WEBD 236

Web Information Systems Programming

Week 6

Todd Whittaker and Scott Sharkey
(sharkesc@franklin.edu)

FRANKLIN UNIVERSITY

Agenda

- This week’s expected outcomes
- This week’s topics
- This week’s homework
- Upcoming deadlines
- Questions and answers

Week 6 Outcomes

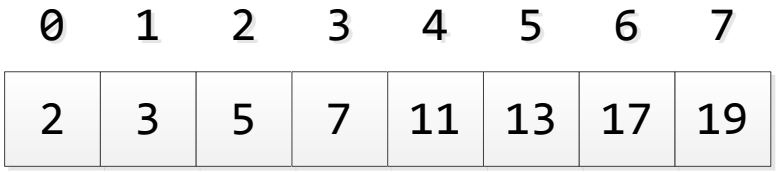
- Employ algorithms to work with arrays and associative arrays.
- Use common array functions.
- Describe the security implications of session tracking.
- Employ sessions to maintain per-user data on the server.

Arrays

- Create arrays with the array() function

```
$arr = array(2, 3, 5, 7, 11, 13, 17, 19);
```

- Indices are in the range [0, *n*-1] for an array of length *n*.



- Use [] to access elements

```
$arr[3] = $arr[7] + 1;
```

Arrays

- Use the function count() to determine how long an array is. Can then use a loop to process it.

```
$arr = array(2, 3, 5, 7, 11, 13, 17, 19);
$len = count($arr);
for ($i = 0; $i < $len; ++$i) {
    $arr[$i] += 1;
}
```

Arrays

- Assign a value “past” the end of the array to add on to the end.

```
$arr = array();
$arr[0] = 2;
$arr[1] = 3;
$arr[2] = 5;
```

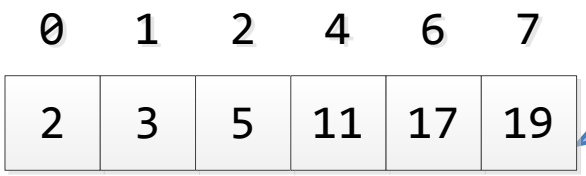
- Shortcut to do the same thing:

```
$arr = array();
$arr[] = 2;
$arr[] = 3;
$arr[] = 5;
```

Arrays

- Remove elements using unset()

```
$arr = array(2, 3, 5, 7, 11, 13, 17, 19);
unset($arr[3]);
unset($arr[5]);
```



Notice that the valid array indices are no longer contiguous! This is a hint that all arrays are actually associative.

Arrays

- Can also use a foreach loop to iterate:

```
function myArrayValues($arr) {
    $result = array();
    foreach ($arr as $element) {
        $result[] = $element;
    }
    return $result;
}
$arr = array(2, 3, 5, 7, 11, 13, 17, 19);
unset($arr[3]);
unset($arr[5]);
$arr = myArrayValues($arr);
```

array_values() is a library function that does this.

0	1	2	3	4	5
2	3	5	11	17	19

Arrays

- Can also use a foreach loop to iterate:

```
function myArrayValues($arr) {
    $result = array();
    foreach ($arr as $element) {
        $result[] = $element;
    }
    return $result;
}
$arr = array(2, 3, 5, 7, 11, 13, 17, 19);
unset($arr[3]);
unset($arr[5]);
$arr = myArrayValues($arr);
```

Associative Arrays

- All PHP arrays are actually associative.

Idea: use strings (as well as integers) as keys

```
$arr = array(4 => "four", "four" => 4, '5' => "five");
print_r($arr);
```

```
Array (
    [4] => four
    [four] => 4
    [5] => five
)
```

Strings containing integers are just interpreted as integers.

Associative Arrays

- Can then index by an arbitrary string.

```
$person = array();
$person['lastName'] = 'Smith';
$person['firstName'] = 'Roger';
$person['dob'] = '12-Nov-1968';
print("Hello ${person['firstName']} ${person['lastName']}.");
```

Arrays of Arrays

- Commonly, arrays of associative arrays are representation of rows and columns in a DB.

```
$firstNames = array('James', 'John', 'Robert', 'Michael');
$lastNames = array('Smith', 'Johnson', 'Williams', 'Jones');
$people = array();
for ($i = 0; $i < 5; $i++) {
    $findex = mt_rand(0, count($firstNames) - 1);
    $lindex = mt_rand(0, count($lastNames) - 1);
    $person = array(
        'firstName' => $firstNames[$findex],
        'lastName' => $lastNames[$lindex]
    );
    $people[] = $person;
}
print_r($people);
```

