

Lecture 10

Cookies and Sessions

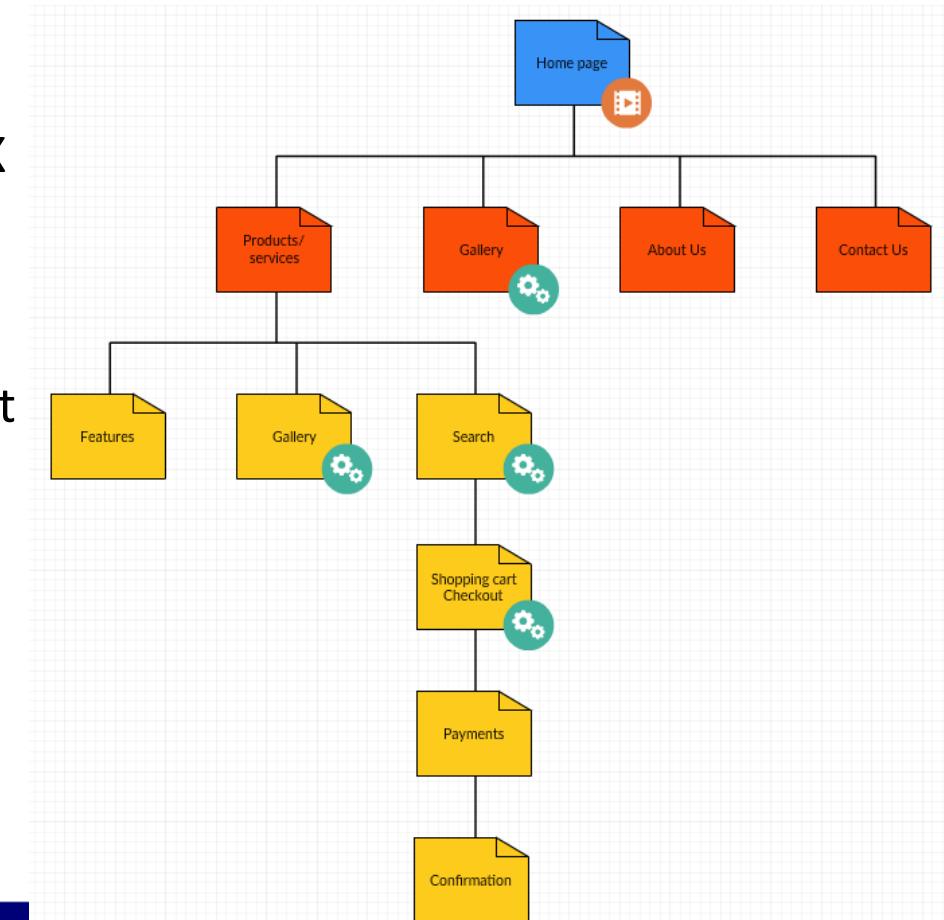
IT1100 Internet and Web technologies

Content

- Introduction to PHP Cookies and Sessions
- Cookies
- Sessions
- Some use of the sessions

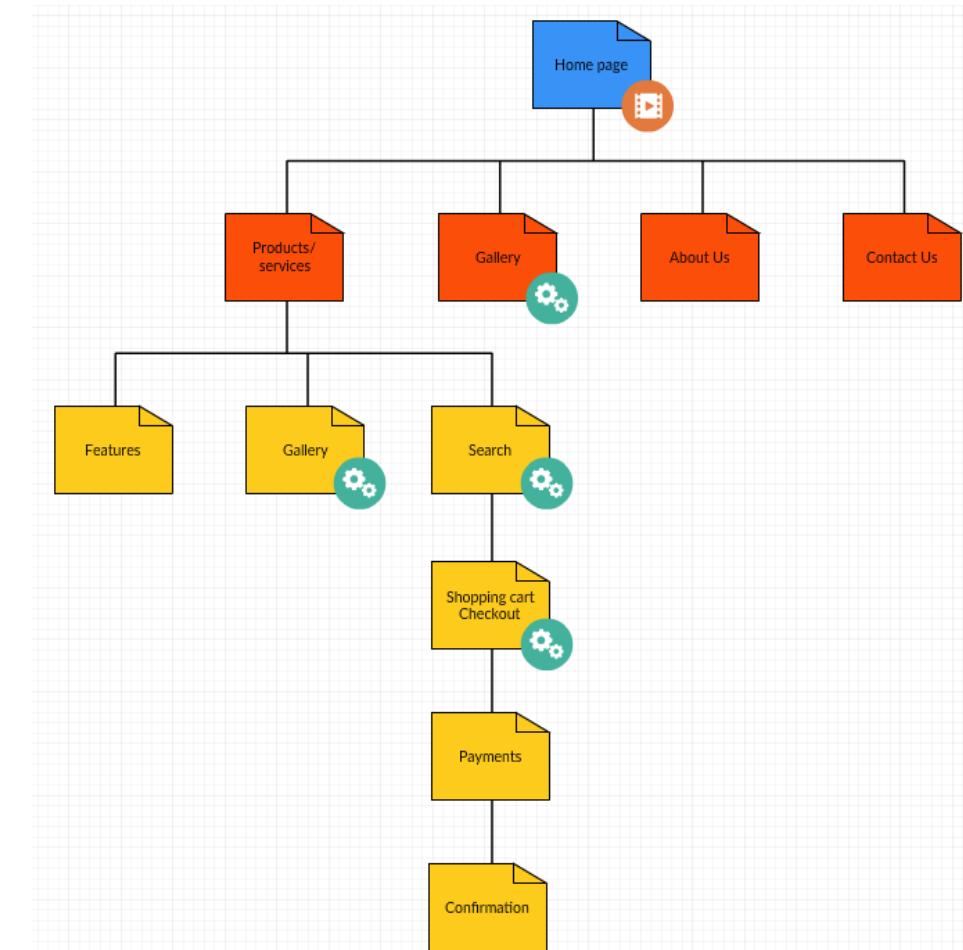
Introduction to PHP Cookies and Sessions

