Cookies

PHP supporte les cookies HTTP de manière transparente. Les cookies sont un mécanisme d'enregistrement d'informations sur le client, et de lecture de ces informations. Ce système permet d'identifier et de suivre les visiteurs. Vous pouvez envoyer un cookie avec la fonction setcookie() ou setrawcookie(). Les cookies font partie des en-têtes HTTP, ce qui impose que setcookie() soit appelée avant tout affichage de texte. Ce sont les mêmes limitations que pour header().

Généralement, on utilise les cookies pour faciliter la vie des utilisateurs en préenregistrant des données les concernant comme un nom d'utilisateur par exemple.
Ainsi, dès qu'un utilisateur connu demande à accéder à une page de notre site, les cookies vont également automatiquement être envoyées dans la requête de l'utilisateur. Cela va nous permettre de l'identifier et de lui proposer une page personnalisée.

Les cookies sont des variables enregistrées pour une durée déterminée, définie par le développeur. Elles sont donc plus appréciées pour la gestion des comptes utilisateurs, afin de ne pas avoir besoin de se connecter au site internet à chaque visite.

Enfin, il faut savoir que les cookies sont stockés sur l'ordinateur de l'utilisateur et non pas sur le serveur du site internet. Les navigateurs ont une petite base de données interne et cachée où sont stockées les informations de navigation comme l'historique, les saisies de formulaires, les mots de passe, les sessions et les cookies.

Avec PHP, on peut à la fois créer et récupérer des valeurs de cookies.

Créer des cookies avec PHP

Un cookie est créé avec la fonction setcookie().

Syntaxe

setcookie(*name, value, expire, path, domain, secure, httponly*);

Seul le paramètre name est requis. Tous les autres paramètres sont facultatifs.

Paramètre	Signification
name	Le nom du cookie. Le nom d'un cookie est soumis aux mêmes règles que les noms des variables.
value	La valeur du cookie. Comme cette valeur est stockée sur l'ordinateur d'un utilisateur, on évitera de stocker des informations sensibles.
expires	La date d'expiration du cookie sous forme d'un timestamp UNIX (nombre de secondes écoulées depuis le 1er janvier 1970). Si aucune valeur n'est passée en argument, le cookie expirera à la fin de la session (lorsque le navigateur sera fermé).
path	Le chemin sur le serveur sur lequel le cookie sera disponible. Si la valeur est '/', le cookie sera disponible sur l'ensemble du domaine. Si la valeur est '/cours/', le cookie ne sera disponible que dans le répertoire (le dossier) /cours/ (et dans tous les sous-répertoires qu'il contient).
domain	Indique le domaine ou le sous domaine pour lequel le cookie est disponible.

secure	Indique si le cookie doit uniquement être transmis à travers une connexion sécurisée HTTPS depuis le client. Si la valeur passée est `true`, le cookie ne sera envoyé que si la connexion est sécurisée.
httponly	Indique si le cookie ne doit être accessible que par le protocole HTTP. Pour que le cookie ne soit accessible que par le protocole http, on indiquera la valeur `true`. Cela permet d'interdire l'accès au cookie aux langages de scripts comme le JavaScript par exemple, pour se protéger potentiellement d'une attaque de type XSS.

```
    string $name,
    string $value = "",
    int $expires_or_options = 0,
    string $path = "",
    string $domain = "",
    bool $secure = false,
    bool $httponly = false
```

PHP Créer/Récupérer un cookie

L'exemple suivant crée un cookie nommé « user » avec la valeur « Charles Carnus ». Le cookie expirera après 30 jours (86400 * 30). Le "/" signifie que le cookie est disponible sur l'ensemble du site (sinon, sélectionnez le répertoire que vous préférez). On récupère ensuite la valeur du cookie "user" (à l'aide de la variable globale $_COOKIE). On utilise également la fonction isset() pour savoir si le cookie est défini :

```
<?php
$cookie_name = "user";
$cookie_value = "Charles Carnus";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30),
"/"); // 86400 = 1 jour
?>

<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
     echo "Le cookie nommé '" . $cookie_name . "' n'est pas
défini!";
} else {
     echo "Le cookie '" . $cookie_name . "' est défini!<br>";
     echo "Sa valeur est : " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

Remarque : La fonction setcookie() doit apparaître AVANT la balise <html>.
Pour information, cette restriction provient du protocole HTTP et non pas de PHP.

```php
<?php
    setcookie('user_id', '1234');
    setcookie('user_pref', 'thème sombre', time()+3600*24, '/',
'', true, true);
?>

<!DOCTYPE html>
<html>
    <head>
        <title>Cours PHP </title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>
        <h1>Cours PHP et BDD</h1>
        <?php
        ?>
        <p>Texte</p>
    </body>
</html>
```

On a créé deux cookies avec `setcookie()` : un premier cookie nommé `user_id` qui stockera l'ID d'un visiteur pour la session actuelle par exemple ce qui permettra de l'identifier pendant sa navigation sur le site et un deuxième cookie qu'on appelle `user_pref` qui stockera les préférences mentionnés par l'utilisateur pour le site (utilisation d'un thème sombre par exemple).

On a précisé les valeurs `true` pour les arguments « secure » (passage par une connexion sécurisée pour transmettre le cookie) et « httponly » (obligation d'utiliser le protocole HTTP pour accéder au cookie).

Récupérer la valeur d'un cookie

Pour récupérer la valeur d'un cookie, nous allons utiliser la variable `$_COOKIE`.