Arrays of Arrays

- Commonly, arrays of ass

representation of rows and

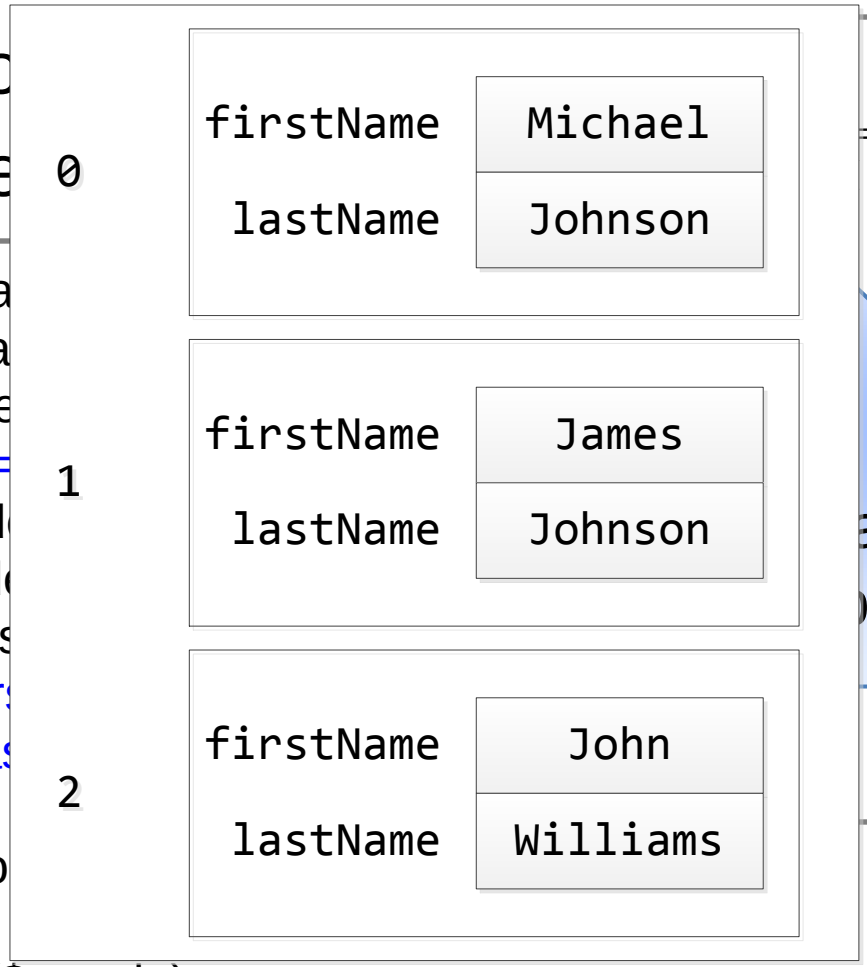
```
$firstNames = array('James', 'John', 'Rober
$lastNames = array('Smith', 'Johnson', 'Wil
$people = array();
for ($i = 0; $i < 5; $i++) {
    $findex = mt_rand(0, count($firstNames)
    $lindex = mt_rand(0, count($lastNames)
    $person = array(
        'firstName' => $firstNames[$findex],
        'lastName' => $lastNames[$lindex]
    );
    $people[] = $person;
}
print_r($people);
```

```
Array
(
    [0] => Array
        (
            [firstName] => Michael
            [lastName] => Johnson
        )

    [1] => Array
        (
            [firstName] => James
            [lastName] => Johnson
        )

    [2] => Array
        (
            [firstName] => John
            [lastName] => Williams
        )
    ...
```

Arrays of Arrays

- Co
repre

```
$firstNa
$lastNa
$people
for ($i =
    $find
    $lind
    $pers
        'fir
        'las
    );
    $peo
}
print_r($people);
```




```
=> Array

    [firstName] => Michael
    [lastName] => Johnson

    [firstName] => John
    [lastName] => Williams
```

ael" is stored at
ple[0]['firstName']

Arrays of Arrays

- Can then iterate over all those arrays.

```
foreach($people as $person) {
    print("Hello ${person['firstName']} ${person['lastName']}");
}
```

```
foreach($people as $person) {
    foreach($person as $key => $value) {
        print("$key: $value");
    }
}
```

Common Array Functions

- Full list http://php.net/manual/en/ref.array.php

Function	Purpose
`range($lo, $hi[, $step])`	Returns an array with values between `$lo` and `$hi`, `$step` apart.
`array_slice($arr, $index[, $len[, $keys]])`	Returns a sub-array of `$arr` starting at `$index` containing `$len` elements.
`array_splice($arr, $index[, $len[, $new]])`	Replaces `$len` elements with `$new` in `$arr` starting at `$index`.
`in_array($val, $arr)`	Returns `true` if `$val` appears in `$arr`
`array_search($val, $arr)`	Returns the index of `$val` in `$arr` or `false` if it doesn't exist

Common Array Functions

- Full list http://php.net/manual/en/ref.array.php

Function	Purpose
`sort($arr[, $compare])`	Sorts an array in ascending order, reindexing. See also `rsort`, `asort`, etc.
`array_map($callback, $arr)`	Applies the function `$callback` to every element of `$arr`, returning result array.
`array_shift($arr)`	Returns the first element, and shifts every element down one position. FIFO.
`array_pop($arr)`	Same as `array_shift`, but with the last element. LIFO.

Sessions

- Problem: HTTP is stateless – each request/response cycle is completely independent.
  - How can your program “remember” things from one click to the next? Things like:
    - Who is logged in?
    - What page in a multi-page “wizard” are you on?
    - What page should you be redirected to after you log in?

Sessions

- Solution: set a cookie
  - Cookies are set by the server, stored by the browser, and are transmitted with every request.

```
HTTP/1.1 200 OK
Content-type: text/html
Set-Cookie: name=value
Set-Cookie: name2=value2; Expires=Wed, 09 Jun 2021 10:18:14 GMT
```

```
GET /spec.html HTTP/1.1
Host: www.example.org
Cookie: name=value; name2=value2
Accept: */*
```

Source: http://en.wikipedia.org/wiki/HTTP_cookie

Sessions

- Problem: browser cookies can't be trusted

```
GET /spec.html HTTP/1.1
Host: www.example.org
Cookie: loggedin=true; rights=admin
Accept: */*
```

Anybody can send a request with any cookie they wish. If we rely on cookies for sensitive data (especially guessable data), this is a severe security risk.

