Manual QA course

Lecture 21. HTTP-протокол

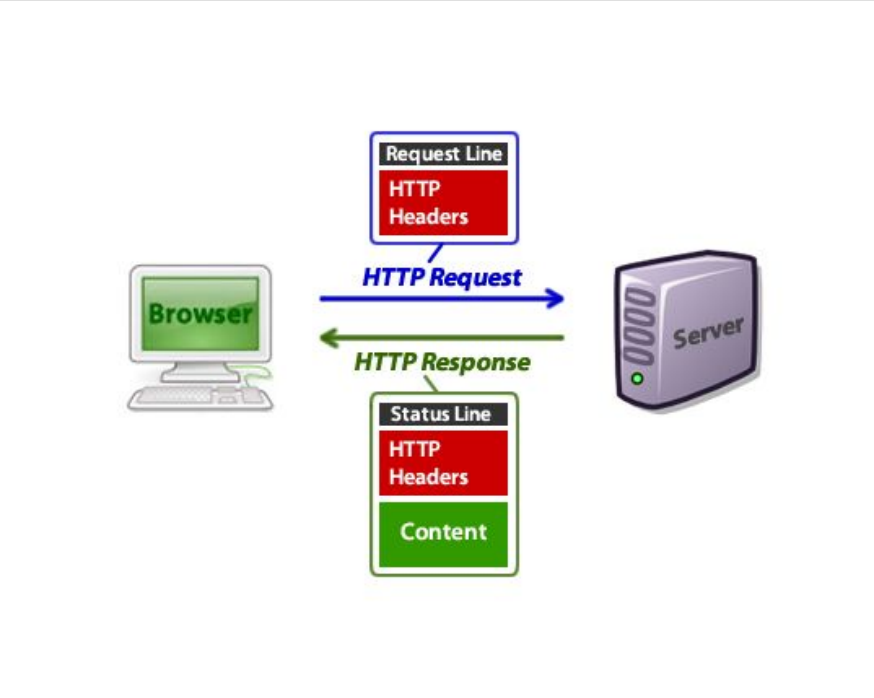
Дорофеев Максим

hillel
компьютерная
школа

HTTP

HyperText Transfer Protocol — «протокол передачи гипертекста» — протокол прикладного уровня передачи данных (изначально — в виде гипертекстовых документов в формате HTML, в настоящий момент используется для передачи произвольных данных). Основой HTTP является технология «клиент-сервер», то есть предполагается существование потребителей (клиентов), которые инициируют соединение и посылают запрос, и поставщиков (серверов), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом

HTTP


Request Line
HTTP Headers
HTTP Request
Browser
Server
HTTP Response
Status Line
HTTP Headers
Content

HTTP

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. (В частности для этого используется HTTP-заголовок.) Именно благодаря возможности указания способа кодирования сообщения клиент и сервер могут обмениваться двоичными данными, хотя данный протокол является текстовым

URI.

URI - отвечает на вопрос: “Где и как найти что - то?”

Пример:

```
http://example.com/just/some/long/path/path
```

HTTP. История

HTTP/0.9

HTTP был предложен в марте 1991 года Тимом Бернерсом-Ли, работавшим тогда в CERN, как механизм для доступа к документам в Интернете и облегчения навигации посредством использования гипертекста. Самая ранняя версия протокола HTTP/0.9 была впервые опубликована в январе 1992 г. (хотя реализация датируется 1990 годом). Спецификация протокола привела к упорядочению правил взаимодействия между клиентами и серверами HTTP, а также четкому разделению функций между этими двумя компонентами. Были задокументированы основные синтаксические и семантические положения.

HTTP/1.0

В мае 1996 года для практической реализации HTTP был выпущен информационный документ RFC 1945, что послужило основой для реализации большинства компонентов HTTP/1.0.

