
SENPAI
ACADEMY

Formularios

SENPAI
ACADEMY

Node.js - Formularios HTML

En HTML tenemos una etiqueta ***<form>*** que representa una sección interactiva que permite a los usuarios enviar información a un servidor web. Estos campos van a ser completados por los usuarios, e incluso pueden cargar archivos.

En el ***<form>*** determinamos los tipos de datos a ingresar, pero del lado del servidor es que decidimos cómo manejarlos.

El atributo ***name*** de cada campo del formulario será el ***ID*** con que nos llegue la información.

```
<form>
  <input id="name" type="text" name="name" placeholder="Ingrese su nombre">
  <input id="lastName" type="text" name="lastName" placeholder="Ingrese su apellido">
  <input id="email" type="text" name="email" placeholder="Ingrese su correo">

  <button id="enviar">Enviar!</button>
</form>
```

SENPAI ACADEMY

Node.js - Formularios HTML - Atributos

method

El método HTTP que el navegador utiliza para enviar el formulario:

- ***post***: Corresponde al método *POST HTTP*, por lo que los datos del formulario van a ir incluidos en el body de la request.
- ***get***: Corresponde al método *GET HTTP*, por lo que los datos del formulario son adjuntados a la URI del atributo action con un '?' cómo separador. Esto se conoce como "Query Params"

action

La URI (endpoint) del servidor que procesará la información enviada en este formulario.

enctype

Cuando el valor del atributo method es **post**, este atributo corresponde al tipo MIME del contenido que se enviará al servidor.

- ***application/x-www-form-urlencoded***: El valor por defecto si el enctype no está especificado.
- ***multipart/form-data***: Usar este valor si se está usando el elemento con el atributo type "file".
- **text/plain**

SENPAI ACADEMY

Node.js - Formularios HTML - Atributos

```html
<form method="POST" action="http://localhost:3000/contacto">
  <input id="name" type="text" name="name" placeholder="Ingrese su nombre">
  <input id="lastName" type="text" name="lastName" placeholder="Ingrese su apellido">
  <input id="email" type="text" name="email" placeholder="Ingrese su correo">

  <button id="enviar">Enviar!</button>
</form>
```

SENPAI ACADEMY

Node.js - Express y los formularios

Ya vimos que para recibir el contenido directamente en el *body* de la request, en Express vamos a utilizar el middleware **body-parser**

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

const app = express();

// parse application/x-www-form-urlencoded
app.use(bodyParser.urlencoded({ extended: false }));

// parse application/json
app.use(bodyParser.json());
```

Una vez que lo colocamos como middleware global, vamos a poder acceder a ***request.body*** en todos nuestros endpoints, accediendo así al contenido de los inputs que haya cargado el usuario en los formularios del Frontend.

SENPAI ACADEMY

Archivos

SENPAI
ACADEMY

Node.js - Subir archivos

En nuestro servidor Express podemos soportar la carga de archivos desde un formulario HTML.

Para esto, el formulario debe tener:

- **action**: El endpoint que resolverá la subida del archivo
- **method**: *POST*
- **enctype**: *multipart/form-data*

Además, debemos tener un campo ***<input>*** con el atributo **type="file"** en el formulario HTML.

```html
<form method="POST" action="http://localhost:3000/profile" enctype="multipart/form-data">
  <input id="cv" name="avatar" type="file" placeholder="Agregue su CV" />
  <button id="enviar">Enviar!</button>
</form>
```

SENPAI ACADEMY

Node.js - Subir archivos

Debemos realizar unos cambios del lado de nuestro backend para soportar la carga de archivos.

Vamos a utilizar un nuevo middleware para procesar este encoding: **multer** (link)

```
npm install multer
```

Una vez instalado debemos requerirlo e inicializarlo. Con la opción **dest** indicamos en qué directorio se almacenarán los archivos subidos.

