

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

Al finalizarla, el estudiante:

- Habrá configurado el componente de *middleware* `cookie-parser`.
- Habrá creado y modificado *cookies*.
- Habrá accedido a los valores de las *cookies* remitidas en los mensajes de petición `HTTP`.
- Habrá creado *cookies* firmadas.

## objetivos

---

El objetivo de la práctica es crear una aplicación `Express` que permita mostrar cómo trabajar con *cookies* en una aplicación `Express`.

## 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",
    "start-dev": "./node_modules/.bin/nodemon ./app.js"
  },
  "dependencies": {
    "express": "*"
  },
  "devDependencies": {
    "nodemon": "*"
  }
}
```

5. Crear el archivo `index.html`:

```
<!doctype html>

<html>
  <head>
    <title>Express app</title>
  </head>

  <body>
    <p>¡Hola Mundo!</p>
  </body>
</html>
```

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

```

"use strict";

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

//app
const app = express();

//rutas
app.get("/", function(req, res) {
  res.sendFile("index.html", {root: __dirname});
});

//listen
http.createServer(app).listen(8080, function() {
  console.log("Listening...");
});

```

7. Instalar dependencias:

```
> npm install
```

8. Iniciar la aplicación **Express**:

```
> npm run start-dev
```

9. Abrir el navegador.

Para esta práctica, por favor, use **Chrome**. Aunque si sabe cómo consultar las cookies en el navegador, puede utilizar cualquier otro.

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

## CONFIGURACIÓN DE COOKIE-PARSER

Veamos cómo añadir el componente de *middleware* **cookie-parser** a la pila para que haga, por nosotros, el trabajo sucio relacionado con las *cookies*:

1. Editar el archivo **package.json**.
2. Añadir el paquete **cookie-parser** a las dependencias de la aplicación:

```

"dependencies": {
  "express": "*",
  "cookie-parser": "*"
}

```

3. Guardar cambios.
  4. Abrir otra consola.
  5. Ir al directorio de la práctica.
  6. Actualizar dependencias:
- ```
> npm update
```
7. Editar el archivo **app.js**.
  8. Añadir la función de *middleware* **cookie-parser** a la pila de la aplicación:

```

//middleware
app.use(require("cookie-parser")());

```

9. Añadir una función de *middleware* personalizada que añada, a toda respuesta, una *cookie* que indique la primera vez que se conectó el usuario y otra la última:

```

app.use(function(req, res, next) {
  if (!req.cookies["first-connection"]) {
    console.log("Creando cookie first-connection...");
    res.cookie("first-connection", new Date().toString());
  }

  console.log("Creando/modificando cookie last-connection...");
  res.cookie("last-connection", new Date().toString());
}

```

No hay que olvidar la invocación a la función `next()` para que la petición **HTTP** continúe en el flujo normal de procesamiento.

- ```
app.get("/cookie", function(req, res) {
  res.set("Content-Type", "text/plain");

  for (let cookie in req.cookies) {
    res.write(cookie + ": " + req.cookies[cookie].toString() + "\n");
  }

  res.end();
});
```

- Observe que la *cookie* *first-connection* no cambia, mientras que *last-connection* sí lo hace.

- Para ello, **Ctrl+Shift+J** e ir **Resources > Cookies > localhost:**

- Comprobar que puede leer sus valores correctamente. Salvando la interpretación de los espacios mediante la secuencia de caracteres `%20`, todo lo demás reflejará claramente las fechas.
- Suprimir las *cookies* seleccionando **Clear all from "localhost"** del menú contextual de las cookies:



1. Editar el archivo `app.js`.
2. Modificar el registro de `cookie-parser` en la pila de *middleware*:  
`app.use(require("cookie-parser")("mi clave de firma"));`
3. Modificar el *middleware* que crea las *cookies*:

```

app.use(function(req, res, next) {
  if (!req.signedCookies["first-connection"]) {
    console.log("Creando cookie first-connection...");
    res.cookie("first-connection", new Date().toString(), {signed: true});
  }

  console.log("Creando cookie last-connection...");
  res.cookie("last-connection", new Date().toString(), {signed: true});

  next();
});

```

No olvidar que el acceso a *cookies* firmadas se hace mediante la propiedad `signedCookies` en vez de `cookies`. Y la creación o modificación, se hace con el mismo método, `cookies()`, pero pasando como opción `signed` a `true`.

4. Modificar el controlador de ruta `/cookie` para que use `signedCookies`, en vez de `cookies`:

```

app.get("/cookie", function(req, res) {
  res.set("Content-Type", "text/plain");

  for (let cookie in req.signedCookies) {
    res.write(cookie + ": " + req.signedCookies[cookie].toString() + "\n");
  }

  res.end();
});

```

5. Guardar cambios.
6. Ir al navegador.
7. Solicitar <http://localhost:8080>.
8. Solicitar <http://localhost:8080/cookie>.
9. Volver a solicitar <http://localhost:8080/cookie>.
10. Consultar el contenido de las *cookies* almacenadas en el navegador y observar que hay caracteres adicionales a la fecha.