



Universiteti i Prishtinës

Programimi në Internet

Session Control in PHP

Dr. Ing. Lule Ahmedi

In This Lecture

- ⑥ What session control is
- ⑥ Cookies
- ⑥ Setting up a session
- ⑥ Session variables
- ⑥ Sessions and authentication

What Session Control Is

HTTP - a *stateless* protocol!

- ⑥ No built-in way of maintaining state between two transactions
 - △ E.g., that the two subsequent HTTP requests came from the same client

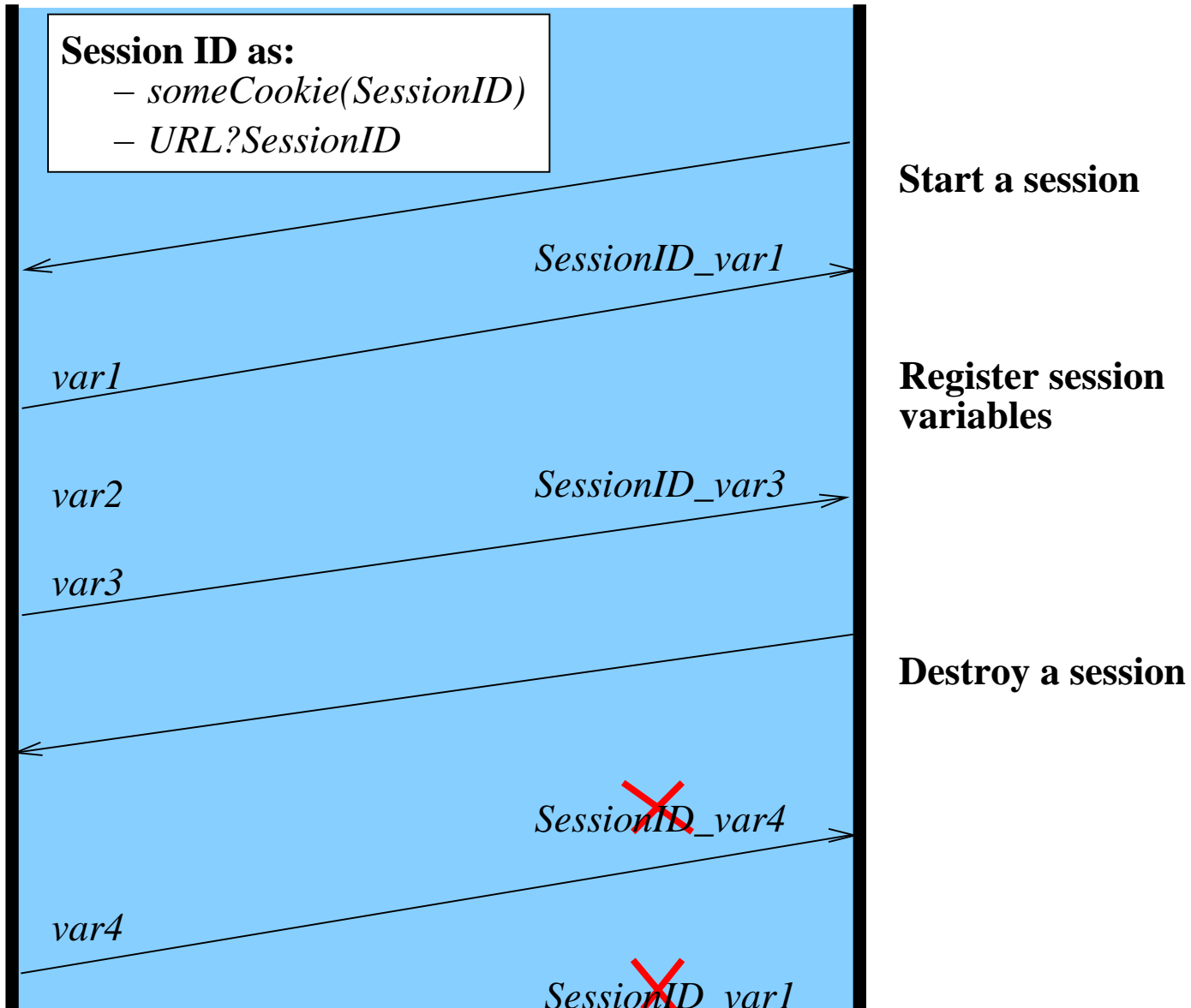
The idea behind session control

- ⑥ To track a user during a single session on a Web site
 - △ E.g., use the login information for authentication or personal profilization

Basic Session Functionality

Web Browser

PHP Server



The Session ID

- ⑥ A randomly generated number, encrypted
- ⑥ Serves as a key that allows particular variables to be registered as session variables
- ⑥ Stored for the lifetime of the session
 - △ As a random looking string appended to the URL
 - △ In form of a cookie
 - △ Embedded (manually) in links to be passed along
`<a href="link.php?<?=SID?">`

What is a Cookie

A small piece of information stored on a client-side machine

Setting cookies:

- ⑥ As HTTP header data

```
Set-Cookie:  NAME=VALUE; [expires=DATE;]  
[path=PATH;] [domain=DOMAIN_NAME;] [secure]
```

