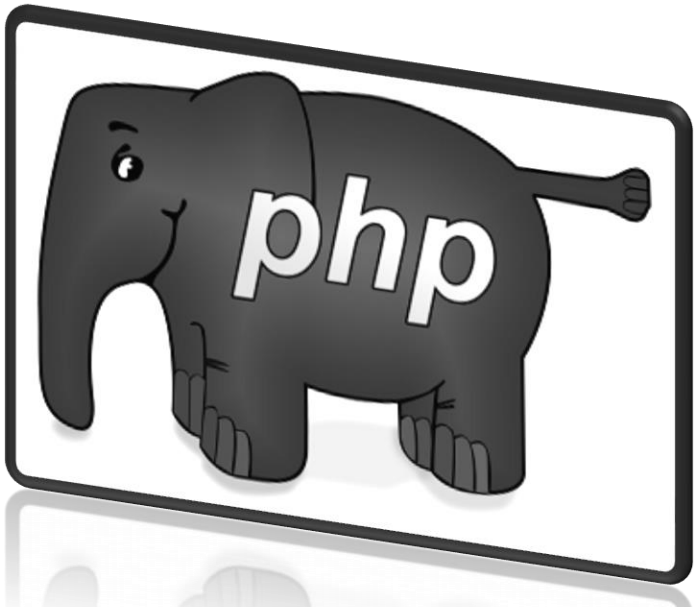
Langage PHP



FAQIR Nada

(nadafaqir@gmail.com)

10-Les cookies et Les sessions

Introduction

- Ce nouveau chapitre introduit le mécanisme des cookies. Nous définirons ensemble ce qu'est un cookie et à quoi il sert. Puis nous aborderons les principes de sécurité relatifs aux cookies. Ensuite nous apprendrons à générer et lire le contenu d'un cookie. Nous terminerons cette partie sur la suppression du cookie, le stockage des valeurs d'un type complexe par sérialisation / désérialisation avant de conclure sur les cas d'utilisation les plus fréquents.

Qu'est-ce qu'un cookie ?

- Le mécanisme des cookies a été inventé par la société Netscape dans le but de palier à certaines faiblesses du protocole HTTP mais aussi d'étendre les possibilités de relation entre le client et le site web. Leur fonction est le stockage, pendant une période donnée, d'une information relative à l'utilisateur (son pseudo, sa date de dernière connexion, son âge, ses préférences...).
- En pratique, les cookies sont de simples fichiers textes ne pouvant excéder 4Ko. Ils sont stockés sur le disque dur de l'internaute et gérés par les navigateurs (Firefox, Internet Explorer, Safari, Opéra, AvantBrowser...). Leur acceptation est soumise aux filtres des navigateurs. En effet, ces derniers sont capables de refuser des cookies. Leur utilisation doit donc être scrupuleusement réfléchie.
- Pour des raisons de sécurité, des "normes" ont fixé à 20 le nombre maximal de cookies envoyés pour un même domaine.

Qu'en est-il de la sécurité ?

- Il y'a encore quelques années les cookies faisaient peur. Certains profanes se persuadaient qu'ils étaient dangereux, qu'ils pouvaient exécuter des programmes malicieux sur leur ordinateur ou bien récupérer des informations personnelles ou confidentielles. Or ce n'est pas le cas. Un cookie n'est ni plus ni moins qu'un fichier texte de très petite taille. Il ne peut donc ni être exécuté ni même lancer des programmes à lui seul. Il ne sert uniquement à stocker une information en vue d'une réutilisation ultérieure. En revanche, rien n'empêche un programme pirate d'être exécuté par l'utilisateur ,qui va récupérer des informations confidentielles ou personnelles, puis les stocker dans un cookie qu'il crée et envoie au serveur Web.
- Ce dernier se chargera de récupérer les informations transmises et de les utiliser contre l'utilisateur.

- La création d'un cookie demande néanmoins le respect de quelques règles de sécurité et de bon sens. Le cookie étant stocké sur le disque dur du client, son accès n'y est donc pas sécurisé. Un cookie pourra être lu, modifié ou supprimé sans difficulté par un utilisateur malintentionné. Il est donc fortement déconseillé de stocker des informations sensibles comme :
  - **des informations confidentielles à l'intérieur (mot de passe par exemple).**
  - **des identifiants de connexion facilement identifiables comme un login.**
- Un cookie devrait généralement ne servir qu'à des fins statistiques, de personnalisation d'affichage ou bien à la multi pagination de formulaires. Et encore dans ce dernier cas, la meilleure solution serait d'avoir recours au mécanisme des sessions.

