Cookies and Sessions

The need for **persistence**

- Consider these examples
  - Counting the number of “**hits**” on a website
  - *i.e.* how many times does a client load your web page source
  - The questionnaire on computing experience
- Somehow your **.php** needs to remember previous instances of it being requested by a client

Persistence

- Persistence is the ability of data to **outlive** the execution of the program that created them.

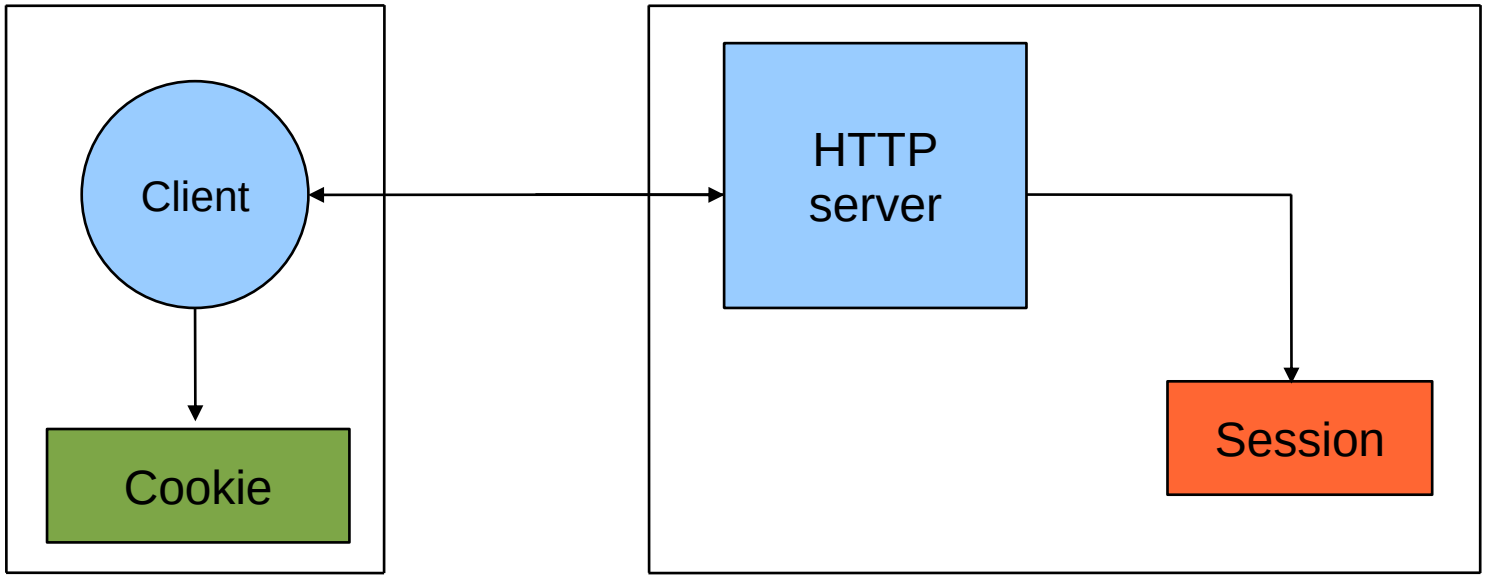
- An obvious way of achieving persistence is to simply save the data in a file

Persistence and HTTP

Recall http is a stateless protocol. It remembers nothing about previous transfers

Two ways to achieve persistence:

- PHP cookies
- PHP sessions



HTTP Cookies

In internet programming, a cookie is a packet of information sent from the server to client, and then sent back to the server each time it is accessed by the client.

Introduces state into HTTP (remember: **HTTP is stateless**)

Cookies are **transferred** between **server** and **client** according to **http**.

PHP supports http cookies

Cookies can also be thought of as tickets used to identify clients and their orders

