Московский государственный технический университет

имени Н. Э. Баумана

В. И. Кузнецов

Взаимодействие с web сервисом через протокол HTTP

Методическое пособие к лабораторной работе 1 по дисциплине

«Языки интернет-программирования»

Редакция от 05.09.2025

Факультет «Информатика и системы управления»

Кафедра «Компьютерные системы и сети»

Кузнецов, В.И.

Взаимодействие посредством протокола HTTP, методы HTTP, взаимодействие в сети интернет, заголовки запросов HTTP: методические указания к выполнению практикума № 1 и лабораторной работы № 1 по дисциплинам «Языки интернет-программирования» и «Практикум по интернет-программированию» / В.И.Кузнецов. — Москва

Приведены основные теоретические сведения о взаимодействии в сети интернет посредством протокола HTTP необходимые для выполнения лабораторной работы и практикума по сетевому взаимодействию через протокол HTTP с помощью различных инструментов. Рассмотрены примеры. Также приведена литература по курсу и даны ссылки на интернет источники.

Для студентов МГТУ им. Н.Э. Баумана, обучающихся по направлению «Информатика и вычислительная техника».

Лабораторная работ № 1

Взаимодействие через протокол HTTP

Цель работы — знакомство с протоколом HTTP, получение и закрепление практических навыков взаимодействие через протокол с использованием различных методов и источников.

Объем работы — 4 часа.

ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

HTTP расшифровывается как **Hypertext Transfer Protocol**, протокол передачи гипертекста. Сейчас это один из самых популярных протоколов интернет, основа Web.

HTTP находится на **прикладном уровне** в моделях OSI и TCP/IP.

Концепцию Web предложил в 1989 году Тим Бернерс-Ли из Европейского центра ядерных исследований (ЦЕРН). Основные компоненты Web по предложению Тима Бернерс-Ли:

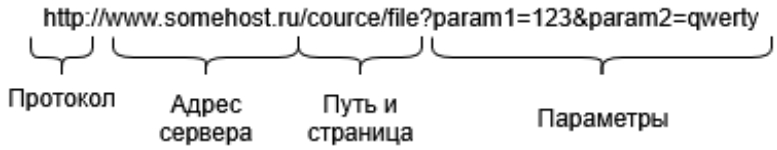
- Язык гипертекстовой разметки страниц HTML.
- Протокол передачи гипертекстовых страниц HTTP.
- Web-сервер.
- Текстовый web-браузер.

Сейчас Web устроен более сложно, браузеры поддерживают не только текст, но и изображения, видео, могут запускать код на JavaScript и многое другое.

Протокол HTTP используется браузером для того, чтобы загрузить с Web-сервера HTML страницы и другие ресурсы, которые нужны для показа страниц. Также HTTP сейчас активно применяется в API.

URL

Важную роль в работе HTTP играет Uniform Resource Locator, сокращенно URL – единообразный определитель местонахождения ресурса. Именно URL используется для того, чтобы указать, к какой странице мы хотим получить доступ.



Структура URL

URL состоит из трех основных частей:

- **Название протокола**, в примере на рисунке протокол HTTP.
- **Адрес сервера**, на котором размещен ресурс. Можно использовать IP-адрес или доменное имя. Адрес сервера отделяется от названия протокола двоеточием и двумя слешами.
- **Адрес ресурса на сервере**. Это может быть HTML-страница, изображение, видео или ресурс другого типа. В примере на рисунке адрес страницы: /courses/networks.

В URL не обязательно использовать только протокол HTTP, вот примеры с другими протоколами:

- https://ya.ru
- ftp://example.com

Также URL не обязательно должен указывать на страницы в HTML, можно использовать ресурсы в разных форматах, например:

- Текст – https://www.ietf.org/rfc/rfc959.txt
- Изображение – https://habr.com/img/maskable_icon.png

URL может включать достаточно большое количество других компонентов, кроме протокола, адреса сервера и адреса ресурса. Более подробно почитать о них можно в документе RFC 1738, Uniform Resource Locators (URL).