- A web application may contain multiple pages
 - a page may contain multiple features
- A user may navigate through multiple pages and use multiple features, while performing complex transactions, when accessing a web application
 - This may cause many request-response cycles
 - E.g. – Online shopping application -> items pages, shopping cart page, checkout page, etc...



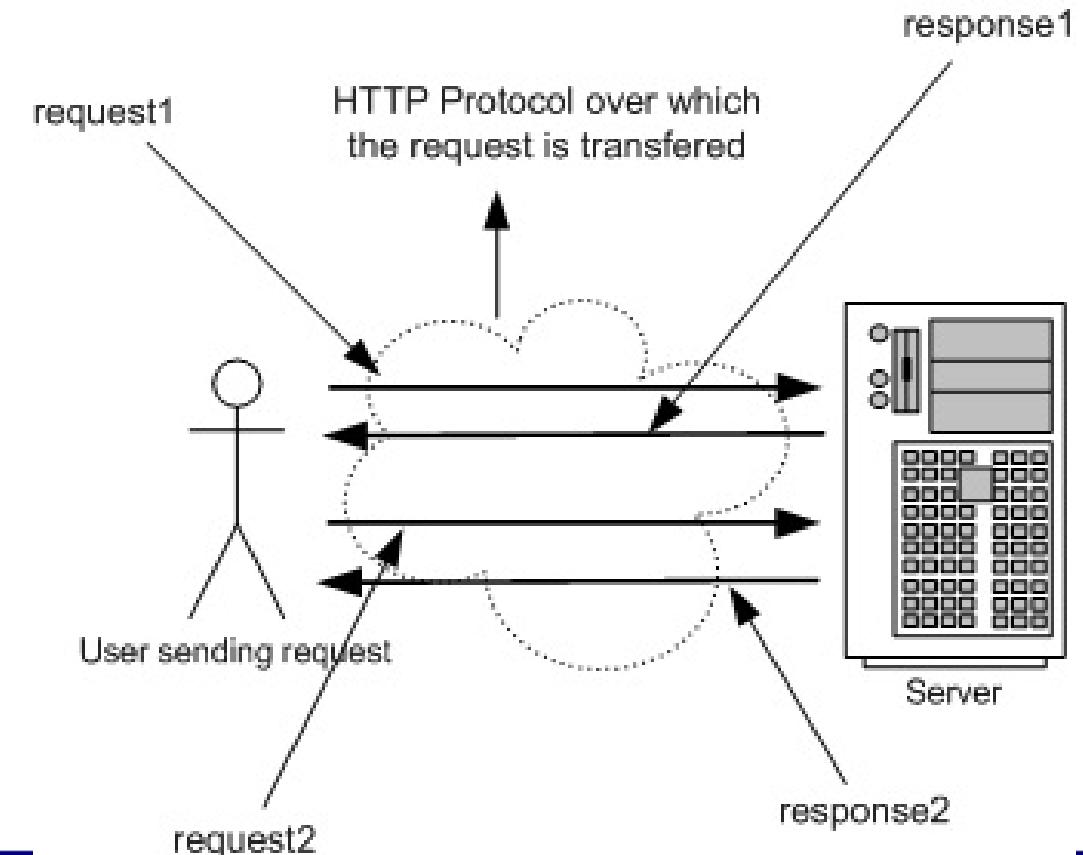
Introduction to PHP Cookies and Sessions

- When the user utilizes a web application to accomplish task(s)/transaction(s), from the point the user **starts using** the application up to the point the user **leaves** the application, it can be seen as a **user session**.
- There can be some common data to be used by the application throughout the user session.
 - E.g. – User details (user name), logging details (authorization), items in the cart, etc...



