Procesamiento POST

DigitalHouse >
Coding School

Manejar peticiones POST

Comúnmente usamos el método post para:

- **Enviar información** sensible al servidor.
- **Crear** un nuevo recurso.

Cuando definimos una ruta podemos hacerlo directamente sobre la **ejecución de Express**, implementar un **sistema de ruteo** y también **incorporar controladores** que se encarguen de manejarlas.

En todos los casos las rutas recibirán dos parámetros:

- Un string con la ruta que estaremos procesando.
- Un callback donde definiremos qué lógica ejecutaremos cuando el cliente pida esa ruta.

Manejar peticiones POST

En un contexto en donde quisiéramos agregar una nueva película a nuestro sistema, tendríamos que crear **dos rutas**: una que muestre el formulario de creación y otra que se encargue de procesar la información.

```
{}
```

```
// ruta que envía un formulario a la vista → GET
router.get('/pelicula/crear', (req,res) => {res.render('crear')});

// ruta que procesa la información del formulario → POST
router.post('/pelicula/crear', (req,res) => {...});
```

*Los nombres de las rutas pueden **ser iguales** porque cada una está implementando un método diferente.*

Configurar el formulario

Para enviar peticiones por **POST** es necesario tener un formulario. Para que ese formulario envíe los datos, debemos configurar dos propiedades:

- **method** → define el **método** HTTP que usaremos para enviar la información, en este caso es POST. Si no configuramos el **method**, por defecto será GET.
- **action** → define la **ruta** a donde viajará esa información para ser procesada. Si no configuramos el **action**, por defecto sería la misma URL donde se encuentra el formulario.

```html
<form method="POST" action="/pelicula/crear">
    ...
</form>
```

Capturar la información

Para poder trabajar con los datos que se envían desde el formulario es necesario **configurar** el entorno de nuestra aplicación para que sea capaz de **capturar** esa información.

Si estamos trabajando con `express-generator`, esta configuración se creará por defecto durante la instalación.

De lo contrario, tendremos que agregar estas dos líneas de código en el archivo `app.js` :

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

De esta forma le estamos **aclarando** a la aplicación que todo aquello que **llegue** desde un **formulario**, queremos capturarlo en forma de **objeto literal**.

Y, a su vez, tener la posibilidad de **convertir** esa **información** en un formato **JSON**, en caso de necesitarlo.

req.body

En el **request** de la petición encontramos la propiedad `body`, un objeto literal que contendrá **toda** la información del formulario:

- El nombre de cada **clave** de ese objeto, será el nombre del atributo `name` de cada input del formulario.
- El **valor** será el dato que se haya ingresado en ese campo.

html

```html
<form method="POST" action="/pelicula/crear">
    Título: <input type="text" name="titulo" value="Batman">
    ...
</form>
```

{}

```
router.post('/pelicula/crear', (req,res) => {
    console.log(req.body) // { titulo: Batman }
});
```

res.redirect()

Para **cerrar** el ciclo del **request** y **response** que hace el servidor es necesario hacer un **redireccionamiento** —después de implementada la lógica— usando el método **redirect()** sobre el **response**.

En el caso de la película que estábamos creando, luego de la creación tal vez queramos llevar al usuario al listado de películas.

{}

```
router.post('/pelicula/crear', (req,res) => {
    // Procesamos la información enviada por el usuario
    // Si todo sale bien, lo redirigimos al listado de películas.
    res.redirect('/peliculas');
});
```


DigitalHouse>
Coding School