

# HTTP

## HyperText Transfer Protocol

RFC 1945, RFC 2616

# HTTP - универсальный протокол прикладного уровня

Прикладной протокол работает поверх HTTP.

WEB-серверы управляют транспортным уровнем  
и мультиплексированием HTTP-запросов.

Легко расширяется.

НТТР используется также в качестве базового протокола для коммуникации пользовательских агентов с прокси-серверами и другими системами Интернет, в том числе и использующие протоколы SMTP, NNTP, FTP, Gopher, XMPP и многих других.

# URI – Uniform Resource Identifier

Центральным объектом в HTTP является ресурс, на который указывает URI в запросе клиента. Обычно такими ресурсами являются хранящиеся на сервере файлы.

# URL - Uniform Resource Locator

Изначально URL предназначался для обозначения мест расположения ресурсов (чаще всего файлов) во Всемирной паутине. Сейчас URL применяется для обозначения адресов почти всех ресурсов Интернета. Стандарт URL закреплён в документе RFC 1738

# URL – URI – URN

- URL — это URI, который, помимо идентификации ресурса, предоставляет ещё и информацию о местонахождении этого ресурса
- URN (Uniform Resource Name) — это URI, который только идентифицирует ресурс в определённом пространстве имён (и, соответственно, в определённом контексте), но не указывает его местонахождения.

URN `urn:ISBN:0-395-36341-1` — это URI, который указывает на ресурс (книгу) 0-395-36341-1 в пространстве имён ISBN, но, в отличие от URL, URN не указывает на местонахождение этого ресурса: в нём не сказано, в каком магазине её можно купить, или на каком сайте скачать. Впрочем, в последнее время появилась тенденция говорить просто URI о любой строке-идентификаторе, без дальнейших уточнений.

# Структура URL

**<схема>://<логин>:<пароль>@<хост>:<порт>/  
<URL-путь>?<параметры>#<якорь>**

URL-путь - уточняющая информация о месте нахождения ресурса; зависит от протокола.

Параметры - строка запроса с передаваемыми на сервер (методом GET) параметрами. Разделитель параметров — знак **&**.

Якорь - идентификатор «якоря», ссылающегося на некоторую часть (раздел) открываемого документа.

# НТТР - протокол без памяти

Протокол не хранит информацию о предыдущих запросах клиентов и ответах сервера.

Компоненты, использующие НТТР, могут самостоятельно осуществлять сохранение информации о состоянии, связанной с последними запросами и ответами.



# Категории ПО НТТР

- Серверы - поставщики услуг хранения и обработки информации (обработка запросов).
- Клиенты - конечные потребители услуг сервера (отправка запросов).
- Прокси-серверы для поддержки работы транспортных служб.

# HTTP-сеанс

"Классическая" схема HTTP-сеанса выглядит так.

1. Установление ТСР-соединения.
2. Запрос клиента.
3. Ответ сервера.
4. Разрыв ТСР-соединения.

