*Лабораторная работа.*

**Тестирование на уровне протокола HTTP**

*Цель работы:* изучение методики тестирования взаимосвязей между компонентами веб-приложения на уровне протокола HTTP.

Методические указания

Большинство пользователей обращается к веб-приложениям через браузер. Рассмотрим вопросы обмена информацией между клиентской частью веб-приложения и его серверной частью по протоколу HTTP.

HTTP – символьно-ориентированный клиент-серверный протокол прикладного уровня без сохранения состояния, используемый сервисом World Wide Web. Данный протокол лежит в основе обмена данными в Интернете. Технические характеристики протокола разработаны Инженерным комитетом IETF (Internet Engineering Task Force) и представлены в спецификациях RFC 7230-7237.

Согласно протоколу HTTP сообщение состоит из трёх частей, которые передаются в следующем порядке (рис.1):

1. Стартовая строка (Starting line) – определяет тип сообщения;

2. Заголовки (Headers) – характеризуют тело сообщения, параметры передачи и прочие сведения (рис. 4.2);

3. Тело сообщения (Message Body) – непосредственно данные сообщения. Тело отделено от заголовков пустой строкой.

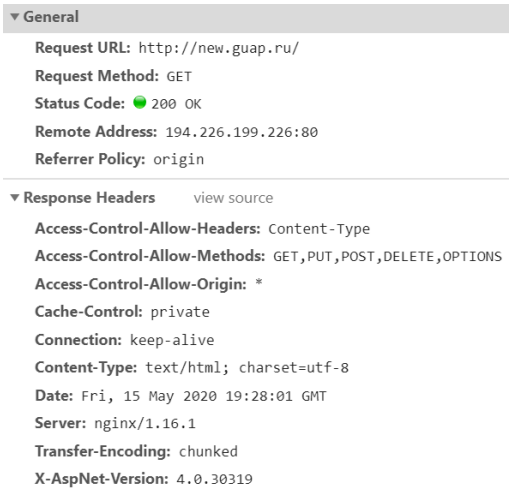


*Рис. 1. Заголовки сообщений*

Все HTTP-заголовки разделяются на четыре основных группы (табл. 1).

*Таблица 1*

**Группы HTTP-заголовков**

| Название заголовка | Описание |
|---|---|
| General Headers (Основные заголовки) | Включают любое сообщение клиента и сервера |
| Request Headers (Заголовки запроса) | Используют только в запросах клиента |
| Response Headers (Заголовки ответа) | Присутствуют только в ответах сервера |
| Entity Headers (Заголовки сущности) | Сопровождают каждую сущность сообщения |

При тестировании обращают внимание на методы получения информации и коды состояний веб-страницы. Метод HTTP (табл. 2) указывает на основную операцию над ресурсом.

*Таблица 2*

**Методы протокола HTTP**

| Название метода | Описание |
|---|---|
| DELETE | Удаляет указанный в запросе ресурс. |
| GET | Используется для запроса содержимого указанного ресурса. Согласно стандарту HTTP многократное повторение одного и того же запроса GET должно приводить к одинаковым результатам. |
| HEAD | Аналогичен методу GET, за исключением того, что в ответе сервера отсутствует тело. Запрос HEAD обычно применяют для извлечения метаданных, проверки наличия ресурса (валидация URL) и чтобы узнать, не изменился ли он с момента последнего обращения. |
| OPTIONS | Используется для определения возможностей веб-сервера или параметров соединения для конкретного ресурса. |
| POST | Используется для запроса содержимого указанного ресурса. Метод POST предполагает, что по указанному URI (Universal Resource Identifier) будет производиться обработка передаваемого клиентом содержимого. |
| PUT | Применяется для загрузки содержимого запроса на указанный в запросе URI. |
| PATCH | Аналогичен PUT, но применяется только к фрагменту ресурса. |
| TRACE | Возвращает полученный запрос так, что клиент может увидеть, что промежуточные сервера добавляют или изменяют в запросе. |

| | |
|---|---|
| LINK/UNLINK | Устанавливает/разрывает связь указанного ресурса с другими. |

Наиболее востребованными являются методы GET и POST – на человеко-ориентированных ресурсах, POST – роботами поисковых машин и оффлайн-браузерами.

