DT228/2 Web Development

Cookies, Sessions, and Authentication

High Level Summary

- The web is “stateless” - the browser does not maintain a connection to the server while you are looking at a page. You may never come back to the same server - or it may be a long time - or it may be one second later
- So we need a way for servers to know “which browser is this?”
- In the browser state is stored in “Cookies”
- In the server state is stored in “Sessions”

Multi-User

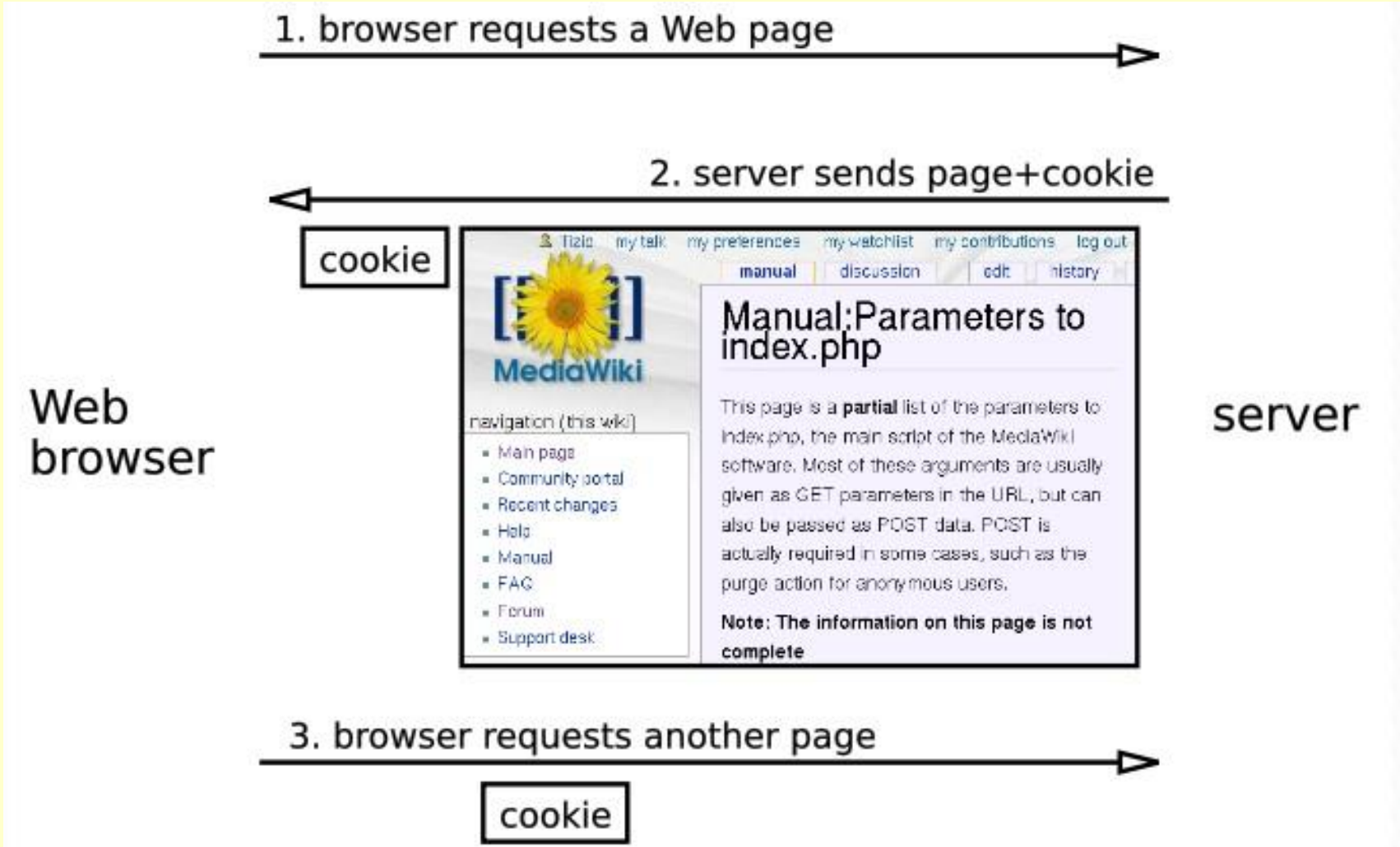
- When a server is interacting with many different browsers at the same time, the server needs to know *which* browser a particular request came from

- Request / Response initially was stateless - all browsers looked identical - this was really really bad and did not last very long at all.

