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 №4 по дисциплине

ТЕХНОЛОГИИ ИНТЕРНЕТ-ПРОГРАММИРОВАНИЯ
на тему
«Запросы *HTTP*, параметры *URL* и формы *HTML*»

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 Вы делаете 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

HTTP — это протокол, позволяющий получать различные ресурсы, например *HTML*-документы. Протокол *HTTP* лежит в основе обмена данными в Интернете. *HTTP* является протоколом клиент-серверного взаимодействия, что означает инициирование запросов к серверу самим получателем, обычно веб-браузером (*web-browser*). Полученный итоговый документ может состоять из различных поддокументов, являющихся частью итогового документа: например, из отдельно полученного текста, описания структуры документа, изображений, видео-файлов, скриптов и многого другого.

Запрос *GET* передает данные в *URL* в виде пар «имя-значение» (другими словами, через ссылку), а запрос *POST* передает данные в теле запроса. Это различие определяет свойства методов и ситуации, подходящие для использования того или иного *HTTP* метода.

Страница, созданная методом *GET*, может быть открыта повторно множество раз. Такая страница может быть кэширована браузерами, проиндексирована поисковыми системами и добавлена в закладки пользователем. Из этого следует, что метод *GET* следует использовать для получения данных от сервера и не желательно в запросах, предполагающих внесений изменений в ресурс.

Запрос, выполненный методом *POST*, напротив следует использовать в случаях, когда нужно вносить изменение в ресурс (выполнить авторизацию, отправить форму оформления заказа, форму обратной связи, форму онлайн заявки). Повторный переход по конечной ссылке не вызовет повторную обработку запроса, так как не будет содержать переданных ранее параметров. Метод *POST* имеет большую степень защиты данных, чем *GET*: параметры запроса не видны пользователю без использования специального ПО, что дает методу преимущество при пересылке конфиденциальных данных, например в формах авторизации.

В *PHP* параметры, переданные через *URL*, хранятся в системной переменной $_*GET*. Она представляет собой ассоциативный массив (или, по-другому, словарь). Ассоциативный массив (или словарь) – это такая структура данных, которая содержит пары ключ-значение. Ключи словаря $_*GET* – это имена параметров. По аналогии с методом *GET*, параметры, переданные методом *POST*, содержатся в системной переменной $_*POST*. Это также, как и $_*GET*, ассоциативный массив. Ключами в массиве $_*POST* являются значения атрибутов *name* у элементов на форме.

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.

В задании 1 для начала необходимо создать папку с изображениями и две страницы *index.php* и *photo.php*. На странице *photo.php* необходимо отобразить определённую картинку. Чтобы понять, какую именно картинку показывать, нужно считать определённый параметр – номер изображения – из *URL*-адреса. Страница *photo.php* показана на рисунке 3.1.

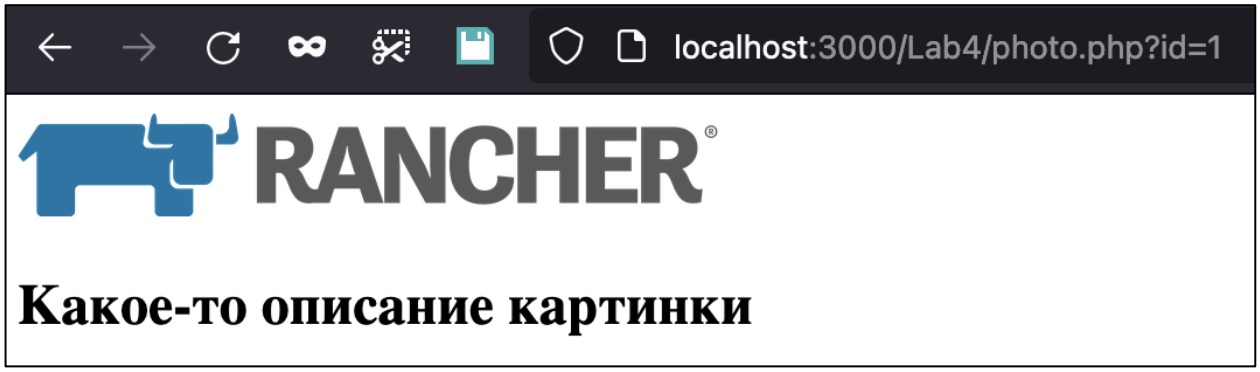

Рисунок 3.1 — Страница *photo.php*

На странице *index.php* находятся ссылки на все изображения. Страница *index.php* показана на рисунке 3.2.