- ⑥ From within PHP

```
int setcookie (string name [, string value [,  
int expire [, string path [, string domain [,  
int secure]]]])
```

- ⑥ `setcookie ("mycookie", "value");`

Accessing and Deleting Cookies

- ⑥ Access a cookie through `$NAME`, or `$HTTP_COOKIE_VARS[$NAME]`
- ⑥ Delete a cookie by calling `setcookie()` again with the same cookie name and an expiry time in the past

When a browser connects to an URL, it first searches the cookies stored locally

- ⑥ If any of them are relevant to the URL being connected to, they will be transmitted back to the server

Cookies and A Session

No need to manually set cookies

- ⑥ The session functions will take care of this for you
 - △ `session_get_cookie_params()` - to see the contents of the cookie set by session control
 - .. Returns an associative array containing the elements `lifetime`, `path`, and `domain`.
 - △ `session_set_cookie_params($lifetime, $path, $domain)` - to set the session cookie parameters.

Session Implementation Steps

1. Start a session
2. Register session variables
3. Use session variables
4. Deregister variables and destroy a session

Step 1: Start a Session

- ⑥ With `session_start()`, or
 - ⑥ Directly when calling `session_register()`, or
 - ⑥ Automatically when your site is visited (in `php.ini`)
-
- ⑥ If there is already a current session ID, it essentially loads the registered session variables so that you can use them
 - ⑥ If not, it will create one

Step 2: Register Session Variables

Such that a variable can be tracked from one script to another

With `session_register()`

- ⑥ E.g., to register the variable `$myvar`:

```
$myvar = 5; session_register("myvar");
```

- △ This will record the variable name and track its value

The variable lifetime:

- ⑥ Until the session ends, or until it is deregistered manually

To register more than one variable at once:

- ⑥ Provide a comma-separated list of variable names

```
session_register("myvar1", "myvar2");
```

Step 3: Using Session Variables

To bring a session variable into scope so that it can be used

- ⑥ First start a session
- ⑥ Then access the variable
 - △ Via the short form name, e.g., `$myvar` (if `register_globals` turned on)
 - △ Via the associative array `$HTTP_SESSION_VARS`, e.g.,
`$HTTP_SESSION_VARS["myvar"]`

A good security feature:

- ⑥ A session variable cannot be overridden by GET or POST data

Step 3: Using Session Variables (cont.)

Checking if session variables have been set:

- ⑥ Via `isset()`, or `empty()`
- ⑥ Caution: Variables could have been set by the user via GET or POST

Check if a variable is a registered session variable:

- ⑥ By calling the `session_is_registered()` function
 - △ `$result = session_is_registered("myvar");`
 - △ This will check whether `$myvar` is a registered session variable and return true or false
- ⑥ By checking the `$HTTP_SESSION_VARS` array for a variable's existence

Step 4: Deregister Variables, Destroy a Session

Deregister session variables:

- ⑥ Call the `session_unregister()` function
`session_unregister("myvar");`
 - △ Can only deregister a single session variable at a time (unlike `session_register()`)
 - △ `session_unset()` - to deregister all the current session variables

Clean up the session ID:

- ⑥ Call `session_destroy()`

A Session Example: Authentication

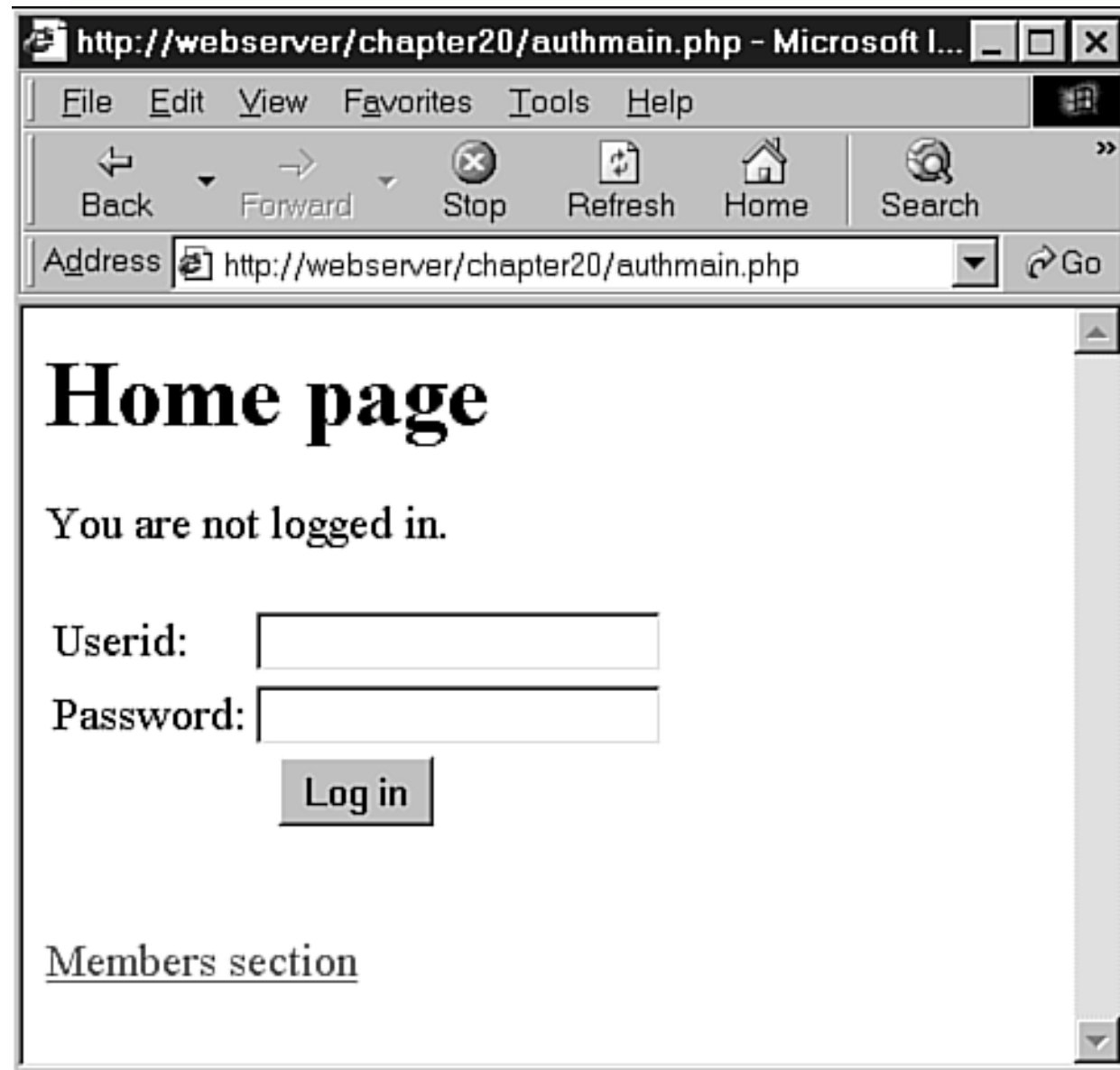
To keep track of users after they have been authenticated via a login form, combine

- ⑥ The information about registered accounts on a MySQL database
- ⑥ The session control mechanism

The example to come consists of three script snippets:

1. The page with the login form and authentication for the site's members
2. The page that displays content only to members who have passed the authentication process successfully
3. The logout page

Authentication Via a Login From



The screenshot shows a Microsoft Internet Explorer window with the title bar "http://webserver/chapter20/authmain.php - Microsoft I...". The address bar contains "http://webserver/chapter20/authmain.php". The main content area displays the following:

Home page

You are not logged in.

Userid:

Password:

Members section

Checking State: Unauthorized Access



Checking State: Authorized Access



The Main Authentication Script

```
<?php
session_start();

if ($userid && $password) {
    // if the user has just tried to log in

    $db_conn = mysql_connect("localhost", "webauth", "webauth");
    mysql_select_db("auth", $db_conn);
    $query = "select * from auth "
              ."where name='$userid' "
              ." and pass=password('$password')";
    $result = mysql_query($query, $db_conn);
    if (mysql_num_rows($result) >0 ) {
        // if they are in the database register the user id
        $valid_user = $userid;
        session_register("valid_user");
    }
}
?>
```

The First Time in a Session

- ⑥ A `session_start()` registers session variables with `session_register`
 - △ In our example, it is the `$userid` variable registered as `$valid_user`

The Main Authentication Script (cont.)

```
<html><body><h1>Home page</h1>
<?php
    if (session_is_registered("valid_user")) {
        echo "You are logged in as: $valid_user <br>";
        echo "<a href=\"logout.php\">Log out</a><br>";
    } else {
        if (isset($userid)) { // tried, failed to log in
            echo "Could not log you in";
        } else { // not tried to log in or have logged out
            echo "You are not logged in.<br>";
        }

        // provide form to log in
        echo "<form method=post action=\"authmain.php\">";
        echo "<table><tr><td>Userid:</td>";
        echo "<td><input type=text name=userid></td></tr>";
        echo "<tr><td>Password:</td>";
        echo "<td><input type=password name=password></td></tr>";
        echo "<tr><td colspan=2 align=center>";
        echo "<input type=submit value=\"Log in\"></td></tr>";
        echo "</table></form>";
    }
?>
<br><a href="members_only.php">Members only</a></body></html>
```

Any Other Time Afterwords in a Session

- ⑥ Just reload the registered session variables as needed within a given session
- ⑥ This is what is meant by maintaining the state as long as the session is alive

Reloading Session Variables

```
<?php
    session_start();

    $old_user = $valid_user; // store to test if logged in
    $result = session_unregister("valid_user");
    session_destroy();
?>
<html><body><h1>Log out</h1>
<?php
    if (!empty($old_user)) {
        if ($result) {
            // if they were logged in and are not logged out
            echo "Logged out.<br>";
        } else {
            // they were logged in and could not be logged out
            echo "Could not log you out.<br>";
        }
    } else {
        // if they weren't logged in but came to this page somehow
        echo "You're not logged in, so have'nt been logged out.<br>";
    }
?>
```