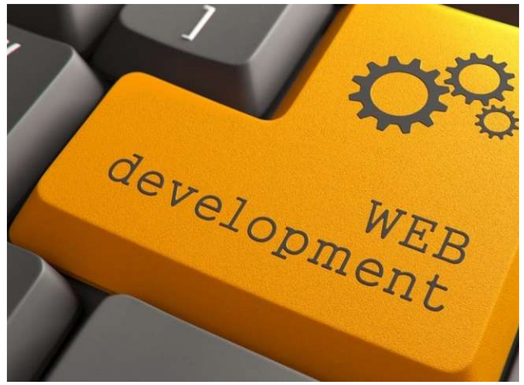
05.04.21 07:34

CSCI 165
Introduction to the Internet and the World Wide Web
Lec 4: Backend Programming III



Jetic Gū

Overview

- Focus: Web Development
- Architecture: Internet
- Core Ideas:
    1. HTTP request
    2. NodeJS cont’

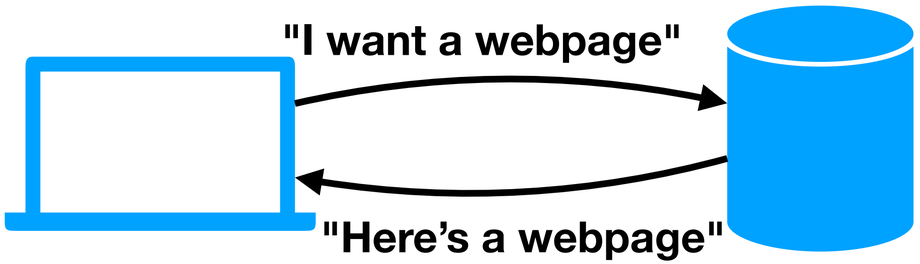
HTTP Protocol

And what are you requesting?

Summary

HTTP

- HTTP stands for Hypertext Transfer Protocol
  - Client-Server network protocol, used since the creation of WWW in 1990
  - Still used, but mostly with encryption (HTTPS, secure HTTP)
  - Client sends Server **request**;
    Server sends Client **respond**



Concept

HTTP Request

- Typical message: a passage written in English, transmitted over TCP/IP protocol

- e.g.

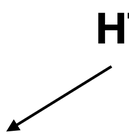
  - **GET** www.google.com/ **HTTP/1.1**
    Host: www.google.com:80
    ……

Demo

HTTP Request

- Typical message: a passage written in English, transmitted over TCP/IP protocol

- e.g.



  - **`GET`** `www.google.com/` **`HTTP/1.1`**
    `Host:` `www.google.com:80`
    ……

Demo

HTTP Request

- Typical message: a passage written in English, transmitted over TCP/IP protocol

- e.g.

  - **`GET`** `www.google.com/` **`HTTP/1.1`**
    `Host:` `www.google.com:80`
    ……

Demo

HTTP Request

- Typical message: a passage written in English, transmitted over TCP/IP protocol

- e.g.

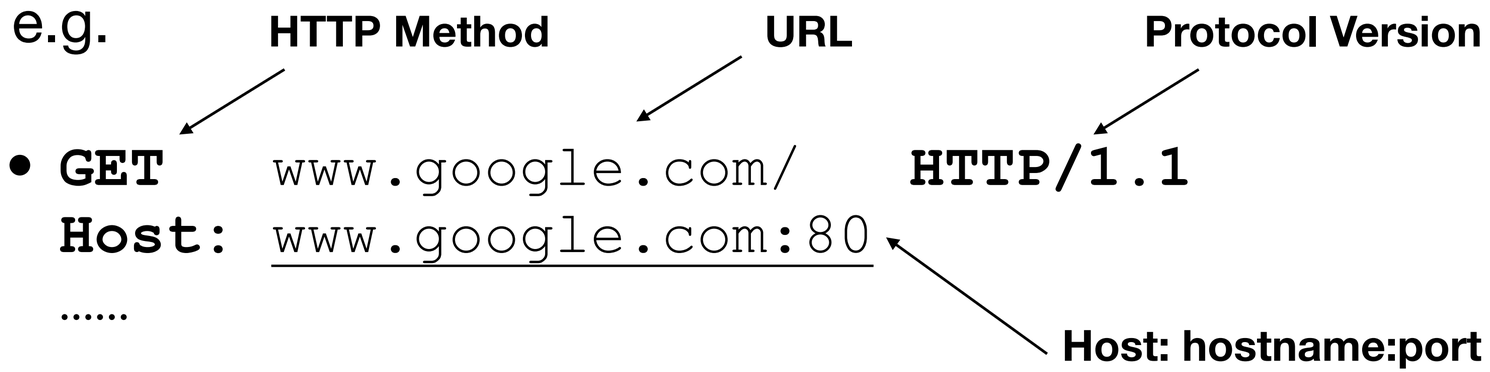
  - **GET** www.google.com/ **HTTP/1.1**
    Host: www.google.com:80
    ……

