|  |
| --- |
| EPAM Systems, CDP RU Dep. |
| Questions  HTTP Protocol Theory |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| REVISION HISTORY | | | | | |
| Ver. | Description of Change | Author | Date | Approved | |
| Name | Effective Date |
| <1.0> | Первая версия |  | <25.10.2014> |  |  |
|  |  |  |  |  |  |

**Name:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 0.** Что такое HTTP?

Answer: HTTP — [протокол](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB) [прикладного уровня](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%D1%8B_%D0%BF%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D1%8F) передачи данных, изначально — в виде гипертекстовых документов в формате [HTML](https://ru.wikipedia.org/wiki/HTML), в настоящее время используется для передачи произвольных данных.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 1.** Что означает Request header: Connection Keep-Alive

Answer: Keep-Alive - время, в течении которого может быть открыто соединение.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 2.** О чем информируют ответы от сервера с кодами 1xx, 2xx, 3xx, 4xx, 5xx?

Answer:

1xx - информирование о процессе передачи.

2xx - информирование о случаях успешного принятия обработки запроса клиента.

3xx - информирование клиента, что для успешного выполнения операции необходимо сделать другой запрос

(как правило по другому URI).

4xx - указание ошибок со стороны клиента.

5xx - информирование о случаях неудачного выполнения операции по вине сервера.

**Question 3.** В чем основное отличие методов POST и GET?

Answer: В отличие от метода GET, метод POST не считается идемпотентным[[2]](https://ru.wikipedia.org/wiki/HTTP#cite_note-http-spec-methods-2), 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).

POST - неограниченное количество данных (шифрование).

GET - ограниченное количество параметров (параметры находя в строке URI в не шифрованном виде).

Сообщение ответа сервера на выполнение метода POST не кэшируется.

**Question 4.** В чем особенность запроса HEAD и для каких целей он служит?

Answer:

Аналогичен методу GET, за исключением того, что в ответе сервера отсутствует тело. Запрос HEAD обычно применяется для извлечения [метаданных](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%B0%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5), проверки наличия ресурса ([валидация](https://ru.wikipedia.org/wiki/%D0%92%D0%B0%D0%BB%D0%B8%D0%B4%D0%B0%D1%86%D0%B8%D1%8F) URL) и чтобы узнать, не изменился ли он с момента последнего обращения.

Заголовки ответа могут кэшироваться. При несовпадении [метаданных](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%B0%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5) ресурса с соответствующей информацией в кэше — копия ресурса помечается как устаревшая.

**Question 5.** Вы запрашиваете изображение в формате gif. Браузер отображает не изображение, а выводит текстовое содержание gif-файла. В чем может быть проблема? Как можно исправить на клиенте, если нет доступа к серверу?

Answer:

HTTP Request Content-Type: text/html.

На клиенте можно исправить созданием нового файла формата gif.

**Question 6.** Что такое cookie? Какие ограничения и минусы использования?

Answer:

Ку́ки — небольшой фрагмент данных, отправленный [веб-сервером](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) и хранимый на [компьютере](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) пользователя. Веб-клиент (обычно [веб-браузер](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80)) всякий раз при попытке открыть страницу соответствующего сайта пересылает этот фрагмент данных веб-серверу в составе [HTTP](https://ru.wikipedia.org/wiki/HTTP)-запроса.

Если на компьютере используется более одного браузера, то, как правило, каждый имеет отдельное хранилище для cookie.

Cookie могут быть украдены с помощью анализа [трафика](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D1%82%D1%80%D0%B0%D1%84%D0%B8%D0%BA) — это называется взломом сессии. Сетевой трафик может быть перехвачен не только его отправителем и получателем (особенно в публичных сетях [Wi-Fi](https://ru.wikipedia.org/wiki/Wi-Fi)).

Хотя теоретически cookie должны сохраняться и отправляться назад на сервер неизменными, злоумышленник может изменить их содержимое перед отправкой.

