# Interacting with users

Chencha Jacob

June 17, 2015

## Ways to retreive user information from user

- URL/Links. Data is in $_GET super global
- Forms. Data is usually in $_POST super global
- Cookies. Data is usually in $_COOKIE super global

## Sample form

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html lang="en">
    <head>
        <title>Form</title>
    </head>
    <body>

        <form action="register.php" method="post">
          Username: <input type="text" name="username" value="" /><br />
          Password: <input type="password" name="password" value="" /><br />
            <br />
          <input type="submit" name="submit" value="Submit" />
        </form>

    </body>
</html>
```

- Note method type we can just as easily use *GET*
- Note key value pairs denoted by *name=value*
- Note submit type

## Processing side data

```php
<?php

var_dump($_POST); //Data sent via POST verb
var_dump($_GET);//Data sent via POST verb
var_dump($_COOKIE);//Data sent via POST verb
var_dump($_REQUEST);//Data sent via POST verb
```

- Start server and try posting
- Now reload the page
- Note the data print out for the various globals

## Working with the data

```
<?php
$username=$_POST['username'];
$password=$_POST['password'];

echo "The user with username {$username} and password {$password} has been registered";
```

- We can extract the values to more easily understandable values which we can then extract
- Use of super globals is generally to be avoided we will discuss why and how in a later lesson

However you may have noted that the value is not guaranteed and as such you may wish to check and set default values.

```
<?php
$username=isset($_POST['username'])?$_POST['username']:"";
$password=isset($_POST['password'])?$_POST['password']:"";

echo "The user with username {$username} and password {$password} has been registered";
```

## Single page application

It is entirely possible to write it all in one page

```
<?php
if (isset($_POST['submit'])) {
        // form was submitted
        $username = $_POST["username"];
        $password = $_POST["password"];

        if ($username == "jacob" && $password == "secret") {
                echo "User {$username} was successfully logged in";
        } else {

                $message = "There were some errors.";
        }
} else {
        $username = "";
        $message = "Please log in.";
}
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html lang="en">
<head>
<title>Form</title>
</head>
<body>

<?php echo $message; ?><br />

<form action="single.php" method="post">
Username: <input type="text" name="username" value="<?php echo htmlspecialchars($username);
Password: <input type="password" name="password" value="" /><br />
<br />
<input type="submit" name="submit" value="Submit" />
</form>

</body>
</html>
```

Some advantages of this method include

- Group logic
- Easy access to posted variables in same page
- Easy error management

## Cookies

A cookie is a small piece of data sent from a website and stored in a user's web browser while the user is browsing that website. Every time the user loads the website, the browser sends the cookie back to the server to notify the website of the user's previous activity.

Why cookies

- HTTP is stateless
- IPs are not dependable, many users can share an IP or even one user can change IP

## Setting cookies

```
<?php
    $name = "test";
    $value = 45;
    $expire = time() + (60*60); // add seconds
    setcookie($name, $value, $expire);
        echo "Cookies we're set";
```

Cookies must be set before any output is sent to browser

A sample response would look something like this.

```
HTTP/1.1 200 OK
Host: localhost:8888
Connection: close
X-Powered-By: PHP/5.5.12-2ubuntu4.4
Set-Cookie: test=45; expires=Wed, 17-Jun-2015 18:38:50 GMT; Max-Age=3600
Content-type: text/html
```

## Reading cookie values

Once we have set cookies we can read them from super global $_COOKIE

```
<?php

    $name = "test";
    $value = 45;
    $expire = time() + (60*60); // add seconds
    setcookie($name, $value, $expire);
        echo $_COOKIE[$name];
```

Note the cookie value persists even if you now remove the *setcookie* function

```
<?php

    $name = "test";
        echo $_COOKIE[$name];
```

Remember even cookies can be unset, it is good to first validate existence of a cookie before using it.

## Removing cookies

Cookies can not be directly deleted from the server. Instead the server responds to the next client request setting the cookie value to null.

```
unset($_COOKIE); //Will not work
setcookie($name,null);//Will unset the cookie in the next request
```

Also changing the expiry date to a date in the past will work but is not a good idea.

## Larger storage with $_SESSION

A session allows us to use cookie information as a pointer to more information.

This is important because:

- Not limited in storage. Cookies can only hold about 4kb
- Reduced latency. Remember cookies are sent back and forth in every request. This can slow down your page
- Enhances security. Cookies are usually sent and stored in plain text. With sessions we store all sensitive information on the server.

## Session usage

```
<?php
  // Sessions use cookies which use headers and as such must appear at the beginning
  session_start();
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html lang="en">
    <head>
        <title>Playing with sessions</title>
    </head>
    <body>

        <?php
            $_SESSION["first_name"] = "chencha";
            $name = $_SESSION["first_name"];
            echo $name;
        ?>

    </body>
</html>
```

You will note sessions are much more easier to use. However there is a caveat to sessions, and that is that they automatically expire whenever the users browser is closed. This can either be a useful feature or bug depending on the context.

## Assignment

Write a script that displays a dump of items currently stored in the cookie. It should be a single page that accepts new items which when submitted are subsequently added to the $_COOKIE variable.