ESTADOS DE SESIÓN

Tiempo estimado: 10min

El estado de sesión es un aspecto muy importante de las aplicaciones webs. Permite almacenar información específica de cada una de las sesiones abiertas actualmente en la aplicación. En este estado, podemos almacenar datos como el identificador de la sesión, el usuario que se encuentra detrás de ella, el perfil del usuario y todo aquello que sea necesario.

Primero, introducimos los conceptos de sesión y estado de sesión. Y a continuación, presentamos dos componentes de *middleware* que facilitan el mantenimiento de los estados de sesión, cookie-session y express-session.

Al finalizar la lección, el estudiante sabrá:

- Qué es una sesión.
- Qué es el estado de sesión.
- Cómo administrar y almacenar los estados de sesión mediante los componentes cookie-session y express-session.

INTRODUCCIÓN

Una sesión (*session*) representa una conexión entre un cliente y la aplicación web. Consiste en todos los mensajes HTTP intercambiados entre los dos extremos. A toda sesión se le puede asociar lo que se conoce formalmente como estado de sesión (*session state*), un conjunto específico de datos relacionados con esa sesión como, por ejemplo, su identificador o el nombre del usuario que se encuentra detrás.

Por lo tanto, cuando tengamos la necesidad de almacenar datos específicos para cada sesión y así mantenerlos accesibles a lo largo de todos los mensajes HTTP intercambiados entre el cliente y la aplicación, el lugar adecuado en el que hacerlo es el estado de sesión.

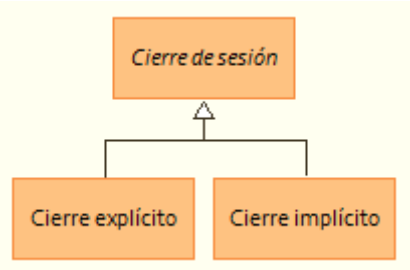
Los estados de sesión se pueden almacenar en *cookies*, en bases de datos e incluso en la memoria de la aplicación. En esta lección, vamos a presentar dos componentes de *middleware* para el almacenamiento de las sesiones: cookie-session, que almacena los datos en *cookies*; y express-session que lo hace en la memoria de la aplicación. Cada uno tiene sus pro y sus contra y hay que tenerlos en cuenta a la hora de usarlos.

IDENTIFICADOR DE SESIÓN

Para ayudar a administrar las distintas sesiones abiertas simultáneamente en una aplicación, se suele asociar un identificador de sesión (*session identifier*), generalmente, una cadena de texto que permite distinguir la sesión de manera única con respecto a las demás. Este identificador se suele generar durante la fase de apertura de sesión (*session open*), operación que detecta que la petición HTTP corresponde a una nueva sesión, le asigna su identificador y prepara su estado para futuros accesos.

CIERRE DE SESIÓN

El cierre de sesión (*session close*) es la operación mediante la cual se da por terminada una sesión. Atendiendo a quién dispara su ejecución, se distingue entre cierres explícitos e implícitos.



Mediante un cierre explícito (*explicit close*), es el usuario el que cierra la sesión manualmente. Este cierre es el que usan muchos portales proporcionando un botón de cierre (*sign out*). En cambio, el cierre implícito (*implicit close*) es aquel que realiza automáticamente la aplicación, generalmente, tras pasar un período de tiempo sin recibir peticiones HTTP del cliente.

Una aplicación puede soportar ambos tipos de cierre.

Middleware cookie-session

cookie-session es un componente de *middleware* que almacena los estados de sesión en *cookies*. Añade automáticamente a las respuestas HTTP el estado de sesión, mediante un campo de cabecera Set-Cookie. De esta manera, la próxima vez que el cliente remita una petición HTTP adjuntará la información de sesión en un campo de cabecera Cookie.

Este *middleware* no se recomienda si el estado de sesión contiene datos de carácter personal o privados, porque las *cookies* se transmiten con todos los mensajes HTTP intercambiados por el cliente y la aplicación. Es ideal para estados de sesión que contienen el identificador de sesión, el tema de diseño usado por el usuario, etc., pero nunca datos privados y/o confidenciales.