Génération d'un cookie avec setcookie()

- La création d'un cookie repose sur l'envoi d'entêtes HTTP au navigateur du client au moyen de la fonction setcookie().
- Cela sous-entend donc qu'il faudra l'appeller avant tout envoi de données au navigateur (print(), echo(), tag html, espace blanc...) sous peine de générer une erreur de type *« Warning : Cannot send session cookie - headers already sent by... »*.
- La fonction setcookie() peut prendre jusqu'à 7 paramètres. Seul le premier est obligatoire car il définit le nom du cookie. Elle renvoie un booléen : true en cas de succès et false si échec.

Signature de la fonction setcookie()

- bool setcookie (string name [, string value [, int expire [, string path [, string domain [, bool secure [, bool httponly]]]]]])

- Le tableau ci-dessous récapitule les valeurs que peuvent prendre chacun de ces paramètres.

Paramètre	Explications
name	Nom du cookie
value	Valeur du cookie
expire	Date d'expiration du cookie au format timestamp UNIX, c'est à dire un nombre de secondes écoulées depuis le 01 janvier 1970
path	Chemin sur le serveur dans lequel le cookie sera valide. Si la valeur '/' est spécifiée alors le cookie sera visible sur tout le domaine. Si '/examples/' est précisé alors le cookie sera visible dans le répertoire /examples/ et tous ses sous-dossiers
domain	Domaine sur lequel le cookie sera valable. Si la valeur est '.domaine.com' alors le cookie sera disponible sur tout le domaine (sous-domaines compris). Si la valeur est 'www.domaine.com' alors le cookie ne sera valable que dans le sous-domaine 'www'
secure	Ce paramètre prend pour valeur un booléen qui définit si le cookie doit être utilisé dans un contexte de connexion sécurisée par protocole HTTPS
httponly	Ce paramètre prend pour valeur un booléen qui définit si le cookie sera accessible uniquement par le protocole HTTP. Cela signifie que le cookie ne sera pas accessible via des langages de scripts, comme Javascript. Cette configuration permet de limiter les attaques via XSS (bien qu'elle ne soit pas supportée par tous les navigateurs). Ajouté en PHP 5.2.0

Les paramètres de la fonction setcookie()

Exemple

- Le code suivant illustre la création d'un cookie caractérisé par un nom, une valeur et une date de validité d'un mois. Il simule l'enregistrement du design d'un site que l'utilisateur préfère. Grâce à ce cookie, le site sera affiché avec ce thème graphique automatiquement à la prochaine visite de l'internaute.

Création d'un cookie

```php
<?php
  // Création du cookie
  setcookie('designPrefere','prairie',time()+3600*24*31);
?>
```

Remarques :

- [1] Lorsqu'un cookie est créé depuis une page, il ne devient disponible qu'à partir de la page suivante car il faut que le navigateur envoi le cookie au serveur.
- [2] Un cookie dont la date d'expiration n'est pas précisée est enregistré dans la mémoire vive de l'ordinateur et non sur le disque dur. Il sera effacé à la fermeture du navigateur.

Lecture d'un cookie

- Lorsqu'un internaute interroge un site web repéré par un nom de domaine, son navigateur envoie au serveur la liste des cookies disponibles pour ce domaine. PHP les réceptionne puis construit un tableau associatif superglobal nommé $_COOKIE. Les clés correspondent aux noms des cookies et les valeurs aux valeurs inscrites dans ces derniers. Il devient alors très simple d'accéder à la valeur d'un cookie en appelant le tableau $_COOKIE avec la clé qui convient. L'exemple ci-dessous nous permet de récupérer la valeur du cookie que nous avions créé précédemment.