```php
<?php
    setcookie('user_id', '1234');
    setcookie('user_pref', 'thème sombre', time()+3600*24, '/',
'', true, true);
?>

<!DOCTYPE html>
<html>
    <head>
        <title>Cours PHP </title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>
```

```
    <body>
        <h1>Cours PHP et BDD</h1>
        <?php
            if(isset($_COOKIE['user_id'])){
                echo 'Votre ID de session est ' .
$_COOKIE['user_id'];
            }
        ?>
        <p>Texte</p>
    </body>
</html>
```

On commence par vérifier qu'un cookie `user_id` existe et a bien été défini et stocké dans `$_COOKIE` avec la fonction `isset()`. Si c'est le cas, on affiche (`echo`) la valeur du cookie.

Il faut bien noter que la variable `$_COOKIE` stocke la liste des cookies renvoyés par le navigateur. Lorsqu'un utilisateur demande à accéder à une page pour la première fois, le cookie `user_id` est créé côté serveur et est renvoyé au navigateur afin qu'il soit stocké sur la machine du visiteur.

Ainsi, la première fois qu'un utilisateur demande une page, la variable `$_COOKIE` ne stocke pas encore le cookie puisque celui-ci n'a pas encore été créé et donc le navigateur du visiteur ne peut rien renvoyer. Le test de la condition `if` échoue donc lors du premier affichage de la page.

Si on actualise ensuite la page, en revanche, le navigateur renvoie bien cette fois-ci la valeur du cookie et son nom et celui-ci est bien stocké dans `$_COOKIE`. Le test de lae condition va alors être vérifié et on va pouvoir afficher (`echo`) la valeur de ce cookie.

Modifier une valeur de cookie

Pour modifier un cookie, il suffit de paramétrer (à nouveau) le cookie à l'aide de la fonction setcookie() :

```
<?php
$cookie_name = "user";
$cookie_value = "Carnus";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30),
"/");
?>
<html>
<body>
<?php
if(!isset($_COOKIE[$cookie_name])) {
     echo "Le cookie nommé '" . $cookie_name . "' n'est pas
défini!";
} else {
     echo "Le cookie '" . $cookie_name . "' est défini!<br>";
     echo "Sa Valeur est : " . $_COOKIE[$cookie_name];
}
?>
</body>
</html>
```

Supprimer un cookie

Pour supprimer un cookie, utilisez la fonction setcookie() avec une date d'expiration dans le passé :

```php
<?php
// Fixer la date d'expiration à une date révolue
setcookie("user", "", time() - 3600);
?>
<html>
<body>
<?php
echo "Le cookie 'user' a été supprimé.";
?>

</body>
</html>
```

```php
<?php
    //On définit deux cookies
    setcookie('user_id', '1234');
    setcookie('user_pref', 'thème sombre', time()+3600*24, '/',
'', false, false);
    //On modifie la valeur du cookie user_id
    setcookie('user_id', '5678');
    //On supprime le cookie user_pref
    setcookie('user_pref', '', time()-3600, '/', '', false,
false);
?>

<!DOCTYPE html>
<html>
    <head>
        <title>Cours PHP </title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>
    <body>
        <h1>Cours PHP et BDD</h1>
        <?php
            if(isset($_COOKIE['user_id'])){
                echo 'Votre ID de session est le ' .
$_COOKIE['user_id']. '<br>';
            }
            if(isset($_COOKIE['user_pref'])){
                echo 'Votre thème préféré est ' .
$_COOKIE['user_pref'];
            }else{
                echo 'Pas de thème préféré défini';
            }
```

```
        ?>

        <p>Texte</p>
    </body>
</html>
```

On commence par définir deux cookies `user_id` et `user_pref`. On modifie ensuite la valeur de notre cookie `user_id` et on passe une date d'expiration passée à notre cookie `user_pref` pour le supprimer.

Vérifiez si les cookies sont activés

L'exemple suivant crée un petit script qui vérifie si les cookies sont activés. Tout d'abord, essayez de créer un cookie de test avec la fonction setcookie(), puis comptez la variable du tableau $_COOKIE :

```
<?php
setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html>
<body>

<?php
if(count($_COOKIE) > 0) {
     echo "Les cookies sont activés.";
} else {
     echo "Les cookies sont désactivés.";
}
?>

</body>
</html>
```

Codes Cookies

Exemple 1 :

exp.php

```php
<?php
// on définit une durée de vie de notre cookie (en secondes), donc un jour dans notre cas
$temps = 1*24*3600;
// on envoie un cookie de nom formation portant la valeur SNIR
setcookie ("formation", "SNIR", time() + $temps);
?>
```

index.php

```php
<html>
<head>
<title>Cours PHP et BDD</title>
<body>

    <?php
    // On teste la déclaration de notre cookie
    if (isset($_COOKIE['formation'])) {
     echo 'Votre spécialité est : '.$_COOKIE['formation'].' !';
    }
    else {
     echo 'Le cookie n\'est pas déclaré.';

