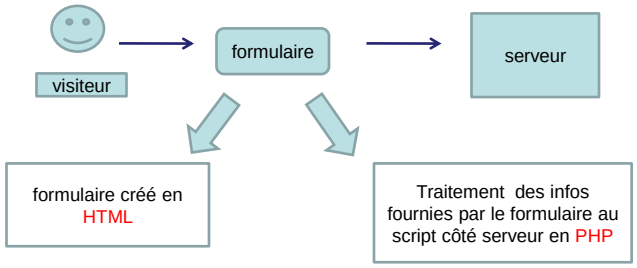
- La fonction isset() permet de vérifier si une variable est définie ou non.
- Le transtypage est une technique qui permet de convertir une variable dans le type de données souhaité. Cela permet de s'assurer par exemple qu'une variable est bien un int (nombre entier).

http://localhost/tests/bonjour.php?nom=Dupont&prenom=Jean&repeter=8

```php
<?php
if (isset($_GET['prenom']) AND isset($_GET['nom']) AND isset($_GET['repeter']))
{
    // 1 : On force la conversion en nombre entier
    $_GET['repeter'] = (int) $_GET['repeter'];

    // 2 : Le nombre doit être compris entre 1 et 100
    if ($_GET['repeter'] >= 1 AND $_GET['repeter'] <= 100)
    {
        for ($i = 0 ; $i < $_GET['repeter'] ; $i++)
        {
            echo 'Bonjour ' . $_GET['prenom'] . ' ' . $_GET['nom'] . ' !<br />';
        }
    }
}
else
{
   echo 'Il faut renseigner un nom, un prénom et un nombre de répétitions !';
}
?>
```

PHP- formulaires

1.2. Transmettre des données par les formulaires




```
<form method="post" action="cible.php">
 <p>
      On mettra ici les éléments de notre formulaire.<br />
      Notez qu'il n'y a pour l'instant pas de PHP.
</p>
</form>
```


PHP- formulaires

1.2. Transmettre des données par les formulaires

```
<form method="post" action="cible.php">
```

2 méthodes pour envoyer les données du formulaire :

La méthode GET (celle qui est utilisée par défaut si rien n'est renseigné) fait circuler les informations du formulaire en clair dans la barre d'adresse en suivant le format ci-après :

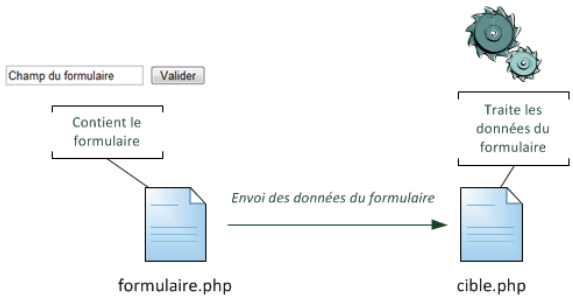
La méthode POST, quant à elle, transmet les informations du formulaire de manière masquée mais non cryptée.
La méthode POST est préférée lorsqu'il y a un nombre important de données à transmettre ou bien lorsqu'il faut envoyer des données sensibles comme des mots de passe. Dans certains cas, seule la méthode POST est requise : un « upload » de fichier par exemple.

PHP- formulaires

1.2. Transmettre des données par les formulaires

```
<form method="post" action="cible.php">
```

L'attribut action sert à définir la page appelée par le formulaire.
C'est cette page qui recevra les données du formulaire et qui sera chargée de les traiter.

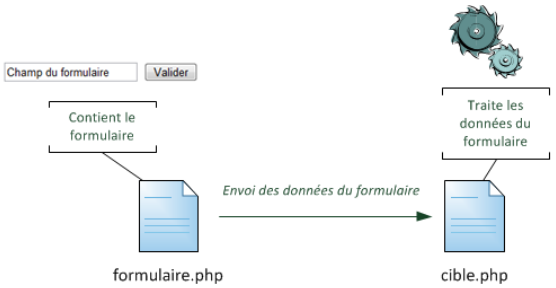


PHP- formulaires

1.2. Transmettre des données par les formulaires

```
<form method="post" action="cible.php">
```

Lorsque le formulaire est envoyé, le serveur Web qui le reçoit le redirige vers le script PHP référencé dans le champ action du formulaire.
PHP va automatiquement créer le tableau $_POST qui contiendra les valeurs saisies dans les champs du formulaire.

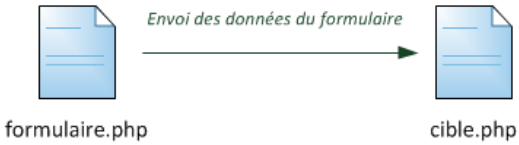


PHP- formulaires

1.2. Transmettre des données par les formulaires

```
<form method="post" action="cible.php">
```

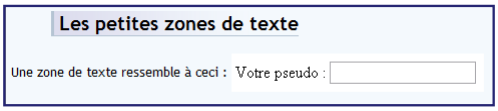
En théorie rien n'empêche le formulaire de s'appeler lui-même.
Il suffirait d'écrire action="formulaire.php". Dans ce cas, la page du formulaire doit être capable aussi bien d'afficher le formulaire que de traiter les données.



PHP- formulaires

2. les champs d'un formulaire

2.1 Les petites zones de texte




```
<input type="text" name="pseudo" value="M@teo21" />
```

Nom du champ (côté serveur)

Valeur par défaut

le texte que le visiteur aura rentré sera disponible dans cible.php sous la forme d'une variable appelée $_POST['pseudo'].

Pour les mots de passe :

type="password" aura pour effet de cacher le texte entré par le visiteur.
À part ce détail, le fonctionnement reste le même.

PHP- formulaires

2.1 Les petites zones de texte

le champ **<input type="submit" />** permet de créer le bouton de validation du formulaire qui commande l'envoi des données, et donc la redirection du visiteur vers la page cible.

formulaire.php

```
<p>
    Cette page ne contient que du HTML.<br />
    Veuillez taper votre prénom :
</p>

<form action="cible.php" method="post">
<p>
    <input type="text" name="prenom" />
    <input type="submit" value="Valider" />
</p>
</form>
```

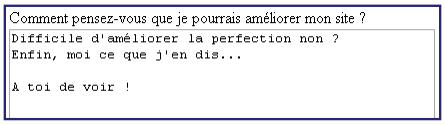
Le contenu de l'attribut **"value"** constitue le texte visible du bouton dans le formulaire.

cible.php

```
<p>Bonjour !</p>

<p>Je sais comment tu t'appelles, hé hé. Tu t'appelles <?php echo $_POST['prenom']; ?> !</p>
```


PHP- formulaires

2.2 Les grandes zones de texte




```
<textarea name="message" rows="8" cols="45"> Votre message ici. </textarea>
```

Définit la taille de la zone de saisie.

Message par défaut (optionnel)

PHP- formulaires

2.3 Les cases à cocher

Nom de la variable à transmettre

Libellé affiché

```
<input type = "checkbox" name= "case"  id= "case" />
  <label for= "case"> ma case </label>
```

L'utilisation de la balise <label> permet d'associer le libellé à la case à cocher qui a le même id que son attribut for, ce qui permet de cliquer sur le libellé pour cocher la case.

Si vous voulez que la case soit cochée par défaut, il faudra lui rajouter l'attribut checked="checked".

```
<input type="checkbox" name="case" checked="checked" />
```


PHP- formulaires

2.3 Les cases à cocher

```
<input type = "checkbox" name= "case"/> <label for= "case"> ma case </label>
```

-Si la case est cochée, alors $_POST['case'] aura pour valeur "on " .

- Si elle n'est pas cochée, alors $_POST['case'] n'existera pas.
Il faut faire un test avec isset($_POST['case']).

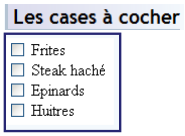
```
<input type = "checkbox" name= "case" value = " valeur " /> ma case
```

Valeur transmise si la case est cochée

PHP- formulaires

2.3 Les groupes de cases à cocher

On peut créer un groupe de cases à cocher.
Il est alors possible de cocher plusieurs cases simultanément.



On récupère les valeurs dans un tableau

```
<input type = "checkbox" name= " aliment [] "  value = "frites" /> Frites
<input type = "checkbox" name= " aliment[]  "  value = "Steak" />  Steak
<input type = "checkbox" name= " aliment[] "   value = "Epinards" /> Epinards
<input type = "checkbox" name= " aliment[] "   value = "Huitres" /> Huitres
```


PHP- formulaires

2.4 Les boutons d'option


```
Aimez-vous les frites ?
<input type="radio" name="frites" value="oui" checked="checked" /> Oui
<input type="radio" name="frites" value="non" /> Non
```

- les deux boutons d'option ont le même nom ("frites"). C'est très important, car les boutons d'options fonctionnent par "groupes" : tous les boutons d'option d'un même groupe ont le même nom. Un groupe contient min 2 boutons.
- A la différence des cases à cocher, un seul choix est autorisé.
- Pour pré-cocher l'un de ces boutons d'option, il faut ajouter un checked.

PHP- formulaires

2.5 Les champs cachés

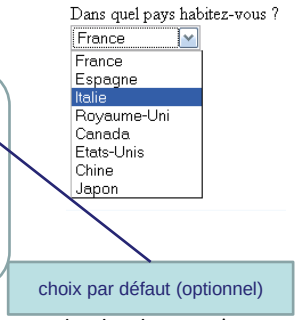
Les champs cachés représentent un code dans votre formulaire qui n'apparaîtra pas aux yeux du visiteur, mais qui va quand même créer une variable avec une valeur. On peut s'en servir pour transmettre des informations fixes.

```
<input type="hidden" name="pseudo" value= "Popeye" />
```

Dans la page cible, une variable $_POST['pseudo'] sera créée et elle aura la valeur "Popeye"

PHP- formulaires

2.6 Les listes déroulantes à choix unique




```
select name="choix">
        <option value="choix1"  selected="selected">
                                    Choix 1 </option>
        <option value="choix2">Choix 2</option>
        <option value="choix3">Choix 3</option>
        <option value="choix4">Choix 4</option>
</select>
```

choix par défaut (optionnel)

- Une variable $_POST['choix'] sera créée, et elle contiendra le choix qu'a fait l'utilisateur.
- S'il a choisi "Choix 3", la variable $_POST['choix'] sera égale au value correspondant, c'est-à-dire "choix3".

PHP- formulaires

2.6 Les listes déroulantes à choix multiple

vos couleurs préférées ?
bleu
jaune

```
<label > vos couleurs préférées ? </label> <br/>
 <select name = "couleur[ ]" multiple="multiple" size="2">
   <option value ="rouge" >rouge</option>
   <option value ="vert" >vert </option>
   <option value ="bleu" >bleu</option>
   <option value ="jaune">jaune</option>
 </select>
```

Dans les listes de sélection à choix multiple, l'utilisateur doit maintenir enfoncée la touche CTRL pour faire plusieurs choix.

PHP- formulaires

3 Les boutons d'envoi multiples

```
<input type= " submit " name = "traitement" value= " création" />
<input type= " submit " name = "traitement" value= " modification" />
<input type= " submit " name = "traitement" value= " suppression" />
```

- Si l'on a plusieurs boutons, le champ name est obligatoire et identique pour tous les boutons.
- idée : effectuer des tâches spécialisées en fonction de la valeur associée.

PHP- formulaires

4 La réinitialisation du formulaire

```
<input type=" reset " value=" effacer" />
```

- Ce bouton réinitialise le formulaire.
- Si les éléments du formulaire ont des attributs value qui définissent des valeurs par défaut, ce sont ces valeurs qui apparaissent au démarrage de la page.

PHP- formulaires

5 Exemple de formulaire

```
<body>
   <!--  //e6_0.php -->
<form method="post" action="e6_0bis.php" >
<!--//zone de texte -->
<input type ="text" name ="nom" />  nom <br/>
<!--// password-->
<input type ="password" name ="motpasse" /> mot passe <br/> <br/>
<!--// case à cocher simple  -->
<input type ="checkbox" name ="content" value="content"/> content ? <br/>
      <br/>
<!--// cases à cocher multiples  -->
 <input type ="checkbox" name ="lang[ ]" value="frans" /> français <br/>
 <input type ="checkbox" name ="lang[ ]" value="engels"/>  anglais<br/>
 <input type ="checkbox" name ="lang[ ]" value="nederlands" />  néerlandais<br/>
 <br/>
```

PHP- formulaires

5 Exemple de formulaire

```
<!--// bouton radio  (bouton d'option)-->
<label> Homme </label> <input type = "radio" name ="sexe" value="homme"/>
      <br/>
 <label> Femme </label> <input type = "radio" name ="sexe" value="femme"/>
      <br/>
 <br/>

 <!--// liste déroulante à choix unique-->
 <label> votre signe astro ? </label> <br/>
 <select name = "signe" >
   <option value ="poisson" >poisson</option>
   <option value ="sagittaire" > sagittaire</option>
   <option value ="taureau" >taureau</option>
   <option value ="vierge">vierge</option>
 </select> <br/>
```


PHP- formulaires

5 Exemple de formulaire

```
<!--// liste déroulante à choix multiple-->
 <label for="choix"> vos couleurs préférées ? </label> <br/>
 <select name = "couleur[ ]" multiple="multiple" size="2">
   <option value ="rouge" >rouge</option>
   <option value ="vert" >vert </option>
   <option value ="bleu" >bleu</option>
   <option value ="jaune">jaune</option>
 </select> <br/>  <br/> <br/>

<!--// envoi du formulaire-->
<input type="submit" value ="envoyer"/>
</form>

</body>
```

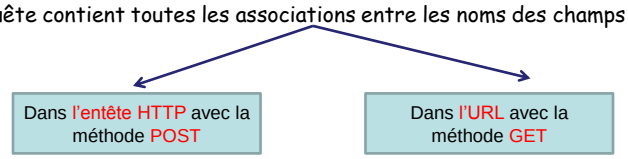
PHP- formulaires

6. Récupération des données du formulaire

Que se passe-t-il lorsque l'utilisateur clique sur le bouton d'envoi ?

Une requête http est envoyée au serveur à destination du script désigné dans l'attribut action de <form>.

La requête contient toutes les associations entre les noms des champs et leur valeur.



Le script récepteur est écrit en PHP.
2 cas apparaissent : -la transmission de valeurs uniques (texte par ex)
- la transmission de valeurs multiples (cases à cocher par ex)

Rem : dans la suite, on n'envisage que l'utilisation de la méthode « post ».

PHP- formulaires

6. Récupération des données du formulaire

A) Récupération des valeurs uniques : ← $_POST [" name"]

Texte, bouton radio, liste de sélection à choix unique

- Ces valeurs sont contenues sur le serveur dans un tableau associatif $_POST.
- Les clés de ce tableau sont les noms associés aux champs par l'attribut « name ».

B) Récupération de valeurs multiples : ← $_POST [" name"] [i]

groupe de cases à cocher, liste de sélection à choix multiple

- Ces valeurs sont contenues sur le serveur dans un tableau associatif $_POST multidimensionnel, en l'occurrence à 2 dimensions.

PHP- formulaires

6. Récupération des données du formulaire

```
<?php   //e6_0bis.php

 // pour les valeurs multiples
  $lang = $_POST ["lang"];
  $couleur= $_POST ["couleur"];
  //
  // trt des valeurs uniques
  //

 // zones de texte
  echo "votre nom est :  ".$_POST ["nom"]."</br>";
  echo "votre mot passe est :  ".$_POST ["motpasse"]."</br>";

 // bouton radio  ( si pas coché, n'existe pas)
 if (isset ($_POST ["sexe"] ) )
  echo "votre êtes un(e) : ".$_POST["sexe"]."</br>« ;
```


PHP- formulaires

6. Récupération des données du formulaire

```
 // case à cocher (si pas cochée, n'existe pas)
 if (isset ($_POST ["content"] ) )
      echo "vous êtes :  ".$_POST ["content"]."</br>";

//liste déroulante à choix unique
 echo " votre signe astro est  : ".$_POST["signe"]."</br>";

//
// trt des valeurs multiples
//

// trt cases à cocher multiples
 echo "vous parlez :  " ;
 foreach ($lang as $valeur)
  {
    echo "<li> $valeur </li> ";
  }
```

PHP- formulaires

6. Récupération des données du formulaire

```
//liste déroulante à sélection multiple
echo " vos couleurs préférées sont :";
foreach ($couleur as $valeur)
 {
   echo "<li> $valeur </li> ";
 }

echo "<a href=\"e6_0.php\"> clique ici pour le retour </a>";
?>
```


PHP- formulaires

7. Gérer les boutons d'envoi multiples

```
<body>
  <!-- //e6_1.php-->

<form method="post" action="e6_1bis.php" >

<label for="n1"> nombre1</label>
<input type ="texte" name ="n1" id="n1"/> <br/>

<label for="n2"> nombre2</label>
<input type ="texte" name ="n2" id="n2"/> <br/> <br/>

<input name="calcul" type="submit" value ="add"/>
<input name="calcul" type="submit" value ="sub"/>

</form>
```

PHP- formulaires

7. Gérer les boutons d'envoi multiples

```
<?php   //e6_1bis.php

$n1 = $_POST ["n1"];
  $n2 = $_POST ["n2"];

 switch ( $_POST ["calcul"])
  {
     case "add":
       $result=$n1+$n2;
       break;
     case "sub":
       $result=$n1-$n2;
       break;
   }
  echo "le résultat est :  ".$result."</br>";
  echo "<a href=\"e6_1.php\"> clique ici pour le retour </a>";
?>
```


PHP- formulaires

8. Les failles de sécurité

8.1 Pourquoi les formulaires ne sont pas sûrs

Toutes les informations qui proviennent de l'utilisateur, à savoir les données de $_GET et de $_POST, doivent être traitées avec la plus grande méfiance.

Exemple 1 : les visiteurs doivent entrer leur date de naissance au format JJ/MM/AAAA. Certains entreront « Je suis né le 4 octobre 1987 » au lieu de « 04/10/1987 ».
Quand le traitement de la date se fera en PHP, il faudra vérifier qu'elle respecte le format demandé.

Exemple 2 : entrer un nombre entre 1 et 100, certains entreront 12124556656.

Exemple3 : les visiteurs peuvent supprimer certains champs du formulaire.

PHP- formulaires

8. Les failles de sécurité

8.1 Pourquoi les formulaires ne sont pas sûrs

La page du formulaire se trouve sur mon site, comment peut faire un visiteur pour modifier ma page web ? Il peut voir le code HTML mais pas le modifier !
Il peut récupérer le code HTML, créer une copie légèrement modifiée de votre formulaire et de la stocker sur son serveur.

La ligne <form method="post" action="cible.php"> est remplacée par :

<form method="post" action="http://www.monsite.com/cible.php">

Adresse absolue

La page "cible.php " ne peut pas savoir de quel formulaire vient le visiteur.

Internet Explorer 8 et Google Chrome embarquent des « outils pour les développeurs » qui permettent de modifier le code HTML de la page que l'on visite en temps réel.

PHP- formulaires

8. Les failles de sécurité

8.2 la faille XSS (cross-site scripting)

C'est une technique qui consiste à injecter du code HTML contenant du JavaScript dans vos pages pour le faire exécuter au visiteur.

Exemple1
le code est : <Tu t'appelles <?php echo $_POST['prenom']; ?>

le visiteur décide d'écrire du code HTML à la place de son prénom :
Badaboum.

Le code source HTML qui sera généré par PHP sera le suivant :
 <Tu t'appelles Badaboum

PHP- formulaires

8. Les failles de sécurité

8.2 la faille XSS (cross-site scripting)

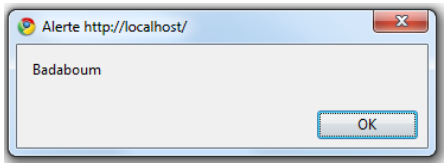
Exemple2

le code est : <Tu t'appelles <?php echo $_POST['prenom']; ?>

le visiteur décide d'ouvrir des balises javascript :

Tu t'appelles <script type="text/javascript">alert('Badaboum')</script> !

Les visiteurs verront s'ouvrir une fenêtre :



PHP- formulaires

8. Les failles de sécurité

8.2 la faille XSS (cross-site scripting)

Exemple3

Les cookies stockent des informations sur votre session et parfois des informations plus confidentielles, comme des pseudonyme et mot de passe.

Si le visiteur est assez malin pour récupérer vos cookies , les choses deviennent critiques.

PHP- formulaires

8. Les failles de sécurité

8.3 comment résoudre le problème ?

Il faut protéger le code HTML en l'échappant, c'est-à-dire en affichant les balises (ou en les retirant) plutôt que de les faire exécuter par le navigateur avec la fonction htmlspecialchars ( ) qui va transformer les chevrons des balises HTML <> en < et > respectivement.

Cela provoquera l'affichage de la balise plutôt que son exécution.

Exemple :

```
1 <p>Tu t'appelles <?php echo htmlspecialchars($_POST['prenom']); ?> !</p>
```

donnera à l'affichage

```
1 <p>Tu t'appelles &lt;strong&gt;Badaboum&lt;/strong&gt; !</p>
```

Il faut penser à utiliser cette fonction sur tous les textes envoyés par l'utilisateur qui sont susceptibles d'être affichés sur une page web.