HTTP/1.1

Текущая версия протокола, принята в июне 1999 года. Новым в этой версии был режим «постоянного соединения»: TCP-соединение может оставаться открытым после отправки ответа на запрос, что позволяет посылать несколько запросов за одно соединение. Клиент теперь обязан посылать информацию об имени хоста, к которому он обращается, что сделало возможной более простую организацию виртуального хостинга.

HTTP/2

11 февраля 2015 года опубликованы финальные версии черновика следующей версии протокола. В отличие от предыдущих версий, протокол HTTP/2 является бинарным. Среди ключевых особенностей мультиплексирование запросов, расстановка приоритетов для запросов, сжатия заголовков, загрузка нескольких элементов параллельно, посредством одного TCP соединения, поддержка проактивных push-уведомлений со стороны сервера.

RFC.

Документы, содержащие технические спецификации и стандарты, широко применяемые во всемирной сети интернет.

Существует более 5000 документов.

HTTP. Структура

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

Стартовая строка (англ. Starting line) — определяет тип сообщения;
Заголовки (англ. Headers) — характеризуют тело сообщения, параметры передачи и прочие сведения;
Тело сообщения (англ. 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. Исключением является версия 0.9 протокола, у которой сообщение запроса содержит только стартовую строку, а сообщения ответа только тело сообщения.

Для версии протокола 1.1 сообщение запроса обязательно должно содержать заголовок Host.

HTTP. Структура. Стартовая строка запроса

Метод URI HTTP/Версия

HTTP. Структура. Стартовая строка запроса

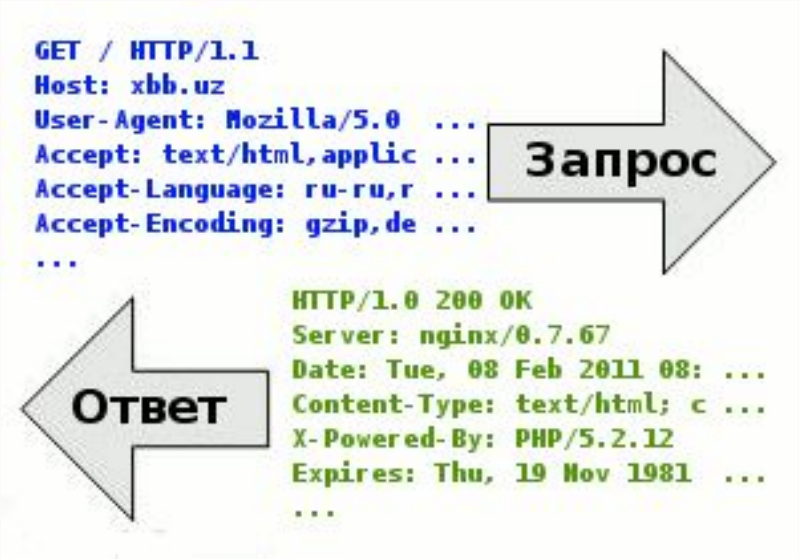
Метод (англ. Method) — название запроса, одно слово заглавными буквами
URI определяет путь к запрашиваемому документу
Версия (англ. Version) — пара разделенных точкой цифр. Например: 1.0

HTTP. Структура. Стартовая строка ответа

HTTP/Версия Код-Состояния Пояснение

Версия — пара разделенных точкой цифр как в запросе.
Код состояния (англ. Status Code) — три цифры. По коду состояния определяется дальнейшее содержимое сообщения и поведение клиента.
Пояснение (англ. Reason Phrase) — текстовое короткое пояснение к коду ответа для пользователя. Никак не влияет на сообщение и является необязательным.

