Université Libanaise

Faculté de Technologie

Génie des Réseaux Informatiques et Télécommunications

الجامعة اللبنانيّة

كلية التكنولوجيا

قسم هندسة شبكات المعلوماتية والاتصالات

Architecture Client Serveur

Langage PHP – Gestion des etats

Préparé par:

Dr. Youssef ROUMIEH

E-mail : youssef.roumieh@gmail.com

Les problèmes rencontrés et les oublis constatés dans ce support sont à signaler à youssef.roumieh@gmail.com

Références

1. http://php.net/manual/fr/

2. **Web Applications Development with Microsoft .NET Framework 4.**

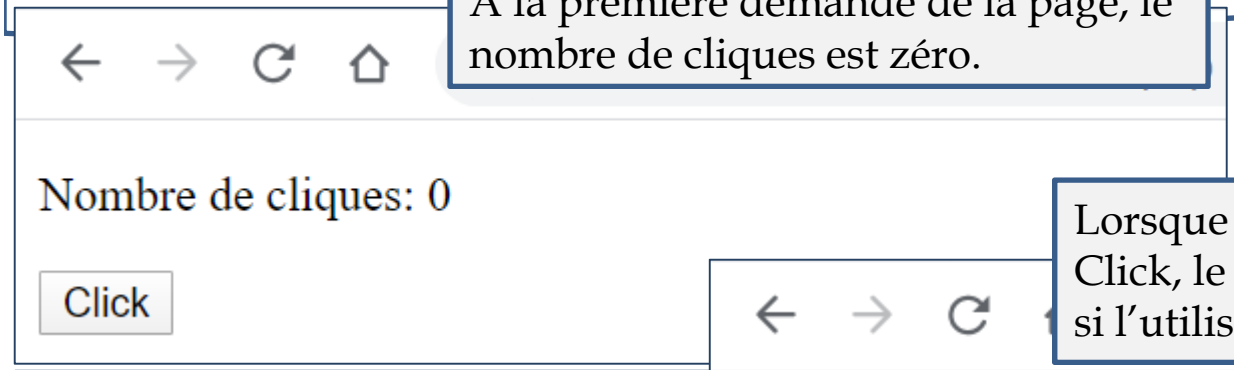
Copyright © 2010 by Tony Northrup and Mike Snell.

ISBN: 978-0-7356-2740-6

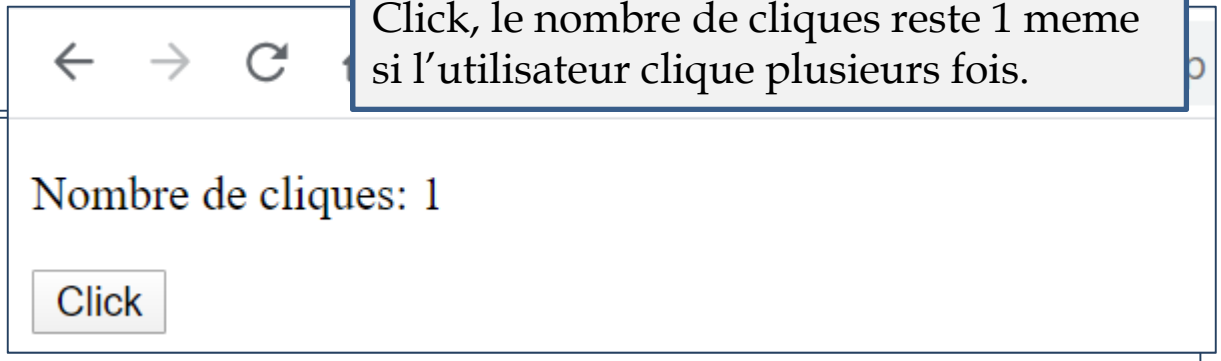
- La plus grande différence entre les applications Windows et les applications Web n'est pas l'interface utilisateur; c'est le cycle de vie.
  - Les utilisateurs ouvrent une application Windows et travaillent avec elle pendant des heures à la fois. Lorsqu'ils la ferment, l'application a la possibilité de sauvegarder leurs données.
  - Les demandes Web sont beaucoup plus à court terme. Un utilisateur peut demander une page à votre site Web et ne jamais y revenir. Même si un utilisateur passe des heures sur votre site, chaque demande individuelle nécessite la création de contrôles et d'objets, puis leur destruction.
    - des pages et des contrôles doivent être créés pour chaque demande d'utilisateur
    - Demander une page → construire la page
    - Demander à nouveau la même page → à nouveau construire la page à partir de zéro.

Demo: Compter le nombre de cliques

A la première demande de la page, le nombre de cliques est zéro.