     // si le cookie n'existe pas, on affiche un formulaire permettant au visiteur de saisir sa formation
     echo '<form action="./traitement.php" method="post">';
     echo 'Votre spécialité : <input type = "texte" name = "nom"><br />';
     echo '<input type = "submit" value = "Envoyer">';
    }
    ?>

</body>
</html>
```

traitement.php

```php
<?php
    If (isset($_POST['nom'])) {
     // 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 formation portant la valeur de
la variable $nom, c'est-à-dire la valeur qu'a saisi la personne
qui a rempli le formulaire
      setcookie ("formation", $_POST['nom'], time() + $temps);

      // fonction 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.';
     }
?>
```

Exemple 2 :

cookie_init.php

```
<?php
      setCookie("panier[Portable]", 0);
      setCookie("panier[Chargeur]", 0);
      setCookie("panier[Disque]", 0);
      setCookie("panier[Câble]", 0);
      header("Location: cookie_ajout.php");
?>
```

cookie_ajout.php

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

      switch ($_GET["ajout"])
      {
            case "Portable":
```

```
                $panier["Portable"]++;
                setCookie("panier[Portable]",$panier["Portable"]);
            break;
            case "Chargeur":
                $panier["Chargeur"]++;
                setCookie("panier[Chargeur]",$panier["Chargeur"]);
            break;
            case "Disque":
                $panier["Disque"]++;
                setCookie("panier[Disque]",$panier["Disque"]);
            break;
            case "Câble":
                $panier["Câble"]++;
                setCookie("panier[Câble]",$panier["Câble"]);
            break;
        }
?>
<html>
<head>
  <meta charset="utf-8">
  <title>Panier</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=Portable">1 portable</a>
(500E)</td>
    <td><?echo $panier["Portable"]?>Qt Portable(s)</td>
  </tr>
  <tr align="center">
    <td><a href="cookie_ajout.php?ajout=Chargeur">1 chargeur</a>
(25E)</td>
    <td><?echo $panier["Chargeur"]?>Qt Chargeur(s)</td>
  </tr>
  <tr align="center">
    <td><a href="cookie_ajout.php?ajout=Disque">1 disque</a>
(100E)</td>
    <td><?echo $panier["Disque"]?>Qt Disque(s)</td>
  </tr>
  <tr align="center">
    <td><a href="cookie_ajout.php?ajout=Câble">1 c&acirc;ble</a>
(10E)</td>
    <td><?echo $panier["Câble"]?>Qt C&acirc;ble(s)</td>
  </tr>
</table>
<p align="center"><a href="cookie_init.php">Vider le panier</a></
p>
```

```
<p align="center"><a href="cookie_calcul.php">Calculer le total</
a></p>
</body>
</html>
```

cookie_calcul.php

```
<?php
    $panier = $_COOKIE["panier"];

    $total  = 0;
    $total += $panier["Portable"]*1;
    $total += $panier["Chargeur"]*1;
    $total += $panier["Disque"]*2;
    $total += $panier["Câble"]*3;
?>

<html>
<head>
     <title>Le total du panier</title>
</head>
<body bgcolor="#FFFFFF">
     <p align="center">Le total de votre panier : <?echo $total ?>
&#8364;.</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>
```

Sessions PHP

Une session est un moyen de stocker des informations (dans des variables) à utiliser sur plusieurs pages. Contrairement à un cookie, les informations ne sont pas stockées sur l'ordinateur de l'utilisateur.

Les variables de sessions ne sont disponibles seulement le temps de visite de l'utilisateur. Elles sont normalement supprimées par le navigateur quand ce dernier est fermé. Mais cela dépend des configurations de l'utilisateur.
Les sessions sont un mécanisme qui permet de stocker temporairement des informations sur les utilisateurs de votre site Web, telles que des identifiants de connexion, des préférences d'utilisateur et des données de panier d'achat.

Qu'est-ce qu'une session PHP ?

Lorsque vous travaillez avec une application, vous l'ouvrez, effectuez quelques modifications, puis vous la fermez. Cela ressemble beaucoup à une session. L'ordinateur sait qui vous êtes. Il sait quand vous démarrez l'application et quand vous la terminez. Mais sur Internet, il y a un problème : le serveur Web ne sait pas qui vous êtes ni ce que vous faites, car l'adresse HTTP ne conserve pas son état.
Les variables de session résolvent ce problème en stockant les informations utilisateur à utiliser sur plusieurs pages (par exemple, nom d'utilisateur, couleur préférée, etc.). Par défaut, les variables de session durent jusqu'à ce que l'utilisateur ferme le navigateur.
Donc; Les variables de session contiennent des informations sur un seul utilisateur et sont disponibles pour toutes les pages d'une seule application.

Une session en PHP est une façon de stocker des données temporaires sur le serveur Web pour une utilisation ultérieure. Lorsqu'un utilisateur accède à un site Web, un identifiant unique appelé session ID est généré et stocké sur le serveur. Ce ID est ensuite renvoyé au navigateur de l'utilisateur sous forme de cookie ou ajouté à l'URL.