Sessions

- Solution: use an opaque identifier (like a surrogate key) that references a file on the server.

```
HTTP/1.1 200 OK
Content-type: text/html
Set-Cookie: PHPSESSID=jokilcf2qsckfuml9mg73jamv0
```

```
GET /spec.html HTTP/1.1
Host: www.example.org
Cookie: PHPSESSID=jokilcf2qsckfuml9mg73jamv0
Accept: */*
```

Sessions

- Solution: use an opaque identifier (like a surrogate key) t n the server.

This isn't guessable (is it?). But, we can use it to "look up" sensitive session data we've stored on the server.

```
HTTP/1.1 200 OK
Content-type: text/html
Set-Cookie: PHPSESSID=jokilcf2
mg73jamv0
```

```
GET /spec.html HTTP/1.1
Host: www.example.org
Cookie: PHPSESSID=jokilcf2qsckfuml9mg73jamv0
Accept: */*
```

Sessions

- Solution: use an opaque identifier (like a surrogate key) that refere… server.

```
HTTP/1.1 200 OK
Content-type: text/html
Set-Cookie: PHPSESSID=jokilcf2qsckfuml9…
```

```
GET /spec.html HTTP/1.1
Host: www.example.org
Cookie: PHPSESSID=jokilcf2qsckfuml9mg73jamv0
Accept: */*
```

Problem: what if someone's session ID is hijacked? Is that even possible?

Sessions

- Problem: Open WiFi connections permit session hijacking
  - Even WEP isn't secure
  - What about other man-in-the-middle attacks?
  - Therefore, all sessions must be over an encrypted connection.
- Solution: force all connections to be HTTPS.
  - All data between browser and server is encrypted.

Sessions

- Quick recap:
  - Force HTTPS connections
  - Set an unguessable cookie
  - Use that cookie to reference a data structure on the server that holds per-user session data.

Sessions

- Forcing HTTPS
  - Use .htaccess (like we did for URL rewriting)

```
Options +FollowSymLinks
IndexIgnore */*
# Turn on the RewriteEngine
RewriteEngine On
#  Force HTTPS for security of cookies
RewriteCond %{HTTPS} !on
RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI} [L]
# Handle URL routing
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . urlrouter.php
```

Sessions

- Forcing HTTPS
  - Use .htaccess (like we di

```
Options +FollowSymLinks
IndexIgnore */*
# Turn on the RewriteEngine
RewriteEngine On
#  Force HTTPS for security of cookies
RewriteCond %{HTTPS} !on
RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI} [L]
# Handle URL routing
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . urlrouter.php
```

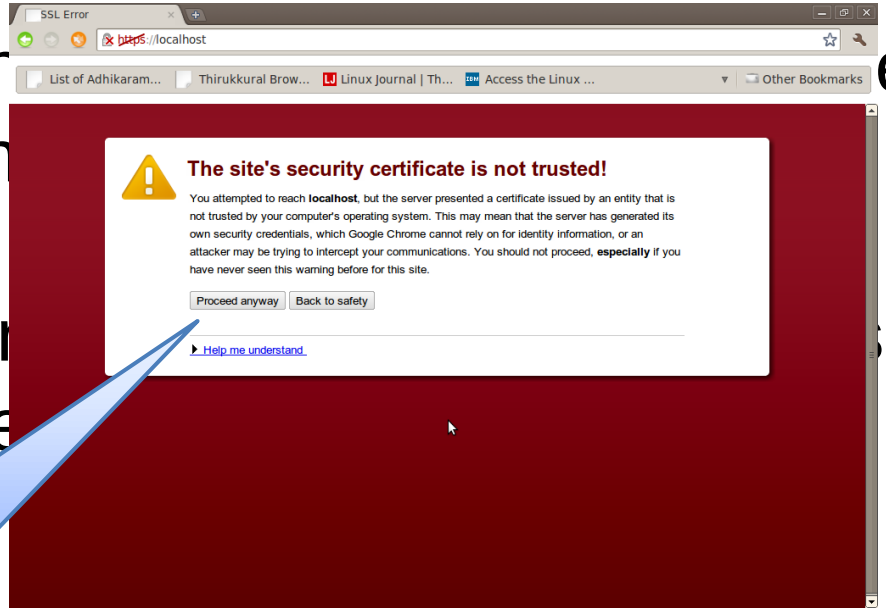
The [L] flag means that if this rule matches, stop processing other rules. “Last rule.”

Sessions

- Forcing HTTPS
  - Real HTTPS connections require a certificate signed by a signing authority (Thawte, Verisign, etc.)
  - XAMPP will still encrypt, but the certificate is self-signed, so the browser will complain.

Sessions

- Forcing HTTPS
  - Real HTTPS conn… e signed by a signin… Verisign, etc.)
  - XAMPP will still e… self-signed, so the…



Don't panic. It's okay to proceed. We can trust our own localhost.

Sessions

- Setting a cookie
  - Use the setcookie() function

```
setcookie($name, $value, $expire,
    $path, $domain, $secure, $httponly);
```

Can read about all of these parameters in the book. But this isn't the path we want to go down. PHP sessions handle setting the session cookie for us.

