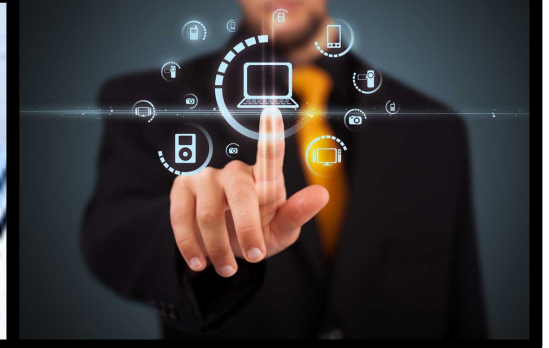


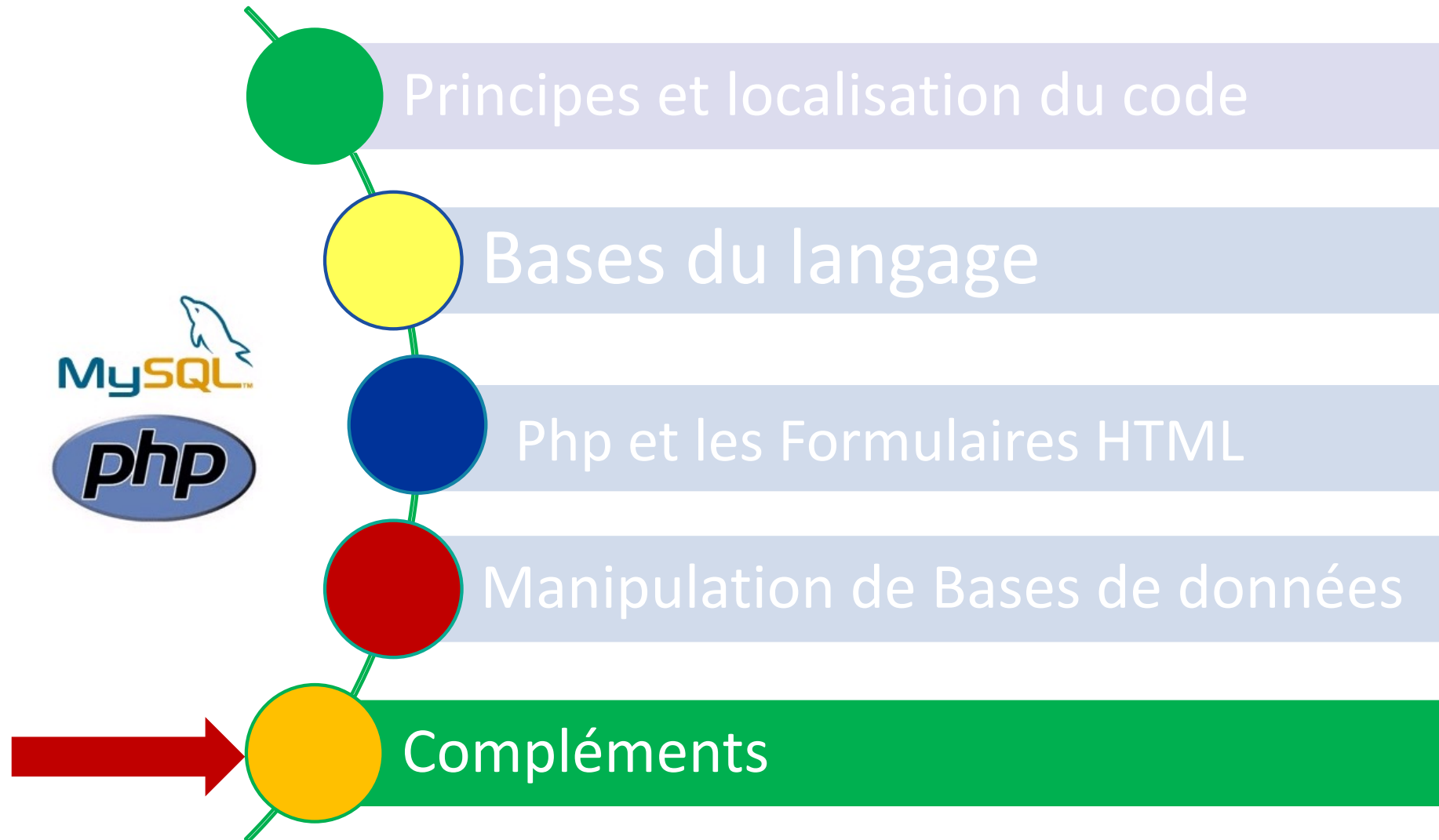
UNIVERSITE DE CORSE
Licence SPI 2ème année
Option Informatique