PHP- formulaires

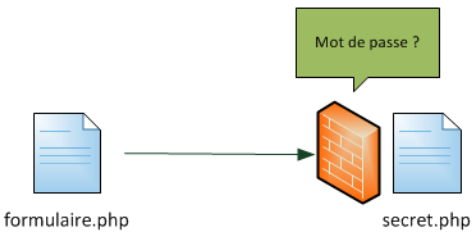
9. Codage des formulaires

9.1 problème posé

Nous voulons mettre en ligne une page web pour donner des informations confidentielles à certaines personnes. Cependant, pour limiter l'accès à cette page, il faudra connaître un mot de passe.

La page ne doit pas s'afficher si le mot de passe n'est pas correct ou inexistant.

9.2 Première solution



PHP- formulaires

9.2 Première solution (formulaire.php)

```
<html>
 <head>    <title> Page protégée par mot de passe </title>    </head>
 <body>
   <p> Veuillez entrer le mot de passe  </p>
   <form action= "secret.php"   method= " post" >
      <p>
      <input type = "password" name= "mot_passe" />
      <input type = "submit" value = "valider" />
      </p>
   </form>
   <p> cette est page est confidentielle </p>
  </body>
</html>
```

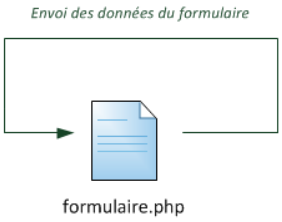

PHP- formulaires

9.2 Première solution (secret.php)

```
<html>
 <head>    <title> Page protégée par mot de passe </title>    </head>
 <body>
 <?php
      if (isset ($_POST ['mot_passe'] ) AND $_POST ['mot_passe'] == "HELHA")
      {
       // on affiche les codes
       echo "<h1> voici les infos </h1>" ;
       echo " <p> <strong>  IG1 – IG2 – IG3  </strong>  </p>";
            echo  "<p> cette est page est confidentielle </p>";
       }
      else   // on affiche un msg d'erreur
      {
         echo   "mot de passe incorrect" ;
      }
 ?>
 </body>
</html>
```

PHP- formulaires

9.3 Deuxième solution



- La première fois que le visiteur charge la page formulaire.php, aucune donnée POST n'est envoyée à la page. C'est donc le formulaire qui s'affiche.
- Une fois qu'on a envoyé le formulaire, la page formulaire.php est rechargée et cette fois, elle reçoit les données POST qu'on vient d'envoyer. Elle peut donc les analyser et, si le mot de passe est bon, elle affiche les codes secrets.

PHP- formulaires

9.3 Deuxième solution

```
<?php
   // le mot de passe n'a pas été envoyé (premier appel)
   if ( ! isset ($_POST ['mot_passe'] ) AND $_POST ['mot_passe']  != "HELHA")
      {
       // on affiche le formulaire de saisie
      }
      elseif    (isset ($_POST ['mot_passe'] ) AND $_POST ['mot_passe'] == "HELHA")
          {
                 // afficher les codes secrets
          }
          else
          {
                // on affiche un msg d'erreur
          }
 ?>
```

PHP- fonctions

1. Les fonctions natives de PHP

2. Créer ses propres fonctions

   2.1 définition et appel de fonction
   2.2 définition et appel de fonction avec retour d'un résultat
   2.3 définition, appel de fonction sans retour de valeur
   2.4 définition, appel de fonction avec retour de plusieurs valeurs
   2.5 passage d'arguments avec valeurs par défaut
   2.6 passage des arguments par référence

3. Portée des variables
   3.1 variables locales globales
   3.2 variables super globales

PHP- fonctions

1. Les fonctions natives de PHP

Il existe des
- fonctions natives écrites en langage C
- Modules d'extension qu'il est possible d'associer à la distribution standard

Avant d'utiliser un module, il convient de vérifier qu'il est bien installé sur le serveur de votre hébergeur.

```
// php 7.1
$tabtext = get_loaded_extensions ();
foreach ($tabtext as $cle=>$valeur)
      {
       echo "$cle    $valeur ";
      }
```

retourne un tableau contenant tous les modules d'extension installés sur le serveur

```
if (function_exists ('mysql_pconnect') )
            echo "la fonction est utilisable";
```

Vérifie la disponibilité d'une fonction connue
Renvoie TRUE si la fonction est utilisable
FALSE sinon

PHP- fonctions

1. Les fonctions natives de PHP

Voici un petit aperçu des fonctions qui existent pour vous mettre l'eau à la bouche :

- une fonction qui permet de rechercher et de remplacer des mots dans une variable
- une fonction qui envoie un fichier sur un serveur
- une fonction qui permet de créer des images miniatures (aussi appelées thumbnails)
- une fonction qui envoie un mail avec PHP (très pratique pour faire une newsletter !)
- une fonction qui permet de modifier des images, y écrire du texte, tracer des lignes, des rectangles etc…
- une fonction qui crypte des mots de passe.
- une fonction qui renvoie l'heure, la date…

PHP- fonctions

2. Créer ses propres fonctions

Ces fonctions personnalisées peuvent être écrites dans le script lui-même ou dans un script séparé qu'il suffira d'inclure dans de nouveaux scripts à l'aide de l'instruction include () ou require ()

2.1 définition et appel de fonction

Définition d'une fonction avec retour

```
function   nomFonction ($par1, $par2,…)
{
  code de la fonction
  return $res;
}
```

Définition d'une fonction sans retour

```
function   nomFonction ($par1, $par2,…)
{
    code de la fonction
}
```

Appel d'une fonction avec retour

```
$resultat = nomFonction ($val1, $val2, …);
```

Appel d'une fonction sans retour

```
nomFonction ($val1, $val2, …)
```

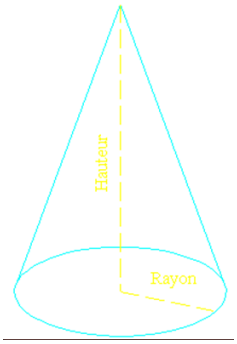
PHP- fonctions

2.2 définition, appel de fonction avec retour d'un résultat

Exemple 1 : calcul du volume d'un cylindre

La formule de calcul est : rayon * rayon * 3.14 * hauteur * (1/3)

```
<?php   //e7_2.php
//définition  de la fonction
   function VolumeCone($rayon, $hauteur)
{
  $volume = $rayon * $rayon * 3.14 * $hauteur * (1/3);
  return $volume;
}
// appel de la fonction
$volume = VolumeCone(3, 1);
echo "Le volume d'un cône de rayon 3 et de hauteur 1 est de $volume";
?>
```



PHP- fonctions

2.3 définition, appel de fonction sans retour de valeur

Exemple 2 : fonction de lecture d'un tableau

```
<?php   //e7_3.php
//définition  de la fonction
// tabuni (nom du tableau, largeur de la bordure, libellé col1, libellé col2)

function tabuni($tab,$bord,$lib1,$lib2)
{
  echo "<table border=\"$bord\" width=\"75%\"> <tbody><tr><th>$lib1</th>
       <th>$lib2</th><tr>";
  foreach ($tab as $cle=>$valeur)
       {
         echo "<tr><td>$cle </td> <td>$valeur </td><tr> ";
       }
  echo "<br/></tbody> </table>";
 }
```

PHP- fonctions

Exemple 2 : fonction de lecture de tableaux (suite)

```
// définition des tableaux associtaifs
 $tab1= array ("France"=>"Paris","Allemagne"=>"Berlin","Espagne"=>"Madrid");
 $tab2= array ("Poisson"=>"requin","Cétacé"=>"dauphin","Oiseau"=>"Aigle");

 // appels de la fonction
 tabuni($tab1,1,"Pays","Capitale");
 tabuni($tab2,6,"Genre","Espèce");
?>
```


PHP- fonctions

2.4 définition, appel de fonction avec retour de plusieurs valeurs

Via l'utilisation d'un tableau (array), une fonction est capable de renvoyer plusieurs valeurs.

Exemple 3 : fonction de calcul (somme et différence)

```
<?php   //e7_4.php
//définition  de la fonction
   function calculVal($a,$b)
   {
     $som = $a + $b;
     $dif = $a - $b;
     return array($som,$dif);
   }
  // appel de la fonction
  $result = calculVal (8, 6);
  echo "somme des 2 nombres : ".$result[0]."<br/>";
  echo "différence des 2 nombres : ".$result[1]."<br/>";
?>
```

PHP- fonctions

2.5 définition, appel de fonction avec valeur par défaut des arguments

Il est possible de fixer des valeurs par défaut pour les arguments.
Ces valeurs doivent être précisées en tant que constantes.
Ces arguments doivent être placés en fin de liste des arguments.

Exemple 4 : fonction de mise en page

```
<?php   //e7_5.php
//définition  de la fonction
   function header_page($title,$bgcolor="red")
   {
     echo "<html>
     <head><title>$title</title></head>
     <body bgcolor=\"$bgcolor\">";
   }

  // appel de la fonction
  header_page("cours PHP");
  echo "<h1>Utilisation d'un paramètre par défaut</h1>";
?>
```

Si on ne donne pas de valeur à ces paramètres lors de l'appel de la fonction, les valeurs par défaut seront prises en compte

PHP- fonctions

2.6 passage des arguments par référence

Jusqu'à présent, les arguments étaient passés par valeur.
Il est possible de passer un argument par référence en faisant précéder l'argument de « & » dans la définition de la fonction.

Exemple 5 : fonction de traitement d'un tableau

Le premier argument est passé par référence , le second par valeur.

PHP- fonctions

Exemple 5 : fonction de traitement d'un tableau

```
<?php   //e7_6.php
//définition  de la fonction
//multiplication des éléments d'un tableau par un coeff
   function prodTab(&$tab,$coeff)
   {
     foreach ($tab as $cle=>$val)
       $tab[$cle]*=$coeff;
   }

 // définition du tableau
  $tabnum = range(1,7);
  echo "tableau avant : ", print_r($tabnum),"<br/>";

 // appel de la fonction
  prodTab($tabnum,3);
  echo "tableau après : ",print_r($tabnum),"<br/>";
?>
```


PHP- fonctions

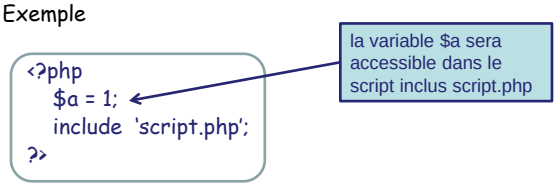
3. Portée des variables

Les variables ont une portée déterminée selon le contexte.

3.1 variables locales et globales

- Une variable définie en dehors d'une fonction est « globale » et accessible partout dans le script qui l'a créée.

Exemple



PHP- fonctions

3.1 variables locales et globales

- lorsque vous définissez une fonction, la portée d'une variable définie dans cette fonction est locale à la fonction.
  Toutes les variables utilisées dans la déclaration d'une fonction sont donc « locales » au bloc de définition de la fonction..
- ce concept diffère du langage C dans lequel une variable globale est automatiquement accessible dans les fonctions, à moins d'être redéfinie localement dans la fonction.

```
<?php  //e7_7.php
//définition d'une variable globale
   $message = "apprendre le langage PHP";
 // fonction utilisant une variable locale
   function affiche()
   {
     echo $message;
   }
 // appel de la fonction
 affiche();
?>
```

- Le script affiche une page blanche.
-La variable $message utilisée dans la fonction a une portée locale et n'a pas été initialisée.
-- la fonction ne fait pas réf à la var $message déclarée globalement.

PHP- fonctions

3.1 variables locales et globales

- Pour pouvoir utiliser une variable globale dans une fonction, il faut la redéclarer explicitement à ce niveau en utilisant le tableau associatif prédéfini « $GLOBALS ».
- Les clés de ce dernier sont les noms des variables globales sans le « $ »

```
<?php  //e7_9.php
//définition de variables globales
   $message = "apprendre le langage PHP";
   $remarque = "c'est amusant<br/>";
 // fonction utilisant des variables globales  (version 2)
   function affiche()
   {
     echo $GLOBALS ["message"].",".$GLOBALS ["remarque"];

   }
 // appel de la fonction
 affiche();
?>
```

PHP- fonctions

3.2 variables super globales

Les variables super globales sont :

$GLOBALS
$_SERVER
$_GET
$_POST
$_FILES
$_COOKIE
$_SESSION
$_REQUEST
$_ENV

PHP- fonctions

3.2 variables super globales

Caractéristiques :

- elles sont écrites en majuscules et commencent toutes, à une exception près, par un underscore (_).
  $_GET et $_POST en sont des exemples que vous connaissez .
- les super globales sont des tableaux associatifs (array) qui contiennent généralement de nombreuses informations.
- ces variables sont automatiquement créées par PHP à chaque fois qu'une page est chargée. Elles existent donc sur toutes les pages et sont accessibles partout : au milieu de votre code, au début, dans les fonctions, etc.

PHP- fonctions

3.2 variables super globales

$_SERVER : ce sont des valeurs renvoyées par le serveur. Elles sont nombreuses et quelques-unes d'entre elles peuvent nous être d'une grande utilité.

$_SERVER['REMOTE_ADDR'] donne l'adresse IP du client qui a demandé à voir la page, ce qui peut être utile pour l'identifier.

$_ENV : ce sont des variables d'environnement toujours données par le serveur. C'est le plus souvent sous des serveurs Linux que l'on retrouve des informations dans cette superglobale.

$_SESSION : on y retrouve les variables de session. Ce sont des variables qui restent stockées sur le serveur le temps de la présence d'un visiteur.

PHP- fonctions

3.2 variables super globales

$_COOKIE : contient les valeurs des cookies enregistrés sur l'ordinateur du visiteur. Cela nous permet de stocker des informations sur l'ordinateur du visiteur pendant plusieurs mois, pour se souvenir de son nom par exemple.

$_GET : elle contient les données envoyées en paramètres dans l'URL.

$_POST : elle contient les informations qui viennent d'être envoyées par un formulaire.

$_FILES : elle contient la liste des fichiers qui ont été envoyés via le formulaire précédent.

PHP – CHAINES CARACTERES

1. Affichage des chaînes
2. Affichage formaté
3. Longueur d'une chaîne et code des caractères
4. Mise en forme d'une chaîne

   4.1 modification de la casse
   4.2 gestion des espaces
   4.3 entités HTML et caractères spéciaux

PHP – CHAINES CARACTERES

5. Recherche de sous-chaînes

   5.1 une chaîne vue comme un tableau
   5.2 les fonctions strstr ( ) et stristr ( )
   5.3 la fonction strrchr ( )
   5.4 les fonctions substr ( ), substr_count ( ), str_replace ()
   5.5 les fonctions strpos ( ), stripos ( ), strrpos( ), strripos( )
   5.6 capture de sous-chaînes dans des variables

6. Comparaison de chaînes
7. Transformation de chaînes en tableaux

PHP – CHAINES CARACTERES

1. Affichage des chaînes

Dans une page WEB, l'essentiel du contenu est constitué de chaînes de caractères.

- Quand le contenu de la page est créé dynamiquement à partir d'un fichier ou d'une DB, ce sont surtout des chaînes de caractères qui sont manipulées.
- toutes les variables issues de l'envoi d'un formulaire sont de type string.

Pour afficher des données : echo

PHP – CHAINES CARACTERES

1. affichage des chaînes

La fonction « echo » est utilisable de plusieurs façons pour afficher plusieurs chaînes à la suite l'une de l'autre.

Concaténation avec .

```
<?php  // e4_1.php
   $nom="isat";
   echo "vous êtes à l'école <b>".$nom."</b> aujourd'hui le ".date('d')."<br/>";

   echo "vous êtes à l'école <b>", $nom, "</b> aujourd'hui le ",date('d'),"<br/>";

   echo "vous êtes à l'école <b> $nom </b> aujourd'hui le date('d')<br/>";
?>
```

Les **fonctions** ne peuvent être incluses dans une chaîne unique

Séparation des expressions par **,**

PHP – CHAINES CARACTERES

2. Affichage formaté

Les fonctions « printf » et « sprintf » permettent d'obtenir des résultats uniformes :

- Aligner des chaînes sur plusieurs lignes
- Superposition correcte de chiffres en colonnes pour des montants

```
void printf (string " format ", string $ch1, string $ch2, …,$chn);
```

Affiche directement le contenu des chaînes $ch1,$ch2,…selon le format spécifié dans la chaîne " format "

```
string sprintf (string " format ", string $ch1, string $ch2, …,$chn);
```

Retourne une chaîne unique composée des chaînes $ch1,$ch2,…selon le format spécifié dans la chaîne " format "

PHP – CHAINES CARACTERES

2. Affichage formaté

```
void vprintf (string " format ", array $tab);
```

```
string vsprintf (string " format ", array $tab);
```

rôle équivalent à printf et sprintf mais les chaînes sont passées en **argument dans un tableau**

```
string str_repeat (string $ch, int N);
```

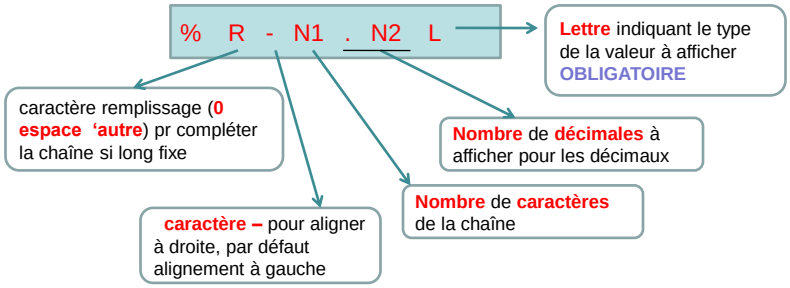
Crée une chaîne contenant N fois la chaîne $ch

PHP – CHAINES CARACTERES

Définition de la chaîne de formatage

Elle est composée de

- un texte ordinaire explicatif
- directives d'affichage (= caractères spéciaux qui indiquent la manière dont les chaînes passées en paramètres doivent être intégrées dans la chaîne)



PHP – CHAINES CARACTERES

Caractères de formatage du type de données

Caractère	Signification
%b	Interprète la chaîne $ch comme un entier et l'affiche en binaire : `$ch="89";` `printf ("En binaire $ch s'écrit %b  ", $ch) ;` `//Affiche :En binaire 89 s'écrit 1011001`
%c	Interprète la chaîne $ch comme un entier et affiche le caractère dont le code ASCII correspond à ce nombre : `$ch="115";` `printf (" Le caractère de code $ch est %c  ", $ch);` `//Affiche: Le caractère de code 115 est s`

%d	Interprète la chaîne $ch comme un entier signé et l'affiche comme un nombre en base 10 :
	`$ch = "-25";`
	`printf ("La valeur de \$ch est %d", $ch);Affiche -25`

%f	Interprète la chaîne $ch comme un nombre de type double et l'affiche avec sa partie décimale à 6 chiffres. Les caractères non numériques de $ch ne sont pas pris en compte :
	`$ch = "25.52 mètres";`
	`printf ("La longueur est de %f m", $ch);`
	`Affiche: La longueur est de 25.520000 m`

PHP – CHAINES CARACTERES

Caractères de formatage du type de données

Caractère	Signification

%s	Interprète $ch comme une chaîne et l'affiche telle quelle :
	`$ch1 = "Monsieur " ; $ch2 = " Schtroumpf" ;`
	`sprintf ("Bonjour,%s %s bravo !",$ch1 $ch2);`
	Équivaut à :
	`echo "Bonjour $ch1 $ch2, bravo !";`

%x
ou
%X
	Interprète la chaîne $ch comme un entier et l'affiche en hexadécimal en minuscules (%x) ou (%X) :
	`$ch ="252756";`
	`printf ("En hexadécimal $ch s'écrit %x ", $ch) ;`
	`Affiche: En hexadécimal 252756 s'écrit 3db54`

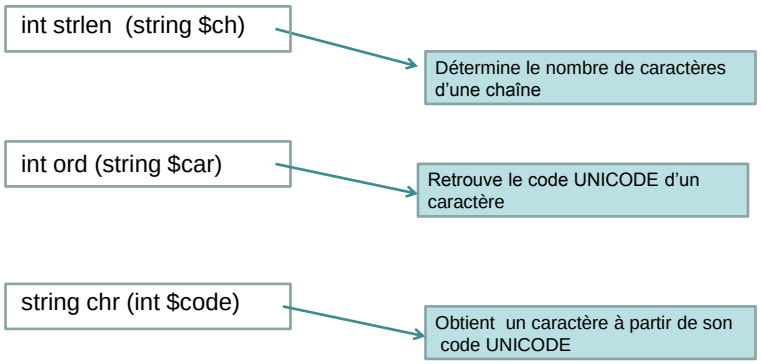
%o	Interprète le chaîne $ch comme un entier et l'affiche en octal :
	`$ch = 252;`
	`printf ("En octal le nombre $ch s'écrit %o", $ch);`
	`Affiche: En octal le nombre 252 s'écrit 374`