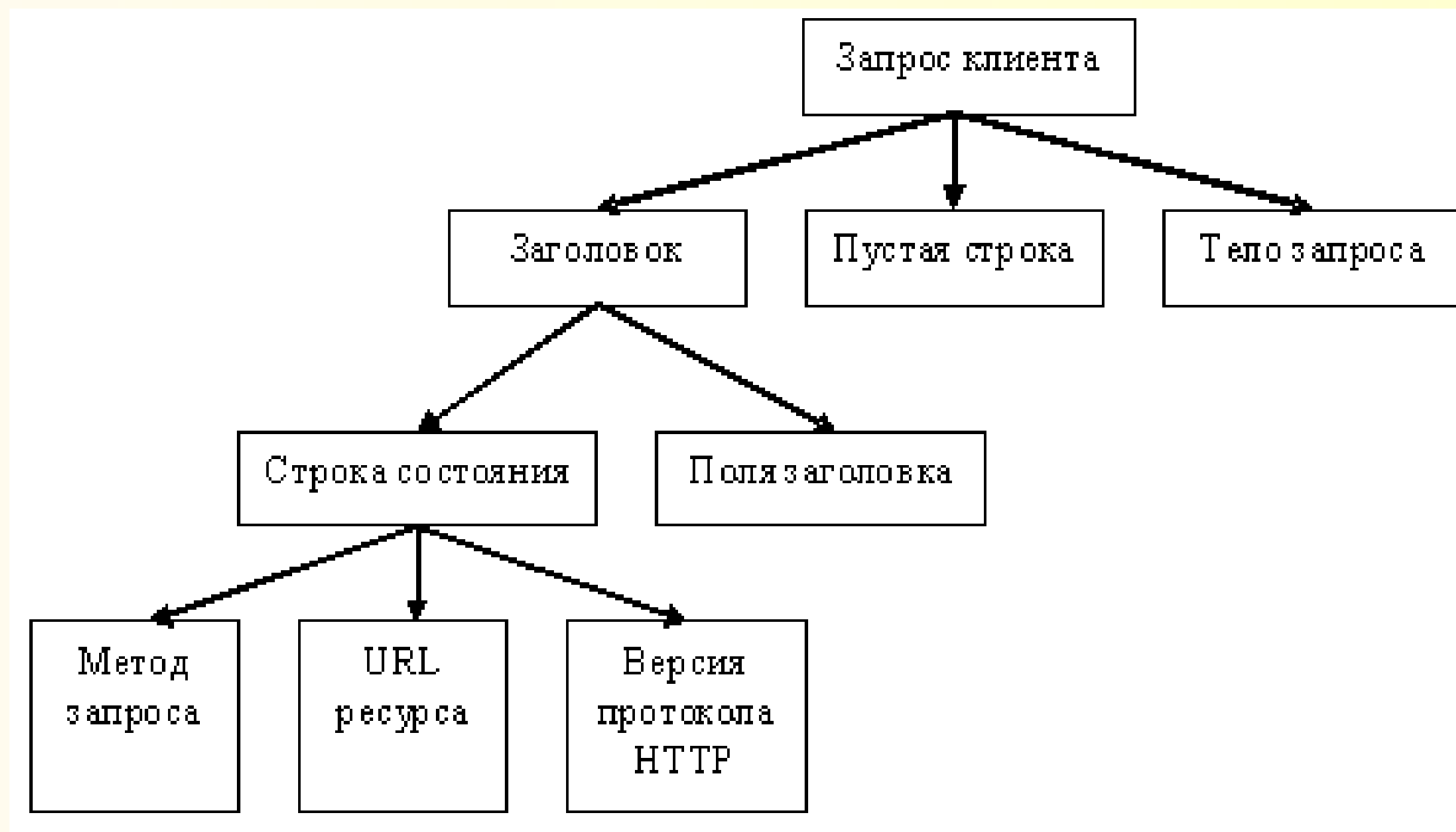
```
telnet ftp.mgts.by 80
C: GET /README HTTP/1.1
C: Host: ftp.mgts.by
C: User-Agent: telnet
C:
S: HTTP/1.1 200 OK
S: Server: nginx
S: Date: Mon, 14 Feb 2011 10:17:05 GMT
S: Content-Type: application/octet-
stream
S: Content-Length: 197
S: Last-Modified: Wed, 27 Jan 2010
23:30:38 GMT
S: Connection: keep-alive
S: Accept-Ranges: bytes
S:
##### Тут сам файл #####
Connection closed by foreign host.
```

HTTP-запрос

# Состав HTTP-запроса

- Строка состояния  
(строка-статус, или строка запроса)
- Поля заголовка.
- Пустая строка.
- Тело запроса.

# Состав HTTP-запроса



# Некоторые методы запроса

OPTIONS

**GET**

**HEAD**

POST

PUT

PATCH

DELETE

TRACE

CONNECT

# Метод OPTIONS

Используется для определения возможностей веб-сервера или параметров соединения для конкретного ресурса.

Формат тела и порядок работы с ним в настоящий момент не определён.

Запросы «OPTIONS \* HTTP/1.1» могут также применяться для проверки работоспособности сервера (аналогично «пингованию») и тестирования на предмет поддержки сервером протокола HTTP версии 1.1.



