Программно-аппаратные средства Web

Сергей Геннадьевич Синица

КубГУ, 2020

sin@kubsu.ru

Hyper Text Transfer Protocol (HTTP)

Life is a distributed object system. However, communication among humans is a distributed hypermedia system, where the mind's intellect, voice+gestures, eyes+ears, and imagination are all components.

Roy T. Fielding, 1998.

Примеры использования

- World Wide Web
- RSS
- Jabber (XMMP)
- Oauth и OpenID
- SOAP, XML-RPC, GraphQL

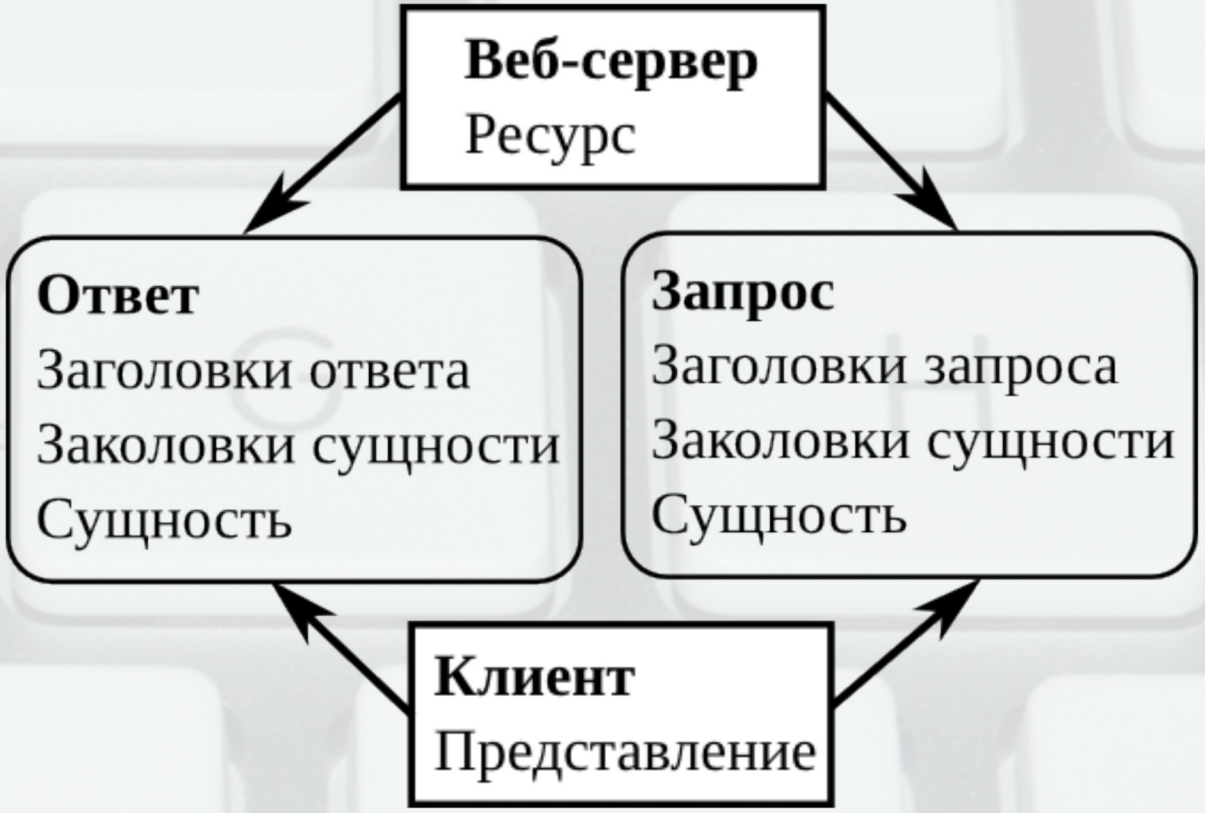
Особенности

- Без сохранения состояния
- Типизация
- Механизм обсуждения (content negotiation)

Основные понятия

- Соединение
- Запрос
- Ответ
- Ресурс
- Сущность (Entitiy)
- Представление (Representation)
- Обсуждение содержимого (Content Negotiation)
- Клиент
- Сервер
- Прокси (Proxy)
- Сервер-источник
- Прозрачный прокси
- Шлюз

Схема работы



Версии протокола

- HTTP 0.9: нет типов
- HTTP 1.0: медиа типы
- HTTP 1.1:
  - Заголовок запроса Host
  - Bad Request!
  - Работа с прокси
  - Эффективность передачи данных
  - Несколько запросов на соединение

Версии протокола

- HTTP 2.0:
  - HTTPS
  - бинарный
  - мультиплексирование
  - сжатие заголовков
  - server push
  - не нужен шардинг доменов и агрегация файлов.

Формат запроса HTTP

```
<метод> <Resource-URI> HTTP/<версия HTTP>
[<ЗаголовокЗапроса 1>: <значение1>]CRLF
...
[<ЗаголовокЗапроса n>: <значение n>]CRLF
[<ЗаголовокСущности 1>: <значение1>]CRLF
...
[<ЗаголовокСущности m>: <значение m>]CRLF
[CRLF<сущность>]
```

Примеры запросов

```
GET / HTTP/1.0

GET /about HTTP/1.1
Host: example.com
```

Формат ответа

```
HTTP/<версия HTTP> <код ответа> <сообщение>
[<ЗаголовокОтвета 1>: <значение1>]CRLF
...
[<ЗаголовокОтвета n>: <значение n>]CRLF
[<ЗаголовокСущности 1>: <значение1>]CRLF
...
[<ЗаголовокСущности m>: <значение m>]CRLF
[CRLF<сущность>]
```

