UNIVERSITÉ PRIVÉE DE MARRAKECH

PHP

Remerciements

- Le matériel des diapositives est pris de différentes sources comprenant :

PHP et les formulaires(1)

- Formulaire HTML
  - Retourne des informations saisies par un utilisateur vers une application serveur
  - La création d'un formulaire nécessite la connaissance de quelques balises HTML indispensables :
    - Structure : un formulaire commence toujours par la balise <form> et se termine par la balise </form>
    - Champ de saisie de text en ligne :

      <input type = "text" name ="nom_du_champ" value="chaîne">
    - Boutons d'envoi et d'effacement :

      <input type=" submit " value = "Envoyer">

      <input type = "reset" name ="efface" value = "Effacer">
    - Case à cocher et bouton radio :

      <input type = "checkbox" name ="case1" value="valeur_case">

      <input type = "radio" name ="radio1" value ="valeur_radio">

- Liste de sélection avec options à choix unique :

```
<select name ="select" size="1">
<option value = "un"> choix </option>
<option value ="deux"> choix2 </option>
</select>
```

- Liste de sélection avec options à choix multiples :

```
<select name ="select" size = "1" multiple>
<option value = "un"> choix1 </option>
<option value = "deux"> choix2 </option>
</select>
```

PHP et les formulaires(3)

- Méthodes d'envoi get et post
  - transmission selon une des deux méthodes d'envoi GET ou POST

    - La méthode GET place les informations d'un formulaire directement à la suite de l'adresse URL de la page appelée.
      - http://www.site.com/cible.php?champ=valeur&champ2=valeur

      - inconvénients :
        - rendre visibles les données dans la barre d'adresse du navigateur.
        - De plus, la longueur totale est limitée à 255 caractères, ce qui rend impossible la transmission d'un volume de données important

    - La méthode POST regroupe les informations dans l'entête d'une requête HTTP
      - Assure une confidentialité efficace des données

PHP et les formulaires(4)

- Récupération des paramètres en PHP
  - Si la directive register_globals de php.ini est TRUE, lors de la soumission, chaque élément de saisie est assimilé à une variable PHP dont le nom est constitué par la valeur de l'attribut name et son contenu par la valeur de l'attribut value

    A DECONSEILLER

  - Les tableaux associatifs $_GET et $_POST contiennent toutes les variables envoyées par un formulaire

PHP et les formulaires(5)

```
<form method="GET | POST" action="page_cible.php">
<input type="text" name="Champ_saisie" value="Texte« >
<select name="Liste_Choix" size="3">
<option value="Option_1">Option_1</option>
<option value="Option_2">Option_2</option>
<option value="Option_3">Option_3</option>
</select>
<textarea name="Zone_Texte" cols="30" rows="5"> Texte par défaut  </textarea>
   <input type="checkbox" name="Case_Cocher[]" value="Case_1"> Case 1<br>
   <input type="checkbox" name="Case_Cocher[]" value="Case_2"> Case 2<br>
   <input type="checkbox" name="Case_Cocher[]" value="Case_3"> Case 3<br>
   <input type="radio" name="Case_Radio" value="Case radio 1"> radio 1<br>
   <input type="radio" name="Case_Radio" value="Case radio 2"> radio 2<br>
   <input type="radio" name="Case_Radio" value="Case radio 3"> radio 3<br>
   <input type="reset" name="Annulation" value="Annuler">
 <input type="submit" name="Soumission" value="Soumettre">
</form>
```

PHP et les formulaires(6)

```
<?php
  $resultat = $_GET["Champ_saisie"] . "<br>";
  $resultat .= $_GET["Liste_Choix"] . "<br>";
  $resultat .= $_GET["Zone_Texte"] . "<br>";
  for ($i = 0; $i < count($_GET["Case_Cocher"]); $i++)
  {
    $resultat .= $_GET["Case_Cocher"][$i] . "<br>";
  }
  $resultat .= $_GET["Case_Radio"] . "<br>";
  echo $resultat;
  ?>
```

PHP et les formulaires(7)

