Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

Отчет по лабораторной работе №5 по дисциплине

ТЕХНОЛОГИИ ИНТЕРНЕТ-ПРОГРАММИРОВАНИЯ
на тему
«*Cookies* и сессии *PHP*»

Выполнил ст. гр. 820601 А.Р. Шведов
Проверил преп. каф. ИТАС А.Л. Гончаревич

Минск 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

PHP — язык программирования, который наиболее распространён в сфере веб-разработки. Язык *PHP* работает на удаленном сервере, поэтому он и называется серверный язык программирования.

Любой скрипт *PHP* состоит из последовательности операторов. Оператор может быть присваиванием, вызовом функции, циклом, условным выражением или пустым выражением. Операторы обычно заканчиваются точкой с запятой. Также операторы могут быть объединены в группу заключением группы операторов в фигурные скобки. Группа операторов также является оператором.

Интернет — это множество компьютеров по всему миру, соединённых между собой проводами в единую сеть. Все компьютеры делятся на две большие группы: клиенты и сервера. Клиенты инициируют запросы на сервера, а те, в свою очередь, их принимают, обрабатывают и отправляют клиенту ответ.

PHP позволяет решить множество задач связанных с клиент-серверной архитектурой, например:

1 С помощью *HTML* можно только создать форму. А обработать то, что ввёл пользователь, может лишь *PHP*.

2 Если делать блог на чистом *HTML*, то на каждую статью требуется создавать новый файл. Добавлять и редактировать записи придётся вручную. PHP позволяет обойтись с помощью одного файла, а статьи хранить в базе данных. Благодаря этому, можно сделать админку, из которой можно будет добавлять и редактировать контент.

3 *PHP* позволяет реализовать механизм авторизации на сайте.

1 ПОСТАНОВКА ЗАДАЧИ

Изучить семантику, синтаксис и возможности языка *PHP*. Изучение базового синтаксиса в языке *PHP*, работу с переменными, разными типами данных, операциями сравнения и логическими операциями. Ознакомление с основами серверных технологий.

2 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Cookies — это небольшой кусок текстовой информации, которую сервер сохраняет в браузере пользователя. Можно установить *cookies* при помощи функций *setcookie()* или *setrawcookie()*. Cookies являются частью *HTTP*-заголовка, *поэтому setcookie()* должна вызываться до любого вывода данных в браузер. Это то же самое ограничение, которое имеет функция *header()*. Можно использовать функции буферизации вывода, чтобы задержать вывод результатов работы скрипта до того момента, когда будет известно, понадобится ли установка *cookies* или других заголовков.

Любые cookies, отправленные серверу браузером клиента, будут автоматически включены в суперглобальный массив *$_COOKIE*, если директива *variables_order* содержит букву "C".

Сессии являются простым способом хранения информации для отдельных пользователей с уникальным идентификатором сессии. Это может использоваться для сохранения состояния между запросами страниц. Идентификаторы сессий обычно отправляются браузеру через сессионный *cookie* и используются для получения имеющихся данных сессии. Отсутствие идентификатора сессии или сессионного cookie сообщает *PHP* о том, что необходимо создать новую сессию и сгенерировать новый идентификатор сессии.

Сессии используют простую технологию. Когда сессия создана, *PHP* будет либо получать существующую сессию, используя переданный идентификатор (обычно из сессионного cookie) или, если ничего не передавалось, будет создана новая сессия. *PHP* заполнит суперглобальную переменную *$_SESSION* сессионной информацией после того, как будет запущена сессия. Когда *PHP* завершает работу, он автоматически сериализует содержимое суперглобальной переменной *$_SESSION* и отправит для сохранения, используя сессионный обработчик для записи сессии.

Сессии могут запускаться вручную с помощью функции *session_start()*. Если директива *session.auto_start* установлена в «1», сессия автоматически запустится, в начале запроса.

Сессия обычно завершает свою работу, когда *PHP* заканчивает исполнять скрипт, но может быть завершена и вручную с помощью функции *session_write_close()*.

3 ХОД РАБОТЫ

Работа выполняется с помощью редактора кода – *Visual Studio Code*. Работа с программой начинается с создания *PHP* файла, в который будут помещаться скрипты.

3.1 Задание 1

В задании 1 необходимо создать 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.

Приведем структуру проекта на рисунке 3.1.