Les données de session peuvent être stockées sur le serveur dans un fichier ou dans une base de données. Les données sont stockées dans un tableau associatif, où chaque paire clé-valeur représente une donnée spécifique. Les données stockées dans une session peuvent être utilisées pour suivre l'activité de l'utilisateur sur le site Web, stocker des préférences utilisateur ou même stocker des données de panier d'achat.

Lorsqu'un utilisateur accède à un site Web, PHP crée une nouvelle session ou récupère une session existante à partir du serveur. PHP utilise le ID de session pour identifier la session et récupère les données stockées dans la session.

Lorsque l'utilisateur quitte le site Web ou ferme son navigateur, la session est détruite et toutes les données stockées sont supprimées du serveur. Cela permet d'éviter l'accumulation de données inutiles sur le serveur et de protéger la vie privée des utilisateurs.

Lorsqu'une session est démarrée, c'est-à-dire lorsqu'un utilisateur qui ne possède pas encore d'identifiant de session demande à accéder à une page contenant `session_start()`, cette fonction va générer un identifiant de session unique qui va généralement être envoyé au navigateur sous forme de cookie sous le nom `PHPSESSID`.

Pour être tout à fait précis, le PHP supporte deux méthodes pour garder la trace des sessions : via des cookies ou via l'URL. Si les cookies sont activés, le PHP va préférer leur utilisation. C'est le comportement recommandé. Dans le cas contraire, les informations de session vont être passées via l'URL.

Démarrer une session PHP

Une session est démarrée avec la fonction session_start().
Cette fonction va se charger de vérifier si une session a déjà été démarrée en recherchant la présence d'un identifiant de session et, si ce n'est pas le cas, va démarrer une nouvelle session et générer un identifiant de session unique pour un utilisateur.
Par ailleurs, notez qu'il va falloir appeler `session_start()` dans chaque page où on souhaite pouvoir accéder aux variables de session.
Les variables de session sont définies avec la variable globale PHP : $_SESSION.
Maintenant, créons une nouvelle page appelée "demo_session1.php". Dans cette page, nous démarrons une nouvelle session PHP et définissons quelques variables de session :

```
<?php
// Démarrer la session
session_start();
?>

<!DOCTYPE html>
<html>
<body>

<?php
// Stocker des données dans la session
$_SESSION["couleur"] = "vert";
$_SESSION["animal"] = "chat";
echo "Les variables de session sont définies.";
?>

</body>
</html>
```

Remarque : La fonction session_start() doit être la toute première chose dans votre document. Avant toute balise HTML.

Autre exemple :

```
<?php
    //On démarre une nouvelle session
    session_start();

    /*On utilise session_id() pour récupérer l'id de session s'il
existe.
     *Si l'id de session n'existe  pas, session_id() renvoie une
chaine
```

```
     *de caractères vide*/
    $id_session = session_id();
?>

<!DOCTYPE html>
<html>
    <head>
        <title>Cours PHP et BDD </title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>
        <h1>BTS SNIR</h1>
        <?php
            if($id_session){
                echo 'ID de session (récupéré via session_id()) :
<br>'
                .$id_session. '<br>';
            }
            echo '<br><br>';
            if(isset($_COOKIE['PHPSESSID'])){
                echo 'ID de session (récupéré via $_COOKIE) :
<br>'
                .$_COOKIE['PHPSESSID'];
            }
        ?>

        <p>Texte</p>
    </body>
</html>
```

Obtenir les valeurs des variables de session PHP

Ensuite, nous créons une autre page appelée "demo_session2.php". Depuis cette page, nous accéderons aux informations de session que nous avons définies sur la première page ("demo_session1.php").
Notez que les variables de session ne sont pas transmises individuellement à chaque nouvelle page, mais sont récupérées de la session que nous ouvrons au début de chaque page (session_start()).
Notez également que toutes les valeurs des variables de session sont stockées dans la variable globale $_SESSION :

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Echo variables de session définies sur la page précédente
```

```
echo "La couleur est : " . $_SESSION["couleur"] . ".<br>";
echo "L'animal est : " . $_SESSION["animal"] . ".";
?>

</body>
</html>
```

Une autre façon d'afficher toutes les valeurs des variables de session pour une session utilisateur consiste à exécuter le code suivant :

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
print_r($_SESSION);
?>

</body>
</html>
```

Comment ça marche? Comment sait-il que c'est moi ?

La plupart des sessions définissent une clé utilisateur sur l'ordinateur de l'utilisateur qui ressemble à ceci : 765487cf34ert8dede5a562e4f3a7e12. Ensuite, lorsqu'une session est ouverte sur une autre page, il analyse l'ordinateur à la recherche d'une clé utilisateur. S'il y a une correspondance, il accède à cette session, sinon, il démarre une nouvelle session.