How Cookies are implemented

- Cookies are sent from the server to the client via “Set-Cookie” headers

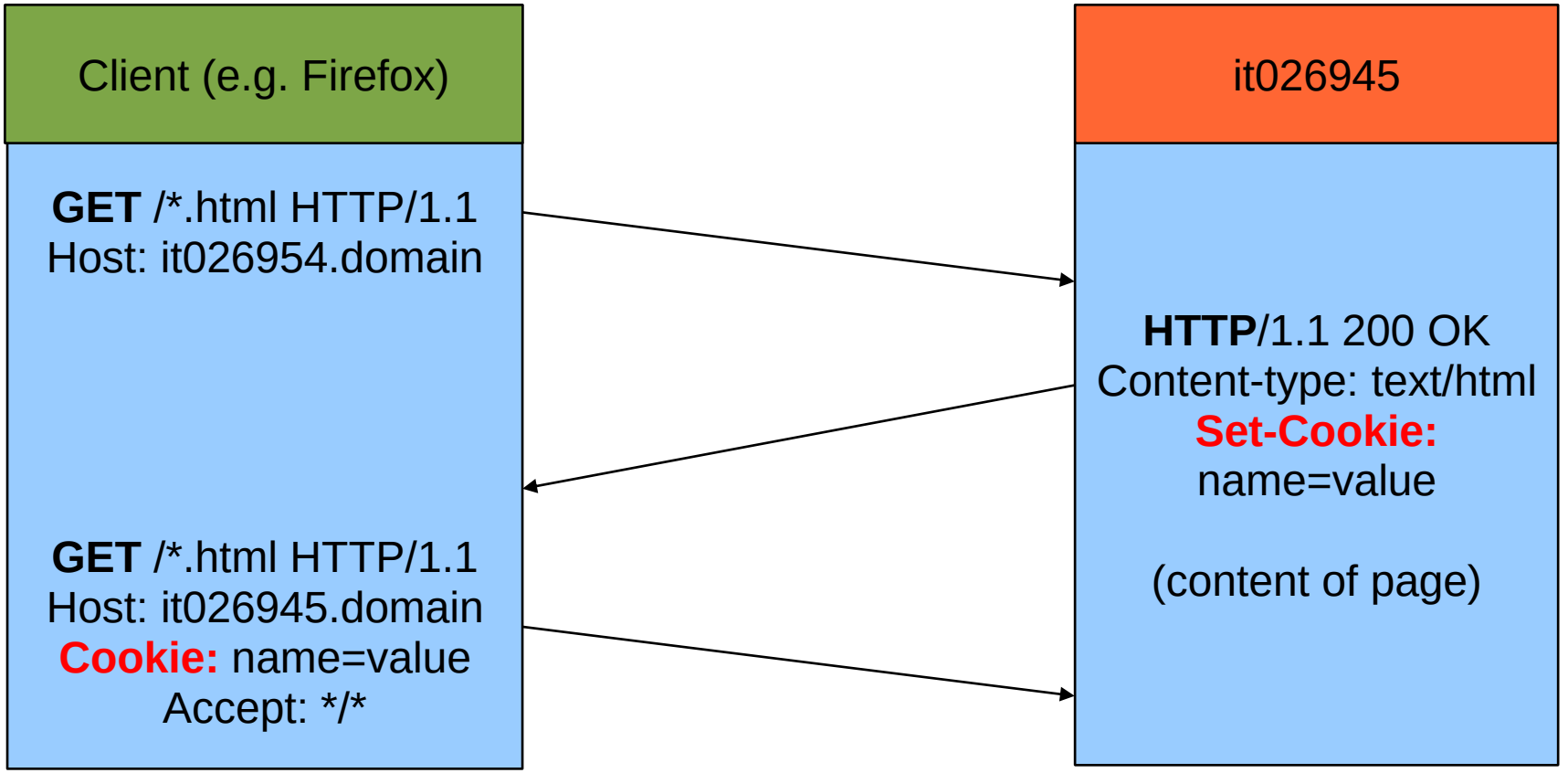
Set-Cookie: NAME=VALUE; expires=DATE; path=PATH; domain=DOMAIN_NAME; secure

- The **NAME** value is a URL-encoded name that identifies the cookie.
- The **PATH** and **DOMAIN** specify where the cookie applies

setcookie(name,value,expire,path,domain,secure)

Parameter	Description
name	(Required). Specifies the name of the cookie
value	(Required). Specifies the value of the cookie
expire	(Optional). Specifies when the cookie expires. e.g. **time()+3600*24*30** will set the cookie to expire in **30 days**. If this parameter is not set, the cookie will expire at the end of the session (when the browser closes).
path	(Optional). Specifies the server path of the cookie. If set to "**/**", the cookie will be available within the entire domain. If set to "**/phptest/**", the cookie will only be available within the test directory and all sub-directories of **phptest**. The default value is the current directory that the cookie is being set in.
domain	(Optional). Specifies the domain name of the cookie. To make the cookie available on all subdomains of example.com then you'd set it to ".example.com". Setting it to www.example.com will make the cookie only available in the www subdomain
secure	(Optional). Specifies whether or not the cookie should only be transmitted over a secure **HTTPS** connection. **TRUE** indicates that the cookie will only be set if a secure connection exists. Default is **FALSE**.

