HTTP

▼ 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».

▼ Метод HTTP запросов

HTTP определяет множество **методов запроса**, которые указывают, какое желаемое действие выполнится для данного ресурса. Несмотря на то, что их названия могут быть существительными, эти методы запроса иногда называются *HTTP глаголами*. Каждый реализует свою семантику, но каждая группа команд разделяет общие свойства: так, методы могут быть

▼ Безопасный

Метод HTTP является **безопасным**, если он не меняет состояние сервера. Другими словами, безопасный метод проводит операции "только чтение" (read-only). Безопасные:GET, HEAD или OPTINS.

Даже если безопасные методы являются по существу "только для чтения", сервер всё равно может сменить своё состояние: например, он может сохранять статистику. Браузеры могут вызывать безопасные методы это позволяет им выполнять некоторые действия, например, предварительная загрузка без риска. Поисковые роботы также полагаются на вызовы безопасных методов.

Правильная реализация безопасного метода - это ответственность **серверного приложения**, потому что сам веб-сервер, будь то Apache, nginx, IIS это соблюсти не сможет.

▼ Идемпотентный

Если повторный идентичный запрос, сделанный один или несколько раз подряд, имеет один и тот же эффект, не изменяющий состояние сервера. Другими словами, идемпотентный метод не должен иметь никаких побочных эффектов (side-effects), кроме сбора статистики или подобных операций. Корректно реализованные методы GET, HEAD, PUL, DELETE

▼ **Кешируемый**

это HTTP-ответы, которые могут быть закешированы, то есть сохранены для дальнейшего восстановления и использования позже, тем самым снижая число запросов к серверу. Не все HTTP-ответы могут быть закешированы. Вот несколько ограничений:

- Метод, используемый в запросе, кешируемый, если это GET или HEAD. Ответ для POST или PATCH запросов может также быть закеширован, если указан признак "свежести" данных и установлен заголовок Content-Location (tn-us)(**Content-Location указывает альтернативное расположение возвращаемых данных. Основное использование заключается в указании URL-адреса ресурса, переданного в результате согласования содержимого.**), но это редко реализуется. Другие методы, такие как PUT и DELETE не кешируемые, и результат их выполнения не кешируется.
- Коды ответа, известные системе кеширования, которые рассматриваются как кешируемые: 200, 203, 204, 206, 300, 301, 404, 405, 410, 414, 501
- Отсутствуют специальные заголовки в ответе, которые предотвращают кеширование: например, Cache-control (используется для задания инструкций кеширования как для запросов, так и для ответов. Инструкции кеширования однонаправленные: заданная инструкция в запросе не подразумевает, что такая же инструкция будет указана в ответе Cache-control)

▼ **Методы**

GET запрашивает представление ресурса. Запросы с использованием этого метода могут только извлекать данные.

HEAD запрашивает ресурс так же, как и метод GET, но без тела ответа.

POST используется для отправки сущностей к определённому ресурсу. Часто вызывает изменение состояния или какие-то побочные эффекты на сервере.

PUT заменяет все текущие представления ресурса данными запроса.

DELETE удаляет указанный ресурс.

CONNECT устанавливает "туннель" к серверу, определённому по ресурсу.

OPTIONS используется для описания параметров соединения с ресурсом.

RACE выполняет вызов возвращаемого тестового сообщения с ресурса.

PATCH используется для частичного изменения ресурса.

▼ **Код ответа HTTP-сервера**

▼ **Код состояния HTTP (***HTTP status code)*

часть первой строки ответа сервера при запросах по протоколу HTTP. Он представляет собой целое трёхразрядное десятичное число. Первая цифра указывает на класс состояния. За кодом ответа обычно следует отделённая пробелом поясняющая фраза на английском языке, которая разъясняет человеку причину именно такого ответа.

Клиент узнаёт по коду ответа о результатах своего запроса и определяет, какие действия ему предпринимать дальше.

▼ **Информационные**

В этот класс выделены коды, информирующие о процессе передачи. При работе через протокол версии 1.0 сообщения с такими кодами должны игнорироваться. В версии 1.1 клиент должен быть готов принять этот класс сообщений как обычный ответ, но серверу отправлять что-либо не нужно. Сами сообщения от сервера содержат только стартовую строку ответа и, если требуется, несколько специфичных для ответа полей заголовка.