Каждый сайт должен иметь свои собственные cookie, и сайт example1.com не должен изменять или устанавливать cookie другого сайта example2.org.

Cookie могут вызвать противоречия между клиентом и сервером. Если пользователь получает cookie, а затем нажимает кнопку «Назад» в браузере, то состояние браузера уже другое по сравнению с моментом получения cookie.

Постоянные cookie критикуются экспертами за свой долгий срок хранения, который позволяет веб-сайтам отслеживать пользователей и создавать их профиль с течением времени. Здесь затрагиваются и вопросы безопасности, поскольку украденные постоянные cookie могут использоваться на протяжении значительного периода времени.

**Question 7.** Чем отличаются ответы от сервера 401 Unauthorized и 403 Forbidden?

Answer:

401 - 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.

403 - 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](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%BA%D0%BE%D0%B4%D0%BE%D0%B2_%D1%81%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D1%8F_HTTP#401), или [407](https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%BA%D0%BE%D0%B4%D0%BE%D0%B2_%D1%81%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D1%8F_HTTP#407) при использовании прокси.

**Question 8.** Какие можете назвать особенности безопасного (https) соединения?

Answer:

HTTPS - оболочка вокруг HTTP с протоколами SSL/TLS.

Защиту данных в HTTPS обеспечивает криптографический протокол [SSL](https://ru.wikipedia.org/wiki/SSL)/[TLS](https://ru.wikipedia.org/wiki/TLS), который шифрует передаваемую информацию.

Используется цифровой сертификат — электронный документ, который используется для идентификации сервера.

**Question 9.** Вы сделали простые изменения в файле на «сервере» (.html, .css, .js или другой), но этот «файл/страница» отображается в браузере без ваших последних изменений. В чём может быть проблема? Что следует сделать (на «сервере», в браузере), чтобы увидеть последние изменения?

Answer:

Проблема в том, что мы не запросили произведенные изменения. Для того, чтобы увидеть изменения необходимо обновить окно браузера (послать запрос на сервер на получение данных).

**Question 10.** Из чего состоит HTTP-сообщение?

Answer:

Структура:

1. Стартовая строка - определяет тип сообщения (Метод URI HTTP/Версия (request) HTTP/Версия КодСостояния Пояснение (response)).
2. Заголовки - характеризуют тело сообщения, параметры передачи и прочие сведения.
3. Тело сообщения - непосредственно данные сообщения. Обязательно должно отделяться от заголовков пустой строкой.

**Question 11.** Что такое заголовок HTTP-сообщения? Для чего он нужен?

Answer:

Заголовки HTTP ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *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](https://tools.ietf.org/html/rfc822)). Заголовки должны отделяться от тела сообщения хотя бы одной пустой строкой.

Заголовки характеризуют тело сообщения, параметры передачи и прочие сведения.

**Question 12.** Что такое тело HTTP-сообщения? Для чего оно нужно?

Answer:

Тело HTTP-сообщения (message-body), если оно присутствует, используется для передачи тела объекта, связанного с запросом или ответом. Тело сообщения отличается от тела объекта (entity-body) только в том случае, когда применяется кодирование передачи, что указывается полем заголовка Transfer-Encoding.

**Question 13.** Что такое HTTP-session? Есть ли у него состояние?

Answer:

HTTP-session - последовательность request-response транзакций.

HTTP - это протокол без сохранения состояния. Протокол без сохранения состояния не требует, чтобы HTTP-сервер сохранял информацию или статус каждого пользователя в течение нескольких запросов. Однако некоторые веб-приложения реализуют состояния или сеансы на стороне сервера, используя, например, файлы cookie HTTP или скрытые переменные в веб-формах.

**Question 14.** Какими способами можно передать данные пользователя в HTTP-запросе?

Answer:

Передать данные пользователя можно с помощью тела методов GET и POST.

**Question 15.** Какими способами можно передать данные пользователя в HTTP-ответе?

Answer:

Передать данные пользователя в ответе можно с помощью тела ответа.