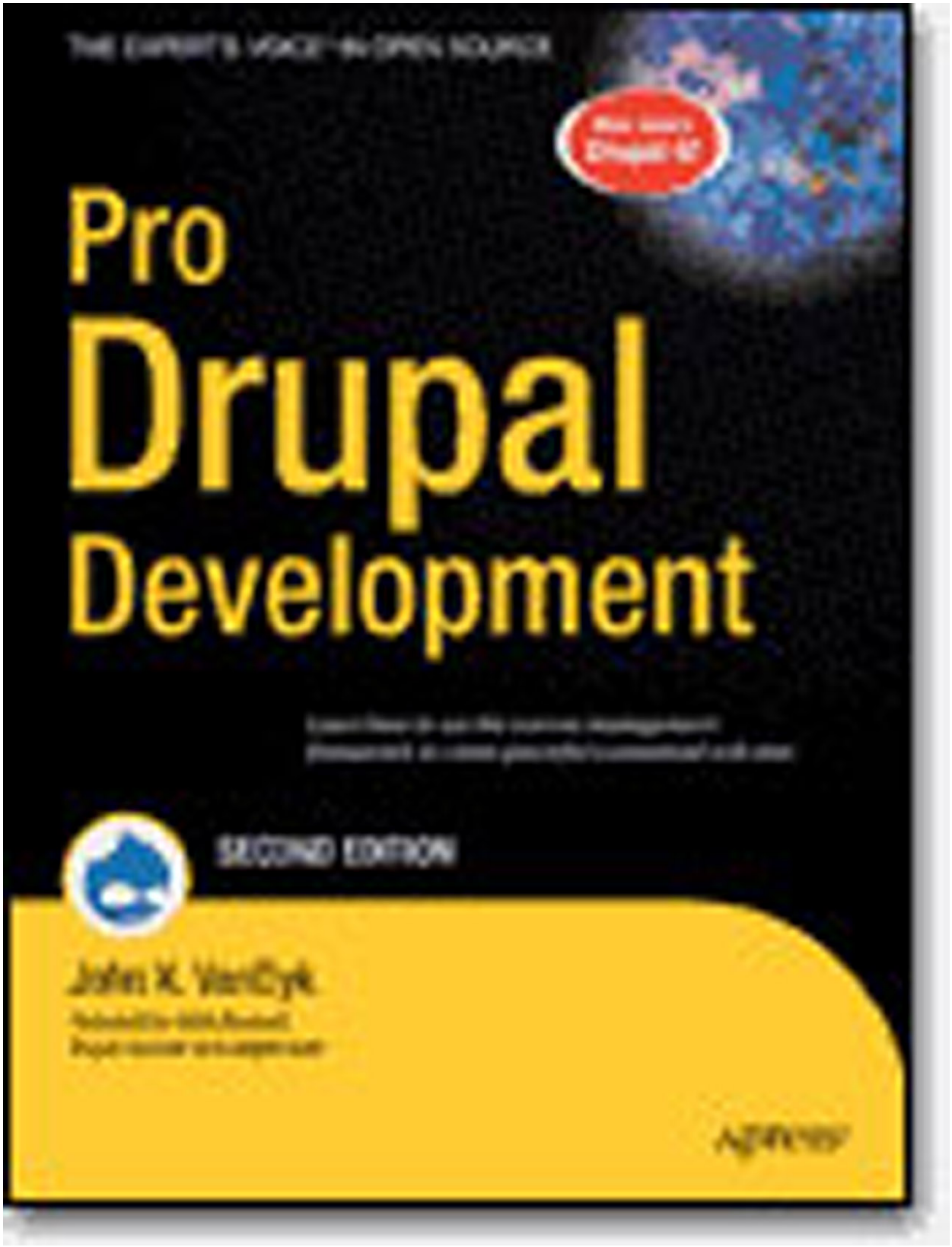
THE EXPERT'S VOICE® IN OPEN SOURCE
Pro
Drupal
Development
SECOND EDITION
John K. VanDyk

Table of Contents

Pro Drupal Development, Second Edition By John K. VanDyk ISBN: 9781430209898
Publisher: Apress
Print Publication Date: 2008/08/01

CHAPTER 16

Sessions

HTTP is a stateless protocol, which means that each interaction between the web browser and server stands alone. So how do you track a user as he or she navigates through a series of web pages on a web site? You use sessions. Starting with version 4, PHP offers built-in support for sessions via the session family of functions. In this chapter, you'll see how Drupal uses PHP's sessions.