Версии протокола HTTP

В ЦЕРН в 1991 году разработали экспериментальную версию протокола **HTTP 0.9**. Первая открытая версия **HTTP 1.0** появилась в 1996 году, она описана в документе RFC 1945.

Почти сразу после HTTP 1.0, в 1997 году, вышла обновленная версия протокола **HTTP 1.1**. В этой версии добавили кэширование, постоянное соединение, аутентификацию и некоторые другие возможности. Документы RFC, описывающие HTTP 1.1, несколько раз обновлялись, сейчас действует RFC 9112 от 2022 года. Версия HTTP 1.1 используется сейчас.

В 2015 году вышла версия **HTTP/2** (описана в RFC 9113), основная задача которой – повышение производительности. Самая последняя версия **HTTP/3**, описанная в RFC 9114, использует на транспортном уровне протокол QUIC вместо TCP.

Версии HTTP/1.1, HTTP/2 и HTTP/3 отличаются форматом передачи данных по сети, но логика работы у них одинаковая. Эта логика описана в документе RFC 9110 HTTP Semantics.

HTTP передает данные по сети в открытом виде, что не очень хорошо с точки зрения безопасности. Шифрование данных используется в протоколе **HyperText Transfer Protocol Secure**, сокращенно HTTPS.

Мы рассмотрим версию HTTP/1.1 и логику работы HTTP в целом.

Протокол HTTP

Протокол HTTP работает в режиме запрос-ответ. Клиент, например, браузер, передает на сервер запрос к определенному ресурсу, например, Web-странице. Сервер в ответ отправляет клиенту этот ресурс или сообщение об ошибке, если ресурс передать нельзя.

HTTP/1.1 использует текстовый режим работы, то есть от клиента к серверу и обратно передаются обычные строки. Это просто и удобно для людей, которые могут понять, что именно передается без использования дополнительных инструментов. Однако для компьютеров это не самый удобный формат, т.к. приходится заниматься парсингом строк. Поэтому HTTP/2 и HTTP/3 используют бинарный режим работы.

На транспортном уровне HTTP использует протокол TCP (кроме HTTP/3, в котором применяется QUIC), порт Web-сервера по-умолчанию: 80.

Запрос HTTP

Запрос HTTP состоит из трех основных частей:

- **Запрос**
- **Заголовки** (не обязательно)
- **Тело сообщения** (не обязательно)

```
GET /courses/networks HTTP/1.1
Host: some.ru
```

- В первой строке указывается сам запрос, который также состоит из трех частей:
- **Метод HTTP** GET указывает, какое действие требуется выполнить с ресурсом. В примере метод GET говорит о том, что мы хотим получить (загрузить) ресурс.
- **Адрес ресурса** /courses/networks – путь к странице на Web-сервере, которую мы хотим загрузить.
- **Версия протокола** HTTP/1.1.

В первой строке указывается сам запрос, который также состоит из трех частей:

- **Метод HTTP** GET указывает, какое действие требуется выполнить с ресурсом. В примере метод GET говорит о том, что мы хотим получить (загрузить) ресурс.

- **Адрес ресурса** /courses/networks – путь к странице на Web-сервере, которую мы хотим загрузить.
- **Версия протокола** HTTP/1.1.

Во второй строке указывается заголовок Host. Этот заголовок является обязательным в версии HTTP/1.1, в нем задается доменное имя сервера, к которому направлен запрос. Сейчас активно используется виртуальный хостинг: один Web-сервер на одном IP-адресе может обслуживать несколько сайтов с разными доменными именами. Чтобы понять, какому именно сайту предназначен запрос, нужно указать доменное имя сайта в заголовке Host.

В HTTP используется следующий **формат** заголовка:

Название заголовка: *значение заголовка*

В примере название заголовка Host, значение – asozykin.ru (доменное имя сайта, к которому направлен запрос).

Если заголовков в запросе несколько, то каждый указывается в отдельной строке.