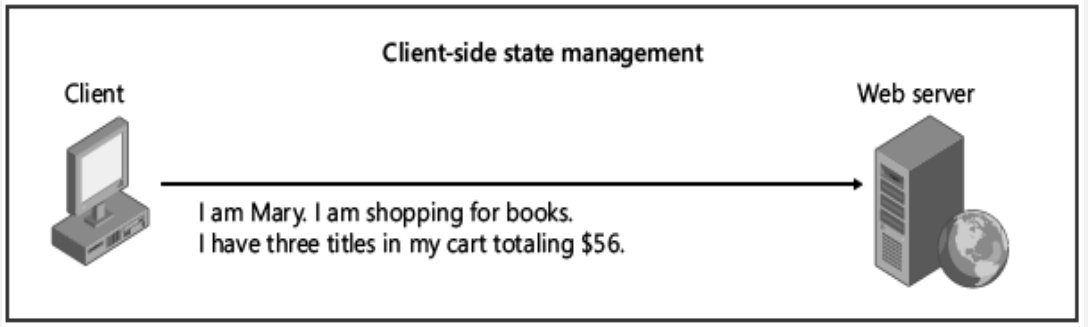
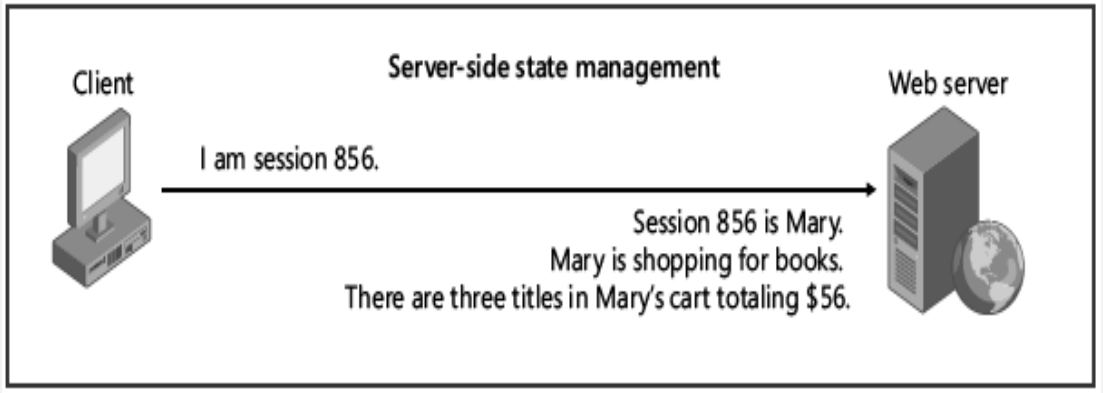
Lorsque l'utilisateur clique le bouton Click, le nombre de cliques reste 1 meme si l'utilisateur clique plusieurs fois.




```
<body>
  <?php
    $nb = 0;
    if(isset($_POST['btnClick']))
      $nb++;
  ?>
  <form method='post'>
    <p>Nombre de cliques: <?php echo $nb;?></p>
    <p><input type='submit' name='btnClick' value='Click'></p>
  </form>
</body>
```

Choisir entre la gestion d'état côté client ou celle côté serveur

- Les informations telles que le nom d'utilisateur, options de personnalisation, ou contenu de caddy, peut être stockée sur le client ou sur le serveur.

Gestion de l'état – cote Client : Le client envoie les informations au serveur à chaque requête.	
Gestion de l'état – cote Serveur : Le serveur stocke les informations, mais le serveur suit le client à l'aide d'une technique de gestion de l'état – côté client.	

- **Avantages de stocker les informations chez le client**
  - **une meilleure évolutivité (better scalability):**
    - moins de **consommation de la mémoire du serveur**. Servir plus de demandes.
  - **Support pour plusieurs serveurs web:**
    - Fournir les informations à n'importe quel serveur web.

- **Avantages de stocker les informations sur le serveur**
  - **une meilleure sécurité:**
    - **Les utilisateurs peuvent modifier les informations** de gestion d'état côté client. Vous ne devez jamais utiliser la gestion d'état côté client pour stocker des informations confidentielles telles que mots de passe, les niveaux d'accès, ou l'état d'authentification.
  - **Bande passante réduite:**
    - La sauvegarde de grandes quantités d'information sur le client augmente l'utilisation de la bande passante et les temps de chargement des pages, puisqu'à chaque aller-retour les informations sont envoyées.

- Le choix que vous faites pour gérer l'état de l'application doit être décidé en fonction de ces compromis. En général, on utilise un mélange de la gestion d'état de côté client et côté serveur.

- PHP fournit plusieurs techniques pour stocker des informations d'état (sur le client / sur le serveur) entre les requêtes:

Gestion des états - côté client	**Gestion des états - côté Serveur**
▪ Hidden Fields ▪ Query String ▪ Cookies	▪ Sessions

Utilisation de la gestion de l'état côté client

- Si votre application a besoin de s'adapter à des **milliers d'utilisateurs**, alors vous devriez sérieusement envisager **d'utiliser le client pour stocker l'état d'application**.
  - **La suppression de cette charge du serveur libère des ressources, permettant au serveur de traiter plus de demandes des utilisateurs**.

Champs cachés (Hidden Fields)

- Un champs caché n'est pas affiché à l'utilisateur (sauf si l'utilisateur choisit de voir la source de la page), puis envoyé vers le serveur lorsque la page est publiée (POST).