What Are Sessions?

When a browser first requests a page from a Drupal site, PHP issues the browser a cookie containing a randomly generated 32-character ID, called `PHPSESSID` by default. This is done by the inclusion of one line in the HTTP response headers sent to the browser the first time it visits the site:

```
HTTP/1.1 200 OK
Date: Thu, 17 Apr 2008 20:24:58 GMT
Server: Apache
Set-Cookie: PHPSESSID=3sulj1mainvme55r8udcc6j2a4; expires=Sat, 10 May 2008 23:58:19
  GMT; path=/
Last-Modified: Thu, 17 Apr 2008 20:24:59 GMT
Cache-Control: store, no-cache, must-revalidate
Cache-Control: post-check=0, pre-check=0
Content-Type: text/html; charset=utf-8
```

On subsequent visits to the site, the browser presents the cookie to the server by including it in each HTTP request:

```
GET / HTTP/1.1
User-Agent=Mozilla/5.0 (Macintosh; U; Intel Mac OS X; en-US; rv:1.8.1.14)
  Gecko/20080404 Firefox/2.0.0.14
Cookie: PHPSESSID=3sulj1mainvme55r8udcc6j2a4
```

This allows PHP to keep track of a single browser as it visits the web site. The 32-character ID, known as the *session ID*, is used as the key to the information Drupal stores about the session and allows Drupal to associate sessions with individual users.

Usage

Drupal uses sessions for several important functions internally to store transient information regarding an individual user's state or preferences. For example, `drupal_set_message()` needs to carry over a status message or an error message for the user from the page on which the error occurred to the next page. This is done by storing the messages in an array named `messages` inside the user's session:

```
/**
 * Set a message which reflects the status of the performed operation.
 *
 * If the function is called with no arguments, this function returns all set
 * messages without clearing them.
 *
 * @param $message
 *   The message should begin with a capital letter and always ends with a
 *   period '.'.
 * @param $type
 *   The type of the message. One of the following values are possible:
 *   'status', 'warning', 'error'
 * @param $repeat
 *   If this is FALSE and the message is already set, then the message won't
 *   be repeated.
 */
function drupal_set_message($message = NULL, $type = 'status', $repeat = TRUE) {
  if ($message) {
    if (!isset($_SESSION['messages'])) {
      $_SESSION['messages'] = array();
    }

    if (!isset($_SESSION['messages'][$type])) {
      $_SESSION['messages'][$type] = array();
    }

    if ($repeat || !in_array($message, $_SESSION['messages'][$type])) {
      $_SESSION['messages'][$type][] = $message;
    }
  }

  // Messages not set when DB connection fails.
  return isset($_SESSION['messages']) ? $_SESSION['messages'] : NULL;
}
```

Another example is from `comment.module`, where the session is used to store viewing preferences for anonymous users:

```
$_SESSION['comment_mode'] = $mode;
$_SESSION['comment_sort'] = $order;
$_SESSION['comment_comments_per_page'] = $comments_per_page;
```


Pro Drupal Development, Second Edition By John K. VanDyk ISBN: 9781430209898
Publisher: Apress
Print Publication Date: 2008/08/01

Drupal also uses sessions to keep a handle on file uploads when a node is being previewed, to remember viewing preferences when filtering the list of site content at Administer ➤ Content management ➤ Content or the list of recent log entries at Administer ➤ Reports ➤ Recent log entries, and for the installation and update systems (`install.php` and `update.php`).

Drupal creates sessions for both users that are logged into a site (authenticated users) and are not logged in (anonymous users). In the row of the `sessions` table representing an anonymous user, the `uid` column is set to `0`. Because sessions are browser specific (they're tied to the browser's cookie), having multiple browsers open on a single computer results in multiple sessions.

■ **Caution** Drupal doesn't store session information the first time an anonymous user visits a site. This is to keep evil web crawlers and robots from flooding the `sessions` table with data. As a developer, this means you cannot store session information for the first visit from an anonymous user.

The actual data stored in a session is stored as serialized data in the `session` column of the `sessions` table. Three rows of a typical sessions table are shown in Table 16-1. The table shows records for the superuser (`uid 1`), an authenticated user (`uid 3`), and an anonymous user (`uid 0`). The superuser has watchdog filtering settings (used by the dblog module) stored in the session.

