ЛАБОРАТОРНАЯ РАБОТА №6
МЕХАНИЗМ СЕССИЙ

6.1 Основные понятия

Сессии являются простым способом хранения информации для отдельных пользователей с уникальным идентификатором сессии. Это может использоваться для сохранения состояния между запросами страниц. Идентификаторы сессий обычно отправляются браузеру через сессионный cookie и используются для получения имеющихся данных сессии. Отсутствие идентификатора сессии или сессионного cookie сообщает PHP о том, что необходимо создать новую сессию и сгенерировать новый идентификатор сессии.

Сессии используют простую технологию. Когда сессия создана, PHP будет либо получать существующую сессию, используя переданный идентификатор (обычно из сессионного cookie) или, если ничего не передавалось, будет создана новая сессия. PHP заполнит суперглобальную переменную $_SESSION сессионной информацией после того, как будет запущена сессия. Когда PHP завершает работу, он автоматически сериализует содержимое суперглобальной переменной $_SESSION и отправит для сохранения, используя сессионный обработчик для записи сессии.

По умолчанию PHP использует внутренний обработчик files для сохранения сессий, который установлен в INI-переменной session.save_handler. Этот обработчик сохраняет данные на сервере в директории, указанной в конфигурационной директиве session.save_path.

Сессии могут запускаться вручную с помощью функции session_start(). Если директива session.auto_start установлена в 1, сессия автоматически запустится, в начале запроса.

Сессия обычно завершает свою работу, когда PHP заканчивает исполнять скрипт, но может быть завершена и вручную с помощью функции session_write_close().

Регистрация переменной с помощью $_SESSION.

```
<?php
session_start();
if (!isset($_SESSION['count'])) {
  $_SESSION['count'] = 0;
} else {
  $_SESSION['count']++;
}
?>
```

Отмена объявления переменной с помощью $_SESSION.

```
<?php
session_start();
unset($_SESSION['count']);
?>
```

Сессии, использующие файлы (по умолчанию в PHP), блокируют файл сессии сразу при открытии сессии функцией session_start() или косвенно при указании session.auto_start. После блокировки, ни один другой скрипт не может получить доступ к этому же файлу сессии, пока он не будет закрыт или при завершении скрипта или при вызове функции session_write_close().

Скорее всего это станет проблемой для сайтов, которые активно используют AJAX и делают несколько одновременных запросов. Простейшим путем решить эту проблему будет вызов функции session_write_close() сразу же как только все требуемые изменения в сессии будут сделаны, предпочтительно ближе к началу работы скрипта. Также можно использовать другой механизм сессии, который поддерживает конкурентный доступ.

Веб-сервер не поддерживает постоянного соединения с клиентом, и каждый запрос обрабатывается, как новый, безо всякой связи с предыдущими. То есть, нельзя ни отследить запросы от одного и того же посетителя, ни сохранить для него переменные между просмотрами отдельных страниц. Вот для решения этих двух задач и были изобретены сессии.

Собственно, сессии, если в двух словах - это механизм, позволяющий однозначно идентифицировать браузер и создающий для этого браузера файл на сервере, в котором хранятся переменные сеанса.

Области применения: корзина покупок в е-магазине, авторизация, защита интерактивных частей сайта от спама.

6.2 Работа с сессиями

Для начала надо как-то идентифицировать браузер. Для этого надо выдать ему уникальный идентификатор и попросить передавать его с каждым запросом.

Сессии используют стандартные, хорошо известные способы передачи данных. Идентификатор - это обычная переменная. По умолчанию ее имя - PHPSESSID.

Задача PHP отправить ее браузеру, чтобы тот вернул ее со следующим запросом. Переменную можно передать только двумя способами: в куках или POST/GET запросом.

PHP использует оба варианта. За это отвечают две настройки в php.ini:

session.use_cookies- если равно 1, то PHP передает идентификатор в куках, если 0 - то нет.session.use_trans_sidесли равно 1, то PHP передает его, добавляя к URL и формам, если 0 - то нет.

Менять эти и другие параметры сессий можно так же, как и другие настройки PHP - в файле php.ini, а так же с помощью команды ini_set()или в файлах настройки веб-сервера.