Lecture d'un cookie

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtm
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
<html>
  <head>
    <title>Lecture d'un cookie !</title>
  </head>
  <body>
    <p>
      Votre thème favoris est :
      <?php
        // Lecture de la valeur du cookie designPrefere
        echo $_COOKIE['designPrefere'];    // Affiche 'prairie'
      ?>
    </p>
  </body>
</html>
```

Remarques :

- [1] Ajouter un couple clé / valeur au tableau $_COOKIE ne crée pas de nouveau cookie.
- [2] Pour modifier la valeur d'un cookie, il faut réutiliser la fonction setcookie().
- [3] Pour connaître tous les cookies utilisés, il faut lister le tableau $_COOKIE au moyen d'une boucle foreach() ou d'un appel à print_r().

Suppression d'un cookie

- Pour supprimer un cookie, il faut de nouveau appeler la fonction setcookie() en lui passant en paramètre le nom du cookie uniquement.
- Dans notre exemple, nous utiliserons le code suivant pour supprimer notre cookie *designPrefere*.

```
<?php
// Suppression du cookie designPrefere
setcookie('designPrefere');
?>
```

Suppression d'un cookie

- Le code précédent demande au navigateur d'effacer le cookie mais ne supprime pas la valeur présente dans le tableau $_COOKIE. Il faut donc penser à supprimer les deux. Pour cela, nous utilisons le script ci-après.

Script de suppression complète d'un cookie

```php
<?php
  // Suppression du cookie designPrefere
  setcookie('designPrefere');
  // Suppression de la valeur du tableau $_COOKIE
  unset($_COOKIE['designPrefere']);
?>
```

Modifier un cookie existant

- Vous vous demandez peut-être comment modifier un cookie déjà existant ? Là encore, c'est très simple : il faut refaire appel à setcookie en gardant le même nom de cookie, ce qui « écrasera » l'ancien.
- Par exemple, si j'habite maintenant en France, je ferai :

```
setcookie('pays','France', time() + 365*24*3600, null, null, false, true);
```

- Notez qu'alors le temps d'expiration du cookie est remis à zéro pour un an.

Les principaux cas d'utilisation des cookies

- Les cookies se révèlent donc très utiles dans des cas de figure particuliers. Parmi eux, nous pouvons recenser :
  - **la sauvegarde du design préféré d'un site pour un internaute**
  - **l'affichage les nouveaux messages depuis la dernière visite de l'internaute**
  - **l'affichage les messages non-lus par le visiteur dans un forum**
  - **la fragmentation d'un formulaire sur plusieurs pages. Les valeurs envoyées sur la première page sont sérialisées et envoyées par cookie à la seconde qui les dé-sérialisera**
  - **les compteurs de visites**
  - **la reconnaissance des visiteurs ayant déjà voté à un sondage**

Conclusion

- Nous retiendrons que les cookies sont un moyen efficace de retenir des informations de faible importance concernant un utilisateur. Néanmoins, pour des raisons de sécurité, leur taille est limitée à 4Ko et leur nombre à 20 pour un même domaine. Il ne faut donc pas les utiliser pour transférer des données nombreuses mais uniquement pour stocker des informations à but statistique ou de personnalisation d'affichage.

Introduction :**Les sessions**

- Les sessions constituent un moyen de conserver des variables sur toutes les pages de votre site. Jusqu'ici, nous étions parvenus à passer des variables de page en page via la méthode GET (en modifiant l'URL : page.php?variable=valeur) et via la méthode POST (à l'aide d'un formulaire).

Introduction

- Mais imaginez maintenant que vous souhaitez transmettre des variables sur toutes les pages de votre site pendant la durée de la présence d'un visiteur. Ce ne serait pas facile avec GET et POST car ils sont plutôt faits pour transmettre les informations une seule fois, d'une page à une autre. On sait ainsi envoyer d'une page à une autre le nom et le prénom du visiteur, mais dès qu'on charge une autre page ces informations sont « oubliées ». C'est pour cela qu'on a inventé les sessions.

Une session c'est quoi ?

- Une session est un mécanisme technique permettant de sauvegarder **temporairement sur le serveur** des informations relatives à un internaute. Ce système a notamment été inventé pour palier au fait que le protocole HTTP agit en mode **non connecté**. A chaque ouverture de nouvelle session, l'internaute se voit attribué un identifiant de session. Cet identifiant peut-être transmis soit en GET (PHPSESSID ajouté à la fin de l'url), POST ou Cookie (cookie sur poste client) selon la configuration du serveur. Les informations seront quant à elles transférées de page en page par le serveur et non par le client. Ainsi, la sécurité et l'intégrité des données s'en voient améliorées ainsi que leur disponibilité tout au long de la session. Une session peut contenir tout type de données : nombre, chaîne de caractères et même un tableau.

- Contrairement à une base de données ou un système de fichiers, la session conserve les informations pendant quelques minutes. Cette durée dépend de la configuration du serveur mais est généralement fixée à 24 minutes par défaut. Le serveur crée des fichiers stockés dans un répertoire particulier.
- Les sessions sont particulièrement utilisées pour ce type d'applications :
  - Les espaces membres et accès sécurisés avec authentification.
  - Gestion d'un caddie sur un site de vente en ligne.
  - Formulaires éclatés sur plusieurs pages.
  - Stockage d'informations relatives à la navigation de l'utilisateur (thème préféré, langues...).

Initialisation (et restauration) d'une session

- PHP introduit nativement une unique fonction permettant de démarrer ou de continuer une session. Il s'agit de session_start(). Cette fonction ne prend pas de paramètre et renvoi toujours **true**. Elle vérifie l'état de la session courante. Si elle est inexistante, alors le serveur la crée sinon il la poursuit.

