# Просто об HTTP

- Как отправить HTTP-запрос?
- Как прочитать ответ?
- А что с безопасностью?
- А есть дополнительные возможности?
- Что-то ещё, кстати, используют?
- И что, всё?

Вашему вниманию предлагается описание основных аспектов протокола HTTP — сетевого протокола, с начала 90-х и по сей день позволяющего вашему браузеру загружать веб-страницы. Данная статья написана для тех, кто только начинает работать с компьютерными сетями и заниматься разработкой сетевых приложений, и кому пока что сложно самостоятельно читать официальные спецификации.

**HTTP** — широко распространённый протокол передачи данных, изначально предназначенный для передачи гипертекстовых документов (то есть документов, которые могут содержать ссылки, позволяющие организовать переход к другим документам).

Аббревиатура HTTP расшифровывается как *HyperText Transfer Protocol*, «протокол передачи гипертекста». В соответствии со спецификацией OSI, HTTP является протоколом прикладного (верхнего, 7-го) уровня. Версия протокола, HTTP 1.1, описана в спецификации RFC 2616.

Протокол HTTP предполагает использование клиент-серверной структуры передачи данных. Клиентское приложение формирует запрос и отправляет его на сервер, после чего серверное программное обеспечение обрабатывает данный запрос, формирует ответ и передаёт его обратно клиенту. После этого клиентское приложение может продолжить отправлять другие запросы, которые будут обработаны аналогичным образом.

Задача, которая традиционно решается с помощью протокола HTTP — обмен данными между пользовательским приложением, осуществляющим доступ к веб-ресурсам (обычно это веб-браузер) и веб-сервером. На данный момент именно благодаря протоколу HTTP обеспечивается работа Всемирной паутины.

Также HTTP часто используется как протокол передачи информации для других протоколов прикладного уровня, таких как SOAP, XML-RPC и WebDAV. В таком случае говорят, что протокол HTTP используется как «транспорт».

API многих программных продуктов также подразумевает использование HTTP для передачи данных — сами данные при этом могут иметь любой формат, например, XML или JSON.

Как правило, передача данных по протоколу HTTP осуществляется через TCP/IP-соединения. Серверное программное обеспечение при этом обычно использует TCP-порт 80 (и, если порт не указан явно, то обычно клиентское программное

обеспечение по умолчанию использует именно 80-й порт для открываемых HTTP-соединений), хотя может использовать и любой другой.

**ПРИМЕЧАНИЕ**_____

**Версии HTTP**

| Год | HTTP версия |
|---|---|
| 1991 | 0.9 |
| 1996 | 1.0 |
| 1997 | 1.1 |
| 2015 | 2.0 |

---

## Как отправить HTTP-запрос?

Самый простой способ разобраться с протоколом HTTP — это попробовать обратиться к какому-нибудь веб-ресурсу вручную. Представьте, что вы браузер, и у вас есть пользователь, который очень хочет прочитать статьи Анатолия Ализара.

Предположим, что он ввёл в адресной строке следующее:

```
http://alizar.habrahabr.ru/
```

Соответственно вам, как веб-браузеру, теперь необходимо подключиться к веб-серверу по адресу alizar.habrahabr.ru.

Для этого вы можете воспользоваться любой подходящей утилитой командной строки. Например, telnet:

```
telnet alizar.habrahabr.ru 80
```

Сразу уточню, что если вы вдруг передумаете, то нажмите Ctrl + «]», и затем ввод — это позволит вам закрыть HTTP-соединение. Помимо telnet можете попробовать nc (или ncat) — по вкусу.

После того, как вы подключитесь к серверу, нужно отправить HTTP-запрос. Это, кстати, очень легко — HTTP-запросы могут состоять всего из двух строчек.

