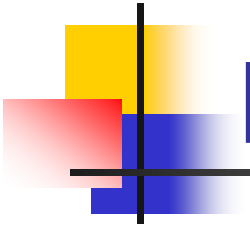
Cookies et Sessions

Sites à voir :

http://www.lephpfacile.com/cours/17-les-cookies
http://www.php-astux.info/sessions-php.php
http://www.vulgarisation-informatique.com/sessions.php



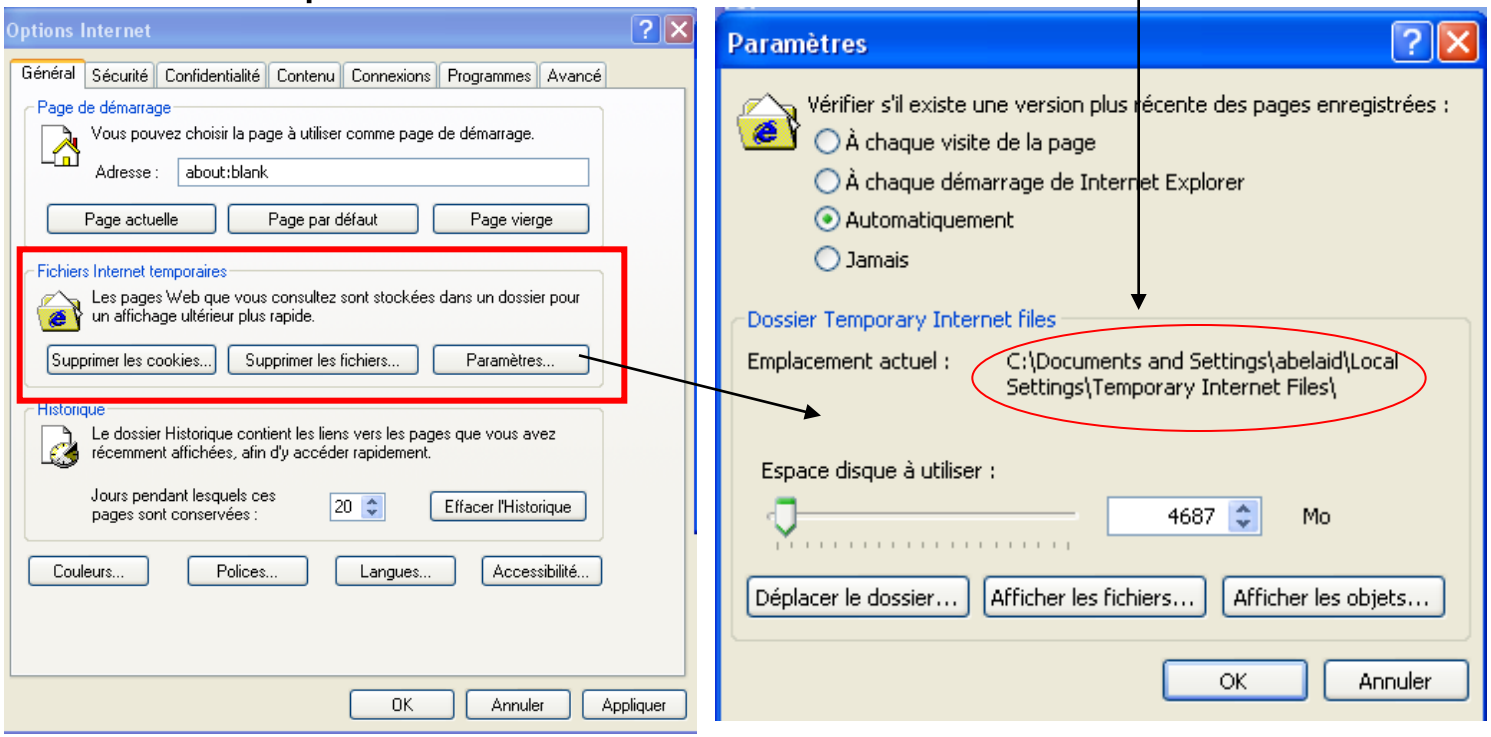
Les cookies

- Qu'est-ce qu'un cookie ?
  - Un petit fichier texte (65 Ko maxi) stocké sur le disque dur du visiteur du site
  - On peut y sauvegarder plusieurs infos concernant ce visiteur et les réutiliser lors de sa prochaine visite
    - Par exemple, on pourrait très bien stocker dans ce cookie le nom du visiteur et par la suite, afficher son nom à chaque fois qu'il se connectera sur le site (ceci bien sur, s'il n'efface pas les cookies de son disque dur)
    - Ceci n'est possible que si le visiteur a entré lui-même ses informations dans un formulaire sur le site

Les cookies

- Où sont placés les cookies ?
  - Cela dépend du navigateur que vous utilisez
  - Pour mon I.E., les cookies sont stockés ici

Outils >> Options Internet



Les cookies

- Où sont placés les cookies pour Mozilla ?
  - Lancer une fenêtre, cliquer sur onglets outils>options
    cliquer sur onglet vie privée
  - Dans le menu déroulant: "utiliser les paramètres personnalisés pour l'historique." bouton "afficher les cookies"

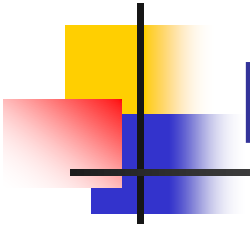
Les cookies

- Voyons à présent comment créer de tels cookies ?
  - On utilise la fonction setcookie()

```
<?php
   /* on définit une durée de vie de notre cookie (en
   secondes), donc un an dans notre cas*/
   $temps = 365*24*3600;
   /* on envoie un cookie de nom pseudo portant la valeur
   Belaid*/
   setcookie ("pseudo", "Belaid", time() + $temps);
?>
```

Les cookies

- Explications
  - On envoie par ce code chez le client (donc le visiteur du site) un cookie de nom **pseudo** portant la valeur **Belaid**
  - De plus, avec **time()**, on impose que le cookie ait une durée de vie de un an (soit en fait l'instant présent plus un an, donc un an)
  - Maintenant, si le visiteur ne supprime pas ce cookie, et bien, dans toutes les pages WEB de notre site, on pourra accéder à la variable **$pseudo** qui contiendra la chaîne de caractères **Belaid** ou celle que vous avez rentrée dans votre formulaire