Если включена только первая, то при старте сессии (при каждом вызове session_start()) клиенту устанавливается кука. Браузер исправно при каждом следующем запросе эту куку возвращает и PHP имеет идентификатор сессии. Проблемы начинаются, если браузер куки не возвращает. В этом случае, не получая куки с идентификатором, PHP будет все время стартовать новую сессию, и механизм работать не будет.

Если включена только вторая, то кука не выставляется. А происходит то, ради чего, в основном, собственно, и стоит использовать встроенный механизм сессий. После того, как скрипт выполняет свою работу, и страница полностью сформирована, PHP просматривает ее всю и дописывает к каждой ссылке и к каждой форме передачу идентификатора сессии. Это выглядит примерно так: Indexпревращается вIndexа к формам добавляется скрытое поле<input type="hidden" name="PHPSESSID" value="00196c1c1a02e4c37ac04f921f4a5eec" />

И браузер при клике на любую ссылку, или при нажатии на кнопку в форме, пошлет в запросе нужную нам переменную - идентификатор сессии! По очевидным причинам идентификатор добавляется только к относительным ссылкам.

К идентификатору привязывается файл с данными на стороне сервера. PHP это сделает за нас. Достаточно просто написать session_start(); $_SESSION['test']='Hello world!';И PHP запишет в файл, связанный с этой сессией, переменную test.

Чтобы поместить переменную в сессию, достаточно присвоить ее элементу массива $_SESSION. Чтобы получить ее значение - достаточно обратиться к тому же элементу. Пример будет чуть ниже.

Сборкой мусора - удалением устаревших файлов PHP тоже занимается сам. Как и кодированием данных и кучей всяких других нужных вещей. В результате этой заботы работа с сессиями оказывается очень простой. Пример: <? session_start(); if (!isset($_SESSION['counter'])) $_SESSION['counter']=0; echo "Вы обновили эту страницу ".$_SESSION['counter']++." раз. "; echo "
<a href=".$_SERVER['PHP_SELF']."">обновить"; ?>Мы проверяем, есть ли у нас в сессии переменная counter, если нет, то создаем ее со значением 0, а дальше выводим ее значение и увеличиваем на единицу. Увеличенное значение запишется в сессию, и при следующем вызове скрипта переменная будет иметь значение 1, и так далее.

Для того, чтобы иметь доступ к переменным сессии на любых страницах сайта, надо написать только одну (!) строчку в самом начале каждого файла, в котором нам нужны сессии: session_start();И далее обращаться к элементам массива $_SESSION. Например, проверка авторизации будет выглядеть примерно так:session_start(); if ($_SESSION['authorized']<>1) { header("Location: /auth.php"); exit; }

Удаление переменных из сессии. Если у вacregister_globals=off, то достаточно написать unset($_SESSION['var']);Если же нет, то тогда рядомс ней надо написать session_unregister('var');

6.3 Область применения

Во-первых, что сессии можно применять только тогда, когда они нужны самому пользователю, а не для того, чтобы чинить ему препятствия. Ведь он в любой момент может избавиться от идентификатора! Скажем, при проверке на то, что заполняет форму человек, а не скрипт, пользователь сам заинтересован в том, чтобы сессия работала - иначе он не сможет отправить форму! А вот для ограничения количества запросов к скрипту сессия уже не годится - злонамеренный скрипт просто не будет возвращать идентификатор.

Во-вторых. Важно четко себе представлять тот факт, что сессия - это сеанс работы с сайтом, так как его понимает человек. Пришел, поработал, закрыл браузер - сессия завершилась. Этому есть и техническое объяснение. Гарантированно механизм сессий работает только именно до закрытия браузера. Ведь у клиента могут не работать куки, а в этом случае, естественно, все дополненные идентификатором ссылки пропадут с его закрытием. Правда, сессия может пропасть и без закрытия браузера. В силу ограничений, механизм сессий не может определить тот момент, когда пользователь закрыл браузер. Для этого используется таймаут – заранее определенное время, по истечении которого мы считаем, что пользователь ушел с сайта. По умолчанию этот параметр равен 24 минутам. Если вы хотите сохранять пользовательскую информацию на более длительный срок, то используйте куки и, если надо - базу данных на сервере. В частности, именно так работают все популярные

системы авторизации: - по факту идентификации пользователя стартует сессия и признак авторизованности передается в ней. - Если надо "запомнить" пользователя, то ему ставится кука, его идентифицирующая. - При следующем заходе пользователя на сайт, для того, чтобы авторизоваться, он должен либо ввести пароль, либо система сама его опознает по поставленной ранее куке, и стартует сессию. Новую сессию, а не продолжая старую.

