Лабораторная работа 06

ПСКП

**Задание 00**

1. Разработайте приложение (сервер) **06-00** 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 | /socket | 05 |
| GET | /resp-status?code=***c***&mess=***m*** | 06 |
| POST | /formparameter | 07 |
| POST | /json | 08 |
| POST | /xml | 09 |
| GET | /files | 10 |
| GET | /files/***filename*** | 11 |
| GET/POST | /upload | 12 |

**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 **KepAliveTimeout=*set***.
3. Продемонстрируйте влияние системного параметра **KeepAliveTimeout** на работу приложения**.**

**Задание 02** /headers

1. Отобразите в окне браузера все заголовки запроса и ответа.
2. 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** /socket

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

**Задание 06** resp-status?code=***c***&mess=***m***

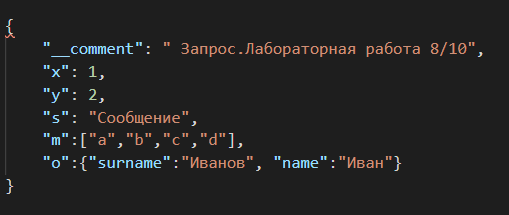
1. При получении этого запроса, сформируйте ответ, имеющий статус, заданный значением ***с***, и пояснение к статусу, заданное значением ***m***.

**Задание 07** /formparameter

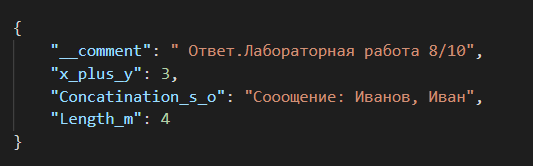
1. Используйте HTML-форму, включающую теги **input** с **type**: **text**, **number, date, checkbox, radiobutton,** тег **textarea**, а также два тега **input type=submit**, имеющих одно и тоже имя, но разные значения.
2. В окно браузера выведите значения параметров, полученных в запросе.

**Задание 08** /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.

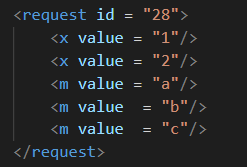
Поле **Concatination\_s\_o** ответа-конкатенациюполей **s** и свойств объекта **о** запроса.

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

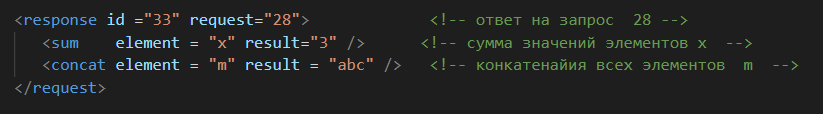
1. Проверьте работоспособность приложения с помощью POSTMAN.

**Задание 09 /**xml

1. Принимайте POST-запросы, содержащие данные в xml-формате и отправляйте 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 **result**).
2. Элемент **concat** в ответе один и содержит сумму всех значений элементов **m** (в атрибуте **result**).
3. Проверьте работоспособность приложения с помощью POSTMAN.

**Задание 10** /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 11** /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.

