ГЛАВА 16

Сеансы

HTTP является протоколом, не запоминающим состояния. Это означает, что каждое взаимодействие между веб-браузером и сервером остается автономным. Каким тогда образом отследить пользователя, который просматривает последовательность веб-страниц на веб-сайте? Для этого служат сеансы. В версии PHP 4 появилась встроенная поддержка для сеансов через семейство функций сеанса. В этой главе речь пойдет о том, как Drupal использует PHP-сеансы.

Что такое сеансы

Когда браузер впервые запрашивает страницу с сайта Drupal, PHP отправляет браузеру cookie-набор, содержащий случайно сгенерированный 32-символьный идентификатор, по умолчанию называемый `PHPSESSID`. Это делается за счет включения одной строки в HTTP-заголовки ответа, отправленные браузеру при первом посещении сайта:

```
HTTP/1.1 200 OK
Date: Thu, 17 Jan 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
```

При последующих посещениях этого сайта браузер передает cookie-набор серверу, включая его в каждый HTTP-запрос:

```
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
```

Это позволяет PHP отслеживать конкретный браузер, с которого посещается веб-сайт. 32-символьный идентификатор, известный как *идентификатор сеанса* (session ID), используется в качестве ключа к информации о сеансе, сохраняемой Drupal, и позволяет ассоциировать сеансы с индивидуальными пользователями.

Использование сеансов

Drupal внутренне использует сеансы для нескольких важных функций, чтобы хранить промежуточную информацию о состоянии или персональных настройках отдельного пользователя. Например, `drupal_set_message()` заботится о переносе сообщений о состоянии или об ошибках для пользователя со страницы, на которой произошла ошибка, на следующую страницу. Это выполняется за счет сохранения сообщений в массиве с именем `messages` внутри пользовательского сеанса.

```
/**
 * Установить сообщение, отражающее состояние выполненной операции.
 *
 * Если функция вызывается без аргументов, она возвращает
 * весь набор сообщений, не очищая их.
 *
 * @param $message
 *  Сообщение должно начинаться с заглавной буквы и
 *  всегда завершаться точкой.
 * @param $type
 *  Тип сообщения. Допускается одно из следующих значений:
 *   'status', 'warning', 'error'
 * @param $repeat
 *  Если равно FALSE и сообщение уже установлено, то сообщение
 *  не будет повторяться.
 */
function drupal_set_message($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][] = $message;
    }
  }
  // Сообщения не устанавливаются в случае сбоя соединения с базой данных.
  return isset($_SESSION['messages']) ? $_SESSION['messages'] : NULL;
}
```

Другой пример взят из модуля `comment.module`, где сеанс используется для сохранения персональных настроек отображения для анонимных пользователей:

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

Drupal также использует сеансы, чтобы отслеживать загрузки файлов при предварительном просмотре узла, для запоминания персональных настроек отображения, когда фильтруется список контента сайта в Administer⇨Content management⇨Content (Администрирование⇨Управление контентом⇨Контент) или список последних журнальных записей в Administer⇨Reports⇨Recent log entries (Администрирование⇨Отчеты⇨Последние журнальные записи), а также для систем инсталляции и обновления (`install.php` и `update.php`).

Drupal создает сеансы как для пользователей, которые зарегистрированы на сайте (аутентифицированных пользователей), так и для незарегистрированных (анонимных) пользователей. В строке таблицы `sessions`, представляющей анонимного пользователя, столбец `uid` имеет значение `0`. Поскольку сеансы специфичны для браузера (они привязаны к cookie-набору браузера), при наличии нескольких браузеров, открытых на одном компьютере, будет создано несколько сеансов.

Внимание! Drupal не сохраняет информацию о сеансе, когда анонимный пользователь посещает сайт в первый раз. Это предохраняет таблицу `sessions` от переполнения данными, возникающими при посещении сайта "веб-червяками" и роботами. Для разработчика это означает, что сохранять информацию о сеансе для первого посещения сайта анонимным пользователем нельзя.

