WEB DEVELOPMENT WITH PHP

COURSE #07

fasttrackit
CLICK TO RESTART

LET'S RECAP

COURSE 6 WAS ABOUT:

- Superglobal variables
- HTML forms & POST
- Introduction to databases & MySQL
- phpMyAdmin

TOPICS

COURSE #07

1. Cookies
2. Session variables
3. MySQL - SELECT statement
4. PHP + MySQL
5. Homework

1. COOKIES

LET'S IMAGINE A SIMPLE EXAMPLE

RETURNING VISITORS

Let's say you visit a page. If it's the first time you see the page, it will show a welcome popup.

When you visit it again (after 1 minute or after 1 year), it will not display the welcome popup.

WHO ARE THE VISITORS

PHP needs a way to uniquely identify visitors.

But who are they?

TAGGING THE VISITORS

OK, SO A BROWSER IS A VISITOR

To know if the same browser has requested a particular page, we need a way to "tag" it, so we can recognize it when we see it again.

We do this by putting a small bit of information on it and on the next visit - we see that he already has the information.

BROWSER COOKIES

All browsers have a storage area, where they can keep information.

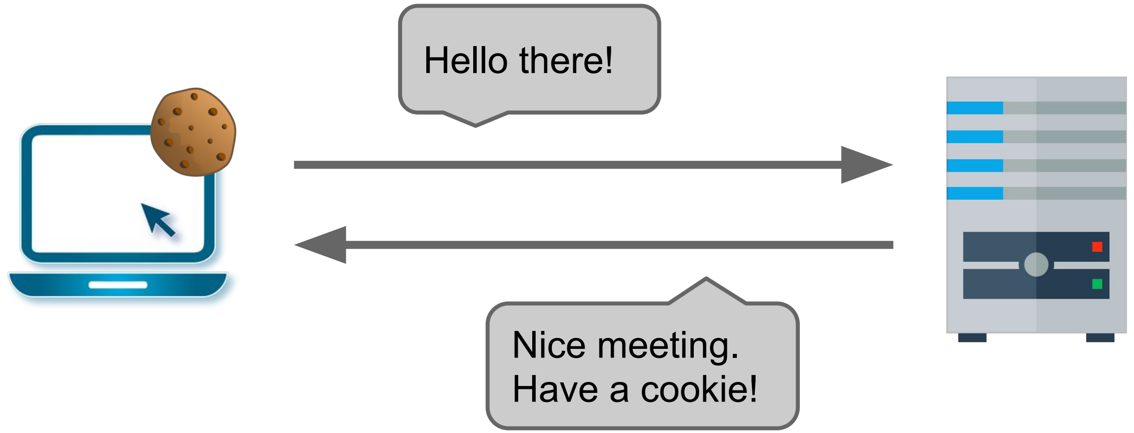
These bits of information are called .. cookies.

BROWSER COOKIES

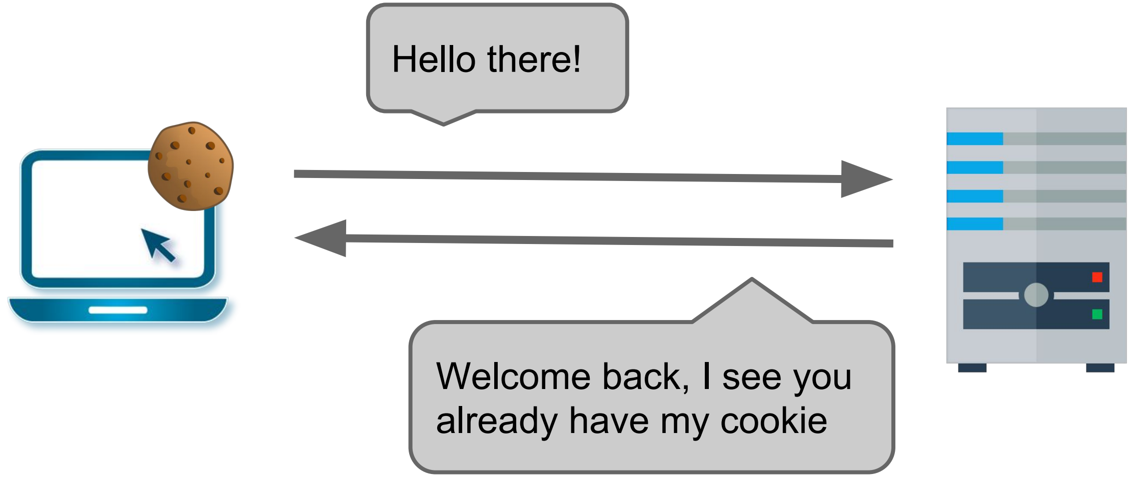
HOW DO COOKIES WORK?