Modifier une variable de session PHP

Pour modifier une variable de session, écrasez-la simplement :

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// pour modifier une variable de session, il suffit de l'écraser
$_SESSION["couleur"] = "rouge";
print_r($_SESSION);
?>

</body>
</html>
```

Détruire une session PHP

Pour supprimer toutes les variables globales de session et détruire la session, utilisez session_unset() et session_destroy() :

```php
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// supprimer toutes les variables de session
session_unset();

// détruire la session
session_destroy();
?>

</body>
</html>
```

Codes LogIn (Sessions)

Index.html

```
<html>
<head>
  <title>Formulaire d'identification</title>
</head>

<body>
  <form action="login.php" method="post">
    Votre login : <input type="text" name="login">
    <br />
    Votre mot de passe : <input type="password" name="pwd"><br />
    <input type="submit" value="Connexion">
  </form>

</body>
</html>
```

login.php

```
<?php
// On définit un login et un mot de passe de base pour tester
l'exemple. Cependant, on peut très bien interroger une base de
données afin de savoir si le visiteur qui se connecte est bien
membre du site
$login_valide = "snir";
$pwd_valide = "rodez";

// on teste si les variables sont définies
if (isset($_POST['login']) && isset($_POST['pwd'])) {

 // on vérifie les informations du formulaire, c.à.d si le login
saisi est bien un login autorisé, de même pour le mot de passe
 if ($login_valide == $_POST['login'] && $pwd_valide ==
$_POST['pwd']) {
     // dans notre cas, tout est ok, on peut démarrer la session

     // on la démarre
     session_start ();
     // on enregistre les paramètres du visiteur comme variables
de session ($login et $pwd).
     $_SESSION['login'] = $_POST['login'];
     $_SESSION['pwd'] = $_POST['pwd'];

     // on redirige le visiteur vers une page de la section membre
     header ('location: page_membre.php');
   }
```

```
   else {
     // Le visiteur n'a pas été reconnu comme étant membre du
site. On utilise alors javascript lui signalant ce fait
     echo '<body onLoad="alert(\'Membre non reconnu...\')">';
     // puis on le redirige vers la page d'accueil
     echo '<meta http-equiv="refresh" content="0;URL=index.htm">';
   }
  }
  else {
   echo 'Les variables du formulaire ne sont pas déclarées.';
  }
?>
```

page_membre.php

```
<?php
// On démarre la session (ceci est indispensable dans toutes les
pages de notre section membre)
session_start ();

// On récupère les variables de session
if (isset($_SESSION['login']) && isset($_SESSION['pwd'])) {

   // On teste pour voir si les variables ont bien été
enregistrées
     echo '<html>';
     echo '<head>';
     echo '<title>Page de la section membre</title>';
     echo '</head>';

     echo '<body>';
     echo 'Votre login est '.$_SESSION['login'].' et votre mot de
passe est '.$_SESSION['pwd'].'.';
     echo '<br />';

     // On affiche un lien pour fermer la session
     echo '<a href="./logout.php">Déconnexion</a>';
    }
    else {
     echo 'Les variables ne sont pas déclarées.';
    }
?>
```

logout.php

```php
<?php
// On démarre la session
session_start ();

// On détruit les variables de la session
session_unset ();

// On détruit la session
session_destroy ();

// On redirige le visiteur vers la page d'accueil
header ('location: index.html');
?>
```

Formulaire

formulaire.html

```
<h1>Formulaire HTML</h1>

        <form action="formulaire.php" method="post">
        <!-- <form action="formulaire2.php" method="post"> --!>
            <div class="c100">
                <label for="prenom">Prénom : </label>
                <input type="text" id="prenom" name="prenom">
            </div>
            <div class="c100">
                <label for="mail">Email : </label>
                <input type="email" id="mail" name="mail">
            </div>
            <div class="c100">
                <label for="age">Age : </label>
                <input type="number" id="age" name="age">
            </div>
            <div class="c100">
                <input type="radio" id="femme" name="sexe"
value="femme">
                <label for="femme">Femme</label>
                <input type="radio" id="homme" name="sexe"
value="homme">
                <label for="homme">Homme</label>
                <input type="radio" id="autre" name="sexe"
value="autre">
                <label for="autre">Autre</label>
            </div>
            <div class="c100">
                <label for="pays">Pays de résidence :</label>
                <select id="pays" name="pays">
                    <optgroup label="Europe">
                        <option value="france">France</option>
                        <option value="belgique">Belgique</option>
                        <option value="suisse">Suisse</option>
                    </optgroup>
                    <optgroup label="Afrique">
                        <option value="algerie">Algérie</option>
                        <option value="tunisie">Tunisie</option>
                        <option value="maroc">Maroc</option>
                        <option value="madagascar">Madagascar</
option>
                        <option value="benin">Bénin</option>
                        <option value="togo">Togo</option>
                    </optgroup>
                    <optgroup label="Amerique">
                        <option value="canada">Canada</option>
                    </optgroup>
```

```
            </select>
        </div>
        <div class="c100" id="submit">
            <input type="submit" value="Envoyer">
        </div>
    </form>
```

formulaire.css

```
form{
    width: 50%;
    background-color: rgba(20,178,202,0.2);
    padding : 5px 0px;
}
.c100{
    width: 100%;
    margin: 20px;
}
label{
    display: inline-block;
    min-width: 20%;
}
input[type="submit"]{
    color: RGB(144,12,63);
    border-radius: 5px;
    padding: 5px 10px;
    font-size: 14px;
    border: 2px solid RGB(144,12,63);
}
input[type="submit"]:hover{
    background-color: RGB(144,12,63);
    color: #fff;
    font-size: 15px;
    cursor: pointer;
    box-shadow: 0px 0px 5px 0px #777;
}
```

formulaire.php