```
<?php
session_start();
?>
```

Lecture et écriture d'une session

- **1. Le tableau $_SESSION**
- Lorsqu'une session est créée, elle est par défaut vide. Elle n'a donc aucun intérêt. Il faut donc lui attribuer des valeurs à sauvegarder temporairement. Pour cela, le langage PHP met en place le tableau superglobal **$_SESSION**. Le terme *superglobal* signifie que le tableau a une visibilité maximale dans les scripts. C'est à dire que l'on peut y faire référence de manière globale comme locale dans une fonction utilisateur sans avoir à le passer en paramètre. Le tableau **$_SESSION** peut-être indexé numériquement mais aussi associativement. En règle générale, on préfère la seconde afin de pouvoir donner des noms de variables de session clairs et porteurs de sens.

2. L'écriture de session

- Pour enregistrer une nouvelle variable de session, c'est tout simple. Il suffit juste d'ajouter un couple clé / valeur au tableau $_SESSION comme l'illustre l'exemple suivant.

```
<?php
// Démarrage ou restauration de la session
session_start();
// Ecriture d'une nouvelle valeur dans le tableau de session
$_SESSION['login'] = 'Dahmani';
?>
```

- Le tableau $_SESSION, qui était vide jusqu'à présent, s'est agrandit dynamiquement et contient maintenant une valeur (**Dahmani**) à la clé associative **login**. Une variable de session est alors créée.

- ## 3. Lecture d'une variable de session
- Après l'écriture, c'est au tour de la lecture. Il n'y a rien de plus simple. Pour lire la valeur d'une variable de session, il faut tout simplement appeler le tableau de session avec la clé concernée. L'exemple ci-dessous illustre tout ça.

```
<?php
// Démarrage ou restauration de la session
session_start();
// Lecture d'une valeur du tableau de session
echo $_SESSION['login'];   /// affiche dahmani
?>
```

Destruction d'une session

- Comme cela a été évoqué plus haut, le serveur détruit lui même la session au bout d'un certain temps si la session n'a pas été renouvellée. En revanche, il est possible de forcer sa destruction au moyen de la fonction session_destroy(). Cela permet par exemple aux webmasters de proposer une page de déconnexion aux membres loggués à leur espace personnel. Cependant, l'utilisation de session_destroy() seule n'est pas très "propre". Le code suivant présente une manière plus correcte de mettre fin à une session.

Exemple

```
<?php
// Démarrage ou restauration de la session
session_start();
// Réinitialisation du tableau de session
// On le vide intégralement
$_SESSION = array();
// Destruction de la session
session_destroy();
// Destruction du tableau de session
unset($_SESSION);
?>
```

Approche pratique

```php
<?php
// On démarre la session AVANT d'écrire du code HTML
session_start();

// On s'amuse à créer quelques variables de session dans $_SESSION
$_SESSION['prenom'] = 'Jean';
$_SESSION['nom'] = 'Dupont';
$_SESSION['age'] = 24;
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
   <head>
      <title>Titre de ma page</title>
       <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    </head>
    <body>

    <p>
        Salut <?php echo $_SESSION['prenom']; ?> !<br />
        Tu es à l'accueil de mon site (index.php). Tu veux aller sur une autre page ?
    </p>