HTTP. Структура



HTTP. Методы

Последовательность из любых символов, кроме управляющих и разделителей, указывающая на основную операцию над ресурсом. Метод представляет собой короткое английское слово, записаное заглавными буквами. Название метода чувствительно к регистру

HTTP. Методы

Каждый сервер обязан поддерживать как минимум методы GET и HEAD. Если сервер не распознал указанный клиентом метод, то он должен вернуть статус 501 (Not Implemented). Если серверу метод известен, но он неприменим к конкретному ресурсу, то возвращается сообщение с кодом 405 (Method Not Allowed). В обоих случаях серверу следует включить в сообщение ответа заголовок Allow со списком поддерживаемых методов.

Кроме методов GET и HEAD, часто применяется метод POST.

HTTP. Методы

OPTIONS
GET
HEAD
POST
PUT
PATCH
DELETE
TRACE
CONNECT

HTTP. Методы

OPTIONS
Используется для определения возможностей веб-сервера или параметров соединения для конкретного ресурса. В ответ серверу следует включить заголовок Allow со списком поддерживаемых методов. Также в заголовке ответа может включаться информация о поддерживаемых расширениях.
Результат выполнения этого метода не кэшируется

HTTP. Методы

GET
Используется для запроса содержимого указанного ресурса. С помощью метода GET можно также начать какой-либо процесс. В этом случае в тело ответного сообщения следует включить информацию о ходе выполнения процесса.

Клиент может передавать параметры выполнения запроса в URI целевого ресурса после символа «?»:
GET /path/resource?param1=value1&param2=value2 HTTP/1.1

Согласно стандарту HTTP, запросы типа GET считаются **идемпотентными**

HTTP. Методы

HEAD
Аналогичен методу GET, за исключением того, что в ответе сервера отсутствует тело. Запрос HEAD обычно применяется для извлечения метаданных, проверки наличия ресурса (валидация URL) и чтобы узнать, не изменился ли он с момента последнего обращения.

Заголовки ответа могут кэшироваться. При несовпадении метаданных ресурса с соответствующей информацией в кэше копия ресурса помечается как устаревшая

HTTP. Методы

POST
Применяется для передачи пользовательских данных заданному ресурсу. Например, в блогах посетители обычно могут вводить свои комментарии к записям в HTML-форму, после чего они передаются серверу методом POST и он помещает их на страницу. При этом передаваемые данные (в примере с блогами — текст комментария) включаются в тело запроса. Аналогично с помощью метода POST обычно загружаются файлы на сервер.

В отличие от метода GET, метод POST не считается идемпотентным, 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).

При результате выполнения 200 (Ok) в тело ответа следует включить сообщение об итоге выполнения запроса. Если был создан ресурс, то серверу следует вернуть ответ 201 (Created) с указанием URI нового ресурса в заголовке Location.

Сообщение ответа сервера на выполнение метода POST не кэшируется

HTTP. Методы

PUT
Применяется для загрузки содержимого запроса на указанный в запросе URI. Если по заданному URI не существует ресурс, то сервер создаёт его и возвращает статус 201 (Created). Если же был изменён ресурс, то сервер возвращает 200 (Ok) или 204 (No Content). Сервер не должен игнорировать некорректные заголовки Content-*, передаваемые клиентом вместе с сообщением. Если какой-то из этих заголовков не может быть распознан или не допустим при текущих условиях, то необходимо вернуть код ошибки 501 (Not Implemented).

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.

Сообщения ответов сервера на метод PUT не кэшируются

HTTP. Методы

PATCH

Аналогично PUT, но применяется только к фрагменту ресурса

HTTP. Методы

DELETE

Удаляет указанный ресурс

HTTP. Методы

TRACE

Возвращает полученный запрос так, что клиент может увидеть, какую информацию промежуточные серверы добавляют или изменяют в запросе

HTTP. Методы

CONNECT
Преобразует соединение запроса в прозрачный TCP/IP-туннель, обычно чтобы содействовать установлению защищённого SSL-соединения через нешифрованный прокси

HTTP. Коды состояния

1xx Informational («Информационный»)
В этот класс выделены коды, информирующие о процессе передачи. В HTTP/1.0 сообщения с такими кодами должны игнорироваться. В HTTP/1.1 клиент должен быть готов принять этот класс сообщений как обычный ответ, но ничего отправлять серверу не нужно. Сами сообщения от сервера содержат только стартовую строку ответа и, если требуется, несколько специфичных для ответа полей заголовка. Прокси-серверы подобные сообщения должны отправлять дальше от сервера к клиенту.