Demo

HTTP Request

- Typical message: a passage written in English, transmitted over TCP/IP protocol

- e.g.

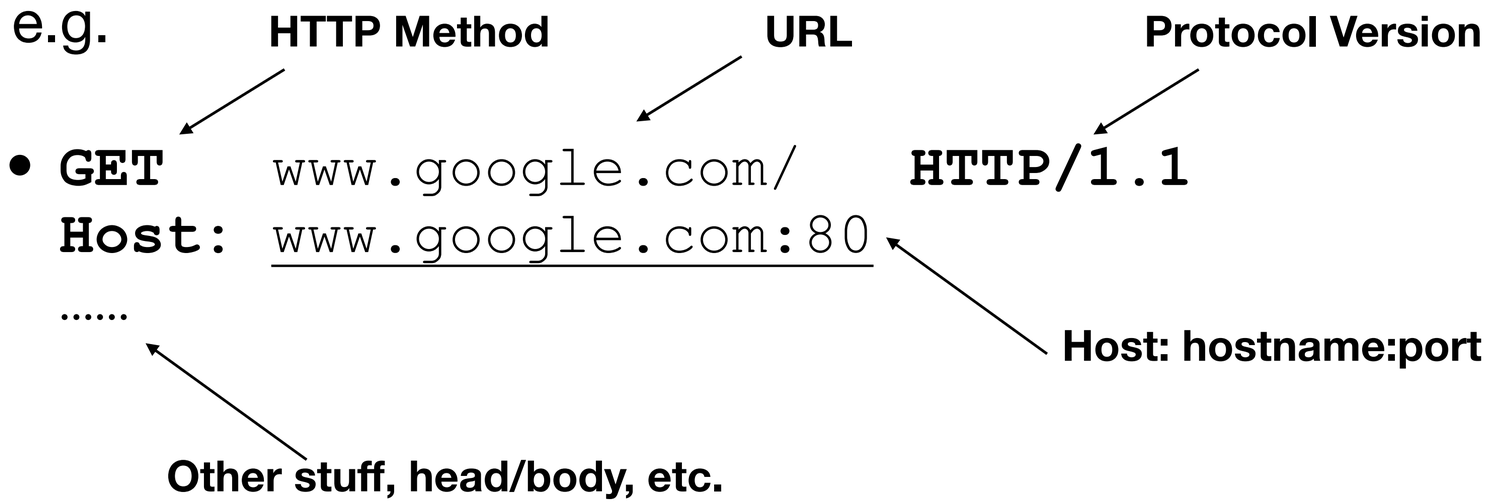


Demo

HTTP Request

- Typical message: a passage written in English, transmitted over TCP/IP protocol

- e.g.



