Лабораторная работа №4

Авторизация и регистрация

Теоретический раздел

Работа с cookies (печеньками)

Cookie – это текстовые строки, которое могут быть сохранены на компьютере клиента и затем передаются веб-серверу при каждом обращении к нему. Например, при авторизации на сайте и установки галочки «запомнить меня», вас запоминают в куках браузера. При повторном добавлении комментария, такие параметры как имя, электронный адрес уже введены – это тоже работа cookies.

Управляет куками удаленный сервер, то есть он может их читать, изменять, добавлять. Для хранения используется компьютер пользователя. 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 следующий запрос:

```
GET /index.html HTTP/1.1
Host: www.example.org
```

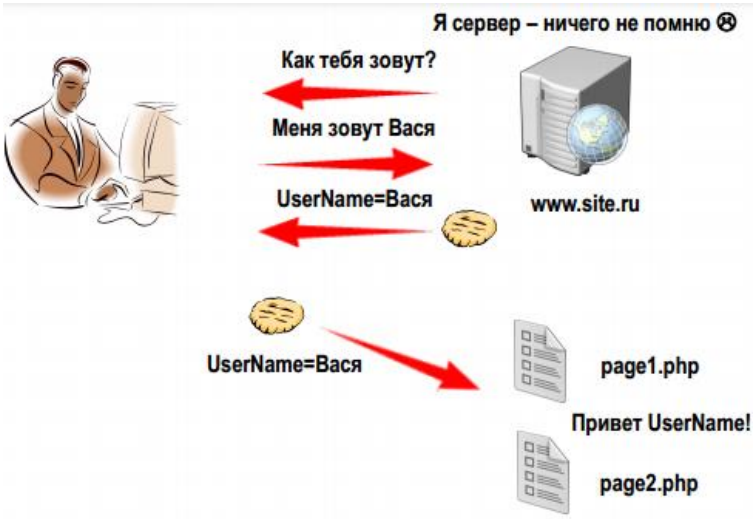
Сервер отвечает, отправляя запрашиваемую страницу вместе с текстом, содержащим HTTP-ответ. Там может содержаться указание браузеру сохранить куки:

```
HTTP/1.1 200 OK
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 браузерами 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 следующий запрос:

```
GET /spec.html HTTP/1.1
Host: 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.



Для установки значения cookie в PHP используется функция setcookie:

bool **setcookie** (string name [, string value [, int expire [, string path [, string domain [, int secure]]]]])

Параметры функции:

- name – имя куки (только латинские буквы, цифры, символ подчеркивания и дефис)
- value – значение куки
- expire – срок действия куки. Если задан, то кука удаляется по истечению этого строка, в противном случае – после закрытия окна браузера. Указывается в формате timestamp. Timespamp - это число секунд, прошедших с 00:00:00 1 января, 1970 года.
- path – путь, который определяет, в какой части домена может использоваться данный файл (/ - означает доступность в пределах всего домена).
- domain – домен, для которого доступен cookie
- secure - указание, что данные cookie должны передаваться только через безопасное соединение HTTPS.

Cookie – часть заголовка HTTP-запроса, который отправляется браузеру, поэтому их значения должны быть установлены до начала формирования HTML-кода, это означает, что вызов ***setcookie*** должен быть расположен до любых HTML-тегов и до любого оператора echo.

```
setcookie ("TestCookie", $value);
setcookie ("TestCookie", $value,time()+3600);
/* период действия - 1 час */
```

Сразу после установки, значение куки не будет доступно скрипту до тех пор, пока страница не будет перезагружена. Это происходит потому, что куки хранятся на компьютере пользователя и отправляются на веб-сервер браузером при запросе очередной страницы.

Для получения доступа к cookie используется суперглобальный массив $_COOKIE.

```
echo $_COOKIE['TestCookie'];
```

Перед использованием cookie рекомендуется удостовериться в ее существовании с использованием функции ***isset.***

Пример: массивы и cookie:

```php
<?php
//Создаем массив
$array = array("name"=>"John","login"=>"root","pass"=>"p@ssw0rd");
// Упаковываем массив в строку
$str = serialize($array);
//Сохраняем массив в cookie
setcookie('user',$str, time() + 3600);
//Считываем строку и переводим в мас сив
$array = unserialize($_COOKIE['user']);
?>
```

Для удаления куки необходимо вызвать функцию setcookie с пустым параметром value. Поскольку куки удаляется браузером, для их гарантированного удаления рекомендуется задавать истекший период действия куки.

```php
setcookie ("TestCookie","", time()-3600);
```

Работа с сессиями

Когда пользователь в рамках одного приложения работает с различными страницами, при каждой загрузке скрипта переменные очищаются. 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.

