пЛабораторная работа 08

ПСКП

ПОИТ-3

**Задание 00**

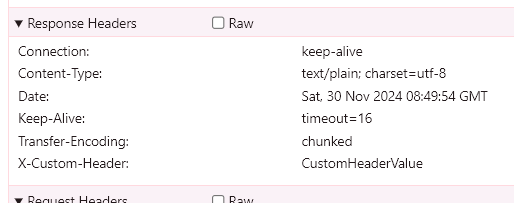
1. Разработайте приложение (сервер) **08-00** предназначенное для обработки следующих запросов.

|  |  |  |
| --- | --- | --- |
| HTTP-метод | URI | Задание |
| GET | /connection?set=***set*** | 01 |
| GET | /headers | 02 |
| GET | /parameter?x=***x***&&y=***y*** | 03 |
| GET | /parameter/***x***/***y*** | 04 |
| GET | /close | 05 |
| GET | /socket | 06 |
| GET | /req-data | 07 |
| GET | /resp-status?code=***c***?mess=***m*** | 08 |
| POST | /formparameter | 09 |
| POST | /json | 10 |
| POST | /xml | 11 |
| GET | /files | 12 |
| GET | /files/***filename*** | 13 |
| GET/POST | /upload | 14 |

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

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

**Задание 02** /headers



Надо было респонс хеадерс показать на странице

1. Отобразите в окне браузера все заголовки запроса и ответа.
2. Сформируйте собственный пользовательский заголовок ответа.
3. Убедитесь, что созданный пользовательский заголовок ответа, доставлен клиенту (с помощью браузера или POSTMAN).
4. Объясните назначение каждого заголовка.

**Задание 03** /parameter?x=***x***&&y=***y***

1. Проанализируйте значения параметров **x** и **y**.
2. Если **x** и **y** имеют числовые значения, то выведите в окно браузера сумму, разность, произведение и частное этих чисел
3. Иначе выведите сообщение об ошибке.

**Задание 04** /parameter/***x***/***y***

1. Проанализируйте значения параметров **x** и **y**.
2. Если **x** и **y** имеют числовые значения, то выведите в окно браузера сумму, разность, произведение и частное этих чисел.
3. Иначе выведите **URI**.

**Задание 05** /close

1. При получении этого запроса, в окно браузера выведите сообщение о закрытии сервера и остановите сервер через 10 секунд.

**Задание 06** /socket

1. При получении этого запроса, в окно браузера выведите ip- адрес, порт клиента и ip-адрес и порт сервера.

**Задание 07** /req-data

1. Продемонстрируйте в этом запросе, подобрав достаточно длинное сообщение в POSTMAN, порционную обработку запроса.

**Задание 08** resp-status?code=***c***?mess=***m***

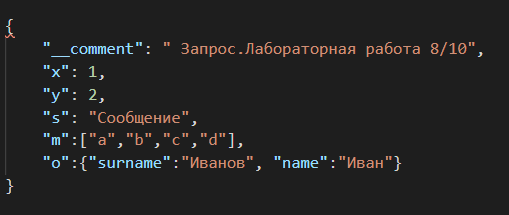
1. При получении этого запроса, сформируйте ответ, имеющий статус, заданный значением ***с*** и пояснение к статусу, заданное значением ***m***.

**Задание 09** /formparameter

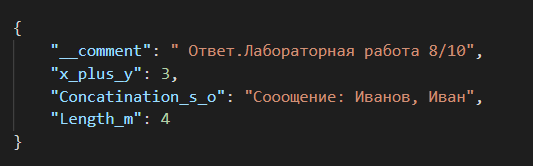
1. Использую HTML-форму включающую теги **input** с **type**: **text**, **number, date, checkbox, radiobutton,** тег **textarea**, а также два тега **input type=submit**, имеющих одно и тоже имя, но разные значения.
2. В окно браузера выведите значения параметров, полученных в запросе.

**Задание 10** /json

1. Принимаются POST-запросы, содержащие данные в json-формате и оправляет 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.

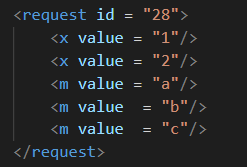
Поле **Concatination\_s\_o** ответаконкатенация полей **s** и свойств объекта **о** запроса.

Поле **Length\_m** ответасодержитколичество элементов в массиве **m** запроса.

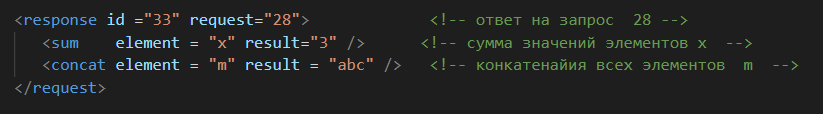
1. Проверьте работоспособность приложения с помощью POSTMAN.

**Задание 11. /**xml

1. Принимаются POST-запросы, содержащие данные в xml-формате и оправляет ответы в xml-формате.
2. Сообщение в запросе имеет следующую структуру.



1. Количество элементов **x** и **m** в запросе может быть произвольным.
2. Сообщение в ответе имеет следующую структуру.



1. Элемент **sum** в ответе один и содержит сумму всех значений элементов **x** (в атрибуте **value**).
2. Элемент **concat** в ответе один и содержит сумму всех значений элементов **m** (в атрибуте **value**).
3. Проверьте работоспособность приложения с помощью POSTMAN.

**Задание 12.** /files

1. В ответ на запрос высылается ответ с заголовком **X-static-files-count:** ***n***, где ***n*** -количество файлов в директории **static*.*** Используйте функции модуля **fs**.
2. Проверьте работоспособность приложения с помощью POSTMAN.

**Задание 13.** /files/***filename***

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

**Задание 14.** /upload

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

**Задание 15.** Ответьте на следующие вопросы.

1. Поясните назначение заголовка **Content-Type.**
2. Поясните назначение заголовка **Accept.**
3. Для чего используется значение **Multipart/form-data**  заголовка **Content-Type.**
4. Как с помощью тега **form**, обеспечить значение **Multipart/form-data**  заголовка **Content-Type.**
5. Какое значение заголовка **Content-Type** отправляется тегом **form** в запросе по умолчанию.
6. Где и в каком формате передаются параметры в GET-запросе?
7. Где и в каком формате передаются параметры в POST-запросе?
8. Поясните понятие **JSON**?
9. Поясните понятие **XML**?
10. Отличие get-запроса от post-запроса