Instituto Politécnico Nacional

Escuela Superior de Cómputo

Web App Development.

Tarea 6 : Comentar Linea a Linea ClaseSimple.java

Profesor: M. en C. José Asunción Enríquez Zárate

Alumno: Aarón Antonio García González

aarongarcia.ipn.escom@gmail.com

3CM9

December 19, 2020

Contents

Lista de figuras

Lista de tablas

1 Introducción

Independientemente de la técnica utilizada, lo que está claro es que hay almacenar los datos de la sesión en algún sitio. Una buena opción es el objeto HttpSession que puede almacenar los datos de la sesión en el servidor. Para utilizar este objeto es necesario tener un objeto de la sesión, leerlo, escribirlo y eliminarlo cuando se cierre la sesión o después de un cierto tiempo si la sesión no se ha cerrado expresamente.

La persistencia del objeto es válida en el contexto de la aplicación Web, por lo que puede ser compartida por varios servlets. Un servlet puede acceder a objetos almacenados en otro servlet. Estos objetos, también llamados atributos, pueden estar disponibles para otros servlets dentro del ámbito de petición, sesión o aplicación.

El objeto HttpSession almacena y accede a la información de la sesión que controla el servlet, por lo que no es necesario que esta información sea manipulada en código. Los métodos más útiles de esta clase son:

- getId(). devuelve una cadena conteniendo el identificador único asignado a la sesión. Es el utilizado en la reescritura de URL para identificar a esa sesión.
- isNew(). devuelve true si el cliente no ha establecido una sesión. Si el cliente tiene deshabilitadas las cookies, la sesión será nueva en cada petición.
- setAttribute(). asigna un objeto a la sesión, utilizando un nombre determinado.
- getAttribute(). devuelve el objeto asignado a la sesión, identificado por el nombre que le indique.
- setInactiveInterval(). especifica el tiempo máximo que ha de pasar entre peticiones consecutivas del cliente para que la sesión se dé como no válida. Un número negativo como argumento hará que nunca se invalide la sesión.
- invalidate(). elimina la sesión actual y libera todos los objetos asociados a ella.

2 Desarrollo

```
    // Declaración de la clase pública llamada ClaseSimple
public class ClaseSimple {
    // Declaración de un método público de ClaseSimple con el nombre
    // metodoUno. Que recibe como parámetros un request de tipo HttpServletRequest
    // y otro llamado response de tipo HttpServletResponse.
    public void metodoUno(HttpServletRequest request, HttpsServletResponse response) {
        /***
         * Declaración de la variable s de tipo HttpSession.
         * Se le asigna el valor que regresa el método request.getSession(boolean create);
         * Si el valor dentro del método getSession es igual a true, significa que en caso
         * de que no exista una sesión en el request la va a crear. En este caso, se creará
         * una nueva sesión si no llegara a existir.
         */
        HttpSession s = request.getSession(true);
        /***
         * A la variable s de tipo HttpSession, se asigna el atributo "nombre" con el valor
         * "valor". Por lo tanto, nuestro objeto de sesión ahora tiene un atributo similar
         * a esto
         *  {
         *      nombre: "valor"
         *  }
         * Indicando que la sesión no estará vacía.
         */
        s.setAttribute("nombre", "valor");
    // Cierre de método
    }

    // Declaración de un método público de ClaseSimple con el nombre
    // metodoDos. Que recibe como parámetros un request de tipo HttpServletRequest
    // y otro llamado response de tipo HttpServletResponse.
    public void metodoDos(HttpServletRequest request, HttpsServletResponse response) {
        /***
         * Declaración de la variable s de tipo HttpSession.
         * Se le asigna el valor que regresa el método request.getSession(boolean create);
         * Si el valor dentro del método getSession es igual a true, significa que en caso
         * de que no exista una sesión en el request la va a crear. En este caso, se creará
         * una nueva sesión si no llegara a existir.
         */
        HttpSession s = request.getSession(true);
        /**
         * De la variable s de tipo HttpSession se remueve el atributo con la llave "nombre"
         * del mismo objeto.
         */
        s.removeAttribute("nombre");
        /**
         * Se valida si s es distinta de nulo
         */
        if( s != null) {
            /**
             * Si la sesión continúa teniendo algún elemento o no es igual a null
             * se invalida la sesión con el método invalidate(). El cual no tiene ningún
             * valor de retorno. Pero deshabilita la sesión dentro del request.
             */
            s.invalidate();
        // Cierre de IF
        }
    // Cierre de método
    }
```

```
    // Declaración de un método público de ClaseSimple con el nombre
    // metodoTres. Que recibe como parámetros un request de tipo HttpServletRequest
    // y otro llamado response de tipo HttpServletResponse.
    public boolean metodoTres(HttpServletRequest request, HttpsServletResponse response) {
        /***
         * Declaración de la variable s de tipo HttpSession.
         * Se le asigna el valor que regresa el método request.getSession(boolean create);
         * Si el valor dentro del método getSession es igual a true, significa que en caso
         * de que no exista una sesión en el request la va a crear.
         * En este caso, no se creará una sesión en caso de no existir. Solamente solicitaremos
         * el atributo dentro del request
         */
        HttpSession s = request.getSession(false);
        /**
         * Validamos que s sea igual a null
         */
        if ( s == null) {
            /**
             * Si la sesión (s) es igual a null, retornamos un valor false.
             * Refiriéndonos a que no existe sesión en este request, por lo que debemos
             * de redirigirlo al JSP asignado por el Servlet que implementó la
             * validación de sesión
             */
            return false;
        /**
         * Cierre de IF
         * Si s es diferente de null
         */
        } else {
            /**
             * Se consulta el atributo "nombre" dentro del objeto de sesión (s).
             * En caso de que sea distinto de nulo se regresará el valor true,
             * de lo contrario será false.
             * Este sería el segundo filtro de nuestra validación de sesión, en caso
             * de que exista una sesión, necesitamos saber que el atributo no es null
             * para poder permitir el acceso a los métodos solicitados.
             */
            return s.getAttribute("nombre") != null;
        // Cierre de else
        }
    // Cierre de método
    }

// Cierre de clase
}
```

3 Conclusión

Una vez revisada línea a línea el código de la clase indicada, el manejo de sesión en Java no es muy diferente a cómo lo he trabajado con otros lenguajes. Para ello, la clase actuaría como un Middleware que revisaría siempre todos los request donde este este implementado, para verificar que exista una sesión y podamos validar el tipo de usuario, verificar su correo electrónico si es que necesitamos enviarle una notificación de algún movimiento que realizó o simplemente para validar que la sesión no haya expirado. Sin embargo, de esta clase creo que el más importante es el método dos y tres. ¿Por qué? Porque el método dos es el encargado de invalidar la sesión, es decir que tal vez nuestro token expiró o que cerramos sesión, por lo que tenemos que remover todos los atributos asignados y proceder con la invalidación de la sesión. Ahí entra el método tres, que es el que se encarga de verificar la sesión, suponiendo que no hayamos eliminado el atributo en el paso dos, se regresa una valor positivo y nuestro middleware debería de volver a regresar al paso 2 hasta que nuestra sesión ya no pueda ser accesada. Es por eso que en mi pensar, los métodos más importantes son el 2 y 3 para esta clase.

Aarón Antonio García González

4 Referencias Bibliográficas

Referencias

[La clase HTTP Session. (s. f.). Java para javatos. Recuperado 15 de diciembre de 2020, de https://javaparajavatos.wordpre
Java para javatos