Node.js RESTFUL API

Sample Code

Rohan Banerjee

5/15/2015

I have tried to the best of my ability to build a server and client with node.js. The server and the client code interact with each other and are able to pass information back and forth.

Client Posts an Order:

```
// to use features of the http protocol. //
 var http = require("http");

// to post orders with a querystring. //
 var querystring = require("querystring");

// generate a random customer ID between 1000 and 1. //
var customerID = Math.floor(Math.random() * (1000 - 1) + 1);

// define the order object and its properties. //
 var postOrder = querystring.stringify({
        make       : 'Toyota',
        model      : 'Highlander',
        package    : 'V6 Limited',
        customer_id : customerID
});

// the options object defines who we are communicating with and how. //
 var options = {
        hostname: '127.0.0.1',
        port: 8000,
        path:'/order',
        method:'POST',
        headers:{
          'Content-Type' :'application/x-www-form-urlencoded',
          'Content-Length' : postOrder.length
        }
};

// Post the order to the server. //
var req = http.request(options, function(res) {
  console.log('STATUS: ' + res.statusCode);
  console.log('HEADERS: ' + JSON.stringify(res.headers));
  res.setEncoding('utf8');
  res.on('data', function (chunk) {
    console.log('BODY: ' + chunk);
  });
});

// On error during the communication process. //
req.on('error', function(e) {
  console.log('problem with request: ' + e.message);
});

// write data to request body. //
req.write(postOrder);
req.end();
```

Server receives an order and responds back to the client.

```
// to use features of the http protocol. //
var http = require('http');

// initialize to empty string. //
var body = "";

// create the server that will receive an order Request. //
var server = http.createServer(function(req,res) {
  res.writeHead(200, {'content-type': 'text/plain'});

  if(req.method == 'POST' && req.url=="/order"){

     // when data is successfully received, a success message is displayed. //
     res.on('data', function(data){
         body += data; // received data is appended. //
       });
     // when receiving order is complete, then log completion on server side. //
      req.on('end', function() {
         console.log("Successfully received order. ");
     });
   }
   // message to the client side. //
   res.end("We have received your order successfully. Thank you!");
});


// An error message is displayed - error event. //
server.on('error', function(e){
console.log("There is a problem with the request:\n" + e.message);
});

// server listens at the following port and localhost (IP). //
server.listen(8000, '127.0.0.1');
```

Validate Customer Address:

Select customer from the customers table with a specific customer id match. The address for the customer must not be in Siberia. There may be other places where shipping is not provided and

this may be added to the $nin list. To find customer address and verify if the address is shippable, we can do the following :-

```
// Mongodb
db.Customers.find (
                { customerId: customer_id }, { address: {$exists:true, $nin:[‘Siberia’]}}
)
```

If there is a return, then the customer who has requested an order has a shippable address. Else, the customer must be notified that the provider doesn’t ship to the address associated. A new shippable address may be requested.

Protect EndPoint

After my initial research, I found that when a client would like to access the data from our system, the process should first fire a request to a /login route (or endpoint) with a valid username and a password. We then check if the credentials are valid. If they are valid, the user is sent a user object back to the client along with an access token.

This token has an expiration time associated with it. And the client is expected to send this token to the server every time a request is made to fetch the data. This checks if the user is authenticated and doesn’t have an expired session during each and every request.

I have implemented this in C#, but haven’t had enough exposure to implement it in this new paradigm.