Les cookies

- Exemple
  - Supposons que sur une page de notre site WEB
    - nous souhaitons faire en sorte que si le visiteur vient pour la première fois (ou qu'il a supprimé ses cookies), il aurait alors, la possibilité de saisir son nom dans un formulaire,
    - ou bien s'il ne s'agit pas de sa première visite, d'afficher tout simplement Bonjour puis son nom
  - On aurait alors le code suivant pour notre page (par exemple **index.php**)

- index.php

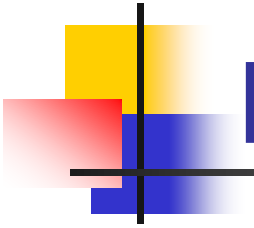
```
<html>
<head>
<title>Index du site</title>
<body>
<?php
    if (isset($_COOKIE['pseudo'])) {
   // si le cookie existe
      echo 'Bonjour '.$_COOKIE['pseudo'].' !';
    }
    else {
      echo 'Notre cookie n\'est pas déclaré.';
      // si le cookie n'existe pas, on affiche un formulaire permettant
      // au visiteur de saisir son nom
      echo '<form action="./traitement.php" method="post">';
      echo 'Votre nom : <input type = "texte" name = "nom"><br />';
      echo '<input type = "submit" value = "Envoyer">';
    }
?>
</body>
</html>
```

- traitement.php

```
<?php
    If (isset($_POST['nom'])) {
      $temps = 365*24*3600;
      // on envoie un cookie de nom pseudo portant la valeur de la variable $nom,
      //c'est-à-dire la valeur qu'a saisie la personne qui a rempli le formulaire
      setcookie ("pseudo", $_POST['nom'], time() + $temps);
      // fonction nous permettant de faire des redirections
      function redirection($url){
                  if (headers_sent()){
                  print('<meta http-equiv="refresh" content="0;URL='.$url.'">');
                  }
                  else {header("Location: $url");}
      }
      // on effectue une redirection vers la page d'accueil
      redirection ('index.php');
    }
    else {
    echo 'La variable du formulaire n\'est pas déclarée.';
    }
?>
```

Les cookies

- Attention !!!
  - Plusieurs conditions sont à respecter afin que l'utilisation des cookies se passe au mieux :
    1. l'envoie d'un cookie doit être la **première fonction PHP** que vous utilisez dans votre script,
       1. ce qui veut dire que vous devez utiliser la fonction **setcookie()** tout en haut de votre script (**AUCUN AFFICHAGE ET AUCUN CODE HTML AVANT UN SETCOOKIE**)
    2. Si d'autres fonctions interviennent avant l'envoi du cookie, celui-ci ne fonctionnera pas



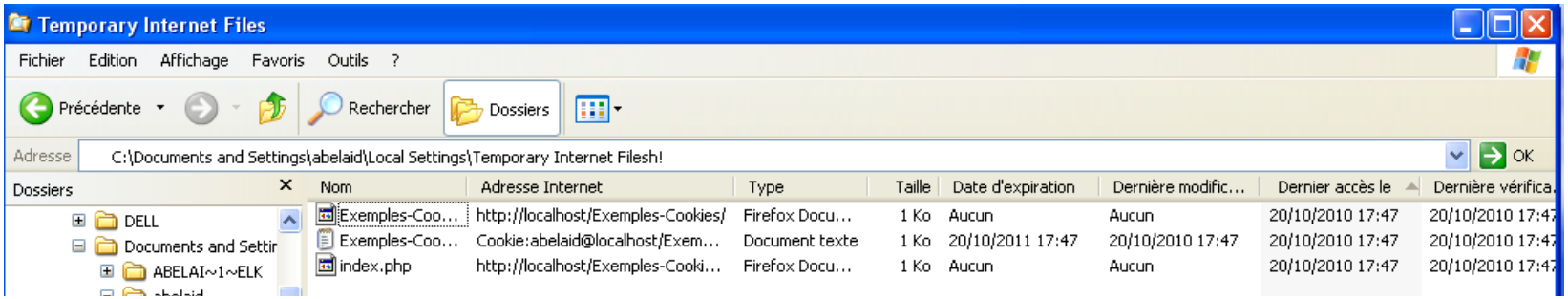
Les cookies

3. Si vous envoyez un cookie sur un poste client, celui-ci **effacera automatiquement l'ancien cookie qui portait le même nom** (si il y en avait un), autrement il le créera
4. **Note :** pour effacer un cookie, vous devez lancer un cookie qui aura le même nom que le cookie que vous voulez effacer, tout en lui donnant une valeur nulle (vous pouvez également l'envoyer avec un temps de vie dépassé)

Les cookies

- Expérience à faire
  - Aller à l'adresse équivalente à celle-ci sur votre I.E.

    C:\Documents and Settings\abelaid\Local Settings\Temporary Internet
  - Vider tout le répertoire
  - Lancer le programme index.php sur I.E. en utilisant : http://localhost/Exemples-Cookies/index.php
  - Regarder ce qui a été créé dans : C:\Documents and Settings\abelaid\Local Settings\Temporary Internet Filesh!