PHP – CHAINES CARACTERES

```
<?php    //e4_2.php
   echo "<h3> votre facture</h3>";
   echo sprintf("<b>%'_25s%'_25s<br/> </b>","prix HT","prix TTC");
   $ht[0] = 30;
   $ht[1] = 100;
   $total=0;
   for ($i=0;$i<2;$i++)
   {
     echo sprintf ("article %d %'_17.2f %'_24.2f<br/>",$i+1,$ht[$i],$ht[$i]*1.25);
     $total += $ht[$i];
   }
   echo str_repeat ("*",50),"<br/>";
   echo sprintf ("TOTAL %'_17.2f%'_24.2f<br/>",$total,$total*1.25);
 ?>
```

PHP – CHAINES CARACTERES

3. Longueur d'une chaîne et code des caractères



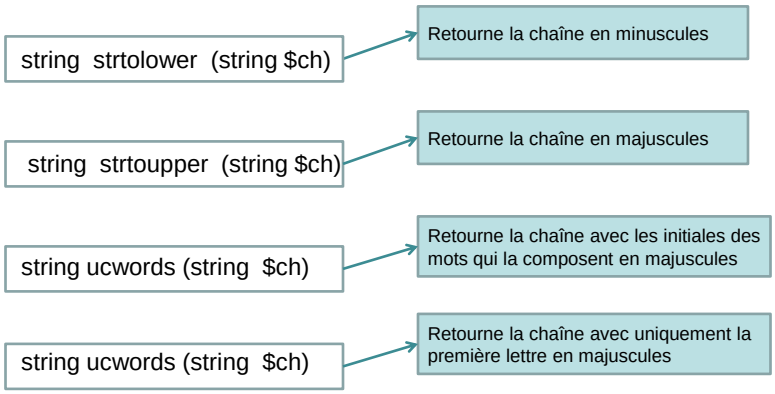
PHP – CHAINES CARACTERES

```php
<?php      //e4_3.php
   //contrôle de login (max 8 car)
   $login="admin";
   if (strlen($login) > 8)
       echo "login erroné";
   else
       echo "votre login est : $login <br/>";
   //génération d'un mot de passe
   $motpasse = "";
   for ($i=1;$i < 6; $i++)
    {
     $nb=rand(65,90);
     $motpasse .= chr($nb);
   }
   echo "votre mot de passe est ", $motpasse;
?>
```

PHP – CHAINES CARACTERES

4. Mise en forme d'une chaîne

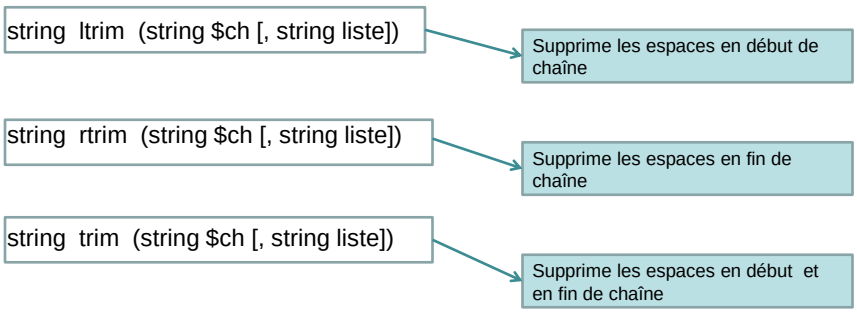
4.1 modification de la casse



PHP – CHAINES CARACTERES

4. Mise en forme d'une chaîne

4.2 gestion des espaces



PHP – CHAINES CARACTERES

4. Mise en forme d'une chaîne

4.2 gestion des espaces

string wordwrap (string $ch [, int N [, string car [, boolean coupe]]])

- Affiche un texte long avec une largeur maximale déterminée
- N définit la largeur
- car contient le caractère à insérer tous les N caractères
- coupe (= TRUE) permet d'effectuer une césure des mots dont la longueur des mots dépasse N

```
<?php      //e4.4php
$ch="cette chaîne anticonstitutionnelle est vraiment
      beaucoup trop longue pour être affichéee en une ligne";
   echo wordwrap ($ch, 10, "<br/>", 1);
?>
```


PHP – CHAINES CARACTERES

4. Mise en forme d'une chaîne

4.3 entités HTML et caractères spéciaux

Définitions :

1. le code *Unicode* est un système de codage des caractères sur 16 bits mis au point en 1991. Le système Unicode permet de représenter n'importe quel caractère par un code sur 16 bits, indépendamment de tout système d'exploitation ou langage de programmation.
   Il regroupe ainsi la quasi-totalité des alphabets existants (arabe, arménien, cyrillique, grec, hébreu, latin, ...) et est compatible avec le code ASCII.

2. Entité HTML : Les *entités* sont des mots mnémotechniques qui s'emploient entre un & initial et un point-virgule final.

Entité HTML	Code unicode	Car affiché
à ou à	représente	à
ô	représente	ô

PHP – CHAINES CARACTERES

4. Mise en forme d'une chaîne

4.3 entités HTML et caractères spéciaux

Utilité des entités :

- Le code source HTML qui sert à produire la page que vous êtes entrain de lire contient, entre autres, le texte proposé à la lecture, en clair ou presque. Or, de nombreuses langues contiennent des signes qui n'existent pas dans la plupart des autres.

  Par exemple le français a des caractères accentués "é", "è", "à", " î " , "ç", signes qu'on ne retrouve pas du tout en anglais. Ces caractères, dans le code source HTML, doivent être remplacés par les *entités HTML* afin que les navigateurs paramétrés pour d'autres langues que le français les affichent correctement.

- Certains caractères ont des significations spéciales en HTML, et doivent être remplacés par des entités HTML pour être affichés. (&, ", ',<,>)

PHP – CHAINES CARACTERES

4. Mise en forme d'une chaîne

4.3 entités HTML et caractères spéciaux

string addslashes (string $ch) → Ajoute le **caractère ** devant les caractères **spéciaux ‘ " \ NULL**

Utile avant d'enregistrer des chaînes dans une base de données

string stripslashes (string $ch) → enlève les **caractères **

Rem : la fonction addslashes () est inutile pour les données en provenance d'un formulaire si la directive magic_quotes_runtime est active dans le php.ini

PHP – CHAINES CARACTERES

4. Mise en forme d'une chaîne

4.3 entités HTML et caractères spéciaux

Certains caractères ont des significations spéciales en HTML, et doivent être remplacés par des entités HTML pour être affichés. 2 fonctions sont pratiques pour éviter que des données fournies par les utilisateurs contiennent des balises HTML : htmlspecialchars () et htmlentities ()

string htmlspecialchars (string $ch [, string charset]]))

- $ch : Chaîne à convertir
- charset : Désigne l'alphabet utilisé, par défaut ISO-8859-1 (Europe occidentale)

"&" devient "&
""" devient """ lorsque **ENT_NOQUOTES** n'est pas utilisée.
"'" (guillemet simple) devient "'" uniquement lorsque **ENT_QUOTES** est utilisée.
"<" (inférieur à) devient "<"
">" (supérieur à) devient ">"

PHP – CHAINES CARACTERES

4. Mise en forme d'une chaîne

4.3 entités HTML et caractères spéciaux

Très pratique par exemple si vous faites un mini-chat et que vous voulez empêcher vos visiteurs d'utiliser du HTML !

```
<?php  //e4_5.php
$variable_html = '<em>Ceci est une variable qui contient du HTML</em>';
$variable_sans_html = htmlspecialchars($variable_html); echo 'Avant : ' .
$variable_html . '<br />Après : ' . $variable_sans_html;
?>
```

affichage

Avant : *Ceci est une variable qui contient du HTML*
Après : Ceci est une variable qui contient du HTML

PHP – CHAINES CARACTERES

4. Mise en forme d'une chaîne

4.3 entités HTML et caractères spéciaux

Si on veut remplacer tous les caractères spéciaux en HTML (code unicode > 128) par des entités HTML

string htmlentities (string $ch [, string charset])

- Chaîne à convertir
- Désigne l'alphabet utilisé, par défaut ISO-8859-1 (Europe occidentale)

```
<?php  //e4_6.php
    $titre = 'Découvrir PHP , et apprendre à programmer"';
    echo "<form method = \"post\" action = \"insert.php\">";
    echo "Titre : ";
    echo "<input type = \"text\" name =\"titre\" size = \"60\" value = \"".htmlentities
      ($titre)."\">";
    echo "</form>";
?>
```


PHP – CHAINES CARACTERES

5. Recherche de sous-chaînes

5.1 une chaîne vue comme tableau

Une chaîne peut être considérée comme un tableau de caractères (indices de 0 à N) .

on peut récupérer le caractère d'indice cste : `$ch [cste]`

ou encore si $ind contient un entier : `$ch[$ind]`

```
<?php    //e4_7.php
  $ch = "bonjour Worldtrade";
  echo "le 9ème caractère est : $ch[8] ";
?>
```

PHP – CHAINES CARACTERES

5. Recherche de sous-chaînes

5.2 les fonctions strstr ( ) et stristr ( )

Elles permettent d'extraire une sous-chaîne d'une chaîne donnée et renvoient tous les caractères allant de la première occurrence de $ch2 jusque la fin de $ch1. (false sinon)

string strstr (string $ch1, string $ch2) ← Sensible à la casse

string stristr (string $ch1, string $ch2) ← insensible à la casse

```
<?php    //e4_8.php
$ch= " Perette et le pot au lait ";
$ssch = strstr($ch, " pot ");
echo $ssch;
?>
```


PHP – CHAINES CARACTERES

5. Recherche de sous-chaînes

5.3 la fonction strrchr ( )

Elle permet d'extraire une sous-chaîne d'une chaîne donnée et renvoie tous les caractères allant de la dernière occurrence de $ch2 jusque la fin de $ch1.

string strrchr (string $ch1, string $ch2) ← sensible à la casse

```
<?php    //e4_9.php
    $ch= "Perette et le pot au lait, c'est pas de pot !";
    $ssch = strrchr($ch, "pot");
    echo $ssch;
?>
```

PHP – CHAINES CARACTERES

5. Recherche de sous-chaînes

5.4 les fonctions substr ( ), substr_count ( ), str_replace ()

Elles permettent d'extraire des sous-chaînes en fonction des indices de caractères dans la chaîne analysée (le premier étant l'indice 0)

string substr (string $ch, integer ind [, integer N])

Retourne la chaîne contenant N caractères de $ch extraits à partir de l'indice ind inclus . (Si N est omis, la sous-chaîne extraite va jusque la fin de $ch.

```
<?php      //e4_10.php
    $url = "http://www.linux.org/";
    if (substr ($url, 0,7) != "http://")
        echo "l 'adresse URL n'est pas complète";
    else
        echo "enregistrement de l'adresse : $url";
?>
```


PHP – CHAINES CARACTERES

5. Recherche de sous-chaînes

5.4 les fonctions substr ( ), substr_count ( ), str_replace ()

Elles permettent d'extraire des sous-chaînes en fonction des indices de caractères dans la chaîne analysée (le premier étant l'indice 0)

int substr_count (string $ch, string $ssch)

Retourne le nombre d'occurrences d'une sous-chaîne $ssch dans une chaîne $ch. (sensible à la casse)

```
<?php      //e4_11.php
      $ch= "Perette et le pot au lait, c'est pas de pot !";
      $ch2 = "pot";
      $nb = substr_count($ch, $ch2);
      echo "le mot \"$ch2\" est présent $nb fois dans \"$ch\"";
?>
```

PHP – CHAINES CARACTERES

5. Recherche de sous-chaînes

5.4 les fonctions substr ( ), substr_count ( ), str_replace ()

string str_replace (string $ch1, string $ch2, string $ch [, string $var])

Retourne la chaîne $ch dans laquelle toutes les occurrences de $ch1 sont remplacées par $ch2. $var contient le nombre de remplacements à effectuer

```
<?php    //e4_12.php
    $text = "HTML est un langage de description de contenu";
    $newtext = str_replace ("HTML", "XML", $text);
    echo $newtext;
?>
```


PHP – CHAINES CARACTERES

5. Recherche de sous-chaînes

5.5 les fonctions strpos ( ), stripos ( ), strrpos( ), strripos( )

Retourne la postion du premier caractère de la première occurrence d'une sous-chaîne $ssch ou FALSE

Retourne la postion du premier caractère de la dernière occurrence d'une sous-chaîne $ssch ou FALSE

Identique à strpos ( ) insensible à la casse

Identique à strrpos ( ) insensible à la casse

Rem : elles ont une syntaxe identique

int strpos (string $ch, string $ssch, [, int N])

N désigne la position à partir de laquelle la recherche s 'effectue dans $ch

PHP – CHAINES CARACTERES

5. Recherche de sous-chaînes

5.5 les fonctions strpos ( ), stripos ( ), strrpos( ), strripos( )

```php
<?php    //e4_13.php
    $text = "HTML est un langage. XML est différent de HTML. ";
    $n = strpos ($text, "HTML");
    if ($n === FALSE)
       echo "pas d'occurence";
    else
       echo $n;
?>
```

Opérateur d'identité

PHP – CHAINES CARACTERES

5. Recherche de sous-chaînes

5.6 capture de sous-chaînes dans des variables

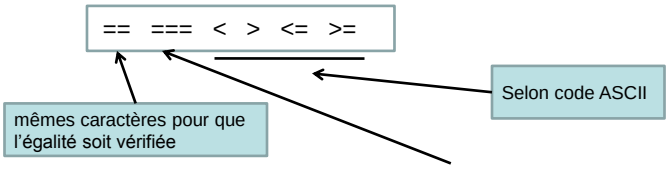
La fonction sscanf ( ) permet de récupérer chacun des éléments d'une chaîne dans des variables séparées.

```php
<?php    //e4_14.php
    $personne = "1685-1750 Jean-Sébastien Bach";
    $format =  "%d-%d %s %s";
    $nb = sscanf ($personne, $format, $dtnai, $dtmort, $prenom, $nom);
    echo "$prenom $nom est né en $dtnai et mort en $dtmort <br/>";
    echo "nous avons $nb infos";
?>
```

PHP – CHAINES CARACTERES

6. Comparaison de chaînes

6.1 utilisation des opérateurs usuels de comparaison



Exemples :

```
<?php     //e4_15.php
$nb = 59;
$ch = "59 rue noire";
if ($ch==$nb) echo  "TRUE  <br/>«";
if ($ch===$nb) echo  "TRUE  <br/>";
```

Égalité , seuls les nombres sont pris en compte car une des 2 chaînes est un nombre

Pas d'égalité , car les types sont différents

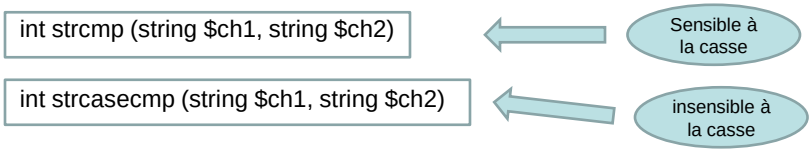
PHP – CHAINES CARACTERES

6. Comparaison de chaînes

6.2 utilisation des fonctions strcmp() , strcasecmp()

elles permettent de comparer 2 chaînes et retournent :

- -1 si $ch1 < $ch2
- 0 en cas d'égalité
- 1 si $ch1 > $ch2



PHP – CHAINES CARACTERES

6. Comparaison de chaînes

6.2 utilisation des fonctions strncmp(), strncasecmp()

elles sont identiques aux précédentes mais en limitant la comparaison aux n premiers caractères

int strncmp (string $ch1, string $ch2, int N) ← Sensible à la casse

int strncasecmp (string $ch1, string $ch2, int N) ← insensible à la casse

```
<?php    //e4_16.php
$ch1 = " Blanc ";
$ch2 = " Bleu";
$ch3 = " blanc";
echo strcmp($ch1,$ch2);
echo strcasecmp($ch1,$ch3);
echo strncmp($ch1,$ch2,2);
```


PHP – CHAINES CARACTERES

6. Comparaison de chaînes

6.2 utilisation des fonctions strnatcmp(), strnatcasecmp()

elles sont identiques aux précédentes mais en respectant l'ordre naturel

int strnatcmp (string $ch1, string $ch2) ← Sensible à la casse

int strnatcasecmp (string $ch1, string $ch2) ← insensible à la casse

```
<?php    //e4_17.php
$ch4 = " page2 ";
$ch5 = " page12";
echo strcmp($ch4,$ch5);
echo strnatcmp($ch4,$ch5);
```

PHP – CHAINES CARACTERES

6. Comparaison de chaînes

6.2 utilisation de la fonction similar_text ()

elle recherche le nombre de caractères communs à 2 chaînes (plus éventuellement le pourcentage de similarité)

Sensible à la casse

```
int similar_text (string $ch1, string $ch2 [, $pourcent])
```

```
<?php    //e4_18.php
$ch4 = " MySQL ";
$ch5 = " PgSQL";
echo  similar_text($ch4,$ch5,$pourc), "caractères communs" ;
echo  "similarité :" , $pourc, "%" ;
```


PHP – CHAINES CARACTERES

7. Transformation de chaînes en tableaux

7.1 utilisation de la fonction explode ()

elle extrait chacun des « mots » d'une chaîne et les place dans un tableau

```
array explode (string sep, string $ch [, int N ])
```

$ch : chaîne à analyser
sep : critère de séparation (souvent espace)
N : nombre max d 'éléments du tableau (le dernier contient toute la fin de la chaîne)

PHP – CHAINES CARACTERES

7. Transformation de chaînes en tableaux

7.2 utilisation de la fonction implode ()

elle retourne une chaîne composée des éléments d'un tableau séparés par un caractère donné

string implode (string sep, string $tab)

```
<?php    //e4_19.php
```


PHP- dates

1. La fonction date()
2. Le timestamp
   2.1 la fonction time()
   2.2 la fonction date() et le timestamp
   2.3 la fonction mktime() et le timestamp
   2.4 limites
3. La fonction checkdate()
4. La date en français

PHP- dates

1. La fonction date()

La fonction date permet d'afficher, le jour, la date et l'heure sur les pages WEB, qu'elles soient statiques ou créées dynamiquement.

```
<?php
$jour = date('d');
echo 'Aujourd\'hui, nous sommes le : ' . $jour;
?>
```

Affiche le numéro du jour dans le mois

Format 1 : date('lettre')

Cfr tableau slide suivant

Remarques :

1) Le plus embêtant avec date c'est que la fonction est faite pour... des anglais. Il n'y a pas moyen qu'elle affiche les jours de la semaine en français.
2) C'est l'heure du serveur qui est renvoyée, et non pas celle du client.

PHP- dates

Caractères de définition du format d'affichage.

La colonne Exemple contient les valeurs telles qu'elles sont apparues le lundi 29 Août 2005 à 0h26

Lettre	Signification	Valeurs possibles	Exemple
s	Secondes	00 à 59	53
i	Minutes	00 à 59	26
H	Heure	00 à 23	00
I	Indique si l'heure d'été est activée (1 = oui, 0 = non)	0 ou 1	1
O	Différence d'heures avec l'heure GMT (Greenwich)	-1200 à +1200	+0200
d	Jour du mois	01 à 31	29
m	Mois de l'année	01 à 12	08
Y	Année, sur 4 chiffres	Beaucoup de possibilités	2005
y	Année, sur 2 chiffres	Beaucoup de possibilités	05
L	Indique si l'année est bissextile (1 = oui, 0 = non)	0 ou 1	0
l	Jour de la semaine écrit en anglais	Sunday à Saturday	Monday
F	Mois écrit en anglais	January à December	August
t	Nombre de jours dans le mois	28 à 31	31
w	Numéro du jour de la semaine	0 (dimanche) à 6 (samedi)	1
W	Numéro de la semaine dans l'année	01 à 52	35
z	Numéro du jour de l'année	0 à 366	240

PHP- dates

1. La fonction date()

```
<?php
echo 'Aujourd\'hui, nous sommes le : ' . date('d/m/Y');
?>
```

date a créé une chaîne de caractères qui contient jour/mois/année.

En fait, vous pouvez mettre ce que vous voulez dans date, dès que la fonction rencontre une lettre qu'elle connaît elle la remplace par la valeur correspondante.
Cela veut dire que vous pouvez mettre des espaces, des tirets, ou des slashs comme j'ai fait pour séparer les éléments de date.

PHP- dates

2. Le timestamp

2.1la fonction time()

Vieux des premiers âges babyloniens, le système sexagésimal consiste à diviser le jour en 2' heures de 60 minutes de soixante secondes.

Les informaticiens ne peuvent se contenter d'un système dans lequel l'ajout d'une seconde peut amener à changer d'heure, de jour,....et même de millénaire.

Pour pallier ces difficultés, une date arbitraire a été définie , correspondant au 1er janvier 1970 00h 00m 00s. A partir de cette date, le temps est compté en secondes.

Ce nombre de secondes est nommé timestamp ou instant UNIX.

En fait, ça représente le début de l'époque où le système d'exploitation Unix a été créé.

```
<?php
 echo 'Le timestamp actuel est : ' . time();
?>
```

Cette valeur n'est pas affichée au visiteur du site.
Elle sert d'intermédiaire à d'autres fonctions pour calculer des durées, des dates passées ou futures.