В-третьих, не стоит стартовать сессии без разбору, каждому входящему на сайт. Это создаст совершенно лишнюю нагрузку. Не используйте сессии по пустякам – к примеру, в счетчиках. Если же приходится показывать одну и ту же страницу как авторизованным, так и не авторизованным пользователям, то тут поможет такой трюк – стартовать сессию только тем, кто ввел пароль, или тем, у кого уже стартовала сессия. Для этого в начало каждой страницы вместо просто session_start() пишем if (isset($_REQUEST[session_name()])) session_start();таким образом, Мы стартуем сессию только тем, кто прислал идентификатор. Соответственно, надо еще в первый раз отправить его пользователю – в момент авторизации. Если имя и проль верные – пишем session_start()

Пример авторизации пользователя:

login.php

<html>

<head>

<title>Страница авторизации<title>

</head>

<body

<form action="authorize.php" method="post">

Вход

Пользователь <input type="text" name="login" />

Пароль <input type="password" name="pass" />

<input type="submit" name="ok" value=" Ok " />

</form>

</body>

</html>

authorize.php

<? // соединяемся с базой

$conn = mysql_connect("localhost","user","pass");

mysql_select_db("my_db"); // составляем запрос

$query = "select * from users where login='". $_POST['login']."' AND pass='". $_POST['pass']."';";

$q = mysql_query($query); // найден ли кто-нибудь

```
$n = mysql_num_rows($q);
if ($n!=0)
{
// стартуем сессию (можно не в начале файла, т.к. никакого вывода в браузер не было)
session_start();
$value=mysql_fetch_array($q); // записываем логин и емейл в сессию
$_SESSION['user_login']=$value['login'];
$_SESSION['email']=$value['email']; // редиректим (перенаправляем) на главную страницу сайта
Header("Location: index.php");
}
else Header("Location: login.php"); // если юзер не найден, то снова на страницу авторизации
mysql_close();
?>
```

index.php

```
<? session_start(); ?>
<html>
<head>
<title>Главная<title>
</head>
<body
<? if ($_SESSION["login"]) echo "<h1>Привет, " . $_SESSION["login"] . "</h1>"; ?>
<!—весь остальной контент -->
</body>
</html>
```

Если пользователь авторизуется, то в index выведет строку приветствия.

Список функций для работы с сессиями:

session_cache_expire - возвращает окончание действия текущего кэша

session_cache_limiter - получает и/или устанавливает текущий ограничитель кэша

session_decode - декодирует данные сессии из строки

session_destroy - уничтожает все данные, зарегистрированные для сессии

session_encode - шифрует данные текущей сессии как строку

session_get_cookie_params - получает параметры куки сессии

session_id - получает и/или устанавливает текущий session id

session_is_registered - определяет, зарегистрирована ли переменная в сессии

session_module_name - получает и/или устанавливает модуль текущей сессии

session_name - получает и/или устанавливает имя текущей сессии

session_readonly - начинает сессию - реинициализирует замороженные переменные, но не записывает в конец запроса

session_register - регистрирует одну или более переменных для текущей сессии

session_save_path - получает и/или устанавливает путь сохранения текущей сессии

session_set_cookie_params - устанавливает параметры куки сессии

session_set_save_handler - устанавливает функции хранения сессии уровня пользователя

session_start - инициализирует данные сессии

session_unregister - дерегистрирует переменную из текущей сессии

session_unset - освобождает все переменные сессии

session_write_close - записывает данные сессии и конец сессии

Задание

Реализовать следующие пункты на сайте:

1. Реализовать регистрацию и авторизацию пользователей на сайте.
2. Реализовать интерфейс администратора (http://site_name/admin/) для управления текстами и пользователями сайта. Использовать механизм сессий. Реализовать разграничение прав доступа (администратор, обычный пользователь
3. отправка email’а через форму обратной связи с полями «Представьтесь», «Ваш E-mail», «Текст сообщения». Использовать дополнительные заголовки. Письма могут отправлять только авторизованные пользователи и все их письма сохраняются в БД. Администратор может их просмотреть.