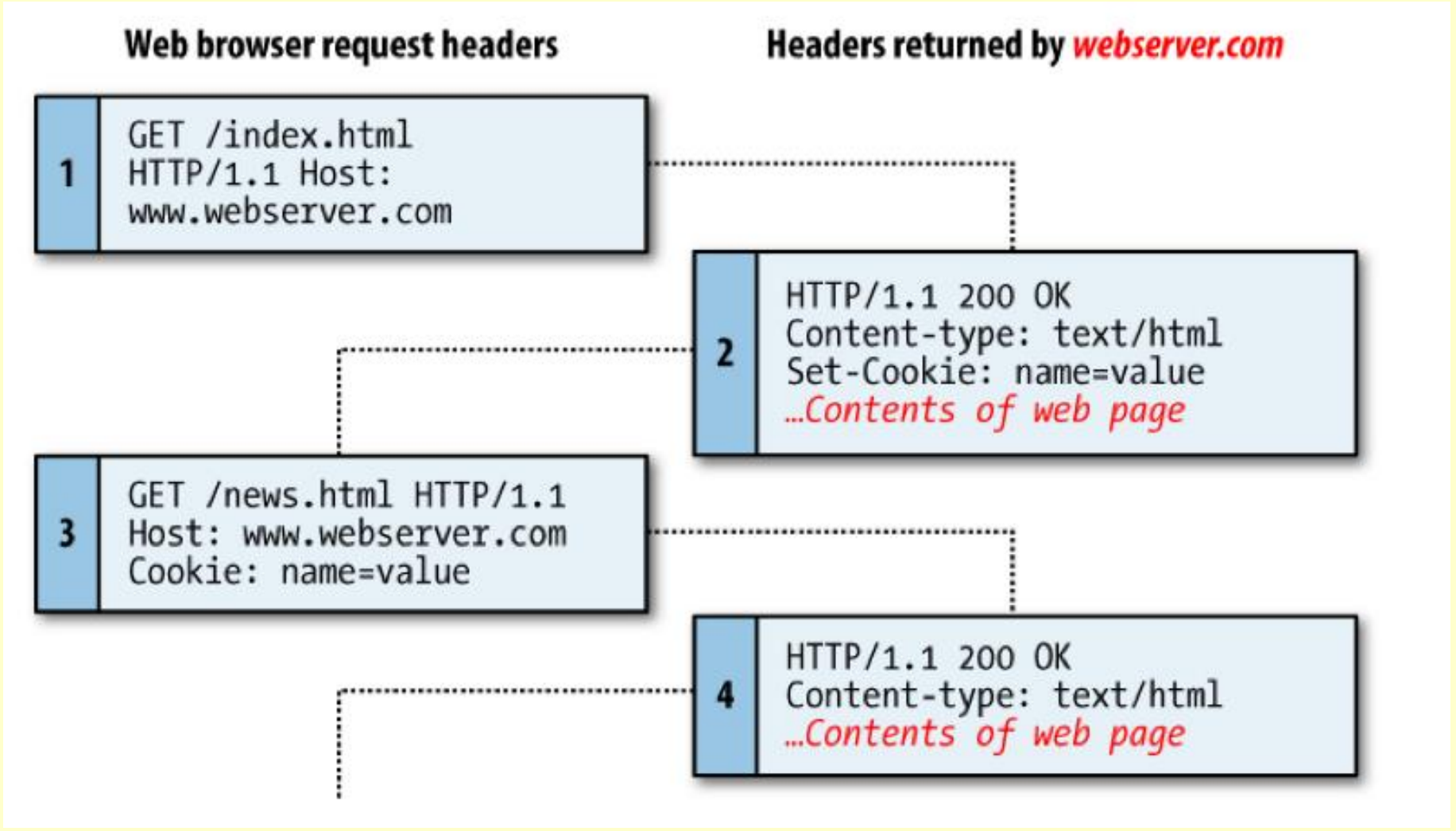
Web Cookies to the Rescue

Technically, cookies are arbitrary pieces of data chosen by the Web server and sent to the browser. The browser returns them unchanged to the server, introducing a state (memory of previous events) into otherwise stateless HTTP transactions. Without cookies, each retrieval of a Web page or component of a Web page is an isolated event, mostly unrelated to all other views of the pages of the same site.

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



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


Web browser request headers
Headers returned by webserver.com
1
GET /index.html
HTTP/1.1 Host:
www.webserver.com
2
HTTP/1.1 200 OK
Content-type: text/html
Set-Cookie: name=value
…Contents of web page
3
GET /news.html HTTP/1.1
Host: www.webserver.com
Cookie: name=value
4
HTTP/1.1 200 OK
Content-type: text/html
…Contents of web page

Cookies In the Browser

- Cookies are marked as to the web addresses they come from - the browser only sends back cookies that were originally set by the same web server
- Cookies have an expiration date - some last for years - others are short-term and go away as soon as the browser is closed

Cookies In the Browser

- A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

- A cookie is created, modified and can be deleted with the setcookie() function.

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

In The Server - Sessions

- In most server applications, as soon as we meet a new browser - we create a session
- We set a session cookie to be stored in the browser which indicates the session id in use
- The creation and destruction of sessions is handled by a web framework or some utility code that we just use to manage the sessions

Session Identifier

- A large, random number that we place in a browser cookie the first time we encounter a browser.
- This number is used to pick from the many sessions that the server has active at any one time.
- Server software stores data in the session which it wants to have from one request to another from the same browser.
    - Shopping cart or login information is stored in the session in the server

PHP Sessions

- We can establish / initialize a PHP Session by calling session_start() before any output has come out
- If the user has cookies set, we can use the array $_SESSION to store data from one request to the next with a particular browser
- We have a bit of data that persists from one request to the next

session_start

(PHP 4, PHP 5)

session_start — Initialize session data

Description

Report a bug

bool **session_start** (void)

session_start() creates a session or resumes the current one based on a session identifier passed via a GET or POST request, or passed via a cookie.

To use a named session, call session_name() before calling **session_start()**.

When session.use_trans_sid is enabled, the **session_start()** function will register an internal output handler for URL rewriting.