Сессии предназначены для сохранения данных определенного пользователя на сервере. Для каждого пользователя формируется уникальный идентификатор сеанса, который используется для автоматического восстановления данных клиента, вне зависимости от количества страниц веб-приложения.

Если есть необходимость запомнить состояние сеанса, все, что нужно сделать – вызвать в начале скрипта функцию ***session_start().*** Данные сеанса содержаться в суперглобальном массиве **$_SESSION.** Например, необходимо сохранить значение выбранного пользователем цвета:

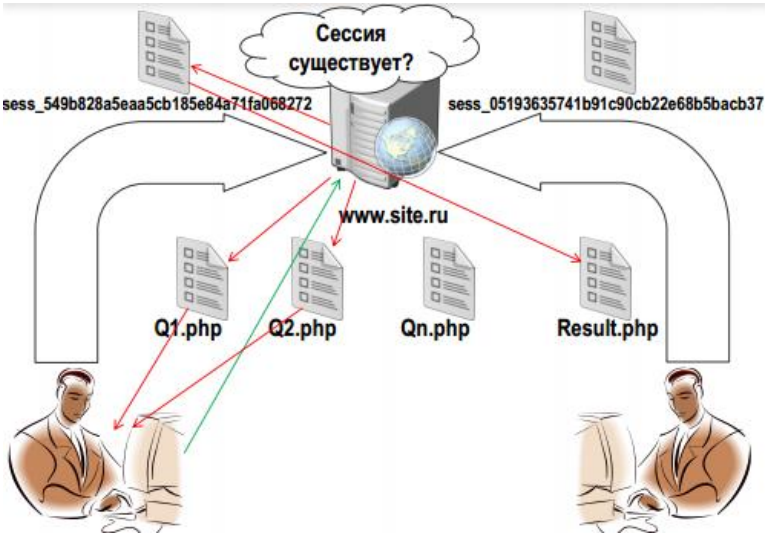
```php
session_start();
$_SESSION['color'] = "blue";
```

Прочитать значение цвета:

```php
session_start();
$color = $_SESSION['color'];
echo $color;
```

Команду ***session_start()*** нужно вызывать во всех скриптах, в которых предстоит использовать переменные *сессии*, причем до вывода каких-либо данных в браузер.

Получить идентификатор сессии (строка в 32 символа) можно с помощью функции ***session_id.*** Идентификатор генерируется, когда пользователь заходит на сайт и уничтожается, когда пользователь уходит с сайта.



Для удаления данных сессий используется функция ***unset*** (для уничтожения заданных переменных) и ***session_destroy*** (удаление текущей сессии целиком).

Пример. Авторизация на сайте.

Сайт 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:

Вход на сайт

Введите имя:

Запоминать меня

Войти

Страница "А"

А и Б сидели на трубе.

Вы вошли как **Дима** | Выход

Страница "Б"

А и **Б** сидели на трубе.

Вы вошли как **Дима** | Выход

Страница login.php

```php
<?php
// Авторизация.
function Login($username, $remember)
{
//  имя не должно быть пустой строкой.
if ($username == '')
    return false;
// Запоминаем имя в сессию
$_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()
{
// Удаляем куки
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();
}
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Вход на сайт</title>
</head>
<body>
<h1>Вход на сайт</h1>
<form action="" method="post">
Введите имя:
<br/>
<input type="text" name="username" />
<br/>
<input type="checkbox" name="remember" /> Запоминать меня
<br/>
<input type="submit" value="Войти" />
</form>
</body>
</html>
```

Страница a.php

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

```php
<?php
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 ($username ==null)
{
    header("Location: login.php");
    exit();
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"><head>
<title>Страница А</title>
</head>
<body>
<h1>Страница "А"</h1>
<b>А</b> и <a href="b.php">Б</a> сидели на трубе.
<br/>
<br/>
Вы вошли как <b><?php echo$username;?></b> | <a
href="login.php">Выход</a>
</body>
</html>
```