Код состояния информирует о результатах выполнения запроса и определяет его дальнейшее поведение (табл. 3). Набор кодов состояния является стандартом, и все они описаны в соответствующих документах RFC. Структура кода состояния трехпозиционна. В старшей позиции стоит цифра, показывающая код класса, в двух других код сообщения.

*Таблица 3*

**Коды состояний выполнения протокола HTTP**

| Коды состояний | Описание | Примеры ответов сервера |
|---|---|---|
| 1xx<br>Информационный | Информируют о процессе передачи сообщения | 100 Continue (Продолжать)<br>101 Switching Protocols (Переключение протоколов)<br>102 Processing (Идёт обработка) |
| 2xx<br>Успешный | Информируют о случаях успешного принятия и обработки запроса клиента. | 200 OK (Успешно).<br>201 Created (Создано)<br>202 Accepted (Принято)<br>204 No Content (Нет содержимого)<br>206 Partial Content (Частичное содержимое) |
| 3xx<br>Перенаправление | Перенаправляют на другую страницу. | 300 Multiple Choices (Множественный выбор)<br>301 Moved Permanently (Перемещено навсегда)<br>304 Not Modified (Не изменялось) |
| 4xx<br>Ошибка клиента | Сообщают об ошибке со стороны клиента. | 401 Unauthorized (Неавторизован)<br>402 Payment Required (Требуется оплата)<br>403 Forbidden (Запрещено)<br>404 Not Found (Не найдено)<br>405 Method Not Allowed (Метод не поддерживается)<br>406 Not Acceptable (Не приемлемо)<br>407 Proxy Authentication Required (Требуется аутентификация прокси) |
| 5xx | Информируют об | 500 Internal Server Error |

| | | |
|---|---|---|
| Ошибка сервера | ошибке выполнения операции по вине сервера. | (Внутренняя ошибка сервера)<br>502 Bad Gateway (Плохой шлюз)<br>503 Service Unavailable (Сервис недоступен)<br>504 Gateway Timeout (Шлюз не отвечает) |

Информацию о статусе загрузке страницы можно узнать программными средствами Google Chrome. Для отображения данных необходимо нажать F12 и перейти на вкладку Network – сетевые компоненты (рис. 2). Просмотр корректной информации возможен в том случае, если после активации инструментов разработчика веб-страница перезагружена и в фильтре (Filter) установлен флаг ALL.

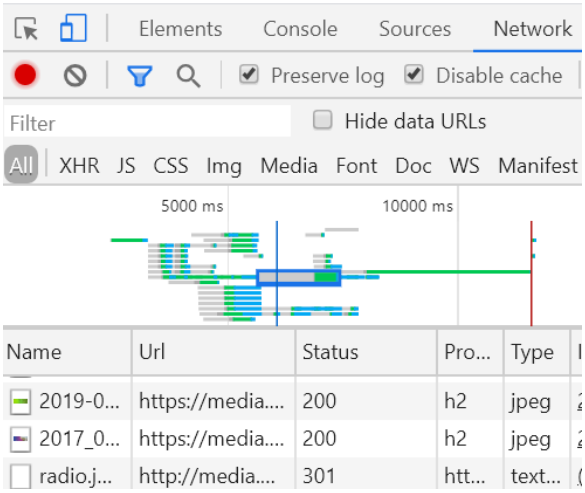


*Рис. 2. Информация о загрузке элементов веб-страницы на вкладке Network*

На вкладке Network отображается информация (рис. 3) о каждом загруженном на веб-страницу элементе (Name), его URL-адресе элемента, статусе загрузки страницы (Status), протоколе соединения (Protocol), типе загружаемого контента (Type), инициаторе запроса подключения элемента, размеру (Size), времени загрузки (Time), последовательности загрузки элементов (Waterfall).

