Протокол HTTP

HTTP (HyperText Transfer Protocol) – протокол передачи гипертекста . Он является главным протоколом Всемирной паутины (World Wide Web) и описывает формат сообщений, которыми могут обмениваться клиенты и серверы. HTTP определен в RFC[1] 2616. В этом протоколе каждое взаимодействие состоит из одного ASCII-запроса, на который следует один ответ стандарта RFC 822 MIME[2].

Данный протокол работает на основе TCP-соединения, и хотя формально это требование не является обязательным, на практике оно почти всегда выполняется. Это обусловлено тем, что в случае использования TCP ни браузеру, ни серверу не надо беспокоиться о потерянных данных и разбиении больших сообщений на части. Все это выполнят службы TCP-протокола.

Протокол HTTP разработан таким образом, что может использоваться не только в веб-технологиях но и в других объектно-ориентированных приложениях.

Каждое HTTP-сообщение состоит из трёх частей, которые передаются в указанном порядке:

1. Стартовая строка (*Starting line*) — определяет тип сообщения;
2. Заголовки (*Headers*) — характеризуют тело сообщения, параметры передачи и прочие сведения;
3. Тело сообщения (*Message Body*) — непосредственно данные сообщения. Обязательно должно отделяться от заголовков пустой строкой.

Заголовки и тело сообщения могут отсутствовать, но стартовая строка является обязательным элементом, так как указывает на тип запроса/ответа.

Стартовая строка

Стартовые строки различаются для запроса и ответа. Строка запроса выглядит так:

`Метод URI` `HTTP/`**`Версия`** — для остальных версий.

Здесь:

- **Метод** (*Method*) — название запроса, одно слово заглавными буквами. В версии HTTP 0.9 использовался только метод `GET`,
- **URI (Universal Resource Identifier) -** определяет путь к запрашиваемому документу.

- **Версия** (*Version*) — пара разделённых точкой арабских цифр. Например: `1.0`.

---

[1] RFC - это серия документов об Интернет, создаваемая с 1969 года. Эти документы описывают множество стандартов, касающихся компьютерных коммуникаций, сетевых протоколов, процедур, программ и концепций. Все стандарты Интернет описаны в RFC.
http://tools.ietf.org/html/rfc2616

[2] MIME определяет механизмы для передачи разного рода информации внутри текстовых данных (в частности, с помощью электронной почты), а именно: текст на языках, для которых используются кодировки, отличные от ASCII, и нетекстовый контент, такой как картинки, музыка, фильмы и программы.

Простейший HTTP запрос может выглядеть так:

```
GET http://www.vk.com/ HTTP/1.0rnrn
```

Сстартовая строка ответа сервера имеет следующий формат:

```
HTTP/Версия КодСостояния Пояснение
```

Здесь:

- **Версия** — пара разделённых точкой арабских цифр как в запросе.
- **КодСостояния** (*Status Code*) — три арабские цифры. По коду статуса определяется дальнейшее содержимое сообщения и поведение клиента.
- **Пояснение** (*Reason Phrase*) — текстовое короткое пояснение к коду ответа для пользователя. Никак не влияет на сообщение и является необязательным.

Например, на предыдущий наш запрос клиентом данной страницы сервер ответил строкой:

```
HTTP/1.0 200 Ok
```

Пример:

```
HTTP/1.1 200 OK

Content-Type: text/html;

charset=UTF-8

Content-Length: 98

<html>

<head>  <title>An Example Page</title> </head>

<body> <p>Hello World</p> </body>

</html>
```

Встроенные методы HTTP-запросов

Метод	Описание
GET	Прочитать веб-страницу
HEAD	Прочитать заголовок веб-страницы
PUT	Сохранить веб-страницу
POST	Добавить к веб-странице
DELETE	Удалить веб-страницу
TRACE	Отослать назад запрос
CONNECT	Зарезервировано для будущего использования

OPTIONS	Отобразить параметры

Метод **GET** запрашивает страницу (под страницей может подразумеваться любой объект, хотя на практике чаще всего это просто обычный файл), закодированную согласно стандарту MIME. Это наиболее часто употребляемый метод. Его структура следующая:

Метод **HEAD** запрашивает заголовок сообщения, причем само страница(тело) не загружается. С помощью данного метода можно, например, узнать время последнего обновления страницы, что необходимо для управления кэшированием страниц. Кроме того, с помощью этого метода можно просто проверить работоспособность данного URL.

С помощью метода **PUT** можно поместить страницу на сервер. Тело этого запроса содержит размещаемую страницу, которая может быть закодирована согласно MIME. Метод требует идентификации пользователя.

