Лабораторная работа № 5 "Cookies и сессии PHP"

1. Что такое Cookies

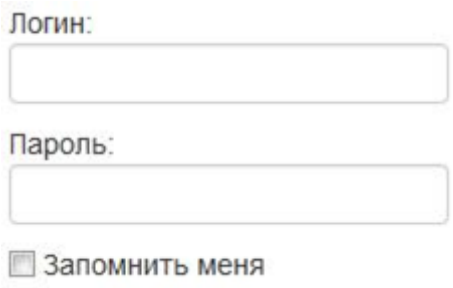
Cookies (в дальнейшем «куки») – это небольшой кусок текстовой информации, которую сервер сохраняет в браузере пользователя.

Для начала разберёмся с тем, какой смысл несут в себе куки. Благодаря им мы можем идентифицировать пользователя, когда тот заходит на наш сайт.

Что же под этим подразумевается? Представим конкретную задачу: мы делаем интернет - магазин. На определённой страничке сайта пользователь может добавить товар в корзину. После этого он переходит на другую страницу, например, на страницу оформления заказа, вот тут начинается самое интересное. Дело в том, что сервер не может понять, что запрос пришёл от того же самого пользователя, который секунду назад положил товар в корзину. Каждый человек кажется ему уникальным посетителем, потому что сервер ориентируется только на текущий HTTP-запрос. В результате, после того, как пользователь перешёл на другую страницу, его корзина товаров снова стала пустой.

Это произошло по той причине, что мы не смогли понять, какой именно пользователь сделал запрос. Для нас они все одинаковы. А для того, чтобы уметь отличать пользователей друг от друга, нужно ставить им метки. Куки как раз такими метками и являются.

Мы этого не замечаем, но многие сайты сохраняют метки у нас на компьютере. Пожалуй, самым ярким и распространённым примером использования кук является галочка «запомнить меня» при авторизации на сайте.



Вспомните, если Вы ставите галочку, то сайт может неделю или месяц, а то и больше, не запрашивать у Вас логин и пароль снова. Дело в том, что эти два значения были сохранены в куки, и браузер теперь вместо Вас вводит логин и пароль для сайта.

Как работать с куками в PHP мы разберём чуть позже, а пока что посмотрим на основные принципы их функционирования.

Итак, кука – это небольшой кусок текстовой информации, сохранённый в браузере пользователя. Как именно она там хранится, нас совершенно не интересует. Главное, что каждая кука представляет собой пару имя=значение.

В предыдущем примере куки могли бы выглядеть примерно таким образом:

site.ru:
login = lamer
password = qwerty

Обратите внимание на то, что мы указали домен, которому принадлежат куки. Дело в том, что куки различных сайтов изолированы друг от друга, т.е. браузер в рамках HTTP-запроса будет передавать серверу только те метки о пользователе, которые принадлежат именно данному сайту.

```
site.ru:
login = lamer                 // Данные куки будут использоваться при посещении
password = qwerty             // сайта site.ru
yandex.ru:
login = qwerty                // А эти - при посещении Яндекса
password = 13572468
```

Изоляция является одной из важнейших особенностью кук и связана, в первую очередь, с безопасностью посещения сайтов. При таком подходе ни один домен не сможет считать чужие куки и начать использовать их в своих целях.

Ещё одним важным моментом для понимания механизма работы кук является то, что они хранятся на компьютере у пользователя, но все команды по добавлению, изменению и удалению этих меток отправляет именно сервер.

В примере, показанном на данной схеме, сервер посылает куку пользователю, который впервые посещает сайт. Затем, при каждом повторном заходе, браузер клиента будет передавать метку, на которую мы сможем ориентироваться в PHP-скрипте.

Кроме пары имя=значение, каждая кука содержит срок действия, путь и доменное имя.

Домен и путь говорят браузеру, в каких случаях кука должна быть отправлена на сервер.

