ESTADOS DE SESIÓN (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 usado los componente de *middleware* cookie-session y express-session para administrar los estados de sesión.
- Habrá accedido a los datos de sesión remitidos por los clientes en las peticiones HTTP.

OBJETIVOS

El objetivo de la práctica es crear una aplicación Express que permita mostrar cómo trabajar con el estado de sesión. Para ello, almacenaremos el número de peticiones HTTP, enviadas durante cada sesión, en el estado de sesión.

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

Uso de cookie-session

Primero, vamos a practicar con cookie-session. Que mantiene el estado de sesión en una *cookie* intercambiada entre el cliente y la aplicación a lo largo de la sesión.

Configuración de cookie-session

Veamos cómo añadir el componente de *middleware* cookie-session a la pila para que administre el estado de las sesiones:

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

```
"dependencies": {
  "express": "*",
  "cookie-session": "*"
}
```

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-session")({
  name: "session",
  keys: ["sign key", "verify key"]
}));
```

9. Añadir una función de *middleware* personalizada que fije el identificador de sesión y actualice el contador de páginas solicitadas:

```
app.use(function(req, res, next) {
  if (req.session.isNew) {
    console.log("Creando id de sesión...");
```

```
    req.session.id = new Date().toString();
    req.session.pages = 0;
  }

  req.session.pages += 1;
  next();
});
```

10. Añadir la ruta /session para mostrar el contenido del objeto sesión:

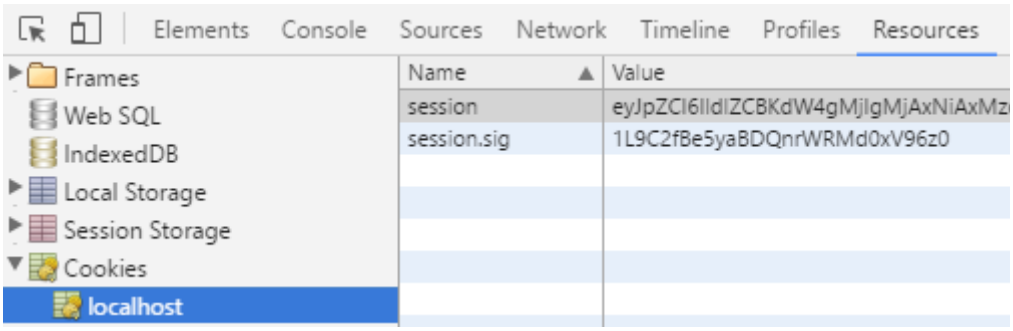
```
app.get("/session", function(req, res) {
  res.json(req.session);
});
```

11. Guardar cambios.
12. Ir al navegador.
13. Solicitar http://localhost:8080.
14. Solicitar http://localhost:8080/session.
15. Volver a solicitar http://localhost:8080/session.

    Observe que no cambia el identificador de sesión, pero sí el valor de la propiedad pages.

16. Consultar el valor de la *cookie* session en el navegador.

    Recordemos que en Chrome hay que hacer Ctrl+Shift+J e ir Resources > Cookies > localhost:



17. Abrir otro navegador.

    No utilice el mismo. Si ha usado Chrome, abra ahora Edge o Firefox.

18. Solicitar http://localhost:8080/session.

    Observe el valor de las propiedades. No coinciden con los del otro navegador. Normal, para la aplicación es una nueva sesión.

19. Cerrar el segundo navegador.

Uso de express-session

Ahora, vamos a utilizar express-session en vez de cookie-session. Recordemos cuál es la diferencia: cookie-session almacena *todo* el estado de sesión en la *cookie*. Mientras que express-session sólo su identificador, el resto en la memoria de la aplicación.

1. Ir al navegador.
2. Suprimir las *cookies* de localhost.
3. Editar el archivo package.json.
4. Reemplazar la dependencia cookie-session por express-session:

```
"dependencies": {
  "express": "*",
  "express-session": "*"
}
```

5. Guardar cambios.

6. Instalar la nueva dependencia:

```
> npm update
```

7. Editar el archivo app.js.

8. Reemplazar el registro del *middleware* cookie-session por express-session:

```
app.use(require("express-session")({
  name: "session",
  secret: "key",
  resave: false,
  saveUninitialized: false,
  genid: function(req) {
    console.log("Creando id de sesión...");
    return Date.now();
  }
}));
```

9. Modificar la función de *middleware* que usamos en cookie-session para asignar el identificador de sesión y actualizar el contador de páginas:

```
app.use(function(req, res, next) {
  if (!req.session.hasOwnProperty("pages")) req.session.pages = 0;
  req.session.pages += 1;
  next();
});
```

10. Modificar la función controladora de la ruta /session:

```
app.get("/session", function(req, res) {
  res.set("Content-Type", "text/plain");
  res.write("Id.: " + req.session.id + "\n");
  res.write("Pages: " + req.session.pages + "\n");
  res.end();
});
```

11. Ir al navegador.

12. Solicitar http://localhost:8080.

13. Solicitar http://localhost:8080/session.

14. Volver a solicitar http://localhost:8080/session.