Фактические данные, сохраняемые в сеансе, сохраняются в сериализованном виде в столбце `session` таблицы `sessions`. Три строки типичной таблицы `sessions` показаны в табл. 16.1. Здесь представлены записи для суперпользователя (`uid 1`), аутентифицированного пользователя (`uid 3`) и анонимного пользователя (`uid 0`). Суперпользователь имеет настройки фильтрации регистратора (watchdog; используется модулем `dblog`), хранящиеся в сеансе.

Таблица 16.1. Примеры строк из таблицы `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	--

Таблица `sessions` очищается, когда выполняется сборщик мусора сеансов PHP. Промежуток времени, в течение которого строка остается в таблице, определяется настройкой `session.gc.maxlifetime` в `settings.php`. Если пользователь выходит из системы, строка его сеанса удаляется из базы данных немедленно. Обратите внимание, что если пользователь зашел на сайт одновременно из множества браузеров (не окон одного браузера) или с несколькими IP-адресами, каждый браузер имеет свой сеанс, поэтому выход пользователя из сайта в одном браузере не означает его выхода из сайта в других браузерах.

Настройки, связанные с сеансом

Есть три места, где Drupal изменяет настройки управления сеансами: в файле `.htaccess`, в файле `settings.php` и в коде начальной загрузки внутри файла `includes/bootstrap.inc`.

.htaccess

Drupal гарантирует, что имеет полный контроль того, когда сеансы начинаются, отключая PHP-настройку `session.auto_start` в файле Drupal `.htaccess` по умолчанию:

```
php_value session.auto_start          0
```

`session.auto_start` — это опция конфигурации, которую PHP не может изменить во время выполнения, поэтому она и находится в `.htaccess`, а не в `settings.php`.

settings.php

Большинство настроек сеансов осуществляется в файле `settings.php`, который находится в `sites/default/settings.php` или в `sites/example.com/settings.php`.

```
ini_set('session.cache_expire',   200000); // 138.9 дня
ini_set('session.cache_limiter',  'none'); // Управление кэшем в другом месте
ini_set('session.cookie_lifetime', 2000000);// 23.1 дня
ini_set('session.gc_maxlifetime', 200000); // 55 часов
ini_set('session.save_handler',   'user'); // Использовать управление сеансом,
                                           // определяемое пользователем.
ini_set('session.use_only_cookies', 1);    // Включить поддержку
                                           // cookie-наборов.
ini_set('session.use_trans_sid',  0);      // Не использовать сеансы
                                           // на основе URL.
```

Наличие этих настроек в `settings.php` вместо `.htaccess` позволяет подсайтам иметь другие настройки, а также дает возможность Drupal изменять настройки сеанса на хостах, выполняющих PHP как CGI (PHP-директивы в `.htaccess` в такой конфигурации не работают).

Drupal использует функцию `ini_set('session.save_handler', 'user');` для переопределения поддержки сеансов по умолчанию, обеспечиваемой PHP, и реализует собственное управление сеансом; *user-defined* (определяемое пользователем) в данном контексте означает "определяемое Drupal" (см. `http://www.php.net/manual/en/function.session-set-save-handler.php`).

bootstrap.inc

PHP предоставляет встроенные функции поддержки сеансов, но позволяет переопределять эти функции, если необходимо реализовать ваши собственные обработчики. PHP продолжает обрабатывать cookie-наборы, в то время как реализация Drupal выполняет конечную обработку и сохранение сеансов.

Следующий запрос в течение фазы `DRUPAL_BOOTSTRAP_SESSION` начальной загрузки настраивает обработчики на функции, содержащиеся в `includes/sessions.inc`, и начинает поддержку сеансов:

```
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();
```

Это — один из немногих случаев, когда имена функций в файле не соответствуют имени файла. Вы могли бы ожидать, что приведенные выше функции будут называться `session_open`, `session_close` и т.д. Однако поскольку PHP имеет собственные встроенные функции в этом пространстве имен, используется более короткий префикс `sess`.