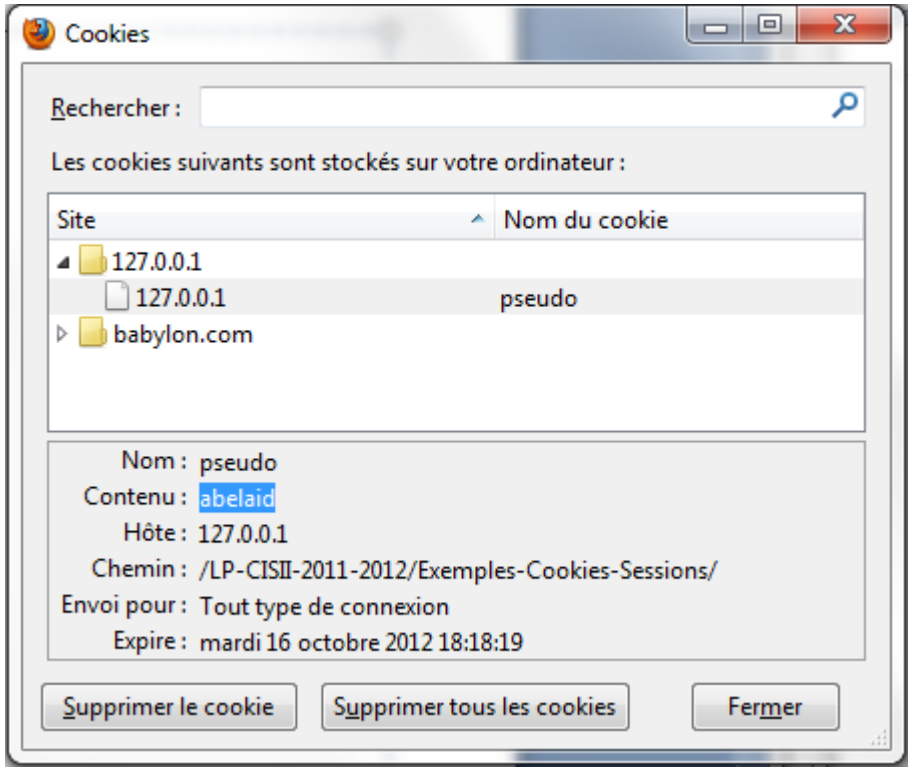


Les cookies

- Regarder le fichier équivalent
  - abelaid@Exemples-Cookies[1].txt
    - [1] indique qu'il y a un seul cookie
  - Son contenu :
    - pseudoBelaidlocalhost/Exemples-Cookies/1024267738662430183231654845424301098 06*
  - On retrouve :
    - pseudo et Belaid

Les cookies

- La même chose sur Firefox
  - Effacer les cookies
  - Lancer index_p.php
  - Puis regarder :



Les cookies

- Un autre exemple
  - http://www.phpfacile.com/creer_un_site_web_en_php/cookies_1.php5
  - Ici nous allons faire une espèce de caddie utilisant des cookies
  - Le premier fichier que l'on appelera *cookie_init.php* servira à mettre à 0 les cookies

```
<?php
   setCookie("panier[pomme]", 0);
   setCookie("panier[poire]", 0);
   setCookie("panier[peche]", 0);
   setCookie("panier[banane]", 0);
   header("Location: cookie_ajout.php");
?>
```

Les cookies

- On met à 0 tous les éléments du panier
- On fait ensuite une redirection vers *cookie_ajout.php*. *setCookie()* et *header()* requièrent tous les 2 qu'il n'y ait pas de données émises avant dans le script
- S'il y avait eu un espace (ou d'autres caractères) devant **<?php**, il y aurait eu une erreur

- *cookie_ajout.php* : on met à jour le bon cookie (grâce à la méthode GET)

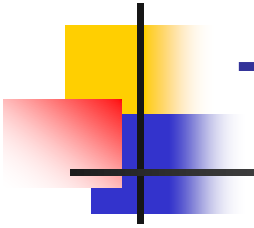
```
<?php
    // Lecture du cookie
    $panier = $_COOKIE["panier"];

    switch ($_GET["ajout"]){
      case "pomme": $panier["pomme"]++;
                    setCookie("panier[pomme]",$panier["pomme"]);break;
      case "poire": $panier["poire"]++;
                    setCookie("panier[poire]",$panier["poire"]);break;
      case "peche": $panier["peche"]++;
                    setCookie("panier[peche]",$panier["peche"]);break;
      case "banane":$panier["banane"]++;
                    setCookie("panier[banane]",$panier["banane"]);
                    break;
    }
?>
```

```
<html>
<head>
<title>Le marché</title>
</head>
<body bgcolor="#FFFFFF">
<table border="4" cellspacing="4" cellpadding="4" align="center">
  <tr align="center">
    <td>Ajouter</td>
    <td>Votre commande</td>
  </tr>
  <tr align="center">
    <td><a href="cookie_ajout.php?ajout=pomme">1Kg pommes</a> (1E)</td>
    <td><?echo $panier["pomme"]?>Kg pomme(s)</td>
  </tr>
  <tr align="center">
    <td><a href="cookie_ajout.php?ajout=poire">1Kg poire</a> (1,5E)</td>
    <td><?echo $panier["poire"]?>Kg poire(s)</td>
  </tr>
  <tr align="center">
    <td><a href="cookie_ajout.php?ajout=peche">1Kg p&ecirc;che</a> (2E)</td>
    <td><?echo $panier["peche"]?>Kg p&ecirc;che(s)</td>
  </tr>
  <tr align="center">
    <td><a href="cookie_ajout.php?ajout=banane">1Kg banane</a> (1E)</td>
    <td><?echo $panier["banane"]?>Kg banane(s)</td>
  </tr>
</table>
<p align="center"><a href="cookie_init.php">vider mon panier</a></p>
<p align="center"><a href="cookie_calcul.php">calculer le total</a></p>
</body>
</html>
```

- Ce troisième script (*cookie_calcul.php*) calcule et affiche le prix total du panier

```
<?php
    $panier = $_COOKIE["panier"];
    $total  = 0;
    $total += $panier["pomme"]*1;
    $total += $panier["poire"]*1.5;
    $total += $panier["peche"]*2;
    $total += $panier["banane"]*1;
?>
    <html>
    <head>
    <title>Le total du panier</title>
    </head>
    <body bgcolor="#FFFFFF">
    <p align="center">Le total de votre panier: <?echo $total?> E.</p>
    <p align="center">
        <a href="cookie_ajout.php">Modifier mon panier</a></p>
    <p align="center">
        <a href="cookie_init.php">Vider mon panier</a></p>
    </body>
    </html>
```

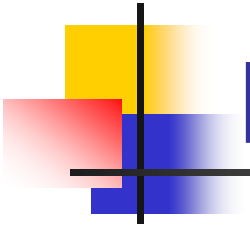


TD

- Exercices 1, 2, 3

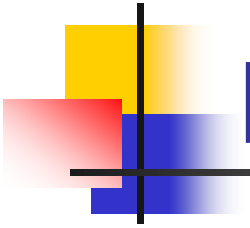
Les sessions

Définition et création



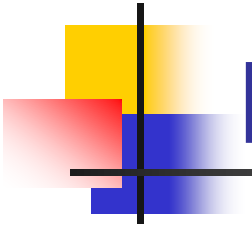
Les sessions

- Idée
  - Aller plus loin que les cookies
    - Permettre à plusieurs pages de partager les mêmes données
    - Conserver plusieurs données lors d'une session
    - Ranger les données de la session sur le serveur
    - Sécuriser le passage des données vers le serveur en
      - donnant un identifiant à la session
      - cryptant l'identifiant
    - Disposer de plusieurs sessions et donner un nom à chacune