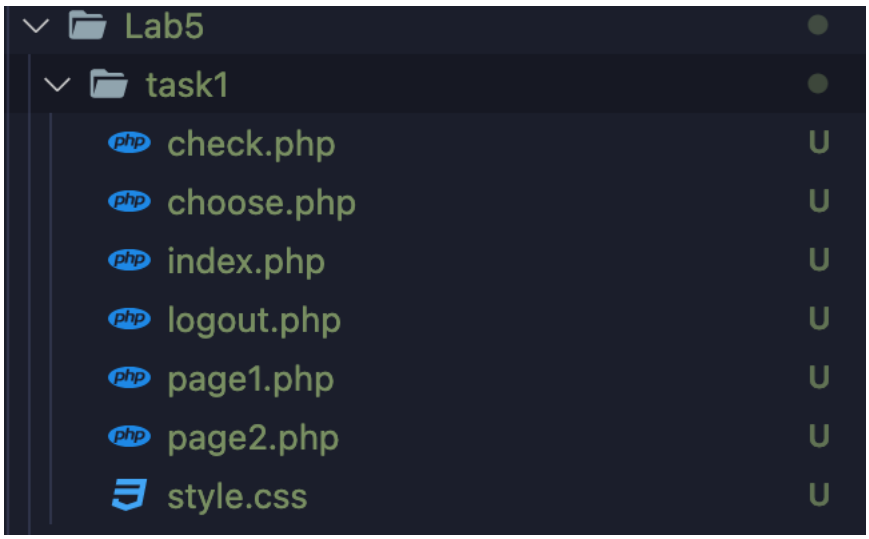

Рисунок 3.1 — Структура файлов

На странице *index.php* пользователь должен иметь возможность авторизоваться, нажать кнопку «Запомнить меня», чтобы его браузер получил куки для дальнейшего использования на сайте. Также добавим валидацию ввода, чтобы пользователь не мог ввести пустые данные. Приведем пример работы на рисунке 3.2.

Рисунок 3.2 — Страница *index.php*

Приведем фрагмент кода, отвечающий за обработку запроса пользователя и введенной формы:

```
session_start([
    'coockie_lifetime' => time() + 360,
]);

if (isset($_COOKIE['last_visited'])) {
    $last_visit = $_COOKIE['last_visited'];
}

// Check if the user is logged in already or not.
if (!empty($_SESSION['logged_in'])) {
    // redirect to last visited, if it was set
    empty($last_visit) ? header("Location: choose.php") : header("Location: $last_visit");
}

// Set coockies
if (!empty($_POST['username']) && !empty($_POST['password'])) {
    if (!empty($_POST["remember"])) {
        setcookie("username", $_POST["username"]);
        setcookie("password", $_POST["password"]);
    } else {
```

```
            setcookie("username", "");
            setcookie("password", "");
        }
        $_SESSION['logged_in'] = True;
        empty($last_visit) ? header("Location: choose.php") : header("Location: $last_visit");
    }
```

Как видно из приведенного выше, если пользователь ввел имя и пароль, он будет перенаправлен на страницу *choose.php* или на предыдущую страницу, где он был, если в его браузере присутствует соответствующая *Coockie*. Так же уже авторизованный пользователь при переходе на *index.php* будет ту же перенаправлен. Фрагмент кода:

```
    <?php
        require_once __DIR__ . '/check.php';
        page_check();?>
    <!DOCTYPE html>
    <html lang="en">
    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>Lab 5 task 1</title>
        <link rel="stylesheet" href="style.css" />
    </head>
    <h1>Choose page</h1>
    <body>
        <ul>
            <li><a href="page1.php">Page 1</a></li>
            <li><a href="page2.php">Page 2</a></li>
        </ul>
    </body>
    <footer>
        <a href="logout.php">Logout</a>
    </footer>
    </html>
```

Приведем страницу *choose.php* на рисунке 3.3.

Рисунок 3.3 — Страница *choose.php*

Заметим, что добавлена кнопка выхода и проверка сессии (через загружаемую из файла *check.php* функцию. Приведем фрагмент ее кода:

```
function page_check()
{
    session_start([
        'cookie_lifetime' => time() + 360,
    ]);

    // Check if the user is logged in already or not.
    if (empty($_SESSION['logged_in'])) {
        header("Location: index.php");
    }else{
        setcookie("last_visited", $_SERVER['REQUEST_URI']);
    }
}
```

Такой подход позволяет обновить сохранить *last_visited* куки и одновременно является защитным механизмом: пользователь вне сессии будет перенаправлен на *index.php* для авторизации. Данная функция вызывается на всех страницах веб-приложения, кроме *index.php*.

На рисунках 3.4 и 3.5 приведем страницы «А» и «Б», на которые пользователь может перейти.