# Метод 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.

# Метод 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.

# Метод 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.

# Метод PUT

Применяется для загрузки содержимого запроса на указанный в запросе URI. Если по заданному URI не существовало ресурса, то сервер создаёт его и возвращает статус 201 (Created).

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.

# Метод PATCH

Аналогично PUT, но применяется только к фрагменту ресурса.

# Метод DELETE

Удаляет указанный ресурс.

# Метод TRACE

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

# Метод CONNECT

Преобразует соединение запроса в прозрачный TCP/IP туннель, обычно чтобы содействовать установлению защищенному SSL соединению через не зашифрованный прокси.



# Часто используемые поля заголовка запроса HTTP

Host

Referer

From

Accept

Accept-Language

Accept-Charset

Content-Type

Content-Length

Accept-Encoding

Range

Connection

User-Agent

# Host

Доменное имя или IP-адрес узла, к которому  
обращается клиент

# Referer

URL документа, который ссылается на ресурс,  
указанный в строке состояния

From

Адрес электронной почты пользователя,  
работающего с клиентом

# Ассерт

MIME-типы данных, обрабатываемых клиентом.

Это поле может иметь несколько значений, отделяемых одно от другого запятыми. Часто поле заголовка Ассерт используется для того, чтобы сообщить серверу о том, какие типы графических файлов поддерживает клиент

# Accept-Language

Набор двухсимвольных идентификаторов, разделенных запятыми, которые обозначают языки, поддерживаемые клиентом

# Accept-Charset

Перечень поддерживаемых наборов символов

# Content-Type

MIME-тип данных, содержащихся в теле запроса  
(если запрос не состоит из одного заголовка)



# Content-Length

Число символов, содержащихся в теле запроса  
(если запрос не состоит из одного заголовка)

# Accept-Encoding

Поле заголовка запроса Accept-Encoding сходно с полем Accept, но регламентирует кодировку содержимого, которая приемлема в отклике. Пустое поле Accept-Encoding указывает на то, что не приемлемо никакое кодирование.

Accept-Encoding: compress, gzip

# Range

Присутствует в том случае, если клиент запрашивает не весь документ, а лишь его часть

# Connection

Используется для управления ТСР-соединением. Если в поле содержится Close, это означает, что после обработки запроса сервер должен закрыть соединение. Значение Keep-Alive предлагает не закрывать ТСР-соединение, чтобы оно могло быть использовано для последующих запросов

# User-Agent

Информация о клиенте

# Пример запроса стандартного браузера

**GET /robots.txt HTTP/1.1**

**Host:** ftp.mgts.by

**User-Agent:** Mozilla/5.0 (X11; U; Linux  
x86\_64; en-US; rv:1.9.2.14pre)

Gecko/20101222 ALTLinux/Sisyphus/3.6.13 -  
alt1.20101222 Firefox/3.6.13

**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:** 115

**Connection:** keep-alive

**Referer:** http://ftp.mgts.by/



HTTP-ответ



# Структура ответа сервера

- Строка состояния.
- Поле заголовка.
- Пустая строка.
- Тело ответа.

# Строка состояния

Версия\_протокола Код\_ответа Пояснительное\_сообщение

**Версия\_протокола** - задается в том же формате, что и в запросе клиента, и имеет тот же смысл

**Код\_ответа** - это трехзначное десятичное число, представляющее в закодированном виде результат обслуживания запроса сервером.

**Пояснительное\_сообщение** - дублирует код ответа в символьном виде. Это строка символов, которая не обрабатывается клиентом.

# Классы кодов ответов

**1 - специальный класс сообщений,** называемых информационными. Код ответа, начинающийся с 1, означает, что сервер продолжает обработку запроса.

**2 - успешная обработка запроса клиента.**

**3 - перенаправление запроса.** Чтобы запрос был обслужен, необходимо предпринять дополнительные действия.

**4 - ошибка клиента.** Как правило, возвращается в том случае, если в запросе клиента встретилась синтаксическая ошибка.

