HERAT UNIVERSITY
COMPUTER SCEINE FACULTY

Web Engineering II (Semester 6)

(Maintaining State in PHP Part II - Sessions)

LECTURER: HAMED AMIRY

So…

Cookies	Sessions
Limited storage space	Practically unlimited space
Insecure storage client-side	Reasonably securely stored server-side
User controlled	No user control

How do ‘Sessions’ work?

They are based on assigning each user a unique number, or **session id**. Even for extremely heavy use sites, this number can for all practical purposes can be regarded as **unique.**

e.g.

26fe536a534d3c7cde4297abb45e275a

How do ‘Sessions’ work?

This **session id** is stored in a cookie, or passed in the URL between pages while the user browses.

The data to be stored (e.g. name, log-in state, etc.) is stored **securely server-side in a PHP superglobal**, and referenced using the session id.

Crucially, sessions are **easy** to implement as **PHP does all the work!**

Starting or Resuming a Session

```
session_start();
```

PHP does all the work: It looks for a valid session id in the **$_COOKIE** or **$_GET** superglobals – if found it initializes the data. If none found, a new session id is created. Note that like **setcookie()**, this function must be called before any echoed output to browser.

Starting or Resuming a Session

```
session_start();
```

When doing anything with sessions, this is always called first!

Storing Session Data

The **$_SESSION** superglobal array can be used to store any session data.

e.g.

```
$_SESSION['name'] = $name;
$_SESSION['age'] = $age;
```

Reading Session Data

Data is simply read back from the **$_SESSION** superglobal array.

e.g.

```
$name = $_SESSION['name'];
$age = $_SESSION['age'];
```

Session Propagation

Sessions need to pass the session id between pages as a user browses to track the session.

It can do this in two ways:

- Cookie propagation
- URL propagation

Cookie Propagation

A cookie is stored on the users PC containing the session id.

It is read in whenever `session_start();` is called to initialize the session.

Default behaviour is a cookie that expires when the browser is closed. Cookie properties can be modified with `session_set_cookie_params` if required.

URL Propagation

The session id is propagated in the URL

(…**some_folder/index.php?sid=26fe536a534d3c7cde4297abb45e275a**)

PHP provides a global constant to append the session id to any internal links, **SID**.

e.g.

```
<a href="nextpage.php?<?=SID?>">Next page</a>
```

Which one..?

The default setup of a PHP server is to use **both** methods.

- it checks whether the user has cookies enabled.
- If cookies are on, PHP uses cookie propagation. If cookies are off it uses URL propagation.

And this means..?

That as developers, we must be aware that sessions can be propagated through URL, and append the constant **`SID`** to any internal links.

If sessions are being propagated by cookies, the constant **`SID`** is an empty string, so the session id is not passed twice.

Destroying a Session

Often not required, but if we want to destroy a session:

```
// clear all session variables
$_SESSION = array();
// delete the session cookie if there is one
if (isset($_COOKIE[session_name()])) {
 setcookie(session_name(),'',time()-42000,'/');
}
// destroy session
session_destroy();
// avoid reusing the SID by redirecting
// back to the same page to regenerate session
header('Location: '.$_SERVER['PHP_SELF']);
```

Session Expiry

By default, PHP sessions expire:

- after a certain length of inactivity (default 1440s), the PHP garbage collection processes deletes session variables. Important as most sessions will not be explicitly destroyed.
- if propagated by cookies, default is to set a cookie that is destroyed when the browser is closed.
- If URL propagated, session id is lost as soon as navigate away from the site.

Long-term Sessions

Although it is possible to customize sessions so that they are maintained after the browser is closed, for most practical purposes PHP sessions can be regarded as short-term.

Long-term session data (e.g. ‘remember me’ boxes) is usually maintained by explicitly setting and retrieving cookie data.

Session Hi-jacking

A security issue: if a malicious user manages to get hold of an active session id that is not their own..

e.g.

- user 1 browsing site with cookies disabled (URL propagation).
- user 1 logs in.
- user 1 sends an interesting link to user 2 by email.. The URL copy and pasted contains his session id.
- user 2 looks at the link before session id is destroyed, and ‘hijacks’ user 1’s session.
- user 2 is now logged in as user 1!!

… rule of thumb …

If you are truly security conscious you should assume that a session propagated by URL may be compromised. Propagation using cookies is more secure, but still not foolproof..


thank you
تشکر فراوان . . .