UE Programmation WEB Back End
CH1.5 – Compléments



Evelyne VITTORI
vittori_e@univ-corse.fr

CH1- Fondamentaux Php



CH1.5 – Compléments

Objectifs

- Comprendre le mécanisme de l'injection SQL et savoir l'éviter
- Transmettre des paramètres à une page avec le tableau get
- Comprendre et savoir manipuler des variables de session
- Savoir manipuler des cookies en php





Injection SQL

Principe
Solution

Notion d'injection SQL

■ Principe

Un champ de recherche d'un formulaire est utilisé pour « injecter » du code SQL malveillant.

Recherche Produits

id du produit :

1; DELETE FROM PRODUITS;

Rechercher Produit

Code php:

```
$id=$_POST['id'];  
$req = "SELECT * FROM  
produits WHERE id_produit = $id";  
$res = $bdd->query($req);
```

requête exécutée:

```
$req = "SELECT * FROM produits WHERE id_produit = 1; DELETE FROM PRODUITS; "
```

Les lignes de la table PRODUITS sont supprimées!!⁵

Exercice : test Injection SQL



- *Récupérez sur l'ENT les fichiers `bdeCommerce.sql` et `recherche.php`.*
- *Exécutez le script `bdeCommerce.sql` sous `phpMyAdmin`.*
- *Modifiez le fichier `recherche.php` au niveau de la connexion à la base de données: numéro de port et mot de passe.*
- *Lancez le script `php recherche.php`.*

Exercice: test Injection SQL (suite)



- Lancez le script php recherche.php.
- Testez le en saisissant l'id_produit 1.
- Saisissez à présent la chaine suivante dans la zone de saisie de l'id_produit:

1 or 1='1';

- Que constatez-vous?
- Saisissez à présent la chaine suivante dans la zone de saisie de l'id_produit:

1 ; DELETE FROM produits;

- Retournez sur phpMyAdmin: Que constatez-vous?

Comment éviter les injections SQL?

- La Solution la plus sûre : Définir des requêtes préparées
 - Construire la chaîne de caractère de la requête avec des ? à la place des variables
 - Utiliser la méthode *prepare*
 - puis Utiliser la méthode *bindParam* pour affecter les valeurs effectives aux paramètres
 - et enfin Utiliser la méthode *execute* pour lancer exécution de la requête
- Autre solution: Vérifier les chaînes saisies dans le formulaire par une fonction php (et/ou javascript)

Ou utiliser *execute* avec en paramètre un tableau cf. CH1.4

Comment éviter les injections SQL?

- Exemple d'utilisation de requêtes préparées.

Ancien Code php:

```
$id=$_POST['id'];  
$req = "SELECT * FROM produits WHERE id_produit = $id";  
$res = $bdd->query($req);  
..... $prod = $res->fetch(PDO::FETCH_ASSOC); ....
```

Nouveau Code php:

```
$id=$_POST['id'];  
$req = "SELECT * FROM produits WHERE id_produit = ?";  
$prep=$bdd->prepare($req);  
$prep->bindParam(1,$id);  
$prep->execute();  
..... $prod = $prep->fetch(PDO::FETCH_ASSOC); ....
```

Pourquoi les requêtes préparées protègent des injections SQL?

