Часть 2. Практическая работа №5

Cookies и сессии PHP.

В этом Практическая работа вы:

* узнаете, что такое Cookies и с чем их едят
* научитесь манипулировать Cookies средствами PHP
* узнаете, что такое сессии PHP и как они работают
* научитесь делать авторизацию на сайте

1. Что такое Cookies и с чем их едят

Cookies переводятся с английского языка как «печеньки». В контексте веб - это небольшой кусочек текстовой информации. Куки нужны для того, чтобы сайт мог сохранить некоторые данные на компьютере пользователя. Например, когда вы авторизуетесь на сайте и отмечаете галочку "запомнить меня", запоминают вас в куках браузера. Если вы оставляли комментарии на сайте Школы Программирования более одного раза, то могли заметить, что при отправке второго комментария ваше имя и адрес email уже введены. Это также сделано благодаря cookies.

Куки различных сайтов изолированы друг от друга. Представляют они собой пары имя- значение. Для простоты можно представить их как текстовый файл такого вида:

yandex.ru:

login = dima

password = a123

some name = some value

prog-school.ru:

comment name = dr.zlo

comment email = [dmlyapin@yandex.ru](mailto:dmlyapin@yandex.ru)

svitter.ru:

comment name = dr.zlo

comment email = [dmlyapin@yandex.ru](mailto:dmlyapin@yandex.ru)

На самом деле каждый браузер хранит это по-разному, что сейчас не имеет для нас с вами никакого значения.

Управляет куками удаленный сервер, то есть он может их читать, добавлять и изменять. А для хранения информации используется компьютер пользователя.

Как это работает?

Запрашивая страницу, браузер отправляет веб-серверу короткий текст с HTTP-запросом. Например, для доступа к странице <http://www.example.org/index.html>, браузер отправляет на сервер [www.example.org](http://www.example.org) следующий запрос:

GET /index.html НТТР/1.1 Host: [www.example.org](http://www.example.org)

Сервер отвечает, отправляя запрашиваемую страницу вместе с текстом, содержащим HTTP-ответ. Там может содержаться указание браузеру сохранить куки:

НТТР/1.1 200 ОК Content-type: text/html Set-Cookie: name=value

(содержимое страницы)

Строка Set-cookie отправляется лишь тогда, когда сервер желает, чтобы браузер сохранил куки. В этом случае, если куки поддерживаются браузером и их приём включён, браузер запоминает строку name=value (имя = значение) и отправляет её обратно серверу с каждым последующим запросом. Например, при запросе страницы <http://www.example.org/spec.html> браузер пошлёт серверу [www.example.org](http://www.example.org) следующий запрос:

GET /spec.html НТТР/1.1 Host: [www.example.org](http://www.example.org) Cookie: name=value Accept: \*/\*

Этот запрос отличается от первого запроса тем, что содержит строку, которую сервер отправил браузеру ранее. Таким образом, сервер узнает, что этот запрос связан с предыдущим. Сервер отвечает, отправляя запрашиваемую страницу и, возможно, добавив новые куки.

Кроме пары имя/значение, куки может содержать срок действия, путь и доменное имя. Домен и путь говорят браузеру, что куки должна быть отправлена обратно на сервер при запросах URL для указанного домена и пути. Если они не указаны, используются домен и путь запрошенной страницы.

Фактически, куки определяются тройкой параметров имя-домен-путь. Иными словами, куки с разными путями или доменами являются разными, даже если имеют одинаковые имена.

Дата истечения указывает браузеру, когда удалить куки. Если срок истечения не указан, куки удаляется по окончании пользовательского сеанса, то есть с закрытием браузера. Дата истечения указывается в формате «Нед, ДД-Мес-ГГГГ ЧЧ:ММ:СС GMT». Например:

Set-Cookie: RMID=732423sdfs73242; expires=Fri, 31-Dec-2010 23:59:59 GMT; path=/; domain=.example.net

Куки из примера выше имеет имя RMID и значение «732423sdfs73242». Срок его хранения истечёт 31 декабря 2010 года в 23:59:59. Путь «/» и домен «example.net» показывают браузеру, что нужно отправить куки при просмотре любой страницы в домене example.net.

1. Манипулируем Cookies средствами PHP

Все же PHP - это язык, придуманный для людей. Оценить его удобство и естественность можно и на примере работы с куками.

Чтобы установить куку на период сессии (до закрытия браузера) воспользуйтесь функцией setcookie(). Например, вот так:

setcookie("name", "value");

Если вы хотите запомнить куку на некоторое время, предположим, на неделю, воспользуйтесь той же функцией, но с тремя параметрами:

setcookie("name", "value", time() + 3600 \* 24 \* 7);

Последний параметр здесь - время истечения куки. Он указывается в формате timestamp. Timespamp - это число секунд, прошедших с 00:00:00 1 января, 1970 года. Функция time() возвращает текущее время, к нему мы прибавляем неделю (3600 \* 24 \* 7 секунд).

