HTTP (англ. *HyperText Transfer Protocol* — «протокол передачи гипертекста») — протокол прикладного уровня передачи данных для распределенных, совместных гипермедиа информационных систем.[1] HTTP является основой передачи данных для World Wide Web.

HTTP был предложен Тим Бернерс-Ли в CERN в 1989 году разработки стандартов HTTP координировались Internet Engineering Task Force(IETF) и Консорциумом World Wide Web (W3C), что привело к публикации серии документов RFCs. Первое определение HTTP / 1.1, версия HTTP общего пользования, определено в RFC 2068 в 1997 году, но потом дополнено RFC 2616 в 1999 году, а затем снова RFC 7230 и еще в 2014 году.
Более поздняя версия протокола HTTP / 2 , была стандартизирована в 2015 году, и в настоящее время поддерживается основными веб - серверами[2]
В соответствии со спецификацией OSI, HTTP является протоколом прикладного (верхнего, 7-го) уровня. Актуальная на данный момент версия протокола, HTTP 1.1, описана в спецификации RFC 2616.

Основой HTTP является технология «клиент-сервер», то есть предполагается существование:

- Потребителей (клиентов), которые инициируют соединение и посылают запрос ;
- Поставщиков (серверов), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом. Ответ содержит информацию о запросе о состоянии завершения и может также содержать запрошенное содержимое в своем теле сообщения.

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.

HTTP используется также в качестве «транспорта» для других протоколов прикладного уровня, таких как SOAP, XML-RPC, WebDAV.
Основным объектом манипуляции в HTTP является ресурс, на который указывает URI (Uniform Resource Identifier) в запросе клиента. Обычно такими ресурсами являются хранящиеся на сервере файлы, но ими могут быть логические объекты или что-то абстрактное. Особенностью протокола HTTP является возможность указать в запросе и ответе способ представления одного и того же ресурса по различным параметрам: формату, кодировке, языку и т. д.

HTTP сессии

HTTP сессия представляет собой последовательность операций запрос-ответ сети. Клиент HTTP инициирует запрос путем создания протокола управления передачей соединения (TCP) к определенному порту на сервере (обычно порт 80, иногда порт 8080, см Список TCP и UDP номера порта). HTTP-сервер прослушивает этот порт ожидает сообщения с запросом клиента. При получении запроса, сервер отправляет обратно строку состояния, например "HTTP / 1.1 200 OK", и сообщение о своей собственной. Тело этого сообщения, как правило, это запрашиваемый ресурс, хотя это может быть сообщение об ошибке или другая информация, которая также может быть возвращена.[1]

Методы

HTTP определяет методы, чтобы указать желаемое действие , которое будет выполняться на идентифицированном ресурсе. Часто, ресурс указывает путь файлу или выходу исполняемого файла, размещенного на сервере. HTTP / 1.0 спецификация [11] определил GET, POST и HEAD - методы и HTTP / 1.1 спецификации [12] добавили 5 новых методов: OPTIONS, PUT, DELETE, TRACE и CONNECT. Любой клиент может использовать любой метод, и сервер может быть настроен для поддержки любой комбинации методов. Если метод неизвестен, он будет рассматриваться как небезопасные и не-идемпотентного метод. Нет ограничений на количество методов, которые могут быть определены, и это позволяет для будущих методов возможность быть указаными без нарушения существующей инфраструктуры. Например, WebDAV определил 7 новых методов и RFC 5789 уточнила PATCH метод. *(См. Список полей заголовка HTTP)*

GET
Метод GET запрашивает представление указанного ресурса. Запросы, использующие метод GET должны только извлекать данные и не должны иметь никакого другого эффекта. (Это также верно в отношении некоторых других методов HTTP.) [1] W3C опубликовал принципы руководство по этому вопросу различия, говоря: "При проектировании Веб – приложения следует проинформировать вышеуказанными принципами, а также соответствующими ограничениями."[13] См безопасные методы ниже.
HEAD
Метод HEAD запрашивает ответ, аналогичный запросу GET, но без тела ответа. Это полезно для извлечения мета-информации, записанную в заголовках ответа, без необходимости транспортировать все содержимое.
POST
Метод POST Применяется для передачи пользовательских данных заданному ресурсу. Данные POSTa могут быть, к примеру, аннотацией для существующих ресурсов; сообщением на доске объявлений, группой новостей, списком рассылок или потоком комментариев; блоком данных, который является результатом представления веб – формы в процессе обработки данных; или элементом для добавления в базу данных.[14]

В отличие от метода `GET`, метод `POST` не считается идемпотентным[4], то есть многократное повторение одних и тех же запросов `POST` может возвращать разные результаты (например, после каждой отправки комментария будет появляться очередная копия этого комментария).

PUT
Метод PUT применяется для загрузки содержимого запроса на указанный в запросе URI. Если URI относится к уже существующему ресурсу, он изменяется; если URI не указывает на существующий ресурс, то сервер может создать ресурс с этим URI.[15]
Фундаментальное различие методов `POST` и `PUT` заключается в понимании предназначений URI ресурсов. Метод `POST` предполагает, что по указанному URI будет производиться обработка передаваемого клиентом содержимого. Используя `PUT`, клиент предполагает, что загружаемое содержимое соответствует находящемуся по данному URI ресурсу.