- PHP et les formulaires
  - La plupart des éléments d'un formulaire n'acceptent qu'une seule et unique valeur, laquelle est affectée à la variable correspondante dans le script de traitement.

```
$Champ_Saisie ← "Ceci est une chaîne de caractères.";
```

  - Pour des cases à cocher et les listes à choix multiples, plusieurs valeurs peuvent être sélectionnées entraînant l'affectation d'un tableau de valeurs aux variables correspondantes.

```
$Case_Cocher[0] ← "Case radio 1";
$Case_Cocher[1] ← "Case radio 3";
```

Autres Fonctions PHP: Les dates et les heures

- **Les fonctions de date et d'heure**

true | false = checkdate(mois, jour, année);

vérifie la validité d'une date.

$chaine = date(format [, nombre]);

retourne une chaîne de caractères date/heure selon le format spécifié et représentant la date courante par défaut.

$tableau = getdate([nombre]);

retourne les éléments de date et d'heure dans un tableau associatif.

$tableau = gettimeofday();

retourne l'heure courante dans un tableau associatif.

$chaine = gmdate(format [, nombre]);

retourne une chaîne de caractères date/heure GMT/CUT selon le format spécifié et représentant la date courante par défaut.

$nombre = gmmktime(heure, minute, seconde, mois, jour, année [, 1/0]);

retourne l'instant UNIX d'une date GMT spécifiée et avec éventuellement une heure d'hiver

Les cookies (1)

- Principe
  - Un cookie est un fichier texte créé par un script et stocké sur l'ordinateur du visiteur d'un site
  - Les cookies permettent de conserver des renseignements utiles sur chaque utilisateur, et de les réutiliser lors de sa prochaine visite
    - Exemple : personnaliser la page d'accueil ou les autres pages du site
      - un message personnel comportant par exemple son nom, la date de sa dernière visite, ou tout autre particularité.
  - Les cookies étant stockés sur le poste client, l'identification est immédiate et ne concernent que les renseignements qui le concernent
  - Pour des raisons de sécurité, les cookies ne peuvent être lus que par des pages issues du serveur qui les a créés

Les cookies(2)

- Principe
  - Le nombre de cookies qui peuvent être définis sur le même poste client est limité à 20 et la taille de chacun est limitée à 4ko.
  - Un navigateur peut stocker un maximum de 300 cookies
  - La date d'expiration des cookies est définie de manière explicite par le serveur web chargé de les mettre en place.
  - Les cookies disponibles sont importés par PHP sous forme de variables identifiées sous les noms utilisés par ces cookies
  - La variable globale du serveur $_COOKIES enregistre tous les cookies qui ont été définis

Les cookies(3)

- Exemple d'application des cookies
  - Mémorisation des paniers dans les applications d'e-commerce
  - Identification des utilisateurs
  - Des pages web individualisées
    - Afficher des menus personnalisés
    - Afficher des pages adaptées aux utilisateurs en fonction de leurs précédentes visites

Les cookies(4)

- Écrire des cookies
  - L'écriture de cookies est possible grâce à la fonction setcookie()
    - il faut cette fonction dès le début du script avant l'envoi d'aucune autre information de la part du serveur vers le poste client.

```
Setcookie("nom_var", "valeur_var", date_expiration, "chemin",
  "domain", " secure")
```

```
<?php
setcookie ("PremierCookie", "Salut", time() +3600*24*7) ;
…
if (!$PremierCookie) {
  echo "le cookie n'a pas été défini";}
else {
echo $premierCookie, "<br>";
}
?>
```

Les cookies(5)

- Écriture de cookies
  - **Nom_var :** nom de la variable qui va stocker l'information sur le poste client et qui sera utilisée pour récupérer cette information dans la page qui lira le cookie.
    - C'est la seule indication obligatoire pour un cookie
  - **Valeur_var :** valeur stockée dans la variable. Par exemple une chaîne de caractères ou une variable chaîne, en provenance d'un formulaire
  - **Date_expiration :** la date à laquelle le cookie ne sera plus lisible et sera effacé du poste client
    - on utilise en général la date du jour, définie avec la fonction time() à laquelle on ajoute la durée de validité désirée
    - Si l'attribut n'est pas spécifié, le cookie expire à l'issue de la session