Table 16-1. *Example Rows from the Sessions*

uid	sid	hostname	timestamp	cache	session
1	f5268d678333a1a7cce27e7e42b0c2e1	1.2.3.4	1208464106	0	dblog_overview_filter|a:0:{}
3	be312e7b35562322f3ee98ccb9ce8490	5.6.7.8	1208460845	0	--
0	5718d73975456111b268ed06233d36de	127.0.0.1	1208461007	0	--

The `sessions` table is cleaned when PHP's session garbage collection routine runs. The length of time a row remains in the table is determined by the `session.gc_maxlifetime` setting in `settings.php`. If a user logs out, the row for that session is removed from the database immediately. Note that if a user is logged in via multiple browsers (not browser windows) or multiple IP addresses at the same time, each browser has a session; therefore, logging out from one browser doesn't log the user out from the other browsers.

Session-Related Settings

There are three places where Drupal modifies session-handling settings: in the `.htaccess` file, in the `settings.php` file, and in the bootstrap code in the `includes/bootstrap.inc` file.

Pro Drupal Development, Second Edition By John K. VanDyk ISBN: 9781430209898 Prepared for Kyle Skrinak, Safari ID: kyle@skrinakcreative.com
Publisher: Apress
Print Publication Date: 2008/08/01 User number: 1718612

In .htaccess

Drupal ensures that it has full control over when sessions start by turning off PHP's `session.auto_start` functionality in the Drupal installation's default `.htaccess` file with the following line:

```
php_value session.auto_start             0
```

`session.auto_start` is a configuration option that PHP cannot change at runtime, which is why it lives in the `.htaccess` file instead of `settings.php`.

In settings.php

You'll set most session settings within the `settings.php` file, located at `sites/default/settings.php` or `sites/example.com/settings.php`.

```
ini_set('session.cache_expire',     200000);  // 138.9 days.
ini_set('session.cache_limiter',    'none');  // Cache control is done elsewhere.
ini_set('session.cookie_lifetime',  2000000); // 23.1 days.
ini_set('session.gc_maxlifetime',   200000);  // 55 hours.
ini_set('session.save_handler',     'user');  // Use user-defined session handling.
ini_set('session.use_only_cookies', 1);       // Require cookies.
ini_set('session.use_trans_sid',    0);       // Don't use URL-based sessions.
```

Having these settings in `settings.php` instead of `.htaccess` allows subsites to have different settings and allows Drupal to modify the session settings on hosts running PHP as a CGI (PHP directives in `.htaccess` don't work in such a configuration).

Drupal uses the `ini_set('session.save_handler', 'user');` function to override the default session handling provided by PHP and implement its own session management; *user-defined* in this context means "defined by Drupal" (see `http://www.php.net/manual/en/function.session-set-save-handler.php`).

In bootstrap.inc

PHP provides built-in session-handling functions but allows you to override those functions if you want to implement your own handlers. PHP continues to handle the cookie management, while Drupal's implementation does the back-end handling of session storage.

The following call during the `DRUPAL_BOOTSTRAP_SESSION` phase of bootstrapping sets the handlers to functions in `includes/sessions.inc` and starts session handling:

```
require_once variable_get('session_inc', './includes/session.inc');
session_set_save_handler('sess_open', 'sess_close', 'sess_read', 'sess_write',
  'sess_destroy_sid', 'sess_gc');
session_start();
```

This is one of the few cases where the names of the functions inside a file don't match the file's name. You would expect the preceding function names to be `session_open`, `session_close`, and so on. However, because PHP already has built-in functions in that namespace, the shorter prefix `sess_` is used.

Pro Drupal Development, Second Edition By John K. VanDyk ISBN: 9781430209898
Publisher: Apress
Print Publication Date: 2008/08/01

Notice that the file being included is defined by a Drupal variable. This means that you can cleanly implement your own session handling and plug in that instead of using Drupal's default session handling. For example, the memcache module (`drupal.org/project/memcache`) implements the `sess_open()`, `sess_close()`, `sess_read()`, `sess_write()`, `sess_destroy_sid()`, and `sess_gc()` functions. Setting the `session_inc` Drupal variable causes Drupal to use this code for sessions instead of using default session handling:

```
<?php
  variable_set('session_inc', './sites/all/modules/memcache/memcache-session.inc');
?>
```

You could also override the variable by setting it in your `settings.php` file:

```
$conf = array(
  'session_inc' => './sites/all/modules/memcache/memcache-session.inc,
  ...
);
```

Requiring Cookies

If the browser doesn't accept cookies, a session cannot be established because the PHP directive `sessions_use_only_cookies` has been set to `1` and the alternative (passing the `PHPSESSID` in the query string of the URL) has been disabled by setting `sessions.use_trans_sid` to `0`. This is a best practice, as recommended by Zend (see `http://php.net/session.configuration`):