Cookies from HTTP



Creating PHP cookies

Cookies can be set by directly manipulating the HTTP header using the PHP header() function

```
<?php
  header("Set-Cookie: mycookie=myvalue; path=/; domain=.coggeshall.org");
?>
```

Creating cookies with **setcookie()**

Use the PHP setcookie() function:

setcookie (name, value, expire, path, domain, secure)

e.g.

```
<?php
  setcookie("MyCookie",      $value, time()+3600*24);
  setcookie("AnotherCookie", $value, time()+3600);
?>
```

- Name: name of the file
- Value: data stored in the file
- Expire: data string defining the life time
- Path: subset of URLs in a domain where it is valid
- Domain: domain for which the cookie is valid
- Secure: set to '1' to transmit in HTTPS

Reading cookies

To access a cookie received from a client, use the PHP **$_COOKIE** superglobal array

```
<?php

  foreach ($_COOKIE as $key=>$val) {
    print $key . " => " . $val . "<br/>";
  }

?>
```

Each key in the array represents a cookie - the key name is the cookie name.

Creating and using cookies example

```php
<?php
   setcookie("MyCookie",      $value, time()+7200);
   setcookie("AnotherCookie", $value, time()+7);
?>

<?php
   foreach ($_COOKIE as $key=>$val) {
     print $key . " => " . $val . "<br/>";
   }
?>
```

- Cookies only become visible on the next page load

Using headers

setcookie() did not run before information was sent to the browser...

Cookies have to be sent ***before*** the heading elements

Using headers

```
<?php
$strValue = "This is my first cookie";
setcookie ("mycookie", $strValue);
echo "Cookie set<br>";
?>
```

```
<!DOCTYPE html PUBLIC "=//W3C//DTD XHMTL 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
  <html xmlns="http://www.w3.org/1999/xhmtl" xml:lang="en">
<head><title>PHP Script using Cookies</title>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
</head>
<body>
   <?php
      echo “<p> A cookie has been set. </p>”;
   ?>
</body>
</html>
```

This is the correct approach!

Deleting a cookie

- Set the cookie with its name only:

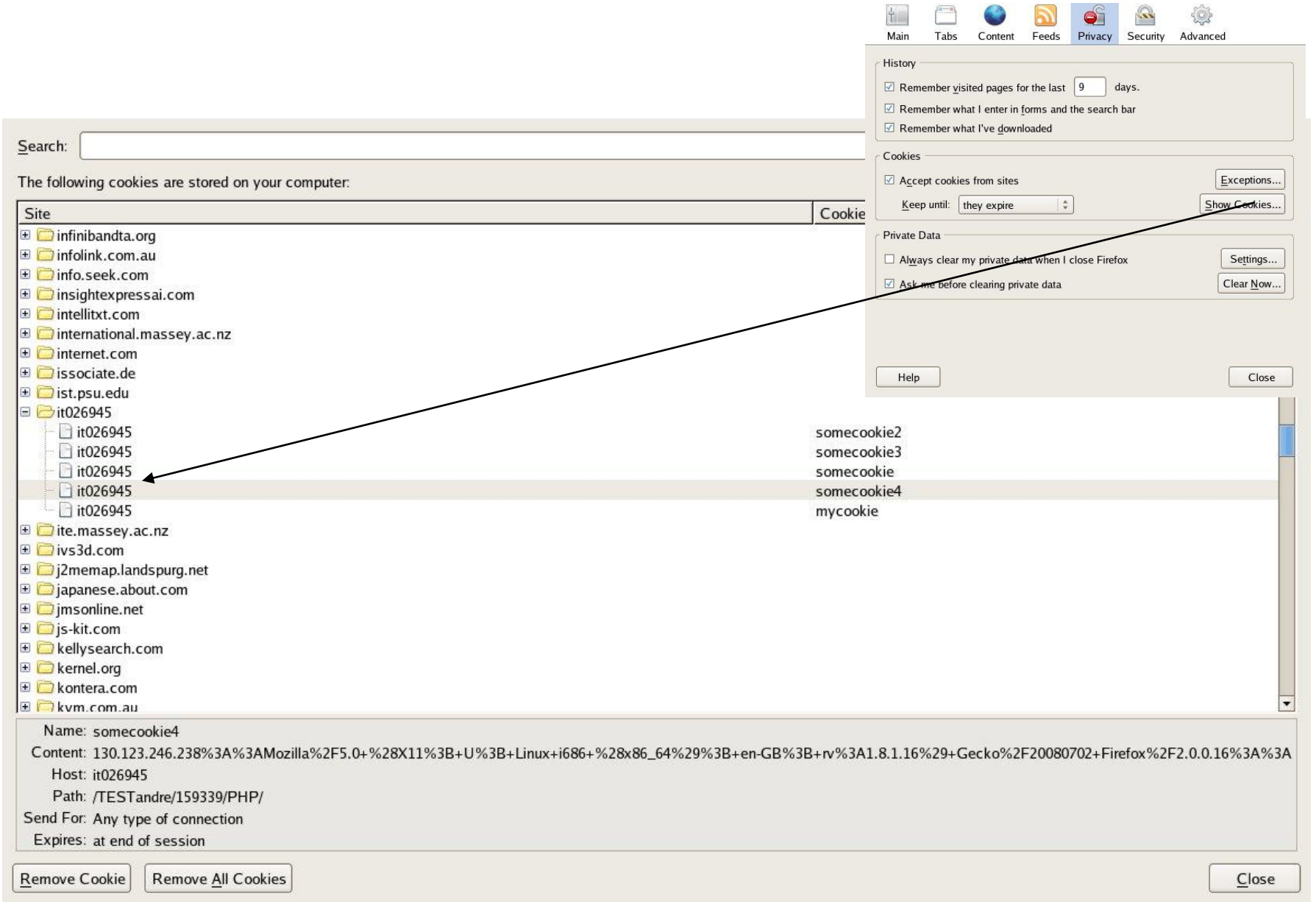
```
setcookie(“mycookie”);
```

Multiple data items

- Use explode() e.g.

```
<?php
$strAddress = $_SERVER['REMOTE_ADDR'];
$strBrowser = $_SERVER['HTTP_USER_AGENT'];
$strOperatingSystem = $_ENV['OS'];
$strInfo = "$strAddress::$strBrowser::$strOperatingSystem";
setcookie ("somecookie4",$strInfo, time()+7200);
?>
<?php
$strReadCookie = $_COOKIE["somecookie4"];
$arrListOfStrings = explode ("::", $strReadCookie);
echo "<p>$strInfo</p>";
echo "<p>Your IP address is: $arrListOfStrings[0] </p>";
echo "<p>Client Browser is: $arrListOfStrings[1] </p>";
echo "<p>Your OS is: $arrListOfStrings[2] </p>";
?>
```

Where is the cookie stored?



Where is the cookie stored

- Depends on the browser...
- e.g., firefox/mozilla under /home/a_____
  - Look for cookies.txt in .mozilla directory
  - Usually under:
    - /home/a_____/.mozilla/firefox/asdkfljy.default
  - Cookie is stored only if there is an expiry date
  - Otherwise it is deleted when leaving browser
  - ***Persistent*** only if an *expiry date is set*

Problems

- User can disable cookies in the browser
- Cookies may be viewed by other users
- Can only store 20 cookies; max 4KB.
- Some browsers may display incorrectly unless all options are set in setcookie() (eg expiration time, path)
- C:\documents and settings \jceddia \cookies \jceddia

  @phpbuilder[1].txt

PHP Sessions

Difference between Session and Cookies

Cookies	Sessions
Limited storage space	Practically unlimited space
Insecure storage client-side	Reasonably securely stored server-side
User controlled	No user control

PHP Sessions

You can store user information (e.g. username, items selected, etc.) in the **server side** for later use using PHP session.

Sessions work by creating a unique id (**UID**) for each visitor and storing variables based on this **UID**.

The **UID** is either stored in a **cookie** or is **propagated in the URL**.

When should you use sessions?

- Need for data to stored on the server
- Unique session information for each user
- Transient data, only relevant for short time
- Data does not contain secret information
- Similar to Cookies, but it is stored on the server
- More secure, once established, no data is sent back and forth between the machines
- Works even if cookies are disabled
- Example: we want to count the number of "**hits**" on our web page.

Before you can store user information in your PHP session, you must first start up the session.

session_start() function must appear BEFORE the **<html>** tag.

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

<html>
<body>

</body>
</html>
```

PHP Sessions

- Starting a PHP session:

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

- This tells PHP that a session is requested.
- A **session ID** is then **allocated** at the server end.

- **session ID** looks like:

sess_f12347812374681237 68asjkhfa7891234g

Session variables

- $_SESSION
- e.g., $_SESSION[“intVar”] = 10;

- Testing if a session variable has been set:

```
session_start();
if(!$_SESSION['intVar']) {...} //intVar is set or not
```

Registering session variables

- Instead of setting superglobals, one can register one's own session variables