Функцию setcookie() можно вызывать и с большим числом параметров, если есть необходимость установить прочие атрибуты куки. Однако в большинстве случаев достаточно двух приведенных выше примеров.

Чтобы прочитать cookies, следует воспользоваться системной переменной $\_COOKIES (словарь):

echo $ COOKIE['name'];

Проверить, установлена ли кука, можно так:

if (isset($ COOKIE['name']))

echo $ COOKIE['name'];

Кстати, используя словарь $\_COOKIES можно и сохранять значения куков, но только на период сессии (без указания времени истечения). Операция

$ COOKIE['name'] = 'value';

аналогична

setcookie("name", "value");

1. Что такое сессии PHP и как они работают

Протокол HTTP является протоколом "без сохранения состояния". То есть не имеет встроенной возможности поддерживать сеанс работы с сайтом.

Представьте интернет-магазин. Пользователь выбирает товар, кладет его в корзину, а затем оплачивает. Для оформления покупки нужно посетить несколько страниц. Причем сайт должен понимать, что это один и тот же пользователь, а также запоминать товары, которые он отложил в корзину.

Для решения таких задач в PHP реализован механизм поддержки сессий. Если есть необходимость запоминать состояние сеанса, все, что нужно сделать - вызвать в начале скрипта функцию session\_start().

session start();

Эта функция проверит, существует ли идентификатор сессии. Если нет, то он будет выделен и создастся файл, в котором можно будет сохранять информацию, актуальную в рамках сессии (например, список товаров в корзине).

Но где искать этот самый идентификатор сессии?

PHP предлагает два механизма:

1. хранить идентификатор сессии в куках
2. добавлять идентификатор сессии к внутренним ссылкам в качестве параметра URL

Последний вариант имеет смысл только тогда, когда в браузере отключены cookies. Он нехорош тем, что внутренние ссылки станут выглядеть очень некрасиво. Например, ссылка

index.php?page=main

превратится в

index.php?page=main & PHPSESSID=ldc9 fcb7 3 la 12 3ecl6fb2e49ece325ed

Я рекомендую пользоваться только первым вариантом. Сейчас все современные браузеры поддерживают работу с cookies и можно смело полагаться, что сессии на вашем сайте будут работать с их помощью. Если пользователь отключил cookies в браузере, можно считать, что ему не нужна поддержка сеансов работы с веб-сайтами.

Информация, актуальная в рамках сессии, хранится в системной переменной (словаре) $\_SESSION. Вот так можно сохранить данные:

$ SESSION['username'] = 'Vasya';

А вот так прочитать:

echo 'Привет, ' . $ SESSION['username'] . '!';

Когда сессия больше не нужна, например, пользователь нажал кнопку "Выход", следует уничтожить ее. Для этого существует функция session\_destroy(). Однако перед ее вызовом нужно удалить все сохраненные в сессии данные. Например, вот так:

unset($ SESSION['username']); session destroy();

