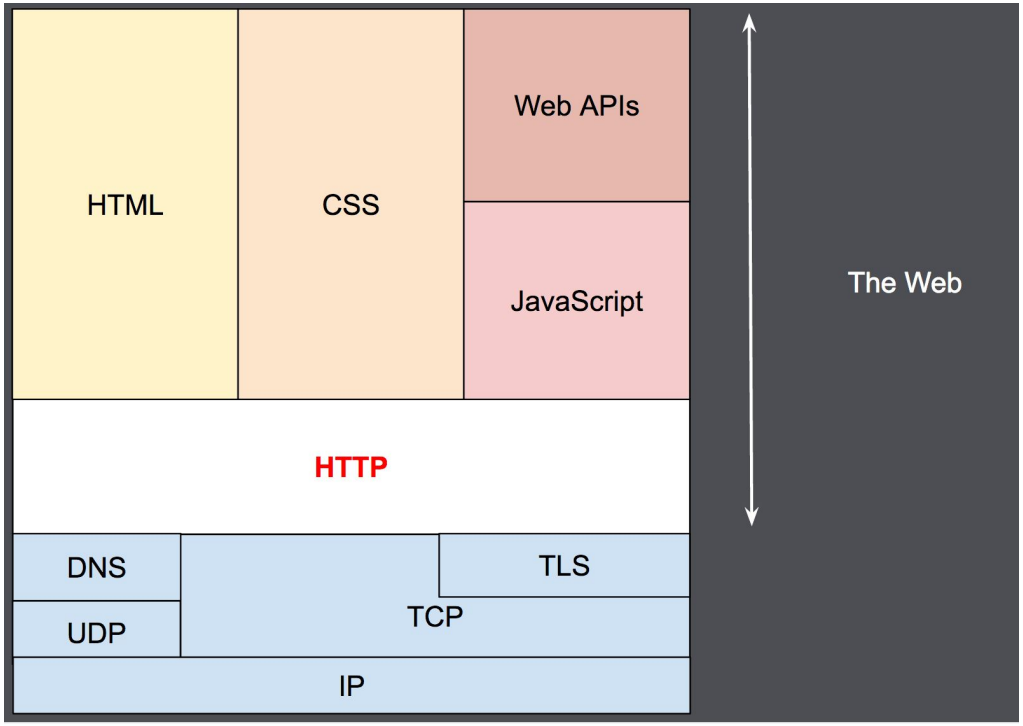
Создание веб-сервера на Ruby

HTTP протокол, простой сервер на ruby, модульный интерфейс rack

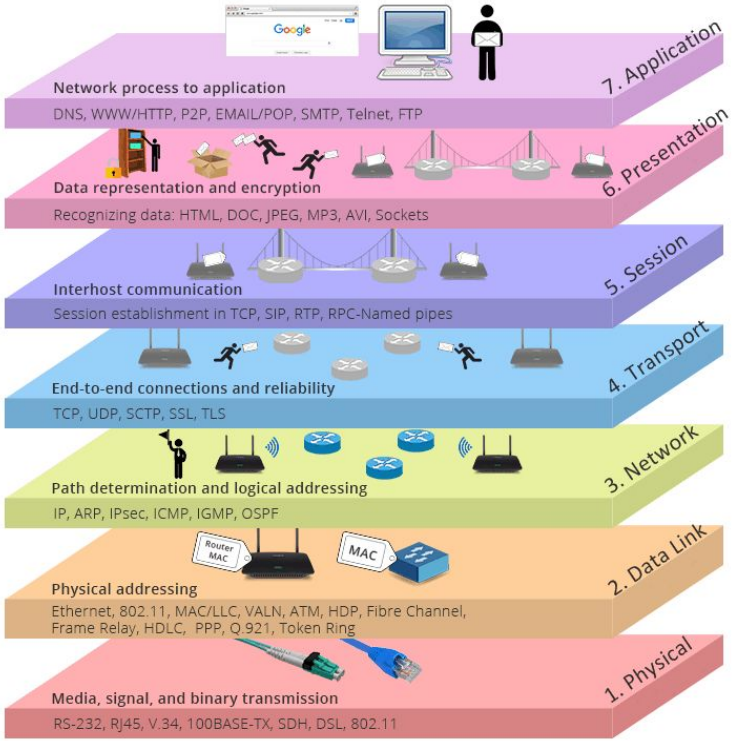
HTTP протокол

- полное название – *Hypertext Transfer Protocol*;
- является протоколом прикладного уровня;
- версии 1.0 и 1.1 разработаны в 1990-х годах, в 2010-х создана версия HTTP/2;
- не хранит состояния между двумя парами “запрос-ответ”;
- для хранения контекста можно использовать куки;

