TEMA 3 EXPRESS.JS

Parte II – Express aplicado a servicios REST

II.2 Ejemplo práctico de Enrutamiento Simple

Estructura de archivos y carpetas

- Vamos a realizar un ejemplo usando Mongoose. Crearemos una carpeta llamada “PruebaContactosExpress”
- Instalaremos Express y Mongoose en ella.
- Tendremos:
  - Fichero principal app.js
  - Carpeta models: Y dentro contacto.js. Tendremos los modelos de datos

Modelo

- En models/contacto.js pondremos el modelo que ya conocemos:

```
const mongoose = require('mongoose');
let contactoSchema = new mongoose.Schema({
    nombre: {
        type: String,
        required: true,
        minlength: 1,
        trim: true
    },
    telefono: {
        type: String,
        required: true,
        trim: true,
        match: /^\d{9}$/
    },
    edad: {
        type: Number,
        min: 18,
        max: 120
    }
});
let Contacto = mongoose.model('contacto', contactoSchema);
module.exports = Contacto;
```

Esqueleto servidor principal

```
const express = require('express');
const mongoose = require('mongoose');
const Contacto = require('./models/contacto');

mongoose.Promise = global.Promise;
mongoose.connect('mongodb://localhost:27017/contac
    tos',
{useMongoClient: true});

let app = express();

app.listen(8080);
```

Definir los Servicios

- Realizaremos 3 servicios:
  - Listado de todos los contactos
  - Listado de un contacto a partir de su *id*
  - Inserción de un nuevo contacto

Listados de todos los contactos

- Atenderemos por GET a la URI */contactos*, y en el código haremos un find de todos los contactos, devolviéndolo directamente como resultado:

```
app.get('/contactos', (req, res) => {
    Contacto.find().then(resultado => {
        res.send(resultado);
    }).catch (error => {
        res.send([]);
    });
});
```

- Observad que, gracias a Express, no es necesario transformar a JSON el resultado obtenido. Se encarga el propio framework automáticamente.

Ficha de un contacto por ID

- Veamos ahora cómo procesar con Express URIs dinámicas.
- En este caso, accederemos por GET a una URI con el formato */contactos/:id*
- Si especificamos la URI con ese mismo formato en Express, automáticamente podemos acceder a él con el objeto **req.params** de la petición:

```
app.get('/contactos/:id', (req, res) => {
    Contacto.findById(req.params.id).then(resultado => {
        if(resultado)
            res.send({error: false, resultado: resultado});
        else
            res.send({error: true,
            mensajeError: "No se han encontrado contactos"});
        }).catch (error => {
            res.send({error: true,
            mensajeError: "Error buscando el contacto indicado"});
    });
});
```

Ficha de un contacto por ID II

- En el caso de querer pasar los parámetros en la *query string* (es decir, por ejemplo, */contactos?id=XXX*) no hay forma de establacer dichos parámetros en la URI del método get.
- En ese caso deberemos comprobar si existe el parámetro correspondiente dentro del objeto **req.query**:

```
app.get('/contactos', (req, res) => {
    if(req.query.id) {
    // Buscar por id
    }
    else {
    // Listado general de contactos
    });
});
```

Insertar un nuevo contacto

- Como siempre recibiremos en el cuerpo de la petición los datos del mismo (nombre, teléfono y edad) en formato JSON.
- En esta ocasión, vamos a valernos de un *middleware* para Express llamado ***body-parser***, que facilita el procesamiento del cuerpo de las peticiones para acceder directamente a los datos que se envían en ella.
- En primer lugar, deberemos instalar el módulo en nuestro proyecto:
  - **npm install body-parser**

Insertar un nuevo contacto II

- Incluimos con require junto al resto:

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
...
```

- Añadimos como *middleware* con app.use, justo después de iniciarlizar la app. Como lo que vamos a hacer es trabajar con objetos JSON, añadiremos el procesador JSON de *body-parser*:
  - let app = express();
  - **app.use(bodyParser.json());**
  - ...

Insertar un nuevo contacto III

- Ahora vamos a nuestro servicio POST. Observa qué sencillo queda empleando este *middleware*:

```
app.post('/contactos', (req, res) => {
    let nuevoContacto = new Contacto({
        nombre: req.body.nombre,
        telefono: req.body.telefono,
        edad: req.body.edad
    });
    nuevoContacto.save().then(resultado => {
            res.send({error: false, resultado: resultado});
    }).catch(error => {
            res.send({error: true,
            mensajeError: "Error al insertar contacto"});
    });
});
```

Insertar un nuevo contacto IV

- Al principio, construimos el contacto a partir de los datos JSON que llegan en el cuerpo, accediendo a cada campo por separado con req.body.nombre_campo, gracias a *body-parser*.
- El resto del código es el que ya conoces de ejemplos previos con Mongoose (llamada a save y procesamiento de la promesa)

Otras opciones de body-parser

- Existen otras posibilidades de uso del *middleware body-parser*.
- En el ejemplo anterior lo hemos empleado para procesar cuerpos con formato JSON.
- Pero es posible también que empleemos formularios tradicionales HTML, que envían los datos como si fueran parte de una *query-string*, pero por POST:
- *id=12&nombre=Nacho*

Otras opciones de body-parser II

- Para procesar contenidos de este otro tipo, basta con añadir de nuevo la librería como *middleware*, indicando en este caso otro método:

  ```
  app.use(bodyParser.json());
  app.use(bodyParser.urlencoded({extended:false}));
  ```

- El parámetro extended indica qué tipo de *parser* queremos utilizar para procesar los datos de la petición:
  - FALSE: Emplearemos la librería *querystring*,
  - TRUE: Ee empleará la librería *qs*, con algunas opciones algo más avanzadas para incluir objetos más complejos.
- En cualquier caso, (En el postman) deberemos asegurarnos de que el tipo de contenido de la petición se ajusta al *middleware* correspondiente:
  - Para peticiones en formato JSON, el contenido deberá ser *application/json*
  - Mientras que para enviar los datos del formulario en formato *query-string*, el tipo deberá ser *application/x-www-form-urlencoded*.