

# Logins, Sessions and Cookies

INFO/CS 2300:  
Intermediate Web Design and  
Programming

# Homework 3 - AJAX

- Optional
- Worth up to 3% extra credit on final grade
  - A -> A+
  - B- -> B
  - etc.
- Planning to release a full week before Spring Break so it will overlap with P3
- Due the Tuesday after Spring Break

# Mini Crash Course

- AJAX and debugging
- Thursday March 10
- 4:30 – 5:30
- ACCEL orange

Quiz Friday

# Project 3 Update

Additional requirement:

**Logins table will be required** for the P3 final

Feedback

**Read the TA comments**

TAs made an effort to provide suggestions

If you are not going to implement the suggestions you need to have a good reason why and explain it with your file submission.

Simple logins: *A warm-up*

# Site logins

It is pretty easy to have a login for protecting one page.

Let's do it...



```
<?php
if ( ! isset( $_POST['username'] ) && ! isset( $_POST['password'] ) ) {
?>
    <h2>Log in</h2>
    <form action="login.php" method="post">

        Username: <input type="text" /> <br>
        Password: <input type="password" /> <br>

        <input type="submit" value="Submit">
    </form>
```

```
<?php
} elseif ( <input type="text" /> ) {
?>
    <p>You have accessed the secret content of this page.</p>
<?php
} else {
```

```
<?php
if ( ! isset( $_POST['username'] ) && ! isset( $_POST['password'] ) ) {
?>
    <h2>Log in</h2>
    <form action="login.php" method="post">

        Username: <input type="text" name="username"> <br>
        Password: <input type="password" name="password"> <br>

        <input type="submit" value="Submit">
    </form>
```

```
<?php
} elseif ( $_POST['username'] == "smohlke"
           && $_POST['password'] == "mypassword" ) {
?>
    <p>You have accessed the secret content of this page.</p>
<?php
} else {
```



# Problems

What are some problems with this type of login?

# Problems

What are some problems with this type of login?

- Only works for one page
- Password stored as plain text
- Only one username/password
- Have to update the file for password changes

# Multiple pages via sessions



# \$\_SESSION

Session variables:

- Can be **set and read** by PHP
- **Persist** from page to page.
- **Temporary**: Usually expire when the user closes the browser
- **Stored on the server**
- Only good for **small quantities of data**

# Sessions persist page to page

When a user successfully logs in, we'll **set a session variable**. On the other pages, we'll check if this is set.

# Starting sessions

We start a session on any page in which we want to set or read a `$_SESSION` variable

**This must start *before* the HTML.**

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

```
<!DOCTYPE html>
```

```
<html>
```

```
...
```

# The login in a session variable

Then we store information in the associative array `$_SESSION`.

```
$username = filter_input( INPUT_POST, 'username', FILTER_SANITIZE_STRING );
$password = filter_input( INPUT_POST, 'password', FILTER_SANITIZE_STRING );

if ( empty( $username ) || empty( $password ) ) {
    echo "<p>Congratulations, $username ...<p>";
    $_SESSION[ 'logged_user' ] = $username;
} else {
    echo '<p>You did not login successfully.</p>'
    echo '<p>Please <a href="login.php">login</a></p>'
}
```

# Other pages

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

```
<!DOCTYPE html>
```

```
<html>
```

```
...
```

```
<?php
```

```
if ( isset( $_SESSION[ 'logged_user' ] ) ) {
```

```
    //Protected content here
```

```
    $logged_user = $_SESSION[ 'logged_user' ];
```

```
    print "<p>Welcome, $logged_user !</p>";
```

```
} else {
```

```
    print "<p>Please <a href='login.php'>login</a></p>";
```

```
}
```

```
?>
```



# Ending a session

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

Need to start sessions on the page in which you wish to end them

```
<!DOCTYPE html>
```

```
<html>
```

```
...
```

```
<?php
```

```
unset($_SESSION[ "logged_user" ] );
```

Clear one session variable

```
unset( $_SESSION );  
$_SESSION = array();
```

Clear ALL session variables. Not reversible

```
session_destroy();  
?>
```

Not as permanent as it sounds. Session can start a session and access it again

# Sessions aren't just for logins

Good for things like shopping carts

Click In!