Sessions

- Session parameters
  - How long should the cookie live?
  - What paths on the server should apply?
  - What's the name of the domain to send to?
  - Should it only be sent on encrypted connections?
  - Should only HTTP read it (no JavaScript)?

```
session_set_cookie_params($seconds, $path, $domain,
    $secure, $httponly);
```

Sessions

- Starting a session

```
session_set_cookie_params(60*60*24*14, '/',
    $_SERVER['SERVER_NAME'], true, false);
session_start();
```

- Storing data in a session

```
$_SESSION['loggedin'] = true;
$_SESSION['username'] = 'Fred';
```

Sessions

- Reading data from a session

```
function safeParam($arr, $index, $default) {
    if ($arr && isset($arr[$index])) {
        return $arr[$index];
    }
    return $default;
}

$user = safeParam($_SESSION, 'username', false);
if ($user) {
    print("Hello $user!");
}
```

Sessions

- Removing data from a session

```
// remove a single variable
unset($_SESSION['username']);

// delete all variables
$_SESSION = array();
```

- Ending a session

```
session_destroy();
```

Sessions

- Getting the current session ID

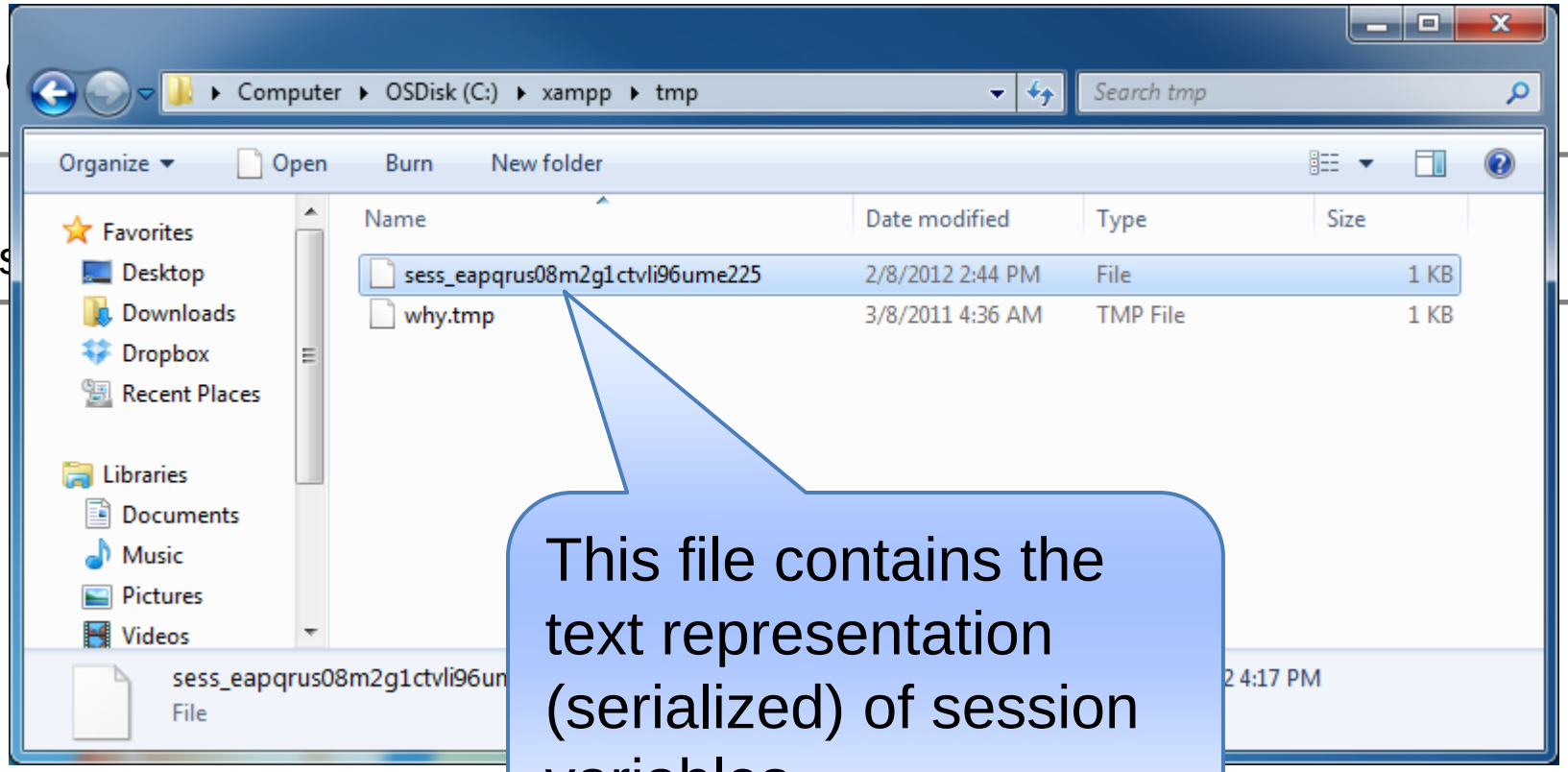
```
// can be used to see if a session is active
$sessid = session_id();
```

- Where is session data kept?
  - In XAMPP it is C:\xampp\tmp.
  - On a production server, you schedule cleanups.
  - Also possible to store session data in the database and have a trigger clean it up.