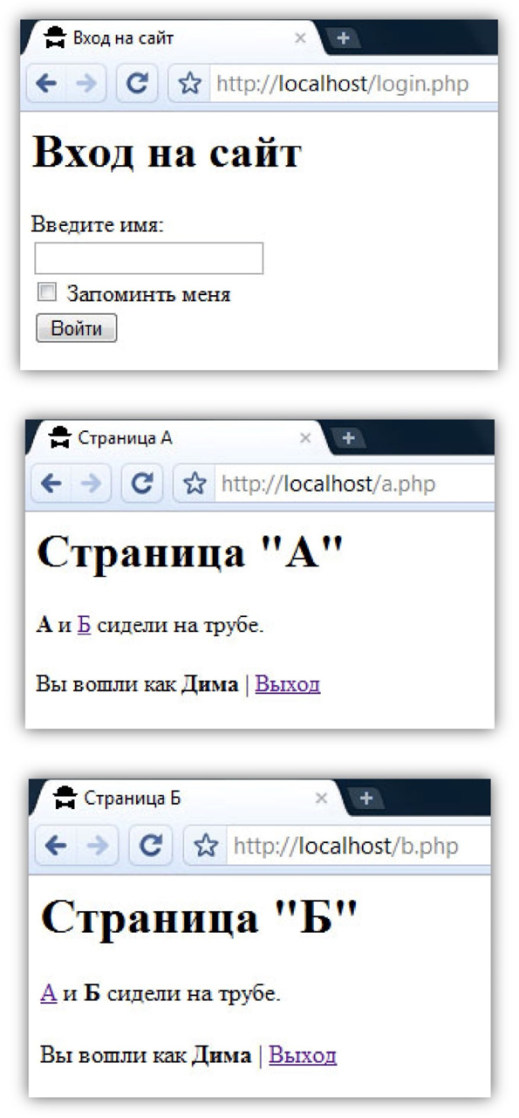
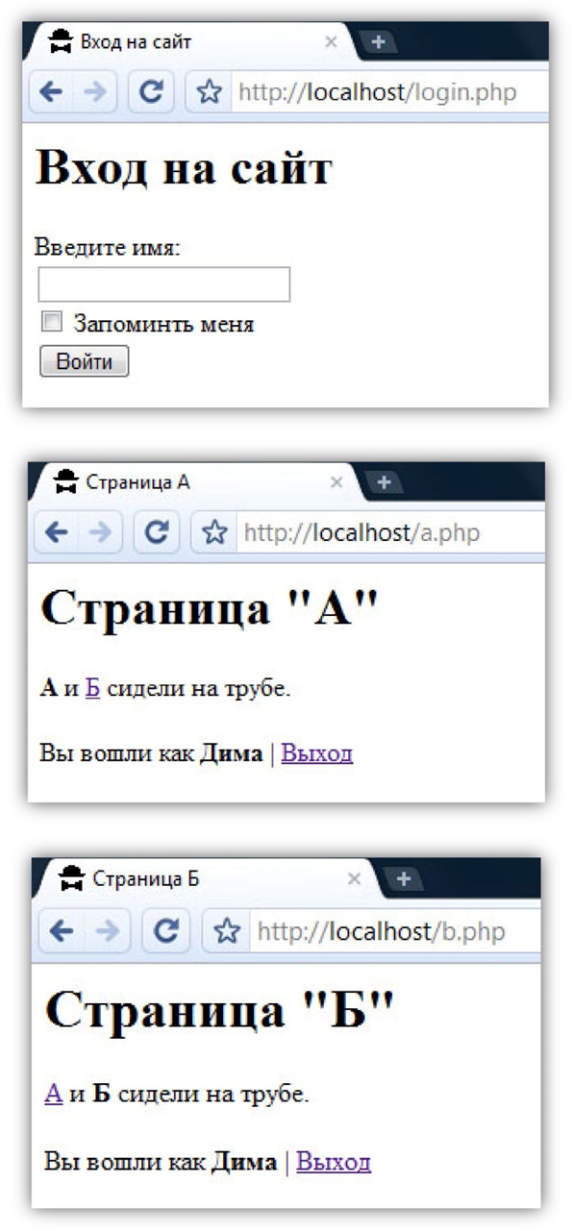
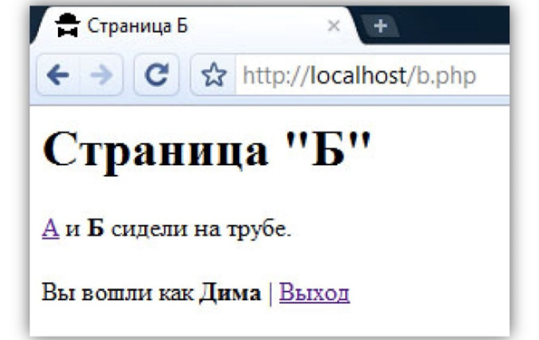
1. Делаем авторизацию на сайте

Переходим к практике. Сейчас мы сделаем сайт с авторизацией.

Он будет состоять из трех страниц:

1. страница авторизации;
2. страница с буквой «А»;
3. страница с буквой "Б".

Страницы "А" и "Б" могут посещать только авторизованные пользователи. Выглядеть это будет следующим образом:

Начнем со страницы login.php: <?php

//

// Авторизация.

//

function Login($username, $remember)

{

// Фмя не должно быть пустой строкой. if ($username == '')

**return false**;

// Запоминаем имя в сессии $ SESSION['username'] = $username;

// и в cookies, если пользователь пожелал запомнить его (на неделю). if ($remember)

setcookie('username', $username, time() + 3600 \* 24 \* 7);

// Успешная авторизация.

**return true**;

}

//

// Сброс авторизации.

//

**function** Logout()

{

// Делаем cookies устаревшими (единственный способ их удаления). setcookie('username', '', time() - 1);

// Сброс сессии.

unset($ SESSION['username']);

} \_

//

// Точка входа.

//

session start();

$enter site = false;

// Попадая на страницу login.php, авторизация сбрасывается.