Обратите внимание на то, что включаемый файл определен переменной Drupal. Это означает, что вы можете реализовать собственный механизм поддержки сеансов и подключить его вместо поддержки сеансов Drupal, предлагаемой по умолчанию. Например, модуль `memcache` (`drupal.org/project/memcache`) реализует функции `sess_open()`, `sess_close()`, `sess_read()`, `sess_write()`, `sess_destroy_sid()` и `sess_gc()`. Установка Drupal-переменной `session_inc` заставляет Drupal использовать для сеансов этот код вместо кода по умолчанию:

```
<?php
  variable_set('session_inc', './sites/all/inmemorysessions.inc');
?>
```

Переопределить переменную можно также и устанавливая ее в файле settings.php:

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

Включение поддержки cookie-наборов

Если браузер не принимает cookie-наборы, сеанс не может быть установлен, потому что PHP-директива sessions_use_only_cookies будет иметь значение 1, а альтернативное поведение (передача PHPSESSID в строке запроса URL) заблокировано установкой sessions.use_trans_sid в 0. Наилучшая практика, соответствующая рекомендациям Zend (см. http://php.net/session.configuration) гласит следующее.

> *Управление сеансами на базе URL привносит дополнительные риски безопасности по сравнению с управлением сеансами на основе cookie-наборов. Пользователи могут послать URL, который содержит идентификатор активного сеанса, своим друзьям по электронной почте или же сохранить URL, который содержит идентификатор сеанса, в закладках браузера и, например, всегда получать доступ к вашему сайту с одним и тем идентификатором сеанса.*

Когда в строке запроса сайта появляется PHPSESSID, это обычно служит признаком того, что провайдер хостинга заблокировал PHP и не позволяет функции ini_set() устанавливать директивы PHP во время выполнения. Альтернатива состоит в перемещении параметров настройки в файл .htaccess (если хост выполняет PHP как модуль Apache) или в локальный файл php.ini (если хост выполняет PHP как CGI).

Чтобы предотвратить взлом сеанса, идентификатор сеанса обновляется при входе пользователя на сайт (см. функцию user_authenticate_finalize() в modules/user/user.module). Сеанс также регенерируется, когда пользователь изменяет свой пароль.

Хранилище

Информация о сеансе сохраняется в таблице sessions, которая связывает идентификаторы сеанса с идентификаторами пользователя Drupal в ходе выполнения фазы DRUPAL_BOOTSTRAP_SESSION начальной загрузки (в главе 15 процесс начальной загрузки Drupal описан более подробно). Фактически объект $user, который широко используется повсюду в Drupal, впервые строится во время этой фазы с помощью sess_read() из includes/sessions.inc (построение объекта $user рассматривается в главе 6). В табл. 16.2 показана структура таблицы, в которой сохраняется информация о сеансах.

Таблица 16.2. Структура таблицы sessions

Поле	Тип	Длина	Описание
uid	int		Идентификатор аутентифицированного пользователя (0 – для анонимного пользователя)
sid	int	64	Идентификатор сеанса, сгенерированный PHP
hostname	varchar	128	IP-адрес, последний раз использовавший этот идентификатор сеанса
timestamp	int		Временная метка Unix последнего запроса страницы
cache	int		Время последней отправки пользователя, которое используется для установки минимального времени жизни кэша
session	text	big	Сериализованная версия данных, хранимых в $_SESSION

Когда система Drupal обслуживает страницу, последняя задача состоит в сохранении данных сеанса в таблице `sessions` (см. `sess_write()` в `includes/session.inc`). Это делается только в случае, если браузер предоставляет допустимый cookie-набор, что позволяет избежать переполнения таблицы `sessions` за счет сеансов "веб-червяков".

Жизненный цикл сеанса

Жизненный цикл сеанса показан на рис. 16.1.