> *URL based session management has additional security risks compared to cookie-based session management. Users may send a URL that contains an active session ID to their friends by e-mail or users may save a URL that contains a session ID to their bookmarks and access your site with the same session ID always, for example.*

When `PHPSESSID` appears in the query string of a site, it's typically a sign that the hosting provider has locked down PHP and doesn't allow the `ini_set()` function to set PHP directives at runtime. Alternatives are to move the settings into the `.htaccess` file (if the host is running PHP as an Apache module) or into a local `php.ini` file (if the host is running PHP as a CGI executable).

To discourage session hijacking, the session ID is regenerated when a user logs in (see the `user_authenticate_finalize()` function in `modules/user/user.module`). The session is also regenerated when a user changes his or her password.

Storage

Session information is stored in the `sessions` table, which associates session IDs with Drupal user IDs during the `DRUPAL_BOOTSTRAP_SESSION` phase of bootstrapping (see Chapter 15 to learn more about Drupal's bootstrapping process). In fact, the `$user` object, which is used extensively throughout Drupal, is first built during this phase by `sess_read()` in `includes/sessions.inc` (see Chapter 6 to see how the `$user` object is built).

Pro Drupal Development, Second Edition By John K. VanDyk ISBN: 9781430209898 Prepared for Kyle Skrinak, Safari ID: kyle@skrinakcreative.com
Publisher: Apress
Print Publication Date: 2008/08/01 User number: 1718612

Table 16-2 shows the table structure in which sessions are stored.

Table 16-2. *The Structure of the Sessions Table*

Field	Type	Length	Description
uid	int		User ID of authenticated user (0 for anonymous user)
sid	int	64	Session ID generated by PHP
hostname	varchar	128	IP address that last used this session ID
timestamp	int		Unix timestamp of last page request
cache	int		Time of user's last post, which is used to enforce minimum cache lifetime
session	text	big	Serialized contents of data stored in $_SESSION

When Drupal serves a page, the last task completed is to write the session to the sessions table (see sess_write() in includes/session.inc). This is only done if the browser has presented a valid cookie to avoid bloating the sessions table with sessions for web crawlers.

Session Life Cycle

The session life cycle is shown in Figure 16-1. It begins when a browser makes a request to the server. During the DRUPAL_BOOTSTRAP_SESSION phase of Drupal's bootstrap routines (see includes/bootstrap.inc) the session code begins. If the browser doesn't present a cookie that it had previously received from the site, PHP's session management system will give the browser a new cookie with a new PHP session ID. This ID is usually a 32-character representation of a unique MD5 hash, though PHP 5 allows you to set the configuration directive session.hash_function to 1, optionally giving you SHA-1 hashes that are represented by 40-character strings.

■ **Note** MD5 is an algorithm for computing the hash value of a string of text and is the algorithm of choice for computing hashes within Drupal. For information on MD5 and other hash algorithms, see http://en.wikipedia.org/wiki/Cryptographic_hash_functions.

Drupal then checks the sessions table for the existence of a row with the session ID as the key. If found, the sess_read() function in includes/sessions.inc retrieves the session data and performs an SQL JOIN on the row from the sessions table and on the corresponding row from the users table. The result of this join is an object containing all fields and values from both rows. This is the global $user object that's used throughout the rest of Drupal (see Chapter 6). Thus, session data is also available by looking in the $user object, specifically in $user->session, $user->sid, $user->hostname, $user->timestamp, and $user->cache. Roles for the current user are looked up and assigned to $user->roles in sess_read() as well.

Pro Drupal Development, Second Edition By John K. VanDyk ISBN: 9781430209898 Prepared for Kyle Skrinak, Safari ID: kyle@skrinakcreative.com
Publisher: Apress
Print Publication Date: 2008/08/01 User number: 1718612

But what happens if there's no user in the `users` table with a user ID that matches the user ID in the session? This is a trick question. Because Drupal's installer creates a row in the `users` table with the user ID of `0`, and because unauthenticated (anonymous) users are assigned the `uid` of `0` in the `sessions` table, the join always works.

■ **Caution** Never delete all rows from the users table of your Drupal installation. The row containing user ID `0` is needed for Drupal to function properly.