- 100 Continue — сервер удовлетворён начальными сведениями о запросе, клиент может продолжать пересылать заголовки. Появился в HTTP/1.1.
- 101 Switching Protocols — сервер выполняет требование клиента и переключает протоколы в соответствии с указанием, данным в поле заголовка Upgrade. Сервер отправляет заголовок ответа Upgrade, указывая протокол, на который он переключился. Появился в HTTP/1.1.
- 102 Processing — запрос принят, но на его обработку понадобится длительное время. Используется сервером, чтобы клиент не разорвал соединение из-за превышения времени ожидания. Клиент при получении такого ответа должен сбросить таймер и дожидаться следующей команды в обычном режиме. Появился в *WebDAV*.
- 103 Early Hints — используется для раннего возврата части заголовков, когда заголовки полного ответа не могут быть быстро сформированы.

▼ **Успех**

Сообщения данного класса информируют о случаях успешного принятия и обработки запроса клиента. В зависимости от статуса сервер может ещё передать заголовки и тело сообщения.

- 200 OK — успешный запрос. Если клиентом были запрошены какие-либо данные, то они находятся в заголовке и/или теле сообщения. Появился в HTTP/1.0.
- 201 Created — в результате успешного выполнения запроса был создан новый ресурс. Сервер может указать адреса (их может быть несколько) созданного ресурса в теле ответа, при этом предпочтительный адрес указывается в заголовке Location. Серверу рекомендуется указывать в теле ответа характеристики созданного ресурса и его адреса, формат тела ответа определяется заголовком Content-tyre. При обработке запроса новый ресурс должен быть создан до отправки ответа клиенту, иначе следует использовать ответ с кодом 202. Появился в HTTP/1.0.
- 202 Accepted — запрос был принят на обработку, но она не завершена. Клиенту не обязательно дожидаться окончательной

передачи сообщения, так как может быть начат очень долгий процесс. Появился в HTTP/1.0.

- 203 Non-Authoritative Information — аналогично ответу 200, но в этом случае передаваемая информация была взята не из первичного источника (резервной копии, другого сервера и т. д.) и поэтому может быть неактуальной. Появился в HTTP/1.1.
- 204 No Content — сервер успешно обработал запрос, но в ответе были переданы только заголовки без тела сообщения. Клиент не должен обновлять содержимое документа, но может применить к нему полученные методанные. Появился в HTTP/1.0.
- 205 Reset Content — сервер обязывает клиента сбросить введённые пользователем данные. Тела сообщения сервер при этом не передаёт и документ обновлять не обязательно. Появился в HTTP/1.1.
- 206 Partial Content — сервер удачно выполнил частичный GET-запрос, возвратив только часть сообщения. В заголовке Content-Range сервер указывает байтовый диапозоны содержимого. Особое внимание при работе с подобными ответами следует уделить кэшированию. Появился в HTTP/1.1.
- 207 Multi-Status — сервер передаёт результаты выполнения сразу нескольких независимых операций. Они помещаются в само тело сообщения в видеXML документа с объектом multistatus. Не рекомендуется размещать в этом объекте статусы из серии 1xxx из-за бессмысленности и избыточности. Появился вWebDAV.
- 208 Already Reported — члены привязки DAV уже были перечислены в предыдущей части (multistatus) ответа и не включаются снова.
- 226 IM Used — заголовок A-IM от клиента был успешно принят и сервер возвращает содержимое с учётом указанных параметров.

▼ **Перенаправление**

Коды этого класса сообщают клиенту, что для успешного выполнения операции необходимо сделать другой запрос, как правило, по другому URL. Из данного класса пять кодов 301, 302, 303, 305, 307 относятся непосредственно к перенаправлениям.