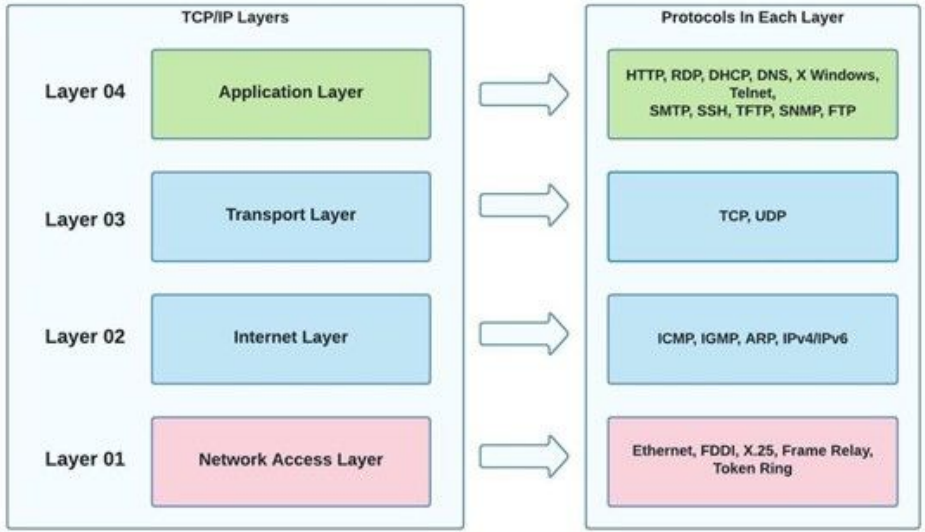


Модели передачи данных

- Модель OSI



- Модель TCP/IP



Структура HTTP запроса-ответа

Requests

```
POST / HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (Macintosh;… )… Firefox/51.0
Accept: text/html,application/xhtml+xml,…,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=-12656974
Content-Length: 345

-12656974
(more data)
```

start-line

HTTP headers

empty line

body

Responses

```
HTTP/1.1 403 Forbidden
Server: Apache
Content-Type: text/html; charset=iso-8859-1
Date: Wed, 10 Aug 2016 09:23:25 GMT
Keep-Alive: timeout=5, max=1000
Connection: Keep-Alive
Age: 3464
Date: Wed, 10 Aug 2016 09:46:25 GMT
X-Cache-Info: caching
Content-Length: 220

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML
2.0//EN">
(more data)
```

HTTP методы

OPTIONS	Доступные запросы
GET	Получение ресурса
POST	Создание ресурса
HEAD	Получение заголовков
PUT	Обновление ресурса
PATCH	Обновление части ресурса
DELETE	Удаление ресурса

Создание простого веб-сервера

```ruby
require 'socket'

class RubyWebServer
  def initialize(hostname: 'localhost', port: 5678)
    @hostname = hostname
    @port = port
    @server = TCPServer.new(hostname, port)
  end

  def start
    notify_client

    loop do
      client = @server.accept
      request = client.readpartial(2048)

      puts "\n\n", client.class, client.class.ancestors, "\n\n"
      puts request

      client.close
    end
  end

  private

  def notify_client
    puts "Server is running on #{@hostname}:#{@port}"
  end
end

RubyWebServer.new.start

# curl "localhost:5678"
# curl -X POST "localhost:5678" -H 'Content-Type: application/json'
#  -d '{"login":"my_login","password":"my_password"}'
```

Чтение данных из запроса

```ruby
require 'socket'
require_relative 'request_parser'

class RubyWebServer
  def initialize(hostname: 'localhost', port: 5678)
    @hostname = hostname
    @port = port
    @server = TCPServer.new(hostname, port)
  end

  def start
    notify_client

    loop do
      client = @server.accept
      request = client.readpartial(2048)

      parsed_request = RequestParser.new(request: request).parse
      p parsed_request

      client.close
    end
  end

  private

  def notify_client
    puts "Server is running on #{@hostname}:#{@port}"
  end
end

RubyWebServer.new.start
```

```ruby
class RequestParser
  def initialize(request:)
    @request = request
  end

  def parse
    method, path, _version = @request.lines[0].split

    { path: path, method: method, headers: headers }
  end

  private

  def headers
    headers = {}

    @request.lines[1..-1].each do |line|
      return headers if line == "\r\n"

      header, value = line.split
      header = header.gsub(':', '').downcase.to_sym

      headers[header] = value
    end
  end
end
```

Отправка ответа клиенту

```ruby
require 'socket'
require_relative 'request_parser'
require_relative 'response_builder'

class RubyWebServer
  def initialize(hostname: 'localhost', port: 5678)
    @hostname = hostname
    @port = port
    @server = TCPServer.new(hostname, port)
  end

  def start
    notify_client

    loop do
      client = @server.accept
      request = client.readpartial(2048)

      parsed_request = RequestParser.new(request: request).parse
      response = ResponseBuilder.new(request: parsed_request).build

      client.write(response)
      client.close
    end
  end

  private

  def notify_client
    puts "Server is running on #{@hostname}:#{@port}"
  end
end

RubyWebServer.new.start
```

```ruby
require_relative 'response'

class ResponseBuilder
  def initialize(request:)
    @request = request
  end

  def build
    if @request[:path] == '/'
      respond_with('views' + '/index.html')
    else
      respond_with('views' + @request[:path])
    end
  end

  private

  def respond_with(path)
    if File.exist?(path)
      Response.new(code: 200, data: File.binread(path))
    else
      Response.new(code: 404)
    end
  end
end
```

```ruby
class Response
  def initialize(code:, data: '')
    @code = code

    @response =
      "HTTP/1.1 #{code}\r\n" +
        "Content-Length: #{data.size}\r\n" +
        "\r\n" +
        "#{data}"
  end

  def to_s
    @response
  end
end
```