Sessions

- (

```
//
$s
```



This file contains the text representation (serialized) of session variables

Sessions

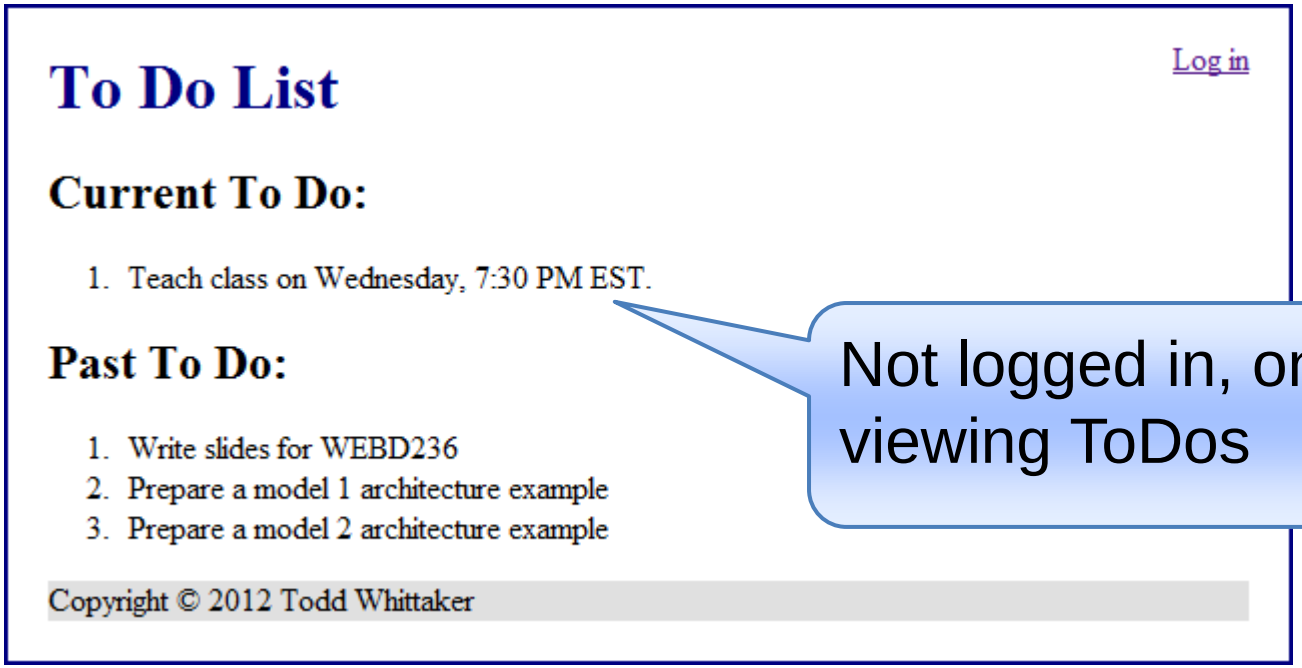
- Full list of session-related functions:

http://php.net/manual/en/ref.session.php

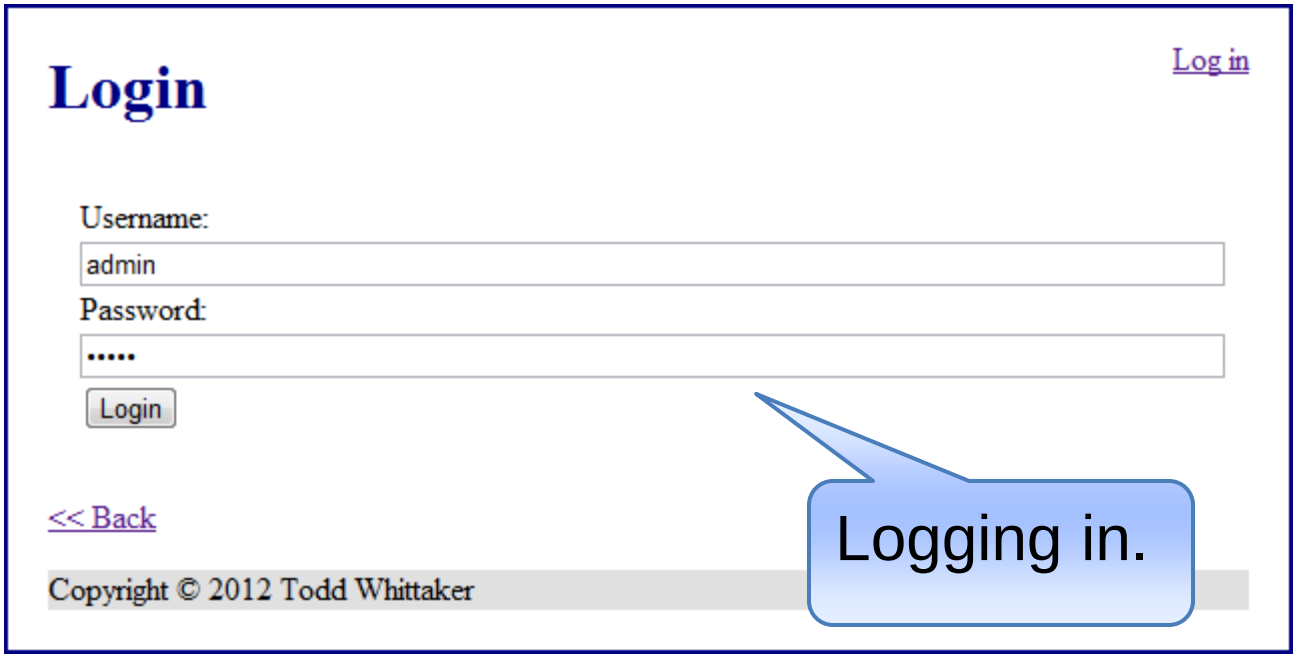
Session Example

- In-depth example: adding login/logout and minimal authentication requirements to our ToDo List application
  - Specifications:
    - Non-logged in users can only see landing page
    - Logged in users can add, edit, delete ToDos
    - Don’t permit URL fishing
    - Provide login/logout capabilities

