# Node.js: effettuare richieste ad un web service in SOAP

Per inviare una richiesta a un web service in SOAP con Node.js, è necessario seguire alcuni passaggi fondamentali.

In primo luogo, è necessario installare il modulo NPM `soap`. Questo modulo consente di interagire con i web service in SOAP in modo facile e veloce.

Dopo aver installato il modulo `soap`, è possibile creare una nuova istanza del client SOAP passando come parametro l'URL del web service. Ad esempio, il seguente codice crea un nuovo client SOAP per il web service all'indirizzo `http://www.example.com/Service.asmx`:

```
'use strict';

const soap = require('soap');
const url = 'http://www.example.com/Service.asmx?
WSDL';

soap.createClient(url, (err, client) => {
    if (err) throw err;
    console.log(client.describe());
});
```

Una volta creato il client SOAP, è possibile inviare una richiesta al web service utilizzando il metodo `call()` del client. Ad esempio, il seguente codice invia una richiesta al metodo `HelloWorld` del web service:

```
'use strict';

const soap = require('soap');
const url = 'http://www.example.com/Service.asmx?
WSDL';

soap.createClient(url, (err, client) => {
    if (err) throw err;
    client.HelloWorld((err, result) => {
        if (err) throw err;
        console.log(result);
    });
});
```

In questo esempio, la risposta del web service viene passata alla funzione di callback come parametro `result`. La risposta è un oggetto JavaScript che contiene i dati restituiti dal web service.

È anche possibile inviare parametri al servizio Web utilizzando il metodo `call()`. Ad esempio, il seguente codice invia una richiesta al metodo `Add` del web service, passando i parametri a e b:

```
'use strict';

const soap = require('soap');
const url = 'http://www.example.com/Service.asmx?
WSDL';

soap.createClient(url, (err, client) => {
    if (err) throw err;
    const args = {a: 5, b: 10};
```

```
    client.Add(args, (err, result) => {
        if (err) throw err;
        console.log(result);
    });
});
```

In questo esempio, i parametri vengono passati al servizio Web come un oggetto JavaScript.

Se si vuole evitare l'eccessivo uso di funzioni di callback, si può utilizzare il metodo util.promisify() di Node.js per usare le Promise, in quanto tutte le funzioni viste seguono lo schema *error-first callback*.