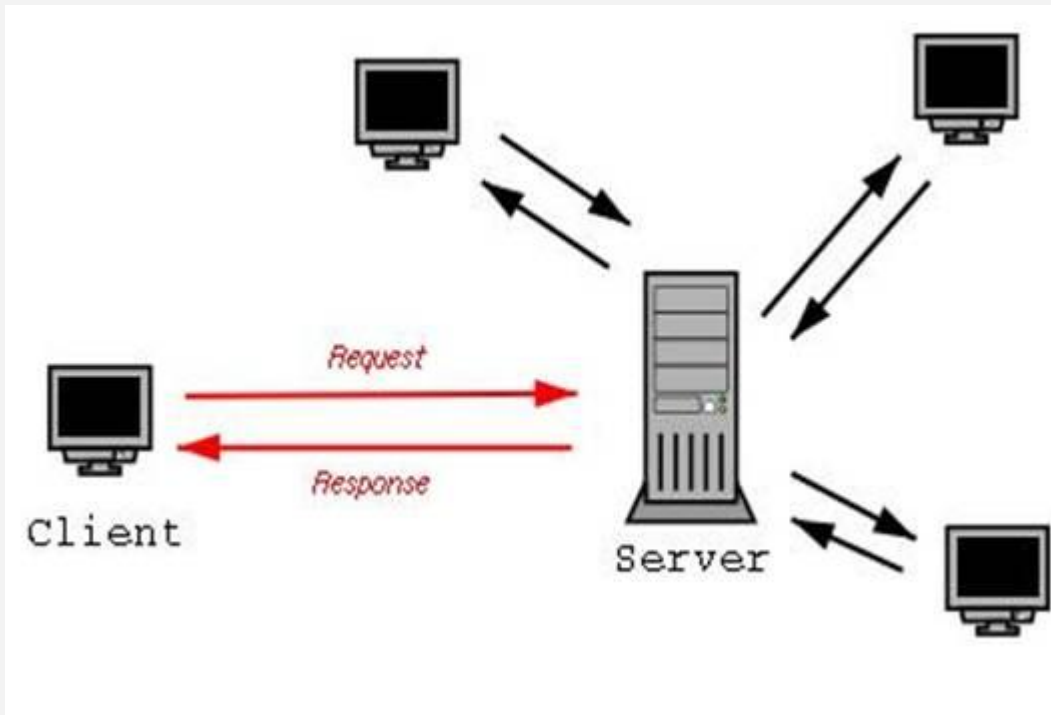


Лекция 8: Клиент-серверная архитектура

Курс лекций по основам web-разработки на языке программирования Ruby

Модель взаимодействия клиент-сервер



- **Mac** - Media Access Control (MAC) это низкоуровневый сетевой протокол. С ним, в той или иной мере приходится сталкиваться всем пользователям. Используется он для идентификации сетевых устройств.
- **IP** – Internet Protocol, имеющий две основные разновидности IPv4 и IPv6. Он назначает компьютерам уникальные IP-адреса, благодаря которым устройства могут себя обнаруживать в сети.
- **ICMP** - (Internet control message protocol), отвечающий за обмен информацией. Не используется для передачи данных. Именно ICMP используется в известной вам команде ping.
- **TCP** - (Transmission control protocol). Этот сетевой протокол управляет передачей данных. TCP дает гарантия в том, что все переданные пакеты данных будут приняты правильно и ошибки будут полностью исключены.
- **UDP** - (user datagram protocol) похож на TCP, но работает быстрее, так как в нем данные при получении не проверяются. В некоторых случаях использование UDP бывает вполне достаточным.
- **File Transfer Protocol** служит для передачи файлов. Советуем не использовать его для передачи важных данных, так как в FTP не поддерживается необходимая безопасность.
- **POP3** и **SMTP** - Два протокола для работы с электронной почтой. POP3 отвечает за получение почты, SMTP – за отправку. Оба протокола позволяют доставлять email, передавая его почтовому клиенту.
- **SSH** (Secure Shell) - относится к уровню приложений. Создает защищенный канал для удаленного управления другой операционной системой. Поддерживает различные алгоритмы шифрования.
- **HTTP** - По своей распространенности в Интернет Hyper Text Transfer protocol находится на первом месте, ведь именно на его основе работают все сайты. С его помощью с локального компьютера можно открыть веб-сервис на удалённом сервере.

