PHP: использование механизмов Cookies и Session

Протокол HTTP является протоколом "без сохранения состояния". Это означает, что когда пользователь открывает сначала одну страницу сайта, а затем переходит на другую страницу этого же сайта, то основываясь только на средствах, предоставляемых протоколом HTTP невозможно установить, что оба запроса относятся к одному пользователю. Необходим механизм, при помощи которого можно было бы отслеживать информацию о пользователе в течение одного сеанса связи с Web-сайтом. В PHP такой механизм реализован при помощи сессий (**session**) и куки (**cookies**).

Сессии и куки предназначены для хранения сведений о пользователях при переходах между несколькими страницами. При использовании сессий данные сохраняются во временных файлах на сервере. Файлы с cookies хранятся на компьютере пользователя, и по запросу отсылаются браузером (browser) серверу.

Использование сессий и куки очень удобно и оправдано в таких приложениях как интернет-магазины, форумы, доски объявлений и блоги когда, во-первых, необходимо сохранять информацию о пользователях на протяжении нескольких станиц, а, во-вторых, своевременно предоставлять пользователю новую информацию.

COOKIES

Для установки Cookies используется функция [SetCookie](http://www.php.net/setcookie)(). Для этой функции можно указать шесть параметров, первый из которых является обязательным:

*name* - задает имя (строк), закрепленное за Cookie;  
*value* - определяет значение переменной (строка);  
*expire* - время "жизни" переменной (целое число). Если данный параметр не указать, то Cookie будут "жить" до конца сессии, то есть до закрытия браузера. Если время указано, то, когда оно наступит, Cookie самоуничтожится.  
*path* - путь к Cookie (строка);  
*domain* - домен (строка). В качестве значения устанавливается имя хоста, с которого Cookie был установлен;  
*secure* - передача Cookie через защищенное HTTPS-соединение.  
Обычно используются только три первые параметра.

Пример установки Cookies:

**<?php**

*// Устанавливаем Cookie до конца сессии:*

[SetCookie](http://www.php.net/setcookie)("Test","Value");

*// Устанавливаем Cookie на один час после установки:*

[SetCookie](http://www.php.net/setcookie)("My\_Cookie","Value",[time](http://www.php.net/time)()+3600);

**?>**

При использовании Cookies необходимо иметь в виду, что Cookies должны устанавливаться до первого вывода информации в браузер (например, оператором echo или выводом какой-либо функции). Поэтому желательно устанавливать Cookies в самом начале скрипта. Cookies устанавливаются с помощью определенного заголовка сервера, а если скрипт выводит что-либо, то это означает, что начинается тело документа. В результате Cookies не будут установлены и может быть выведено предупреждение. Для проверки успешности установки Cookies можно использовать такой метод:

**<?php**

*// Устанавливаем Cookie до конца сессии:*

*// В случае успешной установки Cookie, функция SetCookie возвращает TRUE:*

if ([SetCookie](http://www.php.net/setcookie)("Test","Value")) echo "<h3>Cookies успешно установлены!</h3>";

**?>**

Функция [SetCookie](http://www.php.net/setcookie)() возвращает TRUE в случае успешной установки Cookie. В случае, если Cookie установить не удается [SetCookie](http://www.php.net/setcookie)() возвратит FALSE и возможно, предупреждение (зависит от настроек PHP). Пример неудачной установки Cookie:

**<?php**

*// Cookies установить не удастся, поскольку перед отправкой*

*// заголовка Cookie мы выводим в браузер строку 'Hello':*

echo "Hello";

*// Функция SetCookie возвратит FALSE:*

if ([SetCookie](http://www.php.net/setcookie)("Test","Value")) echo "<h3>Cookie успешно установлен!</h3>";

else echo "<h3>Cookie установить не удалось!</h3>";

*// Выводит 'Cookie установить не удалось!'.*

**?>**

Cookie установить не удалось, поскольку перед посылкой заголовка Cookie мы вывели в браузер строку "Hello".

**Чтение значений Cookies**

Получить доступ к Cookies и их значениям достаточно просто. Они хранятся в суперглобальных массивах и $\_COOKIE и $HTTP\_COOKIE\_VARS.

Доступ к значениям осуществляется по имени установленных Cookies, например:

echo $\_COOKIE['my\_cookie'];

*// Выводит значения установленной Cookie 'My\_Cookie'*

Пример установки Cookie и последующего его чтения:

**<?php**

*// Устанавливаем Cookie 'test' со значением 'Hello' на один час:*

[SetCookie](http://www.php.net/setcookie)("test","Hello",[time](http://www.php.net/time)()+3600);

*// При следующем запросе скрипта выводит 'Hello':*

echo @$\_COOKIE['test'];

**?>**

В рассмотренном примере при первом обращении к скрипту устанавливается Cookie "test" со значением "hello". При повторном обращении к скрипту будет выведено значение Cookie "test", то есть строка "Hello".

СЕССИИ

Сессия - это сеанс, во время которого система идентифицирует зашедшего на сайт пользователя. Для этого надо выдать ему уникальный идентификатор и попросить передавать его с каждым запросом. 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 cookies 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 cookies, если 0 - то нет.  
session.use\_trans\_sid если равно 1, то PHP передает его, добавляя к URL и формам, если 0 - то нет.

