

Это заголовки запросов HTTP и их значения:

1. **Accept**: Этот заголовок определяет типы контента, которые клиент готов принять от сервера. В данном случае указаны различные форматы, такие как HTML, XML, изображения и т. д., с их соответствующими качествами (q-values), указывающими на приоритет.
2. **Accept-Encoding**: Этот заголовок указывает, какие методы сжатия данных клиент поддерживает. В данном случае указаны методы gzip, deflate и br (Brotli).
3. **Accept-Language**: Этот заголовок указывает на предпочтительные языки клиента для содержимого ответа. В данном случае указаны русский и английский языки с их качествами (q-values).
4. **Connection**: Этот заголовок указывает на тип соединения, который клиент хочет установить с сервером. В данном случае указано keep-alive, что означает сохранение соединения открытым после завершения текущего запроса для возможных последующих запросов.
5. **Cookie**: Этот заголовок содержит информацию о куки, сохраненных на клиентской стороне, которые клиент отправляет обратно на сервер. Куки могут использоваться для идентификации пользователя и сохранения состояния сеанса между запросами.
6. **Host**: Этот заголовок указывает на доменное имя сервера, к которому направлен запрос.
7. **Sec-Ch-Ua**, **Sec-Ch-Ua-Mobile**, **Sec-Ch-Ua-Platform**: Эти заголовки являются частью механизма "Client Hints", который сообщает серверу о характеристиках клиента (браузера), таких как бренд, версия, мобильность и платформа.
8. **Sec-Fetch-Dest**, **Sec-Fetch-Mode**, **Sec-Fetch-Site**, **Sec-Fetch-User**: Эти заголовки являются частью механизма "Fetch Metadata", который сообщает серверу о контексте запроса. Например, они указывают на то, что запрос направлен на загрузку документа (document), каким образом был инициирован запрос (navigate), что запрос кросс-доменный (cross-site) и информацию о пользователе.
9. **Upgrade-Insecure-Requests**: Этот заголовок указывает, что клиент предпочитает загружать ресурсы через защищенное соединение (HTTPS), если это возможно.
10. **User-Agent**: Этот заголовок содержит информацию о браузере и операционной системе клиента. В данном случае указан браузер Chrome для Windows.

Вот объяснение каждого из представленных заголовков ответов HTTP:

1. **Cache-Control**: Этот заголовок указывает на инструкции по кэшированию содержимого ответа на стороне клиента и прокси-серверов. В данном случае:
   * **no-store**: Запрещает кэширование содержимого.
   * **no-cache**: Кеширование разрешено, но кэш должен проверяться на устаревшие данные.
   * **must-revalidate**: Кэш должен перепроверяться на устаревшие данные перед каждым использованием.
2. **Connection**: Этот заголовок указывает на тип соединения между клиентом и сервером. В данном случае **Keep-Alive** означает, что соединение будет сохранено открытым после завершения текущего запроса для возможных последующих запросов.
3. **Content-Encoding**: Этот заголовок указывает на метод сжатия, примененный к содержимому ответа. **gzip** означает, что содержимое было сжато с использованием алгоритма GZIP.
4. **Content-Type**: Этот заголовок определяет тип содержимого ответа и его кодировку. В данном случае это HTML-документ с кодировкой UTF-8.
5. **Date**: Этот заголовок указывает на дату и время, когда был отправлен ответ.
6. **Expires**: Этот заголовок указывает на дату и время, когда содержимое ответа становится недействительным и должно быть перезапрошено у сервера.
7. **Keep-Alive**: Этот заголовок указывает на параметры поддержания активного соединения между клиентом и сервером. Здесь указано, что таймаут соединения - 5 секунд, а максимальное количество запросов - 100.
8. **Pragma**: Этот заголовок является устаревшим и редко используется, но в данном случае **no-cache** указывает на то, что содержимое не должно кэшироваться.
9. **Server**: Этот заголовок указывает на серверное программное обеспечение, которое обрабатывает запрос.
10. **Transfer-Encoding**: Этот заголовок указывает на метод передачи содержимого, в данном случае это **chunked**, что означает, что содержимое передается по частям (chunk) вместо одного большого блока.
11. **Vary**: Этот заголовок указывает на заголовки запросов, которые могут привести к разным версиям содержимого в ответе. В данном случае это **Accept-Encoding**, что означает, что содержимое может изменяться в зависимости от заголовка Accept-Encoding в запросе.
12. Протокол HTTP описывается в документе, называемом "RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1". Этот документ содержит спецификацию протокола HTTP, определяющую его основные концепции, методы, заголовки и статусы ответов.
13. HTTP-протокол находится на прикладном уровне в модели OSI/ISO и на уровне приложений в модели TCP/IP.
14. Web-приложение - это программное обеспечение, разработанное для выполнения определенных задач через Интернет при помощи веб-браузера. Они доступны через веб-браузер и могут предоставлять различные функциональные возможности, такие как обработка данных, доступ к базам данных, социальные сети и так далее.
15. HTTP-протокол использует стандартные порты TCP/IP: 80 для HTTP и 443 для HTTPS (защищенная версия HTTP с использованием шифрования SSL/TLS).
16. RFC (Request for Comments) - это серия документов, содержащих описания и спецификации интернет-протоколов, архитектур и других связанных с интернетом технологий. RFC разрабатываются и публикуются сообществом интернет-инженеров и стандартизируют важные аспекты интернета.
17. HTTP-протокол называется ассиметричным из-за того, что он обладает асимметричным характером запроса и ответа. В клиент-серверной модели, используемой HTTP, клиент отправляет запросы на сервер, а сервер отвечает на эти запросы, что создает асимметрию в обмене данными между ними.
18. MIME (Multipurpose Internet Mail Extensions) - это стандарт, используемый для кодирования различных типов файлов для передачи по электронной почте или через протокол HTTP. MIME определяет типы контента и способы их кодирования, что позволяет эффективно передавать файлы различных форматов через Интернет.