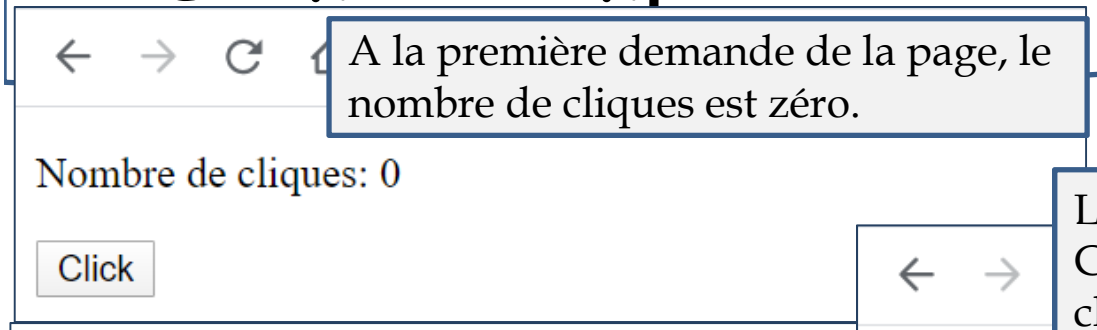
- Les champs cachés **ne stockent des informations que pour une seule page**.

- **Pour utiliser les champs cachés, vous devez soumettre vos pages au serveur en utilisant HTTP POST** (ce qui se produit lorsque l'utilisateur appuie sur un bouton de soumission).

```
<input type='hidden' name='txtID' value='100'>
```

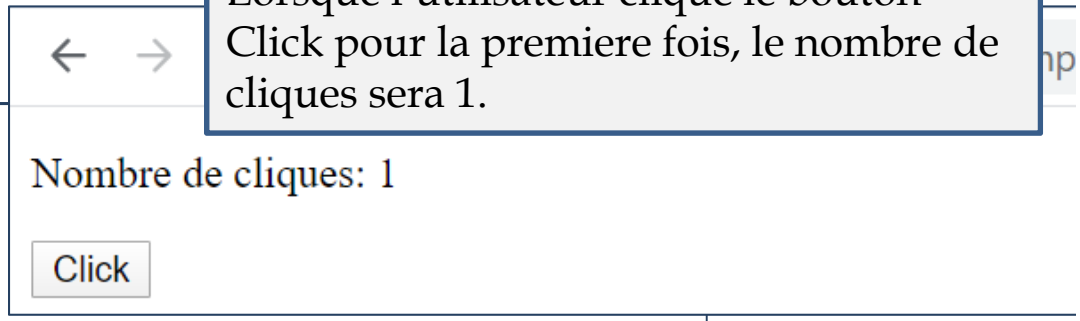
Demo: Compter le nombre de cliques

A la première demande de la page, le nombre de cliques est zéro.

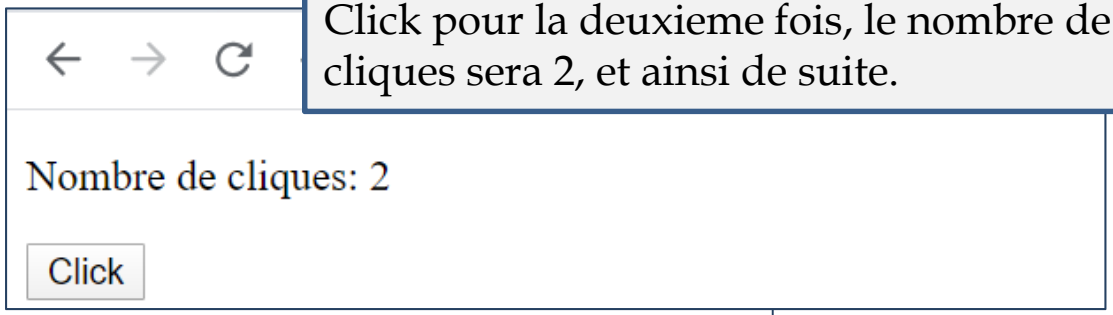



```
<body>
  <?php
    if(isset($_POST['btnClick'])){
      $nb = $_POST['txtNb'] + 1;
    }
    else {
      $nb = 0;
    }
  ?>
  <form method='post'>
    <p>Nombre de cliques: <?php echo $nb;?></p>
    <input type='hidden' name='txtNb' value='<?php echo $nb;?>'>
    <p><input type='submit' name='btnClick' value='Click'></p>
  </form>
</body>
```

Lorsque l'utilisateur clique le bouton Click pour la premiere fois, le nombre de cliques sera 1.



Lorsque l'utilisateur clique le bouton Click pour la deuxieme fois, le nombre de cliques sera 2, et ainsi de suite.



Chaînes de requête (Query Strings)

- Une chaîne de requête est une valeur stockée à la fin d'une URL.