Методы HTTP

Метод HTTP говорит о том, какое действие с ресурсом мы хотим совершить. В примере запроса мы видели метод GET, который предназначен для получения ресурса. Кроме GET в HTTP есть и другие методы, наиболее важные из которых определены в документе RFC 9110 HTTP Semantics.

Название метода	**Описание метода**
GET	Запрос на передачу ресурса.
HEAD	Запрос на передачу ресурса, но сам ресурс в ответе не передается, только заголовки.
POST	Передача данных на сервер для обработки указанного ресурса.
PUT	Размещение ресурса на сервере (если такой ресурс уже есть на

	сервере, то он замещается).
DELETE	Удаление ресурса на сервере.
CONNECT	Установка соединение с сервером на основе ресурса.
OPTIONS	Запрос поддерживаемых методов HTTP для ресурса и других параметров коммуникации.
TRACE	Запрос на трассировку сообщения: сервер должен включить в свой ответ исходный запрос, на который он отвечает. Это полезно, когда запрос проходит через промежуточные устройства, которые могут изменить запрос, например, добавить заголовки.

Полный список всех существующих методов HTTP находится в документе Hypertext Transfer Protocol (HTTP) Method Registry, который сопровождается организацией Internet Assigned Numbers Authority (IANA). IANA поддерживает не только этот документ, но и большое количество других документов с возможными значениями различных параметров сетевых протоколов.

Следует отметить, что методы HTTP и соответствующие им действия – это соглашения, которые желательно, но не обязательно соблюдать. Вполне возможно, что некоторая реализация Web-сервера в ответ на запрос GET будет удалять запрошенный ресурс. Однако большая часть реализаций соблюдает соглашения.

На практике в Web чаще всего используются два метода: GET для запроса Web-страниц и POST для передачи данных из браузера на сервер, например, после заполнения формы.

Ответ HTTP

Ответ HTTP, как и запрос, состоит из трех частей:

- **Статус ответа**
- **Заголовки** (не обязательно)
- **Тело сообщения** (не обязательно)

Пример ответа на запрос выше:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Content-Length: 5161