- ✓ When a browser requests a site, it creates a "pocket" only for that site, where it can store the cookies
- ✓ Both the browser and the site can put cookies there
- ✓ Information stored in a pocket is only seen by the browser and the site for which the pocket was created
- ✓ Like all food, cookies have an expiration date.

COOKIES IN ACTION



COOKIES IN ACTION



CODING TIME

LET'S SAVE A COOKIE

Create a new php file called cookie.php

```
<?php

setcookie('friend', 'yes', time() + 3600);

?>
```

Did it save it? Check in your browser.

THE setcookie() FUNCTION

PARAMETERS

A cookie is like a variable. It has a <u>name</u> and a <u>value</u>.

```
<?php

setcookie('friend', 'yes', time() + 3600);

?>
```

In addition to variables, it also has an <u>expiration date</u>.

Check the documentation for more parameters.

UNIX TIME (EPOCH TIME)

A system for representing time, defined as <u>the number of seconds that have elapsed since 1 January 1970</u>, 00:00:00 UTC (Coordinated Universal Time).

THE setcookie() FUNCTION

HOW TO SET THE EXPIRATION DATE

We need to tell the browser that the cookie should expire after a certain number of seconds since now. To do this, we first determine the number of seconds that have passed until now and add the number of seconds that should pass from now.

```php
<?php
setcookie('friend', 'yes', time() + 3600);
?>
```

TIPS & TRICKS

ATTENTION

setcookie() function must be called before any HTML tag or writing / output (echo, print)

WORKS

```
<?php
setcookie(..);
echo '<h1>Page title</h1>';
?>
```

DOESN'T WORK

```
<?php
echo '<h1>Hello</h1>';
setcookie(..);
?>
```

DOESN'T WORK

```
<h1>Hello</h1>
<?php
setcookie(..);
?>
```

CODING TIME

READING A COOKIE

Reading the cookie is done through the $_COOKIE global variable.

The value we saved earlier is available as $_COOKIE['friend']

```
<?php

echo $_COOKIE['friend'];

?>
```

CODING TIME

DELETING A COOKIE

To delete a cookie, we use the same function we used to create it and set it to expire at a past date.

Yes, not an ideal method, but what can we do .. :)

```
<?php

setcookie('friend', null, time() - 3600);

?>
```

CODING TIME

LET'S DO AN EXERCISE

Create a page which does the following:

- ✓ When you open it for the first time, it says "Hello"
- ✓ When you open it for the second time, it says "Hello again"
- ✓ When you open it for the third time, it says "Goodbye"

And then .. it starts from the beginning.

CODING TIME

```php
<?php

if (!isset($_COOKIE['visits'])) {
  setcookie('visits', 1, time() + 3600);
  echo "Hello";
}
else {
  if ($_COOKIE['visits'] == 1) {
    setcookie('visits', 2, time() + 3600);
    echo "Hello again";
  }
  else {
    setcookie('visits', null, time() - 3600);
    echo "Goodbye";
  }
}

?>
```

2. SESSION VARIABLES

SESSION VARIABLES

WHAT ARE SESSION VARIABLES?