- Les requêtes préparées sont « **pré-interprétées** » lors de leur préparation (méthode prepare)
- Les paramètres effectifs sont envoyés au SGBD uniquement comme des **données**, ils ne sont plus interprétés
- Donc même si les paramètres effectifs contiennent des commandes sql, ces commandes ne seront pas exécutées.

Exercice: Eviter les Injections SQL

- Modifiez le script php recherche.php afin qu'il n'utilise qu'une requête préparée.
- Testez à nouveau les 3 exemples précédents afin de vérifier que l'injection sql n'est plus possible:
 - `id_produit = 1`
 - `id_produit=1 or 1='1';`
 - `id_produit= 1 ; DELETE FROM produits;`





Ouverture d'une page avec paramètres

- Envoi de données par formulaire get
- Appel direct par URL

Principes de la transmission get

- Envoi de données par un formulaire en mode get → les données sont transmises avec l'URL
- Appel direct de la page avec les paramètres en ligne de commande → URL spécifiée dans le code
- Récupération des données dans le tableau associatif supraglobal `$_GET`

Envoi de données par formulaire get

Formulaire

Nom : Toto

Valider

syntaxe:
script.xxx?var1=val1?var2=val1 ...

nom de la donnée (propriété name)

valeur saisie

file:///D:/Enseignement/Web/Sites/exemplesCours/script.php?nom=Toto

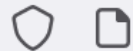
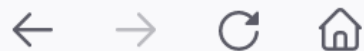
```
<h1>Formulaire</h1>
<form method="get" action="script.php">
  Nom : <input type="text" name="nom"><br/>
  <input type="submit" value="Valider" >
</form>
```