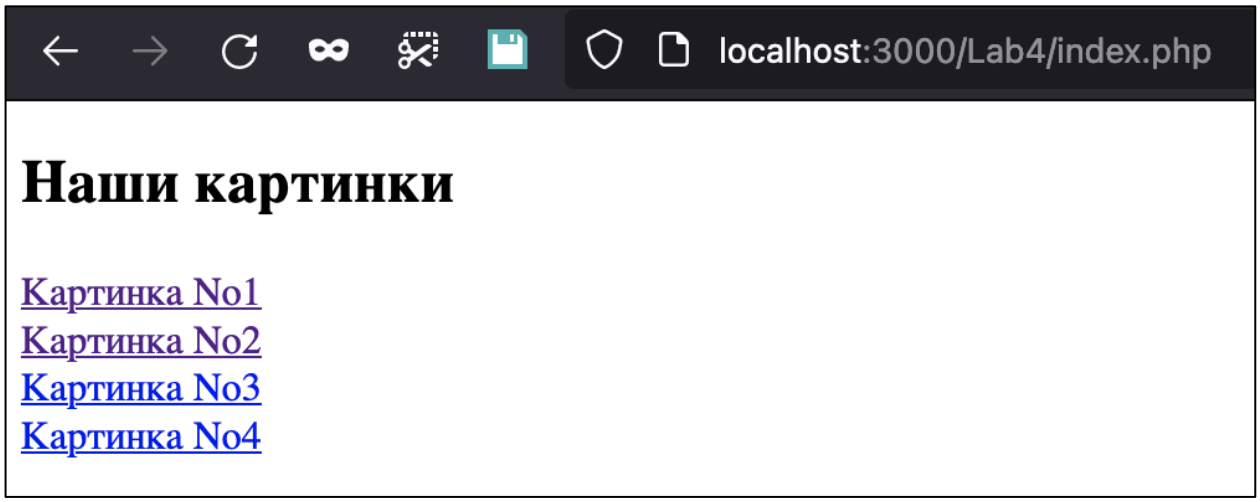

Рисунок 3.2 — Страница *index.php*

На странице *photo.php* запросы отправлялись *GET* методом. На следующей странице *sum.php* используется *POST* запрос. На этой странице реализован скрипт складывающий числа. Страница *sum.php* показана на рисунке 3.3.

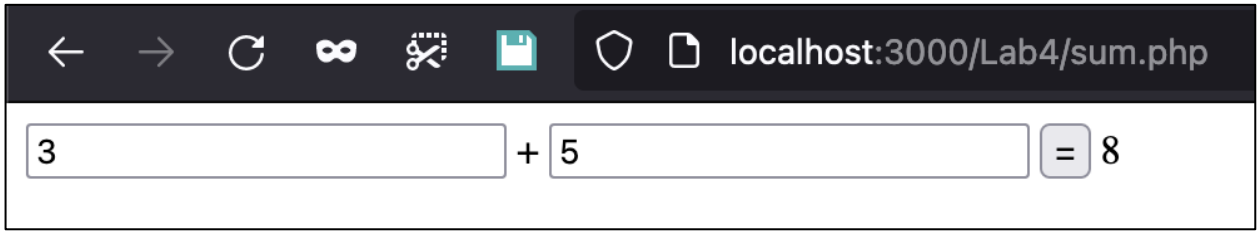

Рисунок 3.3 — Страница *sum.php*

В задании 2 необходимо превратить получившийся сумматор из предыдущего задания в калькулятор с четырьмя операциями: сложение, вычитание, умножение, деление. Результат выполнения задания и варианты работы продемонстрированы на рисунках 3.4 и 3.5. Фрагмент кода:

```
if (isset($_POST['a']) && isset($_POST['b'])) {
    $operator = $_POST['operator'];
    switch ($operator) {
        case "Add":
            $result = $_POST['a'] + $_POST['b'];
            break;
        case "Subtract":
            $result = $_POST['a'] - $_POST['b'];
            break;
        case "Multiply":
            $result = $_POST['a'] * $_POST['b'];
            break;
        case "Divide":
            if ($_POST['b'] == 0) {
                $result = "you can't do this here!";
                break;
            } else {
                $result = $_POST['a'] / $_POST['b'];
                break;
            }
```

```
        default:
            $result = "";
            break;
    }
} else {
    $result = "";
}
```