Текст Web-страницы…
```

В HTTP/1.1 ответ, так же как и запрос, представляет собой обычные текстовые строки.

Первая строка ответа состоит из двух частей:

- **Версия протокола**, в примере HTTP/1.1. Версию указывать не обязательно.
- **Код статуса ответа,** в примере 200 OK, запрос обработан успешно.

Ответ содержит два заголовка, которые следуют после строки с кодом статуса:

- **Content-Type**, тип содержимого, которое передается в теле сообщения. Значение "text/html; charset=UTF-8" означает, что в теле сообщения страница HTML в кодировке UTF-8.
- **Content-Length**, размер содержимого, в байтах. В примере размер страницы в теле сообщения 5161 байт.

После заголовков в теле ответа находится запрошенная Web-страница (в примере она не показана для сокращения объема). Тело сообщения отделено от заголовков пустой строкой.

Коды статусов ответов HTTP

Первая строка ответа HTTP содержит код статуса ответа – число в диапазоне от 100 до 599, которое характеризует результат выполнения запроса. Возможные коды статусов ответов описаны в документе RFC 9110 HTTP Semantics.

Коды статусов ответов разделены на пять классов, которые определяются по первой цифре кода:

- **1XX (информация)**: запрос получен, обработка продолжается.
- **2XX (успешное выполнение)**: запрос был успешно принят и понят.
- **3XX (перенаправление)**: для выполнения запроса необходимо предпринять дополнительные действия.
- **4XX (ошибка клиента)**: запрос содержит синтаксическую ошибку или не может быть выполнен.
- **5XX (ошибка сервера)**: запрос от клиента оформлен правильно, но при его обработке произошла ошибка на стороне сервера.

Часто встречающиеся коды статусов ответов приведены в таблице.

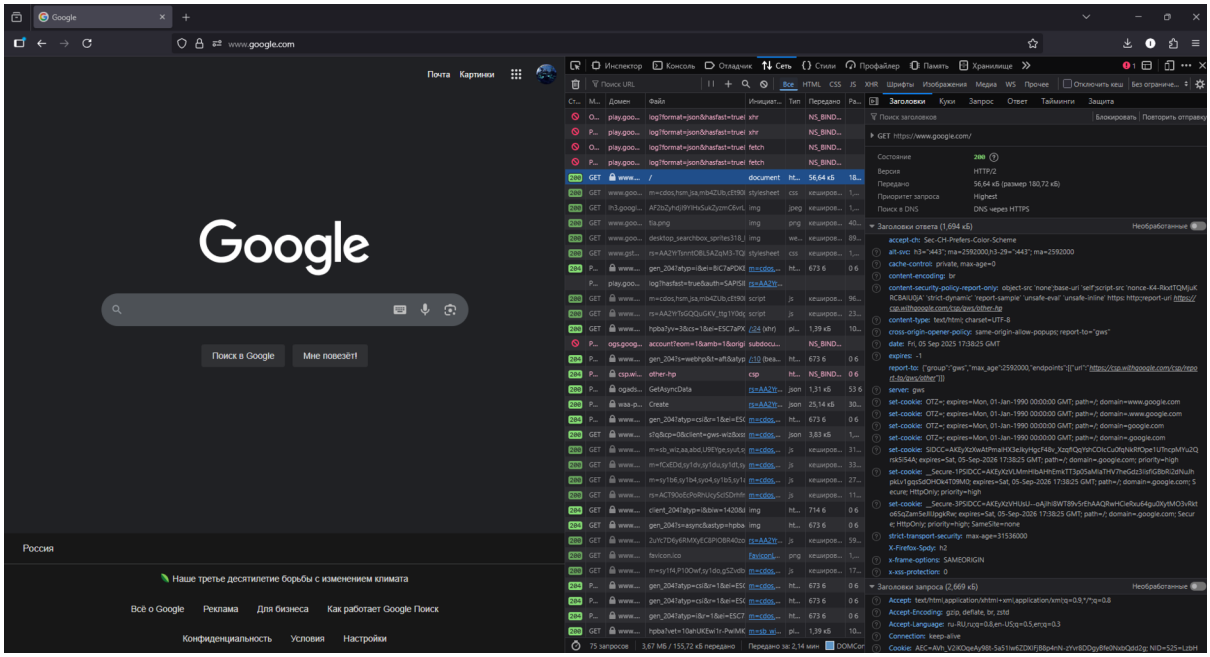
Код статуса ответа	**Описание**
1XX (информация)	
101 Switching Protocols	Запрос принят, сервер предлагает дальнейшее взаимодействие выполнять по другому протоколу (например, WebSocket). Желаемый протокол должен быть указан в заголовке *Upgrade* ответа.
