Node.js: validare un certificato SSL con le API di SSL Labs e i WebSocket

In questo tutorial vedremo come usare le API di SSL Labs per verificare la qualità di un certificato SSL con Node.js e i WebSocket.

Per usare le API di SSL Labs occorre effettuare chiamate ad intervalli regolari verso un medesimo endpoint per ottenere lo status attuale del processo di valutazione del certificato.

A tale scopo occorre creare una classe che ci consenta di ripetere la medesima azione senza incorrere nel rischio dei memory leak insito nell'uso diretto del metodo `setInterval()`.

```
'use strict';

const { EventEmitter } = require('events');

class Scheduler extends EventEmitter {
    constructor(action = function () {}, ms = 2000) {
        super();
        this.action = action;
        this.handle = undefined;
        this.interval = ms;
        this.on('timeout', () => {
            this.action(this);
        });
    }

    start() {
```

```
        if (!this.handle) {
            this.handle = setInterval(() =>
this.emit('timeout'), this.interval);
        }
    }

    stop() {
        if (this.handle) {
            clearInterval(this.handle);
            this.handle = undefined;
        }
    }
}

module.exports = Scheduler;
```

Usando gli eventi possiamo controllare con molta più precisione il ripetersi della medesima richiesta HTTP alle API remote. Creiamo quindi una classe per effettuare tale richiesta. Useremo i moduli got (versione 11.8.3) per le richieste HTTP e validator per la validazione dei dati.

```
'use strict';

const API_URL =
'https://api.ssllabs.com/api/v3/analyze';
const got = require('got');
const validator = require('validator');

class ApiClient {
    static fetch(host) {
        if(!validator.isFQDN(host)) {
            return Promise.reject(new Error('Invalid
parameter.'));
        }
```

```
        return got(`${API_URL}?host=${host}`).json();
    }
}

module.exports = ApiClient;
```

In ExpressJS possiamo creare un WebSocket sfruttando l'istanza del server HTTP creata dall'applicazione. Quindi riceviamo dal client il nome dell'host e restituiamo i dati che ci giungono dalle API remote ad intervalli regolari. Usiamo il modulo ws per creare e gestire i WebSocket.

```
const wsServer = new ws.Server({ noServer: true });
wsServer.on('connection', socket => {

    socket.on('message', message => {
        const host = message.toString();
        const task = new Scheduler(async timer => {
            try {
                const data = await
Client.fetch(host);
                const { status, endpoints } = data;
                socket.send(JSON.stringify({
                    status,
                    endpoints
                }));
                if(status.toLowerCase() === 'ready')
{
                    timer.stop();
                }
            } catch(err) {
                timer.stop();
            }
        }, 2000);
        task.start();
```

```
    });
});
```

Quando SSL Labs ci comunica che i test si sono conclusi (lo status finale è "READY") abbiamo accesso anche al voto finale (da A ad F) contenuto negli array degli endpoint che corrisponde all'elenco dei test effettuati.