Les cookies(6)

- Écriture de cookies
  - **Chemin :** définit la destination (incluant les sous-répertoire) à laquelle le navigateur doit envoyer le cookie.
  - **Domain set :** le nom du domaine à partir duquel peuvent être lus les cookies.
    - On peut aussi utiliser la variable d'environnement $SERVER_NAME à la place.
  - **Secure :** un nombre qui vaut 0 si la connexion n'est pas sécurisée, sinon, il vaut 1 pour une connexion sécurisée

Les cookies(7)

- Lecture de cookies
  - Si la directive register_globals de php.ini est TRUE, un cookie sera automatiquement passé en paramètre au script et sa valeur sera directement accessible dans la variable $NomDuCookie.
    A DECONSEILLER
  - Les cookies sont accessibles dans le tableau associatif $_COOKIE
    - Si celui-ci visite une des pages PHP de ce même domaine dont le chemin est inclut dans le paramètre chemin (si le chemin est / le cookie est valide pour toutes les pages de ce site).
    - Il faut d'abord vérifier l'existence des variables dont les noms et les valeurs ont été définis lors de la création du cookie.
      - Cette vérification s'effectue grâce à la fonction isset($_COOKIE["nom_var"]) qui renvoie true si la variable $nom_var existe et false sinon.

```
<?php
if (isset($_COOKIE["nom_var"]{
echo "<h2> Bonjour isset($_COOKIE["nom_var"]</h2>" ;}
else echo "pas de cookie" ;
?>
```

Les cookies(8)

- Écriture de plusieurs variables par un cookie
  - Utilisation de la fonction compact() pour transformer les variables en un tableau
  - Convertir le tableau en une chaîne de caractère à l'aide de la fonction implode()
  - Affecter la chaîne créée à l'attribut « nom_cookie »

```
<?
$col="#FF0000" ;
$size=24;
$font="Arial" ;
$text="Je suis le cookie« ;
$arr=compact("col" , " size" , " font" , "text" );
$val=implode(" &" , $arr);
Setcookie("la_cookie" , $val, time()+600);
```

Les cookies(9)

- Lecture de plusieurs variables d'un cookie
  - Décomposé la chaîne stockée par la cookie et retrouver un tableau en utilisant la fonction explode()

```
<?php
echo " <b> voici le contenu de la chaîne cookie : </b><br>";
echo $le_cookie, "<br> <br>";
}
$arr=explode("&", $la_cookie);
echo "<b> ces variables ont été établies à partir de la chaîne
   cookie : </b> <br> <br>";
foreach ($arr as $k=>$elem) {
echo "$k=>$elem <br>";
}
?>
```

Les cookies(10)

- Supprimer un cookie
  - Il suffit de renvoyer le cookie grâce à la fonction setcookie() en spécifiant simplement l'argument NomDuCookie

```
<?php
setcookie("Visites");
?>
```

  - Une autre méthode consiste à envoyer un cookie dont la date d'expiration est passée:

```
<?php
setcookie("Visites","",time()-1 )
?>
```

Les cookies(11)

- Quelques précisions sur les cookies
  - Setcookie() doit être utilisée avant l'envoi de données HTML vers le navigateur
  - Le cookie n'est pas visible avant le prochain chargement de page
  - Avec PHP3, si plusieurs cookies sont envoyées de suite, les appels seront traités en ordre inverse, alors qu'avec PHP4 il seront traités dans l'ordre
  - Certains navigateurs ne traitent pas bien certains cas liés aux cookies
    - Microsoft Internet Explorer 4 avec le Service Pack 1 ne traite pas correctement les cookies qui ont le paramètre chemin défini
    - Netscape Communicator 4.05 et Microsoft Internet Explorer 3.x ne traitent pas correctement les cookies qui n'ont pas les paramètres chemin et expiration définis.