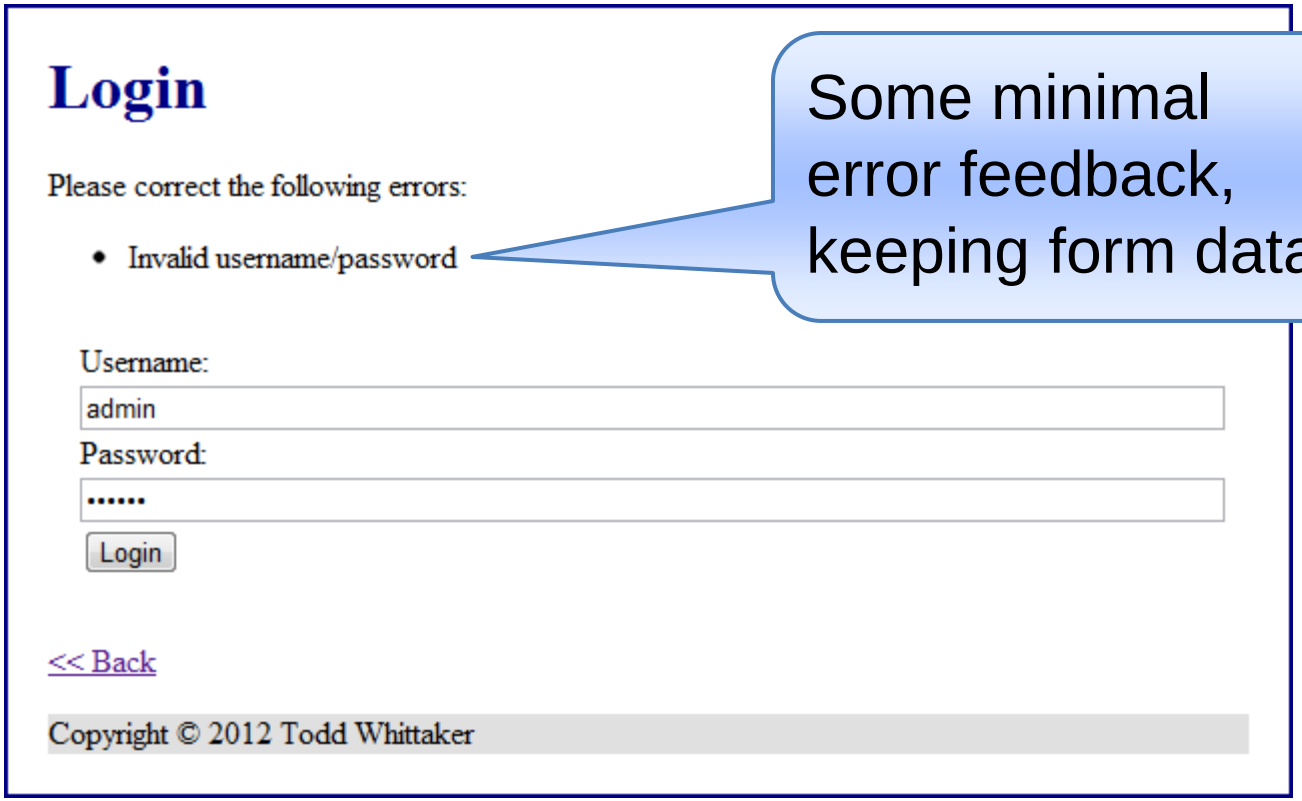
Session Example



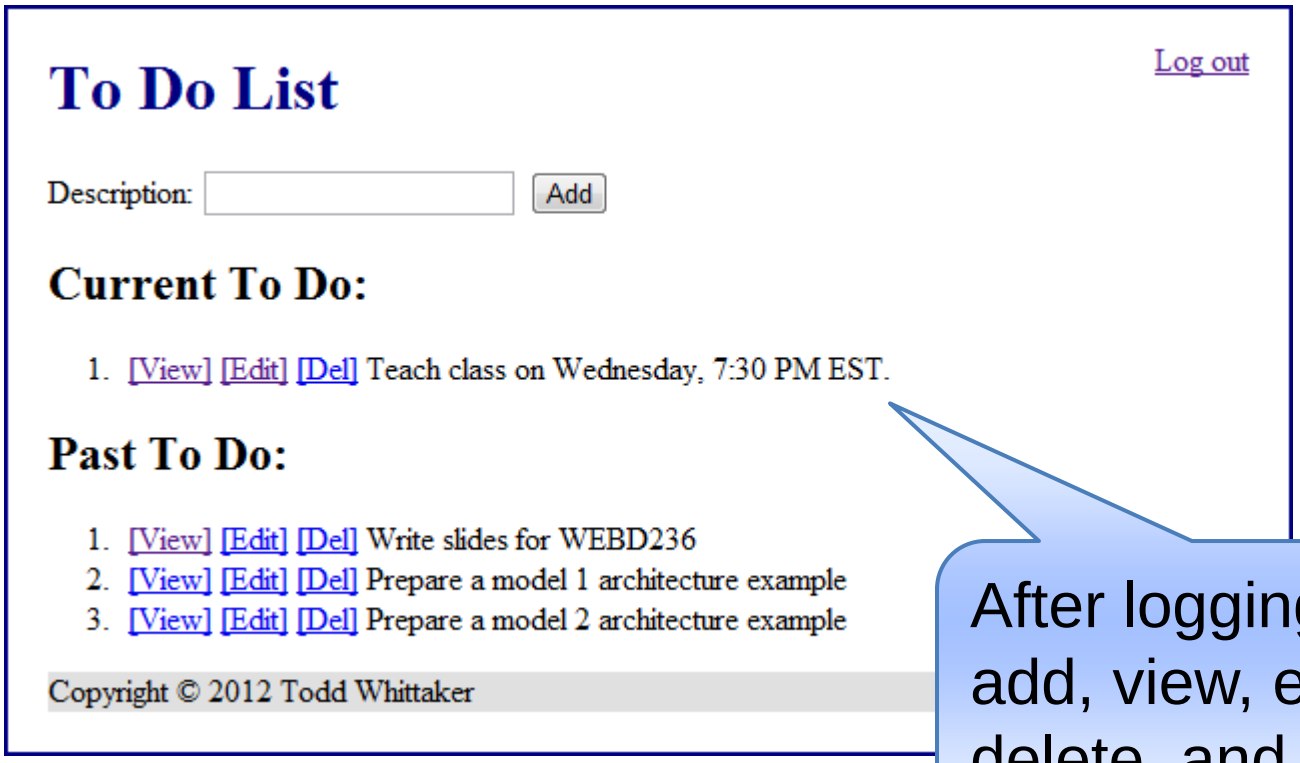
Session Example



Session Example



Session Example



After logging in: add, view, edit, delete, and log out.

Session Example

- URL fishing:
  - When not logged in, navigating directly to https://localhost/webd236/LoginExample/todo/view/1 should redirect to a login screen and after logging in, the user should be redirected back to the todo they attempted to view.

Session example

- Set up your .htaccess file

```
Options +FollowSymLinks
IndexIgnore */*
# Turn on the RewriteEngine
RewriteEngine On
#  Force HTTPS for security of cookies
RewriteCond %{HTTPS} !on
RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI} [L]
# Handle URL routing
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . urlrouter.php
```

Session example

- Sessions should start immediately

```
// inside the urlrouter.php file
session_set_cookie_params(60*60*24*14, '/',
    $_SERVER['SERVER_NAME'], true, false);
session_start();
routeUrl();
```

Session example

- A useful function in Lib/Util.inc

```
function isLoggedIn() {
    $inSession = session_id();
    if (!empty($inSession)) {
        if (isset($_SESSION['loggedin'])) {
            return $_SESSION['loggedin'];
        }
    }
    return false;
}
```

Session example

- Changes to the views/header.html

```
<!-- snipped top section -->
    <body>
        <div class="content">
[[ include_once 'Lib/Util.inc'; ]]
[[ if (isLoggedIn()) : ]]
    <p class='login'><a href='@@auth/logout@@'>Log out</a></p>
[[ else : ]]
    <p class='login'><a href='@@auth/login@@'>Log in</a></p>
[[ endif; ]]
```

Class login is a float left style.

This means we'll need an auth controller

Session example

- Changes to the views/index.inc

```
%% views/header.html %%
<h1>{{$title}}</h1>
[[if (isLoggedIn()) : ]]
<form action="@@todo/add@@" method="post">
    <label for="description">Description:</label>
    <input type="text" id="description" name="description" />
    <input type="submit" value="Add" />
</form>
[[ endif; ]]
<h2>Current To Do:</h2>
<ol>
    [[ foreach ($todos as $todo) : ]]
    <!-- more changes omitted -->
```

Only display form if user is logged in. Same for edit, delete and view links (omitted)

Session example

- Create the controllers/auth.inc