- Les chaînes de requête sont définies à partir de l'URL avec un point d'interrogation (?) Suivi du terme de la chaîne de requête (ou du nom du paramètre), d'un signe égal (=) et de la valeur.
  - Vous pouvez ajouter plusieurs paramètres de chaîne de requête à l'aide de l'ampersand (&).
- Exemple :
  - http://support.microsoft.com/Default.php?kbid=315233
  - http://search.microsoft.com/results.php?mkt=en-US&setlang=en-US&q=hello+world

- **Accès aux paramètres de la chaîne de requête**
  - Valeurs envoyées à votre page via la chaîne de requête peuvent être récupérées par le serveur via le tableau associative $_GET ou $_REQUEST. Par exemple: $_GET['mkt']

- **Limitations**: Les chaînes de requête constituent un moyen simple mais limité pour conserver les informations d'état entre plusieurs pages.
  - Certains navigateurs imposent une limite de 2083 caractères sur la longueur de l'URL.
  - Les utilisateurs peuvent facilement modifier les chaînes de requête (les paramètres et les valeurs sont visibles pour l'utilisateur dans sa barre d'adresse).
- **Avantages :** L'un des grands avantages des chaînes de requête est que leurs données sont incluses dans les signets et les URL envoyées par courrier électronique.
  - les chaînes de requête sont le seul moyen permettant à un utilisateur d'inclure les données d'état lorsque vous copiez et colliez une URL à un autre utilisateur.
- ***Note:***
  - Vous devez vous attendre à ce que les utilisateurs modifient les données dans vos chaînes de requête. Pour cette raison, **vous devez toujours valider les données extraites d'une chaîne de requête.**

Demo: Compter le nombre de cliques

Nombre de cliques: 0

Click

A la première demande de la page, le nombre de cliques est zéro.

Lorsque l'utilisateur clique le bouton Click pour la premiere fois, le nombre de cliques sera 1.

Nombre de cliques: 1

Click

Nombre de cliques: 2

Click

Lorsque l'utilisateur clique le bouton Click pour la deuxieme fois, le nombre de cliques sera 2, et ainsi de suite.

```
<body>
  <?php
    if(isset($_POST['btnClick'])){
      $nb = $_GET['nb'] + 1;
    }
    else {
      $nb = 0;
    }
  ?>
  <form method='post' action='seance10_1.php?nb=<?php echo $nb;?>'>
    <p>Nombre de cliques: <?php echo $nb;?></p>
    <p><input type='submit' name='btnClick' value='Click'></p>
  </form>
</body>
```

Cookies

- Un cookie est une **petite quantité de données stockée chez le client** et transmise avec les demandes de votre site.
  - Le navigateur envoie cette donnée avec chaque requête d'une page au même serveur.

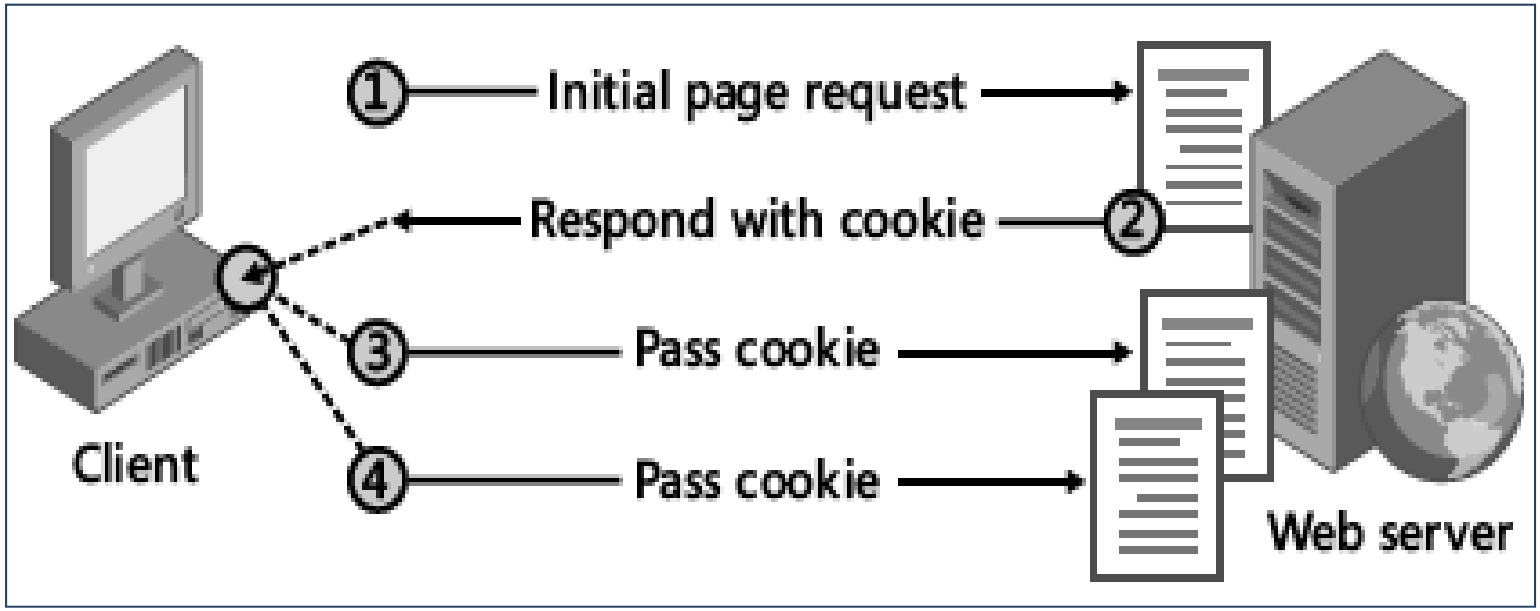
- Types de cookies :
  - **Cookies Persistant :** Les cookies sont **des fichiers texte** sur l'ordinateur client. Ces cookies restent existants même si l'utilisateur ferme le navigateur et rouvre à un moment ultérieur.
  - **Cookies Temporaires :** stockés dans la mémoire du navigateur du client.
    - Ces cookies ne sont utilisés que lors de la session Web en cours.
    - Ils sont perdus lorsque le navigateur se ferme.