# Click In!

Which is not true about Session variables

- A. They always remain valid as long as the browser is open
- B. They are stored on the server
- C. They are good for small amounts of data
- D. They persist through a page reload
- E. They persist on other pages on the same server

# Click In!

Which is not true about Session variables

- A. They always remain valid as long as the browser is open
- B. They are stored on the server
- C. They are good for small amounts of data
- D. They persist through a page reload
- E. They persist on other pages on the same server

# Back to logins

Sessions take care of the first problem

- ~~Only works for one page~~
- Password stored as plain text
- Only one user
- Have to update the file for password changes

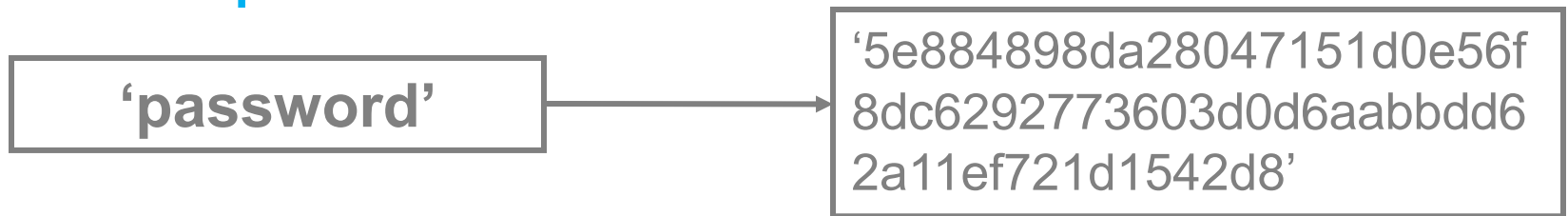
# Securing passwords with hashing

We don't want to have the user's password visible in our PHP file. What can we do?



# Hashing

*A cryptographic hash function obscures the password.*



Hashes should be  
“one-way”

Hashes should be  
“collision-free”

# A hash function

`hash('sha256', string)`

Returns a 64-character string hash of input *string*.

There are others besides sha256 but this is good for us for now.

# Hash the password

```
$post_username = filter_input( ..., 'username', ...);  
$post_password = filter_input( ..., 'password', ...);  
$hashed_password = hash("sha256",  
                          $post_password);  
if ($post_username == "smohlke" &&  
    $hashed_password ==  
    '5e884898da28047151d0e56f8dc6292773603  
    d0d6aabbbdd62a11ef721d1542d8') {  
    $_SESSION['logged_user'] = $post_username;  
}
```

# 2 down, 2 to go

Sessions take care of the first problem

- ~~Only works for one page~~
- ~~Password stored as plain text~~
- Only one user
- Have to update the file for password changes

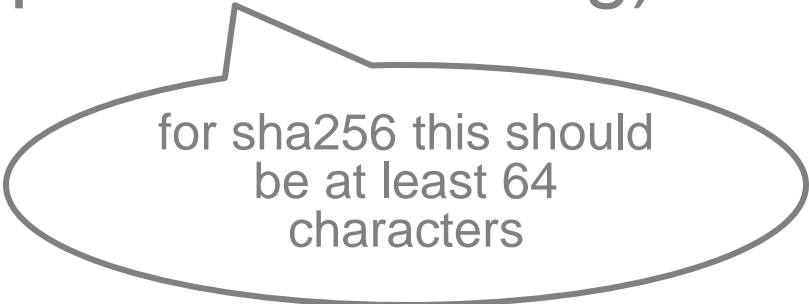
# Multiple users via MySQL



# Multiple users

We can **create a table** in which we store valid users and their (hashed) passwords; check this upon login.

Users(username: string, name: string,  
hashpassword: string)



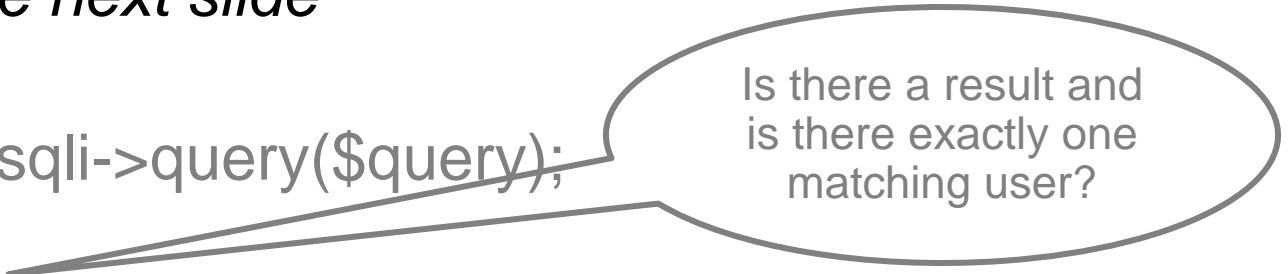
for sha256 this should  
be at least 64  
characters

```
CREATE TABLE IF NOT EXISTS `users` (  
    `userID` int(11) NOT NULL AUTO_INCREMENT,  
    `hashpassword` varchar(64) NOT NULL,  
    `username` varchar(50) NOT NULL,  
    `name` varchar(50),  
    PRIMARY KEY (`userID`),  
    UNIQUE KEY `idx_unique_username` (`username`)  
    ) ENGINE=InnoDB DEFAULT CHARSET=utf8  
    COLLATE=utf8_unicode_ci AUTO_INCREMENT=1 ;
```

```
if ( empty( $post_username ) || empty( $post_password ) ) {  
    // Ask for password – form goes here  
} else {  
    require_once '../config.php';  
    $mysqli = new mysqli( DB_HOST, DB_USER,  
                          DB_PASSWORD, DB_NAME);
```

```
$query = see next slide
```

```
$result = $mysqli->query($query);
```