- 300 Multiple Choices — по указанному URI существует несколько вариантов предоставления ресурса по типу MIME, по языку или по другим характеристикам. Сервер передаёт с сообщением список альтернатив, давая возможность сделать выбор клиенту автоматически или пользователю. Появился в HTTP/1.0.
- 301 Moved Permanently— запрошенный документ был окончательно перенесен на новый URI, указанный в поле Location заголовка. Некоторые клиенты некорректно ведут себя при обработке данного кода. Появился в HTTP/1.0.
- 302 Found, 302 Moved Temporarily — запрошенный документ временно доступен по другому URI, указанному в заголовке в поле Location.
- 303 See Other — документ по запрошенному URI нужно запросить по адресу в поле Location заголовка с использованием метода Get несмотря даже на то, что первый запрашивался иным методом. Этот код был введён вместе с кодом 307 для избежания неоднозначности, чтобы сервер был уверен, что следующий ресурс будет запрошен методом GET. Например, на веб-странице есть поле ввода текста для быстрого перехода и поиска. После ввода данных браузер делает запрос методом POST, включая в тело сообщения введённый текст. Если обнаружен документ с введённым названием, то сервер отвечает кодом 303, указав в заголовке Location его постоянный адрес. Тогда браузер гарантировано его запросит методом GET для получения содержимого. В противном случае сервер просто вернёт клиенту страницу с результатами поиска. Введено в HTTP/1.1.
- 304 Not Modified — сервер возвращает такой код, если клиент запросил документ методом GET, использовал заголовок If-modified-since или If-None-Match и документ не изменился с указанного момента. При этом сообщение сервера не должно содержать тела. Появился в HTTP/1.0.
- 305 Use Proxy — запрос к запрашиваемому ресурсу должен осуществляться через Прокси-Сервис, URI которого указан в поле Location заголовка. Данный код ответа могут использовать только исходные HTTP-сервера (не прокси). Введено в HTTP/1.1.

- 306 (зарезервировано) — использовавшийся в ранних версиях спецификации код ответа, в настоящий момент зарезервирован.
- 307 Temporary Redirect — запрашиваемый ресурс на короткое время доступен по другому URI, указанный в поле Location заголовка. Метод запроса (GET/POST) менять не разрешается. Например, POST-запрос должен быть отправлен по новому URI тем же методом POST. Этот код был введён вместе с 303-м вместо 302-го для избежания неоднозначности.
- 308 Permanent Redirect — запрашиваемый ресурс был окончательно перенесен на новый URI, указанный в поле Location заголовка. Метод запроса (GET/POST) менять не разрешается. Например, POST-запрос должен быть отправлен по новому URI тем же методом POST. Этот код был введён вместо 301-го для избежания неоднозначности.

▼ **Ошибка клиента**

Класс кодов 4xx предназначен для указания ошибок со стороны клиента. При использовании всех методов, кроме Head, сервер должен вернуть в теле сообщения гипертекстовое пояснение для пользователя.

- 400 Bad Request — сервер обнаружил в запросе клиента синтаксическую ошибку. Появился в HTTP/1.0.
- 401 Unauthorized — для доступа к запрашиваемому ресурсу требуется аунтефикация. Для доступа к запрашиваемому ресурсу клиент должен представиться, послав запрос, включив при этом в заголовок сообщения поле Authorization с требуемыми для аутентификации данными. Если запрос уже включает данные для авторизации, ответ 401 означает, что в авторизации с ними отказано.
- 402 Payment Required — предполагается использовать в будущем. В настоящий момент не используется. Этот код предусмотрен для платных пользовательских сервисов.
- 403 Forbidden — сервер понял запрос, но он отказывается его выполнять из-за ограничений в доступе для клиента к указанному ресурсу. Иными словами, клиент не уполномочен совершать операции с запрошенным ресурсом. Если для доступа к ресурсу

требуется аутентификация средствами HTTP, то сервер вернёт ответ 401, или 407 при использовании прокси. В противном случае ограничения были заданы администратором сервера или разработчиком веб-приложения и могут быть любыми в зависимости от возможностей используемого ПО. В любом случае серверу следует сообщить причины отказа в обработке запроса.