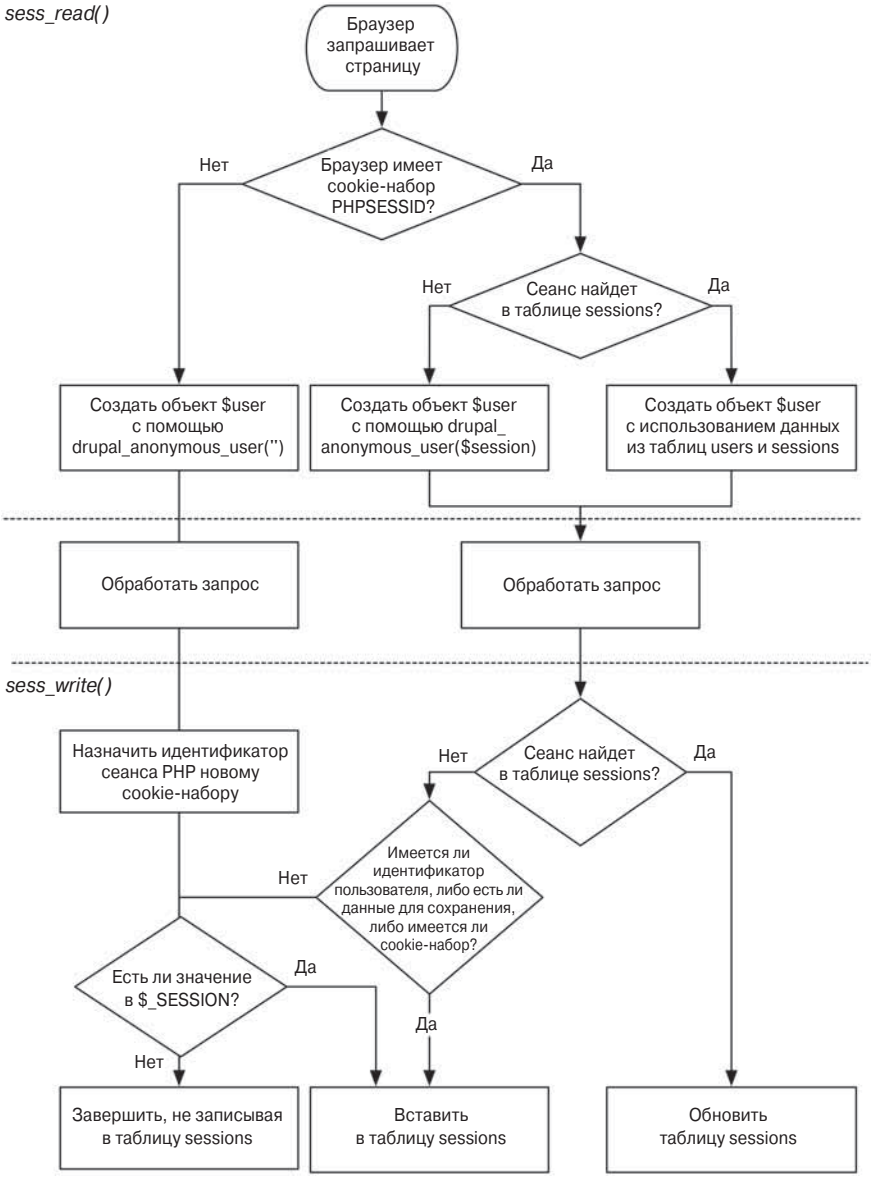

Рис. 16.1. Как Drupal использует сеансы для создания объекта `$user`

Сеанс начинается, когда браузер делает запрос к серверу. В течение фазы DRUPAL_BOOTSTRAP_SESSION начальной загрузки Drupal (см. includes/bootstrap.inc) начинается выполнение кода сеанса. Если браузер не предоставляет cookie-набор, который он ранее получил от сайта, система управления сеансами PHP выдаст браузеру новый cookie-набор с новым идентификатором сеанса PHP. Этот идентификатор — обычно 32-символьное представление уникального хеша MD5, хотя PHP 5 позволяет установить директиву конфигурации hash_function в 1 и включить алгоритм хеширования SHA-1, который имеет дело с 40-символьными строками.

