

CONTENIDO ESTÁTICO (PRÁCTICA)

Tiempo estimado: 15min

El objeto de esta práctica es asentar y consolidar los conceptos presentados en la parte teórica de la lección.

Al finalizarla, el estudiante:

- Habrá servido contenido estático mediante el *middleware* `serve-static`.
- Habrá configurado varios directorios públicos mediante `serve-static`.
- Habrá configurado el *middleware* `serve-favicon` para servir el favicono de la aplicación.

objetivos

En esta práctica, vamos a crear una aplicación `Express` que sirva contenido estático mediante el *middleware* `serve-static`. En un primer intento, usaremos un único directorio público; en el segundo, dos.

creación del proyecto

Para comenzar, vamos a crear el proyecto de la aplicación:

1. Abrir una consola.
2. Crear el directorio de la práctica.
3. Ir al directorio de la práctica.
4. Crear el archivo `package.json` con el siguiente contenido:

```
{
  "name": "express-app",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "start": "node ./app.js"
  },
  "dependencies": {
    "express": "*"
  }
}
```

5. Crear el archivo `app.js` con el siguiente contenido:

```
"use strict";

//imports
const express = require("express");

//app
const app = express();
const index = "<!doctype html>\n" +
  "<html>\n" +
  "<head><title>Express app</title></head>\n" +
  "<body><p>¡Hola Mundo!</p></body>\n" +
  "</html>";

//config app
app.get("/", function(req, res) { res.send(index); });

//listen
app.listen(8080, function() {
```

```
    console.log("Listening...");
  });
```

6. Instalar dependencias:

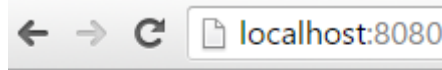
```
> npm install
```

7. Iniciar la aplicación **Express**:

```
> npm start
```

8. Abrir el navegador.

9. Ir a <http://localhost:8080>:



¡Hola Mundo!

CONFIGURACIÓN DE UN DIRECTORIO PÚBLICO

En este punto, ya tenemos la aplicación inicial. Ahora, vamos a usar **serve-static** para servir contenido estático.

1. Editar el archivo **package.json**.
2. Añadir **serve-static** como dependencia:

```
"dependencies": {
  "express": "*",
  "serve-static": "*"
}
```

3. Guardar cambios.
4. Crear el directorio público, **public**, en el directorio del proyecto.
5. Crear el archivo **one.txt** en el directorio **public**:

ONE!

6. Crear el archivo **two.html** en el directorio público:

```
<!doctype html>

<html>
  <head>
    <title>Contenido estático</title>
  </head>

  <body>
    <p>
      Esto es un ejemplo de contenido estático.
    </p>
  </body>
</html>
```

7. Editar el archivo **app.js**.
8. Importar el **middleware serve-static** y el módulo **path**:

```
//imports
const express = require("express");
const serveStatic = require("serve-static");
const path = require("path");
```

9. Configurar **serve-static** para que sirva el contenido estático del directorio **public**:

```
//config app
app.use(serveStatic(path.join(__dirname, "public"), {index: "index.html"}));
app.get("/", function(req, res) { res.send(index); });
```

Recordemos que la variable global **__dirname** contiene el directorio donde reside el *script* de **Node** en ejecución.

10. Guardar cambios.
11. Ir a la consola.

12. Detener la aplicación web.
13. Instalar nuevas dependencias configuradas en el archivo `package.json`:
`> npm update`
14. Arrancar de nuevo la aplicación web:
`> npm start`
15. Ir al navegador.
16. Ir a <http://localhost:8080>.

