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 > Zolotykh 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

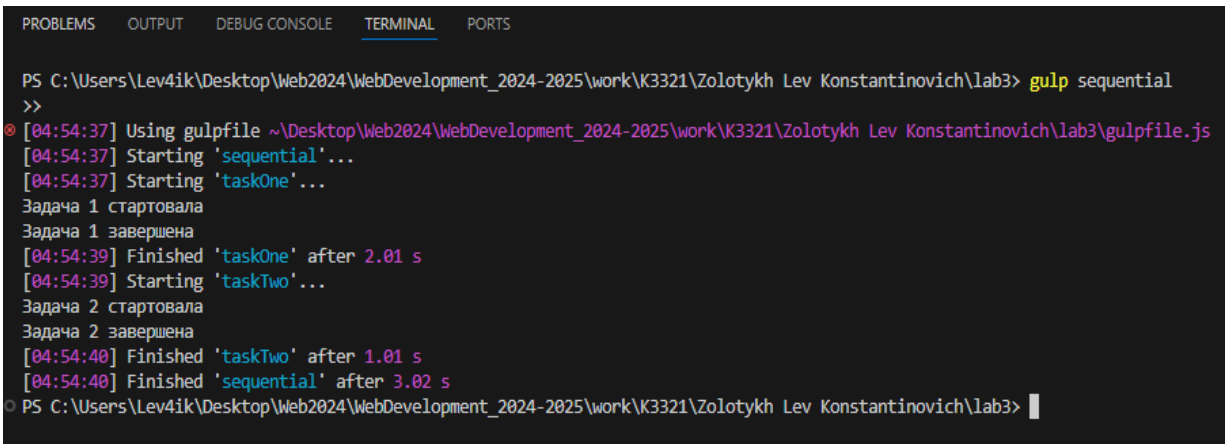

Рисунок 2 – результат для gulp sequential

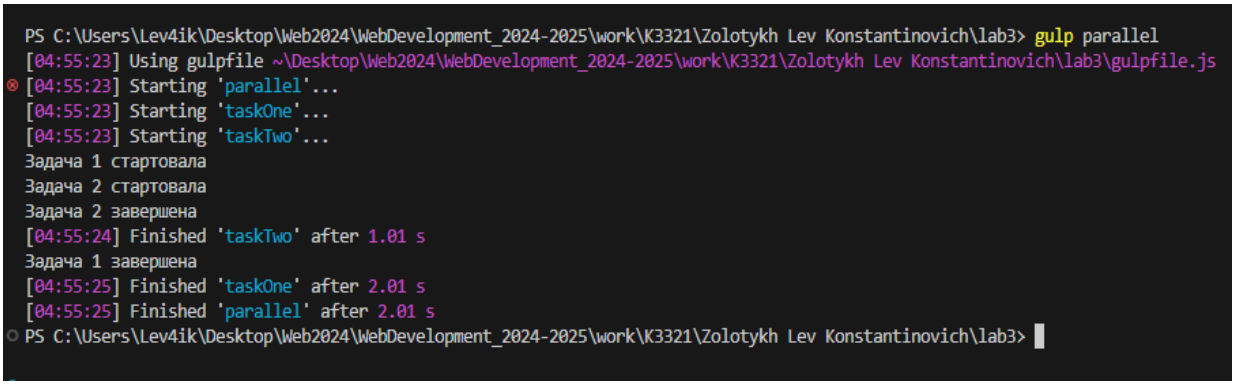

Рисунок 3 – результат для gulp parallel

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 экземлпяр browserSync, а в gulpfile была добавлена ещё одна таска:

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:

```
const browserSync = require("browser-sync").create();
function serve(cb) {
  browserSync.init({
    server: {
      baseDir: "./",
    },
  });
  cb();
}

function watchFiles(cb) {
  gulp.watch("*.html").on("change", browserSync.reload);
  gulp.watch("css/*.css").on("change", browserSync.reload);
  gulp.watch("js/*.js").on("change", browserSync.reload);
  cb();
}

exports.serve = gulp.series(serve, watchFiles);

```

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.