На заметку! MD5 — это алгоритм вычисления хеш-значения для строки текста, по умолчанию выбираемый в Drupal. Дополнительную информацию по MD5 и другим алгоритмам хеширования можно найти по адресу http://ru.wikipedia.org/wiki/Хеширование.

Затем Drupal проверяет таблицу sessions на предмет наличия строки с идентификатором сеанса в качестве ключа. Если такая строка найдена, функция sess_read() из includes/sessions.inc извлекает данные сеанса и выполняет SQL-соединение JOIN на строке из таблицы sessions и соответствующей ей строке из таблицы users. Результат этого соединения — объект, содержащий все поля и значения из обеих строк. Это — глобальный объект $user, который используется повсюду в Drupal (см. главу 6). Таким образом, данные сеанса также становятся доступными через $user, в частности $user->session, $user->sid, $user->hostname, $user->timestamp и $user->cache. Роли для текущего пользователя также извлекаются и присваиваются $user->roles.

Но что случится, если в таблице users нет пользователя с идентификатором, соответствующим идентификатору пользователя в сеансе? Подобное невозможно. Поскольку инсталлятор Drupal создает строку в таблице users с идентификатором пользователя 0, а неаутентифицированным (анонимным) пользователям в таблице sessions назначается uid со значением 0, соединение будет работать всегда.

Внимание! Никогда не удаляйте все строки из таблицы users вашей инсталляции Drupal. Строка, содержащая идентификатор пользователя, равный 0, необходима для правильного функционирования самой системы Drupal.

Если необходимо узнать, когда пользователь в последний раз обращался к странице, можно либо заглянуть в $user->timestamp (это берется из таблицы sessions), либо в $user->access, значение которого хранится в таблице users. Из этих двух вариантов $user->timestamp даст более точный результат, если он есть, потому что обновление $user->access в таблице users зависит от регулировки и не может происходить чаще, чем каждые 180 секунд по умолчанию. Это значение может быть изменено установкой Drupal-переменной session_write_interval. Ниже приведен фрагмент кода из sess_write() в includes/session.inc:

```
// Время последнего доступа обновляется не чаще, чем раз в 180 секунд.
// Это сокращает число параллельных доступов к таблице users.
$session_write_interval = variable_get('session_write_interval', 180);
if ($user->uid && time() - $user->access > $session_write_interval) {
  db_query("UPDATE {users} SET access = %d WHERE uid = %d",
           time(), $user->uid);
}
```

Разумеется, ни $user->timestamp, ни $user->access не существуют для пользователей, заходящих на сайт впервые, поскольку для них еще не сохранено никаких временных меток.

Когда веб-страница доставлена браузеру, последний шаг состоит в закрытии сеанса. PHP вызывает функцию `sess_write()` из `includes/session.inc`, которая записывает всю накопившуюся в `$_SESSION` информацию (в течение запроса) в таблицу `sessions`. Лучше всего хранить данные в `$_SESSION` только в том случае, если вы действительно нуждаетесь в них, и только тогда, когда вы уверены, что пользователь аутентифицирован. Причина — предотвратить переполнение таблицы `sessions` строками, сгенерированными "веб-червяками", поскольку размер таблицы влияет на производительность.

Сеансовые диалоги

Ниже представлены некоторые примеры того, что происходит с точки зрения сеансов, когда вы посещаете сайт Drupal из браузера.

Первое посещение

Браузер: Я хотел бы получить страницу.

Drupal: Я могу видеть ваш cookie-набор?

Браузер: К сожалению, у меня нет cookie-набора, я посещаю сайт впервые.

Drupal: Хорошо, вот вам cookie-набор.

Второе посещение

Браузер: Могу я получить еще одну страницу?

Drupal: Я могу видеть ваш cookie-набор?

Браузер: Вот он. В нем сообщается идентификатор сеанса `6tc47s8jd6rls9cugkdrrjm8h5`.

Drupal: Я не могу найти вас в записях. Но все равно, вот запрошенная вами страница. Я сделаю запись о вас, если вы посетите сайт снова.