PHP- dates

- Le timestamp

2.2 la fonction date() et le timestamp

Il est possible de fournir un second paramètre à date (après les lettres) : le timestamp sur lequel vous voulez obtenir des informations.

Par exemple, lorsque vous écrirez une news, il vous suffira d'enregistrer juste le timestamp, et vous serez capables grâce à ce nombre de ressortir toutes les infos possibles et imaginables dessus : le jour où la news a été postée, l'heure qu'il était etc...

```
<?php   // e8_1.php
$timestamp = 1080513608;
echo "<p>Voici plein d'infos sur mon timestamp :</p>";
echo "<ul>";
echo "<li>toto a écrit ces lignes le ". date('d/m/Y', $timestamp)."</li>";
echo "<li>Ce jour-là était un ". date('l', $timestamp)."</li>";
echo "<li>Il était exactement : ". date('H\h i\m\i\n s\s', $timestamp)."</li>";
echo "<li>Il y avait ". date('t', $timestamp)."jours ce mois-là.</li>";
echo "<li>C'était le ".date('z', $timestamp). "ème jour de l'année </li>";
echo "</ul>";
?>
```


PHP- dates

- Le timestamp

2.3 la fonction mktime() et le timestamp

Je veux le timestamp du 5 Février 1998 à 13h 45min 26s

```
<?php   //e8_2.php
$vieux_timestamp = mktime(13, 45, 26, 2, 5, 1998);
echo 'Le timestamp du 05/02/1998 à 13h 45min 26s était : ' . $vieux_timestamp;
?>
```

Utilité : calcul de la durée entre 2 dates

```
<?php   //e8_3.php
$timespasse = mktime(13, 45, 26, 2, 5, 1998);
$timejour=time();
$duree=$timejour - timepasse;
echo "entre le 05/02/1998 à 13h 45min 26s et maintenant il s'est écoulé : "
    .$duree."secondes";
?>
```

PHP- dates

- Le timestamp

  2.4 limites

- L'inconvénient de ce système est de fournir des timestamp négatifs pour des dates antérieures à 1970 !
- le timestamp est celui calculé côté serveur , il n'est pas forcément identique à celui du poste client.
- le timestamp devient de plus en plus gros, et ce nombre sera tellement gros en 2037 que ça ne marchera plus.

PHP- dates

- La fonction checkdate()

La chaîne de caractères contenue dans $_POST[" date "] est décomposée grâce à la fonction explode(). Chaque élément de la date est récupéré dans un élément de tableau. Ces données sont ensuite transmises à la fonction checkdate() dans l'ordre moi, jour, année.

```
//e8_4.php
<form method="post" action="e8_4bis.php" >
<legend>Entrez votre date de naissance JJ/MM/AAAA</legend>
<input type = "text" name = "date"/>
<input type ="submit" value = "envoi"/>
</form>
```

```
<?php   //e8_4bis.php
$date = $_POST["date"];
$tabdate = explode("/",$date);
if(!checkdate($tabdate [1],$tabdate[0],$tabdate[2]))
echo "la date n'est pas valide. recommencez <hr/> ";
else
  echo "<hr/> la date $date est valide! <hr/>";
?>
```

PHP- dates

4. La date en français

- On récupère le no du jour de la semaine avec date('j')
- On récupère le no du mois avec date('n')

```
<?php  //e8_5.php
// afficher la date en français
$semaine= array (" dimanche "," lundi "," mardi ", " mercredi ",
            " jeudi "," vendredi "," samedi ");
$mois= array (1=>" janvier"," février ", " mars ", " avril ",
             " mai ", " juin ", " juillet ", " aout ",
             " septembre ", " octobre", " novembre ", " décembre ");
echo "aujourd'hui ".$semaine[date ('w')]." ".date('j')." ".
          $mois[date('n')]." ".date('Y')."<br/>";
?>
```

Il serait préférable de définir une fonction qui réalise ce traitement

PHP- fichiers – bases de données

1. Les fichiers

2. Les DB - SGBD
   - Les SGBD relationnels
   - la structure d'une DB
   - Le langage des SGBD = le SQL
   - MySQL
   - Le rôle de PHP et MySQL

PHP- fichiers – bases de données

1.1 Ouverture d'un fichier

Mode d'ouverture

```
$id_file = fopen  (string $nomfich, string $mode);
```

'monfich.txt'
ou
'..\chemin\nomfich'
ou
'http://site.be/ch/nf'

identifiant du fichier

Mode ouverture	Explication
'r'	lecture (fichier existe !)
'r+'	lecture-écriture(début fich) (fichier existe!)
'w'	écriture début fich (créé si inexistant)
'w+'	lecture-écriture début fich (créé si inexistant)
'a'	Ajout en fin (créé si inexistant)
'a+'	ajout-lecture en fin (créé si inexistant)

Il sera utilisé comme premier paramètre de la plupart des fonctions de manipulation du fichier.

En cas d'échec à l'ouverture

$id_file = FALSE

PHP- fichiers – bases de données

1.2. Existence d'un fichier

```
If  ( file_exists ( string $nomfich ) )…
```

Renvoie 'true' ou 'false'

1.3. fermeture d'un fichier

```
fclose ($id_file );
```

1.4. Formatage des données

Un enregistrement est composé de champs. L'enregistrement correspond à une ligne se terminant par '\n'.

Les champs sont séparés par un caractère quelconque ne faisant pas partie des caractères utilisés dans les données (par ex * ou ;)

PHP- fichiers – bases de données

1.5. Lecture d'un fichier

Nous lisons un enregistrement à la fois (cfr formatage)
2 fonctions sont disponibles :

`$chaine = fgets($id_file, Nbytes)` ou `$array = fgetcsv ($id_file, $separateur)`

Lecture d'un enregistrement dont le résultat est placé :
- soit dans une chaîne
- soit dans un tableau (un champ = une case du tableau en utilisant le séparateur) .

La lecture s'arrête après :
- N bytes
- si '\n' est rencontré

En fin de fichier, les 2 fonctions renvoient la valeur 'false'

PHP- fichiers – bases de données

1.5. Lecture d'un fichier

```
$id_file=fopen($file,"r");
$i=1;
echo "<h3>Lecture du fichier \"$file\" ligne par ligne<br /> Récupération de chaque
donnée </h3> ";
echo "<table border=\"1\"> <tbody>";
echo "<tr><th>Numéro </th> <th>prenom</th> <th>nom</th> <th>date</th> </tr>";

while($ligne=fgets($id_file,100) )
  {
            $tab=explode (";",$ligne);
            $time=intval($tab[2]);
            $jour= date("j/n/Y H:i:s",$time);
            echo "<tr><td>$i</td> <th>$tab[0]</th> <th>$tab[1]</th> <th>$jour</th> </tr>";
            $i++;
  }

fclose($id_file);
echo "</tbody></table> ";
```

PHP- fichiers – bases de données

1.5. Lecture d'un fichier

```
if($id_file=fopen("livre.txt","r"))

{
   echo "<table border=\"2\"> <tbody>";
   $i=0;

   while  ($tab=fgetcsv($id_file,200,":") )
     {
              $i++;
             echo "<tr> <td>n° $i : de: $tab[0] </td> <td> <a href=\"mailto: $tab[1]\" >
             $tab[1] </a></td> <td>le: ",date("d/m/y", $tab[2])," </td></tr>";
             echo "<tr > <td colspan=\" 3 \">", stripslashes($tab[3]) ,"</td> </tr> ";
     }

    fclose($id_file);
}

 echo "</tbody></table> ";
```


PHP- fichiers – bases de données

1.6. Ecriture dans un fichier

```
fputs($id_file, $chaine,[N])
```

Si N est précisé, seuls les N premiers caractères de la chaîne seront écrits.

```
if($id_file=fopen("noms.txt","a"))
 {
     fwrite($id_file,$prenom.";".$nom.";".$date."\n");
     fclose($id_file);
  }
else { echo "Fichier inaccessible";}
```

PHP- fichiers – bases de données

2. Notions de BD - SGBD

- **Une base de données** permet d'enregistrer des données de façon organisée et hiérarchisée. Elle représente une alternative aux fichiers.
  On peut écrire dans des fichiers, mais cela devient vite très compliqué dès que le nombre de données augmente.

- Pour gérer ces données, il existe des **SGBD** qui garantissent :

  systèmes de gestion de bases de données

  + La qualité des données enregistrées (retrouve-t-on ce qu'on a enregistré ?)
  + Leur cohérence (le client de chaque commande est-il répertorié ?)
  + Leur protection en cas d'incident
  + Permettre à plusieurs utilisateurs d'y accéder simultanément
  + Contrôler l'accès aux données confidentielles
  + Offrir des bonnes performances d'accès aux applications

PHP- fichiers – bases de données

2. 1 Les SGBD relationnels

Les bases de données relationnelles trouvent leur origine dans les travaux de E.F. CODD, chercheur d'IBM, article paru en 1970.

– **MySQL** : libre et gratuit, c'est probablement le SGBD le plus connu. (1995)

– **PostgreSQL** : libre et gratuit comme MySQL, avec plus de fonctionnalités mais un peu moins connu.

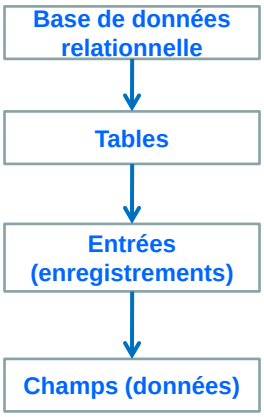
– **SQLite** : libre et gratuit, très léger mais très limité en fonctionnalités. (2000)

– **Oracle** : utilisé par les très grosses entreprises ; sans aucun doute un des SGBD les plus complets, mais il est coûteux. (1980)

– **Microsoft SQL Server** : le SGBD de Microsoft. (1989)

PHP- fichiers – bases de données

2. 2 La structure d'une DB



PHP- fichiers – bases de données

Exemple table Livres

champ

Livres

IDLivre	Prix	Titre	CodeGenre
1	40	Le glaive magique	BD
3	40	Gaffes en gros	BD
4	40	Lagaffe nous gâte	BD
5	45	QRN sur Bretzelburg	BD
6	80	Tour de manège	RG
7	45	Le spectre aux balles d'or	BD
8	30	La bonne chanson	Po
9	50	La jeune Parque	Po
10	50	Michel Strogoff	Ro
11	50	La Serpe d'or	BD
12	70	Toujours aimer	RG
13	70	Toujours aimante	RG
14	72	Toujours aimé	RG

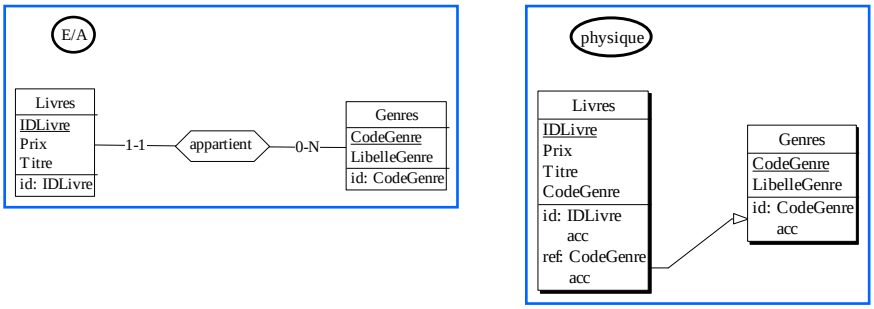
Entrée (enregistrement)

PHP- fichiers – bases de données

Exemple de structure de tables dans une DB

Les bases de données relationnelles trouvent leur origine dans les travaux de E.F. CODD, chercheur d'IBM, article paru en 1970.

Les données sont organisées sous forme de tables (tables), colonnes (columns), d'identifiants primaires (primary keys) et de colonnes de référence (foreign keys).



PHP- fichiers – bases de données

Exemple de structure de tables dans une DB

Livres

IDLivre	Prix	Titre	CodeGenre
1	40	Le glaive magique	BD
3	40	Gaffes en gros	BD
4	40	Lagaffe nous gâte	BD
5	45	QRN sur Bretzelburg	BD
6	80	Tour de manège	RG
7	45	Le spectre aux balles d'or	BD
8	30	La bonne chanson	Po
9	50	La jeune Parque	Po
10	50	Michel Strogoff	Ro
11	50	La Serpe d'or	BD
12	70	Toujours aimer	RG
13	70	Toujours aimante	RG
14	72	Toujours aimé	RG

Genres

CodeGenre	LibelleGenre
BD	Bande Dessinée
Po	Poésie
RG	Roman de gare
Ro	Roman
SF	Science Fiction

Clé primaire

Clé étrangère

Clé primaire

PHP- fichiers – bases de données

2. 3 Le langage des SGBD = le SQL

Structured Query Language

Le **SQL** permet de communiquer avec le SGBD.

Ce langage permet à l'utilisateur de créer des tables, de leur ajouter des colonnes, d'y ranger des données, de les modifier, de consulter les données, de définir les autorisations d'accès.

On y décrit les données que l'on recherche en spécifiant des conditions de sélection mais on ne spécifie pas le moyen de les obtenir, décision laissée à l'initiative du SGBD.

Exemple de commande SQL:

```
SELECT Titre FROM Livres
```


PHP- fichiers – bases de données

Code de création de la DB Biblio en SQL

Structured Query Language

```
create database biblio;
use biblio;

create table Genres (
    CodeGenre char(3) not null,
    LibelleGenre char(30) not null,
    constraint ID_Genres_ID primary key (CodeGenre));

create table Livres (
    IDLivre char(10) not null,
    Prix int not null,
    Titre char(30) not null,
    CodeGenre char(3) not null,
    constraint ID_Livres_ID primary key (IDLivre));
```

PHP- fichiers – bases de données

Code de création de la DB Biblio en SQL

Structured Query Language

```
alter table Livres add constraint FKappartient_FK
    foreign key (CodeGenre)
    references Genres (CodeGenre);



create unique index ID_Genres_IND
    on Genres (CodeGenre);

create unique index ID_Livres_IND
    on Livres (IDLivre);

create index FKappartient_IND
    on Livres (CodeGenre);
```


PHP- DB

2. 4 MySQL

Enregistrement des données dans MySQL

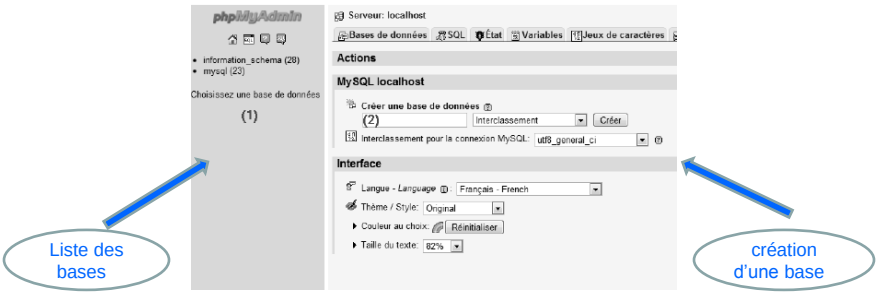
- Quand MySQL enregistre des tables, il les enregistre **dans des fichiers** !
- Par exemple, avec MySQL sous Windows, si vous utilisez WAMP, les fichiers où sont stockées les informations se trouvent dans **C:\wamp\mysql\data.**
- Dans la pratique, **on n'ira jamais toucher à ces fichiers directement**. On demandera TOUJOURS à MySQL de manipuler les tables à l'aide du SQL.

PHP- fichiers – bases de données

Comment accéder à la base de données MySQL ?

A) soit en utilisant une interface d'administration pour MySQL : phpMyAdmin

phpMyAdmin est livré avec WAMP.

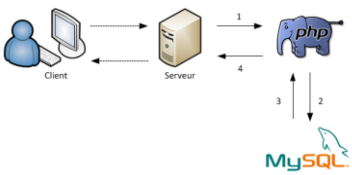


page d'accueil de phpMyAdmin

PHP- fichiers – bases de données

Comment accéder à la base de données MySQL ?

B) soit en exécutant les requêtes SQL en PHP



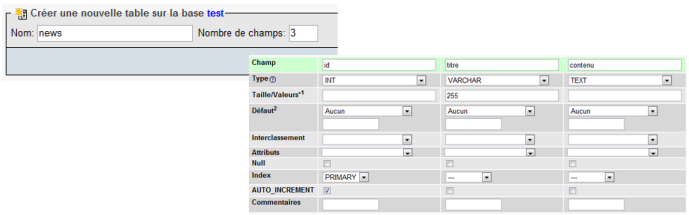
1. la requête est préparée dans le code PHP
2. PHP rencontre la requête SQL, il la transmet à MySQL.
   En effet, le code PHP contient à un endroit « Va demander à MySQL de consulter ou modifier telles données ».
3. MySQL exécute la requête que PHP lui avait soumise et lui renvoie le résultat;
4. PHP récupère le résultat, le met en forme et renvoie le résultat au serveur.

PHP- fichiers – bases de données

Utilisation de phpMyAdmin

- création d'une BD

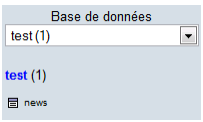
- création d'une table



PHP- fichiers – bases de données

Utilisation de phpMyAdmin

- Liste des tables

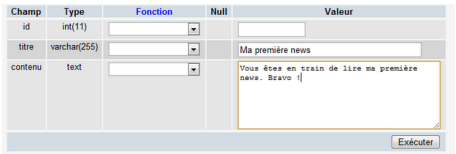


- structure de la table

PHP- fichiers – bases de données

Utilisation de phpMyAdmin

- insertion d'une entrée



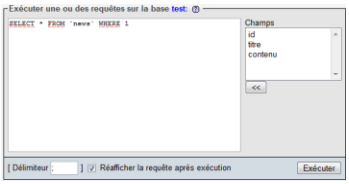
- afficher le contenu d'une table

			id	titre	contenu
☐	✎	✕	1	Ma première news	Vous êtes en train de lire ma première news. Bravo…
☐	✎	✕	2	Autre news	Ceci est une autre news !
☐	✎	✕	3	Exclusif !	Ceci est une news !

PHP- fichiers – bases de données

Utilisation de phpMyAdmin

- exécuter une requête SQL



PHP- fichiers – bases de données

La clause select en SQL

SELECT Titre, Prix, Genre **FROM** Livres;

SELECT * **FROM** Livres **WHERE** IDLivre = 7;

Livres

IDLivre	Prix	Titre	CodeGenre
7	45	Le spectre aux balles d'or	BD

SELECT Titre, Prix, CodeGenre **FROM** Livres **WHERE** CodeGenre = ″BD″;

SELECT * FROM Livres **WHERE** IDLivre = 7 **AND** Prix > 50;

PHP- fichiers – bases de données

La clause select en SQL

SELECT * FROM Livres **WHERE** Prix **IN** (40, 50, 72);

Livres

IDLivre	Prix	Titre	CodeGenre
1	40	Le glaive magique	BD
3	40	Gaffes en gros	BD
4	40	Lagaffe nous gâte	BD
9	50	La jeune Parque	Po
10	50	Michel Strogoff	Ro
11	50	La Serpe d'or	BD
14	72	Toujours aimé	RG

SELECT * FROM Livres **WHERE** (Prix **BETWEEN** 40 **AND** 50) **AND** (CodeGenre = 'BD');

PHP- fichiers – bases de données

La clause select en SQL

SELECT Titre, Livres.CodeGenre, LibelleGenre, Prix **FROM** Livres, Genres **WHERE** Livres.CodeGenre = Genres.CodeGenre;

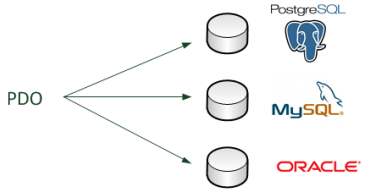
Titre	Livres.CodeGenre	LibelleGenre	Prix
Le glaive magique	BD	Bande Dessinée	40
Gaffes en gros	BD	Bande Dessinée	40
Lagaffe nous gâte	BD	Bande Dessinée	40
QRN sur Bretzelburg	BD	Bande Dessinée	45
Tour de manège	RG	Roman de gare	80
Le spectre aux balles d'or	BD	Bande Dessinée	45
La bonne chanson	Po	Poésie	30
La jeune Parque	Po	Poésie	50
Michel Strogoff	Ro	Roman	50
La Serpe d'or	BD	Bande Dessinée	50
Toujours aimer	RG	Roman de gare	70
Toujours aimante	RG	Roman de gare	70
Toujours aimé	RG	Roman de gare	72

PHP- fichiers – bases de données

2.5 Connexion à une BD MySQL avec PHP

PHP propose plusieurs moyens de se connecter à une base de données MySQL.

- L'extension mysql_ : ce sont des fonctions considérées comme obsolètes.
  .
- L'extension mysqli_ : ce sont des fonctions améliorées (extension de PHP)
- L'extension PDO : c'est un outil complet qui permet d'accéder à n'importe quel type de base de données. On peut donc l'utiliser pour se connecter aussi bien à MySQL que PostgreSQL ou Oracle.