2XX (успешное выполнение)	
200 OK	Запрос выполнен успешно.
201 Created	В результате выполнения запроса на сервере был успешно создан ресурс (например, в ответ на запрос PUT).
3XX (перенаправление)	
301 Moved Permanently	Запрошенный ресурс был перемещен. Новый URL ресурса указывается в заголовке ответа *Location*. В дальнейшем

	клиенту рекомендуется использовать новый URL.
302 Found	Запрошенный ресурс был временно перемещен в другое место. Новый URL ресурса указывается в заголовке ответа *Location*. В дальнейшем клиенту рекомендуется использовать старый URL, т.к. перемещение временное.
304 Not Modified	Запрошенный ресурс не был изменен, поэтому можно взять ресурс из кэша, а не передавать его по сети.
4XX (ошибка клиента)	
400 Bad Request	Запрос не может быть обработан из-за ошибки синтаксиса.
403 Forbidden	Доступ к запрошенному клиентом ресурсу запрещен.
404 Not Found	Запрошенный ресурс не найден на сервере.
5XX (ошибка сервера)	
500 Internal Server Error	Запрос не может быть выполнен из-за внутренней ошибки в программном обеспечении сервера.
501 Not Implemented	Сервер не поддерживает запрошенную функциональность, например, не может выполнить запрошенный метод HTTP для указанного ресурса.
505 HTTP Version Not Supported	Версия HTTP, указанная в запросе, не поддерживается.

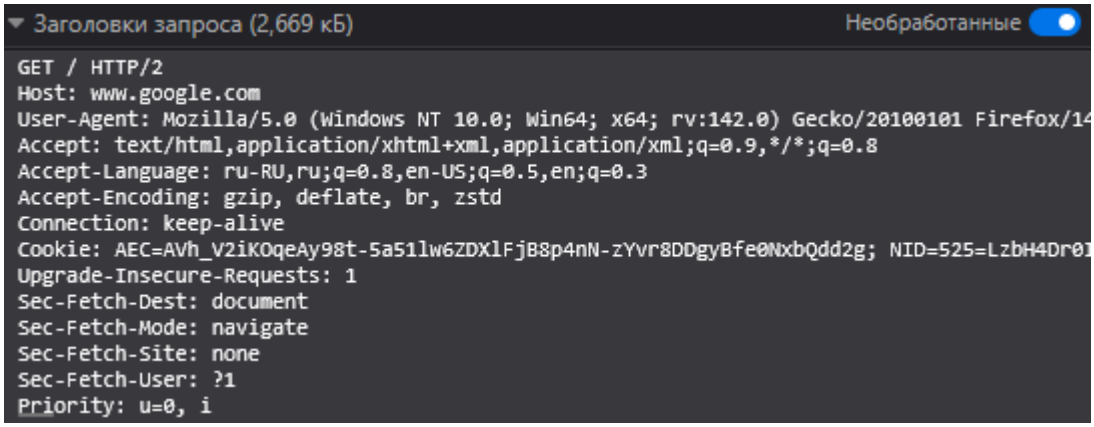
Полный список кодов ответов с описанием можно посмотреть в разделе «Status Codes» документа RFC 9110.

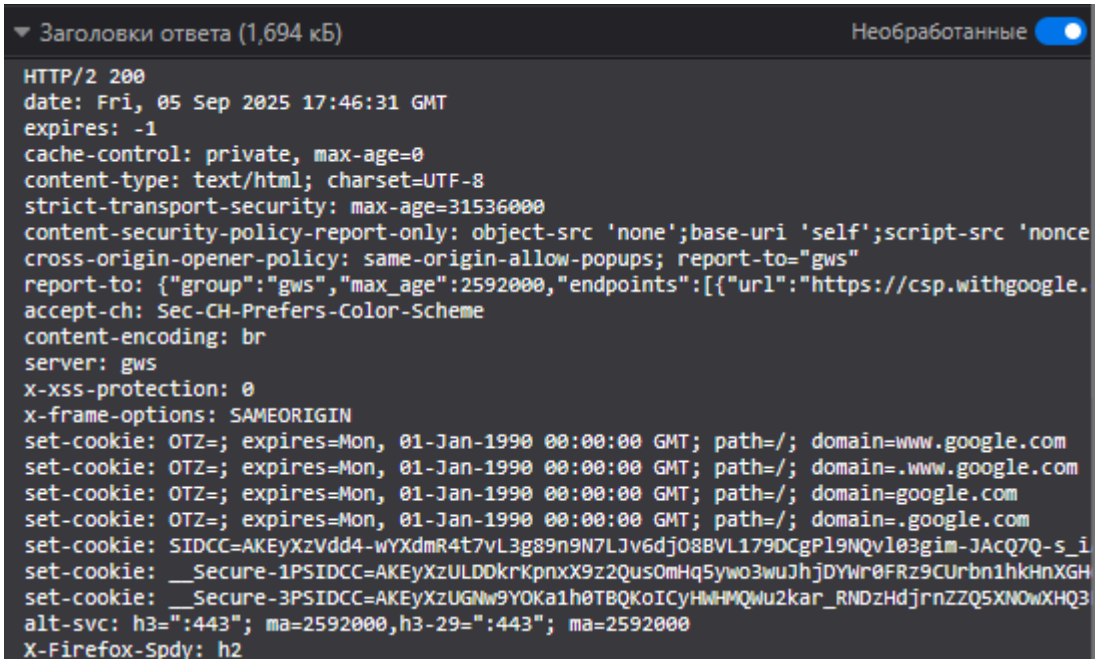
Инструменты для анализа и проведения запросов

Самым доступным инструментом является любой современный браузер имеющий встроенный режим разработчика.



Открытие осуществляется по нажатию клавиши F12. В разделе сеть описываться все запросы, произведенные браузером при переходе по любому адресу. Описываются заголовки как отправленные, так и полученные в ответ на запрос. А также тело ответа, в котором находится данные, отправленные сервером.