шифрование шттпс жизненный цикл сессии

Куки выглядят как словарь это временное хранение данных

Что такое шттп стейтлис протокол не сохраняет состояние (не сохраняет данные)

Типы заголовков дженерал энитити реквест респонс

Сессия тоже хранит данные

**Интернет:** Всемирная компьютерная сеть, построенная на основе стека протоколов TCP/IP.

**Службы**

1. DNS, E-mail(STMP,POP3, IMAP), IRC(обмен сообщений в реальном времени), FTP, Telnet (управление удаленным компьютером в терминальном режиме), WWW.

**Жизненый цикл**

1. Клиент устанавливает TCP соединения (или другое соединение, если не используется TCP транспорт).
2. Клиент отправляет запрос и ждёт ответа.
3. Сервер обрабатывает запрос и посылает ответ, в котором содержится код статуса и соответствующие данные.

**Компоненты интернета (службы ресурс организации которой они катируются стек тсп\ип)**

**Типы взаимодействия и отличия**

1. **Дуплекс**: Обмен данными может происходить в обоих направлениях одновременно. Например, в телефонной связи оба абонента могут слышать друг друга одновременно.
2. **Полудуплекс**: Обмен данными возможен только в одном направлении за раз. Например, рация позволяет передавать данные только в одном направлении в определенный момент времени.
3. **Симплекс**: Обмен данными происходит только в одном направлении. Например, радиовещание или телевизионное вещание, где информация передается от передатчика к получателям, но не наоборот.

**Версии шттп**

Первая версия HTTP (0.9) была сильно ограничена. Эта версия позволяла клиентам запрашивать информацию с сервера только с помощью одной операции: GET.

1. **HTTP/1.0**: Основная версия HTTP с простыми запросами и ответами. Нет поддержки длинных и управляемых соединений.
2. **HTTP/1.1**: Внес улучшения в производительность, включая поддержку длинных соединений и заголовков Keep-Alive. помимо уже доступных методов HTTP 1.0, в версии 1.1 добавлено шесть дополнительных методов: PUT, PATCH, DELETE, CONNECT, TRACE и OPTIONS.
3. **HTTP/2**: Внедрил мультиплексирование, компрессию заголовков, приоритизацию запросов и другие улучшения для повышения производительности.
4. **HTTP/3**: Использует протокол QUIC вместо TCP для улучшения производительности и