Для того, чтобы сформировать HTTP-запрос, необходимо составить стартовую строку, а также задать по крайней мере один заголовок — это заголовок Host,

который является обязательным, и должен присутствовать в каждом запросе. Дело в том, что преобразование доменного имени в IP-адрес осуществляется на стороне клиента, и, соответственно, когда вы открываете TCP-соединение, то удалённый сервер не обладает никакой информацией о том, какой именно адрес использовался для соединения: это мог быть, например, адрес alizar.habrahabr.ru, habrahabr.ru или m.habrahabr.ru — и во всех этих случаях ответ может отличаться. Однако фактически сетевое соединение во всех случаях открывается с узлом 212.24.43.44, и даже если первоначально при открытии соединения был задан не этот IP-адрес, а какое-либо доменное имя, то сервер об этом никак не информируется — и именно поэтому этот адрес необходимо передать в заголовке Host.

Стартовая (начальная) строка запроса для HTTP 1.1 составляется по следующей схеме:

Метод URI HTTP/Версия

Например (такая стартовая строка может указывать на то, что запрашивается главная страница сайта):

```
GET / HTTP/1.1
```

**Метод** (в англоязычной тематической литературе используется слово *method*, а также иногда слово *verb* — «глагол») представляет собой последовательность из любых символов, кроме управляющих и разделителей, и определяет операцию, которую нужно осуществить с указанным ресурсом. Спецификация HTTP 1.1 не ограничивает количество разных методов, которые могут быть использованы, однако в целях соответствия общим стандартам и сохранения совместимости с максимально широким спектром программного обеспечения как правило используются лишь некоторые, наиболее стандартные методы, смысл которых однозначно раскрыт в спецификации протокола.

| Метод | Назначение |
|---|---|
| GET | Прочитать ресурс |
| PUT | Создать ресурсу |
| POST | Отредактировать данные ресурса |
| DELETE | Удалить ресурс |

**URI** (*Uniform Resource Identifier*, унифицированный идентификатор ресурса) — путь до конкретного ресурса (например, документа), над которым необходимо осуществить операцию (например, в случае использования метода GET подразумевается получение ресурса). Некоторые запросы могут не относиться к какому-либо ресурсу, в этом случае вместо URI в стартовую строку может быть добавлена звёздочка (астериск, символ «*»). Например, это может быть запрос, который относится к самому веб-серверу, а не какому-либо конкретному ресурсу. В этом случае стартовая строка может выглядеть так:

```
OPTIONS * HTTP/1.1
```

**Версия** определяет, в соответствии с какой версией стандарта HTTP составлен запрос. Указывается как два числа, разделённых точкой (например **1.1**).

Для того, чтобы обратиться к веб-странице по определённому адресу (в данном случае путь к ресурсу — это «/»), нам следует отправить следующий запрос:

```
GET / HTTP/1.1
Host: alizar.habrahabr.ru
```

При этом учитывайте, что для переноса строки следует использовать символ возврата каретки (Carriage Return), за которым следует символ перевода строки (Line Feed). После объявления последнего заголовка последовательность символов для переноса строки добавляется дважды.

Впрочем, в спецификации HTTP рекомендуется программировать HTTP-сервер таким образом, чтобы при обработке запросов в качестве межстрочного разделителя воспринимался символ LF, а предшествующий символ CR, при наличии такового, игнорировался. Соответственно, на практике бóльшая часть серверов корректно обработает и такой запрос, где заголовки отделены символом LF, и он же дважды добавлен после объявления последнего заголовка.

Если вы хотите отправить запрос в точном соответствии со спецификацией, можете воспользоваться управляющими последовательностями \r и \n:

```
echo -en "GET / HTTP/1.1\r\nHost: alizar.habrahabr.ru\r\n\r\n" | ncat
alizar.habrahabr.ru 80
```

**ПРИМЕЧАНИЕ**_____
Разберём адрес HTML документа в сети интернет чуть подробнее. Полный формат выглядит так:

```
https://<логин>:<пароль>@<хост>:<порт>/<путь>?<параметры>#<якорь>
```