2xx Success («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.

3xx Redirection («Перенаправление»)
Коды класса 3xx сообщают клиенту что для успешного выполнения операции необходимо сделать другой запрос (как правило по другому URI). Из данного класса пять кодов 301, 302, 303, 305 и 307 относятся непосредственно к перенаправлениям (редирект). Адрес, по которому клиенту следует произвести запрос, сервер указывает в заголовке Location. При этом допускается использование фрагментов в целевом URI.

4xx Client Error («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.

5xx Server Error («Ошибка сервера»)
Коды 5xx выделены под случаи неудачного выполнения операции по вине сервера. Для всех ситуаций, кроме использования метода HEAD, сервер должен включать в тело сообщения объяснение, которое клиент отобразит пользователю.

HTTP. Коды состояния

1xx: Informational (информационные)
100 Continue («продолжай»)
101 Switching Protocols («переключение протоколов»)
102 Processing («идёт обработка»)

HTTP. Коды состояния

2xx: Success (успешно)

- 200 OK («хорошо»)
- 201 Created («создано»)
- 202 Accepted («принято»)
- 203 Non-Authoritative Information («информация не авторитетна»)
- 204 No Content («нет содержимого»)
- 205 Reset Content («сбросить содержимое»)
- 206 Partial Content («частичное содержимое»)
- 207 Multi-Status («многостатусный»)

HTTP. Коды состояния

3xx: Redirection (перенаправление)
300 Multiple Choices («множество выборов»)
301 Moved Permanently («перемещено навсегда»)
302 Moved Temporarily («перемещено временно»)
302 Found («найдено»)
303 See Other (смотреть другое)
304 Not Modified (не изменялось)
305 Use Proxy («использовать прокси»)
306 — зарезервировано (код использовался только в ранних спецификациях)
307 Temporary Redirect («временное перенаправление»)

HTTP. Коды состояния

4xx: Client Error (ошибка клиента)
400 Bad Request («плохой, неверный запрос»)
401 Unauthorized («не авторизован»)
402 Payment Required («необходима оплата»)
403 Forbidden («запрещено»)
404 Not Found («не найдено»)
405 Method Not Allowed («метод не поддерживается»)
406 Not Acceptable («неприемлемо»)
407 Proxy Authentication Required («необходима аутентификация прокси»)
408 Request Timeout («истекло время ожидания»)
409 Conflict («конфликт»)
410 Gone («удалён»)
411 Length Required («необходима длина»)
412 Precondition Failed («условие ложно»)

HTTP. Коды состояния

4xx: Client Error (ошибка клиента)
413 Request Entity Too Large («размер запроса слишком велик»)
414 Request-URI Too Large («запрашиваемый URI слишком длинный»)
415 Unsupported Media Type («неподдерживаемый тип данных»)
416 Requested Range Not Satisfiable («запрашиваемый диапазон не достижим»)
417 Expectation Failed («ожидаемое неприемлемо»)
422 Unprocessable Entity («необрабатываемый экземпляр»)
423 Locked («заблокировано»)
424 Failed Dependency («невыполненная зависимость»)
425 Unordered Collection («неупорядоченный набор»)
426 Upgrade Required («необходимо обновление»)
428 Precondition Required («необходимо предусловие»)
429 Too Many Requests («слишком много запросов»)
431 Request Header Fields Too Large («поля заголовка запроса слишком большие»)
434 Requested host unavailable. («Запрашиваемый адрес недоступен»)
444 Закрывает соединение без передачи заголовка ответа. Нестандартный код
449 Retry With («повторить с»)
451 Unavailable For Legal Reasons («недоступно по юридическим причинам»)