Demo

HTTP Methods

What we are using

- **GET**: request information, expecting transmission of data (head, body, etc.)
- **HEAD**: ask for a response identical to **GET**, but expecting header only (nobody)
- **POST**: submit an entry to the server, changing the state of the server
- **PUT**, **DELETE**, **CONNECT**, **OPTIONS**, **TRACE**, **PATCH**, ...
- All of this already implemented by the browser and NodeJS, you just sit tight

Concept

P1
HTTP

NodeJS HTTP Request

- Let’s take a look at an actual request in HTTP

```
1   var http = require("http");
2   var port = 80;
3
4
5   var server = http.createServer(function (request, response) {
6       response.writeHead(200, {'Content-Type': 'text/html'});
7       response.end('<h1>Server is running!</h1>');
8
9       console.log('Request received:');
10      console.log('Request method: ', request.method);
11      console.log('Request url: ', request.url);
12      console.log('Request headers: ', request.headers);
13  })
14
15  server.listen(port, function() {
16      console.log('Server unning at http://localhost:80');
17  })
18
```

Demo

P1
HTTP

NodeJS HTTP Request

- Browser: http://localhost (equivalent to http://localhost:80)

```
Request received:
Request method:  GET
Request url:  /
Request headers:  {
  host: 'localhost',
  'upgrade-insecure-requests': '1',
  accept: 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q
  'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6) Apple
  'accept-language': 'de-de',
  'accept-encoding': 'gzip, deflate',
  connection: 'keep-alive'
}
```

Demo

P1
HTTP

NodeJS HTTP Request

- Browser: http://localhost (equivalent to http://localhost:80)

```
Request received:
Request method:  GET
Request url:  /
Request headers:  {
  host: 'localhost',
  'upgrade-insecure-requests': '1',
  accept: 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q
  'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6) Apple
  'accept-language': 'de-de',
  'accept-encoding': 'gzip, deflate',
  connection: 'keep-alive'
}
```

Demo

P1
HTTP

NodeJS HTTP Request

- Browser: http://localhost/index.html (equivalent to http://localhost:80/index.html)

```
Request received:
Request method:  GET
Request url:  /index.html
Request headers:  {
  host: 'localhost',
  'upgrade-insecure-requests': '1',
  accept: 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
  'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6) AppleWebK
  'accept-language': 'de-de',
  'accept-encoding': 'gzip, deflate',
  connection: 'keep-alive'
}
```

Demo

P1
HTTP

NodeJS HTTP Request

- Browser: http://localhost/index.html (equivalent to http://localhost:80/index.html)

```
Request received:
Request method:  GET
Request url:  /index.html
Request headers:  {
  host: 'localhost',
  'upgrade-insecure-requests': '1',
  accept: 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
  'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6) AppleWebK
  'accept-language': 'de-de',
  'accept-encoding': 'gzip, deflate',
  connection: 'keep-alive'
}
```

Demo

NodeJS Cont’

Serving HTML, CSS, JS

Summary

P2
NodeJS

Previously

- Transmission of single line of text
- Today
  - Transmission of entire webpage (html file)
  - Handling different requests (like css, js)

Review

Transmitting entire files

- Module required: `fs`, stands for file system. Allows NodeJS to read local files

```
var fs = require("fs");
```

- Send content of a file named `index.html` as

```
fs.readFile("./index.html", "UTF-8", function(err, html){
    response.writeHead(200, {"Content-Type": "text/html"});
    response.end(html);
});
```

- Grammar
  - `fs.readFile(filePath, encode, function(err, content) {`
    `..... do stuff`
    `})`

P2
NodeJS

Transmitting entire files

- Grammar
  - ```
    fs.readFile(filePath, encode, function(err, content) {
        ..... do stuff
    })
    ```

  - `filePath`: actual file path on local directory

  - `encode`: just use `"UTF-8"`

  - `function` here: if no error (`err`), do stuff with content

Technical
    ```

P2
NodeJS

Transmitting entire files

- Switching between different requests using `if` condition

```
if (request.url.match("\.js$")) {
    fs.readFile("./" + request.url, "UTF-8", function(err, content){
        response.writeHead(200, {"Content-Type": "script/javascript"});
        response.end(content);
    });
} else {
    fs.readFile("./index.html", "UTF-8", function(err, html){
        response.writeHead(200, {"Content-Type": "text/html"});
        response.end(html);
    });
}
```

If requested file is *.js

return index.html otherwise

- One could do the same thing with CSS! Just one more if condition

Technical