Les cookies(12)

- Exemples d'utilisation de cookies

```
//Un script permettant de savoir si un visiteur est déjà venu sur le site pendant le mois
setcookie("Visites","Oui",time()+2592000,"/", " .mondomaine.fr",0);
//  Envoi d'un cookie qui disparaitra après la fermeture du navigateur
SetCookie("UserSessionCookie",$login.":".$pass);
// Envoi d'un cookie qui restera présent 24 heures
SetCookie("DejaVisite","1",time()+3600*24,"/",". mondomaine.fr ",0);
// Envoi d'un cookie qui s'effacera le 1er janvier 2003
SetCookie("An2002","1",mktime(0,0,0,1,1,2003),"/",".   mondomaine.fr ",0);
//script permettant de compter le nombre de visite de la page par le visiteur
    <?php
    $Visites++;
    setcookie("Visites",$Visites,time()+2592000,"/",".  mondomaine.fr ",0);?>
```

Les sessions

- Principe
  - Est un mécanisme permettant de mettre en relation les différentes requêtes du même client sur une période de temps donnée.
  - Les sessions permettent de conserver des informations relatives à un utilisateur lors de son parcours sur un site web
  - Des données spécifiques à un visiteur pourront être transmises de page en page afin d'adapter personnellement les réponses d'une application PHP
  - Chaque visiteur en se connectant à un site reçoit un numéro d'identification dénommé identifiant de session (SID)
  - La fonction session_start() se charge de générer automatiquement cet identifiant unique de session et de créer un répertoire. Elle doit être placée au début de chaque page afin de démarrer ou de continuer une session.

```
<?php
  session_start();
  $Session_ID = session_id();
  // $Session_ID = 7edf48ca359ee24dbc5b3f6ed2557e90    ?>
```

Principe

- Un répertoire est créé sur le serveur à l'emplacement désigné par le fichier de configuration php.ini, afin de recueillir les données de la nouvelle session.

```
[Session]
session.save_path= C:\PHP\sessiondata
; Rép session = \sess_7edf48ca359ee24dbc5b3f6ed2557e90
```

- Le fichier php.ini peut également préciser un nom de session par l'option session.name ou sa durée de vie par session.gc_maxlifetime

- La session en cours peut être détruite par la fonction session_destroy(). Cette commande supprime toutes les informations relatives à l'utilisateur.

```
session_destroy();
```

- Le traitement des variables de session
  - Les variables de session sont chargées dans une session par l'intermédiaire de la fonction session_register()

```
<?php
 session_start();
 session_register("nom_variable");
 ...
 session_register("nom_variableN");
?>
```

  - Une fois la variable enregistrée, elle est accessible à travers le tableau associatif $_SESSION["nom_variable"]

Le traitement des variables de session

Le transport des informations entre les documents est réalisé par l'entremise

- soit d'un cookie
- soit d'une requête HTTP
  - Cette dernière solution est la plus fiable puisque les cookies peuvent ne pas être acceptés par le client ou celui-ci pourrait les détruire en cours de session.
  - Il suffit de concaténer l'identifiant de session à l'adresse URL de la page cible pour que cette dernière puisse accéder aux informations conservées par la session.

```
echo '<a href=" http://www.site.com/doc.php? '. session_name(). '=' . session_id().'>
    lien</a>'
```

Le traitement des variables de session

- Par défaut, PHP tente de passer par les cookies pour sauvegarder l'identifiant de session dans le cas où le client les accepterait. Il est possible d'éviter cela, il suffit de désactiver l'option de configuration session.use_cookies dans le fichier php.ini.

```
[Session]
session.use_cookies 0; //désactive la gestion des sessions par cookie
```