PHP- fichiers – bases de données

Les commandes Mysqli_ : connexion à la DB

```php
<?php
if($bdd = mysqli_connect('localhost', 'root', '', 'base'))
{
    // Si la connexion a réussi, rien ne se passe.
}
else // Mais si elle rate…
{
    echo 'Erreur'; // On affiche un message d'erreur.
}
?>
```

$bdd orrespond à une variable où seront stockées les informations de la base de données
localhost correspond au serveur SQL.
root correspond au nom d'utilisateur pour se connecter au serveur SQL.
mot_de_passe correspond au mot de passe pour le serveur SQL !
base correspond à votre base de données du serveur SQL.

PHP- fichiers – bases de données

Les commandes Mysqli_ : requête simple

```php
<?php
$resultat = mysqli_query($bdd, 'requete');
?>
```

Exemple :

```php
<?php
$resultat = mysqli_query($bdd, 'SELECT * FROM membres LIMIT 0, 10');
?>
```

mysqli_query — Exécute une requête sur la base de données et le résultat est stocké dans la variable ***$resultat***

PHP- fichiers – bases de données

Les commandes Mysqli_ : affichage du résultat d'une requête simple

```php
<?php
while($donnees = mysqli_fetch_assoc($resultat))
{
    echo $donnees['id'];
    echo "\n";
    echo $donnees['pseudo'];
}
?>
```

mysqli_fetch_assoc($resultat) — Retourne un tableau associatif de chaînes qui contient la ligne lue dans le résultat ***$resultat***, ou bien **FALSE** s'il ne reste plus de lignes à lire. La fonction déplace le pointeur interne de données.

Le tableau associatif ***$donnees*** utilise les noms des colonnes demandées dans la requête.

```php
<?php
mysqli_free_result($resultat);
?>
```

Fermer le curseur permet de libérer les données récupérées lors d'une requête SQL.

PHP- fichiers – bases de données

Les commandes Mysqli_ : résumé

```php
<?php
// include du fichier de connexion à SQL.

$resultat = mysqli_query($bdd, 'SELECT * FROM membres LIMIT 0, 10');
while($donnees = mysqli_fetch_assoc($resultat))
{
    echo $donnees['id'];
    echo "\n";
    echo $donnees['pseudo'];
}
mysqli_free_result($resultat);
?>
```

PHP- fichiers – bases de données

Les commandes Mysqli_ : commandes supplémentaires

```php
<?php
// include de la connexion à la BDD.
$req = mysqli_query($bdd, 'SELECT * FROM commentaires');
$nb = mysqli_num_rows($req);

echo 'Il y a ' . $nb . ' commentaire(s).';
?>
```

mysqli_num_rows — Retourne le nombre de lignes dans un résultat

http://www.php.net/manual/fr/mysqli.summary.php

PHP – Méthodes et principes d'échanges de données

1. Le fichier de paramétrage
2. Le FTP
3. Les cookies
4. Les variables de session
5. Les e-mails

LE FICHIER DE PARAMETRAGE

Généralités - Définition

- Un **fichier INI** est un **fichier de configuration** dans un **format de données** introduit par les systèmes d'exploitation Windows en 1985. Par convention les noms de ces fichiers portent l'**extension** « .ini ».
- L'utilisation de ces fichiers s'est étendue sur d'autres systèmes d'exploitation tels que **Unix**.

Généralités - Format

- Les fichiers INI sont des **fichiers textes**: ils peuvent être manipulés avec un **logiciel** courant de type **éditeur de textes**.
- Les fichiers sont divisés en **sections**. Chaque section comporte un certain nombre de paramètres de **configuration**. Chaque section commence par un titre placé entre **crochets** « [» et «] ».
- La valeur de chaque paramètre de configuration est indiquée par une formule: **paramètre = valeur**.
- Les fichiers peuvent contenir des **commentaires**. Les commentaires sont souvent utilisés pour décrire les paramètres et les valeurs à introduire. Ils sont précédés d'un **point-virgule** ou plus rarement d'**un dièse**.

Implémentation en PHP – Exemple de fichier

```
[Base de données]
serveur = localhost
login = root
pwd =

[Coordonnées]
nom = ANCKAERT Michaël
fonction = Informaticien : Analyste-programmeur
adresse = rue du Spinois, 55
cp = 6224
localite = WANFERCEE-BAULET
tel = 071/81.78.55
gsm = 0485/99.33.02
web = http://www.easydeal.be
```

Implémentation en PHP – Instruction d'accès

- **Description**
  - array **parse_ini_file** (string $filename [, bool $process_sections = false [, int $scanner_mode = INI_SCANNER_NORMAL]])
  - **parse_ini_file()** charge le fichier **filename** et retourne les configurations qui s'y trouvent sous forme d'un **tableau associatif**.
  - La structure des fichiers de configuration lus est similaire à celle de **php.ini**.

Implémentation en PHP – Instruction d'accès

- **Liste de paramètres**
  - **Filename** Le nom du fichier de configuration à analyser.
  - **process_sections** En passant le deuxième paramètre optionnel à **process_sections**, vous obtiendrez un tableau multidimensionnel avec les noms des sections. La valeur par défaut de ce paramètre est **FALSE**.
  - **scanner_mode** Peut être **INI_SCANNER_NORMAL** (défaut) ou **INI_SCANNER_RAW**. Si **INI_SCANNER_RAW** est fourni, alors les valeurs en option ne seront pas analysées.
- **Valeurs de retour**
  - La configuration est retournée sous la forme d'un tableau associatif en cas de succès, et **FALSE** si une erreur survient.

Implémentation en PHP – Exemple d'implémentation

```
function ouvrir()
{
      $myIniFile = parse_ini_file ("../config.ini",TRUE);

      $serveurDb=($myIniFile["Base de
données"]["serveur"]);

      $loginDb=($myIniFile["Base de données"]["login"]);

      $pwdDb=($myIniFile["Base de données"]["pwd"]);

      $_SESSION["dbh"] = serialize(new
Connection($serveurDb, $loginDb, $pwdDb));
}
```


LE FTP

Généralités

- Le **protocole FTP** (*File Transfer Protocol*) est, comme son nom l'indique, un protocole de transfert de fichier.
- Le protocole FTP définit la façon selon laquelle des données doivent être transférées sur un réseau TCP/IP.

Objectifs

- Le protocole FTP a pour objectifs de :
  - permettre un **partage de fichiers** entre machines distantes;
  - permettre une indépendance aux systèmes de fichiers des **machines clientes et serveur**;
  - permettre de **transférer** des données de manière **efficace**.

Format

- Pour effectuer une connexion de type FTP, nous avons besoin de:
  - Un **nom de serveur**;
  - Un **login**;
  - Un **mot de passe**;
  - Un **port de communication** (21) éventuellement sécurisé comme les protocoles **SSL** ou **TLS**.

Utilisation - Connexion

- Le mécanisme est finalement équivalent à une connexion sur une base de donnée MySQL:
  - Ouverture de la communication;
  - Connexion grâce au login en utilisant les paramètres fournis par l'hébergeur Internet.
- Mais forcément, les commandes seront différentes. La première chose est d'établir une **liaison avec le serveur**.

Utilisation - Connexion

- **ftp_connect**(string $host [, int $port = 21 [, int $timeout = 90]])

  où:
  - **$host est l'adresse du serveur FTP**.
    Ce paramètre ne doit jamais avoir de slash final et ne doit pas être préfixé par *ftp://*.
  - **$port**
    Ce paramètre spécifie un numéro de port alternatif pour la connexion. S'il est omis ou définie à zéro, alors le port par défaut utilisé sera 21.
  - **$timeout**
    Ce paramètre spécifie le délai de connexion pour toutes les opérations de sous séquences du réseau. S'il est omis, la valeur par défaut sera 90 secondes. Le délai de connexion peut être modifié et interrogé à n'importe quel moment avec les moments **ftp_set_option()** et **ftp_get_option().**

Utilisation - Connexion

- **<?php**
  $ftp_server = "ftp.ybet.be";
  $connect = ftp_connect($ftp_server);
  ?>

- Ensuite, on établit une connexion comme utilisateur en reprenant le **nom utilisateur** et le **mot de passe**;

Utilisation - Connexion

- Ceci donne pour la connexion en langage PHP:

```
<?php
$login="YBET";
$password="PS";
$login_result = ftp_login($connect, $login,
 $password);
?>
```


Utilisation - Connexion

- Il faut s'assurer que la connexion est effective:

```
<?php
if(ftp_login($connect, $login, $password))
    echo "Connecté en tant que $login sur
$ftp_server</br>";
else echo "Connexion impossible en tant
que ".$login."</br>";
?>
```

Utilisation - Connexion

- Une fois la connexion établie, nous pouvons effectuer nos échanges d'informations et à la fin fermer la connexion:

```
<?php
ftp_close($connect);
?>
```

$connect est le paramètre récupéré de ftp_connect

Utilisation - Gestion

- **Le transfert de fichier**

```
<?php
$ftp_server = "ftp.ybet.be";
$login="YBET";
$password="PS";
$connect = ftp_connect($ftp_server);
if(ftp_login($connect, $login, $password))
        echo "Connecté en tant que $login sur $ftp_server</br>";
else echo "Connexion impossible en tant que ".$login."</br>";

$destination_file="/www/tests.php";
$source_file="tests.php";
$upload = ftp_put($connect, "$destination_file", "$source_file", FTP_ASCII);

if (!$upload) echo "Le transfert Ftp a échoué!";
else echo "Téléchargement de ". $source_file." sur ". $ftp_server." en ".$destination_file;
?>
```

FTP doit être renseigné, il prend la valeur **FTP_BINARY** ou **FTP_ASCII**.

Utilisation - Gestion

- **La création d'un dossier**

```
<?php
$directory="/www/directory";
$dossier=ftp_mkdir($connect,$directory);
?>
```

- **La modification du répertoire courant**

```
<?php
$directory="/www/hardware";
$dossier=ftp_chdir($connect,$directory);
?>
```


Utilisation - Gestion

- **Le dossier de destination courant**

```
<?php
$dossier=ftp_pwd($connect);
?>
```

- **Les fichiers dans le répertoire courant**

```
<?php
$tableau=ftp_nlist($connect,$repertoire);
$nombre=count($tableau);
echo "nombre: ".$nombre."</br>";
$i=0;
while ($i<21)
{
        echo $tableau[$i]."</br>";
        $i=$i+1;
}
?>
```

La commande **FTP_RAWLIST()** est identique mais renvoie également les informations sur le fichier sous forme de tableau.

Utilisation - Gestion

- **La création d'un dossier**

```php
<?php
$dir="directory";
$valeur=ftp_mkdir($connect, $dir));
?>
```

- **La suppression d'un dossier**

```php
<?php
$dir="directory";
$valeur=ftp_rmdir($connect, $dir));
?>
```


Utilisation - Gestion

- **La taille d'un fichier**

```php
<?php
$directory="/www/hardware";
$dossier=ftp_chdir($connect,$directory);

$dossier=ftp_pwd($connect);
echo "dossier courant ".$dossier;
$tableau=ftp_nlist($connect,$repertoire);
$nombre=count($tableau);
echo "nombre: ".$nombre."</br>";
$i=0;
while($i<$nombre)
{
        echo $tableau[$i]." - ".ftp_size($connect,$tableau[$i])."</br>";
        $i=$i+1;
}
?>
```

- **La suppression d'un fichier**

```php
<?php
$file="tests.php";
$valeur=ftp_delete($connect,$file);
?>
```

Utilisation - Gestion

- **La gestion des droits**

```
<?php
$fichier="tests.php";
$int=0644;
// la valeur est constituée de 4 chiffres en octal (de 0 à 7).
$valeur=ftp_chmod($connect,$int,$fichier);
?>
```


LES COOKIES

PHP – COOKIES

1. Définition

Le cookie est un petit fichier pouvant être écrit par un script sur l'ordinateur du visiteur d'un site.

Limites:

- Un site déterminé ne peut écrire que max 20 cookies sur le même poste client.
- Chaque cookie ne peut pas dépasser 4ko
- Un cookie n'est accessible que par le site qui l'a écrit

2. Ecriture des cookies

Attention: aucun contenu HTML ne peut avoir été envoyé avant l'écriture d'un cookie

PHP – COOKIES

```
$boolean = setcookie($nomcookie, $valeur, $datefin, $chemin,
    $domaine, $securite);
```

$nomcookie	nom que l'on donne au cookie (chaîne)
$valeur	valeur que l'on donne au cookie (chaîne)
$datefin	entier (timestamp) servant à déterminer la date d'expiration du cookie. Si absent -> valable uniquement pendant la session exemple: $datefin = time()+86400; // valable 24 heures
$chemin	chemin d'accès des scripts autorisés à accéder au cookie (facultatif)
$domaine	nom complet du domaine qui peut accéder au cookie (facultatif) ex : www.isat.be
$securite	TRUE transmis par https, FALSE par http (défaut)

PHP – COOKIES

NB setcookie() renvoie FALSE en cas de problème

Exemple:

```
$ok=setcookie("monCookie",  "42", time()+86400, "/client/fact");
```

3. Supprimer un cookie

   Il suffit d'appeler **setcookie**($nomCookie) sans donner de valeur

4. Rendre un cookie innaccessible

   Il suffit de lui donner une date d'expiration antérieure à la date actuelle.

   Exemple: `$ok=setcookie("monCookie",  "42", time()-3600);`

PHP – COOKIES

5. Plusieurs valeurs sous un même nom de cookie

```
setcookie("perso[nom]",  "Ecu-de-Chêne", time()+86400, );
setcookie("perso[prenom]",  "Torin", time()+86400, );
```

Attention: les clés sans guillemets.

Ou bien:

À partir d'un tableau associatif:

```
foreach($tablassoc as $cle=>$valeur)
   setcookie("client[$cle]", $valeur, time()+86400, );
```

PHP – COOKIES

6. Lire un cookie

Les cookies sont accessibles via la variable superglobale **$_COOKIE.**

Il s'agit d'un tableau associatif ayant comme clé le nom du cookie.

Exemple:

```
$nom = $_COOKIE["nom"];
$prenom = $_COOKIE["client"]["prenom"];
foreach($_COOKIE["client"] as $cle=>$valeur)
    $tablassoc[$cle] = $valeur;
```


LES VARIABLES DE SESSION

PHP – SESSIONS

1. Définition

   Mécanisme qui permet de **conserver des informations provenant d'une page web**, pour être utilisées dans d'autres pages du site par un même visiteur. Aucun autre visiteur n'a accès à ces informations.

2. Mise en œuvre des sessions

   - En début de chaque page ayant accès aux variables de session, il faut appeler la fonction $ok=**session_start()**;
   - Chaque utilisateur reçoit un **sessionID** (**SID**) qui sera transmis d'une page à l'autre de deux manières différentes:
     - Soit dans un cookie
     - Soit en étant ajouté à l'URL de la page

PHP – SESSIONS

- Le script définit les variables de session et y accède en utilisant le tableau associatif superglobal **$_SESSION** dont les clés sont les noms des variables.
  Exemple: **$_SESSION**[**'mavar'**] = 25;
- Pour transférer certains objets, il faudra sérialiser et désérialiser ceux-ci (Voir cours de POO PHP de 2ème).
- Fermeture de la session et destruction des variables de session

  session_unset(); **session_destroy**();

Remarque

Les variables de session sont stockées sur le serveur et non sur le poste client. Elles sont en général stockées dans un dossier **/tmp** ou **/sessions**. Ces fichiers ont pour nom **sess_xxxxxxxx** (où xxxxxxx est le sessionID)

PHP – SESSIONS

4. Session avec cookies

Il s'agit d'une manière simple de transmettre le sessionID d'une page à l'autre du site.
Pour utiliser cette méthode, il faut 3 conditions:

- php.ini → **session.use_cookies *on***
- Le navigateur doit accepter les cookies
- Chaque page du site commence par l'appel de **session_start**()

NB Si dans php.ini on a configuré **session.auto_start *on***, le serveur générera automatiquement un **session_start**() pour toutes les pages du site.

PHP – SESSIONS

5. Session sans cookies

Le nom de la session par défaut est PHPSESSID et l'identifiant aléatoire de la session est contenu dans la constante **SID** sous la forme **PHPSESSID**=xxxxxxxxxxxxxxx

La transmission du SID se fera par son ajout dans l'url de la manière suivante:

```
<a href='page2.php?<?php echo SID ?>'> vers page2</a>
```

6. Autres fonctions pour les sessions

$name = **session_name**();	Retourne le nom de la session
session_name('nouveauNom');	Définit un nouveau nom pour la session
$id = **session_id**();	Retourne l'identifiant de la session
session_id('newID');	Définit un nouveau sessionID

PHP – SESSIONS

$path = **session_save_path**();	Retourne le chemin du serveur où sont stockées les données de la session
session_write_close();	Ecrit les variables session sur le serveur et ferme la session

PHP – SESSIONS

7. Afficher toutes les variables de session

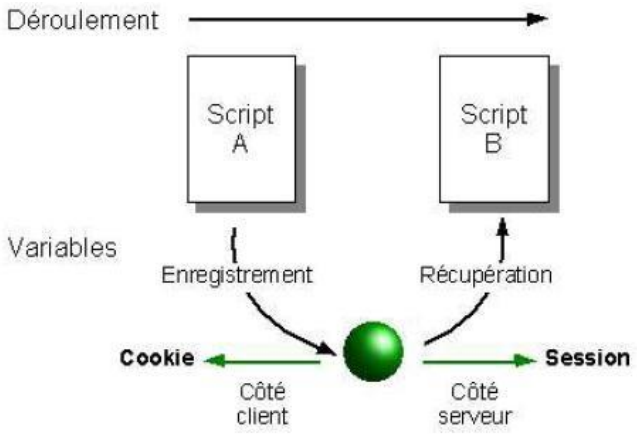
```
<?php
  // begin the session
  session_start();

  // loop through the session array with foreach
  foreach($_SESSION as $key=>$value)
  {
     // and print out the values
     echo 'The value of $_SESSION['."'".$key."'".'] is '."'".$value."'".'<br/>';
  }
?>
```

PHP – SESSIONS

8. Comment sont gérées les variables de session?

```php
<?php
    /*** begin a session ***/
    session_start();

    if(isset($_COOKIE['PHPSESSID']))
        echo 'The session ID has been store in a cookie';

    if(defined(SID))
        echo 'The session ID has been stored in the query string';
?>
```


PHP – SESSIONS



LES E-MAILS

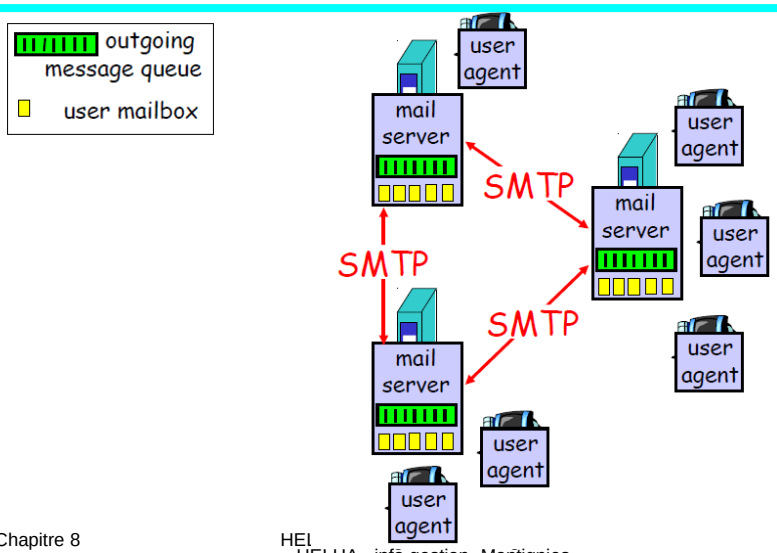
PHP – E-MAIL – GENERALITES – INTERVENANTS

- Le mécanisme de transport d'un e-mail nécessite l'intervention:
  - d'un **agent utilisateur**;
  - d'un **serveur mail**;
  - du **protocole SMTP** (Simple Mail Transfer Protocol).
- L'agent utilisateur:
  - est un **lecteur de mails**;
  - va **composer, éditer et lire** les e-mails;
  - utilise une boîte e-mail comme Eudora, Outlook, elm, Mozilla Thunderbird.

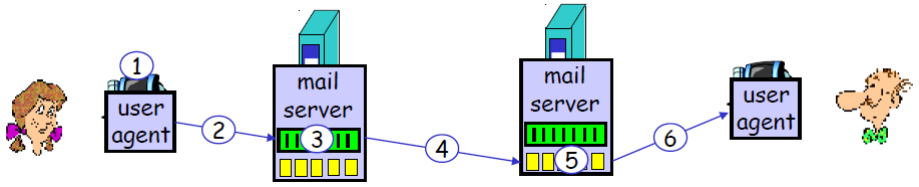
PHP – E-MAIL – GENERALITES – INTERVENANTS

- Le serveur mail:
  - **Reçoit, envoie et stocke** les e-mails;
  - est conscient que c'est la boîte e-mail de l'agent utilisateur qui contient (copie locale) les e-mails d'un utilisateur.
- Le protocole SMTP:
  - permet d'**envoyer** un message (partie d'e-mail) d'un **agent utilisateur** vers son **serveur** mail;
  - permet d'**envoyer** un message d'un **serveur** mail vers un **autre**;
  - utilise un **transfert direct** entre les intervenants.

