Web APIs

Software Architecture

Brae Webb

April 15, 2024

• Review existing networking knowledge.

- Review existing networking knowledge.
- Understand *URLs*.

- Review existing networking knowledge.
- Understand *URLs*.
- Understand *HTTP* protocol and methods.

- Review existing networking knowledge.
- Understand *URLs*.
- Understand *HTTP* protocol and methods.
- Understand *RESTful* APIs.

- Review existing networking knowledge.
- Understand *URLs*.
- Understand *HTTP* protocol and methods.
- Understand *RESTful* APIs.
- Build a basic RESTful API.

\S Networking

Application Layer

Presentation Layer

Session Layer

Transport Layer

Network Layer

Data Link Layer

Physical Layer

Application Layer

Presentation Layer

Session Layer

Transport Layer

Network Layer

Data Link Layer

Physical Layer

Application Layer Presentation Layer Session Layer Transport Layer TCP/UDP (CSSE2310) Network Layer Data Link Layer Physical Layer

TCP/UDP

Low-level with *minimal abstraction*.

TCP/UDP

Impractical for building web APIs.

Application Layer

Presentation Layer

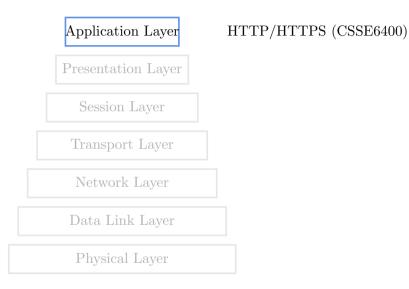
Session Layer

Transport Layer

Network Layer

Data Link Layer

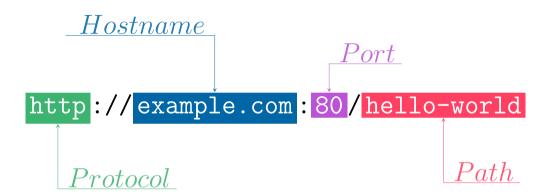
Physical Layer

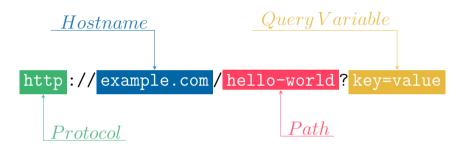


§ URLs

The anatomy of URLs

Hostname http://example.com/hello-world PathProtocol





§ HTTP

HTTP

A request-response abstraction for networking.

HTTP Request

URL An endpoint to send request to.

Method Described later.

Headers Specify type of data, e.g. JSON, HTML, etc.

Body Optional extra data to include.

HTTP Response

Status Code A number between 100 and 599 giving details about the response.

Headers Specify type of response data, e.g. JSON, HTML, etc.

Body Content of the response.

Status Codes

- 200s Indicate the request was *successful*, 200 is most common.
- 300s *Redirects* the client to another location.
- 400s Indicates that the request was wrong
 - e.g. 404 meaning that the request was for something that doesn't exist.
- 500s Indicates that the server had a problem fulfilling the request.

Types of HTTP communication
HTTP Methods

GET Query for information.

GET Query for information. POST Create resource.

GET Query for information.

POST *Create* resource.

PUT *Update* resource.

GET Query for information.

POST *Create* resource.

PUT *Update* resource.

DELETE *Delete* resource.

§ API Examples

```
>> cat app.py
   from flask import Flask
   app = Flask(__name__)
   @app.route("/")
   def hello_world():
       return "Hello, World!"
   if name == " main ":
       app.run(port=6400)
10
```

Result



```
>> cat app.js
const express = require('express')
const app = express()
const port = 6400
app.get('/', (req, res) \Rightarrow {
   res.send('Hello, World!')
})
app.listen(port, () => {
    console.log(`Example app listening on port ${port}`)
})
```

10

11

```
>> cat app.py
   from flask import Flask
   app = Flask(__name__)
   @app.route("/health")
5
   def hello_world():
       return {"status": "okay!"}
7
   if name == " main ":
       app.run(port=6400)
10
```

Result



```
>> cat app.js
const express = require('express')
const app = express()
const port = 6400
app.get('/', (req, res) => {
   res.send({"status": "okay!"})
})
app.listen(port, () => {
   console.log(`Example app listening on port ${port}`)
})
```

10

11

```
>> cat app.py
   from flask import Flask
   from flask import request
   app = Flask(__name__)
   @app.route("/echo", methods=["POST"])
   def hello world():
       return request.json.say
   if name == " main ":
10
       app.run(port=6400)
11
```

```
>>> curl -X POST \
-H "Accept: application/json" \
-H "Content-Type: application/json" \
"http://localhost:6400" \
-d '{
    "say" : "Hello, World",
}'
Hello, World
```

```
>> cat app.js
const express = require('express')
const app = express()
const port = 6400
app.post('/', express.json(), (req, res) => {
   res.send(req.body.say)
})
app.listen(port, () => {
   console.log(`Example app listening on port ${port}`)
})
```

10

11