

НАЗВАНИЕ УНИВЕРСИТЕТА

Факультет «Факультет инфокоммуникационных систем и технологий»
Направление подготовки «Проектирование инфокоммуникационных систем»

Лабораторная работа №3

Выполнил:
Золотых Лев Константинович
Группа №К3321
Проверила:
Марченко Елена Вадимовна

Санкт-Петербург
2024

Цель работы: Освоить работу с gulp, создать форму для отправки информации по обратной связи, установить инструментарий для отладки проектов и настроить портал test.site.

Ход работы:

Задание 1. Пункт а

В данном пункте требовалось создать два таска и настроить их на последовательное и параллельное выполнение.

Для этого в файле gulpfile.js подключаем библиотеку gulp, прописываем сами таски. Для наглядности последовательного и параллельного выполнения сделаем искусственную задержку. Так мы увидим наверняка, когда задачи выполняются последовательно, а когда параллельно. Полный скрипт приведён ниже

```
WebDevelopment_2024-2025 > work > K3321 > Zolotych Lev Konstantinovich > lab3 > gulpfile.js > ...
1  const gulp = require("gulp");
2
3  function taskOne(cb) {
4    console.log("Задача 1 стартовала");
5    setTimeout(() => {
6      console.log("Задача 1 завершена");
7      cb();
8    }, 2000);
9  }
10
11 function taskTwo(cb) {
12   console.log("Задача 2 стартовала");
13   setTimeout(() => {
14     console.log("Задача 2 завершена");
15     cb();
16   }, 1000);
17 }
18
19 exports.sequential = gulp.series(taskOne, taskTwo);
20
21 exports.parallel = gulp.parallel(taskOne, taskTwo);
22
```

Рисунок 1 – код тасок

Для запуска необходимо в консоли прописать `gulp sequential` для последовательного выполнения или `gulp parallel` для параллельного

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Lev4ik\Desktop\Web2024\WebDevelopment_2024-2025\work\K3321\Zolotykh Lev Konstantinovich\lab3> gulp sequential
>>
[04:54:37] Using gulpfile ~\Desktop\Web2024\WebDevelopment_2024-2025\work\K3321\Zolotykh Lev Konstantinovich\lab3\gulpfile.js
[04:54:37] Starting 'sequential'...
[04:54:37] Starting 'taskOne'...
Задача 1 стартовала
Задача 1 завершена
[04:54:39] Finished 'taskOne' after 2.01 s
[04:54:39] Starting 'taskTwo'...
Задача 2 стартовала
Задача 2 завершена
[04:54:40] Finished 'taskTwo' after 1.01 s
[04:54:40] Finished 'sequential' after 3.02 s
PS C:\Users\Lev4ik\Desktop\Web2024\WebDevelopment_2024-2025\work\K3321\Zolotykh Lev Konstantinovich\lab3> |
```

Рисунок 2 – результат для gulp sequential

```
PS C:\Users\Lev4ik\Desktop\Web2024\WebDevelopment_2024-2025\work\K3321\Zolotykh Lev Konstantinovich\lab3> gulp parallel
[04:55:23] Using gulpfile ~\Desktop\Web2024\WebDevelopment_2024-2025\work\K3321\Zolotykh Lev Konstantinovich\lab3\gulpfile.js
[04:55:23] Starting 'parallel'...
[04:55:23] Starting 'taskOne'...
[04:55:23] Starting 'taskTwo'...
Задача 1 стартовала
Задача 2 стартовала
Задача 2 завершена
[04:55:24] Finished 'taskTwo' after 1.01 s
Задача 1 завершена
[04:55:25] Finished 'taskOne' after 2.01 s
[04:55:25] Finished 'parallel' after 2.01 s
PS C:\Users\Lev4ik\Desktop\Web2024\WebDevelopment_2024-2025\work\K3321\Zolotykh Lev Konstantinovich\lab3> |
```

Рисунок 3 – результат для gulp parallel

Как видно из рисунков, в случае последовательного выполнения сначала начинается и заканчивается первая задача, но при параллельном одновременно начинаются все две задачи, но вторая заканчивается быстрее из-за меньшей задержки.

Задание 1. Пункт б

В данном пункте задания требовалось настроить отображение файлов проекта в браузере и его перезагрузки при изменении файлов.

Для этого создаётся экземпляр browserSync, а в gulpfile была добавлена ещё одна задача:

В качестве директории, обслуживаемой сервером, указывается нынешняя.

Функция watchFiles говорит просматривать все файлы директории с указанными расширениями и вызывать перезагрузку сервера при изменении любого из них. Конечный скрипт представлен ниже:

```

23 const browserSync = require("browser-sync").create();
24 function serve(cb) {
25   browserSync.init({
26     server: {
27       baseDir: "./",
28     },
29   });
30   cb();
31 }
32
33 function watchFiles(cb) {
34   gulp.watch("*.html").on("change", browserSync.reload);
35   gulp.watch("css/*.css").on("change", browserSync.reload);
36   gulp.watch("js/*.js").on("change", browserSync.reload);
37   cb();
38 }
39
40 exports.serve = gulp.series(serve, watchFiles);
41

```

Рисунок 4 – скрипт для “живого сервера”

Задание 2

В данном задании требовалось создать форму для отправки обратной связи с радиокнопками и флажками. Поскольку для получения записанных данных нужен только метод POST, было принято решение записывать данные файл, а затем при желании, демонстрировать их. Для этого понадобится метод GET

В файле `save_data.php` мы проверяем, что запрос к скрипту выполнен методом POST, после чего получаем данные из одноимённого массива, записывая в переменные. Далее генерируется уникальный `id`, создаётся переменная, содержащая путь до файла, после чего данные сохраняются по заданному пути в нужном формате. В конце в новом окне выводится информация о том, что данные были успешно отправлены.

