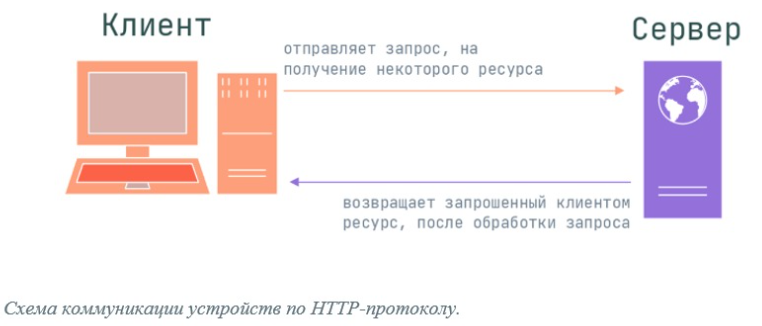
## Лекция 2.6. Парсинг HTML-страниц при помощи Python

В ходе лекции будут рассмотрены:  
1. Парсинг.  
2. Понятие и назначение.  
3. Протокол HTTP.  
4. Методы запросов.  
5. Версии HTTP.

### Глава 1. Парсинг HTML-страниц при помощи Python

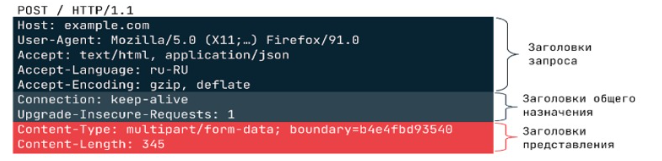
HTTP (HyperText Transfer Protocol, дословно — «протокол передачи гипертекста») представляет собой протокол прикладного уровня, используемый для доступа к ресурсам Всемирной Паутины. Под термином гипертекст следует понимать текст, в понятном для человека представлении, при этом содержащий ссылки на другие ресурсы.  
Данный протокол описывается спецификацией RFC 2616. На сегодняшний день наиболее распространённой версией протокола является версия HTTP/2, однако нередко все ещё можно встретить более раннюю версию HTTP/1.1.  
В обмене информацией по HTTP-протоколу принимают участие клиент и сервер. Происходит это по следующей схеме:  
1. Клиент запрашивает у сервера некоторый ресурс.  
2. Сервер обрабатывает запрос и возвращает клиенту ресурс, который был запрошен.



### HTTP-сообщения: запросы и ответы Данные между клиентом и сервером в рамках работы протокола передаются с помощью HTTP-сообщений. Они бывают двух видов: 1. Запросы (HTTP Requests) — сообщения, которые отправляются клиентом на сервер, чтобы вызвать выполнение некоторых действий. Зачастую для получения доступа к определённому ресурсу. Основой запроса является HTTP-заголовок. 2. Ответы (HTTP Responses) — сообщения, которые сервер отправляет в ответ на клиентский запрос. Само по себе сообщение представляет собой информацию в текстовом виде, записанную в несколько строчек. В целом, как запросы HTTP, так и ответы имеют следующую структуру: 1. Стартовая строка (start line) — используется для описания версии используемого протокола и другой информации: вроде запрашиваемого ресурса или кода ответа. Как можно понять из названия, её содержимое занимает ровно одну строчку. 2. HTTP-заголовки (HTTP Headers) — несколько строчек текста в определённом формате, которые либо уточняют запрос, либо описывают содержимое тела сообщения. 3. Пустая строка, которая сообщает, что все метаданные для конкретного запроса или ответа были отправлены. 4. Опциональное тело сообщения, которое содержит данные, связанные с запросом, либо документ (например: HTML-страницу), передаваемый в ответе.

### Глава 2. Парсинг HTML-страниц при помощи Python

### Методы Методы позволяют указать конкретное действие, которое мы хотим, чтобы сервер выполнил, получив наш запрос. Так некоторые методы позволяют браузеру (который в большинстве случаев является источником запросов от клиента) отправлять дополнительную информацию в теле запроса — например, заполненную форму или документ.