Cookies

- **Les cookies sont la meilleure façon de stocker les données d'état qui doivent être disponibles pour de multiples pages Web sur un site Web**.
  - Les applications Web ont souvent **besoin de suivre les utilisateurs entre les demandes de page**.
  - Ces applications doivent assurer que l'utilisateur effectue la première requête est le même utilisateur effectuant les demandes subséquentes.

- **Les cookies peuvent être consultés à travers des visites différentes (par exemple, les préférences de l'utilisateur)**.

Les serveurs Web utilisent des cookies pour suivre les clients Web.



- Les **utilisateurs peuvent supprimer les cookies** de leur ordinateur à tout moment.
- Les cookies **ne permettent pas de résoudre le problème d'un utilisateur déplaçant d'un ordinateur à un autre** (ou d'un ordinateur vers un appareil mobile).
  - Dans ces cas, les préférences des utilisateurs ne vont pas toujours avec eux.
  - L'utilisateur besoin de se connecter pour réinitialiser leurs cookies.

Lecture et écriture des Cookies

- Une application web crée un cookie par **l'envoi au client comme un en-tête dans une réponse HTTP**.
  - Utiliser la function **`setcookie()`** pour définir un cookie.

```
bool setcookie ( string $name [, string $value ="" [,
     int $expire =0 [, string $path = "" [,
     string $domain = "" [, bool $secure = false [,
     bool $httponly = false ]]]]]] )
```

- **Parameters**
  - **name:** the name of the cookie.
  - **value:** The value of the cookie. This value is stored on the clients computer; do not store sensitive information. Assuming the **name** is *'cookiename'*, this value is retrieved through `$_COOKIE['cookiename']`
  - **expire:** The time the cookie expires. This is a Unix timestamp so is in number of seconds since the epoch.
    - Use the time() function plus the number of seconds before you want it to expire. Example: *time()+60*60*24*30* will set the cookie to expire in 30 days.
    - Or you might use mktime().
    - If set to 0, or omitted, the cookie will expire at the end of the session (when the browser closes).

```
bool setcookie ( string $name [, string $value ="" [,
  int $expire =0 [, string $path = "" [, string $domain = "" [,
  bool $secure = false [, bool $httponly = false ]]]]] )
```

- **Parameters**
  - **path:** The path on the server in which the cookie will be available on.
    - If set to '/', the cookie will be available within the entire **domain**.
    - If set to '/foo/', the cookie will only be available within the /foo/ directory and all sub-directories such as /foo/bar/ of **domain**.
    - The default value is the current directory that the cookie is being set in.
  - **domain:** The (sub)domain that the cookie is available to.
  - **secure:** Indicates that the cookie should only be transmitted over a secure HTTPS connection from the client (set to TRUE).
  - **Httponly:** When **TRUE** the cookie will be made accessible only through the HTTP protocol. This means that the cookie won't be accessible by scripting languages, such as JavaScript.
- **Valeurs de retour**
  - Si la sortie existe avant d'appeler cette fonction, setcookie () échouera et retournera FALSE. Si setcookie () s'exécute avec succès, il retournera TRUE. Cela n'indique pas si l'utilisateur a accepté le cookie.

```
<?php
        setcookie(“idclient”, “21000”, time()+600, “/”);
        setcookie(“language”, “en”);
?>
▪ Le premier cookie:
    ✓ expire 10 minutes après l’envoi
    ✓ envoyer à toute demande sur le site
▪ Le deuxième cookie:
    ✓ expire à l'envoi de la session.
    ✓ envoyé à toute demande de script dans le même
      répertoire que le script a envoyé le cookie.
```

- Une fois les cookies définis, vous pourrez y accéder lors du prochain chargement de page avec le tableau `$_COOKIE`. Les valeurs de cookie peuvent également exister dans `$_REQUEST`.
- Ce qui suit montre un exemple de ce code: `$_COOKIE['idclient']`

- **Note:**
  - Pour supprimer un cookie, vous écrasez le cookie et définissez une date d'expiration dans le passé. Vous ne pouvez pas supprimer directement les cookies car ils sont stockés sur l'ordinateur du client.

