Лабораторная работа 06а

ПвИ

**Задание 00**

1. Разработайте два сервера **06-00http** и **06-00express** с использованием модуля http и пакета express соответственно.
2. Оба сервера должны обрабатывать запросы, описанные в следующих заданиях.

**Задание 01** /connection?set=***set***

1. При GET-запросе **/connection** в окно браузера вывести текущее значение параметра **keepAliveTimeout.**
2. При GET-запросе **/connection?set=*set*** установить новое значение системного параметра **keepAliveTimeout=*set*** и вывести в окно браузера сообщение, что установлено новое значение параметра **keepAliveTimeout=*set***.
3. Продемонстрируйте влияние системного параметра **keepAliveTimeout** на работу приложения**.**

**Задание 02** /headers

1. Отобразите в окне браузера все заголовки запроса и ответа. Объясните назначение каждого заголовка.

**Задание 03**/user/***id***

1. Проанализируйте значениe параметра ***id***.
2. Если ***id*** имеет числовое значение, то выведите из файла users.json информацию о пользователе c соответствующим ***id***.
3. Иначе возвращайте клиенту ошибку (например, неверный параметр, пользователя с таким id нет). Для express-сервера ошибку необходимо обработать в специальном middleware для ошибок.

**Задание 04** middleware

1. Создайте промежуточное ПО (middleware), в котором будет выводиться в консоль информация о каждом запросе (метод, путь, время).
2. Примените промежуточное ПО ко всем маршрутам.

**Задание 05** status-code

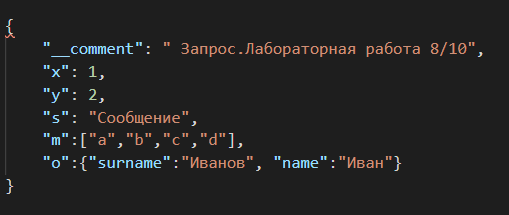
1. При получении запроса с неподдерживаемым url или методом, отличным от GET/POST, сформируйте ответ, имеющий необходимый статус и собственное пояснение к статусу.

**Задание 06** /formparameter

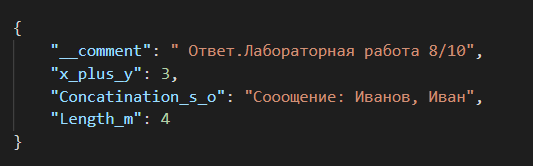
1. Используйте HTML-форму, включающую теги **input** с **type**: **text**, **number, checkbox, radiobutton,** тег **textarea**, а также два тега **input type=submit**, имеющих одно и тоже имя, но разные значения.
2. В окно браузера выведите значения параметров, полученных в запросе.

**Задание 07** /json

1. Принимайте POST-запросы, содержащие данные в json-формате и отправляйте ответы в json-формате.
2. Сообщение в запросе имеет следующую структуру:



1. Сообщение в ответе имеет следующую структуру:



Поле **x+y** ответасодержитсумму полей **x** и **y** запроса.

Поле **Concatination\_s\_o** ответа-конкатенациюполей **s** и свойств объекта **о** запроса.

Поле **Length\_m** ответа-количество элементов в массиве **m** запроса.

1. Проверьте работоспособность приложения с помощью POSTMAN.

**Задание 08** /public/***filename***

1. В ответ на запрос высылается файл с именем ***filename*** из директории **public.**
2. Если файл ***filename*** ненайден, возвращается ответ со статусом 404.
3. Проверьте работоспособность приложения с помощью браузера.

**Задание 09** /upload

1. В ответ на GET-запрос на **/upload** высылается web-форма, позволяющая отправить POST-запрос на **/upload,** присылающийсерверу файл.
2. Сервер сохраняет файл в директории **public.**
3. Проверьте работоспособность приложения с помощью браузера.

**Задание 10** Ответьте на следующие вопросы.

1. **Поясните назначение заголовка Content-Type.**

Content-Type: Заголовок HTTP, который указывает тип данных, содержащихся в теле запроса или ответа. Например, Content-Type: application/json указывает, что данные представлены в формате JSON.

1. **Поясните назначение заголовка Accept.**

Accept: Заголовок HTTP, который указывает типы медиа-ресурсов, которые клиент (браузер) может принимать. Например, Accept: application/json означает, что клиент готов принять данные в формате JSON.

1. **Для чего используется значение multipart/form-data заголовка Content-Type.**

Значение multipart/form-data в заголовке Content-Type используется для передачи файлов или бинарных данных в теле запроса. Этот тип контента позволяет кодировать данные так, чтобы они могли быть представлены в виде формы для отправки файлов.