    <p>
        <a href="mapage.php">Lien vers mapage.php</a><br />
        <a href="monscript.php">Lien vers monscript.php</a><br />
        <a href="informations.php">Lien vers informations.php</a>
    </p>

    </body>
</html>
```

```php
<?php
session_start(); // On démarre la session AVANT toute chose
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" >
   <head>
      <title>Titre de ma page</title>
      <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
   </head>
   <body>

   <p>Re-bonjour !</p>

   <p>
      Je me souviens de toi ! Tu t'appelles <?php echo $_SESSION['prenom'] . ' ' . $_SESSION['nom']; ?> !<br />
      Et ton âge hummm... Tu as <?php echo $_SESSION['age']; ?> ans, c'est ça ? :-D
   </p>

   </body>
</html>
```

Conclusion

- Nous vous recommandons vivement de jeter un œil à la documentation de PHP au sujet des sessions pour regarder du côté des autres fonctions existantes non évoquées ici. Ceci n'était que le fondement théorique des sessions basé sur un exemple concret et pratique. Vous serez à présent en mesure de construire vos propres applications web à partir de sessions.

Les **sessions** sous PHP permettent de **conserver des données d'un utilisateur** coté serveur, lors de sa navigation de page en page du site, notamment lors d'un **processus d'authentification**.

→ les Variables de Session