**5 - ошибка сервера.**

# Часто используемые поля заголовков ответов

Server

Age

Allow

Content-Language

Content-Type

Content-Length

Content-Encoding

Last-Modified

Date

Expires

Location

Cache-Control

Transfer-Encoding

# Server

Имя и номер версии сервера

# Age

Время в секундах, прошедшее с момента  
создания ресурса

# Allow

Список методов, допустимых для данного ресурса

# Content-Language

Языки, которые должен поддерживать клиент  
для того, чтобы корректно отобразить  
передаваемый ресурс



# Content-Type

MIME-тип данных, содержащихся в теле запроса  
(если запрос не состоит из одного заголовка)

# Content-Length

Число символов, содержащихся в теле запроса  
(если запрос не состоит из одного заголовка)

# Content-Encoding

Значения кодировки содержимого указывают на кодовое преобразование, которое было или может быть выполнено над объектом. Кодировки содержимого первоначально применены для того, чтобы иметь возможность архивировать документ или преобразовать его каким-то другим способом без потери идентичности или информации.

# Last-Modified

Дата и время последнего изменения ресурса

# Date

Дата и время, определяющие момент генерации  
ответа

# Expires

Дата и время, определяющие момент, после которого информация, переданная клиенту, считается устаревшей

# Location

В этом поле указывается реальное расположение ресурса. Оно используется для перенаправления запроса

# Cache-Control

Директивы управления кэшированием.  
Например, no-cache означает, что данные не  
должны кэшироваться



# Transfer-Encoding

Поле общего заголовка Transfer-Encoding указывает, какой тип преобразования (если таковое использовано) применен к телу сообщения, для того чтобы безопасно осуществить передачу между отправителем и получателем. Это поле отличается от Content-Encoding тем, что транспортное кодирование является параметром сообщения, а не объекта.

Например – передача ответов с заранее неизвестной длиной, либо кодирование 8-битной информации для передачи по 7-битным линиям.

Transfer-Encoding: chunked

# Пример ответа сервера

**HTTP/1.1 200 OK**

**Server:** nginx

**Date:** Mon, 14 Feb 2011 11:17:40 GMT

**Content-Type:** application/octet-stream

**Content-Length:** 197

**Last-Modified:** Wed, 27 Jan 2010  
23:30:38 GMT

**Connection:** keep-alive

**Accept-Ranges:** bytes

# Безопасность передачи данных HTTP

Протокол HTTP предназначен для передачи  
символьных данных в открытом  
(незашифрованном) виде.

Самым простейшим является расширение  
HTTPS, при котором данные, передаваемые по  
протоколу HTTP, "упаковываются" в  
криптографический протокол SSL или TLS, тем  
самым обеспечивая защиту этих данных. В  
отличие от HTTP, для HTTPS по умолчанию  
используется TCP-порт 443

# Аутентификация в HTTP

Basic - базовая аутентификация, при которой имя пользователя и пароль передаются в заголовках http-пакетов.

Digest - пароль пользователя передается в хешированном виде. Использование SSL является обязательным.

Integrated - интегрированная аутентификация, с помощью протоколов NTLM или Kerberos.

# “Печеньки” :-)

HTTP-сервер не помнит предыстории запросов клиентов, поэтому каждый запрос обрабатывается независимо от других, и у сервера нет возможности определить, исходят ли запросы от одного клиента или разных клиентов.

Механизм **cookie** позволяет серверу хранить информацию на компьютере клиента и извлекать ее оттуда.

# Cookie

Инициатором записи cookie выступает сервер.

Если в ответе сервера присутствует поле заголовка **Set-cookie**, клиент воспринимает это как команду на запись cookie. В дальнейшем, если клиент обращается к серверу, от которого он ранее принял поле заголовка Set-cookie, помимо прочей информации он передает серверу данные cookie. Для передачи указанной информации серверу используется поле заголовка **Cookie**.

# Формат полей для печенек

В одном запросе или ответе может содержаться несколько полей Set-Cookie/Cookie.

**Set-Cookie:** name=newvalue; expires=date; path=/  
domain=.example.org.

Set-Cookie: RMID=732423sdfs73242; expires=Fri, 31-Dec-  
2010 23:59:59 GMT; path=/.; domain=.example.net

**Cookie:** name=value; name2=value2