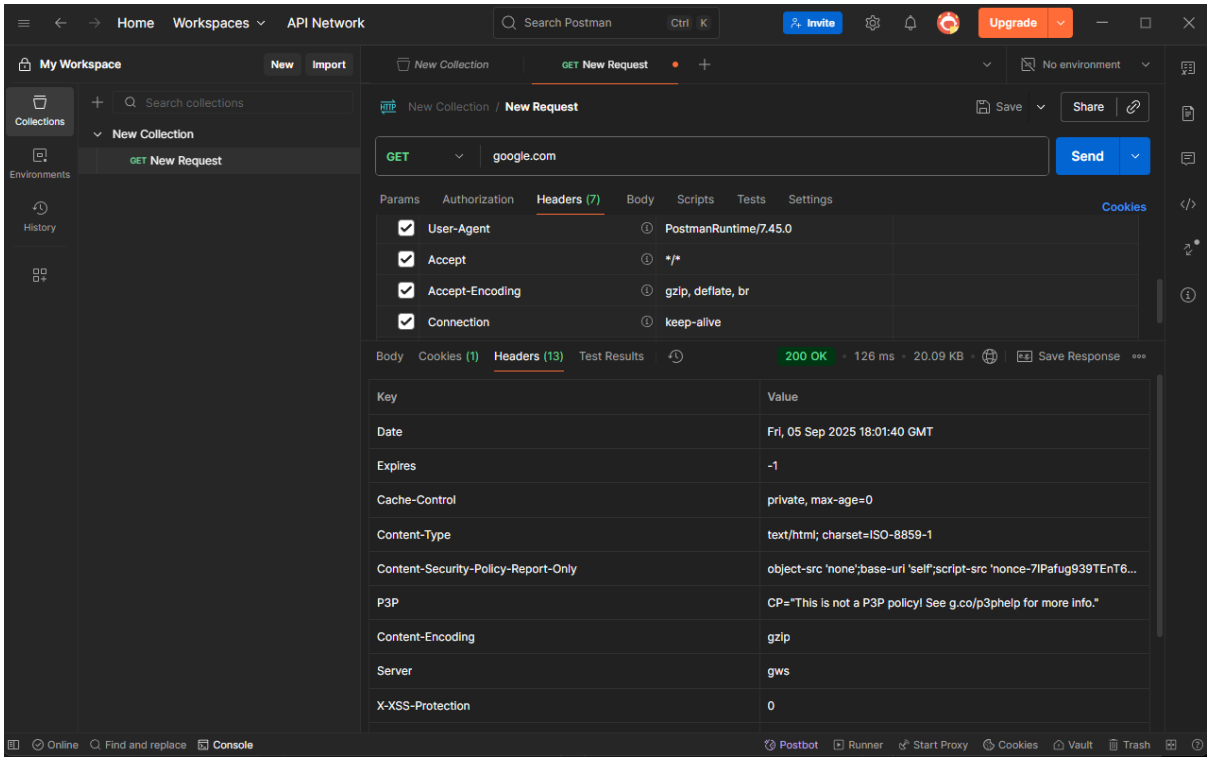




Браузер как инструмент удобен для проверки ответов от удаленного веб сервера. Им не удобно делать свои запросы, но удобно проверять отправляемые браузером или запущенным внутри браузера скриптами запросы.

Другим инструментом являются инструментом для тестирования API. Это инструменты позволяющие проводить запросы с различными параметрами и по различным адресам, группирую их или автоматизируя по порядку исполнения и данным. Они так же позволяют проводить запросы других протоколов GRAPHQL, gRPC, MQTT, WebSocket и тд. Создаваемый запрос можно обогащать своими хедерами, куками, параметрами и отправляемыми данным

Двумя самыми используемыми инструментами являются POSTMAN и INSOMNIA. POSTMAN распространяется бесплатно с ограниченным функционалом, но достаточным для выполнения работы.


Home
Workspaces
API Network
Search Postman
Ctrl K
Invite
Upgrade
My Workspace
New
Import
Collections
Environments
History
Search collections
New Collection
GET New Request
New Collection
GET New Request
No environment
New Collection / New Request
Save
Share
GET
google.com
Send
Params
Authorization
Headers (7)
Body
Scripts
Tests
Settings
Cookies
User-Agent
PostmanRuntime/7.45.0
Accept
/
Accept-Encoding
gzip, deflate, br
Connection
keep-alive
Body
Cookies (1)
Headers (13)
Test Results
200 OK
126 ms
20.09 KB
Save Response
Key
Value
Date
Fri, 05 Sep 2025 18:01:40 GMT
Expires
-1
Cache-Control
private, max-age=0
Content-Type
text/html; charset=ISO-8859-1
Content-Security-Policy-Report-Only
object-src 'none';base-uri 'self';script-src 'nonce-7IPafug939TEnT6...
P3P
CP="This is not a P3P policy! See g.co/p3phelp for more info."
Content-Encoding
gzip
Server
gws
X-XSS-Protection
0
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Vault
Trash

Порядок выполнения задания

