

CSCI 241

Sessions

Department of CS & QM
Winthrop University
 Fall 2011

Prepared by X. Wang

Topics

- What is a Session?
- Session Management
- Session Management in PHP
- Designing Session-Based Applications

Prepared by X. Wang

2

What is a Session?

- A fundamental characteristic of the Web is the stateless interaction between browsers and web servers
- This stateless nature suits applications that only browse or search collections of documents
- However, some applications require **multiple user interactions**, and there exists **information (states)** that must be shared among the interactions
- These applications cannot be implemented as a series of unrelated and stateless web pages

Prepared by X. Wang

3

What is a Session?

- These **multiple related user interactions** are called a **session**
- For example, the shopping cart for an online store cannot be implemented with stateless web pages because the items on a shopping cart must be stored somewhere for all the pages accessed by the customer during the shopping period
- The Web database applications with states can be built on the stateless environment using **sessions**

Prepared by X. Wang

4

Session Management

- There are two ways to keep the states (variables) of a session
 1. Store session variables in the browser and send them to the server with each request
 2. Store session variables on the server
- Storing session variables in the browser is usually a less attractive option because it requires additional network traffic, and is insecure
- We will discuss how to store session variables on the server

Prepared by X. Wang

5

Session Management

- Sessions are essential to most web database applications. **A session is related to an individual user**
- When storing session variables on the server, we can set an **ID** for representing each session
- A session has two components: **Session Variables** and **Session ID**
- **Session variables** are used to represent various states that are related to user's interactions in an application
- Session variables can be stored **at the web server or database server**, and are located using the **session ID**

Prepared by X. Wang

6

Session Management

- Using **session ID**, all of the session variables do not need to be transmitted over the Web
- Instead, the **session ID** is transmitted between the browser and server with each HTTP request and response
- The session ID is usually transmitted as a **cookie**
- When we store session variables at the server, we need to store them for all sessions from browsers
- Then, **how long** should the session variables be stored for?

Prepared by X. Wang

7

Session Management

- Simply, we can say storing the session variables until the end of the session
- However, how can the server know a session ends?
- Because HTTP is stateless, there is no way for the server to know when a session has ended
- Ideally, that the user logs out can explicitly ends the session
- But, a server can never be sure that a user is still there for using the session

Prepared by X. Wang

8

Session Management

- Unused sessions may consume many resources on the server and present a security risk
- So the server needs to set a **timeout** for each session
- If a session is timeout, the server will clean it up
- How long the timeout should be set depends on the needs of the application
- For example, the selected items on a shopping cart may be kept for a day, or a week as a wish list

Prepared by X. Wang

9

Session Management

- In summary, there are **three characteristics** of session management over the Web:
 1. Information or states must be stored (as **session variables**)
 2. Each HTTP request must carry an ID that allows the server to process the request with the correct session variables (transmitting **session ID**)
 3. Sessions need to have a **timeout**. This can guarantee that the old sessions will be cleaned up

Prepared by X. Wang

10

Session Management in PHP

- Developing applications that use PHP sessions is straightforward. The previous **three characteristics** of session management are mostly taken care of by the PHP session management
- **Session variables** in PHP can be configured to be stored in **files**, **memory**, or **database**
- By default, PHP stores **session variables in files**
- Using files for storing session variables usually requires that the number of concurrent sessions is not too large
- If it is too large, we should consider to store session variables into database

Prepared by X. Wang

11

Session Management in PHP

- **Starting a Session**
- The **`session_start()`** function is used to create a new session
- The **first time** a script calls **`session_start()`**, PHP generates a new **session ID** and creates an empty file to store **session variables**
- PHP also sends a **cookie** back to the browser to store the **session ID** at the client-side

Prepared by X. Wang

12

Session Management in PHP

Starting a Session

- Because the cookie is sent as part of the HTTP headers in the response to browser, we need to call `session_start()` before any other output is generated, just as that we call the `header()` function before
- The session ID is a random string of 32 hexadecimal digits generated by PHP
- When a new session is started, PHP creates a session file and uses the session ID prefixed with `sess_` for the filename

Prepared by X. Wang

13

Session Management in PHP

Setting Session Variables

- After starting a session, we can store session variables to the session file by storing the session variables to the PHP global array variable `$_SESSION`
- For example, for a login session in an online store, we can keep the `cust_id` as a session variable for the session
- Through this `cust_id` session variable, we can get any information for the customer from the database
- If a `cust_id` is currently in a variable `$cust_id`, to keep `$cust_id` as the session variable `cust_id`, we can do:


```
$_SESSION["cust_id"] = $cust_id;
```