If a user uses *ob_gzhandler* or similar with ob_start(), the function order is important for proper output. For example, *ob_gzhandler* must be registered before starting the session.

http://php.net/manual/en/function.session-start.php

session_destroy

(PHP 4, PHP 5)

session_destroy — Destroys all data registered to a session

Description

Report a bug

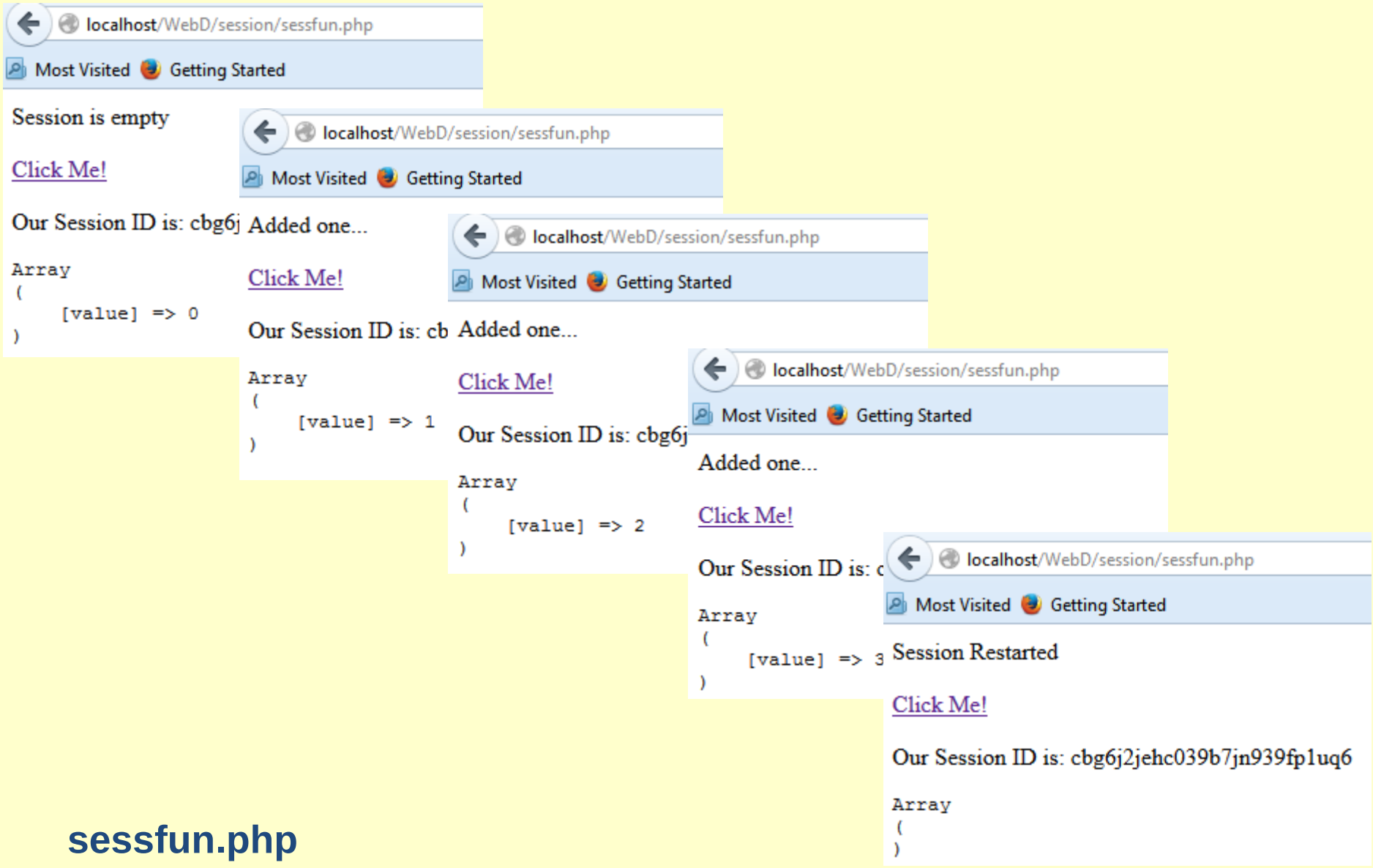
bool **session_destroy** (void)

session_destroy() destroys all of the data associated with the current session. It does not unset any of the global variables associated with the session, or unset the session cookie. To use the session variables again, session_start() has to be called.

In order to kill the session altogether, like to log the user out, the session id must also be unset. If a cookie is used to propagate the session id (default behavior), then the session cookie must be deleted. setcookie() may be used for that.

