Web APIs

Software Architecture

Brae Webb

April 14, 2025

Goals

- Review existing networking knowledge.

Goals

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

Goals

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

Goals

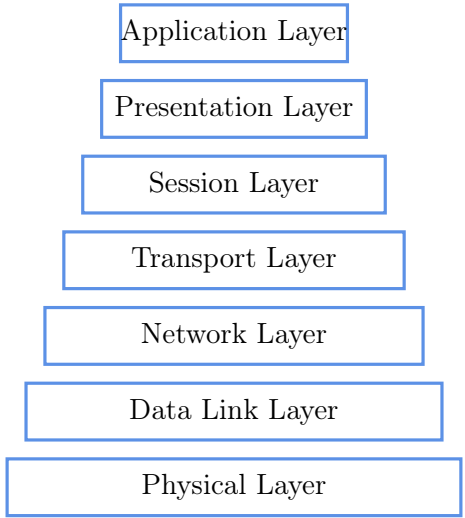
- Review existing networking knowledge.
- Understand *URLs*.
- Understand *HTTP* protocol and methods.
- Understand *RESTful* APIs.

Goals

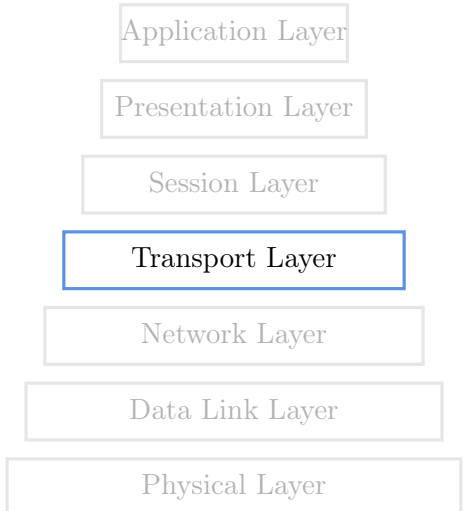
- Review existing networking knowledge.
- Understand *URLs*.
- Understand *HTTP* protocol and methods.
- Understand *RESTful* APIs.
- *Build* a basic RESTful API.

§ Networking

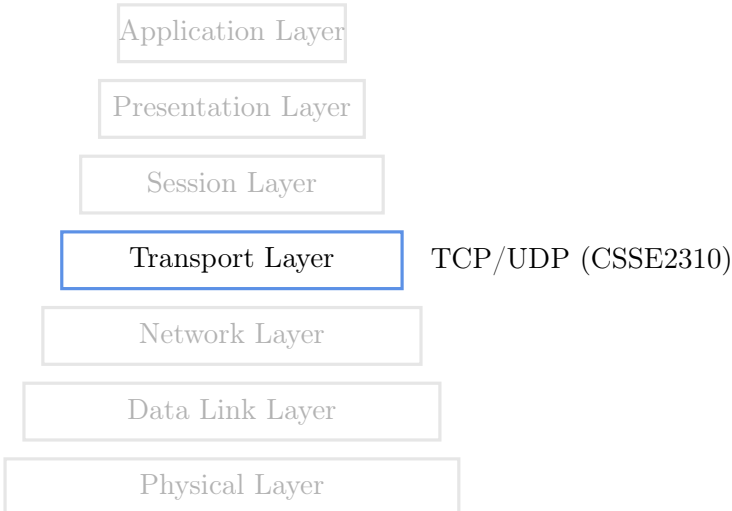
OSI Model


Application Layer
Presentation Layer
Session Layer
Transport Layer
Network Layer
Data Link Layer
Physical Layer

OSI Model


Application Layer
Presentation Layer
Session Layer
Transport Layer
Network Layer
Data Link Layer
Physical Layer

OSI Model


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

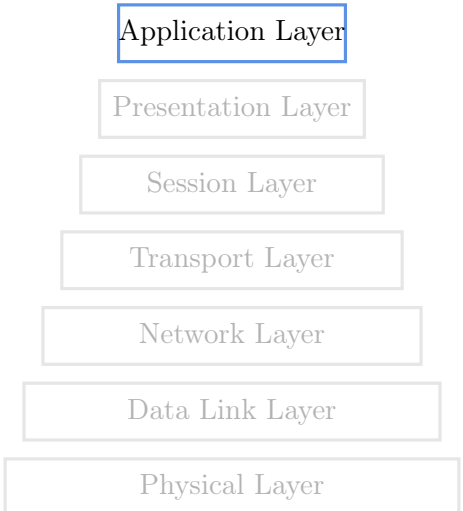
TCP/UDP

Low-level with *minimal abstraction*.