Разберёмся с каждым из названных элементов подробнее. См. таблицу 1.

|  |  |
| --- | --- |
| Метод | Описание |
| GET | Позволяет запросить некоторый конкретный ресурс. Дополнительные данные могут быть переданы через строку запроса (Query String) в составе URL (например, param=value).О составляющих URL мы поговорим чуть позже. |
| POST | Позволяет отправить данные на сервер. Поддерживает отправку различных типов файлов, среди которых текст, PDF-документы и другие типы данных в двоичном виде. Обычно метод POST используется при отправке информации (например, заполненной формы логина) и загрузке данных на веб-сайт, таких как изображения и документы. |
| HEAD | Здесь придётся забежать немного вперёд и сказать, что обычно сервер в ответ на запрос возвращает заголовок и тело, в котором содержится запрашиваемый ресурс. Данный метод при использовании его в запросе позволит получить только заголовки, которые сервер бы вернул при получении GET-запроса к тому же ресурсу. Запрос с использованием данного метода обычно производится для того, чтобы узнать размер запрашиваемого ресурса перед его загрузкой. |
| PUT | Используется для создания (размещения) новых ресурсов на сервере. Если на сервере данный метод разрешён без надлежащего контроля, то это может привести к серьёзным проблемам безопасности. |
| DELETE | Позволяет удалить существующие ресурсы на сервере. Если использование данного метода настроено некорректно, то это может привести к атаке типа «Отказ в обслуживании» (Denial of Service, DoS) из-за удаления критически важных файлов сервера. |
| OPTIONS | Позволяет запросить информацию о сервере, в том числе информацию о допускаемых к использованию на сервере HTTP-методов. |
| PATCH | Позволяет внести частичные изменения в указанный ресурс по указанному расположению. |

### URL Получение доступа к ресурсам по HTTP-протоколу осуществляется с помощью указателя URL (Uniform Resource Locator). URL представляет собой строку, которая позволяет указать запрашиваемый ресурс и ещё ряд параметров.

### Глава 3. Парсинг HTML-страниц при помощи Python

### Версии HTTP Последняя стабильная, наиболее стандартизированная версия протокола первого поколения (версия HTTP/1.1) вышла в далеком 1997 году. Годы шли, веб-страницы становились сложнее, некоторые из них даже стали приложениями в том виде, в котором их понимают сейчас. Кроме того, объём медиафайлов и скриптов, которые добавляли интерактивность страницам, рос. Это, в свою очередь, создавало перегрузки в работе протокола версии HTTP/1.1. Стало очевидно, что у HTTP/1.1 есть ряд значительных недостатков: 1. Заголовки, в отличие от тела сообщения, передавались в несжатом виде. 2. Часто большая часть заголовков в сообщениях совпадала, но они продолжали передаваться по сети. 3. Отсутствовала возможность так называемого мультиплексирования — механизма, позволяющего объединить несколько соединений в один поток данных. С выходом HTTP/2 было предложено следующее решение: HTTP/1.X-сообщения разбивались на так называемые фреймы, которые встраивались в поток данных. Фреймы данных (тела сообщения) отделялись от фреймов заголовка, что позволило применять сжатие. Вместе с появлением потоков появился и ранее описанный механизм мультиплексирования — теперь можно было обойтись одним соединением для нескольких потоков. HTTP-заголовок представляет собой строку формата «Имя-Заголовок: Значение», с двоеточием (:) в качестве разделителя. Название заголовка не учитывает регистр, то есть между Host и host, с точки зрения HTTP, нет никакой разницы. Однако в названиях заголовков принято начинать каждое новое слово с заглавной буквы. Структура значения зависит от конкретного заголовка. Несмотря на то, что заголовок вместе со значениями может быть достаточно длинным, занимает он всего одну строчку. Категории заголовков: 1. Общего назначения, которые применяются ко всему сообщению целиком. 2. Заголовки запроса уточняют некоторую информацию о запросе, сообщая дополнительный контекст или ограничивая его некоторыми логическими условиями. 3. Заголовки представления, которые описывают формат данных сообщения и используемую кодировку. Добавляются к запросу только в тех случаях, когда с ним передаётся некоторое тело.