Is there a result and  
is there exactly one  
matching user?

```
if ( $result && $result->num_rows == 1 ) {
```

```
    $_SESSION['logged_user'] = $_POST['username'];
```

```
}
```

```
}
```



```
//hash the entered password for comparison with the db  
$hashed_password = hash( "sha256",$post_password );
```

```
//Check for a record that matches the POSTed credentials
```

```
$query = "SELECT *
```

```
FROM users
```

```
WHERE
```

```
    username = '$post_username'
```

```
    AND hashpassword = '$hashed_password'; ";
```

# Logouts

What do we need to do to log out a user?

```
<?php
    session_start();
    if (isset($_SESSION['logged_user'])) {
        $olduser = $_SESSION['logged_user'];
        unset( $_SESSION[ 'logged_user' ] );
    } else {
        $olduser = false;
    }
}
```

```
?>
```

```
<!DOCTYPE html>
```

```
<html>
```

```
...
```

```
<?php
```

```
    if ( $olduser ) {
        print( "<p>Thanks for using our page, $olduser!</p>");
        print( "<p>Return to our <a href='login.php'>login page</a></p>");
    } else {
        print( "<p>You haven't logged in.</p>");
        print( "<p>Go to our <a href='login.php'>login page</a></p>" );
    }
}
```

```
?>
```

Some security issues

# Password hash issues

There's sometimes an easy way to figure out what a password is given its hash.  
What is it?

# Bad guy method

1. Take lots of common passwords
2. Compute all of their sha256 hashes
3. Get access to the hashed passwords of lots of users
4. See if any of their hashed passwords match yours

# A fix

Check passwords for common words and variants before accepting them.

```
<?php
    if ( isset( $_POST['password'] ) ) {
        if ( ! crack_check( $_POST['password'] ) ) {
            print( "Choose another password: " );
            print( crack_getlastmessage() );
            // Go back and get another password
        }
    }
?>
```

NOTE: These functions not a part of standard PHP installs.  
Not expected for this course  
[www.php.net/manual/en/function.crack-check.php](http://www.php.net/manual/en/function.crack-check.php)



# Add salt

Add extra information (e.g. a random string)  
before hashing the password

```
$salt = 'yadayadayada';
```

```
...
```

```
if (...hash('sha256', $post_password . $salt) ==  
    $row[ 'hashpassword' ] ) {
```

# Hashed credentials in db

Major problems solved

- ~~Only works for one page~~
- ~~Password stored as plain text~~
- ~~Only one user~~
- ~~Have to update the file for password changes~~

# Cookies



# Cookies

Cookies are one way to save limited amounts of information between visits of a user.

# Saving a cookie

`setcookie($name, $value)`

`setcookie($name, $value, $expiration)`

Without `$expiration` argument, cookie goes away when the browser closes. `$expiration` is number of seconds after January 1, 1970. Set as follows:

```
setcookie( "name", "value", time() + 60 * 60 );
```

Must be at the top of the .php file (*before* the DOCTYPE or any other HTML).

# Getting a cookie

If a cookie was set using `setcookie("name", "value")`, and the expiration date has not passed, then when the user returns, the variable `$_COOKIE["name"]` will contain "value".