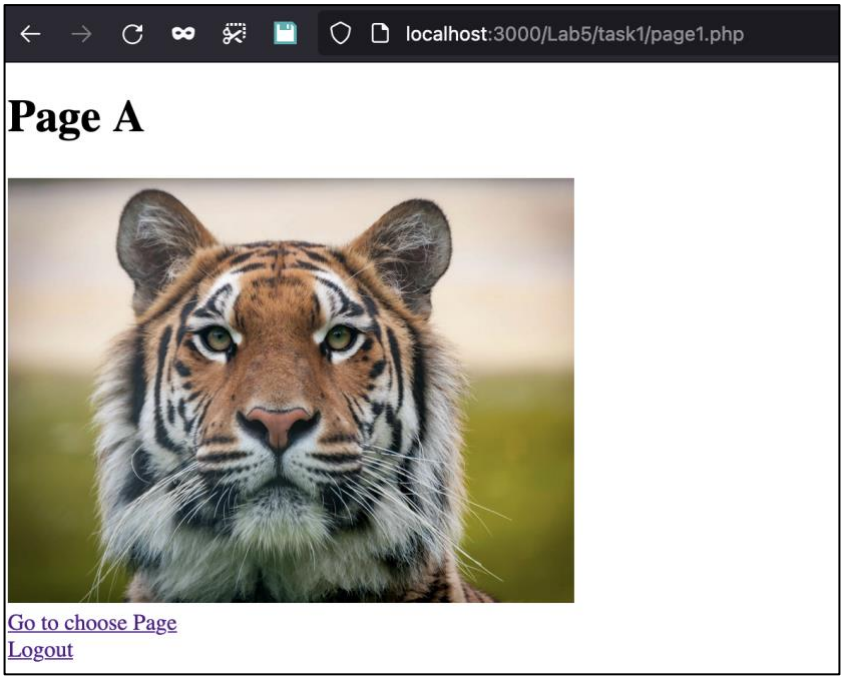

Рисунок 3.4 — Страница А

Рисунок 3.5 — Страница Б

После нажатия кнопки «*Logout*» пользователь перенаправляется на файл *logout.php*, где закрывается и очищается сессия, затем пользователь перенаправляется на страницу входа *index.php*. Фрагмент кода:

```
<?php
session_start();
session_destroy();
header("Location: index.php");
?>
```

3.2 Задание 2

В задании 2 необходимо создать три *css* файла с разными стилями. Затем 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.

Приведем структуру проекта на рисунке 3.6.

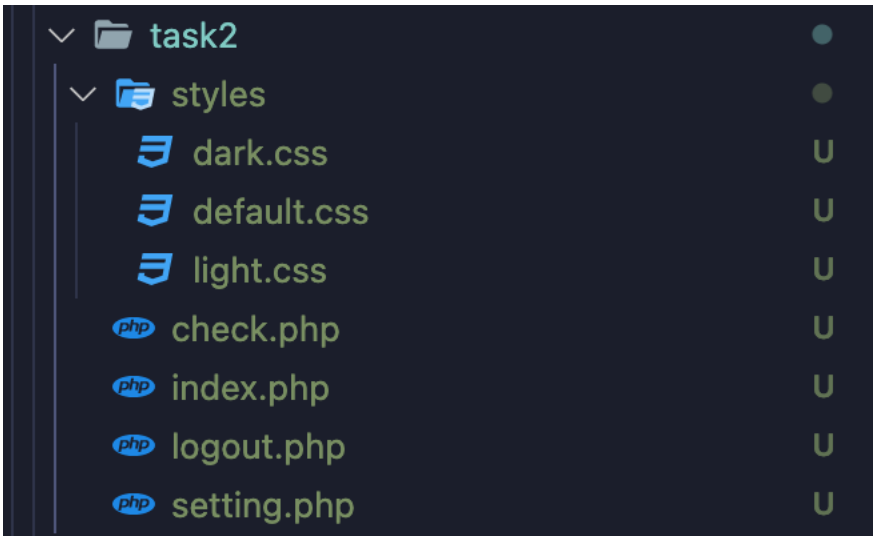

Рисунок 3.6 — Структура файлов

Переиспользуем страницу *index.php* и авторизацию пользователя в сессии из предыдущего задания. После авторизации пользователь перенаправляется на страницу *setting.php*. Фрагмент кода:

```
<?php
require_once __DIR__ . '/check.php';
```

```
page_check();

// set style coockie
setcookie("style", "default.css");
isset($_POST["style"]) ? $_COOKIE["style"] = $_POST["style"] . ".css" : $_COOKIE["style"] = "default.css";
?>
<h1>Style settings</h1>
<form action="<?php echo htmlspecialchars($_SERVER['PHP_SELF']) ?>" method="post">
    <div>
        <label for="style">Style</label>
        <select name="style" id="style">
            <option value=>--- Choose a style ---</option>
            <option value="light" <?php echo $_COOKIE["style"] == "light.css" ? "selected" : ""; ?>>Light</option>
            <option value="dark" <?php echo $_COOKIE["style"] == "dark.css" ? "selected" : ""; ?>>Dark</option>
            <option value="default" <?php echo $_COOKIE["style"] == "default.css" ? "selected" : ""; ?>>Default</option>
        </select>
    </div>
    <div>
        <button type="submit">Select</button>
    </div>
</form>
```

Приведем страницу на рисунке 3.7.