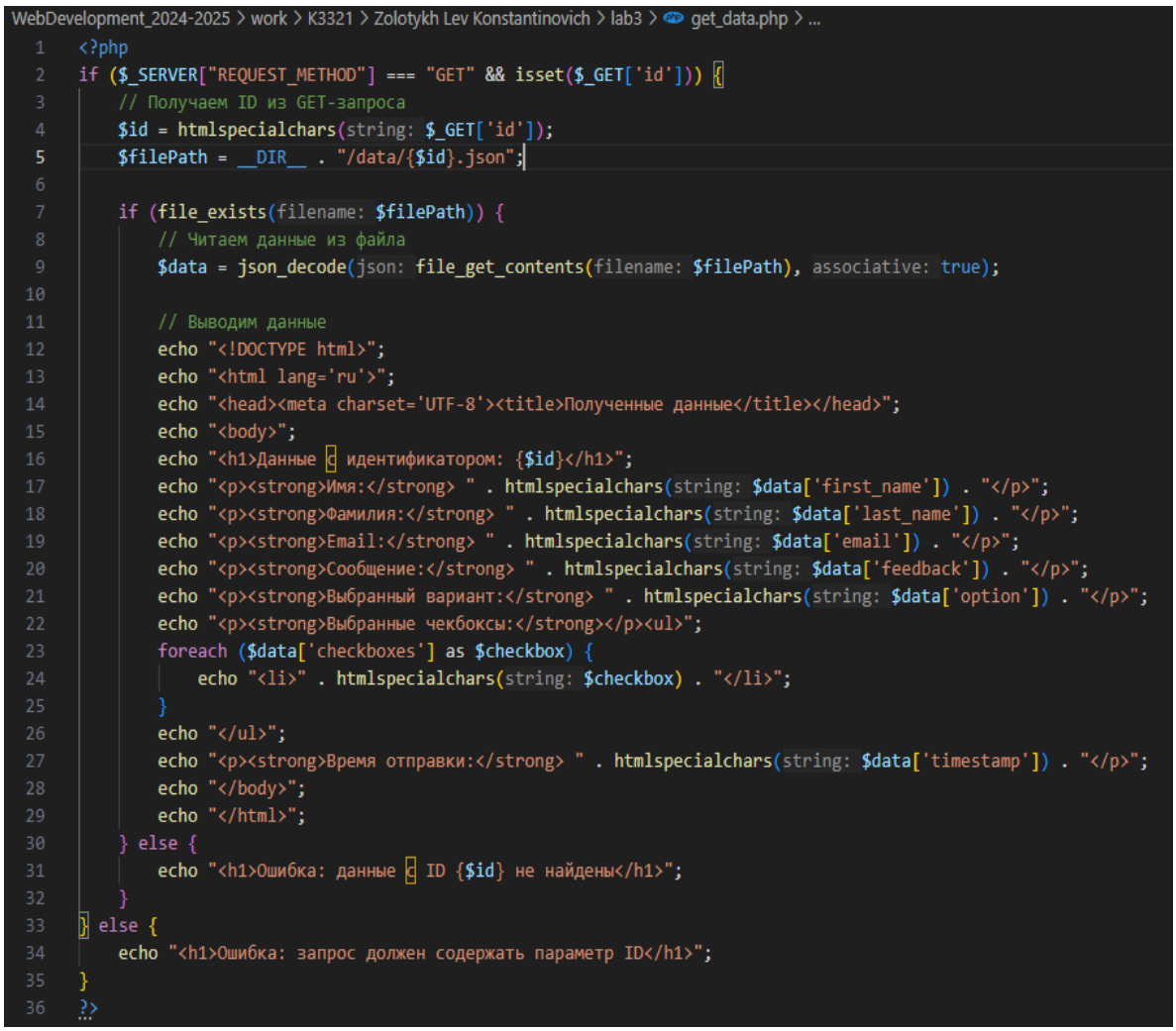

Рисунок 5 – save_data.php

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.