http://php.net/manual/en/function.session-destroy.php

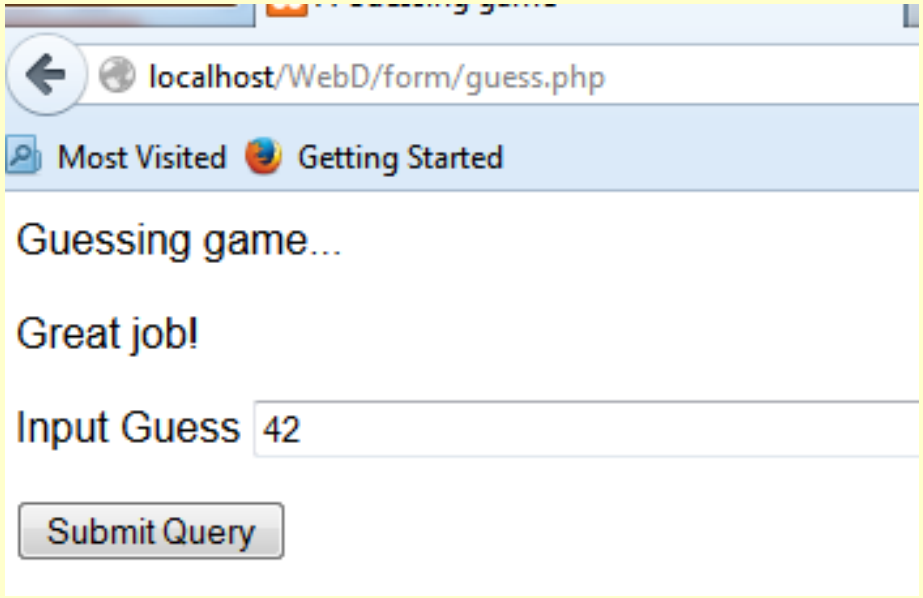
```
<?php
  // Note - cannot have any output before this
  session_start();
  if ( ! isset($_SESSION['value']) )
  {     echo("<p>Session is empty</p>\n");
  $_SESSION['value'] = 0;
} else if ( $_SESSION['value'] < 3 )
{
    $_SESSION['value'] = $_SESSION['value'] + 1;
       echo("<p>Added one...</p>\n");
  } else {session_destroy();
       session_start();
       echo("<p>Session Restarted</p>\n");  }
  ?>
  <p><a href="sessfun.php">Click Me!</a></p>
  <p>Our Session ID is: <?php echo(session_id());
?></p>
  <pre><?php print_r($_SESSION); ?></pre>
```


localhost/WebD/session/sessfun.php
Most Visited
Getting Started
Session is empty
Click Me!
Our Session ID is: cbg6j
Array
(
[value] => 0
)
localhost/WebD/session/sessfun.php
Most Visited
Getting Started
Added one...
Click Me!
Our Session ID is: cb
Array
(
[value] => 1
)
localhost/WebD/session/sessfun.php
Most Visited
Getting Started
Added one...
Click Me!
Our Session ID is: cbg6j
Array
(
[value] => 2
)
localhost/WebD/session/sessfun.php
Most Visited
Getting Started
Added one...
Click Me!
Our Session ID is: c
Array
(
[value] => 3
)
localhost/WebD/session/sessfun.php
Most Visited
Getting Started
Session Restarted
Click Me!
Our Session ID is: cbg6j2jehc039b7jn939fp1uq6
Array
(
)

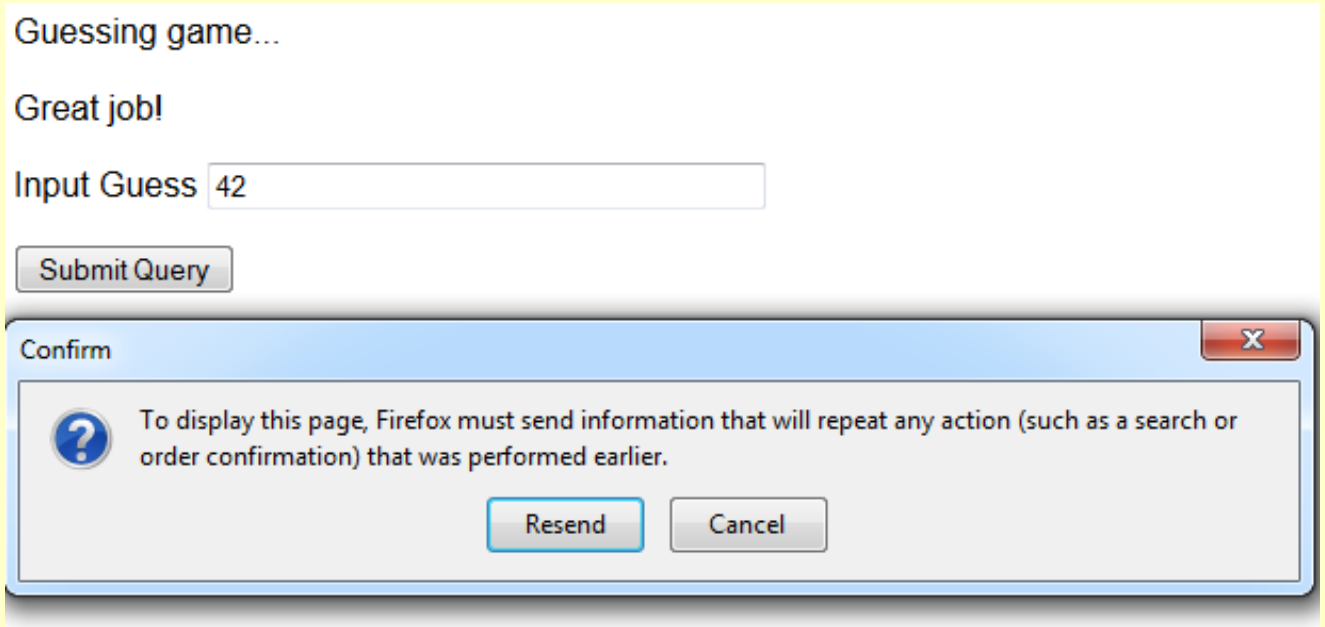
sessfun.php

POST / GET Gesture

- Once you do a POST, if you do refresh, the browser will re-send the POST data a second time
- The user gets a popup that tries to explain what is about to happen



Press Refresh



guess.php

No Double Posts

- Typically POST requests are adding or modifying data whilst GET requests view data
- It may be dangerous to do the same POST twice (say withdrawing funds from a bank account)
- So the browser insists on asking the user (out of your control)
- Kind of an ugly UI / bad usability