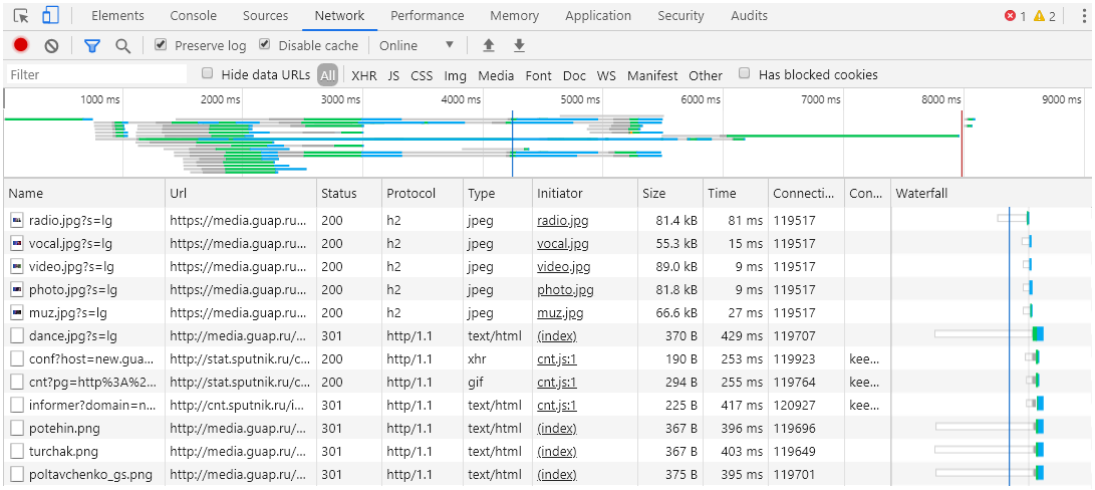


*Рис.3. Информация о сетевых компонентах веб-страницы*

При тестировании необходимо проверить каждую ссылку на внешние источники трафика и какой контент по ним загружается. В качестве примера рассмотрите все графические элементы контента. Для этого необходимо задать в параметрах фильтрации IMG. На выбранном элементе (рис.4) нужно нажать левой кнопкой мышки и активировать панель с информацией о нем: типе запроса (Headers), предпросмотре контента элемента (Prewiew), ответе сервера (Response), инициаторе запроса (Initiator), времени его выполнения (Timing) и куков (Cookies).

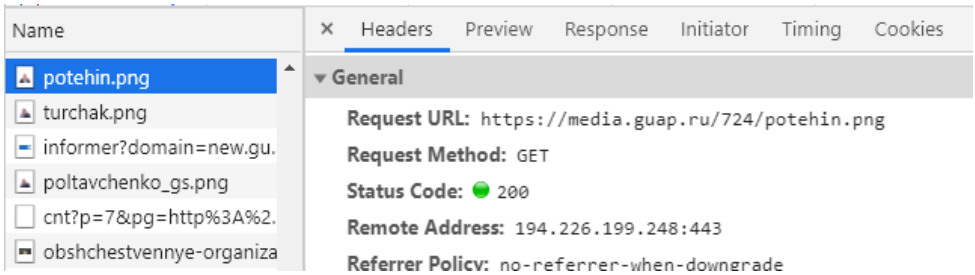


*Рис. 4.Информация об элементе potechin.png*

В Preview отображается контент в визуально-понятном для тестировщика виде (картинка), в Response – неотформатированный исходник. При запросе HTML, JS, PHP файлов вкладку Response требуется протестировать и убедиться, что в ней находится тот контент, который был запрошен.

Для тестирования протоколов HTTP разработано специализированное программное обеспечение, например Fiddler (`https://www.telerik.com/fiddler`) или Postman

(https://www.postman.com/). Идеология тестирования заключена в установке специального программного инструмента между клиентской и серверной частями веб-приложения (рис. 5).

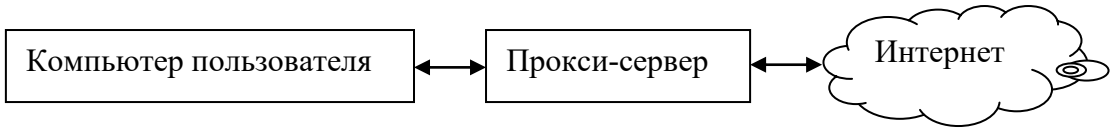


*Рис. 5. Схема передачи сообщений через прокси-сервер*

Рассмотрим последовательность работы с прокси-сервером Fidler.]

1. Запустим Fiddler и очистим Web Sessions list.

2. Активируем захват нового трафика при открытии браузера (рис. 6).