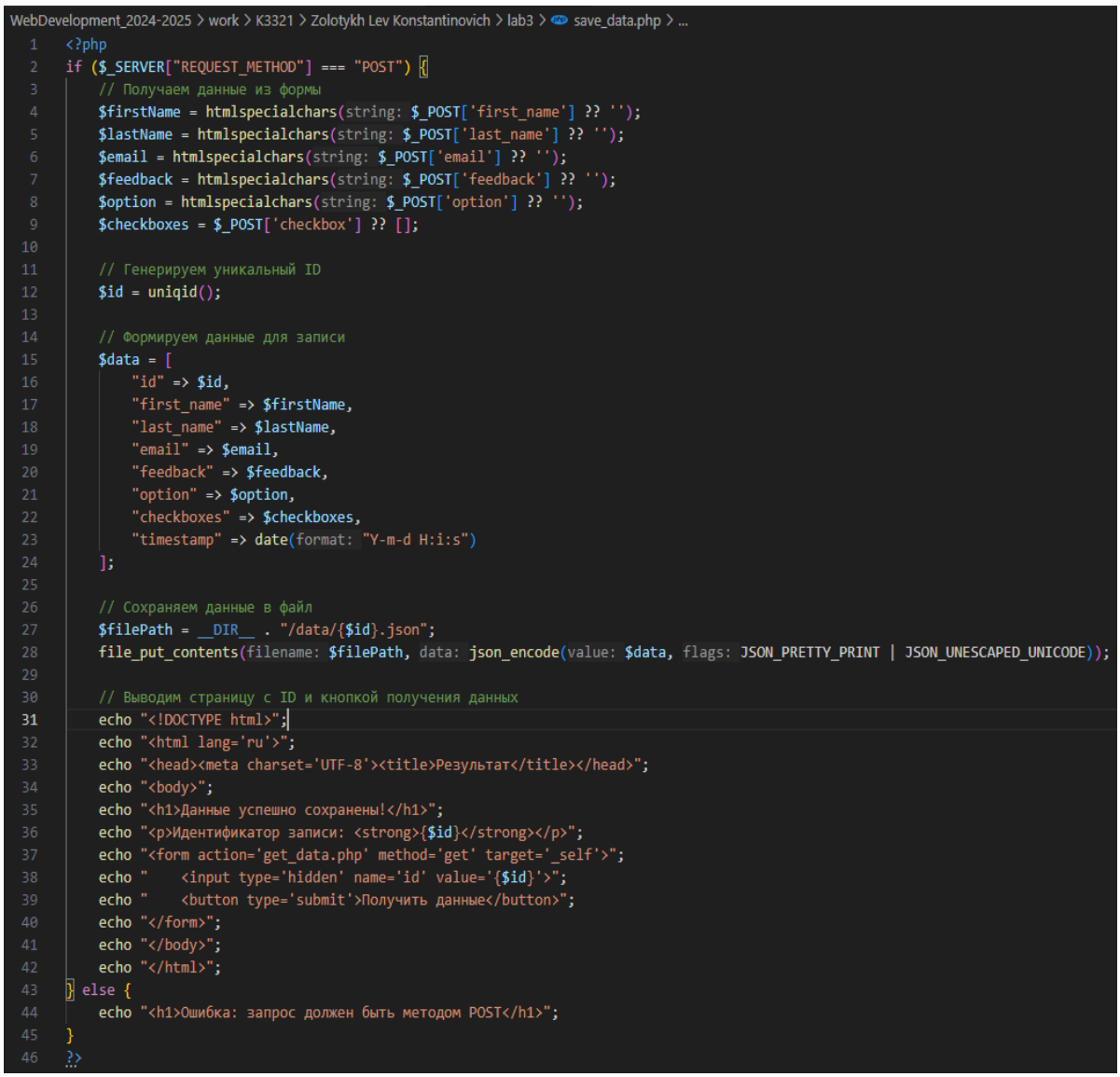

Рисунок 6 – get_data.php

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

```
##<VirtualHost *:80>
    ##ServerName dummy-host2.example.com
    ##ErrorLog "logs/dummy-host2.example.com-error.log"
    ##CustomLog "logs/dummy-host2.example.com-access.log" common
##</VirtualHost>

<VirtualHost *:80>
    DocumentRoot "C:/xampp/htdocs/test.site"
    ServerName test.site