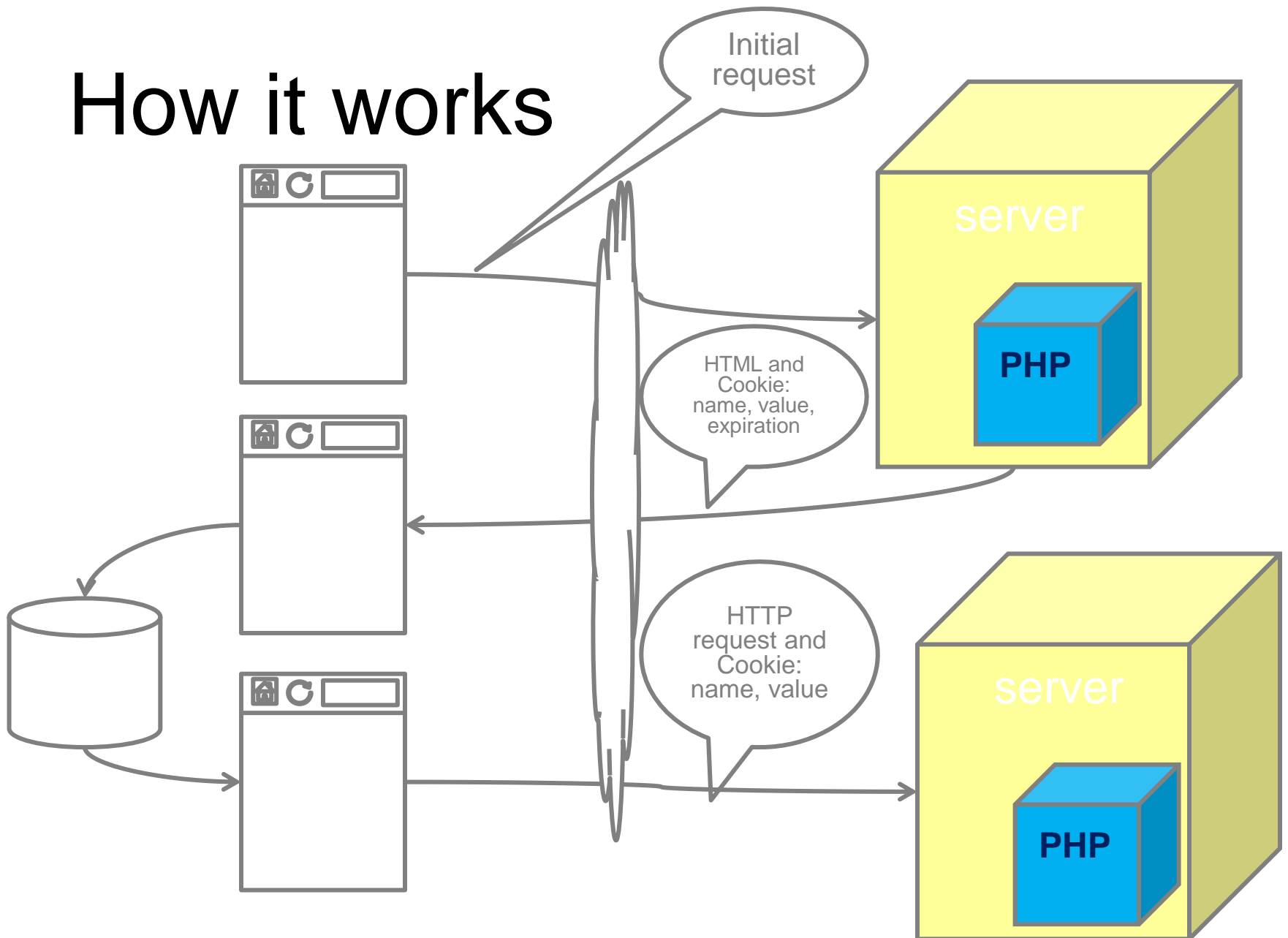
```
<?php
$username = filter_input( INPUT_COOKIE, 'username', ... );

if( empty( $username ) ) {
    $username = filter_input( INPUT_POST, 'username', ... );
}

if( empty( $username ) ) {
    ?>
    <form method="post" action="cookie-form.php">
        <p>What is your name?
        <input type="text" name="username">
        </p>
        <input type="submit" value="Click to submit">
    </form>
    <?php
} else {
    print("<p>Welcome, $username!</p>");
}
?>
```



# How it works





# Deleting a cookie

Set its expiration date to be in the past.

```
setcookie('name', 'value', time() - 3600);
```

# How long can a cookie last?

The largest time that can be represented by a 32-bit integer is 03:14:08 January 19, 2038.

# Cookies for logins

How could this be done safely?

# Project 3

Albums have title, date created, date last modified.

Photos have caption, date taken, url.

Photos can be in multiple albums.

What should we do for a schema?