Cookies

Определение

Cookies – небольшой фрагмент данных, отправляемый веб-сервером и хранимый на компьютере пользователя.

Принцип работы:

1. сервер в HTTP-заголовках передает файлы cookies;
2. браузер сохраняет полученные файлы cookies;
3. при загрузке следующей страницы из того же домена, что и страница, содержавшая cookies, браузер возвращает те cookies, которые уже были сохранены ранее;
4. при получении cookies PHP преобразовывает их в переменные. Они сохраняются в глобальном массиве $_COOKIE.

Посылка cookie

```
int setcookie (
    string name [, string value [, int expire
    [, string path [, string domain [, bool secure [, bool httponly >]
)
```

Функция посылает переменную cookie в заголовке HTTP.

name – имя cookie.

value – значение, которое необходимо сохранить.

expire – время хранения cookie в формате UNIX time.

path, domain – можно указать список каталогов и домен, для которого необходимо возвращать значения cookie.

secure – передать cookie в зашифрованном виде через протокол HTTPS.

httponly – cookie будут доступны только через HTTP протокол.

В этом случае не будут доступны скриптовым языкам, вроде JavaScript.

Примеры установки и удаления cookie-файлов

```
setcookie('TestCookie1', 'Test Value');

# Устаревает через час
setcookie('TestCookie2', $value, time() + 3600);

setcookie(
    'TestCookie3', $value, time() + 3600,
    '/~igor/', '.spb.ru', 1, 1
);
```

```
# Удаляет cookie
setcookie('TestCookie1');

# Set the expiration date to one hour ago
setcookie('TestCookie2', '', time() - 3600); #время уже прошло

setcookie(
    'TestCookie3', '', time() - 3600, '/~igor/', '.spb.ru', 1, 1
);
```

Недостатки использования cookies

1. Ограниченный объем хранимых данных.
2. Кража и подмена cookies.
3. Для одной страницы нельзя установить более 20 cookies.
4. Время жизни cookies.
5. Не точная идентификация пользователя.
6. Могут быть отключены в браузере.

Задача №1

Используя cookies посчитать сколько раз пользователь обновлял страницу и вывести значение на экран.

```php
// Ваш код.

echo "Вы посетили эту страницу: $count раз.";
```

Сессии

Определение

Поддержка сессий в PHP заключается в способе сохранения данных между несколькими последовательными запросами пользователя.

Пользователю присваивается уникальный идентификатор сессии. Он хранится либо в cookie, либо передается через URL.

Данные между запросами сохранаются в суперглобальном массиве $_SESSION.

Когда пользователь выполняет запрос, PHP проверяет наличие идентификатора сессии и восстанавливает сохраненное ранее окружение.

Параметры сессии

string session_name ([string name])

Возвращает имя текущей сессии.

Если передан аргумент name, то изменяет имя сессии.

string session_id ([string id])

Возвращает идентификатор текущей сессии.

Если передан аргумент id, то изменяет идентификатор сессии.

string session_save_path ([string path])

Возвращает имя каталога, в котором сохранаются файлы сессий.

Если передан аргумент path, то изменяет каталог на переданный.

void session_set_cookie_params (<эквивалентны setcookie()>)

Установка параметров cookie сессии (кроме имени и значения).

array session_get_cookie_params (void)

Возвращает параметры cookie сессии.

Инициализация/уничтожение данных сессии

bool session_start (void)

Создает массив данных сессии или восстанавливает ранее сохраненные переменные сессии.

bool session_destroy (void)

Уничтожает все данные, принадлежащие текущей сессии.

bool session_regenerate_id (void)

Заменяет значение идентификатора текущей сессии новым автоматически сгенерированным значением.

Все данные сессии сохранаются.

Манипуляции с переменными сессии

```
$_SESSION['username'] = 'kyzima-spb';
```

Добавление переменной сессии.

```
unset($_SESSION['username']);
```

Удаление переменной сессии.

```
isset($_SESSION['username']);
```

Проверка принадлежности переменной к сессии.

```
void session_unset ( void )
```

Удаление всех переменных текущей сессии.

Примеры

```php
session_name('kyzima_vk');
session_start();

if (!isset($_SESSION['token'])) {
    $_SESSION['token'] = new VKApi()->getToken();
}
```

```php
session_start();

$_SESSION['count'] = isset($_SESSION['count'])
    ? ++$_SESSION['count'] : 0;
```

Задача №2.Часть 1

Простая система аутентификации.

Для хранения информации о пользователе использовать CSV файл.

```
id;username;password;sess_id
id;username;password;sess_id
id;username;password;sess_id
```

Реализовать функцию добавления пользователя в файл.

Идентификатор пользователя и имя должны быть уникальными. Реализовать функцию генерации уникального идентификатора. Реализовать функцию поиска совпадений по имени пользователя.

Пароль нельзя хранить в открытом виде, поэтому необходимо его захешировать с помощью функции md5().

В поле sess_id хранится идентификатор сессии. Это поле будет обновляться при каждом новом входе пользователя на сайт. Учесть и реализовать функцию обновления информации о пользователе по уникальному идентификатору.

Задача №2. Часть 2

Сверстать HTML-форму для регистрации пользователя.

Написать обработчик регистрационной формы: операция добавить нового пользователя в файл (логин пользователя должен быть уникальным).

Сверстать HTML-форму входа пользователя на сайт.

Написать обработчик формы входа, который будет проверять существование пользователя в файле, сверять введенный хеш пароля и хеш из файла на равенство и заносить в файл новый идентификатор сессии в случаи успешного входа.

Создать закрытую страницу сайта, которая будет доступна только тем пользователям, которые прошли регистрацию и выполнили вход на сайт.

Домашнее задание

Для уникальной идентификации пользователя одного идентификатора сессии не достаточно, так как данный идентификатор может быть подменен.

Для того, чтобы себя обезопасить, вместо идентификатора сессии лучше использовать хэш, состоящий из: идентификатора сессии, ip адреса пользователя и имени его браузера (user agent). Такой хэш иногда называют *токеном*.

Модернизировать задачу 2 с использованием токена.

Конец