- Si vous limitez la portée à un répertoire, seules les pages de ce répertoire auront accès au cookie.
  - Exemple: `setcookie ( 'fn', 'mike', 0, '/MyApplication')`
  - La portée étant limitée au chemin d'accès /MyApplication de l'application, le navigateur envoie le cookie à n'importe quelle page du répertoire /MyApplication. Les pages situées en dehors de ce répertoire ne reçoivent pas le cookie, même si elles se trouvent sur le même serveur.

Stocker plusieurs valeurs dans un cookie

- La taille de votre cookie dépend du navigateur. Chaque cookie peut avoir une longueur maximale de 4 Ko.

- En outre, vous pouvez généralement stocker jusqu'à 20 cookies par site. Cela devrait être plus que suffisant pour la plupart des sites.

- Toutefois, si vous devez contourner la limite de 20 cookies, vous pouvez stocker plusieurs valeurs dans un seul cookie en définissant le nom du cookie et sa valeur de clé.

```
<?php
   setcookie(“idclient”, “21000”, time()+600, “/”);
   setcookie(“language”, “en”);
   setcookie(“client[id]”, “21000”, time()+600, “/”);
   setcookie(“client[language]”, “en”, time()+600, “/”);
?>
```

- Les propriétés de cookie, telles que Expires, Domain et Path, s’appliquent à toutes les valeurs d’un même cookie.

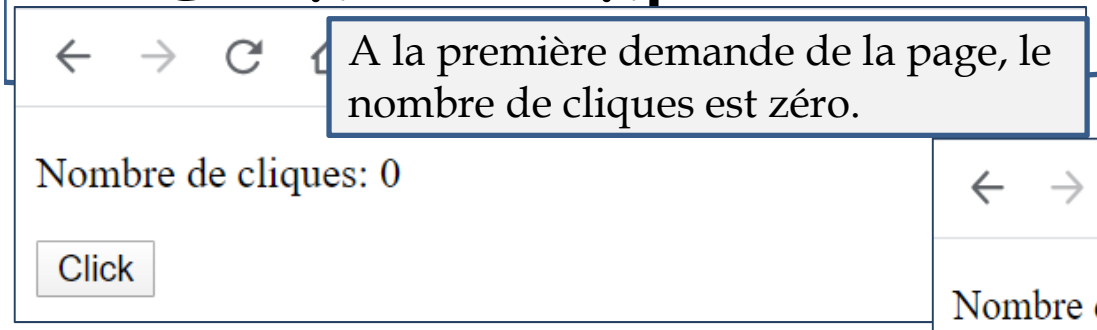
- Le tableau associatif $_COOKIE permet d'accéder à tous les cookies envoyés par le navigateur.

```
<?php
   foreach($_COOKIE as $key => $val){
       if(is_array($val)
          foreach($val as $key2=> $val2)
              echo "$key: $key2 = $val2<br>";
       else
          echo "$key = $val<br>";
       }
?>
```

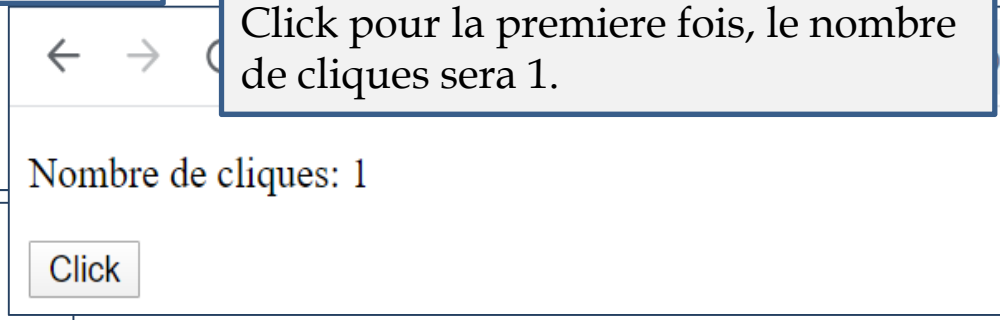
file1.php (dans le meme repertoire):	File2.php (dans un autre repertoire):
language =en	client:id=21000
client:id=21000	client:language = en
client:language = en	idclient = 21000
idclient = 21000	

Demo: Compter le nombre de cliques

A la première demande de la page, le nombre de cliques est zéro.