En général, une **session** est **active tant que le navigateur** coté client est **ouvert** (c'est la durée de vie d'une session).

1) **Principe :**

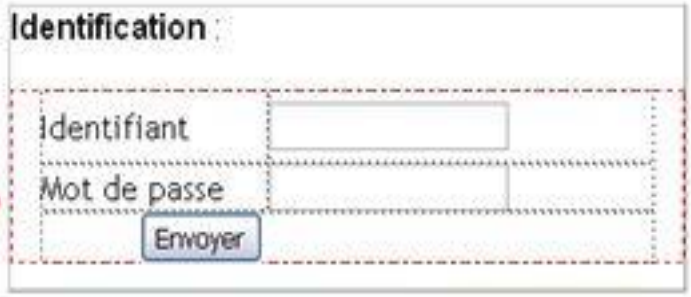
• coté client, un **ID** (identificateur) **de session** est enregistré dans un **cookie**, suite à un en-tête (header) **envoyé par le serveur**.

Lors d'une requête HTTP, le **navigateur renvoie** au serveur le contenu du **cookie**.

2) •passage de l'ID de session par l'**URL** :

Méthode **sans cookie** (si refus du client), **l'ID de session** est **rattaché aux URL** appelées par le client.

https:// permet de mieux sécuriser la session.



Processus d'authentification :

Un **formulaire HTML** permet de saisir l'identifiant et le mot de passe.

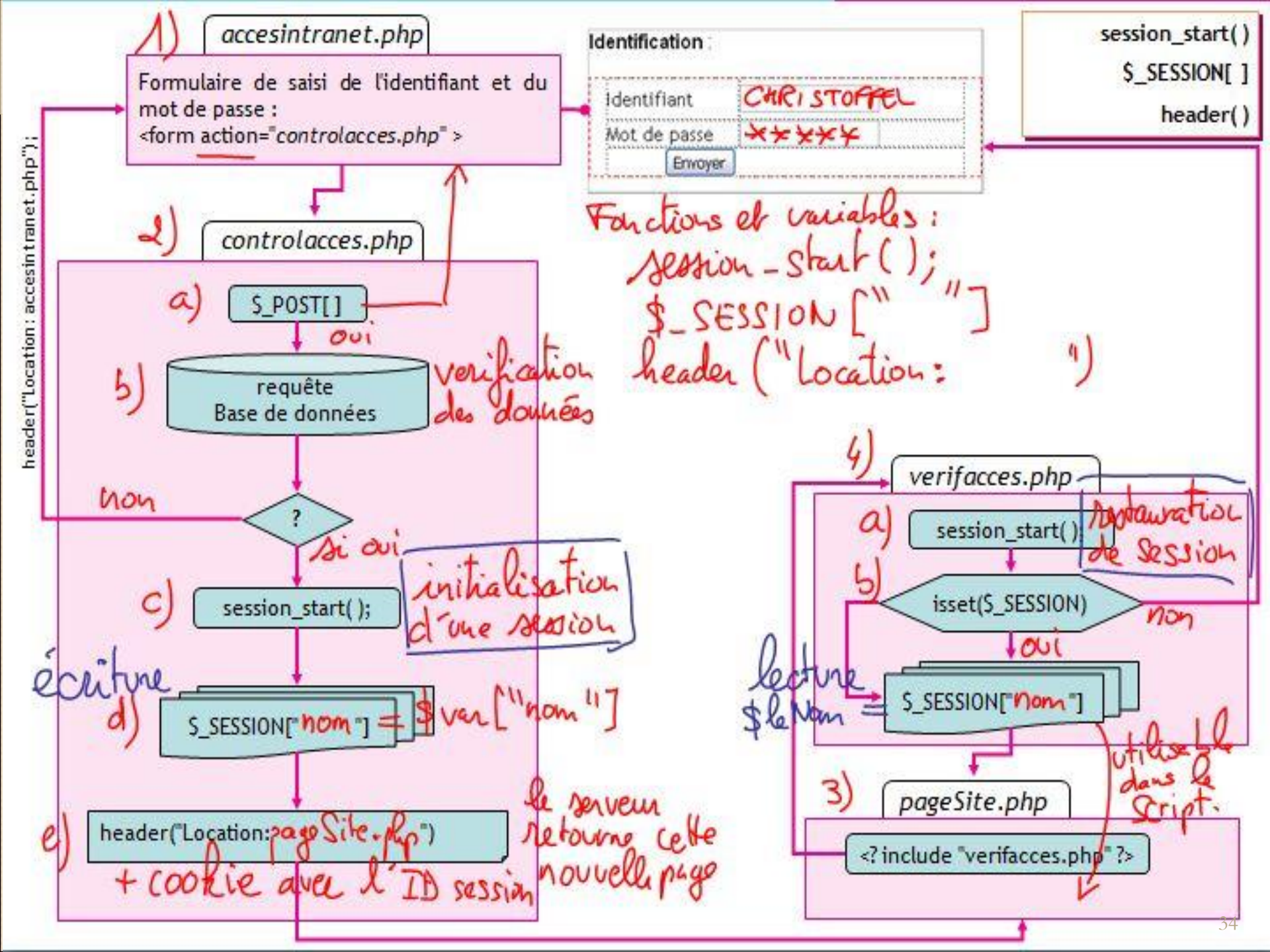
Lors d'un protocole **https://** les **échanges** entre le serveur et le client sont **cryptés**.

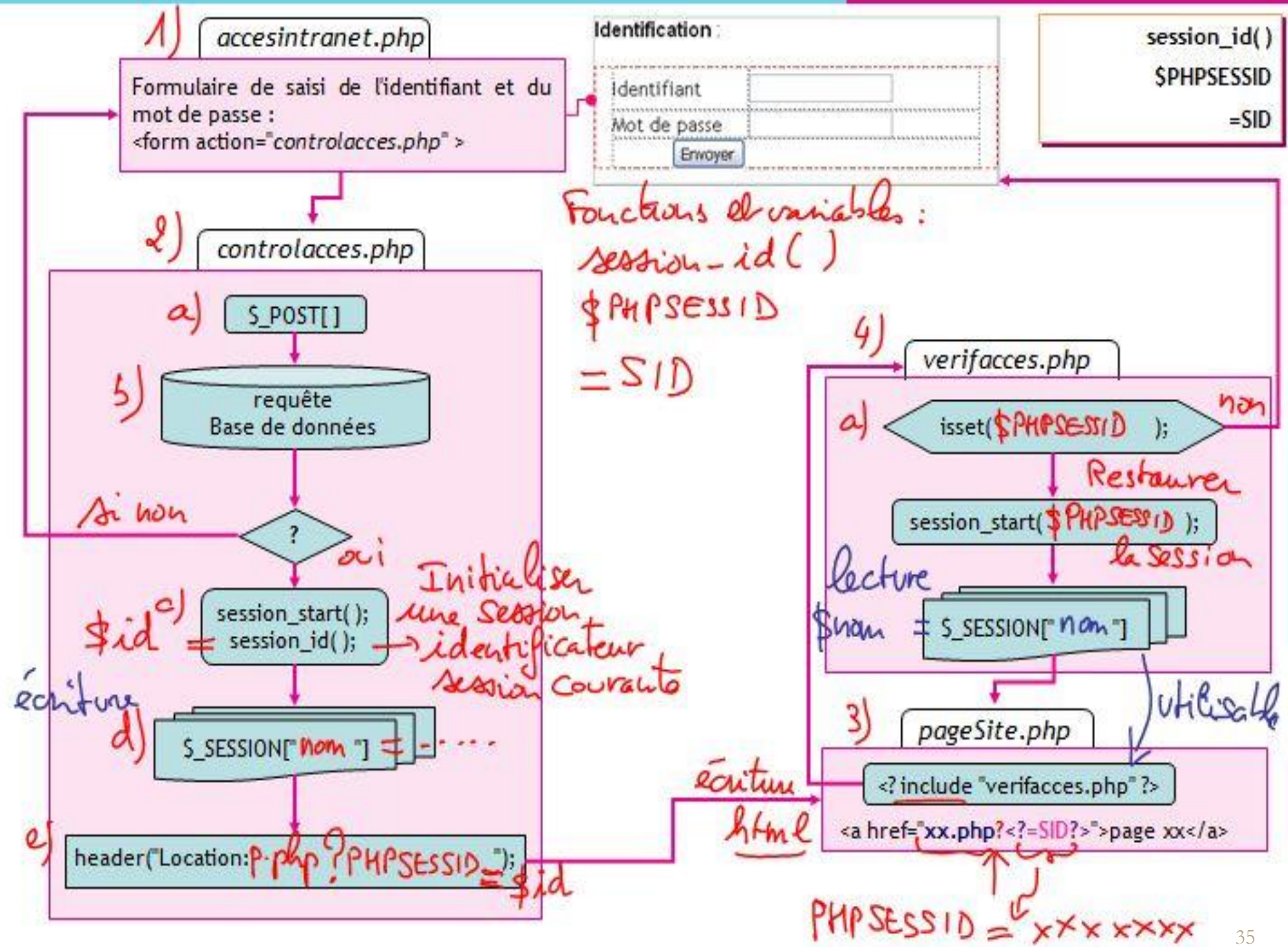
http://cyberzoide.developpez.com/php4/faqsession/

http://www.hsc.fr/ressources/breves/LSO.html.fr

http://www.linux-france.org/article/devl/php3/tut/php3_anx1.html

http://www.phpfrance.com/tutoriaux/index.php/2005/07/20/34-les-sessions-php


1)
accesintranet.php
Formulaire de saisi de l'identifiant et du mot de passe :
<form action="controlacces.php" >
Identification :
identifiant
CHRISTOFFEL
Mot de passe
*****
Enroyer
session_start( )
$_SESSION[ ]
header( )