URI

`scheme://user:password@host:port/path/file?parameters#fragment`

- `scheme` - прикладной протокол, посредством которого получают доступ к ресурсу.
- `user` - пользователь, от имени которого получают доступ к ресурсу либо сам пользователь в качестве ресурса.
- `password` - пароль пользователя для аутентификации при доступе к ресурсу.
- `host` - IP-адрес или имя сервера, на котором расположен ресурс.
- `port` - номер порта, на котором работает сервер, предоставляющий доступ к ресурсу.
- `path` - путь к файлу, содержащему ресурс.
- `file` - файл, содержащий ресурс.
- `parameters` - параметры для обработки ресурсом-программой.
- `fragment` - точка в файле, начиная с которой следует отображать ресурс.

Пример HTTP сообщений

Simple request

```
POST / HTTP/1.1
Host: foo.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 13

say=Hi&to=Mom
```

Simple response

```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
Content-Length: 88
Content-Type: text/html
Connection: Closed

<html>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```

Методы HTTP

GET запрашивает представление ресурса. Запросы с использованием этого метода могут только извлекать данные.

HEAD запрашивает ресурс так же, как и метод GET, но без тела ответа.

POST используется для отправки сущностей к определённому ресурсу. Часто вызывает изменение состояния или какие-то побочные эффекты на сервере.

PUT заменяет все текущие представления ресурса данными запроса.

PATCH используется для частичного изменения ресурса.

DELETE удаляет указанный ресурс.

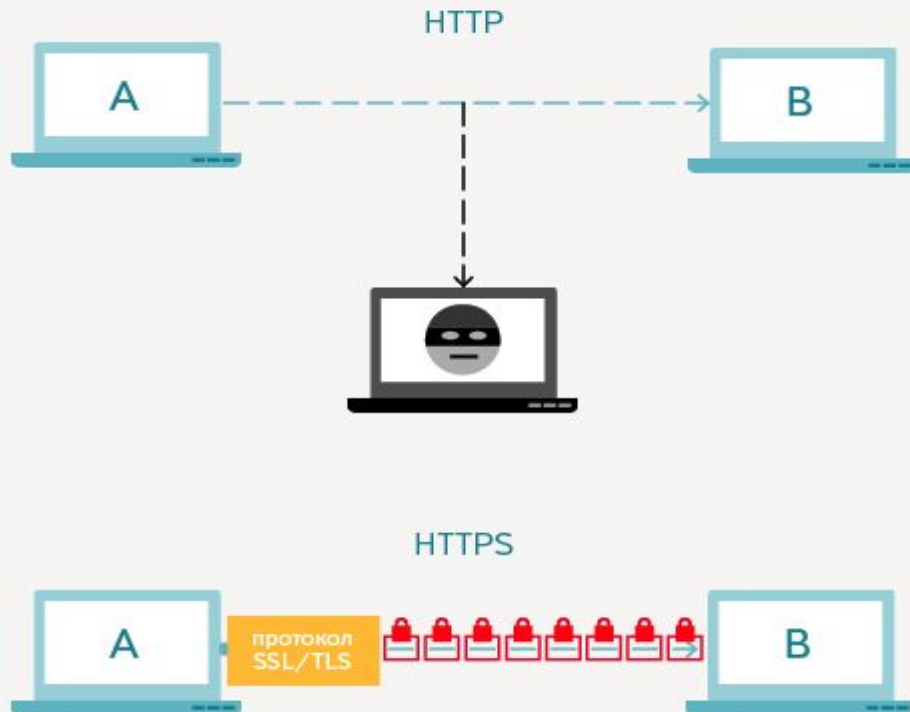
CONNECT устанавливает "туннель" к серверу, определённому по ресурсу.

OPTIONS используется для описания параметров соединения с ресурсом.

TRACE выполняет вызов возвращаемого тестового сообщения с ресурса.

- 1xx: Informational - информационные
 - 101 Switching Protocols - Переключение протоколов. Сервер предлагает выбрать другой протокол, более соответствующий данному ресурсу.
- 2xx: Success - Успешное завершение
 - 200 OK - Хорошо. Запрос к ресурсу выполнен успешно.
 - 201 Created - Создано. Запрос выполнен успешно, новый ресурс создан.
 - 204 No Content - Отсутствует содержимое. Сервер успешно обработал запрос, но не вернул содержимого.
- 3xx: Redirection - Редирект (перенаправление)
 - 301 Moved Permanently - Перемещено окончательно. Запрошенный ресурс был окончательно перенесен на URI, указанный в строке заголовка Location, ответа сервера.
- 4xx: Client Error - Ошибка клиента
 - 401 Unauthorized - Не авторизован. Ресурс требует идентификации пользователя.
 - 403 Forbidden - Запрещено. Сервер отказал в доступе к запрошенному ресурсу ввиду ограничений.
 - 404 Not Found - Не найдено. Сервер не нашел запрошенный ресурс по указанному адресу.
 - 429 Too many requests - Слишком много запросов. Пользователь отправил слишком много запросов в заданный период времени.
- 5xx: Server Error - Ошибка на стороне сервера
 - 500 Internal Server Error - Внутренняя ошибка сервера. Любая внутренняя ошибка на стороне сервера не подпадающая под остальные ошибки из категории 5xx.

HTTPS



Simple web servers