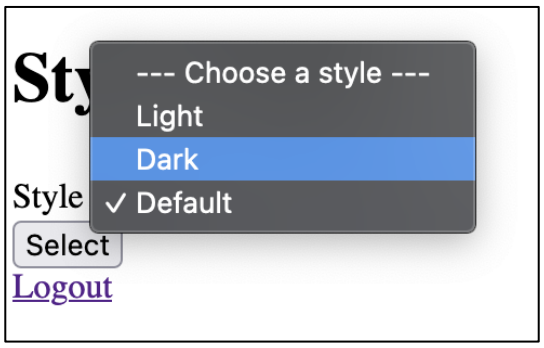

Рисунок 3.7 — Страница *setting.php*

Заметим, как работает выбор стиля и сохранение формы. После того, как пользователь выбирает стиль и нажимает кнопку «*Select*», страница перенаправляется сама на себя. Срабатывает *php* скрипт, который из *HTTP POST* запроса ставит куки с параметром стиля (это сделано тернарным оператором).

Далее стиль страницы определяется строкой *<link rel="stylesheet" href="<?php echo "styles/{$_COOKIE["style"]}"; ?>" />*. Здесь выбирается файл *.css* из массива *$_COOKIE*, в котором всегда гарантировано хотя бы значение *default.css*.

При выборе стиля так же сохраняется выбор пользователя, это сделано через тернарный оператор и *HTML* флаг «*selected*». Пример страницы, после выбора стиля приведен на рисунке 3.8.

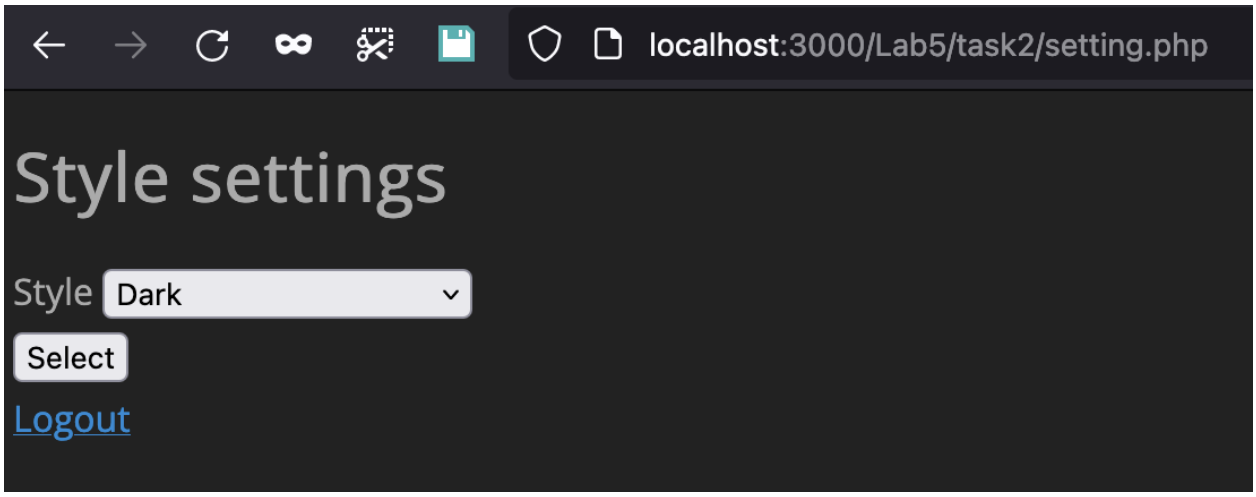

Рисунок 3.8 — Страница *setting.php*

ЗАКЛЮЧЕНИЕ

HTTP — лёгкий в использовании расширяемый протокол. Структура клиент-сервера, вместе со способностью к простому добавлению заголовков, позволяет *HTTP* протоколу продвигаться вместе с расширяющимися возможностями Сети.

PHP позволяет создавать качественные *web*-приложения за очень короткие сроки, получая продукты, легко модифицируемые и поддерживаемые в будущем.

PHP прост для освоения, и вместе с тем способен удовлетворить запросы профессиональных программистов.

Язык *PHP* постоянно совершенствуется, и ему наверняка обеспечено долгое доминирование в области языков *web*-программирования, по крайней мере, в ближайшее время.

В ходе выполнения лабораторной работы я изучил использование *Cookies* и сессий *PHP*, разработал примеры приложений с авторизацией пользователей и сохранением их настроек в их *cookies*.