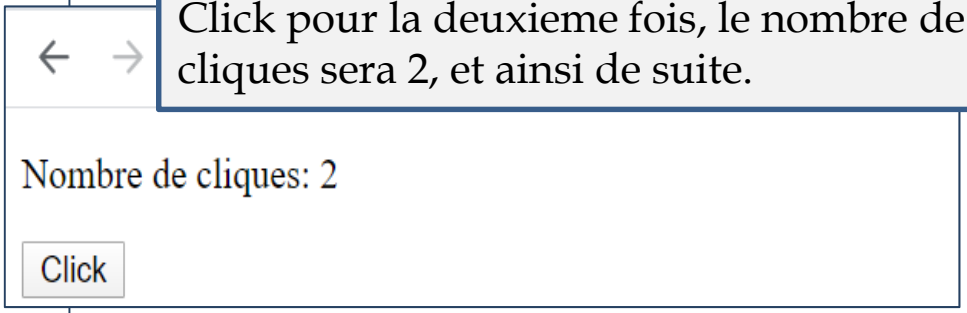


Lorsque l'utilisateur clique le bouton Click pour la premiere fois, le nombre de cliques sera 1.




```php
<?php
  if(isset($_POST['btnClick'])){
    if(isset($_COOKIE['nb_clicks']))
      $nb = $_COOKIE['nb_clicks'] + 1;
    else
      $nb = 1;
    setcookie("nb_clicks", $nb);
  }
  else
    $nb = 0;
?>
```

Lorsque l'utilisateur clique le bouton Click pour la deuxieme fois, le nombre de cliques sera 2, et ainsi de suite.




```html
<body>
  <form method='post'>
    <p>Nombre de cliques: <?php echo $nb;?></p>
    <p><input type='submit' name='btnClick' value='Click'></p>
  </form>
</body>
```

Utilisation de la gestion d'état côté serveur

- Dans certains scénarios, le client n'est pas le bon choix pour stocker l'état.
  - Votre état pourrait être plus impliqué et donc trop grand pour être transmis dans les deux sens.
  - Un état qui doit être sécurisé et même crypté et il ne doit pas être transmis sur un réseau.

Solution : utiliser le serveur pour gérer ces états.

- PHP fournit une manière pour stocker l'état sur le serveur et donc pour partager les informations entre les pages Web sans envoyer les données au client. Cette méthode est appelée *état de session*.

Utilisation de la gestion d'état côté serveur

- L'état de session est un état spécifique à un utilisateur, il est stocké sur le serveur. Il est disponible uniquement aux pages accessibles par un seul utilisateur au cours d'une visite de votre site.
  - Par exemple, si un utilisateur fournit un nom, votre application pourrait le conserver pour le reste de la visite de l'utilisateur en le stockant dans l'état de session.
  - Par exemple, si un utilisateur passe par un processus en plusieurs étapes pour inscrire à votre site, vous pouvez stocker temporairement ces données entre les pages jusqu'à ce que l'utilisateur ait terminé le processus.

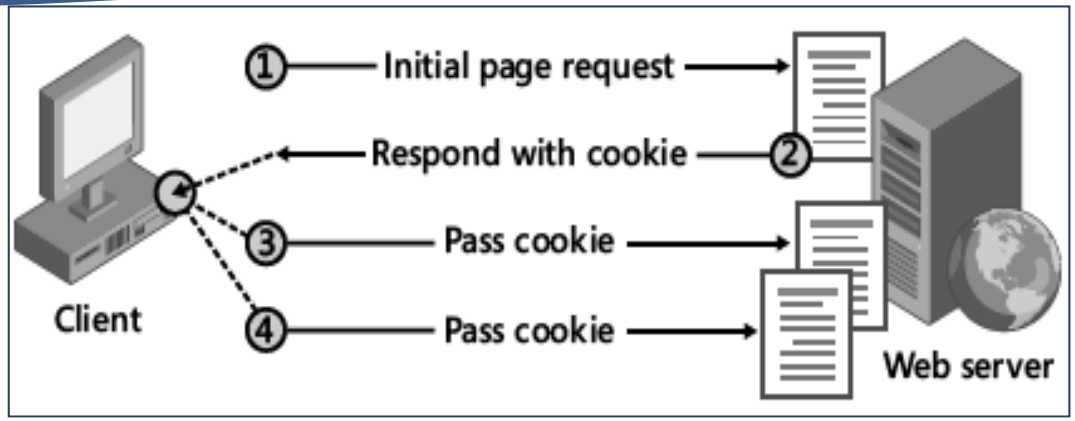
- Chaque utilisateur sur votre site a alors son propre isolement état de session fonctionnant dans le processus de l'application sur le serveur.
  - Cet état est disponible à différentes pages que l'utilisateur effectue des demandes subséquentes au serveur.
  - L'état de session est toutefois perdu si l'utilisateur se termine sa session (ou délai).

Utilisez la fonction session_start ()

crée une session

ou

restaure la session en cours si un identifiant de session a été envoyé par un cookie ou par une URL.



- Créer une session
  - Attribuez un identifiant de session unique (identifiant de session) à l'utilisateur.
  - Chaque client Web reçoit un identifiant de session différent.
  - Par défaut, l'état de la session utilise des cookies pour suivre les sessions des utilisateurs.
  - Les sessions utilisent un cookie appelé PHPSESSID (par défaut). Lorsque vous démarrez une session sur une page, l'interpréteur php vérifie la présence de ce cookie et le définit s'il n'existe pas.
  - La valeur du cookie PHPSESSID est une chaîne alphanumérique aléatoire. Par exemple: 4777dbob92dffdaf49219a27c59d132b
  - Sans les cookies, php suit les sessions en utilisant l'URL, en incorporant l'ID de session dans l'URL si la directive session.use_trans_id est activée.
  - Toutes les URL doivent inclure une partie arguments comme:

```
?PHPSESSID=4777dbob92dffdaf49219a27c59d132b
```

- **Remarque:** Cette fonction doit être écrite dans chaque page utilisant la session.
  - Si vous souhaitez utiliser des sessions dans toutes vos pages, définissez la directive de configuration session auto_start sur on. Il n'est pas nécessaire d'appeler session_start () dans chaque page.
- **Remarque:** session_start () et setcoookie () doivent être appelés avant toute sortie.