**Свойства шттп**

1. **HTTP:** основные свойства

* версии HTTP/1.1 – действующий (текстовый), HTTP/2 – черновой (не распространен, бинарный);
* два типа абонентов: клиент и сервер;
* два типа сообщений: request и response;
* от клиента к серверу – request;
* от сервера к клиенту – response;
* на один request всегда один response, иначе ошибка;
* одному response всегда один request, иначе ошибка;
* TCP-порты: 80, 443;
* для адресации используется URI или URN;
* поддерживается W3C, описан в нескольких RFC.
* **Структура запрос (метод юрл версия шттп заголовки параметры тело)**
* **Ответ Версия статус код пояснение заголовки и тело при наличии**

1. **Заголовки:**

* **General**: общие заголовки, используются в запросах и ответах;
* **Request**: используются только в запросах;
* **Response:** используются только в ответах;
* **Entity**: для сущности в ответах и запросах.

1. **Response: Код состояния:**

* **1xx**: информационные сообщения;
* **2xx**: успешный ответ;
* **3xx**: переадресация;
* **4xx**: ошибка клиента;
* **5xx**: ошибка сервера.

**URI: Uniform Resource Identifier** – унифицированный идентификатор ресурса (документ, изображение, файл, служба, электронная почта,…).

1. **Web-серверы:** IIS (Microsoft), Apache (Apache Software Foundation), WebSphere (IBM), WebLogic (Oracle), Glassfish (Oracle), Apache Tomcat(Apache Software Foundation), nginx (И.Сысоев),…

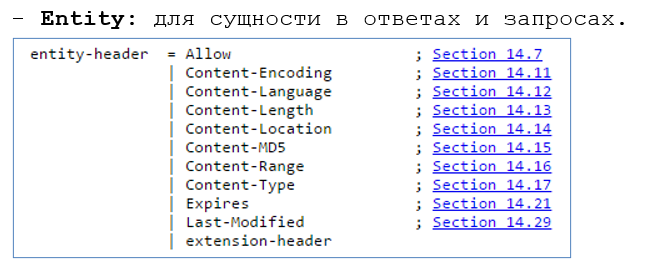
[https://metanit.com/sharp/aspnet5/11.1.php  
способы](https://metanit.com/sharp/aspnet5/11.1.phpспособы) сохранения состояния  
идемпатентность

цикл сессии

энтити заголовки

Метод HTTP является идемпотентным, 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](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/GET), [HEAD](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/HEAD), [PUT](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/PUT) и [DELETE](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/DELETE) **идемпотентны**, но не метод [POST](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/POST). Также все [безопасные](https://developer.mozilla.org/ru/docs/Glossary/Safe) методы являются идемпотентными.

GET, PUT, DELETE, HEAD и OPTIONS.



1. **Cookies (Куки)**: Cookies - это небольшие текстовые файлы, отправляемые сервером и хранимые в браузере пользователя. Они могут содержать информацию о состоянии сеанса, такую как идентификатор сеанса или пользовательские предпочтения. Куки могут быть надежным способом сохранения состояния, но они также могут быть подвержены атакам, таким как кража и подделка.
2. **Session Storage (Хранилище сессии)**: Session Storage - это механизм хранения данных на стороне клиента, который сохраняет данные только в течение сеанса браузера. Данные Session Storage доступны только для конкретной вкладки или окна браузера и удаляются при закрытии вкладки.
3. **Local Storage (Локальное хранилище)**: Local Storage - это другой механизм хранения данных на стороне клиента, который сохраняет данные на постоянной основе. Данные Local Storage доступны в течение длительного времени и могут быть использованы для сохранения состояния сеанса между различными посещениями сайта.
4. **Сессионные переменные на сервере**: Некоторые веб-платформы позволяют хранить состояние сеанса на сервере, используя сессионные переменные. Эти переменные связаны с конкретной сессией пользователя и могут использоваться для сохранения информации о состоянии сеанса, такой как аутентификационные данные или пользовательские предпочтения.

Объект сессии хранится на строне сервера когда клиент устанавливает соед с серваком он отправляет заголовок куки ид сессии а сервак обрабатывая запрос смотри, если не было такоко ид то создает новый обхет сессиии и отправляет его клиенту или когда клиент не смог отправить(куки удалились) сервер генерит новый объект сессиии и передаст ему его, в виде заголовком сет куки сервер отправляет ид сессии а клиент когда к серверу отправляет ид сесиии сервак мщет есть ли такой ид и если есть то соединение проходит через эту сессииию