HTTP Location Header

- If your application has not yet sent any data, it can send a special header as part of the HTTP Response
- The redirect header includes a URL that the browser is supposed to forard itself to
- It was originally used for web sites that moved from one URL to another

http://en.wikipedia.org/wiki/URL_redirection

header

(PHP 4, PHP 5)

header — Send a raw HTTP header

Description

Report a bug

void **header** (string *$string* [, bool *$replace* = *true* [, int *$http_response_code*]])

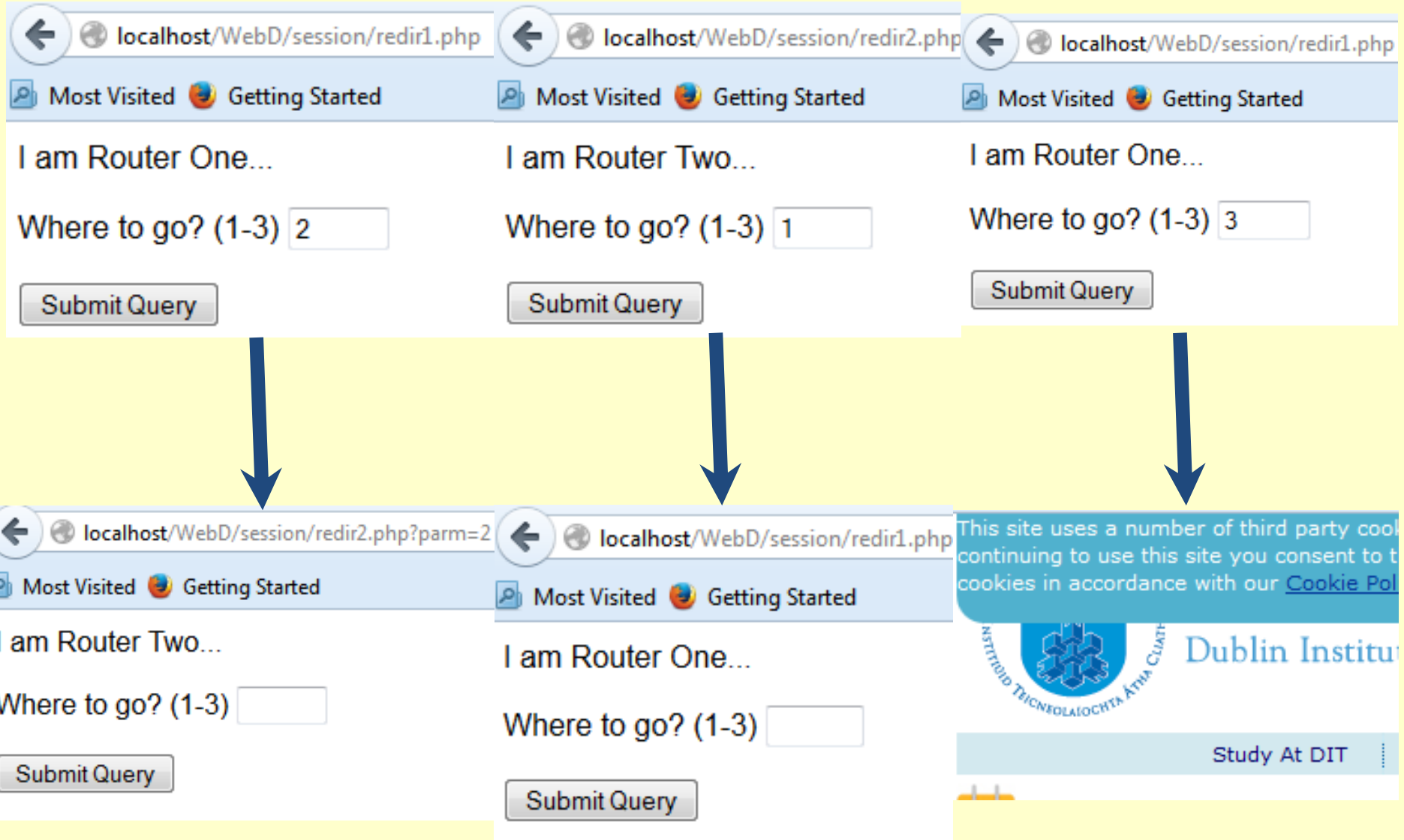
header() is used to send a raw HTTP header. See the » HTTP/1.1 specification for more information on HTTP headers.

Remember that **header()** must be called before any actual output is sent, either by normal HTML tags, blank lines in a file, or from PHP. It is a very common error to read code with include, or require, functions, or another file access function, and have spaces or empty lines that are output before **header()** is called. The same problem exists when using a single PHP/HTML file.

```
<html>
<?php
/* This will give an error. Note the output
 * above, which is before the header() call */
header('Location: http://www.example.com/');
?>
```