Prepared by X. Wang

14

Session Management in PHP

Using Session Variables

- The `session_start()` function is also used to find an existing session
- If the `session_start()` is called, and a session has been started previously, PHP will automatically use the session cookie to find the corresponding session file, and then set the `$_SESSION` array with the session variables stored in the session file
- In the PHP scripts, we can then use session variables stored in the `$_SESSION` array to work coordinately among related session pages

Prepared by X. Wang

15

Session Management in PHP

Using Session Variables

- A call to the `session_start()` will either create a new session or find an existing session
- Also, we get an initialized global array `$_SESSION`
- After a session is started, we can access any session variables in `$_SESSION`, and we can also set any new session variables to the `$_SESSION` array
- Any modification to the `$_SESSION` array will be written to the session file by PHP automatically

Prepared by X. Wang

16

Session Management in PHP

Unsetting Session Variables

- To unset a session variable, we can call the `unset()` function
- For example, to unset the session variable `cust_id`, we can do: `unset($_SESSION["cust_id"]);`
- To unset all session variables, we can unset the whole `$_SESSION` array, or re-assign a new array
- For example, `$_SESSION = array();` will unset all session variables without destroying the session

Prepared by X. Wang

17

Session Management in PHP

Destroying a Session

- At some points in an application, sessions should be destroyed
- For example, when a customer logs out of the bookstore, we should destroy the customer's login session
- To destroy a session, we can call the `session_destroy()` function
- A call to `session_destroy()` removes the session file (of course, also removes the `$_SESSION` array)

Prepared by X. Wang

18

Session Management in PHP

- ❑ Destroying a Session
- But it **doesn't** remove the session cookie from the browser
- Leaving the session cookie at the client-side doesn't disturb the session operation at the server-side
- After destroying a session, the script should **redirect to another receipt page**
- This can **avoid the reload problem** (re-destroy the same session)

Prepared by X. Wang

19

Session Management in PHP

- ❑ Garbage Collection
- With the `session_destroy()` function, we can destroy a session
- We can provide a link (like logout) in a web application to allow the user explicitly destroy a session
- But there is no guarantee that a user will log out by selecting the appropriate PHP script (the link)
- If no explicit call to `session_destroy()`, PHP should provide a mechanism that **ensures unused session will be cleaned up in a certain time**

Prepared by X. Wang

20

Session Management in PHP

- ❑ Garbage Collection
- In the `php.ini` file, there are three parameters to control the garbage collection
- `session_gc_maxlifetime`: the default value is 1440 seconds (24 minutes). After this time, the session will be seen as "garbage", and cleaned up by the garbage collection process
- `session_gc_probability` and `session.gc_divisor` are two parameters to control the probability when the garbage collection process should be started

Prepared by X. Wang

21

Design Session-Based Applications

- Applications that require a user to log in (such as online banking, online shopping stores) often use sessions to manage the user interactions
- For a session-based application, we need to know
 - When and how to start a session
 - What data needs to be stored as session variables
 - When to destroy a session

Prepared by X. Wang

22

Design Session-Based Applications

- A login process usually includes **three pages**:
 1. The first page collects the user credentials (**username** and **password**) using an HTML form and passes them to the second page using the **POST** method
 2. The second page is responsible for **verifying** the user **credentials** and **creating a new session** with the first call to `session_start()` and **setting up the initial session variables**. After the session is started, the page **sends a Location header** to the browser to relocate to the third page. This is to avoid the Reload problem
 3. The third page is a **welcome page**. It just displays a login welcome message

Prepared by X. Wang

23

Design Session-Based Applications

- Compared to the login process, after the user finishes the session, the user needs to log out the session
- A logout process usually includes **two pages**:
 1. The first page is responsible for **destroying** the current session. After the session is destroyed, the page **sends a Location header** to the browser to relocate to the second page. This is to avoid the Reload problem
 2. The second page is a **receipt page**. It just displays a message to thank the user for using the site

Prepared by X. Wang

24

Design Session-Based Applications

- Now let's see the login process for our online bookstore
- The following is the first page (login.php). It only contains an HTML form

