How to retrieve/parse the data from req.body

井民全, Jing, mqjing@gmail.com

Quick

Client

An examle to post an application/json content-type request to server.

```
needle
        .post(url, status, {json:true})
        .on('readable', function() { /* eat your chunks */ })
        .on('done', function(err, resp) {
            console.log('Ready-o!');
        })
```

Server

An example for taking the request and parsing the json body.

```
var bodyParser  = require('body-parser');

let status:StatusType = {numBodyFlag:0, numPosition:0, numFilter:0, strFWversion: 'init'};
var router = express.Router();
let resStream:Response | null = null;

let counter = 0;

// api: status/info
router.post('/info', bodyParser.json(), function(req:Request, res:Response, next) {
    console.log('[server] Got Request: /info');
```

```
    console.log('[server] received, header = ' + JSON.stringify(req.headers))
    console.log('[server] body = ' + req.body);
    console.log('[server] Content-Type = ' + req.get('content-type'));

    console.log('[server] Check json schema ...')
    const Ajv = require("ajv/dist/jtd")
    const ajv = new Ajv() // options can be passed, e.g. {allErrors: true}
    const validate = ajv.compile(StatusSchema)
        console.log('should be valid', validate(status))
        console.log('should be valid req.body', validate(req.body))
    if(!validate(req.body)){
        let strMsg:string = '[server] ERROR. validate(req.body).\nReceived:' +
JSON.stringify(req.body);
        console.log(strMsg);
        console.log('v error = ' + JSON.stringify(validate.errors));
        res.send(strMsg)
        return;
    }
    console.log('[server] JSON schema: ok.');
    status = req.body as StatusType; // update status

    // print
    for (var prop in status) {
        if (status.hasOwnProperty(prop)) {
            console.log('prop = ' + prop + ', value = ' + status[prop as keyof StatusType]);
        }
    }

    // sent event to client
    if(resStream!= null)
        resStream.write(`data: ${JSON.stringify(status)}\n\n`);
    else
        console.log('[server] Error: resStream == null. Act');

    // Response to client
    let strMsg:string = '[server] response. status = ' + JSON.stringify(status);
    console.log(strMsg);
    res.send(strMsg)
});
```

Detail

```
// The API that update the status
```

```
import express, {Request, Response} from 'express';
var bodyParser  = require('body-parser');

interface StatusType  {
   numBodyFlag: number; // "器官 V101/ VOMI = 0~4   V10V = 0~1"
   numPosition: number; // 器官點位
   numFilter:number; // B/ D/ W
   strFWversion:string; // Firmware version
}

const StatusSchema = {
   properties: {
      numBodyFlag: {type: "int32"},
      numPosition: {type: "int32"},
      numFilter: {type: "int32"},
      strFWversion: {type: "string"},
   },
};

let status:StatusType = {numBodyFlag:0, numPosition:0, numFilter:0, strFWversion: 'init'};
var router = express.Router();
let resStream:Response | null = null;

let counter = 0;

// api: status/info
router.post('/info', bodyParser.json(), function(req:Request, res:Response, next) {
   console.log('[server] Got Request: /info');
   console.log('[server] received, header = ' + JSON.stringify(req.headers))
   console.log('[server] body = ' + req.body);
   console.log('[server] Content-Type = ' + req.get('content-type'));

   console.log('[server] Check json schema ...')
   const Ajv = require("ajv/dist/jtd")
   const ajv = new Ajv() // options can be passed, e.g. {allErrors: true}
   const validate = ajv.compile(StatusSchema)
      console.log('should be valid', validate(status))
      console.log('should be valid req.body', validate(req.body))
   if(!validate(req.body)){
      let strMsg:string = '[server] ERROR. validate(req.body).\nReceived:' +
JSON.stringify(req.body);
      console.log(strMsg);
      console.log('v error = ' + JSON.stringify(validate.errors));
      res.send(strMsg)
      return;
   }
   console.log('[server] JSON schema: ok.');
   status = req.body as StatusType; // update status
```

```
    // print
    for (var prop in status) {
        if (status.hasOwnProperty(prop)) {
            console.log('prop = ' + prop + ', value = ' + status[prop as keyof StatusType]);
        }
    }

    // sent event to client
    if(resStream!= null)
        resStream.write(`data: ${JSON.stringify(status)}\n\n`);
    else
        console.log('[server] Error: resStream == null. Act');

    // Response to client
    let strMsg:string = '[server] response. status = ' + JSON.stringify(status);
    console.log(strMsg);
    res.send(strMsg)
});

// api: /status/monitor: Create server-sent-event back channel for client update status
router.get('/monitor', function(req:Request, res:Response, next) {
    console.log('[server] Got Request: /monitor');
    if (resStream != null){
        console.log('[Error] SSE already setup. Act: do not thing.');
        return;
    }
    console.log('[server] resStream = ' + (resStream));

    resStream = res;
    resStream.setHeader('Cache-Control', 'no-cache');
    resStream.setHeader('Content-Type', 'text/event-stream');
    resStream.setHeader('Access-Control-Allow-Origin', '*'); // Accept different origin
    resStream.setHeader('Connection', 'keep-alive');
    resStream.flushHeaders(); // flush the headers to establish SSE with client

    console.log('[server] sent json (status) = ' + JSON.stringify(status));
    resStream.write(`data: ${JSON.stringify(status)}\n\n`);

    // prevent browser closed the idea session.
    let interValID = setInterval(() => {
        if (counter++ >= 100) {
            clearInterval(interValID);
            (resStream as Response).end(); // terminates SSE session
            counter = 0;
            return;
        }
        console.log('----> keep-alive, [server] sent json (status)= ' + JSON.stringify(status));
```

```
        resStream.write(`data: ${JSON.stringify(status)}\n\n`);
    }, 1000 * 60);



    // If client closes connection, stop sending events
    resStream.on('close', () => {
        console.log('[server] client dropped me');
        (resStream as Response).end();
        resStream = null;
    });
});



// // api: /status/increase
// router.get('/increase', function(req:Request, res:Response, next) {
//     status.num++;
//     console.log('/increase. status = ' + JSON.stringify(status));
//     if (resStream == null){
//         let strMsg:string = '[server] Warn::SSE do not setup, yet. Please click [Start Monitor].
Act: do not thing.';
//         console.log(strMsg);
//         res.send(strMsg)
//         return;
//     }
//     resStream.write(`data: ${JSON.stringify(status)}\n\n`);
//     let strMsg:string = '[server] status increase ok.';
//     console.log(strMsg);
//     res.send(strMsg)
// });

// // api: status/update
// router.get('/update', function(req:Request, res:Response, next) {
//     // Step 1: Retrieve the parameters from client
//     let receivedNum = req.query.num;   // get the para: num
//     let receivedMsg = req.query.msg;   // get the para: Msg
//     status.num = parseInt(receivedNum as string); // update
//     status.msg = receivedMsg as string; // update
//     console.log('/update. status = ' + JSON.stringify(status));

//     // Step 2(opt): Sent back to client (SSE part)
//     if (resStream == null){
//         let strMsg:string = '[server] Warn::SSE do not setup, yet. Please click [Start Monitor].
Act: do not thing.';
//         console.log(strMsg);
//         res.send(strMsg)
//         return;
```

```
//     }
//     resStream.write(`data: ${JSON.stringify(status)}\n\n`);

//      // Step 3: Response to client
//     let strMsg:string = '[server] status update ok.';
//     console.log(strMsg);
//     res.send(strMsg)
// });

module.exports = router;
```