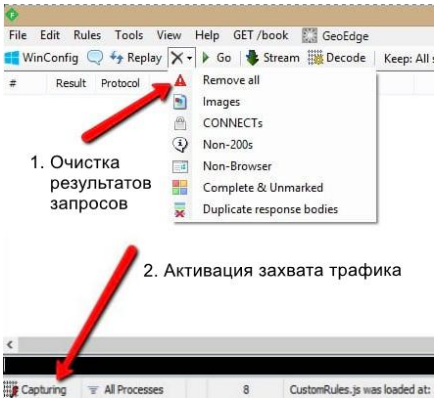


*Рис. 6. Первоначальные настройки Fidler*

3. Выполним запрос к сайту https://www.amazon.com/.

4. Остановим захват трафика нажатием F12. В нашем примере выполнено 79 сессии запросов и ответов (рис. 7).

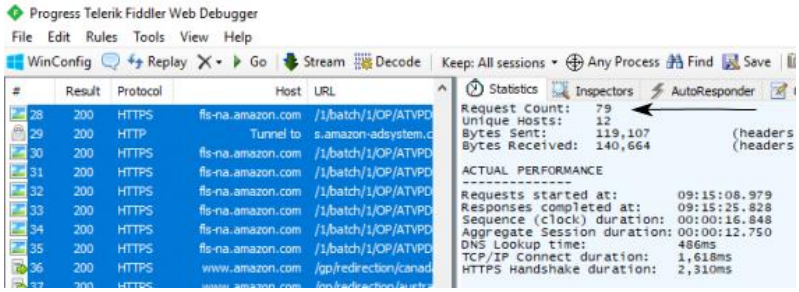


*Рис. 7. Полученные статистические данные в окне Statistics*

В программе предусмотрены возможности:

– получения статистических данных о загрузке веб-странице;

– измерения размера запроса и веса страницы;

– анализа кеша, сжатия и компоновки веб-страниц;

– имититации низкоскоростных и высокоскоростных соединений.

**Порядок выполнения работы**:

1. Откройте выбранное для тестирования приложение в браузере Google Chrome.

2. Активируйте вкладку инструментов веб-разработчика браузера. Для этого нажмите функциональную клавишу F12.

3. Перейдите на вкладку Network и перезагрузите страницу в браузере.

4. Последовательно задайте не менее трех параметров фильтрации: All, Img, JS, CSS (рис.8).

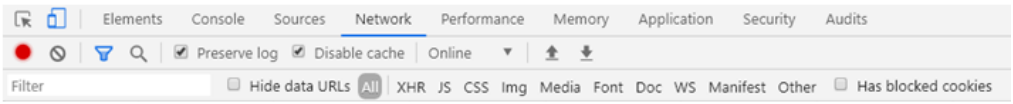


*Рис.8. Параметры фильтрации элементов веб-страницы*

5. Проверьте пять ссылок, идущих от вызывающих запросы элементов интерфейса, на другие веб-страницы или внешние источники данных. Сделайте скриншоты. Проверку проходят только пять ссылок от элементов, относящихся к трем разным категориям.

6. На каждом выбранном элементе нажать левой кнопкой мышки и проверить панель с информацией о нем: тип запроса (Headers), предпросмотр контента элемента (Prewiew), ответ сервера (Response), инициатор запроса (Initiator), времени его выполнения (Timing) и куках (Cookies).

Например, при тестировании заголовков сообщения-ответа от сервера требуется определить статус страницы, метод получения информации: GET, POST, PUT и т.д. В отчете предоставьте скриншоты состояний протокола передачи и объясните, что написано в теле протокола (рис. 9).

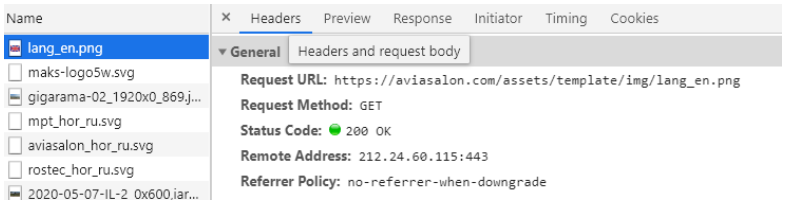


*Рис. 9.Основной заголовок протокола передачи PHP информационного объекта* `lang_en.png`

7. Объясните полученные результаты в выводе.