POST является методом, добавляющим содержимое к уже имеющейся странице. Он может использоваться, например, для добавления записи на форуме.

```
Пример 1:

POST /api/v.1.0/user/create/ HTTP/1.1

Host:  rest-server.ru

Accept: application/json

Content-Type: application/json

Content-Length: 28

{ "Id": 12345, "User": "John" }


Пример 2:

POST http://www.site.ru/registration.html HTTP/1.0

Host: www.site.ru

Referer: http://www.site.ru/index.html

Cookie: isauth=True

Content-Type: application/x-www-form-urlencoded

Content-Length: 35

login=Petya%20Vasechkin&password=qq
```

Метод **DELETE** уничтожает страницу. Как и в методе PUT здесь требуется аутентификация, подтверждающая права пользователя на данное действие.

Метод **TRACE** схож с командой Ping в TCP/IP. Сообщение этого запроса обязательно содержит в заголовке поле “Max-Forwards”. Каждый раз, когда это сообщение проходит через прокси-сервер, значение данного поля уменьшается на единицу. Сервер, который получил это сообщение с нулевым значением, отсылает ответ. Если сообщение получает сервер, которому оно адресовалось, то он отсылает ответ, не обращая внимания на значение поля “Max-Forwards”. Используя последовательность таких запросов, клиент может узнать все прокси-сервера, используемые при передаче запрошенного ресурса.

Метод **CONNECT** зарезервирован и в настоящее время не используется.

При помощи **OPTIONS** клиент может запросить сервер о его свойствах или о свойствах какого-либо конкретного файла.

В ответ на каждый запрос сервер генерирует ответ, содержащий строку состояния, а так же, возможно, и другую информацию, например веб-страницу. Строка состояния может содержать код состояния, информирующий об успешном выполнении запроса, а в противном случае – о причине неудач. Коды разделены на пять основных групп.

Группы кодов состояния

Группа кода	Тип	Коды
1xx	Готовность	100 – сервер готов обрабатывать запросы клиента
2xx	Успех	200 – запрос успешно обработан 204 – содержимое отсутствует
3xx	Перенаправление	301 – страница перемещена 304 – кэшированная страница все еще актуальна
4xx	Ошибка клиента	403 – ошибка доступа 404 – страница не найдена
5xx	Ошибка сервера	500 – внутренняя ошибка сервера 503 – указание предпринять попытку позднее

Некоторые заголовки сообщений HTTP

Заголовок	Тип	Содержимое
User-Agent	Запрос	Информация о браузере и платформе
Accept	Запрос	Поддерживаемые клиентом типы страниц
Accept-Charset	Запрос	Поддерживаемые клиентов наборы символов
Accept-	Запрос	Поддерживаемые клиентом типы кодирования (методы сжатия

Encoding		информации)
Accept-Language	Запрос	Естественные языки, воспринимаемые клиентом
Host	Запрос	DNS-сервера
Authorization	Запрос	Список идентификаторов клиента
Cookie	Запрос	Отправка ранее принятого cookie-файла на сервер
Date	Запрос / ответ	Дата и время отправки сообщения
Upgrade	Запрос / ответ	Поддерживаемые протоколы. Может использоваться для перехода на будущие версии протокола HTTP, которые, возможно, будут несовместимы с предыдущими.
Server	Ответ	Информация о сервере
Content-Encoding	Ответ	Тип кодирования содержимого (методы сжатия информации)
Content-Language	Ответ	Естественный язык, который используется на странице
Content-Length	Ответ	Размер страницы в байтах
Content-Type	Ответ	MIME-тип страницы
Last-Modified	Ответ	Время и дата последнего обновления страницы
Location	Ответ	Команда клиенту на перенаправление его запроса другому серверу. Используется при «переезде» страницы или использовании «зеркал», на которых хранится копия страницы
Accept-Ranges	Ответ	Готовность сервера принимать запросы на страницы указанного размера. Это позволяет пересылать страницы по частям, что может потребоваться, если страница слишком большая и клиент не может принять сразу всю страницу целиком
Set-Cookie	Ответ	Команда клиенту сохранить cookie

Пример. Запрос – ответ.

http://vk.com/id1

```
GET /id1 HTTP/1.1
Host: vk.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0; ru; rv:1.9.1.5)
Gecko/20091102 Firefox/3.5.5
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ru,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: windows-1251,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300m
Connection: keep-alive
Cookie: ***************
Cookie: remixchk=5;
remixsid=7d538efdc5e49616a9de2fdb562583b78f2d3390865cd6e3423a16b1;
remixnews_privacy_filter=0; remixnews_privacy_filter=0; remixclosed_tabs=0
```

```
HTTP/1.1 200 OK
Server: nginx/0.7.59
Date: Wed, 18 Nov 2009 21:31:12 GMT
Content-Type: text/html; charset=windows-1251
Transfer-Encoding: chunked
Connection: keep-alive
X-Powered-By: PHP/5.2.6-1+lenny3
Pragma: no-cache
Cache-control: no-store
Content-Encoding: gzip
Vary: Accept-Encoding
```