PHP – E-MAIL – GENERALITES – FONCTIONNEMENT



PHP – E-MAIL – GENERALITES – FONCTIONNEMENT



1) Sophie rédige un e-mail et l'envoie à l'adresse: roger.rabbit@warnerbross.com;
2) L'agent utilisateur de Sophie envoie l'e-mail dans la queue de messages de son serveur mail;
3) Le côté client du SMTP ouvre une connexion avec le serveur mail de Roger.
4) Le côté client du SMTP envoie le message de Sophie via TCP;
5) Le serveur mail de Roger réceptionne et place l'e-mail dans la boîte e-mail de Roger;
6) Roger invoque son agent utilisateur pour lire l'e-mail qu'il a reçu.

PHP – E-MAIL – GENERALITES – PROTOCOLE SMTP

- Ce protocole:
  - utilise le port (part défaut) **25** ou **2525**;
  - **fonctionne en 3 phases**:
    - **Handshaking**: ("poignée de main" en français) est un processus automatisé de négociation qui établit les paramètres d'une communication entre deux entités avant que la communication commence.
    - **Transfert** des messages;
    - **Fermeture** du transfert.
  - utilise **une interaction commande/réponse**:
    - La commande est du texte codé en ASCII;
    - La réponse est exprimée par un code de retour et du texte.
  - Les messages sont codés sur 7 bits ASCII.
  - peut être utilisé en **mode sécurisé** (SSL **465** /TLS **587**).

PHP – E-MAIL – GENERALITES – PROTOCOLES POP3/IMAP

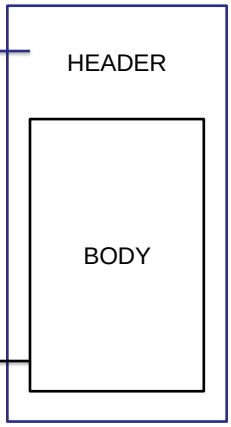
- Le protocole POP3 (Post Office Protocol):
  - utilise le port (part défaut) **109/110**;
  - effectue le **transfert du serveur mail vers l'agent utilisateur avec autorisation** (maintien de copies).
- Le protocole IMAP (Internet Message Access Protocol):
  - utilise le port (par défaut) **143**;
  - est un **protocole alternatif** au protocole POP3 mais offrant **beaucoup plus de possibilités**:
    - permet de gérer plusieurs accès simultanés;
    - permet de gérer plusieurs boîtes aux lettres;
    - permet de trier le courrier selon plus de critères.
- IMAP peut être utilisé en mode sécurisé (SSL **993**).

PHP – E-MAIL – FORMAT

- L'e-mail (electronic mail):
  - est composé:
    - d'une **partie locale**, identifiant généralement une personne (lucas, Jean.Dupont, joe123) ou un nom de service (info, vente, postmaster);
    - le caractère séparateur @ (arobase);
    - un **nom de domaine**: identifiant généralement l'entreprise hébergeant la boîte électronique (exemple.net, exemple.com, exemple.org).
  - grâce à son **nom de domaine identifie le serveur mail** auquel doit être acheminé un message via le protocole SMTP. La transformation du nom de domaine en adresse IP se fait grâce au système de résolution de noms DNS (Domain Name Server).

PHP – E-MAIL – CONTENU

- L'entête
  - reprend des informations comme:
    - From: e-mail de l'émetteur;
    - To: e-mail du récepteur;
    - Subject: Objet de l'e-mail (tr/re).
  - peut être complété par:
    - cc/cci;
    - une ou des pièces jointes.
- Le corps
  - est le message codé en ASCII et/ou HTML.



PHP – E-MAIL – CONFIGURATION DE WAMP

- Si vous n'avez pas configurer l'utilisation du protocole SMTP lors de l'installation de WAMP, vous pouvez le faire en configurant le **php.ini**.

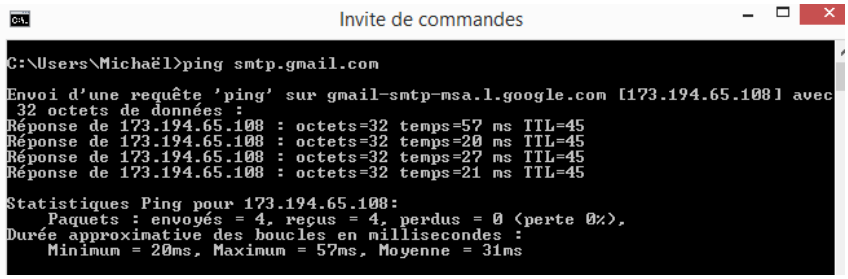
```
[mail function]
; For Win32 only.
; http://php.net/smtp
SMTP = smtp.gmail.com
; http://php.net/smtp-port
smtp_port = 465 (25=None / 465=SSL / 587=TLS)

; For Win32 only.
; http://php.net/sendmail-from
sendmail_from = m.anckaert@gmail.com
```

PHP – E-MAIL – CONFIGURATION DE WAMP

- Si vous avez un message d'erreur qui mentionne STARTTLS, vérifiez que vous avez les options suivantes cochées:
  - Pour PHP dans ses extensions:
    - php_openssl;
    - php_sockets.
  - Pour Apache dans ses modules:
    - ssl_module.

PHP – E-MAIL – CONFIGURATION DE WAMP

- Vérifiez que vous pouvez accéder à votre serveur SMTP grâce à la **commande ping**:



PHP – E-MAIL – CONFIGURATION DE WAMP

- Pour vérifier les ports pris en charge utilisez le **client telnet** (à installer si non présent):

- Utilisez la commande <CTRL>+<$> pour sortir de la connexion établie et <q> pour sortir du client telnet.

PHP – E-MAIL – CONFIGURATION DE SENDMAIL

- Si vous ne parvenez pas à envoyer un e-mail, installez **sendmail** dans c:\wamp et configurez sendmail.ini.

```
…
smtp_server=smtp.gmail.com
smtp_port=465
…
smtp_ssl=auto
…
default_domain=gmail.com
…
auth_username=m.anckaert@gmail.com
auth_password=**********
…
```

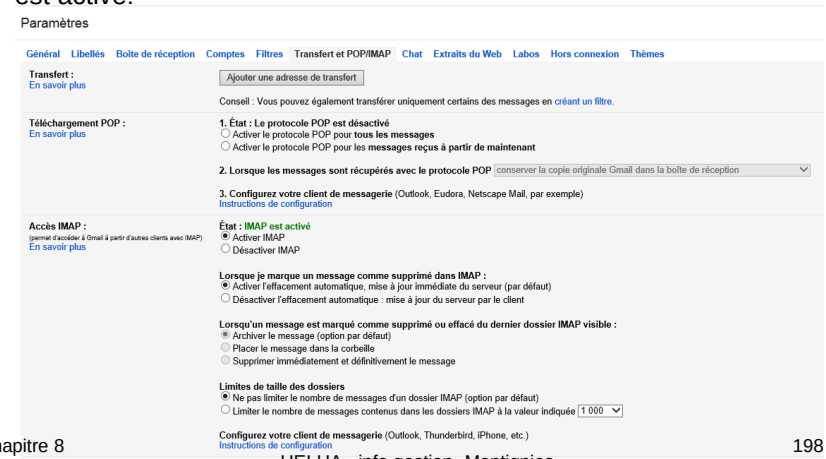
PHP – E-MAIL – CONFIGURATION DE SENDMAIL

- Vous devez également configurer le php.ini:

```
…
sendmail_path="C:\wamp\sendmail\sendmail.exe -t"
…
```


PHP – E-MAIL – CONFIGURATION DU COMPTE E-MAIL

- Suivant votre compte, vous devez vérifier que le transfert vers IMAP est activé:



PHP – E-MAIL – CONFIGURATION DE STUNNEL

- Si vous utilisez Windows 8, vous devez simuler un serveur SMTP avec un outil comme **stunnel**. Il vous faudra dès lors configurer le php.ini comme suit:

```
[mail function]
; For Win32 only.
; http://php.net/smtp
SMTP = localhost
; http://php.net/smtp-port
smtp_port = 25

; For Win32 only.
; http://php.net/sendmail-from
; sendmail_from = m.anckaert@gmail.com
```


PHP – E-MAIL – CONFIGURATION DE STUNNEL

- Stunnel simule un serveur mail de type gmail:

```
[gmail-smtp]
client = yes
accept = 127.0.0.1:25
connect = smtp.gmail.com:465
```

PHP – E-MAIL – OUTILS A UTILISER

- PHP permet d'utiliser la **fonction mail()**;
- Afin d'utiliser cet outil à bon escient vous devez respecter quelques bons principes comme **prévoir deux versions du message** (Texte/HTML);
- Il existe un tutoriel bien rédigé et complet à ce sujet: http://fr.openclassrooms.com/informatique/cours/e-mail-envoyer-un-e-mail-en-php
- A défaut de ce tutoriel, vous pouvez consulter l'aide de php: http://www.php.net/manual/fr/function.mail.php

PHP - SECURITE

1. La sécurité des données entrantes/sortantes
2. Les listes
3. Les fichiers .htaccess

LA SECURITE DES DONNEES ENTRANTES/SORTANTES

PHP – LA SECURITE DES DONNEES ENTRANTES/SORTANTES

1. Principe de l'architecture 3-tier

Une architecture *3-tier* et plus généralement ***n**-tier* est un **système composé de n couches** (*3-tier* ou *3 étages pour le WEB)*.
En fait, on distingue:

- d'une part, le **navigateur** de l'internaute (Firefox, IE, Safari, ...) qui a la charge **d'afficher correctement les donnée**s transmises par le serveur;
- la couche **serveur** avec PHP par exemple qui **traite les données à la fois venant du navigateur mais aussi celles issues de la BDD** (c'est en quelque sorte le messager qui s'assure du transfert des données);
- l'étage qui permet le **stockage et la conservation des informations**: c'est la **BDD** (exemple: MySQL, PostGreSQL...).

PHP – LA SECURITE DES DONNEES ENTRANTES/SORTANTES

Les flèches bleues représentent les flux de données qui transitent depuis l'internaute jusqu'à la base de données et les flèches vertes les données transitant depuis la base de données jusqu'au navigateur de l'internaute. Ces transferts de données sont schématisés par les flèches rouges.

PHP – LA SECURITE DES DONNEES ENTRANTES/SORTANTES

2. Méthodes de sécurité dans le code

Nous avons donc deux types de flux de données à sécuriser:

- Les **données entrantes**: dans un premier temps, nous allons devoir nous occuper de sécuriser les données qui proviennent du membre et donc de formulaires avant de pouvoir les entrer dans la BDD.
  Ce chapitre sera vu en 2ème année au cours de base de données.
  Retenons qu'il sera préférable d'utiliser des requêtes préparées pour lutter contre **l'injection SQL**.

PHP – SECURITE DES DONNEES ENTRANTES/SORTANTES

- Les **données sortantes**: puis, nous verrons comment sécuriser l'affichage des données issues de la BDD.
  Nous avons déjà utilisé ces techniques notamment avec les valeurs des formulaires grâce aux htmlspecialchars et htmlentities pour gérer l'**injection HTML** ou plus communément appelée ***Cross-Site Scripting*** (**XSS**).
  Rappelons-nous de la faille de sécurité de la méthode de transfert GET par rapport à la méthode POST.

LES LISTES

PHP – LES LISTES

1. Les listes blanches

Une liste blanche correspond à l'ensemble des données que l'internaute **a le droit de rentrer**. Instaurer une liste blanche permet un **contrôle total sur les données transmises**.

Par exemple, si vous proposez lors d'une inscription de choisir le pays d'origine, vous pouvez facilement vérifier si le pays est valide ou non. On implémente souvent ce type de liste avec les tableaux en PHP.

Ce qui nous donne:

```
<?php
   $liste_pays = array('Belgique', 'Canada', 'France', 'Hongrie',
   'Slovénie');
?>
```


PHP – LES LISTES

Pour la vérification:

```
<?php
   $pays_transmis = $_POST['pays'];
   // Vérification de la présence du pays dans la liste
   if(in_array($pays_transmis, $liste_pays))
         echo 'pays valide.';
   else
         echo 'pays invalide, veuillez ressaisir votre pays.';
?>
```

La fonction **in_array*()*** renvoie un booléen:

- **true** si la valeur est présente dans le tableau;
- **false** en cas d'absence.

bool in_array (mixed $needle , array $haystack [, bool $strict])

PHP – LES LISTES

Bien entendu, les tableaux ne sont pas la meilleure solution à adopter dans tous les cas. La base de données peut s'avérer plus efficace pour stocker un plus grand nombre de valeurs autorisées. En effet, créer un tableau composé de plus de 500 valeurs devient assez lourd à gérer.

Limites de la liste blanche

Cette technique de contrôle de données est particulièrement efficace mais peut se révéler très vite complexe à gérer. En effet, comment **contrôler un champ prénom ou nom**? Il existe plusieurs milliers de prénoms et ne parlons pas des noms! La liste blanche s'avère alors très dure à mettre en place.

PHP – LES LISTES

2. Les listes noires

Contrairement à la liste blanche, la liste noire **filtre les données interdites**. On autorise donc tout, sauf certaines valeurs que l'on aura précisées. La liste noire permet généralement de **lutter contre le *spam* et les robots**. En effet, le but du *spam* est de promouvoir un produit ou un service en envoyant le plus de messages possibles et de manière à ce qu'ils soient visibles par le plus grand nombre. Vos forums ont peut-être déjà été la cible du *spam* et une bonne solution est de mettre en place une liste noire.

Exemple d'une liste noire qui filtre les insultes:

```
<?php
   $liste_insulte = array('crapule', 'chenapan', 'goujat', 'manant');
?>
```

PHP – LES LISTES

Contrôle des messages à afficher:

```
<?php
    // Récupération depuis une BDD
    $message = $resultat['message'];
    // Le message est considéré comme valide
    $autorisation = true;
    // On parcourt toutes les insultes de la liste noire
    foreach($liste_insulte as $insulte)
    {
            // Si une insulte est comprise dans le message
            if(stripos($message, $insulte) !== false)
            {
                    $autorisation = false;
                    break;
            }
    }
?>
```


PHP – LES LISTES

On obtient une variable **$autorisation** qui est un booléen (*true* ou *false*).

Prototype de la fonction stripos():

int stripos (string $haystack , string $needle [, int $offset])

Limites de la liste noire

La liste noire est une bonne alternative à la liste blanche. Cependant, il faut **toujours la mettre à jour** pour contrer les nouveaux messages de *spam*.
Par exemple, le mot **cr@pule** n'est pas filtré par notre liste noire ; tout comme le mot **CRAPUL3**. Il faut donc prendre en compte tous ces changements pour avoir une liste noire fiable.

Quelle technique pourrait couvrir un maximum de mots en une seule expression?

PHP – LES LISTES

3. Les listes grises

La combinaison d'une liste blanche et d'une liste noire donne … une liste grise!
La meilleure technique pour filtrer les données est d'utiliser en complément la liste blanche et la liste noire. Il est recommandé donc de préparer, pour les cas où c'est possible, deux listes différentes:

- une **liste blanche** qui autorise seulement des valeurs données;
- une **liste noire** qui filtre les données incorrectes.

En effet, pour filtrer un **champ pseudo** par exemple, comment écrire une liste blanche? Donc dans ce cas, la **liste noire** est préconisée.
En revanche pour gérer un **sondage** (nombre de possibilités finie) il est recommandé d'instaurer une **liste blanche**.

PHP – LES LISTES

Ainsi si vous arrivez à gérer parfaitement les différents cas votre application sera contrôlée par un ensemble de listes blanches, noires et donc des listes grises: **le filtrage des données** est **considérablement augmenté**!

Il faut bien comprendre ces techniques si vous devez maitriser vos flux de données pour protéger correctement votre application web.
Les **failles XSS** et les **injections SQL** peuvent facilement être évitées.

Synthèse de ce que vous devez absolument faire pour contrôler vos données :

- Protéger les chaînes de caractères dans vos requêtes, la meilleure solution à ce jour étant la préparation de vos requêtes;
- Protéger les données affichées par le navigateur avec *htmlentities()* ;
- Mettre en place un filtre systématique par liste grise.

LES FICHIERS .HTACCESS

PHP – LES FICHIERS .HTACCESS

1. Intérêt des fichiers htaccess

Les fichiers ***.htaccess*** peuvent être utilisés dans n'importe quel répertoire virtuel ou sous-répertoire.

Les principales raisons d'utilisation des fichiers .htaccess sont :

- Gérer l'accès à certains fichiers;
- Ajouter un mime-type;
- Protéger l'accès à un répertoire par un mot de passe;
- Protéger l'accès à un fichier par un mot de passe;
- Définir des pages d'erreurs personnalisées.
- Le rewriting

PHP – LES FICHIERS .HTACCESS

2. Principe des fichiers htaccess

Le fichier *.htaccess* est placé dans le répertoire dans lequel il doit agir. Il agit ainsi sur les permissions du répertoire qui le contient et de tous ses sous-répertoires.

Vous pouvez placer un autre fichier *.htaccess* dans un sous-répertoire d'un répertoire déjà contrôlé par un fichier *.htaccess*. Le fichier .htaccess du répertoire parent reste en « activité » tant que les fonctionnalités n'ont pas été réécrites.

PHP – LES FICHIERS .HTACCESS

Sous Windows, il est logiquement impossible de créer un fichier .htaccess, puisque Windows ne vous autorisera pas à sauvegarder le fichier tel quel.

Voici la démarche à suivre:

- Créer un fichier texte « fichier.htaccess »
- Renommer le fichier en supprimant « fichier »

Remarque:
Selon votre éditeur, vous pouvez également sauvegarder le fichier directement en .htaccess. Sous Notepad, il suffit de mettre des guillemets autour du nom de fichier tandis que UltraEdit gère le nom lui-même.

PHP – LES FICHIERS .HTACCESS

3. Empêcher l'accès à des ressources

Un fichier *.htaccess* est composé de deux sections:

- Une première section contient les **chemins vers les fichiers contenant les définitions de groupes et d'utilisateurs**:

```
AuthUserFile /repertoire/de/votre/fichier/.FichierDeMotDePasse
AuthGroupFile /repertoire/de/votre/fichier/.FichierDeGroupe
AuthName "Accès protégé"
AuthType Basic
```

- Une seconde section contient la **définition des conditions d'accès**:

```
Require valid-user {instruction d'accès à satisfaire }
```


PHP – LES FICHIERS .HTACCESS

- ***AuthUserFile*** définit le chemin d'accès absolu vers le fichier de mot de passe.
- ***AuthGroupFile*** définit le chemin d'accès absolu vers le fichier de groupe.
- **AuthName** entraîne l'affichage dans le navigateur Internet de: *« Tapez votre nom d'utilisateur et votre mot de passe. Domaine: "Accès protégé" »*
- **AuthType** Basic précise qu'il faut utiliser AuthUserFile pour l'authentification
- ***Require valid-user*** précise que l'on autorise uniquement les personnes identifiées. Il est également possible de préciser explicitement le nom des personnes autorisées à s'identifier: *require user {username}*

PHP – LES FICHIERS .HTACCESS

4. Protéger un répertoire par un mot de passe

Il s'agit d'une des applications les plus utiles du fichier ***.htaccess*** car elle permet de **définir** de façon sûre (à l'aide d'un login et d'un mot de passe) **les droits d'accès à des fichiers par certains utilisateurs**.

La syntaxe est la suivante:

```
AuthUserFile {emplacement du fichier de mot de passe}
AuthGroupFile {emplacement du fichier de groupe}
AuthName "Accès protégé"
AuthType Basic
Require valid-user
```


PHP – LES FICHIERS .HTACCESS

- La commande *AuthUserFile* permet de **définir l'emplacement du fichier contenant les logins et les mots de passe** des utilisateurs autorisés à accéder à une ressource donnée.
- La commande *AuthGroupFile* permet de **définir l'emplacement du fichier contenant les groupes d'utilisateurs autorisés à s'identifier**. Il est possible d'outrepasser cette déclaration en déclarant le fichier suivant: */dev/null*.

PHP – LES FICHIERS .HTACCESS

– Voici un exemple de fichier ***.htaccess***:

```
ErrorDocument 403 http://www.commentcamarche.net/accesrefuse.php3
AuthUserFile /repertoire/de/votre/fichier/.FichierDeMotDePasse
AuthGroupFile /dev/null
AuthName "Accès sécurisé au site CCM"
AuthType Basic
Require valid-user
```


PHP – LES FICHIERS .HTACCESS

– Le fichier de mot de passe est un fichier texte devant contenir sur chacune des ses lignes le nom de chaque utilisateur suivi des deux points (:), puis du mot de passe crypté (solution recommandée) ou en clair.

