API && Websocket

K-Digital Training

API

API(Application Programming Interface)

- 어플리케이션(소프트웨어)에서 사용할 수 있도록 운영체제나 프로그래밍 언어가 제공하는 기능 제어를 위한 인터페이스
- ex) WIN32: Windows 32bit 기반 API, JS API

Web API

- 웹 어플리케이션 개발에서 다른 서비스에 요청을 보내고 응답을 받기 위해 정의된 명세
- Gmail APi, KaKao API

SOAP

- Simple Object Access Protocol
- 구조화된 정보 전송(like XML)
- ACID(원자성, 일관성, 고립성, 지속성) 만족

Sample request

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
<m:GetStockPrice>
<m:StockName>IBM</m:StockName>
</m:GetStockPrice>
</soap:Body>

</soap:Envelope>
```

Sample Response

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
<m:GetStockPriceResponse>
<m:Price>34.5</m:Price>
</m:GetStockPriceResponse>
</soap:Body>

</soap:Envelope>
```

REST

- REpresentational State Transfer
- 2000년 Roy Fielding 의 논문에서 시작
- 자원(URL)의 행동(HTTP Method) 을 표현(Representation)

REST API의 특징

- Uniform Interface: 리소스 조작을 한정적인 인터페이스로 수행
- Stateless: 상태를 따로 관리하지 않아 서버는 요청만 처리(단순한 구현)
- Cacheable: HTTP의 캐싱을 적용
- Self-descriptive: API 메시지만으로 동작 유추 가능
- Client-Server: 서버- API, 클라이언트- Auth, Context(session)을 직접 관리하고 역할이 구분, 의존성 줄어듦

REST API Design

1. URI: 리소스의 위치 표현
2. HTTP Method: 리소스의 행동 표현

- 복수형 사용: userlist (X) / users (O)
- 동사형은 사용금지: get_user (X) / GET users/{userid} (O)
- underscore(_) 대신 hyphen(-)

HTTP Response Status Code

Status Code	Content	Status Code	Content
200	OK	201	Created
301	Moved Permanently	400	Bad Request
401	Unathorized	403	Forbidden
404	Not Found	408	Request Timeout
418	I'm a teapot	429	Too Many Requests
500	Internal Server Error	502	Bad Gateway
503	Service Unavailable	504	Gateway Timeout

Simple Express for Websocket

Express

- node.js 기반의 Web Application Framework
- `$ npm install express`

Commands

```
$ mkdir socket-express
$ cd socket-express
$ touch app.js
$ npm init
$ npm install --save express
```

First Express code

```
const express = require('express');
const app = express();
const http = require('http');
const server = http.createServer(app);

app.get('/', (req,res) => {
    res.send('Express works');
});

server.listen(3000, () => {
    console.log('Express Server is listening on localhost:3000...');
});
```

Add router

```
// app.js
const indexRouter = require("./routes/index");

app.use("/", indexRouter);
```

```
// routes/index.js
const express = require('express');
const router = express.Router();

router.route('/').get((req, res, next) => {
    return res.send("Router works!")
});

module.exports = router;
```

CSV data serve

```
// app.js
const postRouter = require("./routes/posts");

app.use("/posts", postRouter);
```

```
// routes/posts.js
const express = require('express');
const router = express.Router();

router.route('/').get((req, res) => {
    res.send("Post works!")
});

router.route('/:filename/').get((req, res) => {
    const fileName = req.params.filename;
    res.send(fileName)
});

module.exports = router;
```

Sample data

```
League,koTeamName,enTeamName,ReutersCode
Premier League,아스날,Arsenal,ARS
Premier League,맨체스터 유나이티드,Manchester United,MUN
Premier League,사우스햄튼,Southampton,SOU
La Liga,레알 마드리드,Real Madrid,MAD
La Liga,바르셀로나,Barcelona,FCB
```

Split data

```
// data.js
const fs = require('fs');

module.exports = (filePath) => {
    dataToRead = fs.readFileSync(
        filePath,
        {encoding:"utf8"}
    );

    const rows = dataToRead.split('\n');
    rows.pop();
    const headers = rows[0].split(',');

    const rowsData = [];

    rows.slice(1).forEach((row) => {
        const rowData = {};

        for (let headerIndex=0; headerIndex < headers.length; headerIndex++){
            const header = headers[headerIndex];
            rowData[header] = row.split(',')[headerIndex];
        }

        rowsData.push(rowData);
    });

    return rowsData
}
```

```
//app.js
const path = require('path');
const csv = require("../data");

router.route('/:filename/').get((req, res) => {
    const fileName = req.params.filename;
    const filePath = path.join(__dirname, "..", "data", fileName + ".csv");

    const sendData = csv(filePath);
    res.json(sendData);
});
```

Rendering with ejs

- view engines: pug, ejs, ..
- `npm install --save ejs`

```
// app.js
const path = require('path');

app.set("view engine", "ejs");
app.set("views", path.join(__dirname, "views"));
```

Static file serve & partial rendering

```
// app.js
app.use("/public", express.static(__dirname + "/public"));
```

```
<head>
  <%- include('partials/head'); -%>
</head>
```