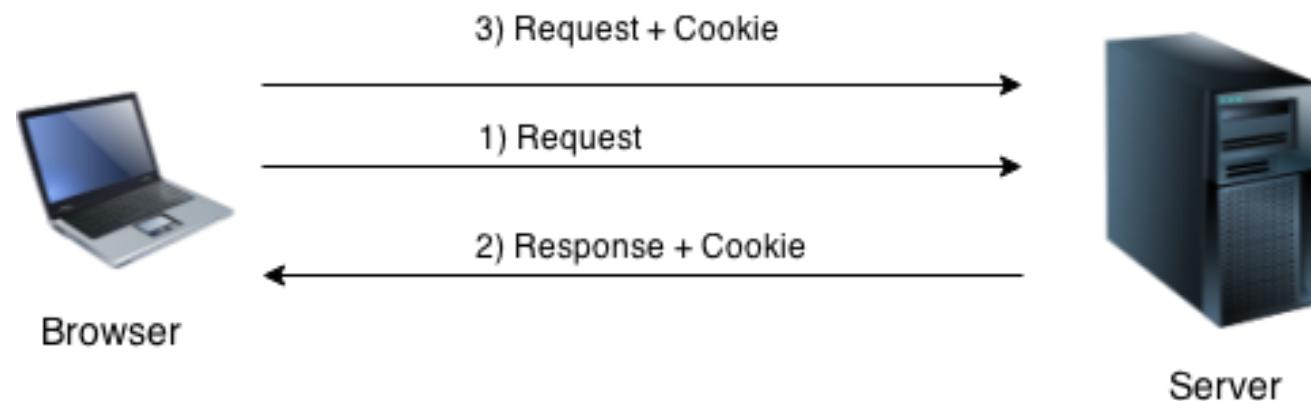
Introduction to PHP Cookies and Sessions

- **HTTP is a stateless protocol**
 - It cannot maintain details/data between multiple requests
 - If need to share data between pages/requests, application level mechanism should be used
- PHP use techniques named **Cookies** and **Sessions**



Cookies - Introduction

- Cookie is a small entry, which is saved in the user's device, by the server
 - Usually managed by the browser in a secured way
- Once a cookie is created/set, it will be sent back to the server with each request
- Cookies can also be processed by JS
 - Because it is saved in the client-side



Cookies - Set a cookie

- A cookies is set as a **name:value** pair with some additional details

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

```
setcookie("Name", "SLIIT", time() + (86400 * 30), "/");
```

Time in seconds
86400 = 1 day

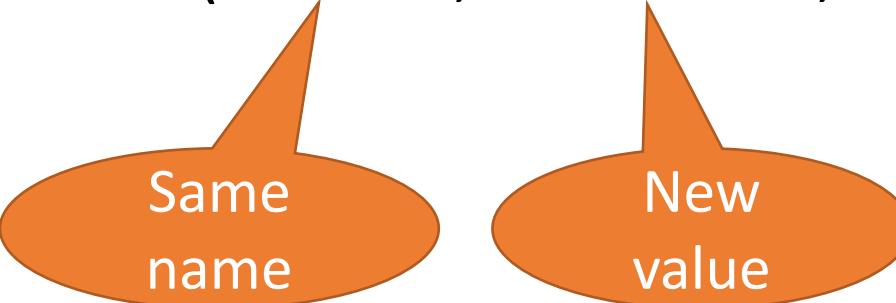
available for the entire
website
(or specify the dir)

Cookies - Set a cookie

- You can see the cookie in the browser.
- Cookie will be expired after the set time
- You can remove the cookie(s) in the browser, using the browser settings

Cookies - Modify a cookie

- To modify a cookie, you may set it again with the new value (and other data)
- `setcookie("Name", "SLIIT", time() + (86400 * 30), "/");`
- `setcookie("Name", "SLIIT FoC", time() + (86400 * 30), "/");`



Same
name

New
value

Cookies - Delete a cookie

- To delete a cookie, modify it with a past expiration time

```
setcookie(" Name ", "", time() - 3600);
```