- 404 Not Found— ошибка в написании адреса Web-страницы. Сервер понял запрос, но не нашёл соответствующего ресурса по указанному URL. Если серверу известно, что по этому адресу был документ, то ему желательно использовать код 410.
- 405 Method Not Allowed — указанный клиентом метод нельзя применить к текущему ресурсу. В ответе сервер должен указать доступные методы в заголовке ALLOW, разделив их запятой. Эту ошибку сервер должен возвращать, если метод ему известен, но он не применим именно к указанному в запросе ресурсу, если же указанный метод не применим на всём сервере, то клиенту нужно вернуть код 501(Not Implemented). Появился в HTTP/1.1.
- 406 Not Acceptable — запрошенный URI не может удовлетворить переданным в заголовке характеристикам. Если метод был не HEAD, то сервер должен вернуть список допустимых характеристик для данного ресурса. Появился в HTTP/1.1.
- 407 Proxy Authentication Required — ответ аналогичен коду 401 за исключением того, что аутентификация производится для прокси-сервера. Появился в HTTP/1.1.
- 408 Request Timeout — время ожидания сервером передачи от клиента истекло. Клиент может повторить аналогичный предыдущему запрос в любое время. . Появился в HTTP/1.1.
- 409 Conflict — запрос не может быть выполнен из-за конфликтного обращения к ресурсу. Такое возможно, например, когда два клиента пытаются изменить ресурс с помощью метода PUT. Появился в HTTP/1.1.
- 410 Gone — такой ответ сервер посылает, если ресурс раньше был по указанному URL, но был удалён и теперь недоступен. Серверу в этом случае неизвестно и местоположение альтернативного документа (например копии). Появился в HTTP/1.1.

- 411 Length Required — для указанного ресурса клиент должен указать Content-Length в заголовке запроса. Без указания этого поля не стоит делать повторную попытку запроса к серверу по данному URI. Появился в HTTP/1.1.
- 412 Precondition Failed — возвращается, если ни одно из условных полей заголовка запроса не было выполнено. Появился в HTTP/1.1.
- 413 Payload Too Large — возвращается в случае, если сервер отказывается обработать запрос по причине слишком большого размера тела запроса. Сервер может закрыть соединение, чтобы прекратить дальнейшую передачу запроса. .
- 414 URI Too Long — сервер не может обработать запрос из-за слишком длинного указанного URI. Появился в HTTP/1.1. Ранее назывался «Request-URI Too Long».
- 415 Unsupported Media Type — по каким-то причинам сервер отказывается работать с указанным типом данных при данном методе. Появился в HTTP/1.1.
- 416 Range Not Satisfiable — в поле Range заголовка запроса был указан диапазон за пределами ресурса и отсутствует поле If-Range. Если клиент передал байтовый диапазон, то сервер может вернуть реальный размер в поле Content-Range заголовка. Данный ответ не следует использовать при передаче типа multipart/byteranges.
- 417 Expectation Failed — по каким-то причинам сервер не может удовлетворить значению поля Expect заголовка запроса. Введено в RFC 2616 (обновление HTTP/1.1).
- 418 I’m a teapot — Этот код был введен в 1998 году как одна из традиционных первоапрельских шуток IETF в RFC 2324, Hyper Text Coffee Pot Control Protocol. Не ожидается, что данный код будет поддерживаться реальными серверами.
- 419 Authentication Timeout (not in RFC 2616) — Этого кода нет. используется в качестве альтернативы коду 401, которые прошли проверку подлинности, но лишены доступа к определенным ресурсам сервера.
- 421 Misdirected Request — запрос был перенаправлен на сервер, не способный дать ответ.

- 422 Unprocessable Entity — сервер успешно принял запрос, может работать с указанным видом данных, однако имеется какая-то логическая ошибка, из-за которой невозможно произвести операцию над ресурсом. Введено в *WebDAV*.
- 423 Locked — целевой ресурс из запроса заблокирован от применения к нему указанного метода. Введено в WebDAV.
- 424 Failed Dependency — реализация текущего запроса может зависеть от успешности выполнения другой операции. Если она не выполнена и из-за этого нельзя выполнить текущий запрос, то сервер вернёт этот код. Введено в *WebDAV*.
- 425 Too Early — сервер не готов принять риски обработки "ранней информации". Введено в RFC 8470 для защиты от атак повторения при использовании 0-RTT в TLS 1.3.
- 426 Upgrade Required — сервер указывает клиенту на необходимость обновить протокол. Заголовок ответа должен содержать правильно сформированные поля Upgrade и Connection. Введено в RFC 2817 для возможности перехода к TLS посредством HTTP.
- 428 Precondition Required — сервер указывает клиенту на необходимость использования в запросе заголовков условий, наподобие If-Match.
- 429 Too Many Requests — клиент попытался отправить слишком много запросов за короткое время, что может указывать, например, на попытку DDoS-атаки. Может сопровождаться заголовком Retry-After, указывающим, через какое время можно повторить запрос.
- 431 Request Header Fields Too Large — Превышена допустимая длина заголовков. Сервер не обязан отвечать этим кодом, вместо этого он может просто сбросить соединение.
- 434 Requested host unavailable — Запрашиваемый адрес недоступен.
- 449 Retry With — возвращается сервером, если для обработки запроса от клиента поступило недостаточно информации.
- 451 Unavailable For Legal Reasons— доступ к ресурсу закрыт по юридическим причинам, например, по требованию органов

государственной власти или по требованию правообладателя в случае нарушения авторских прав.