Directives

- Le comportement de ces fonctions est affecté par les paramètres du fichier *`php.ini`*.
  - `session.name`: name of the session
  - `session.save_handler`: storing and reading data from a session. By default a file is used.
  - `session.save_path`: if file is used to save session: the argument specifies the path where the file will be stored.
  - `session.use_cookies`: method of transmission of session ID to be used. The value off will force php to use the URL.
  - `session.use_trans_sid`: auto-completion of the URL with session ID.

Directives

- Le comportement de ces fonctions est affecté par les paramètres du fichier *php.ini*.
  - session.cookie_lifetime integer
    - specifies the lifetime of the cookie in seconds which is sent to the browser. The value 0 means "until the browser is closed." Defaults to *0*.
  - session.cookie_path string
    - specifies path to set in the session cookie. Defaults to /.
  - session.cookie_domain string
    - specifies the domain to set in the session cookie. Default is none at all meaning the host name of the server which generated the cookie according to cookies specification.
  - session.cookie_secure Boolean
    - specifies whether cookies should only be sent over secure connections. Defaults to *off*.
  - session.cookie_httponly Boolean
    - Marks the cookie as accessible only through the HTTP protocol. This means that the cookie won't be accessible by scripting languages, such as JavaScript. This setting can effectively help to reduce identity theft through XSS attacks (although it is not supported by all browsers).
- phpinfo(): information about the php configuration
  - Session support: value of the session directives (php.ini)

- `string ini_set (string $varname , string $newvalue)`
  - Définit la valeur de l'option de configuration donnée. L'option de configuration conservera cette nouvelle valeur pendant l'exécution du script et sera restaurée à la fin du script.
  - Example: `Init_set("session.use_cookies", 0);`
- `string ini_get ( string $varname )`
  - Retourne la valeur de l'option de configuration en cas de succès

```
<?php
   ini_set("session.use_cookies", 1); // use cookies
   ini_set("session.use_trans_sid", 0);
         // disables the auto-completion of URLs

   session_set_cookie_params(0, "/");
   //par 1: the cookie will be valid until the user close the session
   //par 2: the cookie will be valid for only tree's website

   session_name("test");
   session_id("22");
   session_start();
   echo session_name() . " = ". Session_id();
?>
```

- `string session_save_path ([ string $path ] )`
  - Obtenir et/ou définir le chemin d'enregistrement de la session en cours du répertoire en cours utilisé pour enregistrer les données de la session.
- **Variables of session**
  - La fonction isset vérifie si la variable de session existe.
  - La fonction unset détruit une variable de session. Exemple: unset($_SESSION['v1'])
  - Utilisez le tableau associatif $_SESSION pour lire et écrire une valeur dans la session.
  - Exemple:

```
$_SESSION['var_name'] = value
echo $_SESSION['v1'];
```

- `bool session_destroy ( void )`
  - session_destroy () détruit toutes les données associées à la session en cours. Cela ne désactive pas les variables globales associées à la session, ni le cookie de session. Pour utiliser à nouveau les variables de session, session_start() doit être appelé.
  - Remarque: Nettoyez le tableau $_SESSION plutôt que de détruire les données de session.

Demo: Compter le nombre de cliques

A la première demande de la page, le nombre de cliques est zéro.

Nombre de cliques: 0

Click

Lorsque l'utilisateur clique le bouton Click pour la premiere fois, le nombre de cliques sera 1.

Nombre de cliques: 1

Click

Lorsque l'utilisateur clique le bouton Click pour la deuxieme fois, le nombre de cliques sera 2, et ainsi de suite.

Nombre de cliques: 2

Click

```php
<?php
  session_start();
  if(isset($_POST['btnClick'])){
    if(isset($_SESSION['nb_clicks']))
      $nb = $_SESSION['nb_clicks'] + 1;
    else
      $nb = 1;
    $_SESSION["nb_clicks"] = $nb;
  }
  else
    $nb = 0;
?>
```

```php
<body>
  <form method='post'>
    <p>Nombre de cliques: <?php echo $nb;?></p>
    <p><input type='submit' name='btnClick' value='Click'></p>
  </form>
</body>
```

Exercices

- Ecrire le script php qui permet d'afficher la date et l'heure de la derniere visite de l'utilisateur. La première fois que l'utilisateur visite la page, le message suivant sera affiche «Aucune valeur est définie».
  1. En utilisant un champs cache (Hidden Field)
  2. En utilisant une chaine de requete (Query String)
  3. En utilisant un cookie.
  4. En utilisant une session.