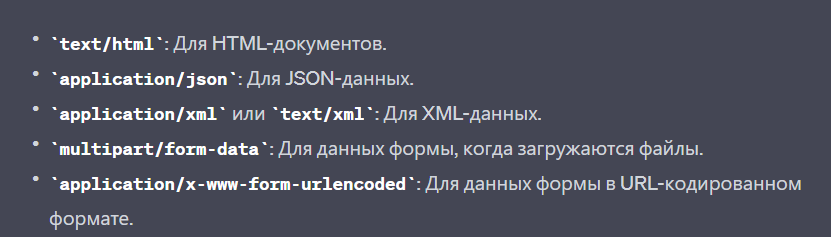
**Задание 12** /upload

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

**Задание 13** 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. Поясните назначение заголовка **Content-Type.**

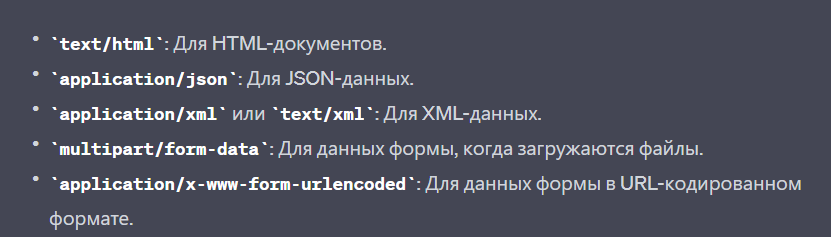
HTTP-заголовок **Content-Type** используется для определения типа данных, который содержится в теле HTTP-сообщения. Этот заголовок сообщает получателю о том, как интерпретировать или обрабатывать данные, которые были переданы в запросе или ответе.



1. Поясните назначение заголовка **Accept.**

HTTP заголовок запроса **Accept** указывает, какие типы контента, выраженные как [MIME типы](https://developer.mozilla.org/ru/docs/Web/HTTP/Basics_of_HTTP/MIME_types), клиент может понять. Используя [согласование контента](https://developer.mozilla.org/ru/docs/Web/HTTP/Content_negotiation), сервер затем выбирает одно из предложений, использует его и информирует клиента о своём выборе с помощью заголовка ответа [Content-Type](https://developer.mozilla.org/ru/docs/Web/HTTP/Headers/Content-Type). Браузеры задают адекватные значения для этого заголовка в зависимости от контекста, в котором выполняется запрос: при получении таблицы стилей CSS для запроса задаётся другое значение, чем при получении изображения, видео или скрипта.

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



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

Для того чтобы установить значение **multipart/form-data** в заголовке **Content-Type** с использованием тега **<form>**, вы должны установить атрибут **enctype** в значение **"multipart/form-data"**. Этот атрибут указывает на тип кодирования данных, используемый при отправке формы.

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

если вы не установили **enctype**, браузер будет использовать **application/x-www-form-urlencoded** в качестве значения **Content-Type** при отправке данных формы методом POST. Это значение предполагает, что данные формы будут отправлены в стандартном URL-кодированном виде.

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

В GET-запросах параметры передаются в URL-адресе. Они добавляются к концу URL после вопросительного знака (**?**) и имеют формат "ключ=значение". Если есть несколько параметров, они разделяются символом амперсанда (**&**).

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

В POST-запросах параметры передаются в теле HTTP-запроса, а не в URL-адресе, как это делается в GET-запросах. Формат передачи данных в теле POST-запроса зависит от заголовка **Content-Type**, который указывает тип контента (тип данных), используемого в теле запроса.

Два основных формата передачи данных в теле POST-запроса:

1. **application/x-www-form-urlencoded:**

Данные отправляются в виде строки, где параметры разделяются символом амперсанда (**&**), а значения параметров закодированы URL-кодированием.

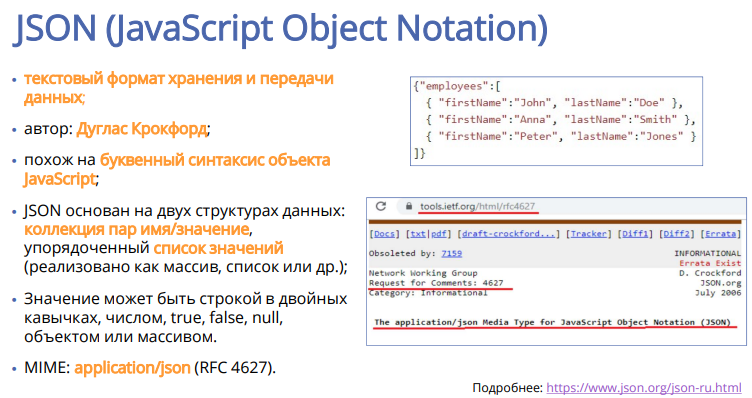
Заголовок Content-Type: **application/x-www-form-urlencoded**.

1. **multipart/form-data:**

Обычно используется при отправке файлов или больших объемов данных. Данные разделены на различные части (multipart parts).

Заголовок Content-Type: **multipart/form-data; boundary=----WebKitFormBoundary**.

1. Поясните понятие **JSON**?



1. Поясните понятие **XML**?