Cookies- Use cookies

- Cookies are accessible using the `$_COOKIE` super global,
 - use the cookie name for the index

```
echo $_COOKIE["Name"]
```

Cookies - Validate cookies

- You can check if the cookies are enabled for the application by checking the cookie count
 - `count($_COOKIE)`

```
if(count($_COOKIE) > 0) {  
    echo "Cookies are enabled.>";  
}  
  
else {  
    echo "Cookies are disabled.>";  
}
```

Cookies - Validate cookies

- You can check for a specific cookie using the `isset()`

```
if(isset($_COOKIE["Name"])) {  
    echo $_COOKIE["Name"];  
}  
  
else {  
    echo "No such cookie";  
}
```

```
<!DOCTYPE html>
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
// 86400 = 1 day
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

<p><strong>Note:</strong> You might have to reload the page to see  
the value of the cookie.</p>

</body>
</html>
```

Output?

Cookie named 'user' is not set!

Note: You might have to reload the page to see the value of the cookie.



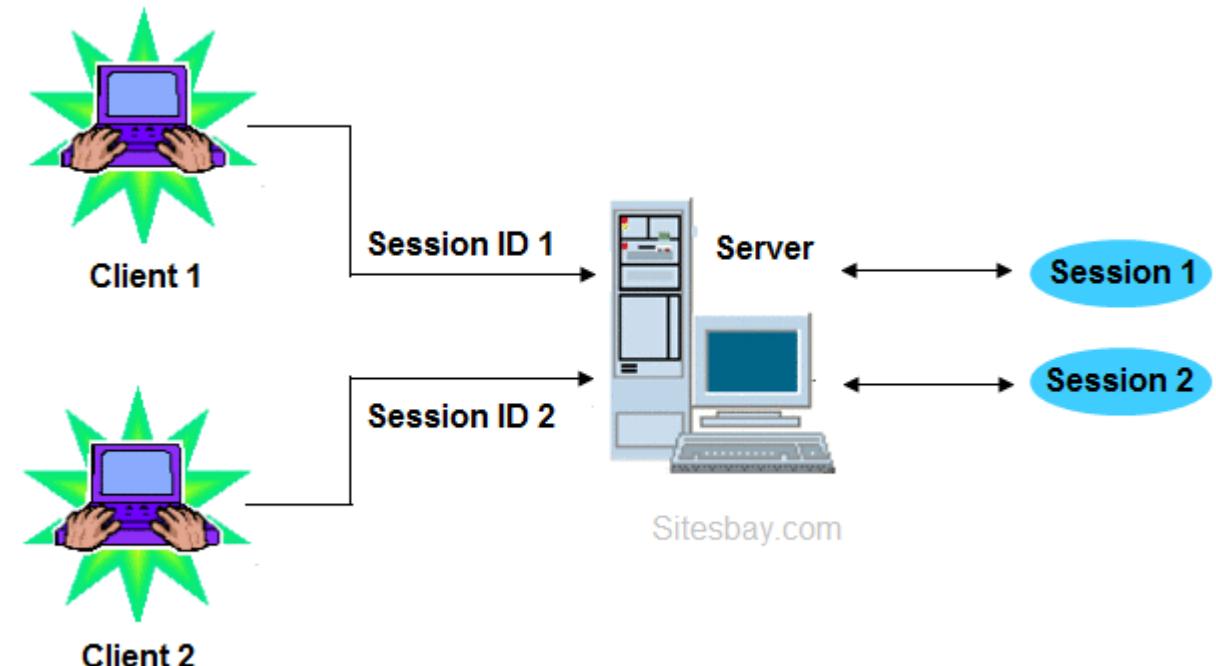
Cookie 'user' is set!

Value is: John Doe

Note: You might have to reload the page to see the value of the cookie.

Sessions - Introduction

- Sessions are stored in the server
- PHP server can identify the user session and maintain the Session variables within a user session
- Unlike cookies, Sessions are discarded when the user leaves the application (closes the browser)



Sessions - Introduction

- When the Session variables are used in an application, you have to start the session on each and every page, which use sessions

//This should be the first line of the page/file

```
session_start();
```

Sessions - Set/modify Session variables

- A Session variable is also a **name:value** pair
 - Use the `$_SESSION` super global variable
 - Name of the Session entry is used as the index

```
$_SESSION["Name"] = "SLIIT";
```

- To modify, you can simply assign the new value

```
$_SESSION["Name"] = "SLIIT FoC";
```

Sessions - Set/modify Session variables

- A Session variable is also a **name:value** pair
 - Use the `$_SESSION` super global variable
 - Name of the Session entry is used as the index

```
$_SESSION["Name"] = "SLIIT";
```