- 499 Client Closed Request — нестандартный код, предложенный и используемый для случаев, когда клиент закрыл соединение, пока обрабатывается запрос.

▼ **Ошибка сервера**

Коды 5xx выделены под случаи необработанных исключений при выполнении операций на стороне сервера. Для всех ситуаций, кроме использования метода HEAD, сервер должен включать в тело сообщения объяснение, которое клиент отобразит пользователю.

- 500 Internal Server Error — любая внутренняя ошибка сервера, которая не входит в рамки остальных ошибок класса.
- 501 Not Implemented — сервер не поддерживает возможностей, необходимых для обработки запроса. Типичный ответ для случаев, когда сервер не понимает указанный в запросе метод.
- 502 Bad Gateway — сервер, выступая в роли шлюза или прокси-сервера, получил недействительное ответное сообщение от вышестоящего сервера. Появился в HTTP/1.0.
- 503 Service Unavailable — сервер временно не имеет возможности обрабатывать запросы по техническим причинам (обслуживание, перегрузка и прочее). В поле Retry-After заголовка сервер может указать время, через которое клиенту рекомендуется повторить запрос.
- 504 Gateway Timeout — сервер в роли шлюза или прокси-сервера не дождался ответа от вышестоящего сервера для завершения текущего запроса. Появился в HTTP/1.1.
- 505 HTTP Version Not Supported — сервер не поддерживает или отказывается поддерживать указанную в запросе версию протокола HTTP. Появился в HTTP/1.1.
- 506 Variant Also Negotiates — в результате ошибочной конфигурации выбранный вариант указывает сам на себя, из-за чего процесс связывания прерывается. Экспериментальное.

- 507 Insufficient Storage — не хватает места для выполнения текущего запроса. Проблема может быть временной. Введено в *WebDAV*.
- 508 Loop Detected — операция отменена, т.к. сервер обнаружил бесконечный цикл при обработке запроса без ограничения глубины. Введено в *WebDAV*.
- 508 Resource Limit Reached — вариант ошибки 508 в CloudLinux, возникающий при исчерпании лимитов хостинга.
- 509 Bandwidth Limit Exceeded — используется при превышении веб-площадкой отведённого ей ограничения на потребление трафика. В данном случае владельцу площадки следует обратиться к своему хостинг-провайдеру.
- 510 Not Extended — на сервере отсутствует расширение, которое желает использовать клиент. Сервер может дополнительно передать информацию о доступных ему расширениях.
- 511 Network Authentication Required — этот ответ посылается не сервером, которому был предназначен запрос, а сервером-посредником — например, сервером провайдера — в случае, если клиент должен сначала авторизоваться в сети, например, ввести пароль для платной точки доступа к Интернету. Предполагается, что в теле ответа будет возвращена Web-форма авторизации или перенаправление на неё.
- 520 Unknown Error, возникает когда сервер CDN не смог обработать ошибку веб-сервера; нестандартный код CloudFlare.
- 521 Web Server Is Down, возникает когда подключения CDN отклоняются веб-сервером; нестандартный код CloudFlare.
- 522 Connection Timed Out, возникает когда CDN не удалось подключиться к веб-серверу; нестандартный код CloudFlare.
- 523 Origin Is Unreachable, возникает когда веб-сервер недостижим; нестандартный код CloudFlare.
- 524 A Timeout Occurred, возникает при истечении тайм-аута подключения между сервером CDN и веб-сервером; нестандартный код CloudFlare.

- 525 SSL Handshake Failed, возникает при ошибке рукопожатия SSL между сервером CDN и веб-сервером; нестандартный код CloudFlare.
- 526 Invalid SSL Certificate, возникает когда не удаётся подтвердить сертификат шифрования веб-сервера; нестандартный код CloudFlare.