header("Location : accesintranet.php");
2)
controlacces.php
a)
$_POST[ ]
oui
b)
requête
Base de données
verification des données
non
?
si oui
c)
session_start( );
initialisation d'une session
écriture
d)
$_SESSION["nom"] = $var["nom"]
e)
header("Location:pageSite.php")
+ cookie avec l'ID session
le serveur retourne cette nouvelle page
Fonctions et variables :
session_start( );
$_SESSION[" "]
header ("location:
4)
verifacces.php
a)
session_start( );
restauration de session
b)
isset($_SESSION)
non
oui
lecture
$leNom = $_SESSION["nom"]
utilise le dans le script.
3)
pageSite.php
<? include "verifacces.php" ?>


1) accesintranet.php
Formulaire de saisi de l'identifiant et du mot de passe :
<form action="controlacces.php" >
Identification :
Identifiant
Mot de passe
Envoyer
session_id( )
$PHPSESSID
=SID
Fonctions et variables :
session_id( )
$PHPSESSID
=SID
2) controlacces.php
a) $_POST[ ]
5) requête Base de données
Si non
?
oui
$id a) = session_start( ); session_id( );
Initialiser une session → identificateur session courante
écriture
d) $_SESSION["nom"] = ...
e) header("Location:P.php?PHPSESSID=$id");
écriture html
4) verifacces.php
a) isset($PHPSESSID);
non
Restaurer
session_start($PHPSESSID);
la session
lecture
$nom = $_SESSION["nom"]
utilisable
3) pageSite.php
<? include "verifacces.php" ?>
<a href="xx.php?<?=SID?>">page xx
PHPSESSID = xxx xxxx

FIN

- Merci pour votre attention