# Project Tutorial 5
Cookies and sessions

Dr. Yuzuko Nakamura
Dept. of Computer Science
Y.Nakamura@ucl.ac.uk

## HTTP is a stateless protocol

- Sequence of unconnected requests to the server (and the server's reply).
- PHP + your database allow serving up dynamic content, but without any state, it is hard to serve up private data.
  - Need to know /remember that a particular user already authenticated in the past.
  - Can't be done by sending GET/POST objects as any user can do that.

## User accounts requires tracking state

- Tracking a little bit of state is useful for:
  - Keeping track of user's login/logout status
  - Only loading user-specific data when authentication has happened in the past.
  - Keeping track of decisions user has made in their visit to your website (shopping cart, recent searches, recently-viewed pages, etc.)
- Must be done in application code (e.g. PHP).

## Cookies

- A cookie is a piece of data stored by the browser on a user’s machine.
  - An application creates a cookie on the server.
  - Sends it to the browser, which stores it.
  - Browser sends the cookie back to the server when a request is made.
    - But only to the server the cookie came from.
- Provides a ‘token’ that can identify a user as each request is made.

## Cookie components

- Three parts: name, value (a string), and expiration date.
- The expiration date determines how long a cookie is stored by a browser.
  - Deleted when date is passed.
  - If no date given, deleted when browser is closed.
- A cookie can persist throughout a user login.
  - And across multiple logins.

## Cookies and PHP

- setcookie function makes it easy to set up a cookie:

```
setcookie('userID', '10'); // Expire cookie when
browser closed.
setcookie('username', 'dilbert', time() + (60 *
60 * 24 * 30)); // Expire in thirty days time.
setcookie('username', '', time() - 3600); //
Expire a cookie now i.e. delete cookie.
```

- Access cookie contents using $_COOKIE superglobal:

```
echo 'You are logged in as ‘ .
$_COOKIE['username'];
```

## Using cookies

- Now have the ability to connect a request to a user.
- At login create a cookie after checking username and password.
  - Typically create user id and username cookies.
  - But not for the password!
- For each request, get the cookie to identify the user.
  - Also check if user is allowed to access the requested page.
- Expiring a cookie disconnects the user (logout).

# Logging in with a cookie

```
$query = "SELECT userID, username FROM User " .
   WHERE username = '$username' AND " .
   "password = SHA('$password')";
$data = mysqli_query($connection, $query);
if (mysqli_num_rows($data) == 1) {
  $row = mysqli_fetch_array($data);
  setcookie('userID', $row['userID']);
  setcookie('username', $row['username']);
  $indexURL = 'http://' . $_SERVER['HTTP_HOST'] .
    dirname($_SERVER['PHP_SELF']) . '/index.php';
  header('Location: ' . $indexURL);
} else {
  echo 'Invalid username or password, try again';
}
```

Make SQL query to verify login

Set cookies if we got something

Redirect to index

Login failed

## Using cookie to check what content to display

```
if (isset($_COOKIE['userID']) &&
    isset($_COOKIE['username'])) {
  // User is identified; display logged-in
  // or user-specific content, make further DB
  // calls, etc.
}
else {
  // User not identified; display generic
  // content, error message, login prompt, etc.
}
```

## Logging out with a cookie

- Expire the cookies:

```
if (isset($_COOKIE['userID'])) {
  setcookie('userID', '', time()-3600);
  setcookie('username', '', time()-3600);
  $indexURL = 'http://' . $_SERVER['HTTP_HOST'] .
    dirname($_SERVER['PHP_SELF']) . '/index.php';
  header('Location: ' . $home_url);
}
```

## Notes about cookies

- Possible to fake or view a cookie on the client.
- Best not to store sensitive data in a cookie.

## Cookie Law

- Privacy legislation requiring a website to get consent from user to store and retrieve information related to that user.
  - EU Directive implemented via Privacy and Electronic Communications Regulations in UK.
- Your responsibilities:
  - Tell site visitors how cookies are being used.
  - Obtain user’s consent to continue using cookies.
  - Do not use cookies if user does not consent.
  - Applies to any storage of user data regardless of the technology used (not just cookies).
  - Do not have to do this for essential cookies.
  - See http://cookiepedia.co.uk/all-about-cookies

## Sessions

## Cookies vs. sessions

- Cookies stored on the client side; session data stored on the server side.
  - Server has more control over storage of session data.
  - Data is more secret.
- Sessions automatically end when browser is closed.
- Can use cookies in addition to sessions.

## Sessions and PHP

- To start a session or join an existing session use:
  `session_start();`
  - Behind the scenes this creates a session ID.
    - A string like ‘tksf820j9hq7f9t7vdt5o1ceb2’
    - This gets added to requests and replies.
- A session is closed using:
  `session_destroy();`
  - Or when the user closes their browser.

## Passing around session ID

- Session ID is added to HTTP requests and replies.
- How?
  - With cookies (again) if cookies are enabled.
    - Stored as a cookie with session ID as the name and empty string value.
  - As a request parameter if cookies are disabled (server handles this work).
- If cookies are enabled, need to also delete the cookie when ending the session:

```
if (isset($_COOKIE[session_name()])) {
  setcookie(session_name(), '', time()-3600);
}
```

## Logging in with a session

```
session_start(); // Must not forget this!

if (!isset($_SESSION['userID'])) {
  // Not already logged in
  // Insert code to check the username & 
password
  // Things you can also do:
  $_SESSION['userID'] = $row['id'];
  $_SESSION['username'] = $row['username'];
  setcookie('userID', $row['id'], time() + 
(60 * 60 * 24 * 30));
  setcookie('username', $row['username'], 
time() + (60 * 60 * 24 * 30));
}
```

Track variables for this session

Cookies can also be used in conjunction with sessions

## Using sessions to choose what content to display

```
session_start();

if (isset($_SESSION['userID'])) {
   // User recognised.
   // Display content, create/update session
variables.
}
else {
   // User not identified; display generic
   // content, error message, login prompt, etc.
}
```

Important! Need this in every file before you can access session variables.

# Example: showing user they are logged in

```
<?php
session_start();
function isLoggedIn() { return isset($_SESSION['userID']); }

if (isLoggedIn())
  echo 'You are logged in as "' . $_SESSION['username']. '".';
?>
<ul>
  <?php
    if (!isLoggedIn()) {
      echo '<li><a href="login.php">Login</a></li>';
      echo '<li><a href="registration.php">Register</a></li>';
    } else {
      echo '<li><a href="logout.php">Logout</a></li>';
    }
  ?>
</ul>
```

## Logging out with a session

```
session_start();
if (isset($_SESSION['userID'])) {
  $_SESSION = array();
  if (isset($_COOKIE[session_name()])) {
    setcookie(session_name(), '', time() - 3600);
  }
  session_destroy();
}

setcookie('userID', '', time() - 3600);
setcookie('username', '', time() - 3600);

$indexURL = 'http://' . $_SERVER['HTTP_HOST'] .
     dirname($_SERVER['PHP_SELF']) . 'index.php';
header('Location: ' . $indexURL);
```

Clean up extra cookies stored (if any)

Redirect to index

## Summary

- You can use cookies and sessions to support user accounts and persistent logins.

## TODOs this week:

- Incorporate feedback on ER diagram in your database (if applicable).
- Full speed ahead on the project: You have enough information to implement core features:
  - Registration
  - Auction creation
  - Search
  - Bidding