```
<!DOCTYPE html>
<html>
    <head>
        <title>Page de traitement</title>
        <meta charset="utf-8">
    </head>
    <body>
        <p>Dans le formulaire précédent, vous avez fourni les
        informations suivantes :</p>
```

```
        <?php
            echo 'Prénom : '.$_POST["prenom"].'<br>';
            echo 'Email : ' .$_POST["mail"].'<br>';
            echo 'Age : ' .$_POST["age"].'<br>';
            echo 'Sexe : ' .$_POST["sexe"].'<br>';
            echo 'Pays : ' .$_POST["pays"].'<br>';
        ?>
    </body>
</html>
```

formulaire2.php

```
<?php
header("location:form-merci.html");
?>
```

form-merci.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Cours</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="formulaire.css">
    </head>
    <body>
        <h1>Formulaire bien envoyé</h1>
        <p>Merci d'avoir remplis le formulaire</p>
    </body>
</html>
```

Exemple 2 :

BDD :

```
<?php
    $serveur = "localhost";
    $dbname = "cours_snir";
    $user = "root";
    $pass = "";

    try{
        //On se connecte à la BDD
        $dbco = new PDO("mysql:host=$serveur;dbname=$dbname",
$user,$pass);
```

```
        $dbco->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

        //On crée une table form
        $form = "CREATE TABLE form(
            id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
            prenom TEXT,
            mail TEXT,
            age INT,
            sexe TEXT,
            pays TEXT)";
        $dbco->exec($form);
    }
    catch(PDOException $e){
        echo 'Erreur : '.$e->getMessage();
    }
?>
```

formulaire3.php

```
<?php
    $serveur = "localhost";
    $dbname = "cours_snir";
    $user = "root";
    $pass = "";

    $prenom = $_POST["prenom"];
    $mail = $_POST["mail"];
    $age = $_POST["age"];
    $sexe = $_POST["sexe"];
    $pays = $_POST["pays"];

    try{
        //On se connecte à la BDD
        $dbco = new PDO("mysql:host=$serveur;dbname=$dbname",
$user,$pass);
        $dbco->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

        //On insère les données reçues
        $sth = $dbco->prepare("
            INSERT INTO form(prenom, mail, age, sexe, pays)
            VALUES(:prenom, :mail, :age, :sexe, :pays)");
        $sth->bindParam(':prenom',$prenom);
        $sth->bindParam(':mail',$mail);
        $sth->bindParam(':age',$age);
        $sth->bindParam(':sexe',$sexe);
        $sth->bindParam(':pays',$pays);
        $sth->execute();
```

```php
        //On renvoie l'utilisateur vers la page de remerciement
        header("Location:form-merci.html");
    }
    catch(PDOException $e){
        echo 'Impossible de traiter les données. Erreur : '.$e->getMessage();
    }
?>
```

Sécurisation et validation de formulaires

Lorsqu'on crée des formulaires, c'est généralement pour demander aux utilisateurs de nous envoyer des données. Si on ne met pas en place des systèmes de filtre sur le type de données qui peuvent être envoyées pour chaque champ et de vérification ensuite de la qualité des données envoyées, les données récoltées vont alors pouvoir être aberrantes ou même potentiellement dangereuses.

Deux niveaux de vérifications (dans le navigateur / côté serveur) doivent être implémentés lors de la création de formulaires. En effet, n'utiliser qu'une validation dans le navigateur laisse de sérieuses failles de sécurité dans notre formulaire puisque les utilisateurs malveillants peuvent désactiver ces vérifications. N'effectuer qu'une série de vérifications côté serveur, d'autre part, serait également une très mauvaise idée d'un point de vue expérience utilisateur puisque ces vérifications sont effectuées une fois le formulaire envoyé.

Les failles XSS et l'injection

Une attaque XSS consiste en l'injection d'un code dans le formulaire qui va permettre au hacker d'exécuter des scripts JavaScript dans le navigateur de la victime.