Envoi de données par formulaire get

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Recherche utilisateurs</title>
</head>
<body>
<h1>Recherche utilisateurs</h1>
<form method="get" action="UtilisateurGet.php">
  Login :<br>
  <input type="text" name="login"><br><br>
  <input type="submit" value="Recherche login"><br>
</form>
```

Recherche utilisateurs

Login :



localhost:8888/TPCOURS/UtilisateurGet.php?login=atoto

Authentification Uni... Applications Web p... Overview (Java Plat... https://www.google.... Mon Drive -

Affichage Utilisateur

Login	Mot de passe	Nom	Categorie
atoto	aaa	toto	administrateur

Envoi de données par formulaire get

← → ↻ 🏠 localhost:8888/TPCOURS/UtilisateurGet.php?login=atoto

Authentification Uni... Applications Web p... Overview (Java Plat... https://www.google... Mon Drive -

Affichage Utilisateur

Login	Mot de passe	Nom	Categorie
atoto	aaa	toto	administrateur

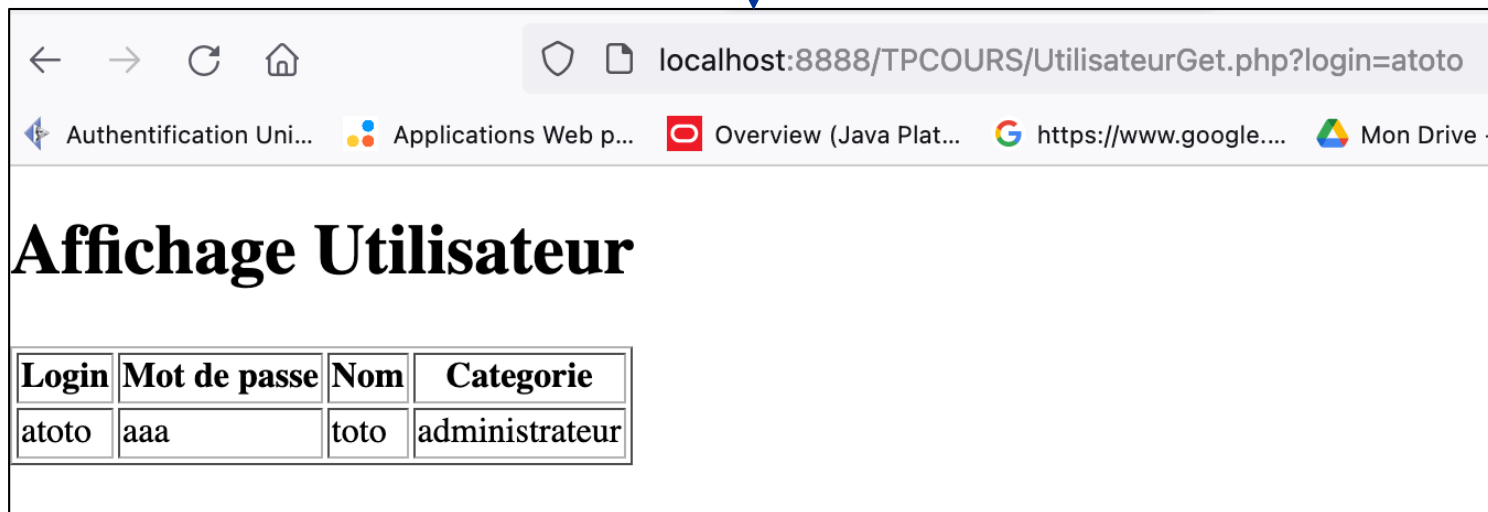
```
<?php
$dbdd = new PDO('mysql:host=localhost:8889;dbname=securite;charset=utf8', 'root', 'root');
$login = $_GET['login'];
$res = $dbdd->query('SELECT * FROM t_utilisateur WHERE login='.$login');
if ($res) {
    $user = $res->fetch(PDO::FETCH_ASSOC);
    if ($user) { //il y a au moins une réponse
        echo "<table border='1'> <tr><th>Login</th>
        <th>Mot de passe</th> <th>Nom</th><th>Categorie</th>
        </tr>";
        echo "<tr>";
        echo "<td>" . $user['login'] . "</td>";
        echo "<td>" . $user['mdp'] . "</td>";
        echo "<td>" . $user['nom'] . "</td>";
        echo "<td>" . $user['categorie'] . "</td>";
        echo "</tr>";
        echo " </table>";
    }
}
```

Appel direct avec paramètres dans l'URL

Un exemple avec une valeur constante
« en dur ! »

```
<a href='UtilisateurGet.php?login=atoto'>Acces direct Utilisateur de login atoto </a>
```

Acces direct Utilisateur de login atoto



The screenshot shows a web browser window with the address bar displaying 'localhost:8888/TPCOURS/UtilisateurGet.php?login=atoto'. The page title is 'Affichage Utilisateur'. Below the title is a table with the following data:

Login	Mot de passe	Nom	Categorie
atoto	aaa	toto	administrateur

Transmission directe par URL

Un exemple plus intéressant avec le lien dans un tableau

Affichage des utilisateurs triés par nom

Login	Mot de passe	Nom	Categorie
btiti	bbb	titi	user
atoto	aaa	toto	administrateur

localhost:8888/TPCOURS/UtilisateurGet.php?login=atoto

Authentification Uni... Applications Web p... Overview (Java Plat... https://www.google... Mon Drive -

Affichage Utilisateur

Login	Mot de passe	Nom	Categorie
atoto	aaa	toto	administrateur