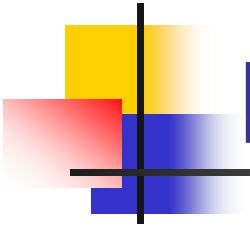
Les sessions

- Qu'est-ce qu'une session ?
  - Une session est donc une période entre un début et une fin d'une activité
  - Par exemple pour Internet,
    - j'ouvre mon navigateur sur http://www.php-astux.info et je referme le navigateur :
      - j'ai ainsi réalisé une session que j'ai commencée, puis terminée
  - Par extension, on associe à une session un ensemble de valeurs définies de manière transparente pour le visiteur et que le serveur conserve de page en page, comme un cookie



Les sessions

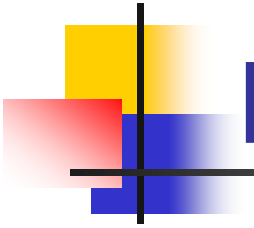
- Qu'est-ce qu'une session ? (suite)
  - Une session est plus ou moins similaire aux cookies HTTP
  - Les données d'une session sont cependant stockées sur le serveur et non chez le client, ce qui l'empêche de pouvoir les modifier manuellement, comme il peut le faire pour un cookie



Les sessions

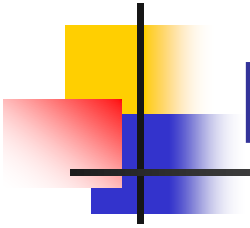
- A quoi peut bien servir une session ?
  - Par exemple
    - Permettre à un visiteur privilégié de pouvoir accéder à des pages particulières de votre site :

      ➔ dans ce cas, l'utilisateur, se connectant avec son login et mot de passe, est reconnu au travers de ces variables sauvegardées dans un cookie, reçoit cette permission
    - Cela permet de réserver des liens à des gens qui ont une inscription à mon site



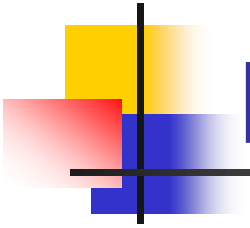
Les sessions

- Autre exemple,
  - Je souhaite que le visiteur remplisse un formulaire mais le nombre de champs et leur diversité (informations personnelles, informations administratives, situation géographique...) ne me permettent pas de tout afficher sur une même page
  - La solution est d'afficher chaque jeu de champs sur une page et d’utiliser les données conservées par le serveur pour faire le lien entre les pages, quand le visiteur passe à la page suivante



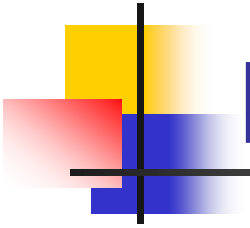
Les sessions

- La durée de vie d'une session
  - Une session est, comme son nom l'indique, une session de navigation
  - Elle commence donc lors de l'accès à une page les utilisant et se termine à la fermeture du navigateur du client
  - Une fois la session terminée, elle est détruite ainsi que toutes les données qu'elle contient
  - Elle ne doit donc contenir que des données temporaires
  - Cependant, la session possède aussi un délai d'expiration
    - Celui-ci peut être modifié dans la configuration du serveur (directive *session.gc_maxlifetime*), mais vaut généralement une trentaine de minutes
    - Une fois ce délai dépassé, la session est supprimée



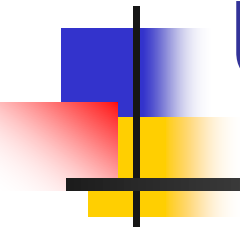
Les sessions

- Comment ça marche ?
  - Lors de l'accès à une page nécessitant une session, PHP va vérifier si une a déjà été ouverte et la réutiliser ou en créer une nouvelle
  - Il va donc lui attribuer un identifiant unique et se débrouiller pour que la prochaine page consultée puisse le connaître
  - Pour cela, il exploite deux fonctionnalités
    - Premièrement, si l'utilisation de cookie est possible (le client ne les a pas désactivés), PHP crée un cookie contenant l'identifiant de session
    - Deuxièmement, si les cookies ne sont pas disponibles, il va réécrire les URL de la page pour inclure cet identifiant dans le lien
  - Dans les deux cas, le script ayant besoin d'accéder aux données de la session recevra l'identifiant et sera capable de charger les données qu'elle contient



Les sessions

- Est-ce que c'est sécurisé ?
  - Une session est toujours plus sécurisée qu'un cookie puisque le client ne peut pas la modifier manuellement
  - Un risque subsiste tout de même si l'identifiant de session peut être découvert
  - Par exemple, les cookies transitent sur le réseau en clair
  - Si quelqu'un est capable d'intercepter les communications entre le client et le serveur, il est capable de voir le cookie et de découvrir l'identifiant de session
  - Il n'aura alors plus qu'à créer un faux cookie et PHP le prendra pour le pauvre internaute s'étant fait voler son cookie
  - Pour éviter cela, on peut utiliser une connexion cryptée ou configurer le serveur de façon à ce qu'il rende la lecture de ce cookie impossible par JavaScript (*Http Only*)



Utiliser les sessions

Utiliser les sessions

- Initialiser une session
  - Pour pouvoir utiliser la fonctionnalité de session de PHP, il faut lancer le moteur de session en utilisant la fonction session_start()
  - Cette fonction doit être en mesure d'envoyer des *entêtes* HTTP
  - Pour cela :
    - ➔ aucune donnée ne doit donc avoir été transmise au navigateur
    - ➔ vous devez placer ce code au tout début de votre script pour éviter que des choses soient transmises avant