- Exemple:

```
<!-- Fichier : formulaire.html -->
<html><body>
  <form method="post" action="traitement.php">
   <table border="0">
    <tr>
     <td><u>Nom :</u></td>
     <td><input type="text" name="Nom" size="20" value="RIVES"></td></tr>
    <tr>
     <td><u>Prénom :</u></td>
     <td><input type="text" name="Prenom" size="20" value="Jean-Pierre"></td></tr>
    <tr>
     <td><u>eMail :</u></td>
     <td><input type="text" name="cEmail" size="20" value=”du@du.com"></td></tr>
    <tr><td> </td>
     <td><input type="submit" name="soumettre" value="Envoyer"></td></tr></table>
</form>
</body
</html>
```


```
<?
  session_start();
  $nom = $_POST["Nom"];
  $prenom = $_POST["Prenom"];
  $email = $_POST["cEmail"];
  session_register("nom");
  session_register("prenom");
  session_register("email");
  $_SESSION["nom"]=$nom;
  $_SESSION["prenom"]=$prenom;
  $_SESSION["email"]=$email;
  header("Location: session.php?" . session_name() . "=" .
    session_id());
?>
```


```
<?
    session_start();
?>
<html><body><?
echo("<u>Identifiant de session :</u> <b>"
                        . session_id() . "</b><br>");
    echo("<u>Nom de la session :</u> <b>"
                        . session_name() . "</b><br><br>");
    echo("<u>Nom :</u> <b>". $_SESSION["nom"] . "</b><br>");
    echo("<u>Prénom :</u> <b>" . $_SESSION["prenom"] . "</b><br>");
    echo("<u>eMail :</u> <b>" . $_SESSION["email"] . "</b><br>");
    //session_destroy();
  ?>
 </body>
</html>
```

Les fonctions de sessions

session_start() -- Initialise les données de session

session_id() -- Affecte et/ou retourne l'identifiant de session courante

session_name() -- Affecte et/ou retourne le nom de la session courante

session_register() -- Enregistre une variable dans la session courante

session_destroy() -- Détruit toutes les données enregistrées d'une session

session_is_registered() -- Indique si une variable a été enregistrée dans la session ou pas

session_unregister() -- Supprime une variable dans la session courante

session_unset() -- Détruit toutes les variables de session

session_cache_expire() -- Retourne la date d'expiration du cache de la session

session_save_path() -- Affecte et/ou retourne le chemin de sauvegarde de la session courante

session_decode() -- Décode les données de session à partir d'une chaîne

session_encode() -- Encode les données de session dans une chaîne

Quelles sont les erreurs possibles ?

- **Répertoire de session inaccessible**

Warning: open(/tmp\sess_3c80883ca4e755aa72803b05bce40c12, O_RDWR) failed: m (2) in c:\phpdev\www\bp\header.php on line 2

Le répertoire de sauvegarde est défini dans le php.ini : session.save_path = /tmp

Il faut donc

Créer un répertoire

Lui donner les droits d'écriture pour tous

En spécifier le chemin dans le php.ini

- **PHP n'est pas autorisé à utiliser les sessions**

Il faut s'assurer que le PHP est bien autorisé a créer des sessions. C'est juste un parameter à activer. Faire un phpinfo() pour voir ces paramètres. La commande phpinfo() se contente d'afficher dans le navigateur le contenu du fichier de configuration php.ini.

Quelles sont les erreurs possibles ?

- **Avoir déjà écrit dans la page**

Warning: Cannot send session cookie - headers already sent by (output started at /home/SiteWeb/SiteAnalyse/index.php:3) in /home/SiteWeb/SiteAnalyse/index.php on line 6

Cette erreur survient lorsqu'on tente d'ouvrir une session après avoir déjà écrit dans le document, ce qui interdit.

Ce qu'il ne faut pas faire :

```
<html>
<body>
<?php session_start();
...
```

ceci non plus :

```
<?php echo "<html>";
...
session_start();
```

La gestion des connexions aux sites web (1)

- Principe
  - Le langage PHP dispose de plusieurs outils permettant de gérer les connexions des utilisateurs sur un site web
    - Il existe trois états possibles en ce qui concerne le statut des connexions.
      1. **NORMAL** : signifie que la connexion est ouverte et le script est en cours d'exécution.
      2. **ABORTED** : signifie que le client a annulé la connexion et le script est arrêté par défaut.
      3. **TIMEOUT** : signifie que le script a provoqué une déconnexion due à la fin de la durée d'exécution convenue.
    - **Les deux états ABORTED et TIMEOUT peuvent survenir en même temps** dans le cas ou d'une part si PHP est configuré de telle sorte à ignorer les déconnexions et d'autre part lorsque le script arrive à expiration