- To modify, you can simply assign the new value

```
$_SESSION["Name"] = "SLIIT FoC";
```

Sessions - Use Session variables

- Session variables can be used in the same way as a regular variable

```
echo $_SESSION["Name"];
```

Sessions - End Session variables

- Session variables can be explicitly discarded by the application

```
// remove all session variables
```

```
session_unset();
```

```
// destroy the session
```

```
session_destroy();
```

E.g. :When the user logs off from the application

Sessions - Validate Session variables

- isset() can be used to check if the Session variable is available

```
if(isset($_SESSION["Name"])) {  
    echo $_SESSION["Name"];  
}  
  
else {  
    echo "No such Session";  
}
```

Sessions - Validate Session variables

- empty() can be used to check if the Session variable contains a value

```
if(!empty($_SESSION["Name"])) {  
    echo $_SESSION["Name"];  
}  
  
else {  
    echo "Session has no value";  
}
```

```
<?php session_start();  
  
if( isset( $_SESSION['counter'] ) ) {  
  
    $_SESSION['counter'] += 1;  
  
} else {  
  
    $_SESSION['counter'] = 1;  
  
}  
  
$msg = "You have visited this page ". $_SESSION['counter']; $msg .= "in this session."  
  
?>  
  
<html>  
  
<head> <title>  
  
Setting up a PHP session</title>  
  
</head>  
  
<body>  
  
<?php echo ( $msg );  
  
?>  
  
</body>  
  
</html>
```

Some use of the sessions

- The common uses of Sessions are
 - To maintain the user details (If the user is logged in or not)
 - To store the items in a shopping cart

User log in – index.php

```
<html>
<head>
</head>
<body>
<h1>Log in</h1>
<form method="post" action="index.php">
    Username: <input type="text" name="txtName" /><br>
    Password: <input type="password" name="txtPass" /><br>
    <input type="submit" />
</form>
</body>
</html>
```

User log in – index.php

- If the request has the `$_POST["txtName"]`, then it is the log in form submission

```
if(isset($_POST["txtName"]))
{
    //Validate the user
    //If a validuser, set a Session
}
```

User log in – index.php

- User can be validated against the data in a DB
 - The sample code below validates the user against some static data just for demonstration

```
if($_POST["txtName"]=="asd" && $_POST["txtPass"]=="123")  
{  
    //Valid user, so set the Session  
    $_SESSION["userName"] = $_POST["txtName"];  
}
```

User log in – index.php

- You may also check, if this is an already logged user.
 - We do not want to show a log in page to an already logged in user

```
if(isset($_SESSION["userName"]))  
{  
    //Redirect to another page  
    header("Location:home.php");  
}
```

User log in – index.php

```
<?php //TOP OF THE PAGE  
session_start();  
  
if(isset($_POST["txtName"])){  
    if($_POST["txtName"]=="asd" && $_POST["txtPass"]=="123") {  
        $_SESSION["userUserName"] = $_POST["txtName"];  
    }  
}  
  
if(isset($_SESSION["userUserName"])) {  
    header("Location:home.php");  
}  
?>
```

User log in – Other pages (home.php)

- We can identify if the user is an already logged in user or not by checking the Session.

```
<?php  
session_start();  
  
$userName = "";  
if(isset($_SESSION["userName"])) { //Already logged in  
    $userName = $_SESSION["userName"]; //Use the session value  
}  
else { // Not logged in  
    header("Location:index.php"); //Redirect to the login page  
}  
?>
```

User log in – Other pages (home.php)

- Other pages may have a log out feature

```
<html>
<head>
</head>
<body>
    <h1>Hello <?php echo $userName; ?></h1>
    <form method="post" action="logoff.php">
        <input name="logoff" type="submit" value="Log off"/>
    </form>
</body>
</html>
```

User log off – logoff.php

- Check if the request is coming from a logoff feature
 - If coming from a log off feature, then end the Session and redirect to log in page

```
if(isset($_POST["logoff"]))  
{  
    session_destroy();  
    header("Location:index.php");  
}
```

User log off – logoff.php

- If someone is trying to visit the page directly, we had better redirect to a proper page

```
else
{
    header("Location:home.php");
}
```

User log off – logoff.php

```
<?php  
session_start();  
  
if(isset($_POST["logoff"])) {  
    session_destroy();  
    header("Location:index.php");  
}  
else {  
    header("Location:home.php");  
}  
?>
```

Summary

- Introduction to PHP Cookies and Sessions
- Cookies
- Sessions
- Some use of the sessions