Session variables are variables which are persistent across multiple pages.

The same way a browser has a pocket for each site it visits, the web server (where the site is hosted) has a pocket for each browser which visits it.

SO, HOW DO WE USE THEM?

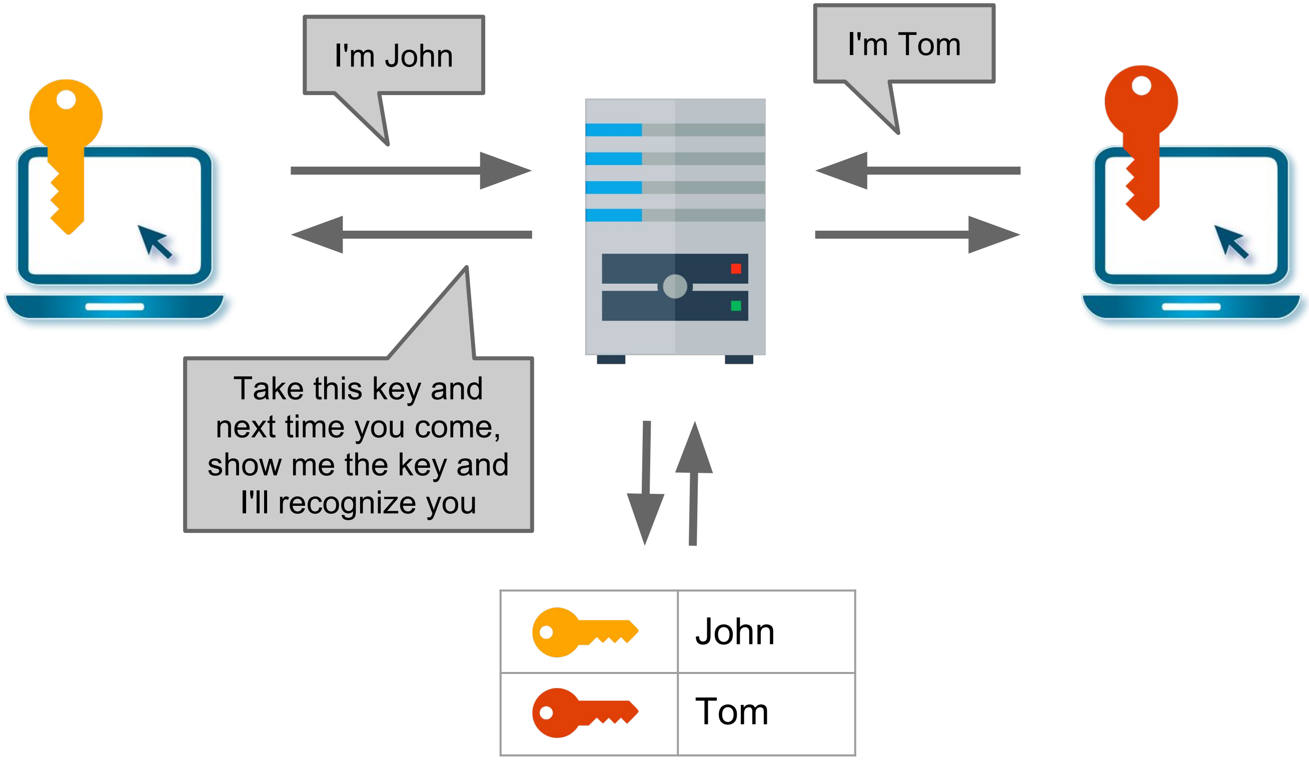
Before using them, we need to tell PHP that we want to work with session vars:

```
session_start();
```