Запуск сервера с гемом Rack

```
1   require 'rack'
2   require 'thin'
3
4   class WebServer
5     def call(env)
6       [200, { 'Content-Type' => 'text/html' }, ['<h1>Hello, World</h1>']]
7       # [200, { 'Content-Type' => 'text/html' }, env]
8     end
9   end
10
11  Rack::Handler::Thin.run(WebServer.new)
```

Добавление middleware логики

```
1   require 'rack'
2   require 'thin'
3
4   class WebServer
5     def call(env)
6       sleep(3)
7       [200, { 'Content-Type' => 'text/html' }, ['<h1>Hello, World</h1>']]
8     end
9   end
10
11  class LoggingMiddleware
12    def initialize(app)
13      @app = app
14    end
15
16    def call(env)
17      before = Time.now.to_i
18      status, headers, body = @app.call(env)
19      after = Time.now.to_i
20      body << "Done at #{after - before} seconds"
21
22      [status, headers, body]
23    end
24  end
25
26  Rack::Handler::Thin.run(LoggingMiddleware.new(WebServer.new))
```

Конфигурационный файл config.ru

```
1   require 'thin'
2
3   app = -> (env) do
4     sleep(3)
5     [200, { 'Content-Type' => 'text/html' }, ['<h1>Hello, World</h1>']]
6   end
7
8   class LoggingMiddleware
9     def initialize(app)
10      @app = app
11    end
12
13    def call(env)
14      before = Time.now.to_i
15      status, headers, body = @app.call(env)
16      after = Time.now.to_i
17      body << "Done at #{after - before} seconds"
18
19      [status, headers, body]
20    end
21  end
22
23  use LoggingMiddleware
24  run app
25
26  # rackup -s thin
```

Создание своего фреймворка

```ruby
1  require 'rack'
2  require 'thin'
3
4  module Application
5    class Base
6      attr_reader :routes
7
8      def initialize
9        @routes = {}
10     end
11
12     def get(path, &handler)
13       route('GET', path, &handler)
14     end
15
16     private
17
18     def route(method, path, &handler)
19       @routes[method] ||= {}
20       @routes[method][path] = handler
21     end
22   end
23 end
24
25 our_app = Application::Base.new
26
27 our_app.get '/hello' do
28   [200, {}, ['Hello from App']]
29 end
30
31 puts our_app.routes
```

```ruby
1  require 'rack'
2  require 'thin'
3
4  module Application
5    class Base
6      attr_reader :routes
7
8      def initialize
9        @routes = {}
10     end
11
12     def get(path, &handler)
13       route('GET', path, &handler)
14     end
15
16     def call(env)
17       @request = Rack::Request.new(env)
18       method = @request.request_method
19       path = @request.path_info
20
21       @routes[method][path].call
22     end
23
24     private
25
26     def route(method, path, &handler)
27       @routes[method] ||= {}
28       @routes[method][path] = handler
29     end
30   end
31 end
32
33 our_app = Application::Base.new
34
35 our_app.get '/hello' do
36   [200, {}, ['Hello from App']]
37 end
38
39 Rack::Handler::Thin.run(our_app)
```

```ruby
16     def call(env)
17       @request = Rack::Request.new(env)
18       method = @request.request_method
19       path = @request.path_info
20
21       handler = @routes[method][path]
22
23       instance_eval(&handler)
24     end
25
26     def params
27       @request.params
28     end
29
30     private
31
32     def route(method, path, &handler)
33       @routes[method] ||= {}
34       @routes[method][path] = handler
35     end
36   end
37 end
38
39 our_app = Application::Base.new
40
41 our_app.get '/hello' do
42   [200, {}, ["Hello from App. You've passed next params: #{params}"]]
43 end
```

```ruby
43 our_app = Application::Base.new
44
45 our_app.get '/hello' do
46   [200, {}, ["Hello from App. You've passed next params: #{params}"]]
47 end
48
49 our_app.post '/create' do
50   [201, {}, request.body]
51 end
52
53 # curl -X POST "localhost:8080/create" -H 'Content-Type: application/json'
54 #  -d '{"login":"my_login","password":"my_password"}'
55
56 Rack::Handler::Thin.run(our_app)
```

Полезные ссылки

https://www.rubyguides.com/2016/08/build-your-own-web-server/ – Статья про создание простого веб-сервера;

https://thoughtbot.com/upcase/videos/rack – Описание создания rack-приложения;

https://thoughtbot.com/blog/lets-build-a-sinatra – Создание простого веб-фреймворка.

Конец! Спасибо!