```
class HelloWorld
  def call(_env)
    [200, { 'Content-Type' => 'text/html' }, ['This is Rack!']]
  end
end

run HelloWorld.new
```

rackup hello_world.ru

```
const express = require('express');
const app = express();

app.get('/', function (req, res) {
  res.send('Hello World!');
});

app.listen(3000, function () {
  console.log('Example app listening on port 3000!');
});
```

node hello_world.js

🏠 yevhenii — lynx bash.im — 159×36

- * bash.im
- * ithappens.me
- * zadolba.li

Bash.im -- Citatnik Runeta

- * novye
- * luchshie
- * sluchajnye
- * po rejtingu

0 proekte

Tema

Novye Bezdna Komiksy Poisk Dobavit'

- * novye
- * luchshie
- * sluchajnye
- * po rejtingu

#407781

25.08.2010 v 10:12

xxx: kak napisat' v plane, chto ya segodnya nichego delat' ne sobirayus'?

yyy: Napishi diagnostika neispravnostej i profilakticheskoe obsluzhivanie serverov)

22581

#416677

23.04.2012 v 12:46

Democrat: ischu sebe vechnuyu lyubov' na leto

8556

-- press space for next page --

Arrow keys: Up and Down to move. Right to follow a link; Left to go back.

H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list

