Node.js: implementare i pagamenti con Braintree in ExpressJS

In questo articolo vedremo come usare Braintree per l'elaborazione dei pagamenti con ExpressJS.

Per prima cosa creiamo un account nella sandbox di Braintree e otteniamo le nostre credenziali di accesso alle API, ossia il merchant ID e le public e private key.

A questo punto installiamo il package ufficiale.

```
npm install braintree --save
```

Inseriamo le nostre credenziali nel file di configurazione principale della nostra applicazione.

```
'use strict';

module.exports = {
    merchantId: '',
    publicKey: '',
    privateKey: ''
};
```

La soluzione di Braintree prevede l'interazione tra l'SDK JavaScript lato client e il codice lato server. Quindi nella nostra view di riferimento inseriamo l'SDK nella sezione `head` della pagina.

```
<script
src="https://js.braintreegateway.com/web/dropin/1.22.
```

```
1/js/dropin.min.js"></script>
```

Per poter funzionare, il codice lato client necessita di un token generato dal server. Nella nostra route di riferimento dobbiamo generare tale token e passarlo alla view.

```
'use strict';

const express = require('express');
const router = express.Router();
const braintree = require('braintree');
const { merchantId, publicKey, privateKey } =
require('../config');
const gateway = braintree.connect({
    environment: braintree.Environment.Sandbox,
    merchantId,
    publicKey,
    privateKey
});

router.get('/', (req, res, next) => {
    gateway.clientToken.generate({}, (err, response)
=> {
        const { clientToken } = response;
        res.render('index', { clientToken } );
    });
});

//...

module.exports = router;
```

A questo punto possiamo inserire il token generato nella view come variabile JavaScript.

```
<script>
  var clientToken = '<%= clientToken %>';
</script>
```

Il codice lato client dell'SDK usa questo token per richiedere un'istanza della classe principale con cui effettuare una richiesta di pagamento e per creare l'interfaccia di pagamento nella view. Questa richiesta a sua volta restituisce un oggetto contenente il nonce univoco della transazione. Useremo questo nonce in una richiesta AJAX alla route che gestisce la transazione passando anche l'importo da pagare.

```
"use strict";

$(function() {
    if( clientToken ) {
        var $button = $( "#request" );

        braintree.dropin.create({
            authorization: clientToken,
            container: "#dropin-container"
          }, function ( createErr, instance ) {
            $button.on( "click", function () {

              var amount = $( "#amount" ).val();
              var amt = $.trim( amount ).replace(
",", "." );

              if( !isNaN( Number( amt ) ) ) {

instance.requestPaymentMethod(function ( err, payload
) {
                    $.post( "/checkout", {
payment_method_nonce: payload.nonce, amount: amt },
```

```
function( res ) {
                        // res.status deve essere
true
                    });
                });
            }
        });
    }
});
```

Non ci resta che elaborare la transazione nella route dedicata.

```
router.post('/checkout', (req, res, next) => {

    const nonceFromTheClient =
req.body.payment_method_nonce;
    const amount = req.body.amount;

    gateway.transaction.sale({
        amount: amount,
        paymentMethodNonce: nonceFromTheClient,
        options: {
          submitForSettlement: true
        }
      }, (err, result) => {
          res.send( { status: result.success  } )
    });
});
```

Come si può notare, i pagamenti con Braintree risultano essere perfettamente bilanciati nell'interazione tra client e server.

Codice sorgente

GitHub

Documentazione

1. Testing
2. Set Up Your Client
3. Set Up Your Server