```
const multer = require('multer');
const uploadMiddleware = multer({ dest: 'uploads/' });
```

SENPAI ACADEMY

Node.js - Subir archivos

multer no funciona como un middleware global, sino que lo configuramos en cada endpoint que sea necesario:

```
app.post('/profile', uploadMiddleware.single('avatar'), function (req, res) {
  // req.file es el archivo 'avatar'
  // req.body tendrá los demás campos de texto del formulario
  console.log(req.file);
});

app.post('/photos/upload', uploadMiddleware.array('photos', 4), function (req, res) {
  // req.files es un array con los archivos 'photos', y limita en 4 la cantidad
  // req.body tendrá los demás campos de texto del formulario
  console.log(req.files);
});
```

SENPAI ACADEMY

Node.js - Subir archivos

Podemos definir la configuración antes, y combinar más de un campo de archivo por endpoint:

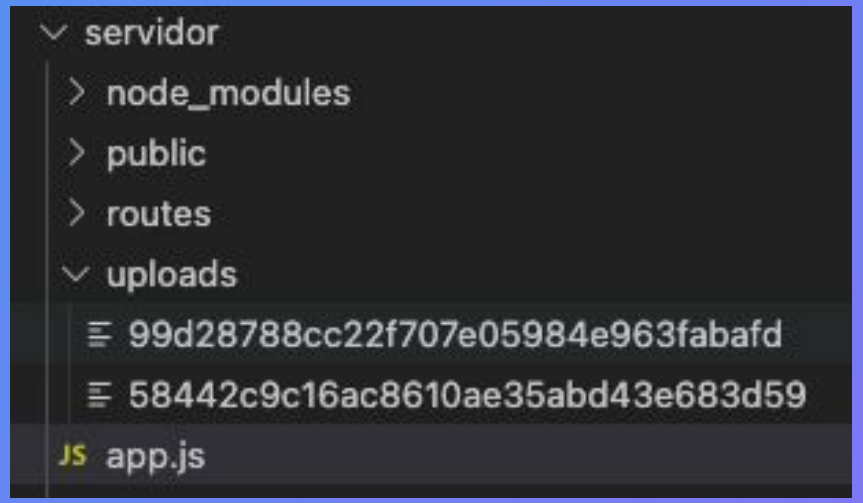
```
const advUpload = uploadMiddleware.fields([
  { name: 'avatar', maxCount: 1 },
  { name: 'gallery', maxCount: 8 }
]);

app.post('/personalPage', advUpload, function (req, res) {
  // req.files es un objeto (String =>  Array)
  //    req.files['avatar'][0] => Primer archivo
  //    req.files['gallery'] => Array con la lista de archivos
  //
  // req.body tendrá los demás campos de texto del formulario
  console.log('avatar', req.files['avatar'][0]);
  console.log('gallery', req.files['gallery']);
  console.log('body', req.body);
});
```

SENPAI ACADEMY

Node.js - Subir archivos

Si solo almacenamos los archivos, perderemos información. multer no guarda el nombre original del archivo ni su extensión.
Esta información debemos guardarla en una base de datos.

```
avatar {
  fieldname: 'avatar',
  originalname: 'fsd7.png',
  encoding: '7bit',
  mimetype: 'image/png',
  destination: 'uploads/',
  filename: '99d28788cc22f707e05984e963fabafd',
  path: 'uploads/99d28788cc22f707e05984e963fabafd',
  size: 122722
}
gallery [
  {
    fieldname: 'gallery',
    originalname: 'fsd6.png',
    encoding: '7bit',
    mimetype: 'image/png',
    destination: 'uploads/',
    filename: '58442c9c16ac8610ae35abd43e683d59',
    path: 'uploads/58442c9c16ac8610ae35abd43e683d59',
    size: 33760
  }
]
```



SENPAI ACADEMY

Node.js - Descarga de archivos

Además del middleware *express.static*, express nos brinda 2 herramientas más para enviar archivos del servidor al cliente:

res.sendFile (path [,options] [, fn])

Transfiere el archivo según el path que se pasa como parámetro. Coloca el header Content-Type basado en la extensión del nombre del archivo.
Información de las opciones: link

res.download (path [, filename] [, options] [, fn])

Transfiere el archivo como un “adjunto”. Usualmente, los navegadores tratarán el archivo como una descarga.
Por detrás utiliza *res.sendFile()* para transferirlo.
Información de las opciones: link

SENPAI ACADEMY

SENPAI
ACADEMY

gustavguez

gustavguez

GUSTAVO RODRIGUEZ

FULL STACK DEVELOPER
SOLCRE