If you want to find out the last time the user accessed a page, you could either look at `$user->timestamp` (remember, that comes from the `sessions` table) or at `$user->access`, which is kept in the `users` table. Of the two, `$user->timestamp` will give you more accurate results if it is present, because updating of `$user->access` in the `users` table is subject to throttling so that writes do not happen more often than every 180 seconds by default. This value can be changed by setting the Drupal variable `session_write_interval`. From `sess_write()` in `includes/session.inc`:

```
// Last access time is updated no more frequently than once every 180 seconds.
// This reduces contention in the users table.
$session_write_interval = variable_get('session_write_interval', 180);
if ($user->uid && time() - $user->access > $session_write_interval) {
  db_query("UPDATE {users} SET access = %d WHERE uid = %d", time(), $user->uid);
}
```

Of course, neither `$user->timestamp` nor `$user->access` will be present for users visiting for the first time, as no timestamp has been saved yet.

When the web page has been delivered to the browser, the last step is to close the session. PHP invokes the `sess_write()` function in `includes/session.inc`, which writes anything that was stashed in `$_SESSION` (during the request) to the `sessions` table. It is a good idea to only store data in `$_SESSION` if you absolutely need to, and even then only when you are sure that the user has authenticated. The reason for this is to prevent the table from bloating up with rows generated by web crawlers, as the size of the table can impact performance.

Pro Drupal Development, Second Edition By John K. VanDyk ISBN: 9781430209898 Prepared for Kyle Skrinak, Safari ID: kyle@skrinakcreative.com
Publisher: Apress
Print Publication Date: 2008/08/01 User number: 1718612

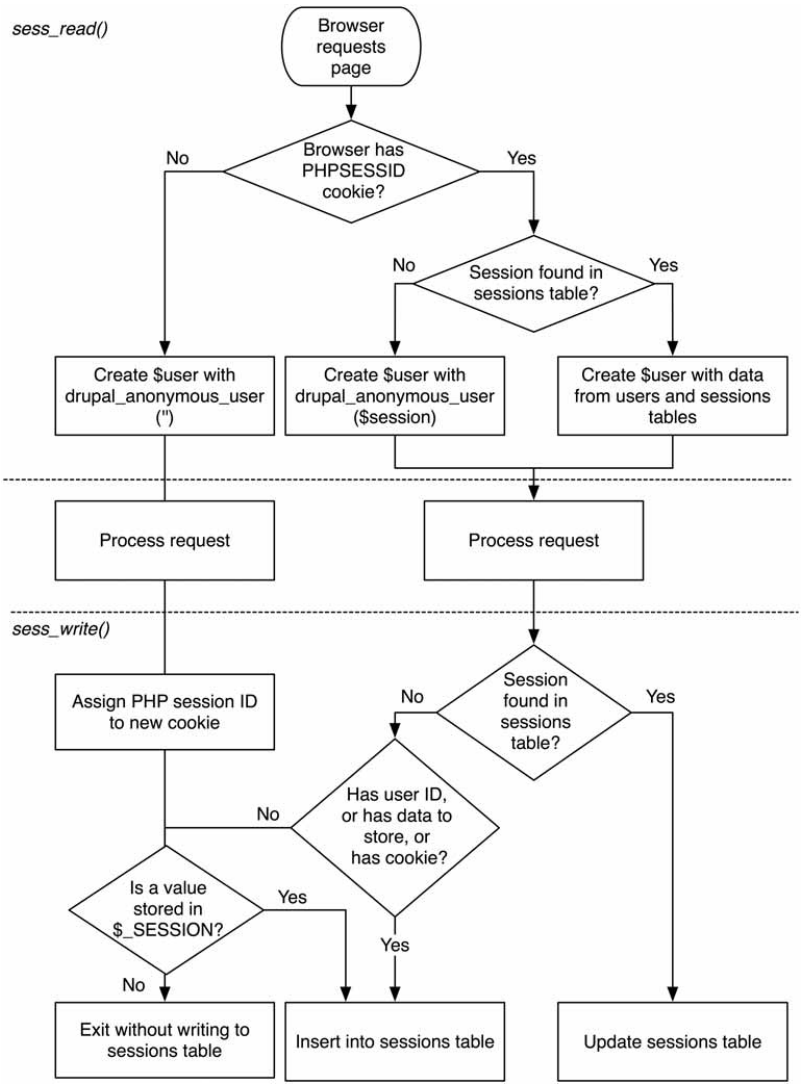

Figure 16-1. *How Drupal uses sessions to instantiate the $user object*

Session Conversations

Here are some examples of what happens when you visit Drupal in your browser, from a sessions perspective.