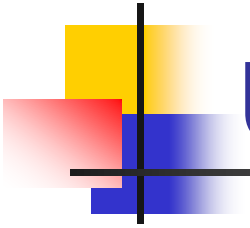
```
<?php
   session_start();
?>
```

Utiliser les sessions

- Une session portant un nom personnalisé
  - Une session porte par défaut le nom "PHPSESSID"
  - C'est lui qui sera utilisé comme nom de cookie ou comme paramètre GET dans les liens
  - Pour le changer, utiliser la fonction session_name()

```
<?php
   session_name('nom_de_session');
   session_start();
?>
```

  - Ce code va donc charger une session avec l'identifiant provenant du cookie ou du paramètre GET portant le nom : nom_de_session



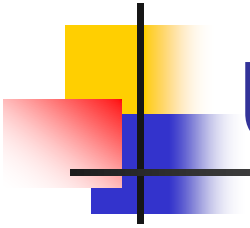
Utiliser les sessions

- Changer manuellement l'identifiant de la session
  - PHP détecte automatiquement l'identifiant à utiliser
    - On peut, par exemple, se baser sur le hachage (md5() ou sha1()) de l'adresse IP du client pour déterminer l'identifiant de la session
    - Attention aux proxys et aux internautes dont l'IP change à chaque requête
  - Cependant, on peut avoir besoin de le changer manuellement, si on veut développer un système alternatif pour passer l'identifiant de session entre les pages
  - Pour définir manuellement l'identifiant de session, on utilise la fonction session_id()

Utiliser les sessions

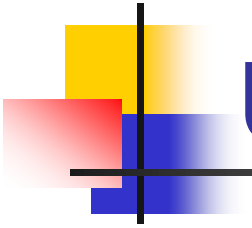
- Exemple

```
<?php
$identifiant = sha1($_SERVER['REMOTE_ADDR']);
    /* Premièrement, nous récupérons l'adresse IP du client, puis
     nous utilisons une fonction de hachage pour générer un code
     valide. En effet, l'identifiant d'une session ne peut contenir que
     des lettres (majuscules ou minuscules) et des chiffres*/
$ancien_identifiant = session_id($identifiant);
    /* Ensuite, nous modifions manuellement l'identifiant de session
     en utilisant session_id() et en lui donnant l'identifiant généré
     plus haut. Comme toujours, la fonction retourne l'ancien
     identifiant, on le place dans une variable, au cas où on aurait
     besoin de le réutiliser. */
session_start();
    /* On démarre la session, PHP va utiliser le nouvel identifiant
     qu'on lui aura fourni*/
?>
```



Utiliser les sessions

- Lire et écrire des données dans la session
  - Les données de la session sont très facilement accessibles au travers d'un simple tableau PHP
  - On utilise le tableau super-global $_SESSION
  - Tout ce qui est stocké à l'intérieur est sauvegardé et accessible depuis toutes les pages PHP utilisant les sessions



Utiliser les sessions

- Exemple : une zone d'administration protégée par mot de passe
  - Dans cet exemple, nous allons fabriquer pas à pas une zone d'administration pour votre site web, protégée par un mot de passe
  - Nous nous occuperons de la partie identification uniquement
  - Dans le chapitre précédent, vous avez également créé le même type de script, mais en utilisant un cookie
  - Nous allons donc adapter le code en utilisant des sessions à la place

Utiliser les sessions

Exemple

- Le formulaire
  - Voici le code du formulaire en HTML, il va afficher une boîte de texte et un bouton "Connexion"

```
<form action="verification.php" method="post">
<input type="password" name="mdp" />
<input type="submit" value="Connexion" />
</form>
```

Utiliser les sessions

Exemple

- La page de vérification
  - Le formulaire ci-dessus pointe vers une page nommée verfication.php
  - Cette page va vérifier que le mot de passe est juste et, si c'est le cas, placer un élément dans le tableau de la session pour que la page suivante puisse vérifier que vous êtes bien autorisé à voir la page
  - Premièrement, nous devons initialiser la session
  - Nous laissons PHP choisir le nom

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

Utiliser les sessions

Exemple

- L'appel à la fonction session_start() a fabriqué le tableau $_SESSION
- Pour l'instant celui-ci est vide
- Il faut maintenant vérifier que le mot de passe fourni est le bon
- Nous créons ensuite une entrée dans le tableau de la session contenant true si le code est le bon
- Nous pouvons alors rediriger le navigateur de l'internaute ou afficher un message d'erreur
- **Principe**

```
//Nouveau mot de passe
$mdp = 'qwertz';
//Est-ce qu'il est égal à l'ancien ?
if($mdp == $_POST['mdp']) {
   $_SESSION['admin'] = true;
   header('Location: admin.php'); // Redirection du navigateur
}
else { echo 'Le mot de passe est erroné'; }
?>
```

Utiliser les sessions

Exemple

- La page d'administration
  - Cette page doit se nommer admin.php
  - Si vous décidez d'utiliser un autre nom, il faudra modifier le script d'identification pour qu'il pointe sur la bonne page
  - Comme dans l'autre page, nous devons commencer par initier une session
  - **Principe**

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

  - Nous avons alors accès au tableau $_SESSION

Utiliser les sessions

Exemple

- Si le visiteur a fourni le bon mot de passe, il existe un élément dans le tableau nommé 'admin' et valant true
- Dans le cas contraire, cet élément n'existe pas
- Il suffit donc de vérifier sa présence pour savoir si le visiteur est réellement l'administrateur du site
- Pour cela, nous utilisons la fonction isset() qui vérifie si la variable (ou l'élément du tableau) existe
- **Principe**

```
<?php
    session_start();
    if(isset($_SESSION['admin']) &&
    $_SESSION['admin']) { // Code à exécuter si le
            visiteur est administrateur }
    else { // Code à exécuter si le visiteur n'est pas
            administrateur
    }
```

Utiliser les sessions

Exemple

- Fermer une session
  - De la même façon que d'autres fonctionnalités de PHP, comme les connexions aux bases de données ou un pointeur de fichier, les sessions n'ont pas besoin d'être fermées
  - Une fois le script PHP terminé, les données de la session sont automatiquement sauvegardée pour le prochain script
  - Cependant, durant toute l'exécution du script, le fichier de la session est verrouillé et aucun autre script ne peut y toucher
  - Ils doivent donc attendre que le premier arrive à la fin