    <Directory "C:/xampp/htdocs/test.site">
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

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:

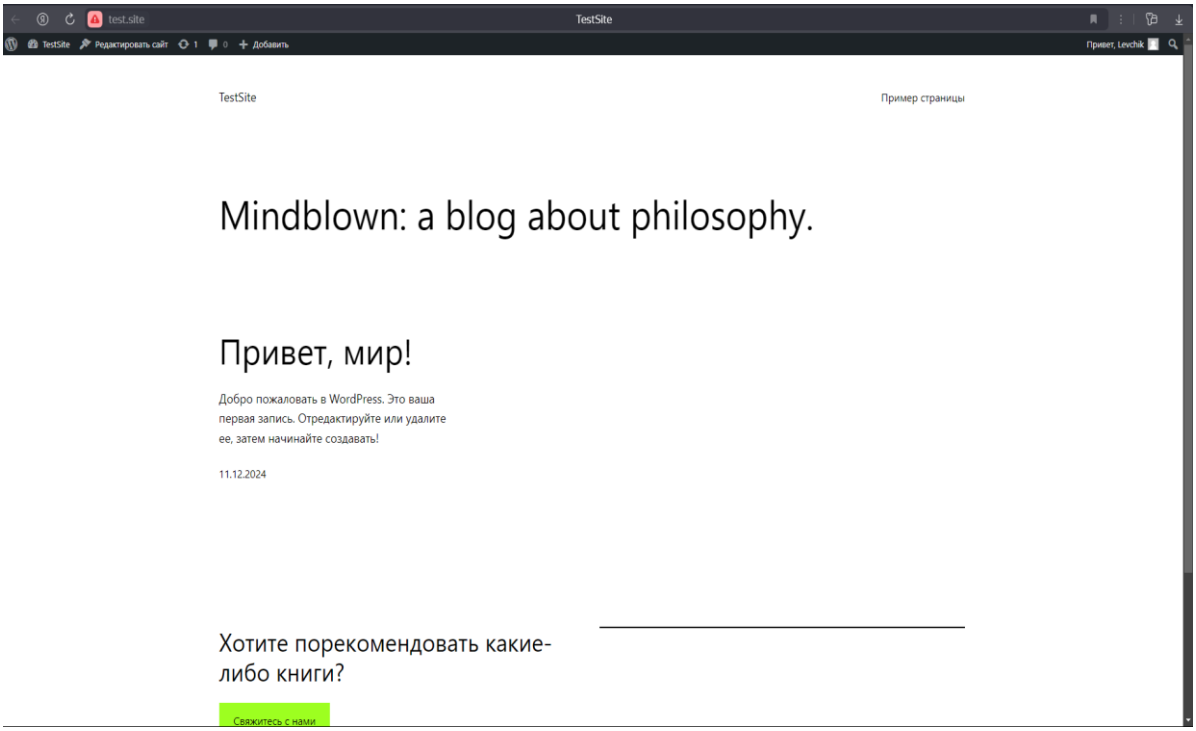

Рисунок 8 – внешний вид страницы test.site

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 инструментарий для локальной разработки сайтов, настроен хост и установлена система управления контентом wordpress.