Websocket

Request & Response

Communication channels

- simplex
- half-duplex
- full-duplex

simplex

단방향통신

> 데이터를 전송하는 방향이 정해져있는 방식

half-duplex

반이중통신

> 전송의 방향은 양방향이나 전송 순간에는 한쪽에서만 전송가능한 방식

full-duplex

양방향통신

> 동시에 송수신이 가능한 방식

Websocket

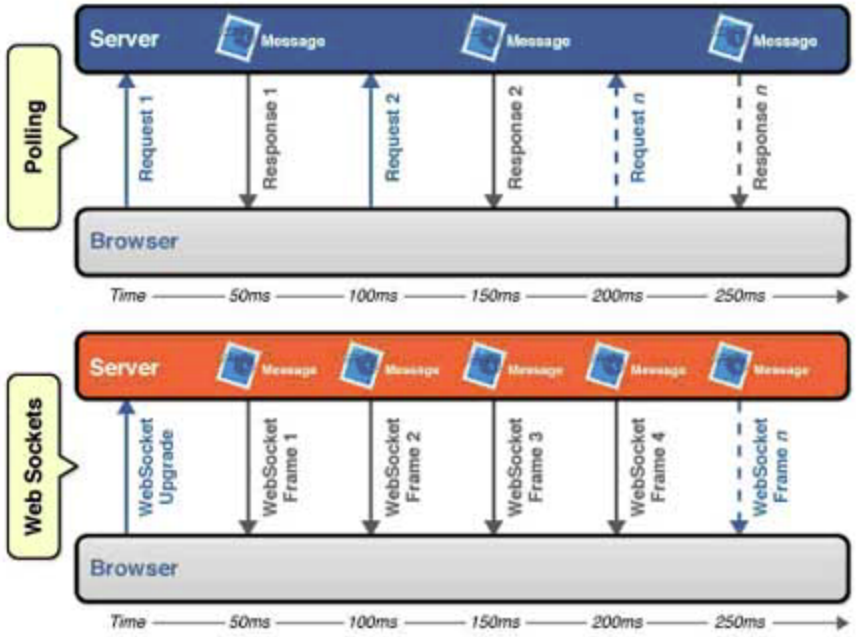
> 웹사이트가 사용자와 상호작용하기 위해 만들어진 기술

W3C가 API를 관리
port:80, HTTP1.1

Before Websocket

- HTTP Request, Response
- Hidden Frame
- Long Polling

Polling vs Websocket



Differences between Socket, Websocket

Socket - HTTP run over TCP/IP

Websocket - run from web browser

webSocket

HTML5의 표준 full-duplex 통신 방식

https://html.spec.whatwg.org/multipage/web-sockets.html#network

socket.io

```
$ npm install --save socket.io
```

- 실시간 통신기술의 웹 브라우저 호환성 문제 해결을 위한 프로젝트
- IE6 부터 최신 브라우저까지 지원
- WebSocket, Flash Socket, AJAX Long Polling, AJAX Multipart Streaming, Forever iframe, JSONP Polling 기술 모두 포함
- 브라우저에 따라 최적화된 기술 사용
- 일관성있는 문법과 API로 개발 가능

https://caniuse.com/#search=web sockets

chat.ejs

```
<style>
        body { margin: 0; padding-bottom: 3rem;
        font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Helvetica, Arial, sans-serif; }

        #form { background: rgba(0, 0, 0, 0.15); padding: 0.25rem;
        position: fixed; bottom: 0; left: 0; right: 0; display: flex;
        height: 3rem; box-sizing: border-box; backdrop-filter: blur(10px); }
        #input { border: none; padding: 0 1rem; flex-grow: 1;
        border-radius: 2rem; margin: 0.25rem; }
        #input:focus { outline: none; }
        #form > button { background: #333; border: none;
        padding: 0 1rem; margin: 0.25rem; border-radius: 3px;
        outline: none; color: #fff; }

        #messages { list-style-type: none; margin: 0; padding: 0; }
        #messages > li { padding: 0.5rem 1rem; }
        #messages > li:nth-child(odd) { background: #efefef; }
</style>

<ul id="messages"></ul>
    <form id="form" action="">
      <input id="input" autocomplete="off" /><button>Send</button>
    </form>
```

Render chatroom

```
// app.js

const chatRouter = require("./routes/chat");

app.use("/chat", chatRouter);
```

```
// routes/chat.js
const express = require('express');
const router = express.Router();

router.route('/').get((req, res, next) => {
    res.render("chatroom")
});

module.exports = router;
```

connect

```
// app.js
const socketio = require("socket.io");

const io = socketio(server);

require("./socket")(io);

app.use((req,res, next) => {
  req.io = io;
  next();
});
```

```
// socket.js
module.exports = (io) => {
  io.on('connection', (socket) => {
    console.log('a user is now connected');
  });
}
```

```
<!-- views/chatroom.ejs -->
<script src="/socket.io/socket.io.js"></script>
<script>
        const socket = io();
</script>
```

disconnect

```
module.exports = (io) => {
  io.on('connection', (socket) => {
    console.log('a user is now connected');
    socket.on('disconnect', () => {
      console.log('user disconnected');
    });
  });
}
```

emit message

```
// socket.js
io.on('connection', (socket) => {
    console.log('a user is now connected');
    socket.on('chat message', (msg) => {
      console.log('message: ', msg);
    });
```

```
// views/chatroom.ejs
const form = document.getElementById('form');
      const input = document.getElementById('input');

      form.addEventListener('submit', (e) => {
        e.preventDefault();
        if (input.value) {
          socket.emit('chat message', input.value);
          input.value = '';
        }
      });
```

broadcast

- unicast : 1:1. 출발지와 목적지가 정해진 전송
- broadcast : 네트워크에 접속된 모든 기기에 정보를 전송
- multicast : 네트워크에 접속된 기기 중 선택하여 전송

broadcast

```
//socket.js
socket.on('chat message', (msg) => {
      io.emit('chat message', msg);
      console.log('message: ', msg);
    });
```

```
// views/chatroom.ejs
const messages = document.getElementById('messages');
socket.on('chat message', (msg) => {
        const item = document.createElement('li');
        item.textContent = msg;
        messages.appendChild(item);
        window.scrollTo(0, document.body.scrollHeight);
      });
```