Voici un exemple de fichier de mot de passe non crypté (ici ***.FichierDeMotDePasse***)

```
JFPillou:Toto504
Damien:Robert(32)
Comma:Joe[leTaxi]
```

Voici le même fichier contenant des mots de passe cryptés:

```
JFPillou:$apr1$Si0.....$teyL5Y7BR4cHj0sX309Jj0
Damien:$apr1$TD1.....$sfPTHjoufoNsda4HsD1oL0
Comma:$apr1$zF1.....$wYKqRu2aVYlAEQnxVkly8.
```

PHP – LES FICHIERS .HTACCESS

.FichierDeMotDePasse est un simple fichier texte contenant les noms des utilisateurs suivis d'un : puis du mot de passe crypté (ou non) de cet utilisateur.

Ce fichier de mot de passe ne devrait pas être mis dans un répertoire virtuel Internet (mais comment faire autrement si l'on ne possède pas de serveur Internet et que notre site est hébergé par un tiers ?). Il faut de plus prendre la précaution de **mélanger des majuscules, des minuscules, des chiffres et des symboles dans le nom du fichier**... .

PHP – LES FICHIERS .HTACCESS

5. Crypter les mots de passe

Apache fournit un **outil** permettant de **générer facilement des mots de passe cryptés** (aussi bien sous Windows que sous Unix), il s'agit de l'utilitaire ***htpasswd*** accessible dans le sous-répertoire *bin* d'Apache.

La syntaxe de cet utilitaire est la suivante:

- Pour créer un nouveau fichier de mots de passe:

htpasswd -c {chemin du fichier de mot de passe} utilisateur

- Pour ajouter un nouvel utilisateur/mot de passe à un fichier existant:

htpasswd {chemin du fichier de mot de passe} utilisateur

PHP – LES FICHIERS .HTACCESS

Le mot de passe sera demandé en ligne de commande avec une confirmation.

Voici un exemple:
htpasswd -c /www/secure/.htpasswd JFPillou

Voici un petit utilitaire vous permettant de crypter vos mots de passe en ligne:

```
<form action="crypt.php3" method="post">
Utilisateur: <input type="text" name="login" size="8">
Mot de passe: <input type="password" name="passwd" size="8">
<input type="submit" value="crypter">
</form>
```


PHP – LES FICHIERS .HTACCESS

6. Empêcher l'accès à un répertoire à un domaine
   La syntaxe pour bloquer l'accès d'un répertoire par un domaine est la suivante:

   Allow (all, [liste de domaines])
   Deny (all, [liste de domaines])
   Order (Allow,Deny ou Deny,Allow)

   Order Deny, Allow
   Deny from LeNomDuDomaine.com

   Toutes les personnes (requêtes) provenant du domaine .LeNomDuDomaine.com ne pourront avoir accès aux ressources comprises dans le répertoire et ses sous-répertoires. La commande Order sert à préciser explicitement que la commande Deny va bien annuler l'effet de Allow et non l'inverse.

PHP – LES FICHIERS .HTACCESS

Voici un exemple de restriction d'accès :

```
ErrorDocument 403 http://www.commentcamarche.net/accesrefuse.php3
AuthUserFile /repertoire/de/votre/fichier/.FichierDeMotDePasse
AuthGroupFile /dev/null
AuthName "Accès sécurisé au site CCM"
AuthType Basic
Order deny,allow
Deny from all
Allow from 193.48.172.2
Require user JFPillou
```

Dans ce cas, l'accès ne sera possible que pour l'utilisateur *JFPillou* à partir de l'adresse IP *193.48.172.2* et avec le bon mot de passe.

.

PHP – LES FICHIERS .HTACCESS

7. Empêcher l'accès à un fichier particulier

Par défaut, Apache applique les restrictions du fichier *.htaccess* à l'ensemble des fichiers du répertoire dans lequel il se trouve ainsi qu'à tous les fichiers contenus dans ses sous-répertoires.

Il est également possible de restreindre l'accès pour un ou plusieurs fichiers du répertoire grâce à la **balise *<Files>*.**

.

PHP – LES FICHIERS .HTACCESS

Voici un exemple de restriction aux fichiers *admin.php3* et *admin2.php3*:

```
<Files admin.php3>
AuthUserFile /repertoire/de/votre/fichier/.FichierDeMotDePasse
AuthGroupFile /dev/null
AuthName "Accès sécurisé au site CCM"
AuthType Basic
Require user JFPillou
</Files>
<Files admin2.php3>
AuthUserFile /repertoire/de/votre/fichier/.FichierDeMotDePasse
AuthGroupFile /dev/null
AuthName "Accès sécurisé au site CCM"
AuthType Basic
Require user JFPillou
</Files>.
```


PHP – LES FICHIERS .HTACCESS

Il faut utiliser **une seule balise <Files> par fichier**.
Sinon, l'erreur suivante est reportée dans le fichier de log des erreurs:

.htaccess: Multiple <Files> arguments not (yet) supported.

Pour info, depuis Apache 1.3, il est conseillé d'ajouter et utiliser la balise <FilesMatch> à la place de la balise <Files>.
Cette nouvelle balise ne supporte aussi qu'un seul argument mais on peut traiter plusieurs fichiers grâce à une **expression régulière**.

PHP – LES FICHIERS .HTACCESS

8. Empêcher l'accès à un type de fichier par un domaine

```
<Files *.png>
Order Deny, Allow
Deny from .LeNomDuDomaine.com
</Files>
```

Toutes les personnes (requêtes) provenant du domaine *.LeNomDuDomaine.com* ne pourront avoir accès aux images, dont l'extension est .png, comprises dans le répertoire et ses sous-répertoires.

PHP – LES FICHIERS .HTACCESS

9. Autoriser l'accès à un groupe de fichiers par un domaine et un pays

```
<Files php*>
Order Allow, Deny
Deny from all Allow from .phpfrance.com
Allow from .fr
</Files>
```

Toutes les personnes (requêtes) provenant du domaine *.phpfrance.com* ou des domaines ayant la terminaison *.fr* pourront avoir accès aux fichiers commençant par php (par exemple, les fichiers *phpbonjour.html*, *phpaurevoir.vxf*) compris dans le répertoire et ses sous-répertoires.

PHP – LES FICHIERS .HTACCESS

10. Protéger un répertoire par un login

Cette méthode (beaucoup moins sûre que la précédente) permet une authentification de bas niveau uniquement par le nom de l'utilisateur.

La syntaxe est la suivante:

Require (user [liste des utilisateurs], group [liste des groupes], valid-user)

Voici un exemple de ligne du fichier *.htaccess*:

Require User Damien Comma PumpPHP Jeff Rastapaye

Tout utilisateur souhaitant rentrer dans le répertoire ou un de ses sous-répertoires sera refusé sauf s'il s'identifie en donnant un nom figurant dans la liste.

PHP – LES FICHIERS .HTACCESS

11. Protéger un répertoire par un login

Obliger un utilisateur à satisfaire à au moins une des conditions

Voici la syntaxe:

Satisfy (any, all)
Order Allow, Deny
Deny from all
Allow from .free.fr
Require User Damien Comma PumpPHP Jeff Rastapaye
Satisfy Any

Ce qui signifie que l'accès au répertoire sera bloqué pour tout le monde à l'exception des personnes qui s'identifient et des requêtes provenant du domaine *.free.fr*.

PHP – LES FICHIERS .HTACCESS

12. Gérer les types de fichiers

    Un **type MIME** (en anglais *MIME type*) est un ensemble de types de fichiers standard, permettant d'associer une extension de fichier donnée à une application, afin d'automatiser le lancement de l'application.

13. Ajouter un Mime-Type à un répertoire

    La syntaxe est la suivante:
    AddType (mime/type [liste d'extension])

    Voici un exemple de mise en œuvre du fichier ***.htaccess***:
    AddType image/x-photoshop PSD
    AddType application/x-httpd-php .php3
    AddType application/x-httpd-php .htm

PHP – LES FICHIERS .HTACCESS

Le serveur enverra au navigateur Internet le fichier en lui disant de lancer le programme PhotoShop (s'il est installé sur votre machine) et de lui donner le fichier.

Habituellement, ceci est utilisé pour des fichiers nécessitant un Plug-In particulier non reconnu par votre navigateur.
Cette commande permet aussi d'annuler tout élément prédéfini. Ainsi, vous pouvez enregistrer un fichier *.wav* avec une extension .gif et préciser au navigateur de considérer les fichiers .gif comme des fichiers audio !

En pratique, on pourra donc utiliser cette commande pour ordonner à PHP de parser d'autres extensions de fichier, *.htm* par exemple

PHP – LES FICHIERS .HTACCESS

14. Forcer tous les fichiers d'un répertoire à un Mime-Type

Voici la syntaxe à adopter:
ForceType (mime/type)

Par exemple avec la ligne suivante, tous les fichiers du répertoire contenant le fichier ***.htaccess*** seront considérés comme étant des fichiers *.jpg* quelle que soit leur extension:

ForceType image/jpg

Ce type de commande ne peut être utilisé dans les bornes!

PHP – LES FICHIERS .HTACCESS

15. Définir les extensions de fichiers par défaut

La syntaxe à suivre est:
DefaultType (mime/type)

Par exemple
DefaultType text/html

Cette option vous permet de définir le comportement par défaut du navigateur face à des extensions lui étant inconnues.
Ici, il prendra tout fichier inconnu (par exemple '*bonjour*', '*Rastapaye.phpfrance*') en tant que document HTML

PHP – LES FICHIERS .HTACCESS

16. Personnalisation des messages d'erreurs

Il s'agit d'une fonctionnalité pratique car elle permet de définir une page par défaut pour un type d'erreur donné.

Cela permet d'une part de guider l'utilisateur au lieu d'afficher la banale page d'erreur du navigateur, ainsi que d'égayer la navigation même en cas d'erreur.

ErrorDocument (code-à-3-chiffres [nom du fichier ou texte ou url])

PHP – LES FICHIERS .HTACCESS

Les deux lignes suivantes permettent de définir des pages d'erreurs personnalisées au cas où l'accès à un document serait interdit ou bien que le document n'existe pas:

ErrorDocument 403 /erreurs/403.php3

ErrorDocument 404 /erreurs/404.php3

Ceci vous permet de donner un message d'erreur personnalisé remplaçant les fichiers fournis avec le navigateur.
Voici quelques-unes des erreurs les plus courantes à personnaliser:

- 401 Unauthorized: la personne n'a pas passé avec succès l'identification.
- 403 Forbidden: le serveur n'a pas le droit de répondre à votre requête.
- 404 Not Found: le serveur n'a pas trouvé le document souhaité.

PHP – LES FICHIERS .HTACCESS

17. Changer le fichier index par défaut

Le fichier *index* est le fichier qui est affiché lorsque aucun nom de fichier n'est défini dans l'URL (par exemple *http://www.monserveur.com/repertoire*). Cela permet d'éviter que le navigateur liste l'ensemble des fichiers contenus dans le répertoire (pour des raisons de confidentialité).

La syntaxe pour effectuer ce type d'opération est la suivante:
DirectoryIndex (fichiers)

Voici un exemple de mise en application:
DirectoryIndex index.php index.html index.phtml /erreurs/403.php

PHP – LES FICHIERS .HTACCESS

Lorsque vous essayez d'accéder au répertoire sans préciser la page à afficher, Apache va avoir recours à la directive *DirectoryIndex*. En général, par défaut, cette directive pointe vers *index.html* puis *index.htm*.

Dans l'exemple précédent, Apache va commencer par chercher *index.php*, puis *index.html*, et ensuite *index.phtml*. Si aucun de ces trois fichiers existent, la page *403.php* (se trouvant dans la racine) sera affichée pour éviter de lister le répertoire.

PHP – LES FICHIERS .HTACCESS

18. L'URL Rewriting

Le principe de la réécriture d'URL est donc de mettre en place un « système » sur le serveur pour qu'il sache **interpréter un nouveau format d'URL**.
Cette réécriture est basée sur les **expressions rationnelles**.

La procédure de mise en œuvre est la suivante:

- Vérifier que votre hébergeur permet l'utilisation de l'URL Rewriting. Si vous êtes chez un hébergeur gratuit qui ne le gère pas, c'est une très bonne raison pour franchir le cap et bénéficier de tous les avantages d'un hébergement professionnel (ça ne coûte pas grand chose...)
- Identifier les pages dynamiques dont l'URL comporte des paramètres, et choisir un nouveau schéma d'URL "propre«
- Ecrire les règles de réécriture dans le fichier .htaccessChanger tous les liens vers chaque fichier dont l'URL a changé
- Mettre à jour votre site et vérifier que tout fonctionne

PHP – LES FICHIERS .HTACCESS

Voici la liste des éléments pris en considération dans les règles de réécriture:

Elément	Explications
^	Indique le début de l'URL à réécrire. Ce caractère est facultatif mais il est plus rigoureux de l'utiliser.
article-	Cette partie de l'URL n'est pas utilisée directement. Nous aurions pu écrire art à la place de article. Ce préfixe peut servir à différencier différents schémas d'URL, et il permet à l'internaute de mieux comprendre l'objet de la page.
()	Les parenthèses servent à encadrer une variable dont la valeur est récupérée dans la 3ème partie de la ligne.
[0-9]+	Indique que la variable est composée d'un ou plusieurs chiffres.
$	Indique la fin de l'URL à réécrire. Ce caractère est facultatif mais il est plus rigoureux de l'utiliser.

PHP – LES FICHIERS .HTACCESS

Elément	Explications
/articles/	Cette partie est parfois facultative (cela dépend de la configuration du serveur). En général il suffit d'indiquer l'emplacement du fichier de manière relative au répertoire dans lequel est situé le fichier .htaccess (donc on peut se passer de cet élément). Sur certains hébergeurs mutualisés (OVH ou Sivit pour n'en citer que deux), vous devez indiquer le chemin complet vers le fichier, à partir de la racine du site, comme dans notre exemple.
article.php	Nom du fichier que le serveur doit utiliser pour afficher la page. C'est le nom d'un fichier qui existe physiquement et qui contient un script (PHP dans notre exemple) de gestion de la page dynamique.
?	Caractère obligatoire précédant la série de variables passées dans l'URL réécrite.
id=$1	Indique que la variable nommée id prendra la valeur située dans la première paire de parenthèses.
&	Caractère utilisé pour séparer 2 variables dans l'URL réécrite.

PHP – LES FICHIERS .HTACCESS

Elément	Explications
rubrique=$2	Indique que la variable nommée rubrique prendra la valeur située dans la deuxième paire de parenthèses.
[L]	Drapeau (option) signifiant « Last », indiquant au module de réécriture qu'il doit s'arrêter. Plus précisément, si l'URL de la page demandée par le visiteur correspond au schéma défini par cette règle, alors le module de réécriture d'URL ne doit pas examiner les autres règles situées dans le reste du fichier .htaccess. Il n'est pas toujours obligatoire mais il ne fera pas de mal!
	Vous devez **activer le module apache rewrite_module** pour pouvoir utiliser la réécriture.

PHP – LES FICHIERS .HTACCESS

Exemple d'utilisation:

```
Options +FollowSymlinks
RewriteEngine on
RewriteRule ^cConnexion$ cConnexion.php
RewriteRule ^inscription$ inscription.php
RewriteRule ^cInscription$ cInscription.php
RewriteRule ^menu$ menu.php
RewriteRule ^cValidation$ cValidation.php [L]
```

- **Options +FollowSymlinks** va indiquer au serveur qu'il est autorisé à suivre les liens symboliques dans ce répertoire.
- **RewriteEngine on** va donner l'ordre d'activer la réécriture d'URL.
- **RewriteRule « RegEx » URL ([L])** est l'instruction de réécriture.

PHP- MVC

1. Partons d'un exemple
2. Premières améliorations
3. Le modèle MVC
4. Améliorations supplémentaires
5. Ajout de fonctionnalités supplémentaires
6. Ajout d'un contrôleur frontal

PHP- MVC

1. **Partons d'un exemple**
   - **1.1 l'énoncé**
   - **1.2 les données**
   - **1.3 la page index.php**
   - **1.4 l'affichage**
   - **1.5 les défauts de la solution**
2. Premières améliorations
3. Le modèle MVC
4. Améliorations supplémentaires
5. Ajout de fonctionnalités supplémentaires
6. Ajout d'un contrôleur frontal

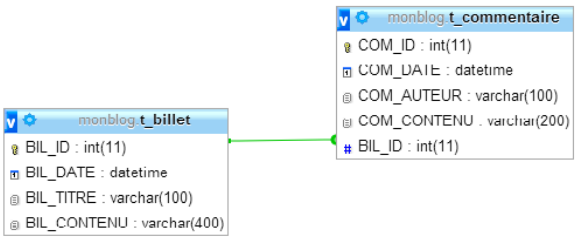
PHP- MVC

1 . Partons d'un exemple

1.1 l'énoncé

Nous développons une page Web PHP de type « blog » puisant ses informations dans une base de données relationnelle.

1.2 les données



PHP- MVC

1.3 la page index.php

Index.php

```
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="UTF-8" />
        <link rel="stylesheet" href="style.css" />
        <title>Mon Blog</title>
    </head>
    <body>
        <div id="global">
            <header>
                <a href="index.php"><h1 id="titreBlog">Mon Blog</h1></a>
                <p>Je vous souhaite la bienvenue sur ce modeste blog.</p>
            </header>
            <div id="contenu">
                <?php
                $bdd = new PDO('mysql:host=localhost;dbname=monblog;charset=utf8',
                        'root', '');
                $billets = $bdd->query('select BIL_ID as id, BIL_DATE as date,'
                        . ' BIL_TITRE as titre, BIL_CONTENU as contenu from T_BILLET'
                        . ' order by BIL_ID desc');
                foreach ($billets as $billet):
                    ?>
```


PHP- MVC

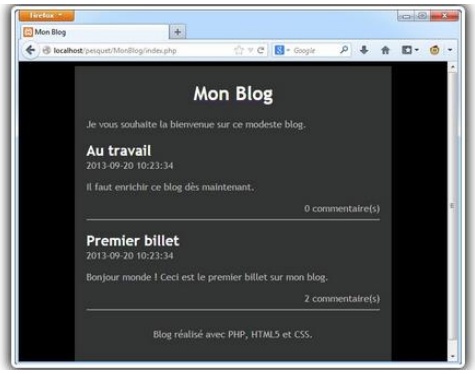
1.3 la page index.php suite

Index.php

```
                    <article>
                        <header>
                            <h1 class="titreBillet"><?= $billet['titre'] ?></h1>
                            <time><?= $billet['date'] ?></time>
                        </header>
                        <p><?= $billet['contenu'] ?></p>
                    </article>
                    <hr />
                <?php endforeach; ?>
            </div> <!-- #contenu -->
            <footer id="piedBlog">
                Blog réalisé avec PHP, HTML5 et CSS.
            </footer>
        </div> <!-- #global -->
    </body>
</html>
```

PHP- MVC

1.4 l'affichage



PHP- MVC

1.5 défauts de la solution

Les principaux défauts de cette page Web sont les suivants :

- elle mélange balises HTML et code PHP ;
- sa structure est monobloc, ce qui rend sa réutilisation difficile.

De manière générale, tout logiciel doit gérer plusieurs problématiques :

- interactions avec l'extérieur, en particulier l'utilisateur : saisie et contrôle de données, affichage. C'est la problématique de **présentation** ;
- opérations sur les données (calculs) en rapport avec les règles métier (« business logic »). C'est la problématique des **traitements** ;
- accès et stockage des informations qu'il manipule, notamment entre deux utilisations. C'est la problématique des **données**.

La page Web actuelle mélange code de présentation (les balises HTML) et accès aux données (requêtes SQL)

PHP- MVC

1. Partons d'un exemple

2. **Premières améliorations**

   2.1 isolation de l'affichage et des données
   2.2 isolation de l'affichage dans un fichier séparé
   2.3 isolation de l'accès aux données dans un fichier séparé
   2.4 conclusion

3. Le modèle MVC

4. Améliorations supplémentaires

5. Ajout de fonctionnalités supplémentaires

6. Ajout d'un contrôleur frontal

PHP- MVC

2 . Améliorations
2.1 isolation de l'affichage et des données

Une première amélioration consiste à séparer
- le code d'accès aux données
- du code de présentation

au sein du fichier **index.php**. (cfr slide suivant)

Le code est devenu plus lisible, mais les problématiques de présentation et d'accès aux données sont toujours gérées au sein d'un même fichier PHP.

PHP- MVC

Index.php

```php
<?php
// Accès aux données
$bdd = new PDO('mysql:host=localhost;dbname=monblog;charset=utf8', 'root', '');
$billets = $bdd->query('select BIL_ID as id, BIL_DATE as date,'
  . ' BIL_TITRE as titre, BIL_CONTENU as contenu from T_BILLET'
  . ' order by BIL_ID desc');
?>

<!-- Affichage -->
<!doctype html>
<html lang="fr">
  <head>
  ...
      <div id="contenu">
        <?php foreach ($billets as $billet): ?>
          <article>
            <header>
              <h1 class="titreBillet"><?= $billet['titre'] ?></h1>
              <time><?= $billet['date'] ?></time>
            </header>
            <p><?= $billet['contenu'] ?></p>
          </article>
          <hr />
        <?php endforeach; ?>
      </div> <!-- #contenu -->
      ...
```


PHP- MVC

2.2 isolation de l'affichage dans un fichier séparé

On peut regrouper le code d'affichage précédent dans un fichier dédié nommé **vueAccueil.php**.

vueAccueil.php

```php
<!doctype html>
<html lang="fr">
<head>
  ...
</head>
<body>
  ...
    <div id="contenu">
      <?php foreach ($billets as $billet): ?>
        <article>
          ...
        </article>
        <hr />
      <?php endforeach; ?>
    </div> <!-- #contenu -->
  ...
</body>
</html>
```

PHP- MVC

2.3 isolation de l'accès aux données dans un fichier séparé

nous pouvons gagner en modularité en isolant le code d'accès aux données dans un fichier PHP dédié. Appelons ce fichier **Modele.php**.