```
curl 'http://localhost:3000/api/customers/sessions' \  
-H 'Connection: keep-alive' \  
-H 'Pragma: no-cache' \  
-H 'Cache-Control: no-cache' \  
-H 'accept: application/json' \  
-H 'DNT: 1' \  
-H 'User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.122 Safari/537.36' \  
-H 'Content-Type: application/json' \  
-H 'Origin: http://localhost:3000' \  
-H 'Sec-Fetch-Site: same-origin' \  
-H 'Sec-Fetch-Mode: cors' \  
-H 'Sec-Fetch-Dest: empty' \  
-H 'Referer: http://localhost:3000/api-docs/index.html' \  
-H 'Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7,uk;q=0.6' \  
-H 'Cookie: Rubymine-27bb48b5=0c319978-355f-40db-93bb-f4e4d2b1be0e' \  
--data-binary '${\n "email": "a@a.aa",\n "password": "aa123456"\n}' \  
--compressed
```

The screenshot displays the Postman application interface. At the top, a dark header bar contains navigation buttons: 'New' (with a plus icon), 'Import', 'Runner', and a dropdown menu. On the right of the header is a 'My Workspace' dropdown. Below the header, the left sidebar features a search bar labeled 'Filter' and three tabs: 'History', 'Collections' (which is active and underlined), and 'APIs BETA'. Under the 'Collections' tab, there is a '+ New Collection' button and a 'Trash' section. A list of collections follows, including '79 requests', 'Users API' (1 request), 'VIP Customers' (4 requests), another 'VIP Customers' (4 requests with a sub-item 'new-customers'), 'Visualizer D3 heatmap demo' (1 request), and 'Visualizer Demo - Open Brewer...' (4 requests). The main panel on the right shows the details for the 'VIP Customers' collection. It has a title bar with a close button. Below the title, it identifies the 'API' as 'Customer API'. There are three buttons: 'Share' (orange), 'Run' (blue), and 'View in web' (grey, with a mouse cursor hovering over it). A three-dot menu button is also present. Below these buttons are tabs for 'Documentation' (active), 'Monitors', 'Mocks', and 'Changelog'. The 'Documentation' tab contains a link 'Learn how to document your requests' and a description: 'API to track **VIP customers** for marketing and rewards.' At the bottom, a list of API endpoints is shown: 'GET Get customers', 'GET Get customer', 'POST Add customer', and 'PATCH Update customer'. On the far right edge of the image, a portion of another API endpoint is visible, showing 'PATCH U' and a table with a 'VALUE' header and a 'Value' row.

New Import Runner My Workspace

Filter

History Collections APIs BETA

+ New Collection Trash

79 requests

Users API
1 request

VIP Customers
4 requests

VIP Customers
4 requests new-customers

Visualizer D3 heatmap demo
1 request

Visualizer Demo - Open Brewer...
4 requests

VIP Customers

API Customer API

Share Run View in web

Documentation Monitors Mocks Changelog

Learn how to document your requests

API to track **VIP customers** for marketing and rewards.

GET Get customers

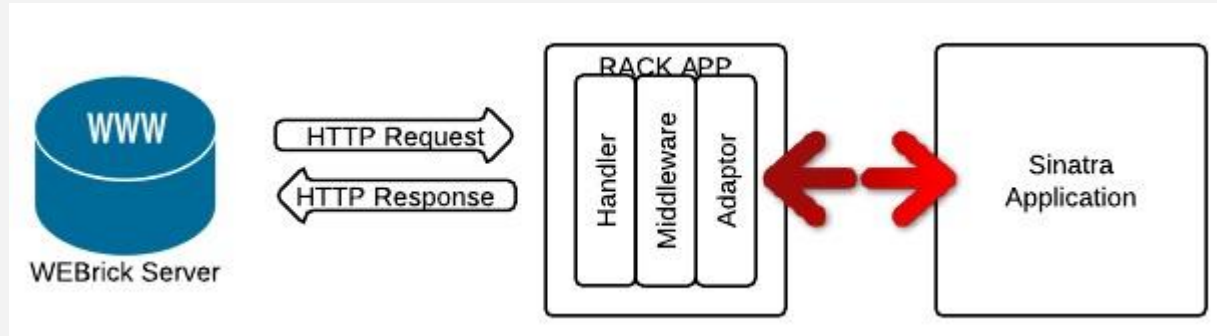
GET Get customer

POST Add customer

PATCH Update customer

PATCH U

VALUE
Value



```
class ResponseTimer
  def initialize(app, message = "Response Time")
    @app = app
    @message = message
  end

  def call(env)
    dup. call(env)
  end

  def _call(env)
    @start = Time.now
    @status, @headers, @response = @app.call(env)
    @stop = Time.now
    [@status, @headers, self]
  end

  def each(&block)
    block.call("<!-- #{@message}: #{@stop - @start} -->\n") if @headers["Content-Type"].include? "text/html"
    @response.each(&block)
  end
end
```

1. Padrino - <http://padrinorb.com/>
2. Cuba - <https://cuba.is/>
3. Scorched - <https://github.com/Wardrop/Scorched>
4. Hanami - <http://hanamirb.org/>
5. Grape - <http://www.ruby-grape.org/>
6. NYNY - <https://alisnic.github.io/nyny/>
7. Crêpe - <https://github.com/crepe/>
8. Nancy - <https://github.com/guilleiguaran/nancy>
9. Celluloid - <https://github.com/celluloid/celluloid>
10. Hobbit - <https://github.com/patriciomacadden/hobbit>

Что почитать?

http://nginx.org/ru/docs/beginners_guide.html

https://www.html5rocks.com/ru/tutorials/internals/howbrowserswork/#The_browser_main_functionality

<https://learning.postman.com/docs/postman/api-documentation/documenting-your-api/>

<https://curl.haxx.se/docs/manual.html>

[https://en.wikipedia.org/wiki/Rack_\(web_server_interface\)](https://en.wikipedia.org/wiki/Rack_(web_server_interface))

Thanks!

Any questions? Feel free to contact us hello@yalantis.com