Пользователь с учетной записью

(Пользователь создал учетную запись и щелкнул на кнопке Log In (Вход).)

Браузер: Я хотел бы получить страницу.

Drupal: Я могу видеть ваш cookie-набор?

Браузер: Вот он. В нем сообщается идентификатор сеанса `6tc47s8jd6rls9cugkdrrjm8h5`.

Drupal: Здравствуйте, Джо! Вы являетесь пользователем с идентификатором 384, предпочитаете вложенные комментарии и черный кофе. Вот новый cookie-набор, чтобы ваш сеанс не взломали. Я сделаю запись о вашем посещении. Всего хорошего.

Общие задачи

Ниже перечислены некоторые общие ситуации, при которых может потребоваться использование сеансов либо изменение настроек сеанса.

Изменение времени истечения действия cookie-набора

Период времени действия cookie-набора, содержащего идентификатор сеанса, задается в `session.cookie_lifetime` в файле `settings.php` и по умолчанию имеет значе-

ние 2 000 000 секунд (около 23 дней). Изменение этого значения на 0 приводит к тому, что после закрытия браузера cookie-набор будет уничтожаться.

Изменение имени сеанса

Общая проблема с сеансами возникает при развертывании Drupal на нескольких поддоменах. Поскольку каждый сайт по умолчанию использует одно и то же значение для session.cookie_domain и session.name для PHPSESSID, пользователи могут войти одновременно только на один сайт. Создание уникального имени сеанса для каждого сайта решает эту проблему. Имя сеанса основывается на хэше MD5, с некоторыми модификациями, на базе URL сайта. За подробностями обращайтесь в conf_init() из includes/bootstrap.inc.

Автоматическую генерацию имен сеансов можно обойти, убрав знак комментария с одной строки в settings.php и задав значение переменной $cookie_domain. Значение должно содержать только алфавитно-цифровые символы. Соответствующий раздел settings.php показан ниже.

```
/**
 * Drupal автоматически генерирует уникальное имя сеанса для каждого сайта
 * на основе его полного доменного имени. Если множество доменов указывают
 * на один и тот же сайт Drupal, можно либо перенаправлять их на
 * единственный домен (см. комментарий в .htaccess), либо убрать знак
 * комментария с приведенной ниже строки и указать их разделяемый базовый
 * домен. В результате пользователи останутся зарегистрированными при
 * переходах между различными доменами.
 */
# $cookie_domain = 'example.com';
```

На заметку! Знаки комментариев в стиле Perl (#) используются в Drupal только в файлах settings.php, .htaccess, robots.txt и в действительных сценариях Perl и командной оболочки, которые хранятся в каталоге scripts.

Хранение данных в сеансе

Хранить данные в пользовательском сеансе удобно, потому что данные автоматически сохраняются системой сеансов. Всякий раз, когда понадобится сохранить данные, которые необходимо связать с пользователем во время его визита (или множества визитов до session.cookie_lifetime), используйте суперглобальный массив $_SESSION:

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

При последующем запросе для извлечения этого значения применяйте такой код:

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

Если известен uid пользователя и нужно записать некоторые данные о пользователе, обычно более практичным способом является их сохранение в объекте $user в виде уникального атрибута, такого как $user->foo = $bar, с помощью вызова функции user_save($user, array('foo' => $bar)), которая сериализует данные для столбца таблицы users. Хорошее эвристическое правило: если информация является переходной и ее потеря не важна, или если нужно сохранить краткосрочные данные для анонимных пользователей, это можно сделать внутри сеанса. Если же требуется привязка к идентичности пользователя, сохраняйте данные в объекте $user.

Внимание! Объект `$user` не должен использоваться для хранения информации для анонимных пользователей.

Резюме

После прочтения этой главы вы должны уметь следующее.

- Понимать, как Drupal изменяет обработку PHP-сеанса
- Знать, в каких файлах содержатся настройки сеанса
- Понимать жизненный цикл сеанса и знать, как в течение запроса создается объект `$user`
- Понимать, как сохраняются и извлекаются данные из пользовательского сеанса