1. **Как с помощью тега form, обеспечить значение multipart/form-data заголовка Content-Type.**

Для установки значения multipart/form-data в теле запроса с использованием тега <form>, нужно добавить атрибут enctype="multipart/form-data" к тегу <form>. Пример: <form action="/upload" method="POST" enctype="multipart/form-data">.

1. **Какое значение заголовка Content-Type отправляется тегом form в запросе по умолчанию.**

Значение application/x-www-form-urlencoded отправляется в запросе по умолчанию, если не указан атрибут enctype тега <form>. Этот тип кодировки используется для отправки формы с данными в URL-кодированном виде.

1. **Где и в каком формате передаются параметры в GET-запросе?**

Параметры в GET-запросе передаются в URL-строке после вопросительного знака (?). Например, http://example.com/page?name=value. Они представлены в виде пар ключ-значение и разделяются символом амперсанда (&).

1. **Где и в каком формате передаются параметры в POST-запросе?**

Параметры в POST-запросе передаются в теле запроса после заголовков. Они также представлены в виде пар ключ-значение, но не видны в URL. Данные могут быть закодированы разными способами, такими как application/x-www-form-urlencoded или multipart/form-data.

1. **Поясните понятие JSON?**

JSON (JavaScript Object Notation): Это формат обмена данными, основанный на синтаксисе объектов JavaScript. JSON предоставляет простой и удобный способ представления структурированных данных, который легко читать как человеку, и легко парсить и генерировать для программ.

1. **Поясните понятие XML?**

XML (eXtensible Markup Language): Это язык разметки, предназначенный для хранения и передачи данных. XML использует различные теги и атрибуты для описания структуры данных, и он может быть использован в различных областях, включая веб-разработку, конфигурацию приложений и другие сферы.

**multipart/form-data:**

**multipart/form-data** - это один из типов контента, который может использоваться при отправке данных через HTTP. Он обычно используется для отправки файлов и других бинарных данных. Формат **multipart/form-data** позволяет кодировать данные так, чтобы они могли быть представлены в виде формы для отправки файлов, и включает в себя следующие ключевые особенности:

1. **Бинарные данные:** Тип контента **multipart/form-data** позволяет отправлять бинарные данные, такие как изображения или файлы, в отличие от **application/x-www-form-urlencoded**, который предназначен для текстовых данных.
2. **Множественные части:** Данные могут быть разделены на несколько частей, каждая из которых представляет собой свою собственную часть данных. Каждая часть имеет свой уникальный идентификатор, что позволяет серверу различать разные части данных.
3. **Граница (boundary):** Для разделения частей данных используется граница, которая указывается в заголовке **Content-Type**. Граница должна быть уникальной и не встречаться в данных. Обычно она состоит из случайной строки.
4. **Заголовки для каждой части:** Каждая часть данных включает в себя собственные заголовки, содержащие метаинформацию о части, такую как имя поля формы, тип контента и другие атрибуты.

**Express vs. http:**

1. **Уровень абстракции:**
   * **http:** Это встроенный модуль Node.js, предоставляющий базовые возможности для работы с HTTP. Взаимодействие с запросами и ответами требует написания довольно много кода.
   * **Express:** Это сторонний веб-фреймворк для Node.js, предоставляющий высокоуровневый API для обработки HTTP-запросов и управления маршрутами. Он упрощает создание веб-приложений.
2. **Маршрутизация:**
   * **http:** Маршрутизация требует ручного разбора URL и определения, какой обработчик вызывать для каждого маршрута.
   * **Express:** Предоставляет удобный интерфейс для определения маршрутов и их обработчиков, делая маршрутизацию более декларативной.
3. **Управление мидлварами:**
   * **http:** Работа с промежуточным ПО (middleware) не столь удобна, и его подключение требует более тщательного кода.
   * **Express:** Имеет встроенную систему middleware, что делает управление промежуточным ПО более гибким и легким.
4. **View-движок и шаблоны:**
   * **http:** Не предоставляет интеграции с view-движками или удобным способом работы с шаблонами.
   * **Express:** Поддерживает view-движки, такие как EJS, Pug, и Handlebars, упрощая создание динамических страниц.
5. **Статические файлы:**
   * **http:** Работа с поддержкой статических файлов требует дополнительного кода.
   * **Express:** Предоставляет встроенный middleware для обслуживания статических файлов, что упрощает их обработку.

В целом, Express предоставляет более высокоуровневый и удобный интерфейс для работы с веб-приложениями, чем базовый модуль http.