Utiliser les sessions

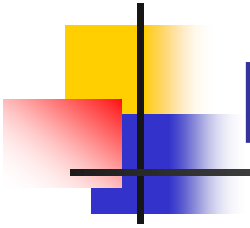
Exemple

- Vous pouvez fermer plus rapidement une session en utilisant la fonction session_write_close()
- Si vous voulez également détruire la session, vous pouvez utiliser la fonction session_destroy() couplée à la fonction session_unset()

Les sessions

- Sauvegarder une variable
  - Les variables de sessions sont stockées dans le tableau super-global : *$_SESSION*

```
<?php
    session_start(); // Création de la session
    $_SESSION['prenom'] = 'Jean-Pierre'; // Sauvegarde dans la session créée de la variable "prenom"
?>
```



Les sessions

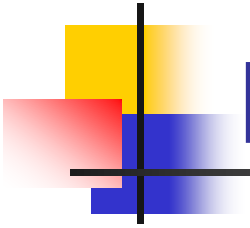
- Récupération de données dans une session
  - Quand on démarre une session avec *session_start()*, le tableau super-global *$_SESSION* est automatiquement initialisé avec les variables de la session
  - S'il contenait quelque chose, ce contenu n'est plus disponible après

```
<?php
   echo $_SESSION['prenom'];
?>
```

Les sessions

- Savoir si une variable appartient à une session
  - Utiliser sur le tableau *$_SESSION* la fonction *isset()* qui renvoie vrai si la variable passée en argument existe réellement
  - Exemple

```
<?php
    if (isset($_SESSION['prenom'])) {
    echo 'La variable prenom est déjà enregistrée !';
    // On est certain de pouvoir y accéder ici
    } else {
            echo 'La variable prenom n\'est pas enregistrée !';
    }
?>
```



Les sessions

- Exemple 2 : réaliser un formulaire de connexion
  - Le formulaire permet, par une page, de s'identifier
  - L'accès à toutes les autres pages sera tributaire de cette identification
  - Voici les étapes à suivre :
    - Initialisez votre session
    - Initialisez vos variables
    - Affichez votre formulaire, ou effectuez son son traitement :
      - dans un cas, mes variables de session seront vides,
      - dans l'autre, elles ne seront remplies que si le traitement retourne un résultat correct
    - Enfin, si le formulaire de connexion est validé, affichez un lien pour passer à la page suivante en "mode connecté"
    - Autrement, pas d'accès aux autres pages

- index_connexion.php

```
<?php
 session_start();
 $_SESSION['login'] = '';
 $_SESSION['password'] = '';

 if (isset($_POST['submit'])){
   // bouton submit pressé, je traite le formulaire
   $login = (isset($_POST['login'])) ? $_POST['login'] : '';
   $pass  = (isset($_POST['pass']))  ? $_POST['pass']  : '';

   if (($login == "Matthieu") && ($pass == "NewsletTux")){
     $_SESSION['login'] = "Matthieu";
     $_SESSION['password'] = "NewsletTux";
     echo '<a href="accueil.php" title="Accueil de la section
    membre">Accueil</a>';
   }
   else{
     // une erreur de saisie ...?
     echo '<p style="color:#FF0000; font-weight:bold;">Erreur de
    connexion.</p>';
   }
 }; // fin if (isset($_POST['submit']))
```

- suite

```
if (!isset($_POST['submit'])))
{
  // Si bouton submit non pressé, alors j'affiche le
  formulaire
  echo '<form id="conn" method="post" action="">'."\n";
  echo '  <p><label for="login">Login :</label><input
  type="text" id="login" name="login" /></p>'."\n";
  echo '  <p><label for="pass">Mot de Passe
  :</label><input type="password" id="pass" name="pass"
  /></p>'."\n";
  echo '  <p><input type="submit" id="submit"
  name="submit" value="Connexion" /></p>'."\n";
  echo '</form>'."\n";
 }; // fin if (!isset($_POST['submit'])))
?>
```

Explications

- J'ai choisi d'effectuer le traitement du formulaire avant son affichage pour pouvoir afficher (en gras, rouge dans cet exemple) un message d'erreur puis réafficher le formulaire ensuite
- Ainsi lorsque le lien Accueil apparaît, je sais :
  - que la personne est correctement identifiée
  - que mes variables de session sont non vides
  - Autrement, mes variables restent vides
- Mais qu'en est-il de la page accueil.php ? Si un visiteur mal intentionné rentre son adresse directement dans le navigateur pour contourner le formulaire ?

- Heureusement, il y a moyen de se protéger. Voici le code de accueil.php :

```
<?php session_start();
   if ((!isset($_SESSION['login'])) ||
            (empty($_SESSION['login']))) {
// la variable 'login' de session est non déclarée ou vide
echo ' <p>Petit curieux... <a
   href="index_connexion.php"
   title"Connexion">Connexion d'abord !</a></p>'."\n";
exit();
}
?>
[...] suite du code : contenu réel de la page.
```

- Dans ce code :
  - J'initialise la session (session récupérée si déjà existante)
  - Je teste si une des variables de session (dans mon exemple, login) est définie et remplie ... Si non définie ou non remplie, dehors :-) (*Notez la présence de **exit();** qui coupe net l'exécution de PHP, ainsi la suite du code source de accueil.php ne sera pas exécutée.*)
  - Je mets ce test tout en haut de page, pour que ça soit le premier élément vérifié par le serveur
  - Si je mets ces quelques lignes dans toutes les pages que je veux protéger, alors ma "section membre" est presque finie ...