HTTP. Коды состояния

5xx: Server Error (ошибка сервера)
500 Internal Server Error («внутренняя ошибка сервера»)
501 Not Implemented («не реализовано»)
502 Bad Gateway («плохой, ошибочный шлюз»)
503 Service Unavailable («сервис недоступен»)
504 Gateway Timeout («шлюз не отвечает»)
505 HTTP Version Not Supported («версия HTTP не поддерживается»)
506 Variant Also Negotiates («вариант тоже проводит согласование»)
507 Insufficient Storage («переполнение хранилища»)
508 Loop Detected («обнаружена петля»)
509 Bandwidth Limit Exceeded («исчерпана пропускная ширина канала»)
510 Not Extended («не расширено»)
511 Network Authentication Required («требуется сетевая аутентификация»)

HTTP. Заголовки

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

HTTP. Заголовки

```
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
```

HTTP. Заголовки

Все заголовки разделяются на четыре основных группы:

- General Headers («Основные заголовки») — могут включаться в любое сообщение клиента и сервера.
- Request Headers («Заголовки запроса») — используются только в запросах клиента.
- Response Headers («Заголовки ответа») — только для ответов от сервера.
- Entity Headers («Заголовки сущности») — сопровождают каждую сущность сообщения

HTTP. Заголовки

Заголовок	Описание	Пример
Accept	Список допустимых форматов ресурса	Accept: text/plain
Accept-Charset	Перечень поддерживаемых кодировок для предоставления пользователю	Accept-Charset: utf-8
Allow	Список поддерживаемых методов	Allow: OPTIONS, GET, HEAD
Referer	URI ресурса, после которого клиент сделал текущий запрос	Referer: http://en.wikipedia.org/wiki/Main_Page
User-Agent	Список названий и версий клиента и его компонентов с комментариями	User-Agent: Mozilla/5.0 (X11; Linux i686; rv:2.0.1) Gecko/20100101 Firefox/4.0.1

HTTP vs HTTPs

HyperText Transfer Protocol Secure — расширение протокола HTTP, поддерживающее шифрование. Данные, передаваемые по протоколу HTTPS, «упаковываются» в криптографический протокол SSL или TLS.
В отличие от HTTP, для HTTPS по умолчанию используется TCP-порт 443

HTTP vs HTTPs

HTTPS не является отдельным протоколом. Это обычный HTTP, работающий через шифрованные транспортные механизмы SSL и TLS. Он обеспечивает защиту от атак, основанных на прослушивании сетевого соединения — от снифферских атак и атак типа man-in-the-middle, при условии, что будут использоваться шифрующие средства и сертификат сервера проверен и ему доверяют

HTTP vs HTTPs

По умолчанию HTTPS URL использует 443 TCP-порт (для незащищённого HTTP — 80). Чтобы подготовить веб-сервер для обработки https-соединений, администратор должен получить и установить в систему сертификат для этого веб-сервера. Сертификат состоит из 2 частей (2 ключей) — public и private. Public-часть сертификата используется для зашифровывания трафика от клиента к серверу в защищённом соединении, private-часть — для расшифровывания полученного от клиента зашифрованного трафика на сервере. После того как пара ключей приватный/публичный сгенерированы, на основе публичного ключа формируется запрос на сертификат в Центр сертификации, в ответ на который ЦС высылает подписанный сертификат. ЦС при подписании проверяет клиента, что позволяет ему гарантировать, что держатель сертификата является тем, за кого себя выдаёт (обычно это платная услуга)

HTTP vs HTTPs

Традиционно на одном IP-адресе может работать только один HTTPS сайт. Для работы нескольких HTTPS-сайтов с различными сертификатами применяется расширение TLS под названием Server Name Indication (SNI)

Вопросы и ответы