Debería mostrar el contenido *¡Hola Mundo!*

17. Ir a <http://localhost:8080/one.txt>.

Debería mostrar el contenido *ONE!*

18. Ir a <http://localhost:8080/two.html>.

Debería mostrar *Esto es un ejemplo de contenido estático.*

19. Ir a <http://localhost:8080/three.jpg>.

`Express` devolverá una respuesta 404 indicando que no encuentra el recurso. Es importante recordar que esta respuesta no la remite el `middleware serve-static`, sino el propio `Express`. Cuando `serve-static` no encuentra el recurso solicitado por el usuario, no remite una respuesta de error, sino que pasa al siguiente componente de `middleware` del flujo de procesamiento. Sólo si la encuentra, cancela el resto del flujo.

CONFIGURACIÓN DE VARIOS DIRECTORIOS PÚBLICOS

Ahora, vamos a configurar una segunda carpeta pública mediante otra función `serve-static`:

1. Crear la carpeta `public2` en el directorio del proyecto.
2. Crear el archivo `three.txt` en el directorio `public2`:

THREE!

3. Editar el archivo `app.js`.
4. Configurar la carpeta `public2` como pública:

```
app.use(serveStatic(path.join(__dirname, "public"), {index: "index.html"}));
app.use(serveStatic(path.join(__dirname, "public2"), {index: "index.html"}));
app.get("/", function(req, res) { res.send(index); });
```

5. Guardar cambios.
6. Ir a la consola.
7. Reiniciar la aplicación.
8. Ir a <http://localhost:8080>.
9. Ir a <http://localhost:8080/one.txt>.
10. Ir a <http://localhost:8080/two.html>.
11. Ir a <http://localhost:8080/three.txt>.

Cuando el primer `serve-static` no encuentra el recurso en el directorio `public`, pasa la solicitud al siguiente componente de `middleware` del flujo de procesamiento. En nuestro caso, al `serve-static` que se encarga de servir el contenido del directorio `public2`. El cual es finalmente el que sirve el recurso `three.txt`.

CONFIGURACIÓN DEL FAVICONO DE LA APLICACIÓN

En este punto, vamos a configurar el `middleware serve-favicon` para servir el favicono de la aplicación:

1. Editar el archivo `package.json`.
2. Añadir `serve-favicon` a las dependencias:

```
"dependencies": {
  "express": "*",
  "serve-favicon": "*",
  "serve-static": "*"
}
```

3. Guardar cambios.
4. Ir a la consola.
5. Detener la aplicación web.
6. Instalar las nuevas dependencias:


```
> npm update
```
7. Crear la carpeta **images** en el directorio público **public**.
8. Ir al navegador.
9. Guardar el favicono de **Express**, en la carpeta **public/images/**.

Para ello, editar el contenido de la página principal y buscar el elemento `<link rel="icon">`. A continuación, solicitarlo mediante el navegador. A menos que hayan cambiado las cosas, el favicono es expressjs.com/images/favicon.png.

10. Editar el archivo **app.js**.
11. Importar el módulo **serve-favicon**:


```
//imports
const express = require("express");
const serveFavicon = require("serve-favicon");
const serveStatic = require("serve-static");
const path = require("path");
```
12. Indicar el favicono en el texto de la página principal:

```
const index = "<!doctype html>\n" +
  "<html>\n" +
  "<head>\n" +
  "  <title>Express app</title>\n" +
  "  <link rel='icon' type='image/png' href='/images/favicon.png'>\n" +
  "</head>\n" +
  "<body><p>¡Hola Mundo!</p></body>\n" +
  "</html>";
```


13. Configurar el servidor del favicono:


```
app.use(serveFavicon(path.join(__dirname, "public/images/favicon.png"), {maxAge: 43200000}));
app.use(serveStatic(path.join(__dirname, "public"), {index: "index.html"}));
app.use(serveStatic(path.join(__dirname, "public2"), {index: "index.html"}));
```

 No olvidar añadir el servidor de favicono al comienzo del flujo de procesamiento.

14. Guardar cambios.
15. Iniciar la aplicación web:


```
> npm start
```
16. Ir al navegador.
17. Ir a <http://localhost:8080>. Debería aparecer el favicono en la pestaña de la página:

 Express app