Если включена только вторая, то cookies 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:

<**[a](http://december.com/html/4/element/a.html)** href="/index.php">Index</**[a](http://december.com/html/4/element/a.html)**>  
превращается в

<**[a](http://december.com/html/4/element/a.html)** href="/index.php?PHPSESSID=9ebca8bd62c830d3e79272b4f585ff8f">Index</**[a](http://december.com/html/4/element/a.html)**>  
а к формам добавляется скрытое поле

<[**input**](http://december.com/html/4/element/input.html) 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!

Затем, на сервере создается файл, имя которого совпадает с этим идентификатором. В этот файл могут писаться какие-то данные из ассоциативного массива $\_SESSION.

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](http://www.php.net/session_start)();

И далее обращаться к элементам массива $\_SESSION:  
$\_SESSION['test']='Hello world!';

Например, проверка авторизации будет выглядеть примерно так:

[session\_start](http://www.php.net/session_start)();

if ($\_SESSION['authorized']<>1) {

[header](http://www.php.net/header)("Location: /auth.php");

[exit](http://www.php.net/exit);

}

**Удаление переменных из сессии**

Если в настройках PHP включена переменная register\_globals=off, то достаточно написать

[unset](http://www.php.net/unset)($\_SESSION['var']);

Если же нет, то тогда рядом с ней надо написать:

[session\_unregister](http://www.php.net/session_unregister)('var');

Сравнение механизмов идентификации пользователей при помощи Cookies и Session

**Cookies** — это просто пара имя-значение, которые сервер может оставить у клиента (браузера). Наглядно:

1. Приходит клиент, спрашивает у сервера страницу.  
2. Сервер по IP определяет геолокацию клиента и в заголовках ответа может установить cookies. Например, выдав заголовок: Set-Cookie: country=russia. При следующем обращении серверу уже не надо проверять IP, он может взять готовое значение из cookies, если они разрешены у клиента.

**Сессии** — это механизм, который позволяет отличить одного клиента от другого и хранить связанные с ним данные.

Как правило, сессии реализуются используя cookies и идентификаторы сессий. Т.е. сервер со своей стороны создает уникальный идентификатор, например, «1a2b3c» (session\_id), а клиента просит его запомнить. Обычно — при помощи cookies, говоря что-то в духе Set-Cookie: PHPSESSID=1a2b3c (где «PHPSESSID» — имя сессии). Со своей стороны сервер где-то в файле хранит различные данные, которые ему приказано связывать с этой сессией.

Если куки у пользователя отключены, то перед отдачей страницы пользователю сервер переписывает все ссылки в ней, добавляя к ним параметр «?PHPSESSID=1a2b3c». Таким образом сервер получает идентификатор сессии.

**Фактически механизм Cookies хранит данные о пользователе на стороне клиента, т.е. у самого пользователя, а механизм сессий хранит у пользователя только идентификатор клиента. Сами данные, связанные с этим пользователем, хранятся на сервере.**

Задание. Работа с Cookies и Session

Тема: Создание и использование Cookies. Открытие и работа с Session. Отличие механизмов Cookies от Session.

Создайте два PHP-скрипта: index.php и test.php. Сделайте в первом скрипте index.php форму опроса пользователя с предложенными ниже в заданиях параметрами (или в варианте 8-9 просто заполните параметры нужными значениями). Сохраните значения введенных параметров в переменных массива $\_COOKIE или $\_SESSION. Передайте управление скрипту test.php без использования GET или POST запросов.

1. В index.php предложите пользователю выбрать цвет ссылок и цвет фона страницы с помощью выпадающего списка. В test.php окрасьте страницу и гиперссылки в тексте в выбранные цвета.

2. В index.php спросите с помощью формы город и возраст пользователя. В test.php сделайте форму с полями 'Имя', 'Возраст', 'Город' причем поля 'Возраст' и 'Город' заполните уже известными значениями.

3. В index.php сделайте 5 картинок с товарами и ценой. Реализуйте корзину в test.php. Под каждой картинкой должна быть ссылка 'положить в корзину'. По нажатию на эту ссылку этот товар должен попасть в корзину и посчитаться общая сумма товара, которую должен заплатить пользователь.

4. На страницах index.php и test.php покажите пользователю баннер с кнопкой 'Не показывать больше'. Если он нажмет на эту кнопку — не показывайте ему баннер в течении 5 минут.

5. В index.php спросите дату рождения пользователя. В test.php напишите сколько дней осталось до его дня рождения. Если сегодня день рождения пользователя — поздравьте его!

6. В index.php создайте форму теста-опросника с двумя вопросами и несколькими вариантами ответов. В test.php  
подведите итоги опроса. Разрешите пользователю менять свои ответы.

7. Создайте вариант игры "Спортлото 6 из 36". В index.php пользователь выбирает 6 номеров из 36. В test.php показывается количество угаданных.

8. Сделайте счетчик обновления страницы index.php пользователем. Скрипт test.php должен выводить на экран количество обновлений страницы index.php. При первом заходе на страницу index.php он должен вывести сообщение о том, что вы еще не обновляли страницу.

9. Запомните время первого захода пользователя на страницу index.php. На странице test.php выводите сколько секунд назад пользователь зашел на сайт.