Principe

- Le langage PHP dispose de plusieurs outils permettant de gérer les connexions des utilisateurs sur un site web

  - L'état ABORTED en principe interrompt logiquement le script dès que l'utilisateur se déconnecte du site. Toutefois, il est possible de modifier ce comportement en activant l'option de configuration ignore_user_abort dans le fichier php.ini.

  - Si une fonction de fermeture a été enregistrée avec l'instruction register_shutdown_function, le moteur de script se rendra compte après que le client se soit déconnecté puis lors de la prochaine exécution du script que celui ci n'est pas terminé, ce qui provoquera l'appel de la fonction de fermeture mettant fin au script.

  - Lorsque le script se termine normalement, cette fonction sera également appelée.

Principe

- L'instruction connection_aborted permet d'exécuter une fonction spécifique à la déconnexion d'un client. Si la déconnexion est effective, la fonction connection_aborted retourne true.
- Un script expire par défaut après une durée de 30 secondes. Néanmoins, ce temps peut être modifié par l'intermédiaire de l'option de configuration max_execution_time dans le fichier php.ini.
- Une fonction se chargeant de terminer le script sera appelée si le délai arrive à expiration ou si le client se déconnecte.
- Les fonctions connection_timeout, connection_aborted et connection_status permettent respectivement de vérifier si l'état de la connexion est ABORTED, TIMEOUT et les trois états possibles.

- Les fonctions de connexions

connection_aborted();

vérifie si le client a abandonné la connexion.

connection_status();

retourne le statut de la connexion.

connection_timeout();

vérifie l'expiration du script en cours.

ignore_user_abort(paramètre);

détermine si lors de la déconnexion d'un client, le script doit continuer ou arrêter son exécution.

Le paramètre peut valoir soit '0' pour un comportement normal, '1' pour un abandon et '2' pour une expiration.

Les en-têtes HTTP (1)

- Principe
  - Les entêtes sont des informations envoyées lors de chaque échange par le protocole HTTP entre un navigateur et un serveur
    - Informations sur les données à envoyer dans le cas d'une requête
  - Permettent aussi d'effectuer des actions sur le navigateur comme le transfert de cookies ou bien une redirection vers une autre page
    - Ce sont les premières informations envoyées au navigateur (pour une réponse) ou au serveur (dans le cas d'une requête),
      - elles se présentent sous la forme: en-tête: valeur
  - la syntaxe doit être rigoureusement respectée
    - aucun espace ne doit figurer entre le nom de l'en-tête et les deux points (:). Un espace doit par contre figurer après celui-ci

Principe

- PHP fournit une fonction permettant d'envoyer des en-tête HTTP manuellement du serveur au navigateur
  - booléen header(chaîne en-tête HTTP)
- La fonction header() doit être utilisée avant tout envoi de données HTML au navigateur
- Exemples
  - Rediriger le navigateur vers une nouvelle page:

```
<?
header("location: http://www.monsite.fr/"); ?>
```

  - Pour envoyer au navigateur une image créé à la volée

```
<?
header("Content-Type: image/gif");
imagegif($image); // envoi de l'image au navigateur // code générant l'image
?>
```

Récupérer les en-têtes de la requête

- Alors que la fonction header() permet d'envoyer des en-têtes HTTP au navigateur, PHP fournit une seconde fonction permettant de récupérer dans un tableau l'ensemble des en-têtes HTTP envoyées par le navigateur

  Tableau getallheaders();

  - Le tableau retourné par la fonction contient les en-têtes indexés par leur nom
  - Exemple : un script permettant par exemple de récupérer des en-têtes particuliers.

```
<?
$headers = getallheaders();
foreach ($headers as $nom => $contenu) {
echo "headers[$nom] = $contenu<br />\n";
}
?>
```

- Les fonctions HTTP

header(chaîne);

envoie une entête HTTP avant toute commande PHP.

true | false = headers_sent();

vérifie si les entêtes HTTP ont bien été envoyés

setcookie(nom, valeur [, date_expiration [, chemin [, domaine [, sécurisation]]]]);

envoie un cookie.