Pro Drupal Development, Second Edition By John K. VanDyk ISBN: 9781430209898 Prepared for Kyle Skrinak, Safari ID: kyle@skrinakcreative.com
Publisher: Apress
Print Publication Date: 2008/08/01 User number: 1718612

First Visit

Browser: Hi, I'd like a page, please.

Drupal: May I see your cookie?

Browser: Sorry, I don't have a cookie; this is my first time here.

Drupal: OK, here's one.

Second Visit

Browser: May I have another page, please?

Drupal: May I see your cookie?

Browser: Right here. It says session number 6tc47s8jd6rls9cugkdrrjm8h5.

Drupal: Hmm, I can't find you in my records. But here's your page anyway. I'll make a note of you in case you visit again.

User with an Account

[The user has created an account and clicked the Log In button.]

Browser: Hi, I'd like a page, please.

Drupal: May I see your cookie?

Browser: Right here. It says session number 31bfa29408ebb23239042ca8f0f77652.

Drupal: Hi, Joe! [Mumbling] You're user ID 384, and you like your comments nested and your coffee black. Here's a new cookie so your session doesn't get hijacked. I'll make a note that you visited. Have a nice day.

Common Tasks

Here are some common ways in which you might want to use sessions or tweak session settings.

Changing the Length of Time Before a Cookie Expires

The length of time before the cookie containing the session ID expires is controlled by `session.cookie_lifetime` in `settings.php` and set by default to 2,000,000 seconds (about 23 days). Modifying this value to `0` causes the cookie to be destroyed when the user closes the browser.

Changing the Name of the Session

A common problem with sessions arises when deploying web sites on multiple subdomains. Because each site uses the same default value for `session.cookie_domain` and the same

Pro Drupal Development, Second Edition By John K. VanDyk ISBN: 9781430209898 Prepared for Kyle Skrinak, Safari ID: kyle@skrinakcreative.com
Publisher: Apress
Print Publication Date: 2008/08/01 User number: 1718612

session.name of PHPSESSID by default, users find themselves able to log into only one site at any given time. Drupal solves this problem by creating a unique session name for each site. The session name is based on a MD5 hash, with some modifications, of the base URL for the site. See conf_init() in includes/bootstrap.inc for details.

The automatic generation of the session name can be bypassed by uncommenting a line in settings.php and specifying the value of the $cookie_domain variable. The value should contain alphanumeric characters only. Here is the relevant section of settings.php:

```
/**
 * Drupal automatically generates a unique session cookie name for each site
 * based on on its full domain name. If you have multiple domains pointing at
 * the same Drupal site, you can either redirect them all to a single domain
 * (see comment in .htaccess), or uncomment the line below and specify their
 * shared base domain. Doing so assures that users remain logged in as they
 * cross between your various domains.
 */
# $cookie_domain = 'example.com';
```

Note The only time Perl-style comment characters (#) are used in Drupal are in settings.php, .htaccess, robots.txt, and the actual Perl and shell scripts in the scripts directory.

Storing Data in the Session

Storing data in a user's session is convenient, because the data is automatically stored by the sessions system. Whenever you want to store data that you want to associate with a user during a visit (or multiple visits up to session.cookie_lifetime), use the $_SESSION superglobal:

```
$_SESSION['favorite_color'] = $favorite_color;
```

Later, on a subsequent request, do the following to retrieve the value:

```
$favorite_color = $_SESSION['favorite_color'];
```

If you know the user's uid and you want to persist some data about the user, it's usually more practical to store it in the $user object as a unique attribute such as $user->foo = $bar by calling user_save($user, array('foo' => $bar)), which serializes the data to the users table's data column. Here's a good rule of thumb to use: If the information is transient and you don't mind if it's lost, or if you need to store short-term data for anonymous users, you can store it in the session. If you want to tie a preference permanently to a user's identity, store it in the $user object.

Caution $user should not be used to store information for anonymous users.

Pro Drupal Development, Second Edition By John K. VanDyk ISBN: 9781430209898 Prepared for Kyle Skrinak, Safari ID: kyle@skrinakcreative.com
Publisher: Apress
Print Publication Date: 2008/08/01 User number: 1718612

Summary

After reading this chapter, you should be able to

- Understand how Drupal modifies PHP's session handling.
- Understand which files contain session configuration settings.
- Understand the session life cycle and how Drupal's `$user` object is created during a request.
- Store data in and retrieve data from a user's session.

Pro Drupal Development, Second Edition By John K. VanDyk ISBN: 9781430209898
Publisher: Apress
Print Publication Date: 2008/08/01