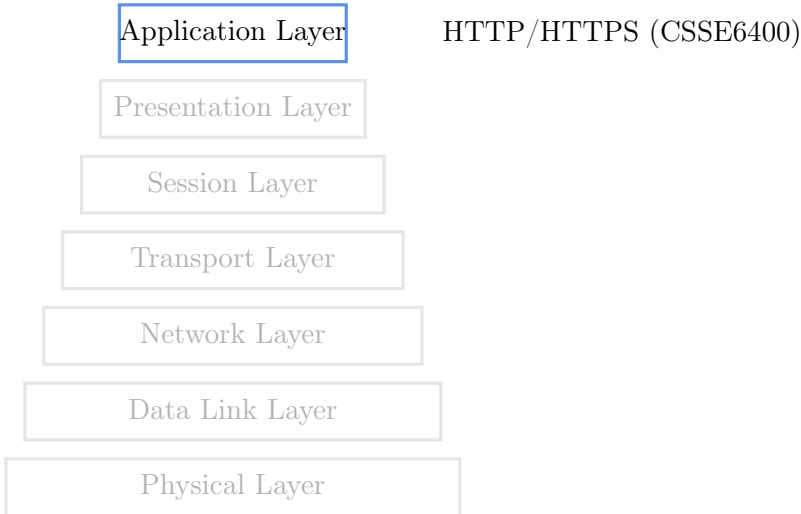
TCP/UDP

Impractical for building web APIs.

OSI Model


Application Layer
Presentation Layer
Session Layer
Transport Layer
Network Layer
Data Link Layer
Physical Layer

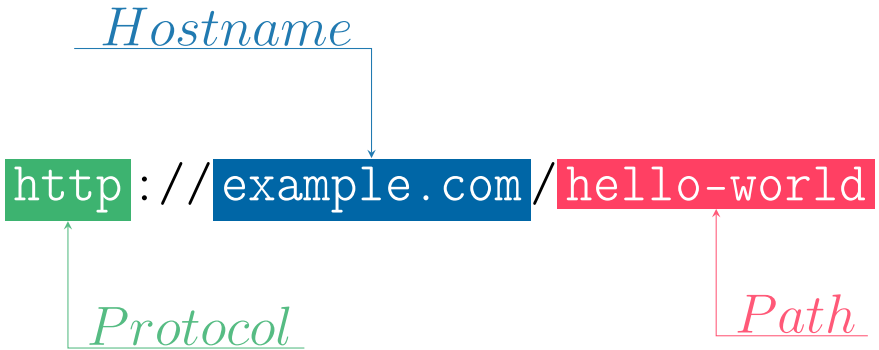
OSI Model

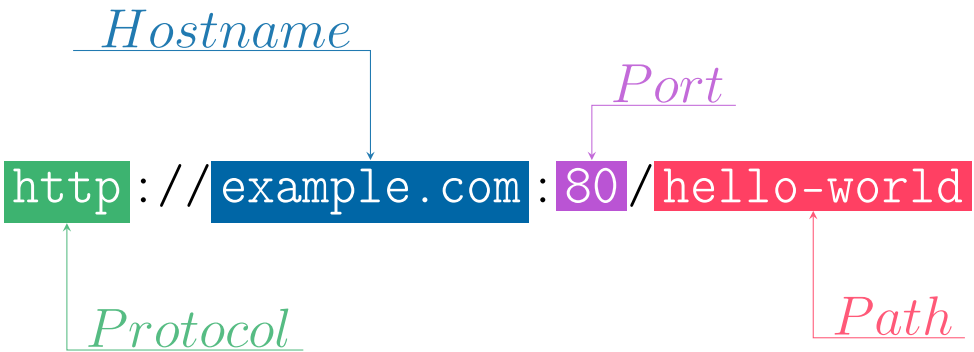

Application Layer
HTTP/HTTPS (CSSE6400)
Presentation Layer
Session Layer
Transport Layer
Network Layer
Data Link Layer
Physical Layer

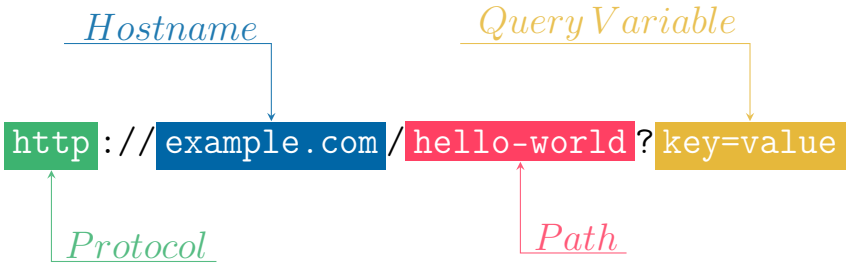
§ URLs

The anatomy of

URLs


Hostname
http://example.com/hello-world
Protocol
Path


Hostname
Port
http://example.com:80/hello-world
Protocol
Path


Hostname
Query Variable
http://example.com/hello-world?key=value
Protocol
Path

§ 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.

HTTP Methods

GET *Query* for information.

POST *Create* resource.