xss.php

```
<!DOCTYPE html>
<html>
    <head>
        <title>Cours PHP / MySQL</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="formulaire.css">
    </head>
    <body>
        <h1>Formulaire HTML</h1>

        <form action="xss.php" method="post">
            <div class="c100">
                <label for="prenom">Prénom : </label>
                <input type="text" id="prenom" name="prenom"
style="width:30em">
            </div>
            <div class="c100">
                <label for="mail">Email : </label>
                <input type="email" id="mail" name="mail">
            </div>
            <div class="c100">
                <label for="age">Age : </label>
                <input type="number" id="age" name="age" min="12"
max="99">
            </div>
            <div class="c100" id="submit">
                <input type="submit" value="Envoyer">
            </div>
        </form>
```

```php
<?php
    $serveur = "localhost";
    $dbname = "cours_snir";
    $user = "root";
    $pass = "";

    $prenom = $_POST["prenom"];
    $mail = $_POST["mail"];
    $age = $_POST["age"];

    try{
        //On se connecte à la BDD
        $dbco = new
PDO("mysql:host=$serveur;dbname=$dbname",$user,$pass);
        $dbco->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

        //On insère les données reçues si les champs sont
remplis
        if(!empty($prenom)  && !empty($mail) && !
empty($age)){
            $sth = $dbco->prepare("
                INSERT INTO form(prenom, mail, age)
                VALUES(:prenom, :mail, :age)");
            $sth->bindParam(':prenom',$prenom);
            $sth->bindParam(':mail',$mail);
            $sth->bindParam(':age',$age);
            $sth->execute();
        }

        //On récupère les infos de la table
        $sth = $dbco->prepare("SELECT prenom, mail, age
FROM form");
        $sth->execute();
        //On affiche les infos de la table
        $resultat = $sth->fetchAll(PDO::FETCH_ASSOC);
        $keys = array_keys($resultat);
        for($i = 0; $i < count($resultat); $i++){
            $n = $i + 1;
            echo 'Utilisateur n°' .$n. ' :<br>';
            foreach($resultat[$keys[$i]] as $key =>
$value){
                echo $key. ' : ' .$value. '<br>';
            }
            echo '<br>';
        }
    }
    catch(PDOException $e){
        echo 'Impossible de traiter les données. Erreur :
'.$e->getMessage();
    }
```

```
        ?>

    </body>
</html>
```

On insère les données dans la table `form,` ensuite, on récupère les données dans la table et on les affiche. On récupère nos données sous forme d'un tableau associatif avec `fetchAll(PDO::FETCH_ASSOC)`.

La variable `$resultat` va alors être n tableau multidimensionnel qui va contenir plusieurs tableaux associatifs (un par entrée). On utilise donc une boucle `foreach` pour récupérer les données dans chaque tableau associatif et une boucle `for` pour passer d'un tableau associatif à l'autre.

Le code ne contient aucune réelle sécurisation. Un utilisateur va donc pouvoir passer plus ou moins ce qu'il veut comme données dans le champ prenom. Par exemple, il peut tout à fait passer un élément `script`.

```
<script type="text/javascript">alert('Ordinateur hacké!')</
script>
```

La valeur donnée dans le champ prénom va être enregistrée sans problème en base de données puis être récupérée et affichée. Cependant, lorsqu'on récupère cette valeur, le navigateur va lire l'élément `script` et donc exécuter le code JavaScript qu'il contient.

La validation des données du formulaire dans le navigateur

Qu'est-ce qu'une validation de formulaire?

Allez sur n'importe quel site avec un formulaire d'inscription et vous remarquerez des retours si vous n'entrez pas les données dans le format attendu. Vous aurez des messages comme :

- « Ce champ est obligatoire » (vous ne pouvez pas le laisser vide)
- « Veuillez entrer votre numéro de téléphone au format xxx-xxxx » (il attend trois chiffres suivis d'un tiret, suivi de quatre chiffres).
- « Veuillez entrer une adresse e-mail valide » (ce que vous avez saisi ne ressemble pas à une adresse e-mail valide).
- « Votre mot de passe doit comporter entre 8 et 30 caractères et contenir une majuscule, un symbole et un chiffre » (sérieusement ?).

Si elles sont correctes, l'application permet que les données soient soumises au serveur et (généralement) sauvegardées dans une base de données ; si ce n'est pas le cas, elle émet des messages d'erreur pour expliquer ce que vous avez fait de mal.

L'objectif est de bloquer l'envoi du formulaire si certains champs ne sont pas correctement remplis et de demander aux utilisateurs de remplir correctement les champs invalides.

Le HTML5 propose aujourd'hui des options de validation relativement puissantes et couvrant la majorité des besoins.

La validation des données en HTML va principalement passer par l'ajout d'attributs dans les éléments de formulaire. Nous allons ainsi pouvoir utiliser les attributs suivants :

Attribut	Définition
size	Permet de spécifier le nombre de caractères dans un champ
minlength	Permet de spécifier le nombre minimum de caractères dans un champ
maxlength	Permet de spécifier le nombre maximum de caractères dans un champ
min	Permet de spécifier une valeur minimale pour un champ de type number ou date
max	Permet de spécifier une valeur maximale pour un champ de type number ou date
step	Permet de définir un multiple de validité pour un champ acceptant des donnés de type nombre ou date. En indiquant `step="4"` , les nombres valides seront -8, -4, 0, 4, 8, etc.
autocomplete	Permet d'activer l'autocomplétion pour un champ : si un utilisateur a déjà rempli un formulaire, des valeurs lui seront proposées automatiquement lorsqu'il va commencer à remplir le champ
required	Permet de forcer le remplissage d'un champ. Le formulaire ne pourra pas être envoyé si le champ est vide
pattern	Permet de préciser une expression régulière. La valeur du champ devra respecter la contrainte de la regex pour être valide

Reprenons le formulaire précédent et ajoutons quelques contraintes sur les données que l'on souhaite recevoir :