Например:

```
https://mylogin:123456@google.com:443/search?text=метрика#results
```

| Параметр | Пример | Назначение |
|---|---|---|
| <логин> | mylogin | Имя пользователя для доступа к ресурсу (если необходимо) |
| <пароль> | 123456 | Пароль для доступа к ресурсу (если необходим) |

| Параметр | Пример | Назначение |
| --- | --- | --- |
| <хост> | google.com | Доменное имя сервера (Domain Name), где расположен ресурс |
| <порт> | 443 | Номер порта по которому доступен ресурс. По умолчанию, 80 для HTTP и 443 для HTTPS – если так, его можно не указывать. На одном и том же сервере по разным портам можно получать разные ресурсы. |
| <путь> | /search | Путь до ресурса. В примере, форма поиска по Интернету. Это может быть HTML страницы или файл |
| <параметры> | text=метрика | Параметры доступа к ресурсу. В примере, запрос «метрика» в форму поиска. |
| <якорь> | results | Прокрутить страницу до элемента с id="results" |

---

## Как прочитать ответ?

Стартовая строка ответа имеет следующую структуру:

HTTP/Версия Код состояния Пояснение

**Версия** протокола здесь задаётся так же, как в запросе.

**Код состояния** (*Status Code*) — три цифры (первая из которых указывает на класс состояния), которые определяют результат совершения запроса. Например, в случае, если был использован метод GET, и сервер предоставляет ресурс с указанным идентификатором, то такое состояние задаётся с помощью кода 200. Если сервер сообщает о том, что такого ресурса не существует — 404. Если сервер сообщает о том, что не может предоставить доступ к данному ресурсу по причине отсутствия необходимых привилегий у клиента, то используется код 403. Спецификация HTTP 1.1 определяет 40 различных кодов HTTP, а также допускается расширение протокола и использование дополнительных кодов состояний.

**Пояснение** к коду состояния (*Reason Phrase*) — текстовое (но не включающее символы *CR* и *LF*) пояснение к коду ответа, предназначено для упрощения чтения ответа человеком. Пояснение может не учитываться клиентским программным обеспечением, а также может отличаться от стандартного в некоторых реализациях серверного ПО.

После стартовой строки следуют заголовки, а также тело ответа. Например:

```
HTTP/1.1 200 OK
Server: nginx/1.2.1
Date: Sat, 08 Mar 2014 22:53:46 GMT
Content-Type: application/octet-stream
Content-Length: 7
Last-Modified: Sat, 08 Mar 2014 22:53:30 GMT
Connection: keep-alive
Accept-Ranges: bytes

Wisdom
```

Тело ответа следует через два переноса строки после последнего заголовка. Для определения окончания тела ответа используется значение заголовка **Content-Length** (в данном случае ответ содержит 7 восьмеричных байтов: слово «Wisdom» и символ переноса строки).

Но вот по тому запросу, который мы составили ранее, веб-сервер вернёт ответ не с кодом 200, а с кодом 302. Таким образом он сообщает клиенту о том, что обращаться к данному ресурсу на данный момент нужно по другому адресу.

Смотрите сами:

```
HTTP/1.1 302 Moved Temporarily
Server: nginx
Date: Sat, 08 Mar 2014 22:29:53 GMT
Content-Type: text/html
Content-Length: 154
Connection: keep-alive
Keep-Alive: timeout=25
Location: http://habrahabr.ru/users/alizar/
```

```
<html>
<head><title>302 Found</title></head>
<body bgcolor="white">
<center><h1>302 Found</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

В заголовке Location передан новый адрес. Теперь URI (идентификатор ресурса) изменился на /users/alizar/, а обращаться нужно на этот раз к серверу по адресу habrahabr.ru (впрочем, в данном случае это тот же самый сервер), и его же указывать в заголовке Host.

То есть:

```
GET /users/alizar/ HTTP/1.1
Host: habrahabr.ru
```

В ответ на этот запрос веб-сервер Хабрахабра уже выдаст ответ с кодом 200 и достаточно большой документ в формате HTML.

Если вы уже успели вжиться в роль, то можете теперь прочитать полученный от сервера HTML-код, взять карандаш и блокнот, и нарисовать профайл Ализара — в принципе, именно этим бы на вашем месте браузер сейчас и занялся.

## А что с безопасностью?

Сам по себе протокол HTTP не предполагает использование шифрования для передачи информации. Тем не менее, для HTTP есть распространённое расширение, которое реализует упаковку передаваемых данных в криптографический протокол **SSL** или **TLS**.

Название этого расширения — **HTTPS** (*HyperText Transfer Protocol Secure*). Для HTTPS-соединений обычно используется TCP-порт 443. HTTPS широко используется для защиты информации от перехвата, а также, как правило, обеспечивает защиту от атак вида **man-in-the-middle** — в том случае, если сертификат проверяется на клиенте, и при этом приватный ключ сертификата не был скомпрометирован, пользователь не подтверждал использование неподписанного сертификата, и на компьютере пользователя не были внедрены сертификаты центра сертификации злоумышленника.

На данный момент HTTPS поддерживается всеми популярными веб-браузерами.

## А есть дополнительные возможности?

Протокол HTTP предполагает достаточно большое количество возможностей для расширения. В частности, спецификация HTTP 1.1 предполагает возможность использования заголовка Upgrade для переключения на обмен данными по другому протоколу. Запрос с таким заголовком отправляется клиентом. Если серверу требуется произвести переход на обмен данными по другому протоколу, то он может вернуть клиенту ответ со статусом «426 Upgrade Required», и в этом случае клиент может отправить новый запрос, уже с заголовком Upgrade.

Такая возможность используется, в частности, для организации обмена данными по протоколу WebSocket (протокол, описанный в спецификации RFC 6455, позволяющий обеим сторонам передавать данные в нужный момент, без отправки дополнительных HTTP-запросов): стандартное «рукопожатие» (handshake) сводится к отправке HTTP-запроса с заголовком Upgrade, имеющим значение «websocket», на который сервер возвращает ответ с состоянием «101 Switching Protocols», и далее любая сторона может начать передавать данные уже по протоколу WebSocket.

## Что-то ещё, кстати, используют?

На данный момент существуют и другие протоколы, предназначенные для передачи веб-содержимого. В частности, протокол **SPDY** (произносится как английское слово *speedy*, не является аббревиатурой) является модификацией протокола HTTP, цель которой — уменьшить задержки при загрузке веб-страниц, а также обеспечить дополнительную безопасность.

Увеличение скорости обеспечивается посредством сжатия, приоритизации и мультиплексирования дополнительных ресурсов, необходимых для веб-страницы, чтобы все данные можно было передать в рамках одного соединения.

Опубликованный в ноябре 2012 года черновик спецификации протокола HTTP 2.0

(следующая версия протокола HTTP после версии 1.1, окончательная спецификация для которой была опубликована в 1999) базируется на спецификации протокола SPDY.

Многие архитектурные решения, используемые в протоколе SPDY, а также в других предложенных реализациях, которые рабочая группа httpbis рассматривала в ходе подготовки черновика спецификации HTTP 2.0, уже ранее были получены в ходе разработки протокола HTTP-NG, однако работы над протоколом HTTP-NG были прекращены в 1998.

На данный момент поддержка протокола SPDY есть в браузерах Firefox, Chromium/Chrome, Opera, Internet Exporer и Amazon Silk.

## И что, всё?

В общем-то, да. Можно было бы описать конкретные методы и заголовки, но фактически эти знания нужны скорее в том случае, если вы пишете что-то конкретное (например, веб-сервер или какое-то клиентское программное обеспечение, которое связывается с серверами через HTTP), и для базового понимания принципа работы протокола не требуются. К тому же, всё это вы можете очень легко найти через Google — эта информация есть и в спецификациях, и в Википедии, и много где ещё.

Впрочем, если вы знаете английский и хотите углубиться в изучение не только самого HTTP, но и используемых для передачи пакетов TCP/IP, то рекомендую прочитать вот эту статью.

Ну и, конечно, не забывайте, что любая технология становится намного проще и понятнее тогда, когда вы фактически начинаете ей пользоваться.

Удачи и плодотворного обучения!