- Les fonctions HTTP

header(chaîne);

envoie une entête HTTP avant toute commande PHP.

true | false = headers_sent();

vérifie si les entêtes HTTP ont bien été envoyés

setcookie(nom, valeur [, date_expiration [, chemin [, domaine [, sécurisation]]]]);

envoie un cookie.

Mail (1)

- La fonction mail envoie un message électronique.

Syntaxe :

mail($recipient, $subject, $message[, $headers, $params]);

Exemple :

```
$message = "Bonjour, ceci est mon message.’’;
$objet ="Bonjour"
mail(‘’info@aricia.fr’’, $objet, $message);
```

Cette fonction ne marche que si un programme de messagerie électronique (appelé « mailer ») est préalablement installé sur le serveur.

- Exemple plus complet :

```
<?php
$recipient = ‘’Tony <tony@tony.com>, ‘’;
$recipient .= “’’Peter <peter@peter.net>’’;
$subject = ‘’Notre rendez-vous’’;
$message = “’’Je vous propose le samedi 15 juin \n’’;
$message .= ’’--\r\n’’;            // Délimiteur de signature
$message .= ‘’Hugo’’;
$headers = ‘’From: Hugo <hugo@hugo.com>\n’’;
$headers .= ‘’Content-Type: text/html; charset=iso-8859-1\n’’ ;
$headers .= ‘’Cc: bruno@aricia.fr\n’’;
mail($recipient, $subject, $message, $headers);
?>
```

Principes de sécurité (1)

- veiller à limiter au strict nécessaire les droits et permissions de l'utilisateur : il vaut mieux l'entendre se plaindre de son manque de liberté plutôt que de constater les dégâts causés par une trop grande libéralité (principe universel)
- ne jamais faire confiance aux données transmises par l'utilisateur (encore moins à l'internaute !)
  - **SQL Injection**
  - **Javascript injection**
- toujours tester l'existence / la validité d'un fichier / code à inclure : et s'il n'était pas celui que vous croyiez ?
- Regarder régulièrement les fichiers journaux (logs)
- Mettre des “alarmes” dans votre code, aux endroits stratégiques (authentification), envoyées par mail ou stockées dans une base de données

- un contrôle des données côté client est bien pratique, mais illusoire : quoi de plus simple que de désactiver Javascript ? Seul un contrôle côté serveur peut prétendre être efficace.
- préférer, quand cela est possible, la méthode POST à la méthode GET : les variables dans l'url, ça fait mauvais genre
- ne pas confondre complexité et sécurité : si votre système de sécurité vous échappe, considérez que vous n'êtes pas en sécurité. De toute façon, vous n'êtes jamais en sécurité.
- s'efforcer de tenir à jour système serveur et interpréteur PHP avec les dernières versions stables : quand la catastrophe aura eu lieu, ce sera ça de moins à vous reprocher...

Masquer PHP

Configurer le serveur web pour qu'il utilise plusieurs extensions de fichiers avec PHP

Vous pouvez utiliser des informations déroutantes comme ceci :

- **Masquer PHP avec un autre langage**

```
# Faire que le code PHP ressemble à un autre langage
   AddType application/x-httpd-php .asp .py .pl
```

- **Masquer PHP avec des types inconnus**

```
# Faire que le code PHP ressemble à un autre langage qui n'existe pas
   AddType application/x-httpd-php .bop .foo .133t
```

- **cachez-le sous forme de html.**

Cela a un léger impact négatif sur les performances générales, car tous les fichiers HTML seront aussi analysés et traités par le moteur PHP :

Utiliser le type html pour les extensions PHP

```
# Faire que le code PHP ressemble à du html
   AddType application/x-httpd-php .htm .html
```