Node.js: richieste HTTP GET con i moduli core

In questo articolo vedremo come effettuare richieste GET usando i moduli core di Node.js.

Il primo passo da compiere è quello di effettuare il parsing dell'URL passato come parametro alla nostra funzione o al nostro metodo principale. Possiamo usare la classe JavaScript `URL` per questo scopo.

```
'use strict';

const parseURL = url => {
    try {
        return new URL(url);
    } catch(err) {
        return null;
    }
};

module.exports = parseURL;
```

Se leggiamo la documentazione noteremo che l'istanziazione di questa classe può sollevare un'eccezione di tipo `TypeError` se l'URL passato al costruttore non è valido, quindi dobbiamo gestire tale possibile eccezione.

Se l'URL è valido, abbiamo ora un oggetto contenente tre proprietà fondamentali per lo scopo che vogliamo ottenere, ossia:

1. `protocol`: il protocollo usato, fondamentale per scegliere tra i moduli core che gestiscono i protocolli HTTP e HTTPS.
2. `hostname`: il nome del dominio su cui effettuare la richiesta.

3. `pathname`: il percorso relativo sul server, ad esempio `/posts/titolo-post`.

A questo punto possiamo definire la nostra funzione o il nostro metodo principale.

```
'use strict';

const parseURL = require('./helpers/parseURL');

const request = url => {
    return new Promise((resolve, reject) => {
        const urlObj = parseURL(url);
        if(urlObj === null) {
            reject({ error: 'Invalid URL' });
        }
        const { protocol, hostname, pathname } =
urlObj;
        if(protocol !== 'http:' || protocol !==
'https:') {
            reject({ error: 'Invalid protocol' });
        }
        const moduleProtocol = protocol.replace(':',
'');
        const module = require(moduleProtocol);
        const port = moduleProtocol === 'https' ? 443
: 80;
        const options = {
            hostname: hostname,
            port: port,
            path: pathname,
            method: 'GET'
            };
            const req = module.request(options, res
```

```
=> {
            let body = '';
            res.on('data', d => {
                body += d;
            });
            res.on('end', () => { return
resolve({body}); });
            });

            req.on('error', error =>  { return
reject({ error }); });

            req.end();
    });
};
```

Se l'URL è valido, ne estraiamo i componenti rilevanti e passiamo all'identificazione del protocollo, che dovrà essere necessariamente o HTTP o HTTPS. Usiamo quindi il nome del protocollo privato del carattere `:` per includere o il modulo core `http` per il protocollo HTTP o il modulo core `https` per il protocollo HTTPS. In base al protocollo in uso, sceglieremo la porta 80 per il protocollo HTTP o la porta 443 per il protocollo HTTPS.

L'intera logica è racchiusa all'interno di una Promise, quindi saremo in grado di usare in futuro il modello async/await per gestire le richieste GET.