http://php.net/manual/en/function.header.php

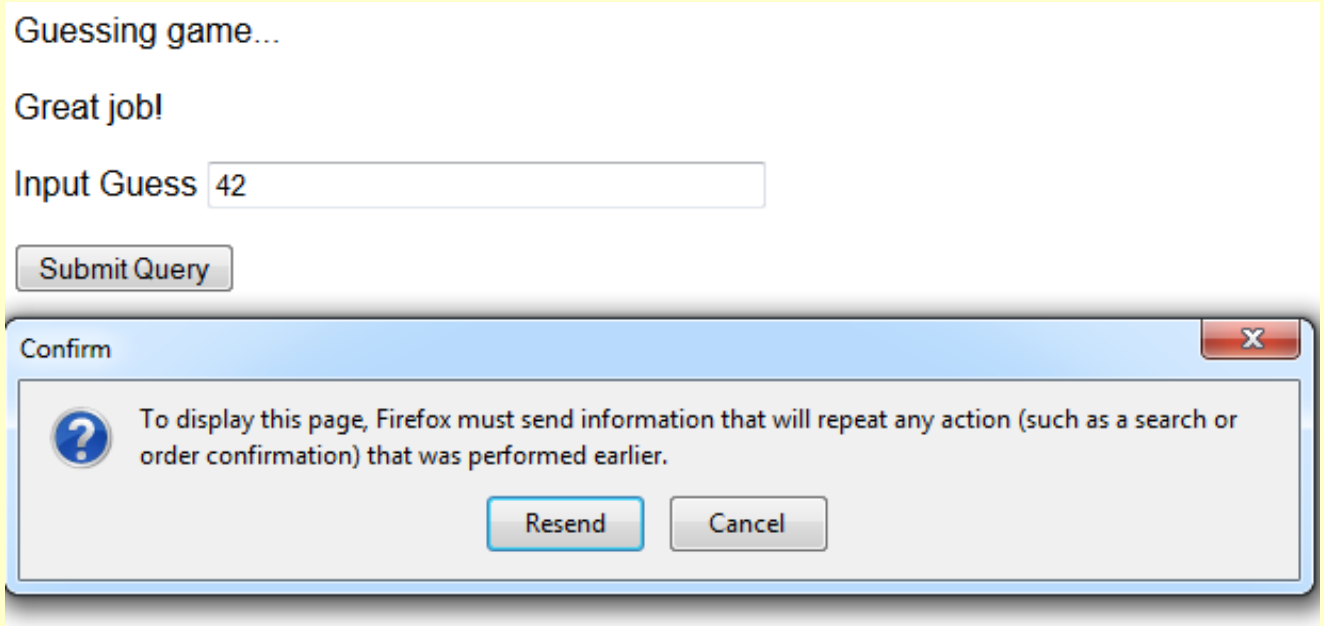
```php
<?php
    session_start();
    if ( isset($_POST['where']) ) {
        if ( $_POST['where'] == '1' ) {
            header("Location: redir1.php");
            return;
        } else if ( $_POST['where'] == '2' ) {
            header("Location: redir2.php?parm=2");
            return;
        } else {
            header("Location: http://www.dit.ie");
            return;
        }
    }
?>
<html>
<body style="font-family: sans-serif;">
<p>I am Router Two...</p>
<form method="post">
    <p><label for="inp9">Where to go? (1-3)</label>
    <input type="text" name="where" id="inp9" size="5"></p>
    <input type="submit"/>
</form>
</body>
```


localhost/WebD/session/redir1.php
Most Visited
Getting Started
I am Router One...
Where to go? (1-3) 2
Submit Query
localhost/WebD/session/redir2.php
Most Visited
Getting Started
I am Router Two...
Where to go? (1-3) 1
Submit Query
localhost/WebD/session/redir1.php
Most Visited
Getting Started
I am Router One...
Where to go? (1-3) 3
Submit Query
localhost/WebD/session/redir2.php?parm=2
Most Visited
Getting Started
I am Router Two...
Where to go? (1-3)
Submit Query
localhost/WebD/session/redir1.php
Most Visited
Getting Started
I am Router One...
Where to go? (1-3)
Submit Query
This site uses a number of third party cook
continuing to use this site you consent to t
cookies in accordance with our Cookie Pol
Dublin Institu
Study At DIT

POST Redirect Rule

- The simple rule for pages intended for a browser is to never generate a page with HTML content when the app receives POST data

- Must redirect somewhere - even to the same script - forcing the browser to make a GET after the POST



Review

```
<?php
$guess = '';
$message = false;
if ( isset($_POST['guess']) ) {
// Trick for integer / numeric parameters
$guess = $_POST['guess'] + 0;
if ( $guess == 42 ) {
$message = "Great job!";
} else if ( $guess < 42 ) {
$message = "Too low";
} else {
$message = "Too high...";
}
}
?>

<html>
<head>
<title>A Guessing game</title>
</head>
<body style="font-family: sans-serif;">
<p>Guessing game...</p>
<?php
if ( $message !== false ) {
echo("<p>$message</p>\n");
}
?>
<form method="post">
<p><label for="guess">Input Guess</label>
<input type="text" name="guess" id="guess" size="40"
<?php
echo 'value="' . htmlentities($guess) . '"';
?>
/></p>
<input type="submit"/>
</form>
</body>
```

