Documentazione codice progetto DoN

Matteo Ballotta

27 maggio 2024

1 Introduzione

La seguente documentazione descrive il funzionamento del codice NodeJS contenuto nel file `server.js` e le funzionalità del logger personalizzato del progetto DirectoryOnNetwork. Tramite `express` il server gestisce le richieste HTTP ed espone una lista di file disponibili al download conenuti in una cartella scelta dall'utente.

2 Source control

Per gestire il codice sorgente è stato utilizzato `GitHub` (link alla repository). Tramite `Git` sono stati caricati i diversi cambiamenti nel corso del tempo.

3 Dipendenze

Il progetto per funzionare richiede le seguenti dipendenze:

- `express`: Framework web per NodeJS.
- `fs` (File System): Modulo per interagire con il file system.
- `logger`: Sistema di log per memorizzazione operazioni.

4 Configurazione

All'avvio del server possono essere forniti tre parametri falcoltativi (come indicato nel)

- `path`: Percorso della cartella da condividere (Default: cartella corrente).
- `server-ip`: Indirizzo IP del server (Default: `localhost`).
- `port`: Porta di ascolto del server (Default: 3000).

5 Server

Il server espone diversi percorsi che offrono varie funzionalità elencate di seguito:

5.1 /

Serve la pagina HTML che visualizza i file disponibili.

```
app.get('/', (req, res) => {
  logger.log(logger.log_level.INFO, `${req.ip} connected to
      the server`);
  res.sendFile(`${__dirname}/frontend/index.html`);
});
```

5.2 /script

Serve il file di JavaScript `script.js` che permette di stampare i file disponibili con una richiesta al server.

```
app.get('/script', (req, res) => {
  res.setHeader('Content-Type', 'application/javascript')
  res.sendFile(`${__dirname}/frontend/script.js`);
});
```

5.3 /style

Serve il file `style.css` contenente gli stili della pagina.

```
app.get('/style', (req, res) => {
  res.setHeader('Content-Type', 'text/css')
  res.sendFile(`${__dirname}/frontend/style.css`);
});
```

5.4 /get_files

Restituisce un array di tutti i file contenuti nella cartella scelta.

```
app.get('/get_files', (req, res) => {
  const filesArray = [];
  filesys.readdir(directory, {withFileTypes: true}, (err,
      files) => {
    if(err) {
      logger.log(log.log_level.ERROR, err.message);
      return;
    }

    const noDir = files
      .filter(file => file.isFile())
      .map(file => file.name);
```

```
12
13      noDir.forEach(file => {
14        filesArray.push(file);
15      });
16      res.send(filesArray);
17    });
18  });
```

5.5 /download/:filename

Permette il download al client di un file specificato. Essendo questa una parte del codice critica che potrebbe essere maggiormente esposta ad attacchi un controllo si assicura che solamente i file presenti nella cartella esposta possano essere scaricati.

```
1  app.get('/download/:filename', (req, res) => {
2    const file = req.params.filename;
3    if(file.includes('/')) {
4      logger.log(logger.log_level.WARNING, `${req.ip} tried to
           access a different folder! (Blocked)`);
5      return;
6    }
7    logger.log(logger.log_level.INFO, `${req.ip} downloaded ${
       file}`);
8    const filepath = `${directory}/${file}`;
9    res.download(filepath);
10 });
```

5.6 Avvio del Server

Il server viene avviato ascoltando sull'hostname e sulla porta specificati o default in caso non vengano forniti.

```
1  app.listen(port, hostname, () => {
2    logger.log(logger.log_level.INFO, `Server is running on
       http://${hostname}:${port}`);
3  });
```

5.7 Ottenimento dei parametri

Ottenimento dei parametri forniti o assegnazione di default.

```
1  let directory = process.argv[2] || './';
2  let hostname  = process.argv[3] || '127.0.0.1';
3  let port      = process.argv[4] || 3000;
```

6 Logger personalizzato

Al fine di creare un server che possa permettere la registrazione delle azioni degli utenti è stato necessario creare un sistema di logging che lo permettesse come mostrato di seguito. A seconda delle azioni degli utenti vengono visualizzati diversi tipi di log, ad esempio se un utente tenta di accedere a file esterni alla cartella scelta il logger lo visualizza tramite un `WARNING`.

```
const log_level = {
  INFO    : "INFO    ",
  WARNING : "WARNING ",
  ERROR   : "ERROR   "
};

function log(level, message) {
  let date = new Date();
  console.log(`[${date.toLocaleString()} - ${level}] ${
      message}`);
}

module.exports = {
  log_level,
  log
};
```

7 Front-end

Il fine del progetto è quello di creare un server che esponga una cartella su una rete locale. Viene nel sorgente fornito un front-end di esempio. Ci si aspetta però che il codice lato server venga utilizzato liberamente per essere integrato in diversi sistemi.