### Глава 4. Парсинг HTML-страниц при помощи Python

Завершающая часть HTTP-запроса — это его тело.  
HTTP-ответ является сообщением, которое сервер отправляет клиенту в ответ на его запрос. Его структура равна структуре HTTP-запроса: стартовая строка, заголовки и тело.  
Стартовая строка HTTP-ответа называется строкой статуса (status line). На ней располагаются следующие элементы:  
1. Уже известная нам по стартовой строке запроса версия протокола (HTTP/2 или HTTP/1.1).  
2. Код состояния, который указывает, насколько успешно завершилась обработка запроса.  
3. Пояснение — короткое текстовое описание к коду состояния. Используется исключительно для того, чтобы упростить понимание и восприятие человека при просмотре ответа.



### Что такое парсинг и зачем он нужен? Парсинг (от англ. parsing — разбор, анализ), или веб-скрейпинг — это автоматизированный сбор информации с интернет-сайтов. Программы, которые собирают и обрабатывают информацию из Сети, называют парсерами (от англ. parser — анализатор). Сам парсинг используется для решения разных задач: с его помощью телеграм-боты могут получать информацию, которую затем показывают пользователям, маркетологи — подтягивать данные из социальных сетей, а бизнесмены — узнавать подробности о конкурентах.

1. Как называется процесс анализа и извлечения структурированных данных из веб-страниц с помощью программного кода?
   1. Парсинг HTML-страниц
   2. Протокол HTML-страниц
   3. Атака HTML-страниц
2. Во время какого этапа процесса парсинга HTML-страницы используются библиотеки или инструменты, чтобы отправить HTTP-запрос и получить HTML-код страницы?
   1. Загрузка HTML-кода
   2. Поиск и выборка данных
   3. Получение HTML-кода
   4. Создание парсера
3. Во время какого этапа процесса парсинга HTML-страницы загружается HTML-код в парсер, чтобы он мог начать его обработку?
   1. Загрузка HTML-кода
   2. Поиск и выборка данных
   3. Получение HTML-кода
   4. Создание парсера
4. Во время какого этапа процесса парсинга HTML-страницы инициализируется парсер HTML для анализа HTML-кода и извлечения данных?
   1. Загрузка HTML-кода
   2. Поиск и выборка данных
   3. Получение HTML-кода
   4. Создание парсера
5. Во время какого этапа процесса парсинга HTML-страницы используются методы парсера, выполнится поиск и выборка данных из HTML-страницы?
   1. Загрузка HTML-кода
   2. Поиск и выборка данных
   3. Получение HTML-кода
   4. Создание парсера
6. Во время какого этапа процесса парсинга HTML-страницы извлечённые данные преобразуются в данные нужного формата, или данные проходят фильтрацию, или сохраняются в базу данных?
   1. Извлечение ошибок
   2. Обработка ошибок
   3. Обработка и сохранение данных
7. Во время какого этапа процесса парсинга HTML-страницы необходимые данные извлекаются из найденных элементов?
   1. Извлечение ошибок
   2. Обработка ошибок
   3. Обработка и сохранение данных
8. Во время какого этапа процесса парсинга HTML-страницы возникшие ошибки обрабатываются с помощью механизмов исключений или проверок, чтобы обеспечить надёжность и корректность парсинга?
   1. Извлечение ошибок
   2. Обработка ошибок
   3. Обработка и сохранение данных
9. Основным протоколом передачи данных в сети Интернет является:
   1. Address Resolution Protocol
   2. Hypertext Transfer Protocol
   3. Нет верного ответа
10. Укажите метод, который используется для частичного обновления ресурса на сервере:
    1. PUT - поставить
    2. GET - получить
    3. DELETE - удалить
    4. PATCH - исправить
    5. POST - пост