- Le prénom est désormais obligatoire et ne doit comporter que des lettres + éventuellement des espaces, tirets ou apostrophes. Sa taille ne doit pas accéder 20 caractères ;
- Le mail doit avoir au moins 1 caractère de type lettre ou chiffre + le symbole « @ » + à nouveau au moins 1 caractère de type lettre ou chiffre + le symbole « . » + au moins deux caractères de type lettre ou chiffre. Il est également obligatoire ;
- L'âge doit être un nombre et être compris entre 12 et 99 ans.

```
        <form action="xss.php" method="post">
            <div class="c100">
                <label for="prenom">Prénom : </label>
                <input type="text" id="prenom" name="prenom"
style="width:30em" required pattern="^[A-Za-z '-]+$"
maxlength="20">
            </div>
            <div class="c100">
                <label for="mail">Email : </label>
                <input type="email" id="mail" name="mail" required
pattern="^[A-Za-z]+@{1}[A-Za-z]+\.{1}[A-Za-z]{2,}$">
            </div>
            <div class="c100">
                <label for="age">Age : </label>
```

```
                <input type="number" id="age" name="age" min="12"
max="99">
            </div>
            <div class="c100" id="submit">
                <input type="submit" value="Envoyer">
            </div>
        </form>
```

On utilise l'attribut `required` pour nos champs « prenom » et « mail » pour indiquer qu'ils doivent être automatiquement remplis.
On utilise l'attribut `pattern` pour ajouter des contraintes sur la forme des données qui doivent être renseignées pour nos champs prenom et mail en fournissant des règles (regex : expressions régulières) adaptées.

La validation des données du formulaire sur serveur

La validation dans le navigateur va nous éviter une immense majorité de données invalides et donc d'avoir des données à priori exploitables.

Cependant, elle n'est pas suffisante contre des utilisateurs malveillants puisque n'importe qui peut neutraliser les attributs HTML ou le JavaScript en les désactivant dans le navigateur avant d'envoyer le formulaire.

Une validation côté serveur, en PHP, va donc également s'imposer pour filtrer les données potentiellement dangereuses.

Le PHP offre différentes options pour sécuriser les formulaires en testant la validité des données envoyées : la première fonction est `htmlspecialchars()`. Cette fonction va permettre d'échapper certains caractères spéciaux comme les chevrons « < » et « > » en les transformant en entités HTML.

En échappant les chevrons, on se prémunit d'une injection de code JavaScript puisque les balises `<script>` et `/<script>` vont être transformées en `& <script>` et `&/<script>` et ne vont donc pas être exécutées par le navigateur.

On va ensuite pouvoir utiliser d'autres fonctions pour nettoyer les données avant de les stocker comme `trim()` qui va supprimer les espaces inutiles et `stripslashes()` qui va supprimer les antislashes que certains hackers pourraient utiliser pour échapper des caractères spéciaux.

formulaire-valid.php

```
<?php
    $serveur = "localhost"; $dbname = "cours_snir"; $user =
"root"; $pass = "";

    $prenom = valid_donnees($_POST["prenom"]);
    $mail = valid_donnees($_POST["mail"]);
    $age = valid_donnees($_POST["age"]);
    $sexe = valid_donnees($_POST["sexe"]);
```

```php
    $pays = valid_donnees($_POST["pays"]);

    function valid_donnees($donnees){
        $donnees = trim($donnees);
        $donnees = stripslashes($donnees);
        $donnees = htmlspecialchars($donnees);
        return $donnees;
    }
?>
```

```php
<?php
    $serveur = "localhost"; $dbname = "cours"; $user = "root";
$pass = "root";

    $prenom = valid_donnees($_POST["prenom"]);
    $mail = valid_donnees($_POST["mail"]);
    $age = valid_donnees($_POST["age"]);
    $sexe = valid_donnees($_POST["sexe"]);
    $pays = valid_donnees($_POST["pays"]);

    function valid_donnees($donnees){
        $donnees = trim($donnees);
        $donnees = stripslashes($donnees);
        $donnees = htmlspecialchars($donnees);
        return $donnees;
    }

    /*Si les champs prenom et mail ne sont pas vides et si les
donnees ont
     *bien la forme attendue...*/
    if (!empty($prenom)
        && strlen($prenom) <= 20
        && preg_match("^[A-Za-z '-]+$",$prenom)
        && !empty($mail)
        && filter_var($mail, FILTER_VALIDATE_EMAIL)){

        try{
            //On se connecte à la BDD
            $dbco = new PDO("mysql:host=$eerveur;dbname=$dbname",
$user,$pass);
            $dbco->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
            //On insère les données reçues
            $sth = $dbco->prepare("
                INSERT INTO form(prenom, mail, age, sexe, pays)
                VALUES(:prenom, :mail, :age, :sexe, :pays)");
            $sth->bindParam(':prenom',$prenom);
            $sth->bindParam(':mail',$mail);
```

```
            $sth->bindParam(':age',$age);
            $sth->bindParam(':sexe',$sexe);
            $sth->bindParam(':pays',$pays);
            $sth->execute();
            //On renvoie l'utilisateur vers la page de
remerciement
            header("Location:form-merci.html");
        }
        catch(PDOException $e){
            echo 'Erreur : '.$e->getMessage();
        }
    }else{
        header("Location:formulaire.html");
    }
?>
```