Modele.php

```php
<?php

// Renvoie la liste de tous les billets, triés par identifiant décroissant
function getBillets() {
  $bdd = new PDO('mysql:host=localhost;dbname=monblog;charset=utf8', 'root', '');
  $billets = $bdd->query('select BIL_ID as id, BIL_DATE as date,'
    . ' BIL_TITRE as titre, BIL_CONTENU as contenu from T_BILLET'
    . ' order by BIL_ID desc');
  return $billets;
}
```

Dans ce fichier, nous avons déplacé la récupération des billets du blog à l'intérieur d'une fonction nommée **getBillets**.

PHP- MVC

2.4 modification du fichier index.php

Le lien entre accès aux données et présentation est effectué par le fichier principal **index.php**. Ce fichier est maintenant très simple.

Index.php

```php
<?php

require 'Modele.php';

$billets = getBillets();

require 'vueAccueil.php';
```

PHP- MVC

2.4 conclusion

Outre la feuille de style CSS, notre page Web est maintenant constituée de trois fichiers :

- **Modele.php** (PHP uniquement) pour l'accès aux données ;
- **vueAccueil.php** (PHP et HTML) pour l'affichage des billets du blog ;
- **index.php** (PHP uniquement) pour faire le lien entre les deux pages précédentes.

Cette nouvelle structure est plus complexe, mais les responsabilités de chaque partie sont maintenant claires. En faisant ce travail de « refactoring », nous avons rendu notre exemple conforme à un modèle d'architecture très employé sur le Web : le modèle **MVC**.

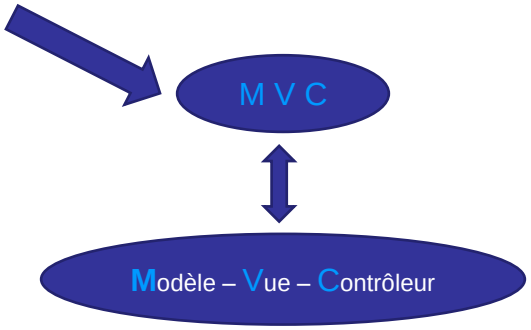
PHP- MVC

1. Partons d'un exemple
2. Premières améliorations
3. **Le modèle MVC**
4. Améliorations supplémentaires
5. Ajout de fonctionnalités supplémentaires
6. Ajout d'un contrôleur frontal

PHP- MVC

3 . Le modèle MVC

En fait, il y a des problèmes en programmation qui reviennent tellement souvent qu'on a créé toute une série de bonnes pratiques que l'on a réunies sous le nom de ***design patterns*.**



PHP- MVC

3 . Le modèle MVC

La partie **Modèle** d'une architecture MVC encapsule la logique métier (« business logic ») ainsi que l'accès aux données. Il peut s'agir d'un ensemble de fonctions (Modèle procédural) ou de classes (Modèle orienté objet).

La partie **Vue** s'occupe des interactions avec l'utilisateur : présentation, saisie et validation des données.

La partie **Contrôleur** gère la dynamique de l'application. Elle fait le lien entre l'utilisateur et le reste de l'application.

.

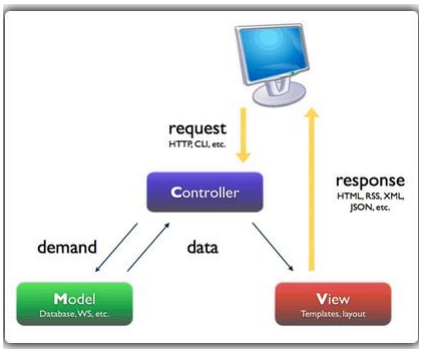
PHP- MVC

3 . Le modèle MVC

La demande de l'utilisateur (exemple : requête HTTP) est reçue et interprétée par le **Contrôleur**.

Celui-ci utilise les services du **Modèle** afin de préparer les données à afficher.

Ensuite, le **Contrôleur** fournit ces données à la **Vue**, qui les présente à l'utilisateur (par exemple sous la forme d'une page HTML).



PHP- MVC

1 . Partons d'un exemple

2. Premières améliorations

3. Le modèle MVC

4. Améliorations supplémentaires
4.1 utilisation d'un « template »
4.2 factorisation de la connexion à la BD
4.3 gestion des erreurs
4.4 nouvelle structure de fichiers

5. Ajout de fonctionnalités supplémentaires

6. Ajout d'un contrôleur frontal

PHP- MVC

4 . Améliorations supplémentaires

4.1 utilisation d'un « template »

nous allons mettre en œuvre l'utilisation d'un modèle de page (gabarit), appelé **template** en anglais.
Ce modèle contiendra tous les éléments communs et permettra d'ajouter les éléments spécifiques à chaque vue.

gabarit.php

```
<!doctype html>
<html lang="fr">
  <head>
    <meta charset="UTF-8" />
    <link rel="stylesheet" href="style.css" />
    <title><?= $titre ?></title>   <!-- Élément spécifique -->
  </head>
  <body>
    <div id="global">
      <header>
        <a href="index.php"><h1 id="titreBlog">Mon Blog</h1></a>
        <p>Je vous souhaite la bienvenue sur ce modeste blog.</p>
      </header>
      <div id="contenu">
        <?= $contenu ?>   <!-- Élément spécifique -->
      </div>
      <footer id="piedBlog">
        Blog réalisé avec PHP, HTML5 et CSS.
      </footer>
    </div> <!-- #global -->
  </body>
</html>
```


PHP- MVC

4.1 utilisation d'un « template »

Au moment de l'affichage d'une vue HTML, il suffit de définir les valeurs des éléments spécifiques, puis de déclencher le rendu de notre gabarit. Pour cela, on utilise des fonctions PHP qui manipulent le flux de sortie de la page.

valeur de l'élément spécifique **$titre**

fonction PHP ob_start. Son rôle est de déclencher la mise en tampon du flux HTML de sortie : au lieu d'être envoyé au navigateur, ce flux est stocké en mémoire ;

la fonction PHP ob_get_clean permet de récupérer dans une variable le flux de sortie mis en tampon depuis l'appel à **ob_start**. La variable se nomme ici **$contenu**, ce qui permet de définir l'élément spécifique associé ;

Enfin, on déclenche le rendu du gabarit. Lors du rendu, les valeurs des éléments spécifiques **$titre** et **$contenu** seront insérés dans le résultat HTML envoyé au navigateur.

vueAccueil.php

```
<?php $titre = 'Mon Blog'; ?>

<?php ob_start(); ?>
<?php foreach ($billets as $billet): ?>
  <article>
    <header>
      <h1 class="titreBillet"><?= $billet['titre'] ?></h1>
      <time><?= $billet['date'] ?></time>
    </header>
    <p><?= $billet['contenu'] ?></p>
  </article>
  <hr />
<?php endforeach; ?>
<?php $contenu = ob_get_clean(); ?>

<?php require 'gabarit.php'; ?>
```

PHP- MVC

4.2 factorisation de la connexion à la base

On peut améliorer l'architecture de la partie Modèle en isolant le code qui établit la connexion à la base de données sous la forme d'une fonction **getBdd** ajoutée dans le fichier **Modele.php**. Cela évitera de dupliquer le code de connexion lorsque nous ajouterons d'autres fonctions au Modèle.

modele.php

```php
<?php

// Renvoie la liste de tous les billets, triés par identifiant décroissant
function getBillets() {
  $bdd = getBdd();
  $billets = $bdd->query('select BIL_ID as id, BIL_DATE as date,'
    . ' BIL_TITRE as titre, BIL_CONTENU as contenu from T_BILLET'
    . ' order by BIL_ID desc');
  return $billets;
}

// Effectue la connexion à la BDD
// Instancie et renvoie l'objet PDO associé
function getBdd() {
  $bdd = new PDO('mysql:host=localhost;dbname=monblog;charset=utf8', 'root', '');
  return $bdd;
}
```


PHP- MVC

4.3 gestion des erreurs

Le meilleur choix est d'implémenter la gestion des erreurs au niveau du **Contrôleur**. Gérer la dynamique de l'application, y compris dans les cas dégradés, fait partie de ses responsabilités. Nous allons tout d'abord modifier la connexion à la base de données afin que les éventuelles erreurs soient signalées sous la forme d'exceptions.

modele.php

```php
...
$bdd = new PDO('mysql:host=localhost;dbname=monblog;charset=utf8',
  'root', '', array(PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION));
...
```

Si une exception est levée lors de son exécution, une page HTML minimale contenant le message d'erreur est affichée.

On peut souhaiter conserver l'affichage du gabarit des vues même en cas d'erreur. Il suffit de définir une vue **vueErreur.php** dédiée à leur affichage.

vueErreur.php

```php
<?php $titre = 'Mon Blog'; ?>

<?php ob_start() ?>
<p>Une erreur est survenue : <?= $msgErreur ?></p>
<?php $contenu = ob_get_clean(); ?>

<?php require 'gabarit.php'; ?>
```

PHP- MVC

4.3 gestion des erreurs

On modifie ensuite le contrôleur pour déclencher le rendu de cette vue en cas d'erreur.

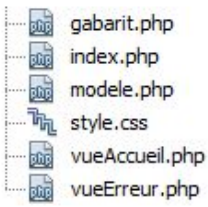
index.php

```php
<?php

require 'Modele.php';

try {
  $billets = getBillets();
  require 'vueAccueil.php';
}
catch (Exception $e) {
  $msgErreur = $e->getMessage();
  require 'vueErreur.php';
}
```


PHP- MVC

4.4 Nouvelle structure de fichiers

Voici la nouvelle structure de notre blog qui respecte un modèle MVC simple.



PHP- MVC

1 . Partons d'un exemple

2. Premières améliorations

3. Le modèle MVC

4. Améliorations supplémentaires

5. Ajout de fonctionnalités supplémentaires
5.1 nouveau besoin pris en compte
5.2 modification de modele.php
5.3 création d'une vue vueBillet.php
5.4 création d'un contrôleur billet.php
5.5 modifier vueAccueil.php
5.6 affichage obtenu
5.7 structure

6. Ajout d'un contrôleur frontal

PHP- MVC

5 . Ajout de fonctionnalités supplémentaires

5.1 nouveau besoin pris en compte

Le clic sur le titre d'un billet doit permettre d'afficher les commentaires associés à ce billet.

Nous devons :

- ajouter dans **modele.php** les instructions d'accès aux données
- créer ensuite une nouvelle vue **vueBillet.php** dont le rôle est d'afficher les informations demandées.
- créer un nouveau fichier contrôleur, **billet.php**, qui fait le lien entre modèle et vue pour répondre au nouveau besoin.

PHP- MVC

5.2 modification de modele.php

modele.php

```
...
// Renvoie les informations sur un billet
function getBillet($idBillet) {
  $bdd = getBdd();
  $billet = $bdd->prepare('select BIL_ID as id, BIL_DATE as date,'
    . ' BIL_TITRE as titre, BIL_CONTENU as contenu from T_BILLET'
    . ' where BIL_ID=?');
  $billet->execute(array($idBillet));
  if ($billet->rowCount() == 1)
    return $billet->fetch();  // Accès à la première ligne de résultat
  else
   throw new Exception("Aucun billet ne correspond à l'identifiant '$idBillet'");
}

// Renvoie la liste des commentaires associés à un billet
function getCommentaires($idBillet) {
  $bdd = getBdd();
  $commentaires = $bdd->prepare('select COM_ID as id, COM_DATE as date,'
    . ' COM_AUTEUR as auteur, COM_CONTENU as contenu from T_COMMENTAIRE'
    . ' where BIL_ID=?');
  $commentaires->execute(array($idBillet));
  return $commentaires;
}
```


PHP- MVC

5.3 création d'une vue vueBillet.php

Bien entendu, cette vue définit les éléments dynamiques **$titre** et **$contenu**, puis inclut le gabarit commun.

vueBillet.php

```
<?php $titre = "Mon Blog - " . $billet['titre']; ?>

<?php ob_start(); ?>
<article>
  <header>
    <h1 class="titreBillet"><?= $billet['titre'] ?></h1>
    <time><?= $billet['date'] ?></time>
  </header>
  <p><?= $billet['contenu'] ?></p>
</article>
<hr />
<header>
  <h1 id="titreReponses">Réponses à <?= $billet['titre'] ?></h1>
</header>
<?php foreach ($commentaires as $commentaire): ?>
  <p><?= $commentaire['auteur'] ?> dit :</p>
  <p><?= $commentaire['contenu'] ?></p>
<?php endforeach; ?>
<?php $contenu = ob_get_clean(); ?>

<?php require 'gabarit.php'; ?>
```

PHP- MVC

5.4 création d'un contrôleur billet.php

Le contrôleur a besoin de recevoir en paramètre l'identifiant du billet.
Il s'utilise donc sous la forme **billet.php?id=<id du billet>**.

billet.php

```php
<?php

require 'Modele.php';

try {
  if (isset($_GET['id'])) {
    // intval renvoie la valeur numérique du paramètre ou 0 en cas d'échec
    $id = intval($_GET['id']);
    if ($id != 0) {
      $billet = getBillet($id);
      $commentaires = getCommentaires($id);
      require 'vueBillet.php';
    }
    else
      throw new Exception("Identifiant de billet incorrect");
  }
  else
    throw new Exception("Aucun identifiant de billet");
}
catch (Exception $e) {
  $msgErreur = $e->getMessage();
  require 'vueErreur.php';
}
```


PHP- MVC

5.5 modifier vueAccueil.php

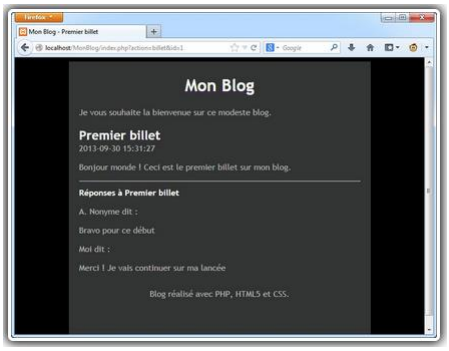
Il faut également modifier la vue **vueAccueil.php** afin d'ajouter un lien vers la page **billet.php** sur le titre du billet

vueAccueil.php

```php
...
    <header>
      <a href="<?= "billet.php?id=" . $billet['id'] ?>">
        <h1 class="titreBillet"><?= $billet['titre'] ?></h1>
      </a>
      <time><?= $billet['date'] ?></time>
    </header>
...
```

PHP- MVC

5.6 affichage obtenu

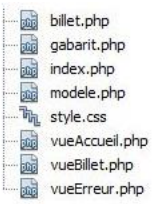


PHP- MVC

5.7 structure

Rappelons les rôles de chaque élément :

- **modele.php** représente la partie Modèle (accès aux données) ;
- **vueAccueil.php**, **vueBillet.php** et **vueErreur.php** constituent la partie Vue (affichage à l'utilisateur). Ces pages utilisent la page **gabarit.php** (template de mise en forme commune) ;
- **index.php** et **billet.php** correspondent à la partie Contrôleur (gestion des requêtes entrantes).



PHP- MVC

1 . Partons d'un exemple

2. Premières améliorations

3. Le modèle MVC

4. Améliorations supplémentaires

5. Ajout de fonctionnalités supplémentaires

6. Ajout d'un contrôleur frontal
6.1 Principe
6.2 Actions rassemblées dans controleur.php
6.3 Modification de index.php
6.4 Modification de vueAccueil.php
6.5 Modification de la structure de répertoires

PHP- MVC

6 . Ajout d'un contrôleur frontal

6.1 principe

L'architecture actuelle, basée sur n contrôleurs indépendants, souffre de certaines limitations :

• elle expose la structure interne du site (noms des fichiers PHP) ;

• elle rend délicate l'application de politiques communes à tous les contrôleurs (authentification, sécurité, etc.).

Pour remédier à ces défauts, on peut ajouter au site un **contrôleur frontal**.

Le contrôleur frontal constitue le point d'entrée unique du site. Son rôle est de centraliser la gestion des requêtes entrantes. Il utilise le service d'un autre contrôleur pour réaliser l'action demandée et renvoyer son résultat sous la forme d'une vue.

Un choix fréquent consiste à transformer le fichier principal **index.php** en contrôleur frontal. Nous allons mettre en œuvre cette solution.

PHP- MVC

6 .1 principe

Ce changement d'architecture implique un changement d'utilisation du site. Voici comment fonctionne actuellement notre blog :

- l'exécution de **index.php** permet d'afficher la liste des billets ;
- l'exécution de **billet.php?id=<id du billet>** affiche les détails du billet identifié dans l'URL.

La mise en œuvre d'un contrôleur frontal implique que **index.php** recevra à la fois les demandes d'affichage de la liste des billets et les demandes d'affichage d'un billet précis. Il faut donc lui fournir de quoi lui permettre d'identifier l'action à réaliser. Une solution courante est d'ajouter à l'URL un paramètre **action**.

Dans notre exemple, voici comment ce paramètre sera interprété :

- si **action** vaut « billet », le contrôleur principal déclenchera l'affichage d'un billet ;
- si **action** n'est pas valorisé, le contrôleur déclenchera l'affichage de la liste des billets (action par défaut).

Toutes les actions réalisables sont rassemblées sous la forme de fonctions dans le fichier **Controleur.php**.

PHP- MVC

6.2 actions rassemblées dans controleur.php

Toutes les actions réalisables sont rassemblées sous la forme de fonctions dans le fichier **Controleur.php**.

controleur.php

```php
<?php

require 'Modele.php';

// Affiche la liste de tous les billets du blog
function accueil() {
  $billets = getBillets();
  require 'vueAccueil.php';
}

// Affiche les détails sur un billet
function billet($idBillet) {
  $billet = getBillet($idBillet);
  $commentaires = getCommentaires($idBillet);
  require 'vueBillet.php';
}

// Affiche une erreur
function erreur($msgErreur) {
  require 'vueErreur.php';
}
```

PHP- MVC

6.3 modification de index.php

L'action à réaliser est déterminée par le fichier **index.php** de notre blog, réécrit sous la forme d'un contrôleur frontal.

Remarque :
l'ancien fichier contrôleur **billet.php** est désormais inutile
et peut être supprimé.

index.php

```php
<?php

require('Controleur.php');

try {
  if (isset($_GET['action'])) {
    if ($_GET['action'] == 'billet') {
      if (isset($_GET['id'])) {
        $idBillet = intval($_GET['id']);
        if ($idBillet != 0)
          billet($idBillet);
        else
          throw new Exception("Identifiant de billet non valide");
      }
      else
        throw new Exception("Identifiant de billet non défini");
    }
    else
      throw new Exception("Action non valide");
  }
  else {
    accueil();  // action par défaut
  }
}
catch (Exception $e) {
    erreur($e->getMessage());
}
```


PHP- MVC

6.4 Modification de vueAccueil.php

Enfin, le lien vers un billet doit être modifié afin de refléter la nouvelle architecture.

vueAccueil.php

```
...
<a href="<?= "index.php?action=billet&id=" . $billet['id'] ?>">
  <h1 class="titreBillet"><?= $billet['titre'] ?></h1>
</a>
...
```

La mise en œuvre d'un contrôleur frontal a permis de préciser les responsabilités et de clarifier la dynamique de la partie **Contrôleur** de notre site :

1. Le contrôleur frontal analyse la requête entrante et vérifie les paramètres fournis ;
2. Il sélectionne et appelle l'action à réaliser en lui passant les paramètres nécessaires ;
3. Si la requête est incohérente, il signale l'erreur à l'utilisateur.

Autre bénéfice : l'organisation interne du site est totalement masquée à l'utilisateur, puisque seul le fichier **index.php** est visible dans les URL.

PHP- MVC

6.5 Modification de la structure de répertoires

Par souci de simplicité, nous avons jusqu'à présent stocké tous nos fichiers source dans le même répertoire.

Nous allons restructurer notre site. La solution la plus évidente consiste à créer des sous-répertoires en suivant le découpage MVC :

- le répertoire **Modele** contiendra le fichier Modele.php ;
- le répertoire **Vue** contiendra les fichiers **vueAccueil.php**, **vueBillet.php** et **vueErreur.php**, ainsi que la page commune **gabarit.php** ;
- le répertoire **Controleur** contiendra le fichier des actions Controleur.php.

On peut également prévoir

- un répertoire **Contenu** pour les contenus statiques (fichier CSS, images, etc.).
- un répertoire **BD** pour le script de création de la base de données.

PHP- MVC

6.5 Modification de la structure de répertoires