```
<?php
include_once “Lib/Util.inc";
include_once "models/users.inc";

function get_login($params) {
    renderTemplate(
        "views/login_form.inc",
        array(
            'title' => 'Login',
        )
    );
}
?>
```

Called when the user clicks the ‘login’ link in the header. Just renders the login form.

Session example

- Create the views/login_form.inc

```
%% views/header.html %%
<h1>{{$title}}</h1>
%% views/errors.html %%
<div class='inputs'>
    <form action="@@auth/login@@" method="post">
        <label for="username">Username:</label>
        <input type="text" id="username" name="username"
            value="{{isset($username) ? $username : ''}}" />
        <!– password omitted for space -->
        <input type="submit" value="Login" />
    <form>
</div>
<p><a href="@@index@@"><< Back</a></p>
%% views/footer.html %%
```

Session example

- Create the views/logi

errors.html pulls errors out and reports them to the user (if they exist)

```
%% views/header.html %%
<h1>{{$title}}</h1>
%% views/errors.html %%
<div class='inputs'>
    <form action="@@auth/login@@" method="post">
        <label for="username">Username:</label>
        <input type="text" id="username" name="username"
            value="{{isset($username) ? $username : ''}}" />
        <!– password omitted for space -->
        <input type="submit" value="Login" />
    <form>
</div>
<p><a href="@@index@@"><< Back</a></p>
%% views/footer.html %%
```

Session example

- Create the views/errors.html

```
[[ if (isset($errors)) : ]]
    <p>Please correct the following errors:</p>
    <ul>
[[   foreach ($errors as $error) : ]]
        <li>{{$error}}</li>
[[   endforeach; ]]
    </ul>
[[ endif; ]]
```

Session example

- Create the views/login_form.inc

```
%% views/header.html %%
<h1>{{$title}}</h1>
%% views/errors.html %%
<div class='inputs'>
    <form action="@@auth/login@@" method="post">
        <label for="username">Username:</label>
        <input type="text" id="username" name="username"
            value="{{isset($username) ? $username : ''}}" />
        <!– password omitted for space -->
        <input type="submit" value="Login" />
    <form>
</div>
<p><a href="@@index@@"><< Back</a></p>
%% views/footer.html %%
```

This means we need a controllers/auth.inc with a function post_login.