Les sessions

- Supprimer une variable d'une session
  - Utiliser *unset()* qui prend en paramètre la variable à détruire
  - Exemple

```
<?php
   unset($_SESSION['prenom']); // La variable de
   //session "prenom" a été supprimée, on ne peut plus
   //y avoir accès !
?>
```

Les sessions

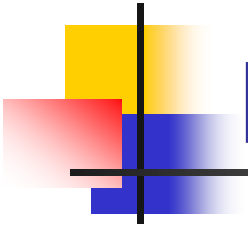
- Détruire une session
  - Utiliser *session_destroy()* qui ne prend aucun paramètre et qui renvoie vrai en cas de succès et faux en cas d'erreur
  - Fonctionnement :

```
<?php
   if (session_destroy()) {
            echo 'Session détruite !';
   } else {
   echo 'Erreur : impossible de détruire la session !';
}
?>
```

Les sessions

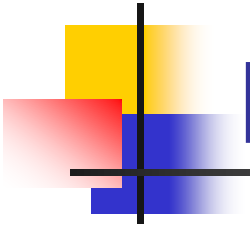
- Détruire toutes les variables d'une session
  - Il est aussi possible de détruire toutes les variables de session, ce qui permet de conserver votre session :
    - il suffit tout simplement de réinitialiser le tableau *$_SESSION*
  - *Il ne faut jamais utiliser unset() directement sur $_SESSION, cela rendrait impossible l'accès aux variables de la session courante jusqu'à sa fermeture*

```
<?php
$_SESSION = array(); // $_SESSION est désormais
   //un tableau vide, toutes les variables de session ont
   //été supprimées
?
```



Les sessions

- Regénérer les identifiants de session
  - Il est possible de demander à PHP un nouvel identifiant de session grâce à la fonction *session_regenerate_id()*
    - Les données ne sont alors pas effacées de la session en cours
    - L'identifiant de session est simplement modifié
    - L'ancien devenant invalide
  - Cela est utile lorsque, par exemple, l'utilisateur change de niveau d'accès, s'il a accès à de nouvelles possibilités ou au contraire n'a plus accès à d'anciennes pages
  - De cette manière, si quelqu'un avait réussi à trouver l'ancien identifiant de session, il n'aura pas accès aux nouveaux privilèges de l'utilisateur
  - À l'inverse, si les identifiants de sessions sont regénérés trop souvent, cela peut poser des problèmes au niveau du référencement des pages web, surtout dans les cas où ils sont passés via l'URL



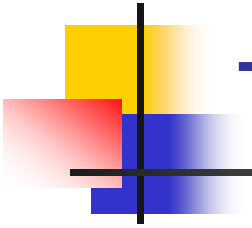
Les sessions

- Utiliser plusieurs sessions dans la même page
  - Il est impossible d'ouvrir simultanément plusieurs sessions
  - Cependant, on peut tout à fait ouvrir plusieurs sessions l'une après l'autre
  - Dans ce cas, il faut
    - fermer la première session sans la détruire, grâce à *session_write_close()*
    - puis assigner les nouveaux session_name et session_id
    - et enfin ouvrir la nouvelle session avec *session_start()*

Les sessions

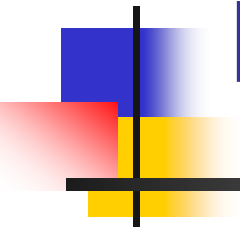
```php
<?php
    session_name('utilisateur');
    session_start(); // Création de la première session
    [...]
    // Utilisation de la première session
    session_write_close(); // Fermeture de la première session, ses données sont sauvegardées.
    session_name('admin'); // Indication du nom de la seconde session
    session_start(); // Ouverture de la seconde session
    [...] // Utilisation de la seconde session.
?>
```

- *Une fois la session fermée, il est toujours possible d'accéder en lecture (les modifications ne seront pas prises en compte) aux variables de l'ancienne session*
- *$_SESSION ne sera vidé et rerempli qu'au prochain appel à session_start()*



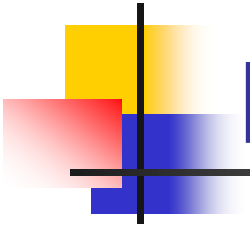
TD

- Exercices 4,5



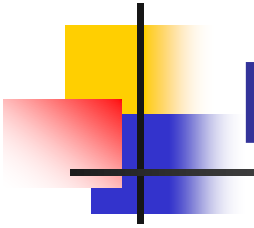
Les sessions

Configurer php.ini



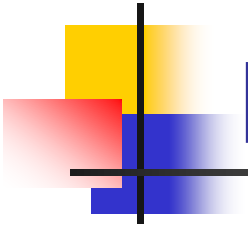
Les sessions

- Transmission de l'identifiant de session
  - Comme vu au début, un identifiant unique identifie chaque session
  - PHP se charge lui-même de transmettre cet identifiant d'une page à l'autre, mais on peut indiquer explicitement comment le faire à l'aide de 4 directives de configurations :
    - session.use_cookies :
      - Cette option fait stocker l'identifiant dans un cookie si elle est égale à 1
      - Sa valeur par défaut est 1
      - Si l'option est à 0, et qu'un coockie de session est présent dans le navigateur, il sera simplement ignoré
    - session.use_only_cookies :
      - Si cette option est égale à 1, alors PHP ignorera les identifiants transmis via l'url pour n'utiliser que ceux contenus dans les cookies
      - Sa valeur par défaut est 0



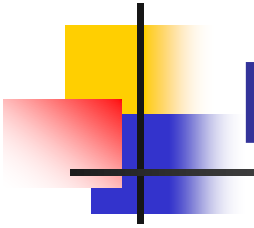
Les sessions

- **session.use_trans_sid** :
  - Si cette option est égale à 1, alors PHP insèrera automatiquement l'identifiant dans tous les liens non absolus (ne commençant pas par http:// ou ftp:// ou mailto: ou d'autres liens absolus
  - les liens commençant par un / (relatifs à la racine du site) seront eux pourvus de l'identifiant)
  - Sa valeur par défaut est 0
- **session.url_rewriter_tags** :
  - Indique où et dans quelles balises HTML insérer l'identifiant de session dans le cas où **session.use_trans_sid** est à 1
  - Sa valeur par défaut est : a=href,area=href,frame=src,input=src,form=fakeentry,fieldset=
  - Le format est balise1=attribut1,balise2=attribut2,...
  - Le cas des balises form et fieldset est particulier, car cela se traduit en fait par l'apparition d'un champ input (de type hidden) supplémentaire