guess.php

```
<?php
$guess = '';
$message = false;
if ( isset($_POST['guess']) ) {
// Trick for integer / numeric
$guess = $_POST['guess'] + 0;
if ( $guess == 42 ) {
$message = "Great job!";
} else if ( $guess < 42 ) {
$message = "Too low";
} else {
$message = "Too high...";
}
}
?>

<html>
<head>
<title>A Guessing game</title>
</head>
<body style="font-family: sans-
<p>Guessing game...</p>
<?php
if ( $message !== false ) {
echo("<p>$message</p>\n");
}
?>
<form method="post">
<p><label for="guess">Input Gue
<input type="text" name="guess"
<?php
echo 'value="' . htmlentities($
?>
/></p>
<input type="submit"/>
</form>
</body>
```

```
<?php                          (Review)
$guess = '';
$message = false;
if ( isset($_POST['guess']) ) {
// Trick for integer / numeric
parameters
$guess = $_POST['guess'] + 0;
if ( $guess == 42 ) {
$message = "Great job!";
} else if ( $guess < 42 ) {
$message = "Too low";
} else {
$message = "Too high...";
}
}
?>
<html>

.....
```

```php
<?php
$guess = '';
$message = false;
if ( isset($_POST['guess']) ) {
// Trick for integer / numeric
$guess = $_POST['guess'] + 0;
if ( $guess == 42 ) {
$message = "Great job!";
} else if ( $guess < 42 ) {
$message = "Too low";
} else {
$message = "Too high...";
}
}
?>

<html>
<head>
<title>A Guessing game</title>
</head>
<body style="font-family: sans-
<p>Guessing game...</p>
<?php
if ( $message !== false ) {
echo("<p>$message</p>\n");
}
?>
<form method="post">
<p><label for="guess">Input Gue
<input type="text" name="guess"
<?php
echo 'value="' . htmlentities($
?>
/></p>
<input type="submit"/>
</form>
</body>
```

```php
…?>                                    (Review)
<html>
<head>
<title>A Guessing game</title>
</head>
<body style="font-family: sans-serif;">
<p>Guessing game...</p>
<?php
if ( $message !== false ) {
echo("<p>$message</p>\n");
}
?>
<form method="post">
<p><label for="guess">Input Guess</label>
<input type="text" name="guess" id="guess"
size="40"
<?php
echo 'value="' . htmlentities($guess) .
'"';
?>
/></p>
<input type="submit"/>
</form>
</body>
```

(Improved)

```
<?php
session_start();
    if ( isset($_POST['guess']) ) {
            $guess = $_POST['guess'] + 0;
            $_SESSION['guess'] = $guess;
        if ( $guess == 42 ) {
                $_SESSION['message'] = "Great job!";
        } else if ( $guess < 42 ) {
                $_SESSION['message'] = "Too low";
        } else {
                $_SESSION['message'] = "Too high...";
        }
        header("Location: guess2.php");
        return;
    }
?>
<html>
```

guess2.php

(Improved)

```
<head><title>A Guessing game</title></head>
<body style="font-family: sans-serif;">
<?php
$guess = isset($_SESSION['guess']) ? $_SESSION['guess'] :'';
$message = isset($_SESSION['message'])?$_SESSION['message']:false;
?>
<p>Guessing game...</p>
<?php
if (
        $message!== false ) {
        echo("<p>$message</p>\n");
        }
?>
<form method="post">
        <p><label for="guess">Input Guess</label>
        <input type="text" name="guess" id="guess‟ size="40"
<?php
        echo 'value="' . htmlentities($guess) . '"';
?>
        /></p>
        <input type="submit"/>
</form>
</body>
```

guess2.php

Login / Logout

- Having a session is not the same as being logged in.
- Generally you have a session the instant you connect to a web site
- The Session ID cookie is set when the first page is delivered
- Login puts user information in the session (stored in the server)
- Logout removes user information from the session

```
<?php    session_start();
unset($_SESSION["account"]);
if ( isset($_POST["account"]) && isset($_POST["pw"]) )
      {    if ( $_POST['pw'] == 'dt228' )
             {      $_SESSION["account"] = $_POST["account"];
                    $_SESSION["success"] = "Logged in.";
                    header( 'Location: index.php' ) ;
                    return;
             } else {
                    $_SESSION["error"] = "Incorrect password.";
                    header( 'Location: login.php' ) ;
                    return;
}    } else if ( count($_POST) > 0 )
      {      $_SESSION["error"] = "Missing Required Information";
             header( 'Location: login.php' ) ;
             return;
      }
?>
<html>
```