Примеры ответов

```
HTTP/1.1 200 OK
Content-Length: 64
Content-Type: text/html; charset=Windows-1251

<html>
 <body>
<h1>Заголовок</h1>
Текст страницы
 </body>
</html>
```

```
HTTP/1.1 400 Bad Request
```

```
HTTP/1.1 404 Not Found
Content-Length: 44
Content-Type: text/html; charset=UTF-8

<h1>404 Ресурс не найден</h1>
```

Запрос Firefox

```
GET / HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0(X11; Linux x86_64;
rv:14.0 Gecko/20100101 Firefox/14.0.1
Accept: text/html, application/xhtml+xml,
application/xml;q=0.9, */*;q=0.8
Accept-Language: ru,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Ответ Apache

```
HTTP/1.1 200 OK

Date: Mon, 17 Sep 2012 10:45:33 GMT

Server: Apache/2.2.16 (Debian)

Last-Modified: Fri, 18 Feb 2011
20:37:42 GMT

Etag: "43356-b1-49c947c5a8482"

Accept-Ranges: bytes

Vary: Accept-Encoding

Content-Encoding: gzip

Content-Length: 146

Keep-Alive: timeout=15, max=100

Connection: Keep-Alive

Content-Type: text/html
```

```
<html><body><h1>It works!</h1>

<p>This is the default web page for
this server.</p>

<p>The web server software is

running but no content has been
added, yet.</p>

</body></html>
```

Формат запроса HTTP

```
<метод> <Resource-URI> HTTP/<версия HTTP>
[<ЗаголовокЗапроса 1>: <значение1>]CRLF
...
[<ЗаголовокЗапроса n>: <значение n>]CRLF
[<ЗаголовокСущности 1>: <значение1>]CRLF
...
[<ЗаголовокСущности m>: <значение m>]CRLF
[CRLF<сущность>]
```

Методы запроса

- GET — запрос представления ресурса
- POST — создание нового ресурса
- PUT — изменение существующего ресурса
- DELETE — удаление ресурса
- HEAD — запрос заголовков без сущности

Безопасные: GET, HEAD, OPTIONS, TRACE

Идемпотентные: безопасные + PUT и DELETE

Основные заголовки HTTP

Заголовки запроса:

- Host
- Accept
- Accept-Language
- Accept-Charset
- Accept-Encoding
- User-agent
- Referer
- Range
- Connection

Заголовки ответа:

- Content-Type
- Content-Length
- Content-Encoding
- Location
- Date
- Etag

Коды статуса ответа HTTP

1xx – подтверждения и уведомления
2xx – успешное завершение
3xx – перенаправление
4xx – ошибка клиента
5xx – ошибка сервера

Основные коды ответа HTTP

200 Ok
201 Created
206 Partial Content
302 Found
304 Not Modified
400 Bad Request
401 Unauthorized
404 Not Found
500 Internal Server Error
503 Service Unavailable

Примеры запросов при отправке HTML-форм!

```
<form action=”/” method=”GET”>
  <input name=”param1” value=”val1” />
  <input type=”submit” />
</form>
```

```
GET /?param1=val1 HTTP/1.1
Host: example.com
```

Примеры

```
<form action=”http://mysite.ru/page1” method=”GET”>
  <input name=”param1” value=”val1” />
  <input type=”submit” />
</form>
```

```
GET /page?param1=val1 HTTP/1.1
Host: mysite.ru
```

Примеры

```
<form action=”http://mysite.ru/page1” method=”GET”>
  <input name=”param1” value=”val1” />
  <input type=”hidden” name=”param2” value=”val2” />
  <input type=”submit” />
</form>
```

```
GET /page?param1=val1&param2=val2 HTTP/1.1
Host: mysite.ru
```

Примеры

```
<form action=”http://mysite.ru/page1” method=”GET”>
  <input name=”param1” value=”val1” />
  <input type=”hidden” name=”param2” value=”val2” />
  <input type=”submit” name=”submit1” value=”Button 1” />
  <input type=”submit” name=”submit2” value=”Button 2” />
</form>
```

```
GET /page?param1=val1&param2=val2&submit2=Button%202 HTTP/1.1
Host: mysite.ru
```

Примеры

```
<form action=”http://mysite.ru/page1” method=”POST”>
  <input name=”param1” value=”val1” />
  <input type=”hidden” name=”param2” value=”val2” />
  <input type=”submit” name=”submit1” value=”Button 1” />
  <input type=”submit” name=”submit2” value=”Button 2” />
</form>
```

```
POST /page HTTP/1.1
Host: mysite.ru
Content-Type: application/x-www-form-urlencoded
Content-Length: 40

param1=val1&param2=val2&submit2=Button%202
```

Hello World на PHP

```
<?php
// Отправляем правильную кодировку.
header('Content-Type: text/html; charset=UTF-8');
print('Привет, мир!');
// Выводим все полученные через POST параметры.
// если запрос 2-5 сделан правильно, то можно будет увидеть
// отправленный комментарий в ответе веб-сервера.
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
  print_r($_POST);
}
?>
Тут может быть ваш HTML!
```

Литература

1. Интернет-программирование: учебное пособие / С.Г. Синица. Краснодар: КубГУ, 2013.

2. Рекомендация RFC2616/3.2. Hypertext Transfer Protocol — HTTP/1.1. URL: http://www.w3.org/Protocols/rfc2616/rfc2616.html.

3. Fielding, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. Докторская диссертация, University of California, Irvine, 2000. URL: http://www.ics.uci.edu/fielding/pubs/dissertation/top.htm.