PUT *Update* resource.

HTTP Methods

GET *Query* for information.

POST *Create* resource.

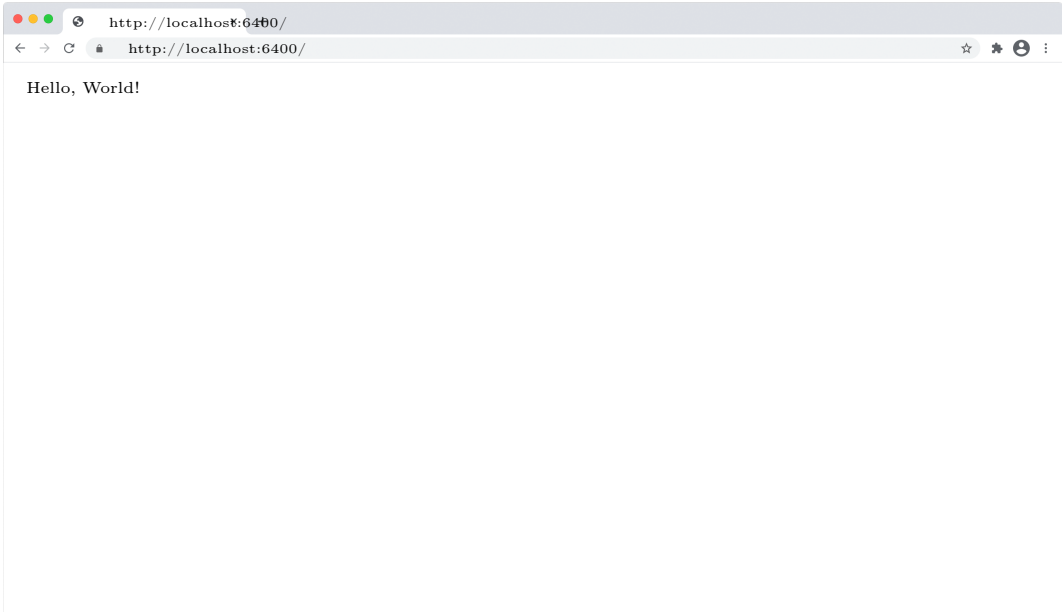
PUT *Update* resource.

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)
```

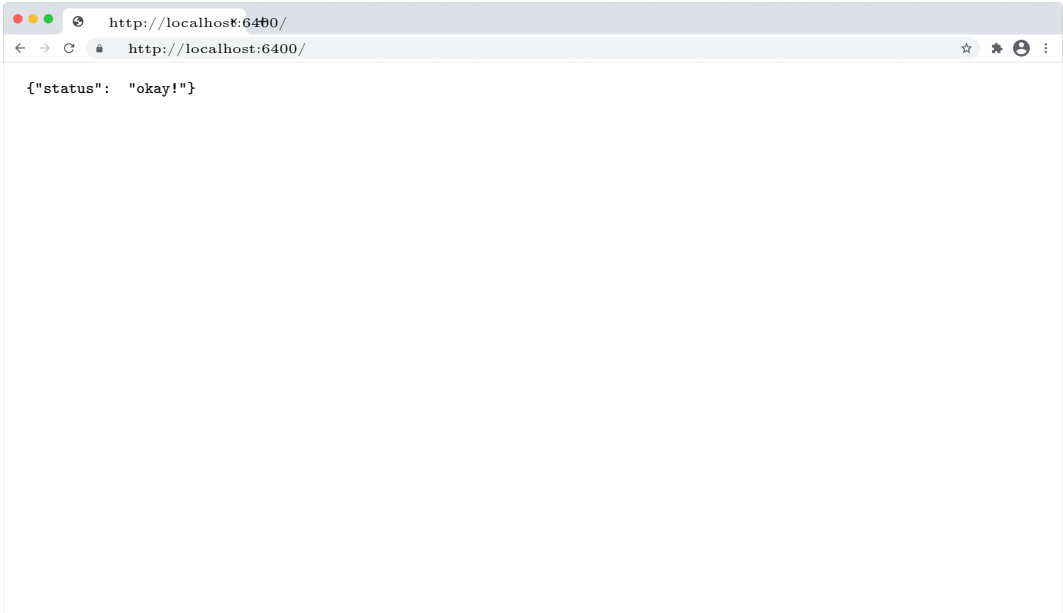
Result



```
» cat app.js

const express = require('express')
const app = express()
const port = 6400

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

app.listen(port, () => {
    console.log(`Example app listening on port ${port}`)
})
```

```
» cat app.py

from flask import Flask

app = Flask(__name__)

@app.route("/health")
def hello_world():
    return {"status": "okay!"}

if __name__ == "__main__":
    app.run(port=6400)
```

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}`)
})
```

```
» 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__":
    app.run(port=6400)
```

```
>>> 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}`)
})
```