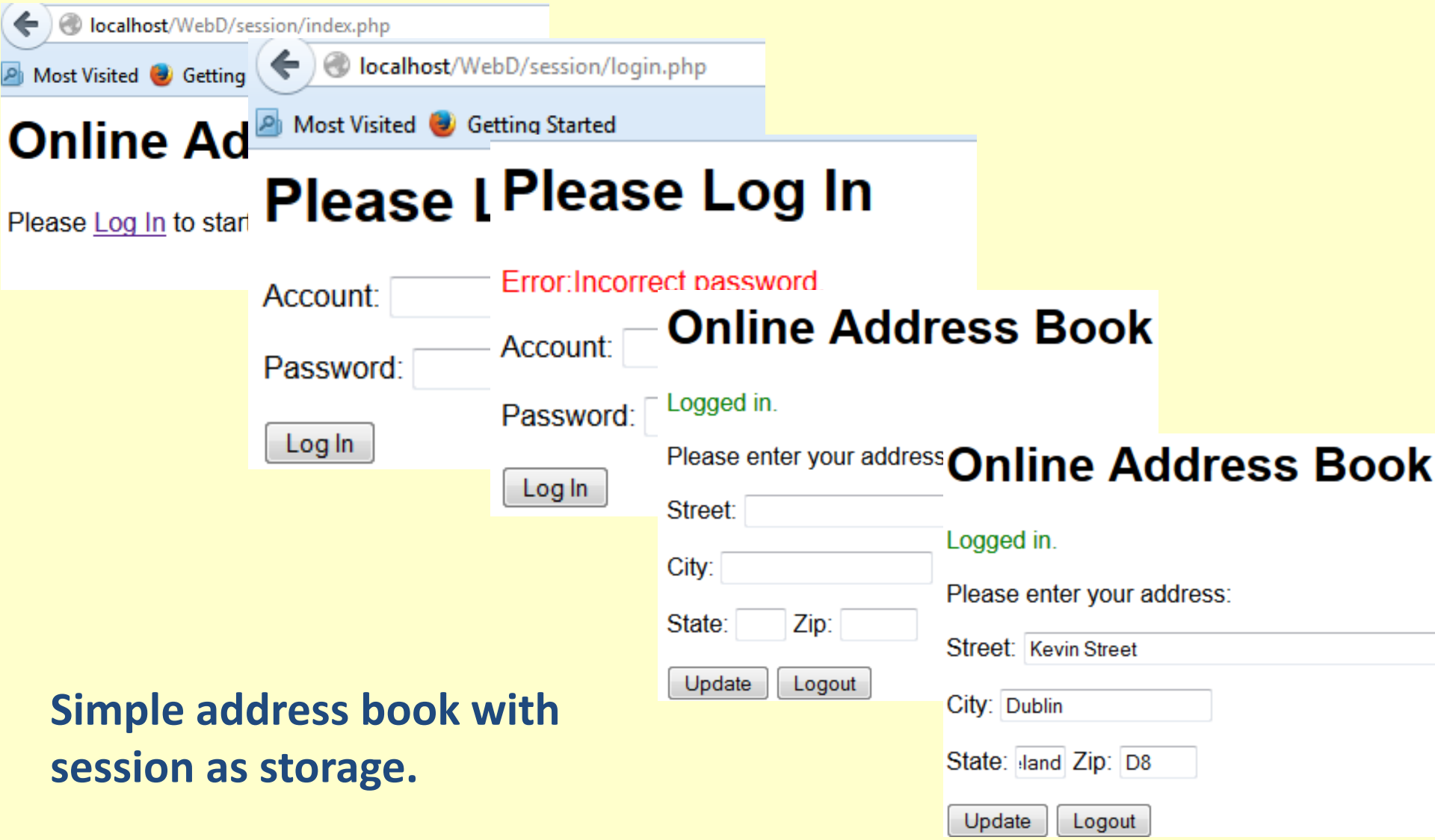
login.php

```
<html>
<head>
</head>
<body style="font-family: sans-serif;">
<h1>Please Log In</h1>
<?php
      if ( isset($_SESSION["error"]) ) {
      echo('<p style="color:red">Error:'.
      $_SESSION["error"]."</p>\n");
      unset($_SESSION["error"]);
      }
?>
<form method="post">
<p>Account: <input type="text" name="account" value=""></p>
<p>Password: <input type="text" name="pw" value=""></p>
<p><input type="submit" value="Log In"></p>
</form>
</body>
</html>
```

login.php

```php
<?php
session_start();
session_destroy();
header("Location: index.php");
```

logout.php



Simple address book with session as storage.

index.php

```
?>
<html>
<head>
</head>
<body style="font-family: sans-serif;">
<h1>Online Address Book</h1>
<?php
if ( isset($_SESSION["error"]) ) {
echo('<p
style="color:red">Error:'.$_SESSION["error"]."</p>\n");
unset($_SESSION["error"]);
}
if ( isset($_SESSION["success"]) ) {
echo('<p style="color:green">'.$_SESSION["success"]."</p>\n");
unset($_SESSION["success"]);
}
// Retrieve data from the session for the view
$street = isset($_SESSION['street']) ? $_SESSION['street'] :
'';
$city = isset($_SESSION['city']) ? $_SESSION['city'] : '';
$state = isset($_SESSION['state']) ? $_SESSION['state'] : '';
$zip = isset($_SESSION['zip']) ? $_SESSION['zip'] : '';
```

index.php

```
 if ( ! isset($_SESSION["account"]) ) { ?>
Please <a href="login.php">Log In</a> to start.
<?php } else { ?>
<p>Please enter your address:
<form method="post">
<p>Street: <input type="text" name="street" size="50"
value="<?php echo(htmlentities($street)); ?> "></p>
<p>City: <input type="text" name="city" size="20"
value="<?php echo(htmlentities($city)); ?>
"></p>
<p>State: <input type="text" name="state" size="2"
value="<?php echo(htmlentities($state)); ?> ">
Zip: <input type="text" name="zip" size="5"
value="<?php echo(htmlentities($zip)); ?> "></p>
<p><input type="submit" value="Update">
<input type="button" value="Logout"
onclick="location.href='logout.php'; return false "></p>
</form>
<?php } ?>
</body>
```

index.php

```
<?php
session_start();
if ( isset($_POST["street"]) && isset($_POST["city"]) &&
isset($_POST["state"]) && isset($_POST["zip"]) ) {
$_SESSION['street'] = $_POST['street'];
$_SESSION['city'] = $_POST['city'];
$_SESSION['state'] = $_POST['state'];
$_SESSION['zip'] = $_POST['zip'];
header( 'Location: index.php' ) ;
return;
} else if ( count($_POST) > 0 ) {
$_SESSION["error"] = "Missing Required Information";
header( 'Location: index.php' ) ;
return;
}
?>
<html>
```

Summery

- Cookies
- Sessions
- Sessions in PHP
- Login / Logout
- POST / Redirect Pattern