Logout () ;

// Если массив POST не пуст, значит, обрабатываем отправку формы. if (count($ POST) > 0)

$enter site = Login($ POST['username'], $ POST['remember'] == 'on');

// Переадресуем авторизованного пользователя на одну из страниц сайта. if ($enter site)

{ \_

header("Location: a.php");

**exit**();

}

?>

<**html**>

<**head**>

^^^>Вход на сайт</title>

</**head**>

<**body**>

<^>Вход на сайт<^1>

Введите имя:

<br/>

<input type="text" name="username" />

<br/>

<input type="checkbox" name="remember" /> Запоминть меня <br/>

<input type="submit" уа1ие="Войти" />

</**form**>

</**body**>

</**html**>

Комментарии в коде вполне позволяют понять смысл происходящего. При вводе имени пользователь перенаправляется на страницу a.php.

Вот так происходит перенаправление:

header("Location: a.php"); exit () ;

Вызовом функции header() мы отправляем заголовок, говорящий браузеру о том, что необходимо обратиться к другой странице. Функция exit() прекращает выполнение скрипта. Ее мы вызываем для того, чтобы HTML формы авторизации не был передан в ответе.

Переходим к странице a.php:

<?php

//

// Точка входа.

//

session start();

// Если в контексте сессии не установлено имя пользователя, пытаемся взять его

// из cookies.

if (!isset($ SESSION['username']) && isset($ COOKIE['username']))

$ SESSION['username'] = $ COOKIE['username'];

// Еще раз ищем имя пользователя в контексте сессии.

$username = $ SESSION['username'];

// Неавторизованных пользователей отправляем на страницу регистрации. if ($username == null)

{

header("Location: login.php");

**exit**();

}

?>

* **html** >

<**head**>

<^^е>Страница А</title>

</**head**>

* **body**>

<Ъ1>Страница "А"</Ь1>

<Ь>А</Ь> и <a href="b.php"f</a> сидели на трубе.

<br/>

<br/>

Вы вошли как <b><?php echo $username; ?>< / > | <a href="login.php">Выход</a>

</**body**>

</**html**>

Вначале мы проверяем факт авторизации. Неавторизованные пользователи отправляются на страницу входа, доступ к содержимому a.php им закрыт.

Рассмотрим подробнее проверку авторизации. Если имени пользователя нет в контексте сессии, то оно может быть запомнено в cookies. В этом случае копируем его из cookies в файл сессии.

if (!isset($ SESSION['username']) && isset($ COOKIE['username']))

$ SESSION['username'] = $ COOKIES['username'];

Затем снова проверяем контекст сессии.

$username = $ SESSION['username'];

Если и на этот раз имя пользователя найти не удалось, значит отправляем его на страницу входа.

if ($username == null)

{

header("Location: login.php"); exit () ;

}

Ссылка "Выход" ведет на страницу входа, которая сбрасывает данные авторизации.

Страница b.php почти такая же: <?php

//

// Точка входа.

//

session start();

// Если в контексте сессии не установлено имя пользователя, пытаемся взять его

// из cookies.

if (!isset($ SESSION['username']) && isset($ COOKIE['username']))

$ SESSION['username'] = $ COOKIE['username'];

// Еще раз ищем имя пользователя в контексте сессии.

$username = $ SESSION['username'];

// Неавторизованных пользователей отправляем на страницу регистрации. if ($username == null)

{

header("Location: login.php");

**exit**();

}

?>

< **html** >

<**head**>

<^^е>Страница Б<^^^>

</**head**>

<**body**>

<^>Страница '^"</h1>

<a href="a.php">A</a> и <Ь>Б</Ь> сидели на трубе.

<br/>

<br/>

Вы вошли как <b><?php echo $username; ?>< / > | <a href="login.php">Выход</a>

</**body**>

</ **html** >

Резюме

Теперь вы можете сделать сайт с сохранением состояния между запросами, например, интернет-магазин. Служит для этого механизм сессий, который в свою очередь реализован на основе Cookies (как правило).

Cookies можно использовать и на период более длительный, чем сессия для хранения различной информации. Например, для блога с возможностью оставлять комментарии без регистрации, удобно хранить псевдоним и email пользователя.

PHP предоставляет удобные механизмы работы с сессиями и куками: это системные переменные $\_SESSION и $\_COOKIE, а также функции session\_start(), session\_destroy(), setcookie().

Задания:

1. Создайте главную страницу сайта index.php, которая будет неавторизованных пользователей отправлять на страницу авторизации, а авторизованных на ту страницу, которую они посещали последний раз ("А" или "Б"). Для пользователя главной страницы не будет видно, она нужна только для перенаправления.
2. Придумайте, чем еще дополнить сайт и реализуйте это! Дополнение должно быть связано с сессиями и/или куками. Творческий подход, как всегда, приветствуется.