```
<?php
bdd = new PDO('mysql:host=localhost:8889;dbname=securite;charset=utf8', 'root', 'root');
$requete="SELECT * FROM t_utilisateur ORDER BY nom";
$res = $bdd->query($requete);
if ($res) {
    while ($user = $res->fetch(PDO::FETCH_ASSOC)) {
        echo "<tr>";
        echo "<td><a href='\"UtilisateurGet.php?login=\" . $user['login'] . \"'>\" . $user['login'] . \"</td>\";
        echo "<td>\" . $user['mdp'] . \"</td>\";
        echo "<td>\" . $user['nom'] . \"</td>\";
        echo "<td>\" . $user['categorie'] . \"</td>\";
        echo "</tr>\";
    }
}
```



Variables de Session

Stockage sur le serveur

Session et variables de session

- La notion de session permet de partager des données entre plusieurs pages d'un site.
- Lorsqu'un utilisateur se connecte au site, une session est créée:
 - Un identifiant est attribué à la session.
 - Les variables sauvegardées dans la session (**variables de session**) seront accessibles à partir de toutes les pages du site

Variables sauvegardées sur le serveur et non chez le client comme les cookies

Gestion des sessions

- **session_start()** : fonction de lancement du système de session (au premier appel un numéro de session est alloué à l'utilisateur)

Cette fonction doit être appelée au tout début de chacune des pages dans lesquelles on utilise des variables de session.

avant <!DOCTYPE>)

- **session_destroy()** : fonction de fermeture de la session de l'utilisateur courant
 - fonction appelée automatiquement si l'utilisateur ne charge pas de pages pendant plusieurs minutes

Tableau \$_SESSION

- Tableau associatif **superglobal** qui contient la description de toutes les variables de session

\$_SESSION

idClient	salaire
Toto	3000


```
$_SESSION["idClient"]="Toto";  
$_SESSION["salaire"]= 3000;
```

- Réinitialisation de toutes les variables de session
 - `$_SESSION = array();`
- Restauration du tableau `$_SESSION` à sa dernière version sauvegardée
 - `session_reset();`

Enregistrer des valeurs dans une variable de session

```
<?php
    session_start();
    if (!isset($_SESSION['compteur'])) {
        $_SESSION['compteur'] = 0;
    } else {
        $_SESSION['compteur']++;
    }
?>
```

A ne pas oublier!!



Supprimer une variable de session

Supprime la variable compteur du tableau superglobal `$_SESSION`