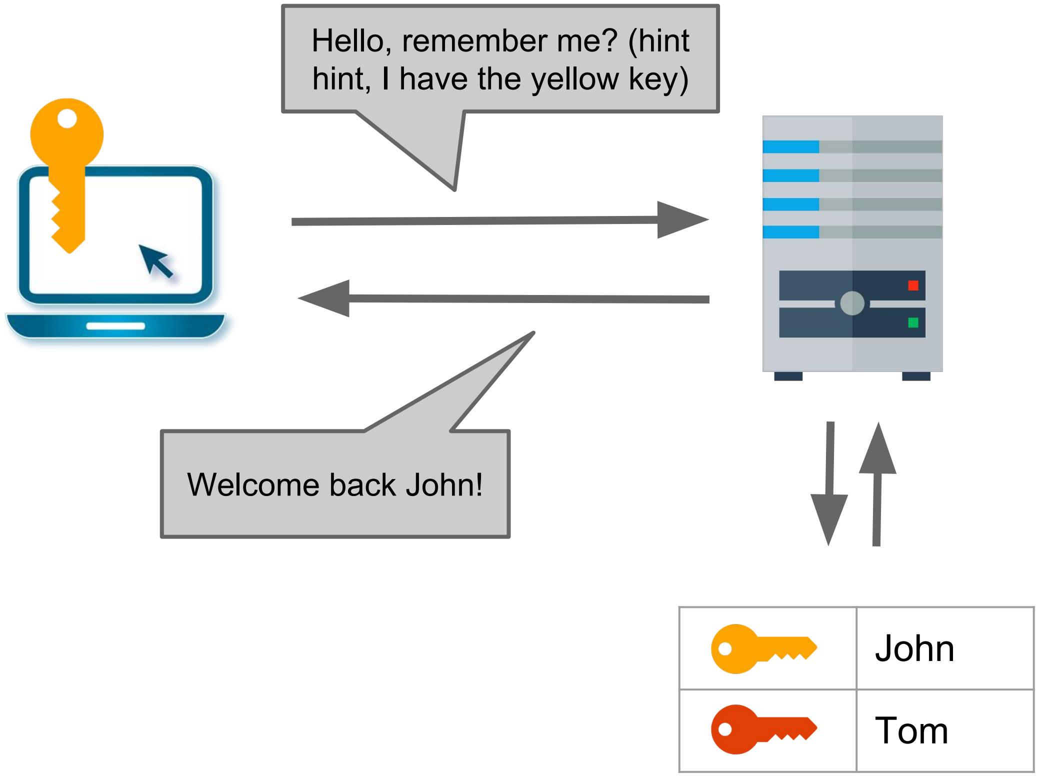
WHAT HAPPENS WHEN YOU DO THIS:

1. PHP generates a unique key for current browser
2. PHP creates a storage place where, based on the key, will store values which are for this browser only
3. PHP gives that key to the browser as a cookie

LET'S SEE HOW THEY WORK


I'm John
I'm Tom
Take this key and next time you come, show me the key and I'll recognize you
John
Tom

LET'S SEE HOW THEY WORK



TIPS & TRICKS

ATTENTION

Session must be started before any HTML tag or output (echo, print)!

CODING TIME

SAVING A SESSION VARIABLE

To set and retrieve session variables we use the global variable $_SESSION.

So all we have to do is to add items to this array:

```
<?php

session_start();
$_SESSION['name'] = 'John';

?>
```

CODING TIME

READING A SESSION VARIABLE

And then we can just read them any time:

```
<?php

session_start();
echo $_SESSION['name'];

?>
```

CODING TIME

DELETING A SESSION VARIABLE

To delete them, just like you delete an item from an array, use the unset() function.

```
<?php

unset($_SESSION['name']);

?>
```

CODING TIME

LET'S DO AN EXERCISE

Create a page which does the following:

- ✓ Starts the session
- ✓ Checks whether we have a $_GET parameter "name" and it saves the name in a session variable
- ✓ After this, it checks whether there is a session variable called "name" and it prints it

CODING TIME

```php
<?php

session_start();

if (isset($_GET['name'])) {
  $_SESSION['name'] = $_GET['name'];
}

if (isset($_SESSION['name'])) {
  echo $_SESSION['name'];
}

?>
```

TIPS & TRICKS

COOKIES VS. SESSION VARIABLES

Do you need to access the data both from the browser and from PHP?