Este componente lo que hace es analizar las *cookies*, extraer aquella que está relacionada con el estado de sesión y ponerlo a disposición de la aplicación mediante la propiedad session del objeto petición. De esta manera, es muy fácil para la aplicación trabajar con el estado de sesión. Cuando la aplicación finalice de procesar la petición HTTP, el mismo componente coge el contenido de esta propiedad y se lo remite al cliente mediante otro campo de cabecera Set-Cookie. Es importante tener claro que *siempre* debemos trabajar con la propiedad req.session. Incluso cuando deseamos modificar el estado de sesión.

Función cookie-session

La función cookie-session devuelve la función de *middleware* a registrar en la pila de procesamiento para que pueda hacer su trabajo según una configuración dada:

```
function cookie-session(options) : function
```

Parámetro	Tipo de datos	Descripción
options	object	Opciones: • name (string). Nombre de la *cookie* en la que almacenar el estado de sesión. • keys (array). Claves usadas por el *middleware* para firmar y verificar el valor de la *cookie*. La primera clave se usa para firmar y la segunda para verificar. • secret (string). Clave de firma, si no se especifica keys. • Opciones de *cookie*: maxAge, expires, path, domain, etc.

Registro de la función de middleware

La función devuelta por cookie-session hay que registrarla en la pila de *middleware* antes que los controladores de petición que usan la propiedad session. Generalmente, al comienzo de la pila de middleware.

Ejemplo:

```
app.use(require("cookie-session")({
  name: "session",
  keys: ["sign key", "verify key"]
}));
```


Propiedad request.session

cookie-session crea automáticamente la propiedad session en el objeto de la petición HTTP en curso, para así facilitar el acceso a la información del estado de sesión. Esta propiedad hay que usarla tanto para leer el estado de sesión como para escribirlo o modificarlo. Es importante recordar que la modificación del estado se hace a través del objeto petición, no de la respuesta como algunos podrían esperar.

El objeto session contiene varias propiedades especiales que proporcionan información sobre la sesión:

- isNew (boolean). Indica si estamos ante una nueva sesión.
- isChanged (boolean). Indica si el objeto sesión ha cambiado durante el procesamiento de la petición.
- IsPopulated (boolean). Indica si el objeto sesión tiene datos.

Apertura de sesión

La apertura de sesión es la operación mediante la cual inicializamos el estado de sesión cuando detectamos que estamos ante una nueva sesión. Tal como acabamos de ver, para saber si estamos ante una nueva sesión, no hay más que consultar la propiedad isNew de req.session.

Generalmente, se utiliza una función de *middleware* específica para llevar a cabo la operación de apertura que como mínimo suele consistir en asignarle su identificador para así poder diferenciarla de las demás. Veamos un ejemplo:

```
app.use(function(req, res, next) {
  if (req.session.isNew) req.session.id = Date.now();
  next();
});
```

Observe que el identificador de sesión no lo fija cookie-session, sino una operación definida por la aplicación.

Cierre de sesión

Para realizar un cierre de sesión explícito, no hay más que asignar el valor null a session:

```
req.session = null;
```

Propiedad request.sessionOptions

El componente también crea la propiedad sessionOptions en el objeto petición. En esta propiedad, se indica las opciones de configuración de la sesión y de la *cookie* de sesión. Como, por ejemplo, maxAge, expires, etc. Se puede utilizar para conocer sus valores e incluso para cambiarlos de cara al resto de la sesión.

Así por ejemplo, para fijar el período de expiración de la *cookie* de una sesión, basta con hacer algo como:

```
req.sessionOptions.maxAge = 60000;
```

Middleware express-session

Una de las principales desventajas de cookie-parser es que todo el estado de sesión se almacena en la *cookie*, la cual transmiten una y otra vez los extremos durante la sesión. No se recomienda almacenar datos privados ni de carácter personal, aun si aseguramos que el estado de sesión sólo se transmita mediante una conexión segura como HTTPS. En estos casos, el estado de sesión se debe almacenar en la memoria de la aplicación o bien en una base de datos.