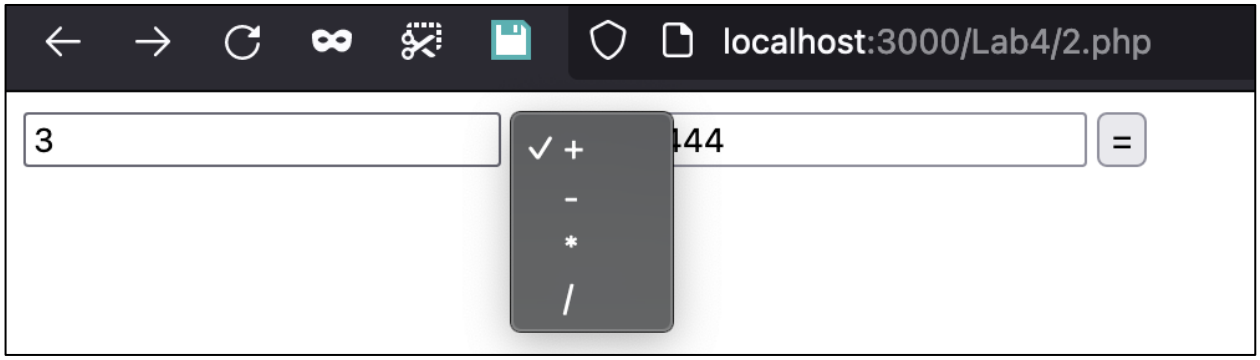

Рисунок 3.4 — Выбор операции

Рисунок 3.5 — Деление на ноль

В задании 3 необходимо создать калькулятор, который будет определять тип выбранной пользователем операции, ориентируясь на нажатую кнопку. Данные, введённые пользователем в поля, должны сохраняться и выводиться вместе с результатом вычисления. Результат выполнения задания продемонстрирован на рисунке 3.6. Фрагмент кода:

```
<body>
    <form method="post">
        a: <input name="a" type="number" value="<?php echo isset($_POST['a']) ? htmlspecialchars($_POST['a'], ENT_QUOTES) : ''; ?>">
        <br>
        b: <input name="b" type="number" value="<?php echo isset($_POST['b']) ? htmlspecialchars($_POST['b'], ENT_QUOTES) : ''; ?>">
```

```
        <p>
            <input type="submit" name="operator" value="+" />
            <input type="submit" name="operator" value="-" />
            <input type="submit" name="operator" value="*" />
            <input type="submit" name="operator" value="/" />
        </p>

        Result: <input type='text' value="<?php echo $result; ?>"><br>
    </form>
</body>
```

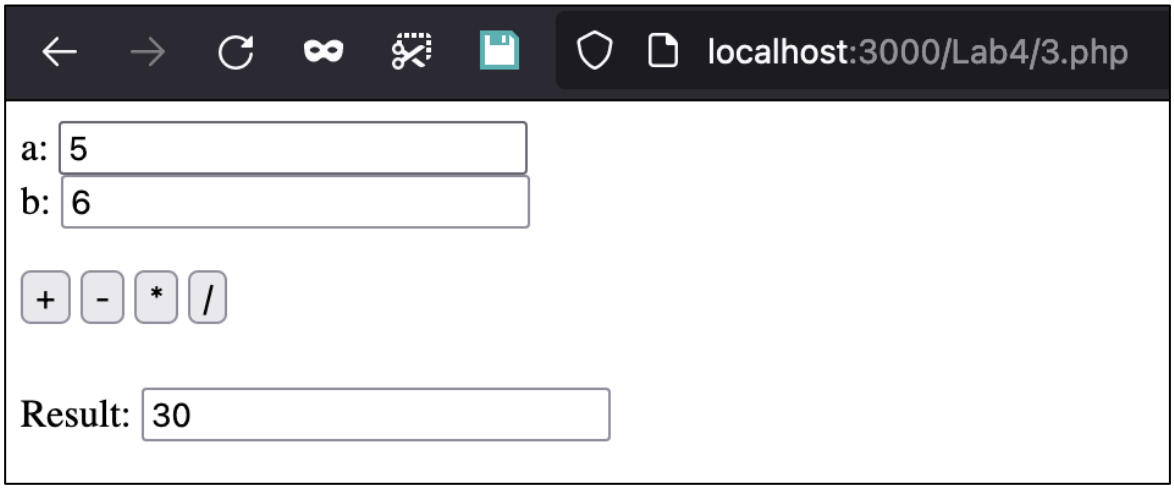

Рисунок 3.6 — Задание 3

Заметим, что функция *htmlspecialchars()* с параметром *ENT_QUOTES* нужна для корректного отображения данных в форме. Эта функция возвращает строку, над которой проведены эти преобразования, в данном случае преобразует как двойные, так и одинарные кавычки. Ее использование не является обязательным для этого задания, однако это хорошая практика оставлять валидацию данных.

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.

*Я*зык *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 *GET* и *POST* методы протокола *HTTP*, изучил как передавать параметры через *URL* и *HTML* форму.