```
<?php
    $barney = "A big purple dinosaur.";
    $myvar_name = "barney";
    session_register($myvar_name);
?>
```

- $barney can now be accessed "globally" from session to session
- This only works if the **register_globals** directive is enabled in **php.ini** - nowadays this is turned off by default

Use of session_register() is deprecated!

Make your own session variables

- With **session_start()** a default session variable is created - the name extracted from the page name
- To create your own session variable just add a new key to the **$_SESSION superglobal**

$_SESSION['dug'] = "a talking dog.";

Use of $_SESSION is preferred, as of PHP 4.1.0.

Session Example 1

```
<?php
        session_start();
        if (!isset($_SESSION["intVar"]) ){
              $_SESSION["intVar"] = 1;
        } else {
              $_SESSION["intVar"]++;
        }
        echo "<p>In this session you have accessed this
          page " .    $_SESSION["intVar"] . "times.</p>";
?>
```

Session Example 2

```
<?php session_start();?>
<?php
$thisPage = $_SERVER['PHP_SELF'];

$pageNameArray = explode('/', $thisPage);
$pageName = $pageNameArray[count($pageNameArray) - 1];
print "The name of this page is: $pageName<br/>";

$nameItems = explode('.', $pageName);
$sessionName = $nameItems[0];
print "The session name is $sessionName<br/>";

if (!isset($_SESSION[$sessionName])) {
        $_SESSION[$sessionName] = 0;
        print "This is the first time you have visited this page<br/>";
}
else {
        $_SESSION[$sessionName]++;
}
print "<h1>You have visited this page " . $_SESSION[$sessionName] .
   " times</h1>";
?>
```

Ending sessions

unset(**$_SESSION**[‘**name**’])

- Remove a session variable

session_destroy()

- Destroys all data registered to a session
- does not unset session global variables and cookies associated with the session
- Not normally done - leave to timeout

Destroying a session completely

```php
<?php
// Initialize the session.
// If you are using session_name("something"), don't forget it now!
session_start();

// Unset all of the session variables.
$_SESSION = array();

// If it's desired to kill the session, also delete the session cookie.
// Note: This will destroy the session, and not just the session data!
if (ini_get("session.use_cookies")) { // Returns the value of the configuration option
    $params = session_get_cookie_params();
    setcookie(session_name(), '', time() - 42000,
        $params["path"], $params["domain"],
        $params["secure"], $params["httponly"]
    );
}

// Finally, destroy the session.
session_destroy();
?>
```

returns the name of the current session

http://nz2.php.net/manual/en/function.session-destroy.php

Session Example 3

```
<?php
session_start();

if(!isset($_SESSION['strColourBg']))   $_SESSION['strColourBg'] = "red";
else echo "Currently Bg set to " . $_SESSION['strColourBg'] . "<br>";
if(!isset($_SESSION['strColourFg'])) $_SESSION['strColourFg'] = "yellow";
else echo "Currently Fg set to " . $_SESSION['strColourFg'];

if(isset($_POST["submit"]) ) {
  $strColourBg = $_POST["strNewBg"];
  $strColourFg = $_POST["strNewFg"];
  $_SESSION['strColourBg'] = $strColourBg;
  $_SESSION['strColourFg'] = $strColourFg;
  echo "<br>New Settings";
}
else {
  $strColourBg = $_SESSION['strColourBg'];
  $strColourFg = $_SESSION['strColourFg'];
  echo "<br>Keep old settings";
}
?>
```

Session Example 3 (cont.)

```
<head> <style type="text/css">
body {background-color: <?php echo $strColourBg ?>}
p {color: <?php echo $strColourFg?>}
h2 {color: <?php echo $strColourFg?>}
</style></head>

<body>
<h2>h2 colour</h2>
<form action = '<?php echo $SERVER["PHP_SELF"] ?>' method='post'>
<label for="strNewBg"> Background colour: </label>
<select name='strNewBg' id='strNewBg'>
 <option>red</option> ...  <option>grey</option>
</select>
<label for="strNewFg"> Text colour: </label>
<select name='strNewFg' id='strNewFg'>
 <option>yellow</option> ... <option>grey</option>
</select>
<input type='submit' name='submit'/>
</form></body>
```

(adapted from Stobart & Parsons, 2008)

Summary

PHP sessions and cookies are **mechanisms for introducing state** into HTTP transactions.

References

- ***https://www.w3schools.com/php/php_cookies.asp***
- ***https://www.pontikis.net/blog/create-cookies-php-javascript***
- ***https://www.tutorialrepublic.com/php-tutorial/php-cookies.php***
- ***https://www.tutorialspoint.com/php/php_sessions.htm***
- ***https://www.w3schools.com/pHp/php_sessions.asp***
- ***https://www.tutorialrepublic.com/php-tutorial/php-sessions.php***