```
<?php  
session_start();  
unset($_SESSION['compteur']);  
?>
```

Supprime toutes les variables du tableau `$_SESSION`

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

Attention !

Assurez-vous de ne pas avoir de balise HTML, d'espaces blancs ou du texte envoyé au navigateur avant cet appel de fonction dans votre fichier PHP



Cookies

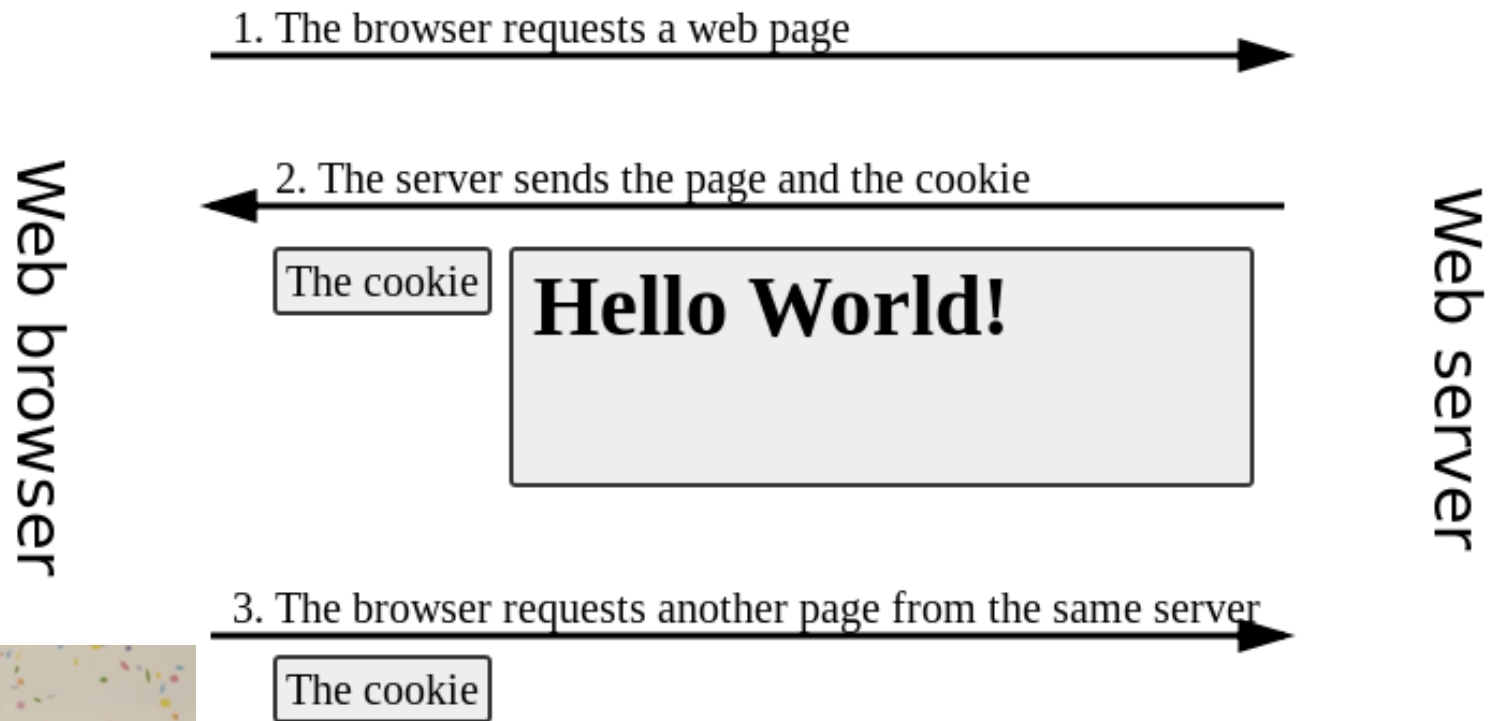
Stockage sur le poste client

Notion de cookie

- Les cookies sont des données stockées dans de petits fichiers de texte sur l'ordinateur du client
- La taille est très limitée (4ko par valeur)
- Un cookie est une paire nom=valeur
- Exemple de cookies:
 - `nomUser="toto "`
 - `mail="toto@gmail.com "`
 - `totalpoints=10`



Principes des cookies



Les Cookies sont stockées sur le client mais transitent aussi avec la page de et vers le serveur

Manipuler des cookies en php

- **Créer** un cookie (sans date d'expiration)

```
setcookie ('nom' , 'toto' ) ;
```


- **Créer** un cookie

```
setcookie ('nom' , 'toto' , time () + 3600 * 24)
```

- **Modifier la valeur d'**un cookie

```
setcookie ('nom' , 'titi' ) ;
```

- **Supprimer** un cookie



Paramètre **expires**
Exprimé en secondes:
Ici 24h

Il suffit de recréer le cookie en définissant
l'attribut *expires* à une date antérieure

```
setcookie ('nom' , 'toto' , time () - 3600)
```

Accéder à la valeur d'un cookie

- **\$_COOKIE** est un Tableau associatif **superglobal** qui contient la description de tous les cookies

\$_COOKIE

idClient	nom
1	toto

```
$_COOKIE["idClient"]="1";  
$_COOKIE["nom"]="Toto";
```

Attention aux types: les cookies sont de type chaîne de caractères:
Il faut les convertir en numérique si nécessaire

Cookies: ce à quoi il faut faire attention!

- Toujours penser à vérifier l'existence d'un cookie avant de tenter d'y accéder:
 - If (isset(\$_COOKIE["nom"]))
- setcookie doit être **la première fonction PHP** de votre script:

A placer avant la balise
<!DOCTYPE>

 - Il ne doit y avoir aucun affichage HTML avant un setcookie.

```
<?php
    if isset($_COOKIE[ "nom"])
        echo "Bonjour " .$_COOKIE['Nom'] ; ?>
<!doctype html>
<html> .....
```