Дата истечения указывает браузеру, когда удалить куку. Если срок истечения не указан, кука 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: login=qwerty; 31-Dec-2013 23:59:59 GMT; path=/; domain=.site.ru
```

Кука из примера выше имеет имя «login» и значение «qwerty». Срок её хранения истечёт 31 декабря 2013 года в 23:59:59. Путь «/» и домен «site.ru» показывают браузеру, что нужно отправить куки при просмотре любой страницы данного сайта.

• 2. Работа с Cookies

При работе с куками в PHP необходимо уметь совершать три основные операции:

1. отправка куки пользователю;
2. проверка наличия куки;
3. чтение данных из куки.

Для установки куки существует специальная функция, которая называется setcookie. Она может принимать различное количество параметров, но чаще всего работа с ней выглядит следующим образом:

setcookie($name, $value, $time);

где $name – имя куки, $value – её значение, $time – срок истечения.

Например,

setcookie("login", "lamer", time() + 3600 * 24 * 7); **setcookie**("password", "qwerty", time() + 3600 * 24 * 7);

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

После того, как куки установлены, при каждом заходе пользователя на страничку PHP-интерпретатор формирует глобальный массив $_COOKIE. Ключами в нём являются названия кук, по которым хранятся их значения.

В итоге, чтения и проверка наличия кук сводятся к стандартной работе с ассоциативным массивом, например:

if (**isset**($_COOKIE['login']))

echo $_COOKIE['login'];

В данном фрагменте, когда мы сначала проверили, существует ли вообще у пользователя кука с именем login, и, в случае истинности условия, прочитали её и вывели на экран.

Работа со всеми остальными куками ведётся абсолютно аналогичным образом.

3. Что такое сессии PHP и как они работают

Представьте, что Вам необходимо решить следующую задачку. Есть две страницы: index.php и login.php. На странице логина пользователь вводит свои данные в поля формы и отправляет запрос. Эти данные мы могли сохранить в переменные, например:

$name = $_POST['name']; // Какие-то данные, которые ввел пользователь
$age = $_POST['age'];

- затем показать их на страничке login.php. Однако теперь пользователь может перейти по ссылке на index.php, и там мы также должны выводить эти переменные. Как Вы думаете, получится ли это сделать, просто написав на индексе

echo $name;
echo $age;

Правильный ответ – «нет!», потому что переменные были стёрты из памяти при завершении скрипта login.php. Получается, что эти данные нам нужно куда-нибудь сохранить. Мы видели, как сохранять информацию в куках. Однако их следует использовать только для важных вещей, так как куки постоянно передаются от клиента к серверу в рамках запросов к сайту. И если кук много, то серьёзно засоряется HTTP-пакет.

В связи с этим был создан механизм сессий. По сути, сессия – это возможность сохранять данные при переходе между страницами. Общая идея заключается в следующем: для того, чтобы однозначно идентифицировать клиента, достаточно одной метки, а вся информация теперь хранится на сервере в специальном файле. Кука лишь указывает на принадлежность файла конкретному пользователю.

Для создания сессии существует специальная функция session_start(). Вот так можно схематично изобразить её работу:

PHPSESSID используется как указатель на конкретный файл. Он также может быть сохранён не в куку, а передаваться методом GET. Однако, сейчас это практически не используется, так как приводит к некрасивым длинным URL-адресам.
Всю работу по записи и чтению информации из файла PHP выполняет автоматически.
Нам лишь остаётся работать с сессией как с ассоциативным массивом.

Например, так мы могли бы сохранить данные в предыдущем примере:

```
session_start();

$_SESSION['name'] = $_POST['name'];

$_SESSION['age'] = $_POST['age'];
```

Для того, чтобы поприветствовать пользователя на странице index, теперь достаточно написать:

```
session_start();

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

echo 'Привет, ' . ($_SESSION['name'];
```

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['name']);

unset($_SESSION['age']);

session_destroy();
```

Сессии в PHP являются механизмом, который гармонично дополняет куки. И то, и другое служит для запоминания данных о пользователе, но у каждого из них есть своё назначение. Куки – это долговременное хранилище, а сессии – кратковремнное. Сессия сама уничтожится после закрытия браузера пользователем.

- **Делаем авторизацию на сайте**

Авторизация на сайте организуется с помощью кук и сессий. Здесь мы проговорим общие идеи, а конкретную реализацию Вам необходимо будет посмотреть в исходниках к уроку.

На одной из страниц пользователь вводит логин, пароль и ставит галочку, следует ли его запомнить. При обработке отправки формы мы точно запишем данные в сессию и сохраним в куку, если пользователь галочку поставил.

После этого на каждой странице сайта нужно проверять наличие сохранённого значения в сессии и куках, так как неавторизованный пользователь также может пытаться попасть на страницу, просто вбив её URL-адрес.

При проверке наличия значения в сессии и куках может возникнуть четыре ситуации:

Сессия	нет	есть	нет	есть
Кука	нет	нет	есть	есть
Пояснение ситуации	пользователь не авторизован	пользователь авторизован, не ставил галочку «запомнить»	пользователь долго отсутствовал на сайте, сессия удалилась	пользователь авторизован, ставил галочку «запомнить»

На основе информации, которую мы найдём в массивах $_COOKIE и $_SESSION, уже и будем принимать решение, что дальше делать с пользователем.

Обязательно скачайте и запустите исходники с примером организации авторизации на сайте.

Контроль

1. Что такое куки
2. Где хранятся куки
3. Куки различных сайтов изолированы друг от друга, это хорошо или плохо
4. Из каких основных частей состоит кука
5. Как с помощью PHP установить куку
6. Как с помощью PHP проверить наличие куки
7. Как с помощью PHP получить значение куки
8. Как с помощью PHP удалить куку
9. Зачем нужны сессии в PHP
10. Отличия хранения данных в Cookies и сессиях
11. Какими способами можно передавать идентификатор сессии
12. Время жизни сессии

Задание

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 не будет видно главной страницы, она нужна только для перенаправления.
2. Создайте три css файла с разными стилями. Затем сделайте страничку setting.php, на которой пользователь сможет выбрать себе один из вариантов оформления сайта. Информация о стиле сохраняется в куках и затем используется при показе страничек.