Страница b.php почти такая же:

```php
<?php
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 ($username ==null)
{
    header("Location: login.php");
    exit();
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"><head>
<title>Страница Б</title>
</head>
<body>
<h1>Страница "Б"</h1>
<a href="a.php">А</a> и <b>Б</b> сидели на трубе.
<br/>
<br/>
Вы вошли как <b><?php echo$username;?></b> | <a
href="login.php">Выход</a>
</body>
</html>
```

Задания к лабораторной работе

Общие правила выполнения заданий

Требования к ПО:

- Рекомендуется установка в качестве среды разработки openServer.
- В качестве IDE могут быть использованы как проприетарные (phpStorm и др.), так и свободно распространяемые инструменты (Sublime Text, NetBeans, Visual Studio Code и др.).

Задание:

Реализовать механизмы разграничения прав доступа к различным частям веб-приложения. В таблице представлен функционал, доступный пользователю в зависимости от его роли в системе.

Роль	**Функционал**
Гость	Просмотр главной страницы
	Регистрация
	Авторизация
Оператор	Просмотр главной страницы
	Авторизация
	Формирование ведомости
Администратор	Просмотр главной страницы
	Авторизация
	Формирование ведомости
	Администрирование таблиц БД (CRUD)
	Назначение пользователям прав

При входе на главную страницу веб-приложения, пользователь-Гость видит лишь системную информацию о веб-приложении, информацию о разработчике, название проекта и т.д. Также ему доступна форма регистрации.

При регистрации пользователя, все введенные им данные должны быть провалидированы (длина пароля, корректный e-mail, заполненность обязательных полей и т.д.). Также необходимо выполнить проверку на существование в базе данных такого пользователя (с таким email, логином и т.д.). Если такой пользователь существует, должно быть выведено сообщение об ошибке. Если все проверки успешно пройдены, в базу данных добавляется информация о пользователе. Пароль хэшируется.

Для работы в системе все пользователи должны авторизоваться. Также должен необходимо наличие флажка «Запомнить меня», обеспечивающего сохранение данных при следующем сеансе.

Авторизованный администратор (назначается изначально в БД) выполняет модерацию поступивших заявок от пользователей-Гостей (просматривает информацию, может вносить в нее изменения). В случае одобрения заявки, Администратор дает пользователю права Оператора.

Оператор после авторизации получает доступ к интерфейсу формирования ведомости (Лаб2), все остальные страницы для него закрыты-заблокированы (либо не отображаются вообще, либо закрыты с сообщением о недостаточных правах для доступа; предлагается пройти регистрацию).

При первом входе в систему оператору выводится приветственное сообщение «Добро пожаловать!». При повторном входе выводится: количество посещений (например, «Вы зашли в 5 раз» и дату/время последнего посещения (например, «Последнее посещение: 2022-11-10 14:12:41).

Администратор имеет доступ ко всем страницам (Модерация заявок, Формирование ведомости, CRUD).

Необходимо предусмотреть возможность Logout (выхода из системы).

Дополнительное задание №1 (*необязательное*):

- Добавить капчу к форме авторизации в системе ***(+20%).***
- После третьего неправильного ввода капчи вывести соответствующее сообщение или запретить доступ к странице на определенное время ***(+10%).***

Дополнительное задание №2 (*необязательное*):

- Реализовать механизм восстановления пароля через электронную почту (+20%).

Требования и рекомендации по выполнению:

- *Необходимо использование session и cookie по назначению*.
- Информация о пользователях хранится в БД.
- Вид навигации не регламентирован.
- В качестве API-интерфейса для доступа к БД может быть использован модуль PDO или mysqli.
- При написании кода необходимо придерживаться *процедурного* или *объектно-ориентированного стиля.*
- После выполнения работы оформить ее по своему усмотрению (настроить шрифты, отступы и т.д.), используя CSS, Bootstrap и др.

Правила оценивания:

- Лабораторная работа оценивается из 100%. При успешном выполнении дополнительных заданий балл может составлять 150%. В случае пересдачи максимально возможный балл – 60%.
- В случае нарушения дедлайнов лабораторная работа оценивается в 1%.