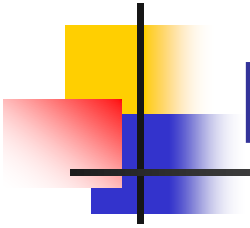
Les sessions

- Voici deux exemples de configuration pour ces options :
  - Le premier est celui qui présente le plus de chance de faire transiter l'identifiant :
    - session.use_cookies à 1
    - session.use_only_cookies à 0
    - session.use_trans_sid à 1
    - session.url_rewriter_tags à a=href,area=href,frame=src,iframe=src,input=src,form=fakeentry,fieldset=
  - Si vous souhaitez produire une page respectant les standards XHTML, nous vous conseillons plutôt la valeur :
    - a=href,area=href,frame=src,iframe=src,input=src,fieldset=
    - et d'utiliser une balise *fieldset* dans vos balises *form*



Les sessions

- Le second est celui qui est totalement invisible à l'oeil d'un visiteur classique mais, si les cookies sont désactivés, il ne fonctionnera pas
  - session.use_cookies à 1
  - session.use_only_cookies à 1
  - session.use_trans_sid à 0
  - session.url_rewriter_tags à vide

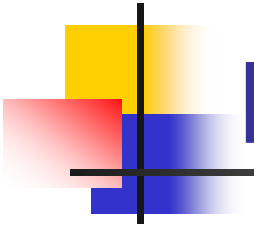


Les sessions

- La sécurité des sessions
  - Il existe plusieurs directives de configuration permettant de jouer sur la sécurité des sessions, en voici quelques unes :
    - Vérifier la provenance du visiteur
  - *session.referer_check* permet de spécifier une chaîne de caractères qui doit être présente dans le "referer" (adresse de la page de provenance) si celui-ci est indiqué par le navigateur
  - Une bonne idée est d'indiquer le nom de domaine de votre site (par exemple)
  - Par contre, gardez à l'idée que cela ne permet pas d'être sûr que le visiteur provient d'une des pages de votre site
  - En effet, le referrer provient du protocole HTTP, lequel est très facile à manipuler !

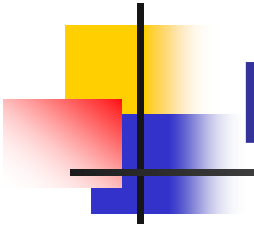
Les sessions

- **Augmenter la complexité de l'identifiant de session**
  - *session.hash_function*
    - permet de choisir entre deux algorithmes de création des identifiants de sessions : MD5 (mettre la valeur 0) ou SHA-1 (mettre la valeur 1), le second permet de générer des identifiants de sessions plus longs
  - *session.hash_bits_per_caracter*
    - permet de définir les plages de caractères utilisées pour l'identifiant de session 4 pour les caractères '0' à '9' et 'a' à 'f', 5 pour '0'-'9' et 'a'-'v' et 6 pour '0'-'9', 'a'-z', 'A'-'Z', '-' et ','



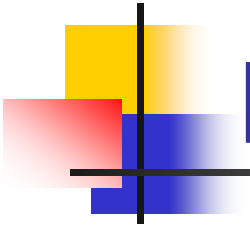
Les sessions

- Il s'agit uniquement d'une représentation de l'identifiant, en aucun cas ça ne modifie sa complexité intrinsèque
- *Ces directives de configuration ne sont disponibles que depuis PHP 5*



Les sessions

- Modifier la façon dont sont formatées les données de sessions
  - *session.serialize_handler*
    - permet de modifier la façon dont sont formatées les données de sessions avant d'être sauvegardées
  - Par défaut, (valeur *php*) c'est une sérialization proche de la fonction serialize qui est utilisée
  - Si votre installation de PHP le supporte, vous pouvez utiliser WDDX (valeur *WDDX*), qui permet de récupérer un format XML

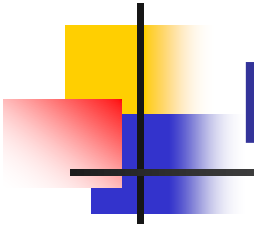


Les sessions

- La durée de vie des sessions
  - Pour modifier la durée de vie des sessions, en plus de jouer sur la durée de vie du cookie et du cache, on peut également jouer sur le temps au-delà duquel PHP considèrera les données stockées comme obsolètes
  - **Pour cela, il existe trois directives de configuration :**
    - *session.gc_maxlifetime* : Il s'agit effectivement de la durée au-delà de laquelle des données de session seront considérées comme périmées.
    - *session.gc_probability* : La vérification du temps de vie des données n'est pas systématique, c'est lancé aléatoirement selon une fréquence prédéfinie. Il s'agit de la probabilité de déclenchement de l'opération.

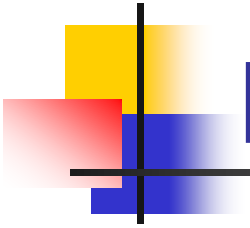
Les sessions

- *session.gc_divisor* : Pour obtenir la fréquence de lancement de la procédure de vérification des données, il faut diviser la probabilité par ce paramètre
- Par exemple, si gc_probability vaut 1 et que gc_divisor vaut 100, alors ce sera lancé 1 fois toutes les 100 ouvertures de sessions



Les sessions

- Modifier localement la configuration des sessions
  - Comme pour toutes les directives de configuration de PHP, il est possible de modifier la configuration des sessions sur certains serveurs
  - On utilise pour cela le fichier *.htaccess*
  - Pour modifier la valeur d'une directive de configuration, il suffit d'indiquer *php_value* ou *php_flag* (dans le cas d'une directive de type on/off), suivi du nom de la directive de configuration et de sa nouvelle valeur (en séparant le tout par des espaces)
  - Cependant, gardez à l'esprit que cela dépend entièrement de la configuration du serveur, il est même possible que seule une partie des directives soit modifiable, voire même aucune...



Les sessions

- Modifier la façon dont sont stockées les données de sessions
  - Comme indiqué plus haut, PHP ne fournit pas de gestionnaire de données de sessions autre que des fichiers non cryptés
  - Il est cependant possible de définir vous-mêmes les opérations à faire pour ouvrir, fermer, lire, écrire et vérifier la durée de vie d'une session
  - Cette fonctionnalité vous permettra par exemple de crypter et de stocker très facilement vos sessions dans une base de données (par exemple) en écrivant les fonctions de requêtage adéquates
  - Une fois ceci fait, vous n'aurez alors plus besoin de définir les opérations d'accès à la base de données pour les sessions !
  - Vous trouverez un peu plus bas un exemple de fonctions vous permettant de stocker les sessions dans une base de données