DELETE
Метод DELETE удаляет указанный ресурс.

TRACE
Метод TRACE повторяет принятый запрос таким образом, чтобы клиент мог видеть, что (если таковые имеются) изменения или дополнения были сделаны в запросх на промежуточных серверах.

OPTIONS
Метод OPTIONS возвращает HTTP-методы, которые поддерживает сервер для указанного URL-адреса. Это может использоваться, чтобы проверить функциональность веб-сервера путем запроса '*' вместо конкретного ресурса. В ответ серверу следует включить заголовок `Allow` со списком поддерживаемых методов. Также в заголовке ответа может включаться информация о поддерживаемых расширениях.
Предполагается, что запрос клиента может содержать тело сообщения для указания интересующих его сведений. Формат тела и порядок работы с ним в настоящий момент не определён; сервер пока должен его игнорировать. Аналогичная ситуация и с телом в ответе сервера.

CONNECT
[16]Метод CONNECT преобразует соединение запроса в прозрачный TCP/IP-туннель, обычно чтобы содействовать установлению защищённого SSL-соединения через незашифрованный прокси.[17] [18] См HTTP CONNECT тунель.

PATCH
Метод PATCH применяется к фрагментам ресурса, модифицируя его аналогично.[19]

Все HTTP сервера общего назначения должны выполнять как минимум методы GET и HEAD,[20] и, по возможности OPTIONS метод.

Коды статуса *См. также:* ***Список кодов состояния HTTP***

С HTTP/1.0 и последующих версиях, первая строка ответа http, называется *строка состояния* и включает в себя числовой *код состояния* (например,"404") и текстовое сообщение в виде *фразы* (например, "не найден"). Как агент пользователя будет обрабатывает ответ в первую очередь зависит от кода, а потом от ответа полей заголовка. Пользовательские коды состояния могут быть использованы тогда когда, агент пользователя встречает неизвестный код, он может использовать первую цифру кода, чтобы определить общий класс ответ.[22]

Стандартные ответы *ключевыми словами* носят рекомендательный характер и могут быть заменены "местные аналоги" на усмотрение веб-разработчика. Если код состояния указанной проблемы, агент пользователя может отображать *поводу фразы* , чтобы пользователю предоставить дополнительную информацию о характере проблемы. Стандарт также позволяет пользователю агент пытаться интерпретировать *причину фраза*, хотя это может быть неразумным, поскольку стандарт явно указывает, что коды состояния являются машиночитаемыми и *причина ключевыми словами* не удобочитаемое. Коды состояния HTTP в первую очередь делятся на пять групп для лучшего объяснения запросов и ответов между клиентом и сервером, их называют: информационные 1XX, 2XX успех, 3xx перенаправление, 4xx ошибка клиента и ошибки сервера с кодом 5xx.

Заголовки *См* ***Заголовки HTTP, Список заголовков HTTP***

Заголовки HTTP (англ. *HTTP Headers*) — это строки в HTTP-сообщении, содержащие разделённую двоеточием пару параметр-значение. Формат заголовков соответствует общему формату заголовков текстовых сетевых сообщений ARPA (см. RFC 822). Заголовки должны отделяться от тела сообщения хотя бы одной пустой строкой.
Примеры заголовков:

```
Server: Apache/2.2.11 (Win32) PHP/5.3.0
Last-Modified: Sat, 16 Jan 2010 21:16:42 GMT
Content-Type: text/plain; charset=windows-1251
Content-Language: ru
```

Запрос

Сообщение запроса состоит из следующего:

- Поле запроса (например, *GET /images/logo.png HTTP/1.1,* который запрашивает ресурс под названием `/images/logo.png с сервера`).
- Поля заголовка запроса (например, *Accept-Language: en*).
- Пустую строку.
- Необязательное тело сообщения.

Поле запроса и другие поля заголовка должны заканчиваться с <СГ><если> (т. е. возврат каретки символ, за которым следует перевод строки символов). Пустая строка должна состоять только последовательностью <CR><LF> и никакой другой пробел.[27] в протоколе http/1.1, все поля заголовка, за исключением ***Host*** не являются обязательными.
Запрос строку, содержащую только имя пути принимается серверами для обеспечения совместимости с http клиентами перед http/1.0 спецификации в документе RFC 1945.[28]

Ответ

Ответное сообщение состоит из следующего:

- Строка статуса, которая содержит код состояния и сообщение (например, *HTTP/1.1 200 OK*, который указывает, что запрос клиента удался).
- Ответ полей заголовка (например, *Content-Type: text/html*).
- Пустую строку.
- Необязательное тело сообщения.

Строка состояния и другие поля заголовка должны заканчиваться последовательностью <CR><LF>. Пустая строка должна содержать только <CR><LF> и никакой другой пробел.[27] Это строгие требования по <CR><LF > будет смягчены внутри сообщения тела для постоянного использования другими системами переносов, таких как одиночный <CR> или <LF>.[29]