Mediante express-session, el estado de sesión se almacena en memoria, aunque el identificador de la sesión se almacena en una *cookie*. La diferencia con respecto a cookie-session es que express-session sólo transmite el identificador de sesión en la *cookie*; mientras que cookie-session, todo el estado de sesión. Por lo que si el servidor cae o se reinicia, con express-session no tendremos acceso a la información de sesión, al perderse entre los reinicios. En cambio, con cookie-session si se reinicia la aplicación, el estado de sesión no se pierde porque se almacena en el cliente.

Función express-session

Al igual que cookie-session, el paquete express-session no es más que una función la cual devuelve, dada una configuración, la función de *middleware* a registrar en el flujo de procesamiento de la aplicación:

```
function express-session(options) : function
```

Parámetro	Tipo de datos	Descripción
options	object	Opciones: • name (string). Nombre de la *cookie* en la que almacenar el identificador de sesión. • cookie (object). Configuración de la *cookie*: domain, expires, maxAge, secure, etc. • genid (function). Función que debe invocar el *middleware* cada vez que necesite generar un nuevo identificador de sesión: fn(req) : string. El valor devuelto debe ser una cadena de texto. • secret (string). Clave con la que firmar la *cookie*. • rolling (boolean). ¿Adjuntar la *cookie* de sesión en toda respuesta sin esperar a que expire su período de vida? Valor predeterminado: false. • resave (boolean). ¿Guardar el estado de sesión en el almacén de sesiones? Generalmente, se asigna false. Valor predeterminado: true. • saveUninitialized (boolean). ¿Guardar el estado de sesión en el almacén de sesiones aun cuando no esté inicializado? Generalmente, se asigna false. Valor predeterminado: true.

Registro de la función de middleware

He aquí un ejemplo de registro de la función de *middleware* en la pila de procesamiento de la aplicación.

```
app.use(require("express-session")({
  name: "sinfo",
  cookie: {maxAge: "60000"},
  secret: "1468654010998",
  resave: false,
  saveUninitialized: false,
  genid: function(req) { return Date.now().toString(); }
}));
```

Propiedad request.session

Al igual que cookie-session, express-session crea una propiedad session para poner a disposición de la aplicación el estado de la sesión que está detrás de la petición HTTP.

Apertura de sesión

Aquí hay una diferencia con respecto a cookie-session. express-session asigna automáticamente el identificador de sesión, aunque permite que pueda generarlo la aplicación mediante una función indicada en la opción genid. Veamos un ejemplo:

```
app.use(require("express-session")({
  name: "session",
  secret: "key",
  genid: function(req) { return Date.now().toString(); }
}));
```

El identificador de sesión se encuentra disponible mediante la propiedad id del objeto req.session.

Cierre de sesión

El cierre de sesión también difiere de cookie-session. Con express-session, hay que invocar el método destroy() de req.session:

```
req.session.destroy(function(err) {
  //...
});
```

Propiedad request.session.cookie

En cookie-session, las opciones de la *cookie* se pueden acceder, en modo lectura/escritura, mediante la propiedad sessionOptions del objeto req. En cambio, con express-session, se dispone de la propiedad cookie de req.session. Ejemplo:

```
//en cookie-session
```

```
req.sessionOptions.maxAge = 60000;

//en express-session
req.session.cookie.maxAge = 60000;
```

Almacenes de sesiones

Un almacén de sesiones (*session store*) es un dispositivo en el que almacenar persistentemente los estados de sesión. De manera predeterminada, se usa la memoria de la aplicación. Pero si se desea, se puede usar bases de datos como Cassandra, MariaDB, MongoDB, PostgreSQL o Redis. express-session permite escribir los estados de sesión en bases de datos. Puede consultar una lista de almacenes compatibles con este componente en https://github.com/expressjs/session#compatible-session-stores.