- ✓ In that case, use Cookies.
- ✓ In all other cases, use Session variables.

3. MYSQL - SELECT STATEMENT

READING FROM DATABASE

LET'S USE THE TABLE FROM THE HOMEWORK

students

student_id	**student_name**	**student_email**	**course_id**
1	Popescu Ion	i.popescu@gmail.com	1
2	Ionescu Vasile	vasile@yahoo.com	2
3	Vasilescu Gheorghe	gv@gmail.com	1

READING FROM DATABASE

SELECT STATEMENT

Reading from a database is done using SELECT statements.

Open phpMyAdmin, select your database and go to SQL tab.

Write the following instruction and click Go.

```
SELECT * FROM students
```

READING FROM DATABASE

SELECT SPECIFIC FIELDS

Instead of retrieving all fields, in practice we often need only some of the values. For example, let's fetch only the students names:

```
SELECT student_name FROM students
```

Maybe we also need the ID:

```
SELECT student_id, student_name FROM students
```

READING FROM DATABASE

ORDER THE RESULTS

We might need the results in a specific order:

```
SELECT student_id, student_name FROM students ORDER BY student_name
```

LIMIT THE RESULTS

In some cases, you might want to get only the first X rows:

```
SELECT * FROM students ORDER BY student_name LIMIT 1
```

READING FROM DATABASE

FILTER THE RESULTS

In most practical situations you'll want to read the rows which match specific criteria, for example:

- Read the row with a specific ID
- Retrieve the student with a specific name
- Read the students attending a specific course
- etc

READING FROM DATABASE

LET'S SEE SOME EXAMPLES

Select student with ID = 2:

```
SELECT * FROM students WHERE student_id = 2
```

Select the students that attend the course with ID = 1

```
SELECT * FROM students WHERE course_id = 1
```

READING FROM DATABASE

COMPLEX QUERIES USING OPERATORS

Let's select students that attend courses with ID = 1 or 2:

```
SELECT * FROM students WHERE course_id = 1 OR course_id = 2
```

Select the students with the ID between 1 and 2:

```
SELECT * FROM students WHERE student_id BETWEEN 1 AND 2
```

https://dev.mysql.com/doc/refman/5.7/en/non-typed-operators.html

4. PHP + MYSQL

CONNECT TO MYSQL FROM PHP

All the MySQL queries can be executed from PHP.

First, we need to tell PHP to connect to the MySQL server and select the database:

```
$server = 'localhost';
$user = 'root';
$pass = '';
$db_name = 'fasttrackit';

$db_conn = mysql_connect($server, $user, $pass);
mysql_select_db($db_name, $db_conn)
```

RETRIEVING VALUES

To retrieve values from the database, we need to pass the SQL statements to MySQL using the mysql_query() function. After that, we use mysql_fetch_assoc() to retrieve each row, one by one.

```
$sql = "SELECT * FROM students WHERE student_id = 1";

$result = mysql_query($sql, $db_conn);
$row = mysql_fetch_assoc($result);
echo $row[student_name'];
```

5. HOMEWORK

HOMEWORK

LOGIN FORM

1. Create a table containing id, username and password fields

2. Populate it with data. Passwords should not be saved in clear text, instead insert MD5(password)

3. Write a simple login.php page, asking for username and password. Upon submit, check if user is valid, i.e. username exists in database and the password matches the one in the db

HOMEWORK

LOGIN FORM (CONTINUED)

4. If valid, set a session variable containing the user ID
5. Then redirect to another page, user.php
6. On the user.php page, display "You are logged in", if you have logged in successful, otherwise display a link to login.php page