1. В браузере открыть ресурс с www.w3.org с включенной консолью разработчика и проинспектировать отправляемые запросы. Найти первичный запрос к странице и определить отправленные и полученные заголовки и данные
2. Скачать и установить инструмент тестирования API (INSOMNIA или POSTMAN). Произвести запрос GET запрос www.w3.org
3. Скачать с GITHUB https://github.com/Emetless/YaIP/blob/main/lb1/base.exe файл с тестовым веб сервером. В случае использование операционных систем LINUX или MAC OS. Установить Python последней версии, выполнить в консоли команду:

   ```
   pip install flask
   ```

   Скачать файл https://github.com/Emetless/YaIP/blob/main/lb1/base.py и запустить его командой консоли

   ```
   python base.py
   ```

   Провести следующие локальные запросы к данному сервису по адресу отображенному в консоли:

   1) GET без указания пути
   2) GET по пути /success
   3) GET по пути /unsuccess
   4) GET по пути /check с заголовком ***MyHeader*** и значением ***check*** и любым другим значением
   5) GET по пути /getparam с любым заполненным параметром запроса
   6) POST по пути /post/check с телом ***check*** и любым другим значением

Отчет по выполненной работе должен содержать все проделанные запросы с результатом их выполнения заголовками и телом запроса и должен быть оформлен в виде pdf-файла.

СОДЕРЖАНИЕ ОТЧЕТА

Отчет должен включать:

1) ФИО студента и номер группы;

2) наименование лабораторной работы;

3) названия выполненных пунктов и тексты реализованных HTML и CSS-разметки с указанием имен файлов.

Отчет предоставляется в электронном виде в формате pdf или odt. Зачет ставится при условии выполнения всех пунктов задания, демонстрации работы программы и при наличии отчета и устных ответов на контрольные вопросы.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Для чего предназначен HTTP протокол?
2. Состав URL
3. Состав HTTP запроса
4. Методы HTTP
5. Классы кодов ответа HTTP