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>Login</title></head>
<body>
<p>Please login with the following information</p>
<form name="form1" id="form1" method="post" action="loginvalidate.php">
Your Email: <input type="text" name="email" size="40" /><br />
Password: &nbsp;&nbsp;&nbsp;<input type="password" name="password" size="40" /><br />
<br /><input name="submit" type="submit" value="Login" />
(For a new customer, please <a href="register.php">Register</a> first)
</form>
</body>
</html>
```

Prepared by X. Wang

25

Design Session-Based Applications

- The page (login.php) will generate a login form as follows



Prepared by X. Wang

26

Design Session-Based Applications

- The following is the second page (loginvalidate.php)
- Please note that we call the `ob_start()` function at the beginning
- This is because we need to create a new session by calling the `session_start()` function if the user information passes the validation, and PHP requires there is no output before the `session_start()` function call
- When creating the session, we set three session variables for storing `cust_id`, `firstname` and `lastname` of the customer

Prepared by X. Wang

27

Design Session-Based Applications

- The second page (loginvalidate.php) is as follows

```
<?php ob_start(); ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>Login</title></head>
<body>
<?php
require "db.php";
if (!isset($_POST["email"]) || !isset($_POST["password"]))
die("Invalid operation!!!");
$goback = "<br /><br />Please <a href='\"login.php\"'>go back</a> and try again";
if (empty($_POST["email"]))
die("<br />The <b>Email</b> field cannot be blank($goback)");
if (empty($_POST["password"]))
die("<br />The <b>Password</b> field cannot be blank($goback)");
$email = trim($_POST["email"]);
$password = trim($_POST["password"]);
```

Prepared by X. Wang

28

Design Session-Based Applications

```
if (!($connection = @mysql_connect($hostname, $dbusername, $dbpassword)))
die("Could not connect to the database system");
if (!(@mysql_select_db($databasename, $connection))
showerror());

$query = "SELECT * FROM customers WHERE email='{$email}' AND
password='{$password}'";
if (!($result = @mysql_query($query, $connection))) showerror();
if (mysql_num_rows($result) == 0)
die("<br />Invalid email or password<br />($goback)");

$row = @mysql_fetch_array($result);
session_start();
$_SESSION["cust_id"] = $row["cust_id"];
$_SESSION["lastname"] = $row["lastname"];
$_SESSION["firstname"] = $row["firstname"];
header("Location: welcome1.php");
?>
</body>
</html>
```

Prepared by X. Wang

29

Design Session-Based Applications

- The third page is called `welcome1.php`
- It is modified from the `welcome.php` page in the register process
- The difference is that we fetch the `cust_id` from the `$_SESSION` array (not from `$_GET` array) since the `cust_id` has already been saved as a session variable
- To let PHP generate the `$_SESSION` array, we need to call the `session_start()` function again at the beginning of the PHP page

Prepared by X. Wang

30

Design Session-Based Applications

- The third page (`welcome1.php`) is as follows

```
<?php session_start(); ?>
<!DOCTYPE html PUBLIC "-//W3C/DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>Welcome</title></head>
<body>
<?php
if (!isset($_SESSION["cust_id"]))
    die("Invalid operation!!<br /> Please login first");

require "db.php";

$scust_id = $_SESSION["cust_id"];
```

Prepared by X. Wang

31

Design Session-Based Applications

```
if (!($connection = @ mysql_connect($hostname, $dbusername, $dbpassword)))
    die("Could not connect");
if (!(@ mysql_select_db($databasename, $connection))) showerror();

$query = "SELECT * FROM customers WHERE cust_id={$scust_id}";
if (!($result = @ mysql_query ($query, $connection))) showerror();
$num_result = mysql_num_rows($result);
if ($num_result == 0)
    die("!!!Invalid operation!!!<br /> Please login first");

$row = @ mysql_fetch_array($result);

print "<p>Hello, {$row['firstname']} {$row['lastname']}<br />";
print "Welcome to this online bookstore!</p>";

?>
</body>
</html>
```

Prepared by X. Wang

32

Design Session-Based Applications

- Now let's see the logout operation for our online bookstore
- The following is the first page (`logout.php`) for the logout. It only contains the PHP code
- We destroy the current session, and then jump to display the `logout.html` page to avoid the reload problem

```
<?php
session_start();
session_destroy();
header("Location: logout.html");
?>
```

Prepared by X. Wang

33

Design Session-Based Applications

- The `logout.html` only contains HTML code

```
<!DOCTYPE html PUBLIC "-//W3C/DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Logout from the Online Bookstore</title>
</head>
<body>
<p><b>Thank you for shopping in this online bookstore!</b></p>
</body>
</html>
```

Prepared by X. Wang

34