Session example

- Modify the controllers/auth.inc

```
function post_login($params) {
    $username = safeParam($_REQUEST, 'username', false);
    $password = safeParam($_REQUEST, 'password', false);
    if (isValidUser($username, $password)) {
        $_SESSION['loggedin'] = true;
        $_SESSION[‘username’] = $username;
        if (isset($_SESSION['redirect'])) {
            $redirect = $_SESSION['redirect'];
            redirect($redirect);
            exit();
        }
        redirectRelative("index");
    } else {
    // continued
```

Session example

- Modify the controllers/login.inc

```
function post_login($params) {
    $username = safeParam($_REQUEST, 'username', false);
    $password = safeParam($_REQUEST, 'password', false);
    if (isValidUser($username, $password)) {
        $_SESSION['loggedin'] = true;
        if (isset($_SESSION['redirect'])) {
            $redirect = $_SESSION['redirect'];
            redirect($redirect);
            exit();
        }
        redirectRelative("index");
    } else {
    // continued
```

isValidUser should query the DB based on username and a hash of the password.

Session example

- Modify the controllers/auth.inc

```
function post_index($params) {
    $username = safeParam($_REQUEST, 'username', false);
    $password = safeParam($_REQUEST, 'password', false);
    if (isValidUser($username, $password)) {
        $_SESSION['loggedin'] = true;
        $_SESSION[‘username’] = $username;
        if (isset($_SESSION['redirect'])) {
            $redirect = $_SESSION['redirect'];
            redirect($redirect);
            exit();
        }
        redirectRelative("index");
    } else {
    // continued
```

In a “real” login/logout situation, we’d want to store a user ID here.

Session

- Modify the controll

If they attempt to access a protected resource without logging in, then this session variable will be set.

```
function post_login($params) {
    $username = safeParam($_REQUEST[    me', false);
    $password = safeParam($_REQUEST[    sword', false);
    if (isValidUser($username, $passwor    ) {
        $_SESSION['loggedin'] = true;
        if (isset($_SESSION['redirect'])) {
            $redirect = $_SESSION['redirect'];
            redirect($redirect);
            exit();
        }
        redirectRelative("index");
    } else {
    // continued
```

Session example

- Modify the controllers/auth.inc

```
    // continued
    } else {
        renderTemplate(
            "views/login_form.inc",
            array(
                'title' => 'Login',
                'errors' => array("Invalid username/password"),
                'username' => $username,
                'password' => $password
            )
        );
    }
}
```

This lets us keep the values the user entered in already.

Session example

- Create models/users.inc

```php
<?php
function isValidUser($username, $password) {
    return $username == 'admin' && $password == 'nimda';
}
?>
```

In a real application, we'd query the database, and likely end up storing the user ID in a session variable.

Session example

- Modify the controllers/auth.inc

```
<?php
include_once “Lib/Util.inc";
include_once "models/users.inc";

function get_logout($params) {
    $_SESSION = array();
    session_destroy();
    redirectRelative("index");
}
?>
```

Just destroy the session and redirect to the home page.

Session example

- Prevent URL fishing in controllers/todo.inc

```
function get_view($params) {
    ensureLoggedIn();
    $id = safeParam($params, 0,
    if ($id === false) {
        die("No todo id specified");
    }

    $todo = findToDoById($id);
    if (!$todo) {
        die("No todo with id $id found.");
    }
    // remainder skipped
```

ensureLoggedIn will make sure that this function can execute only if the user is authenticated. Call this first in every method you want protected.

Session example

- Modify Lib/Util.inc

```
function ensureLoggedIn() {
    if (!isLoggedIn()) {
        $_SESSION['redirect'] = $_SERVER['REQUEST_URI'];
        redirectRelative('login');
        exit();
    }
}

function redirect($url) {
    session_write_close();
    header("Location: $url");
    exit();
}
```

This updates the redirect session variable to know where to go to after logging in.

Session example

- Modify Lib/Util.inc

```
function ensureLoggedIn() {
    if (!isLoggedIn()) {
        $_SESSION['redirect'] = $_SERVER['REQUEST_URI'];
        redirectRelative('login');
        exit();
    }
}

function redirect($url) {
    session_write_close();
    header("Location: $url");
    exit();
}
```

This ensures that session files are updated on a redirection (i.e. no HTML output).

Session example

- Complete source code for the entire working example is available at http://cs.franklin.edu/~sharkesc/webd236/

Upcoming Deadlines

- Readings for next week
  - Chapters 13 and 14 in *PHP and MySQL*
- Assignments
  - Homework 5 due end of week 6
  - Lab 2 due end of week 7
- Next week:
  - Functions and object-oriented programming

General Q & A

- Questions?
- Comments?
- Concerns?

FRANKLIN UNIVERSITY