```

WebDevelopment_2024-2025 > work > K3321 > Zolotykh Lev Konstantinovich > lab3 > get_data.php > ...
1  <?php
2  if ($_SERVER["REQUEST_METHOD"] === "GET" && isset($_GET['id'])) {
3      // Получаем ID из GET-запроса
4      $id = htmlspecialchars(string: $_GET['id']);
5      $filePath = __DIR__ . "/data/{$id}.json";
6
7      if (file_exists(filename: $filePath)) {
8          // Читаем данные из файла
9          $data = json_decode(json: file_get_contents(filename: $filePath), associative: true);
10
11         // Выводим данные
12         echo "<!DOCTYPE html>";
13         echo "<html lang='ru'>";
14         echo "<head><meta charset='UTF-8'><title>Полученные данные</title></head>";
15         echo "<body>";
16         echo "<h1>Данные с идентификатором: {$id}</h1>";
17         echo "<p><strong>Имя:</strong> " . htmlspecialchars(string: $data['first_name']) . "</p>";
18         echo "<p><strong>Фамилия:</strong> " . htmlspecialchars(string: $data['last_name']) . "</p>";
19         echo "<p><strong>Email:</strong> " . htmlspecialchars(string: $data['email']) . "</p>";
20         echo "<p><strong>Сообщение:</strong> " . htmlspecialchars(string: $data['feedback']) . "</p>";
21         echo "<p><strong>Выбранный вариант:</strong> " . htmlspecialchars(string: $data['option']) . "</p>";
22         echo "<p><strong>Выбранные чекбоксы:</strong></p><ul>";
23         foreach ($data['checkboxes'] as $checkbox) {
24             echo "<li> " . htmlspecialchars(string: $checkbox) . "</li>";
25         }
26         echo "</ul>";
27         echo "<p><strong>Время отправки:</strong> " . htmlspecialchars(string: $data['timestamp']) . "</p>";
28         echo "</body>";
29         echo "</html>";
30     } else {
31         echo "<h1>Ошибка: данные с ID {$id} не найдены</h1>";
32     }
33 } else {
34     echo "<h1>Ошибка: запрос должен содержать параметр ID</h1>";
35 }
36 ?>

```

Рисунок 5 – save_data.php

В файле get_data.php Проверяем, что запрос выполнен через GET и передан id. Далее id записывается в переменную и ищется файл с соответствующим именем. Содержимое файла считывается, декодируется в JSON массив, после чего выводится на страницу.

```

WebDevelopment_2024-2025 > work > K3321 > Zolotykh Lev Konstantinovich > lab3 > save_data.php > ...
1  <?php
2  if ($_SERVER["REQUEST_METHOD"] === "POST") {
3      // Получаем данные из формы
4      $firstName = htmlspecialchars(string: $_POST['first_name'] ?? '');
5      $lastName = htmlspecialchars(string: $_POST['last_name'] ?? '');
6      $email = htmlspecialchars(string: $_POST['email'] ?? '');
7      $feedback = htmlspecialchars(string: $_POST['feedback'] ?? '');
8      $option = htmlspecialchars(string: $_POST['option'] ?? '');
9      $checkboxes = $_POST['checkboxes'] ?? [];
10
11     // Генерируем уникальный ID
12     $id = uniqid();
13
14     // Формируем данные для записи
15     $data = [
16         "id" => $id,
17         "first_name" => $firstName,
18         "last_name" => $lastName,
19         "email" => $email,
20         "feedback" => $feedback,
21         "option" => $option,
22         "checkboxes" => $checkboxes,
23         "timestamp" => date(format: "Y-m-d H:i:s")
24     ];
25
26     // Сохраняем данные в файл
27     $filePath = __DIR__ . "/data/{$id}.json";
28     file_put_contents(filename: $filePath, data: json_encode(value: $data, flags: JSON_PRETTY_PRINT | JSON_UNESCAPED_UNICODE));
29
30     // Выводим страницу с ID и кнопкой получения данных
31     echo "<!DOCTYPE html>";
32     echo "<html lang='ru'>";
33     echo "<head><meta charset='UTF-8'><title>Результат</title></head>";
34     echo "<body>";
35     echo "<h1>Данные успешно сохранены!</h1>";
36     echo "<p>Идентификатор записи: <strong>{$id}</strong></p>";
37     echo "<form action='get_data.php' method='get' target='_self'>";
38     echo "    <input type='hidden' name='id' value='{$id}'>";
39     echo "    <button type='submit'>Получить данные</button>";
40     echo "</form>";
41     echo "</body>";
42     echo "</html>";
43 } else {
44     echo "<h1>Ошибка: запрос должен быть методом POST</h1>";
45 }
46 >>

```

Рисунок 6 – get_data.php

Таким образом, метод GET используется для того, чтобы передать данные с формы на сервер, в то время как GET используется для запроса этих самых данных с сервера.

Задание 3

Для выполнения данного задания требовалось установить инструментарий (в моём случае XAMPP) и движок wordpress с одноимённых официальных сайтов

Архив wordpress был распакован внутри XAMPP C:/xampp/htdocs/test.site

На сайте <http://localhost/phpmyadmin> была создана база данных wordpress с кодировкой utf8_general_ci.

В файле httpd-vhosts.conf была добавлена следующая конфигурация:

```

C: > xampp > apache > conf > extra > httpd-vhosts.conf
36  ##<VirtualHost *:80>
39      ##ServerName dummy-host2.example.com
40      ##ErrorLog "logs/dummy-host2.example.com-error.log"
41      ##CustomLog "logs/dummy-host2.example.com-access.log" common
42  ##</VirtualHost>
43
44  <VirtualHost *:80>
45      DocumentRoot "C:/xampp/htdocs/test.site"
46      ServerName test.site
47      <Directory "C:/xampp/htdocs/test.site">
48          AllowOverride All
49          Require all granted
50      </Directory>
51  </VirtualHost>
52

```

Рисунок 7 – конфигурация файла

В файле C:\Windows\System32\drivers\etc\hosts добавлена строка:

127.0.0.1 test.site

Далее перейдя по адресу test.site начал установку, указал имя бд и создал учётную запись. После её создания, входа и настройки темы при переходе на сайт отображается следующая информация:

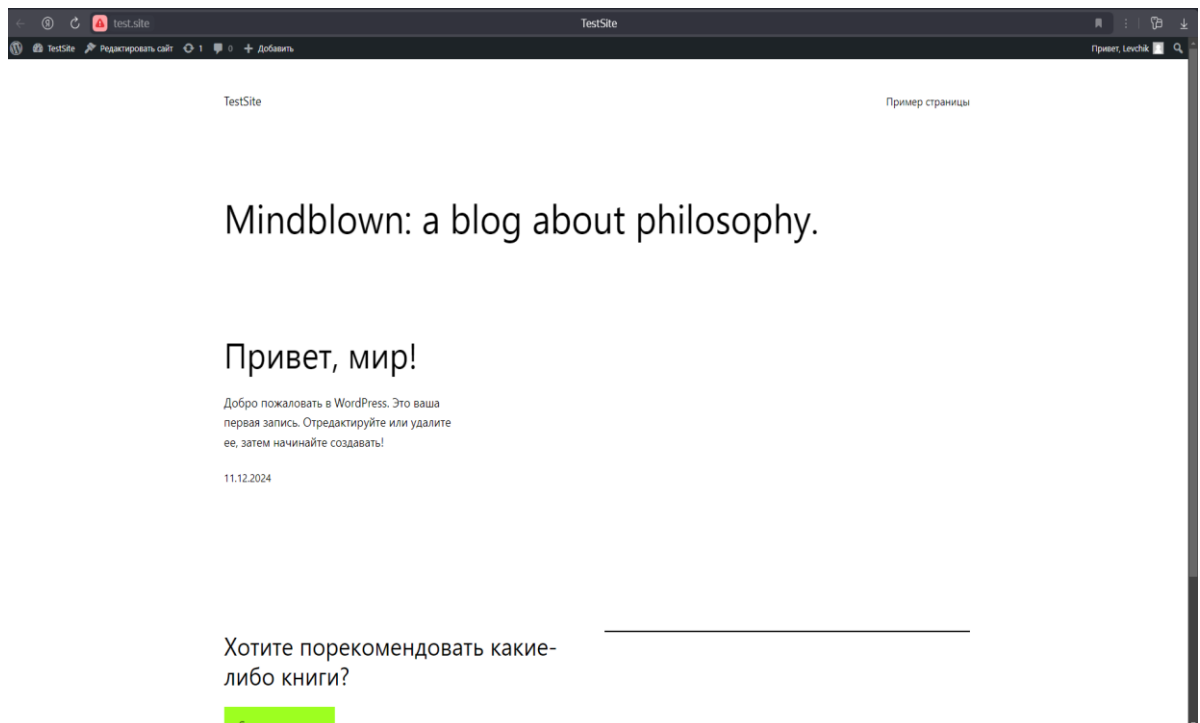


Рисунок 8 – внешний вид страницы test.site

Вывод: в данной лабораторной работе были приобретены навыки работы с gulp, путём создания заданий с их последовательным или же параллельным

выполнением. Также был создан живой сервер с помощью gulp и browser-sync. Была написана форма отправки обратной связи с использованием HTML, CSS, PHP, освоена работа с методами GET, POST. Установлен инструментальный для локальной разработки сайтов, настроен хост и установлена система управления контентом wordpress.