GABRIELE ROMANATO

Effettuare richieste HTTP in Node.js usando i moduli core

In Node.js è possibile effettuare richieste HTTP senza dipendere da librerie esterne come Axios o node-fetch, semplicemente utilizzando i moduli core `http` e `https`. Questo approccio è utile per mantenere basse le dipendenze e avere un controllo più diretto sulla gestione delle richieste.

Utilizzare il modulo http

Per effettuare una richiesta HTTP, possiamo usare il modulo `http` integrato in Node.js. Ecco un esempio di richiesta GET:

```
const http = require('http');

const options = {
  hostname: 'example.com',
  port: 80,
  path: '/',
  method: 'GET'
};

const req = http.request(options, res => {
  console.log(`Status Code: ${res.statusCode}`);

  res.on('data', chunk => {
    console.log(`Body: ${chunk}`);
  });

  res.on('end', () => {
    console.log('No more data in response.');
  });
});

req.on('error', error => {
  console.error(`Error: ${error.message}`);
```

```
});

req.end();
```

Utilizzare il modulo https

Se devi effettuare una richiesta a un server che usa HTTPS, devi usare il modulo `https`. Il codice è molto simile:

```
const https = require('https');

const options = {
  hostname: 'example.com',
  port: 443,
  path: '/',
  method: 'GET'
};

const req = https.request(options, res => {
  console.log(`Status Code: ${res.statusCode}`);

  res.on('data', chunk => {
    console.log(`Body: ${chunk}`);
  });

  res.on('end', () => {
    console.log('No more data in response.');
  });
});

req.on('error', error => {
  console.error(`Error: ${error.message}`);
});

req.end();
```

Note importanti

- Ricorda di chiamare `req.end()` per inviare effettivamente la richiesta, anche se non stai inviando dati nel corpo.
- Puoi anche aggiungere headers personalizzati all'interno dell'oggetto `options`.
- Per richieste POST o PUT, puoi scrivere i dati usando `req.write(data)` prima di chiamare `req.end()`.

Conclusione

I moduli `http` e `https` di Node.js offrono un'API abbastanza semplice e diretta per effettuare richieste web. Anche se all'inizio può sembrare più verboso rispetto a librerie più moderne, conoscere queste basi è fondamentale per comprendere meglio come funziona la comunicazione di rete in ambiente Node.js.