TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE OAXACA

UNIDAD 2

LIBRERÍA

TÓPICOS AVANZADOS DE PORGRAMACIÓN

09:00 – 10:00

4SC

CHAVEZ LOPEZ JESHUA

FELIX ELIEL OLIVERA JIMÉNEZ

INTRODUCCIÓN

La **librería de cierre de sesión** es una solución diseñada para gestionar de manera eficiente el proceso de desconexión de usuarios en aplicaciones Java, especialmente aquellas que requieren un manejo adecuado de la seguridad y la integridad de la sesión. Esta librería está construida con el objetivo de facilitar el proceso de cierre de sesión de una forma estandarizada, permitiendo a los desarrolladores implementarlo de manera sencilla en sus proyectos.

DESCRIPCIÓN Y PROPÓSITO

Métodos estáticos para cierre de sesión: Esta librería proporciona métodos estáticos que permiten el cierre de sesión de manera rápida y sin la necesidad de instanciar objetos adicionales.

Integración sencilla: Puede estar integrado en proyectos Java de manera fácil, ya sea en aplicaciones de escritorio (Swing) o aplicaciones web (servlets, JSP). Su estructura modular facilita su inclusión en otros sistemas de autenticación.

Generación del archivo.jar: La librería está empaquetada como un archivo .jar, lo que permite su reutilización en diferentes proyectos sin necesidad de recompilar el código.

El propósito de la librería de cierre de sesión es proporcionar una solución simple**,** segura y eficiente para gestionar el proceso de cierre de sesión en aplicaciones Java, garantizando que la desconexión de un usuario se realice correctamente. Es una herramienta útil para aquellos que buscan implementar una funcionalidad de cierre de sesión efectiva en sus aplicaciones Java.

EXPLICACIÓN DEL CÓDIGO

1.- MonitorSesion

Propósito: Supervisa una ventana (JFrame) y la cierra automáticamente si la sesión del usuario ha expirado.

Atributos:

- JFrame ventana: la ventana principal que se está supervisando.
- String usuario: nombre del usuario actual.
- Timer timer: Temporizador de Swing que verifica periódicamente si la sesión sigue activa.

Métodos:

- MonitorSesion(JFrame ventana, String usuario, int intervaloChequeoMs): Constructor que inicializa el monitoreo de sesión con un intervalo específico en milisegundos.
    - **Validación implícita:** Solo se ejecuta el cierre si SessionManager.sesionActiva(usuario)devuelve false.
- void iniciar(): Inicia el temporizador para comprobar la sesión cada cierto intervalo.
- void detener(): Detiene el temporizador.

2.- SesionActiva

Propósito: Representa una sesión activa de un usuario. Guarde cuándo inició y cuándo fue su última actividad.

Atributos:

- String usuario: nombre del usuario de la sesión.
- long inicioSesion: momento (en milisegundos) en que se inició la sesión.

- long ultimaActividad: momento de la última acción del usuario.

Métodos:

- SesionActiva(String usuario):
    - Inicializa la sesión y registra el tiempo actual.
    - **Validación:** Lanza IllegalArgumentExceptionsi el usuario es null o está vacío.
- getUsuario(), getInicioSesion(), getUltimaActividad(): Captadores.
- setUltimaActividad (long tiempo): Permite actualizar la última actividad.
- static getUsuarioActivo(): (Método pendiente de implementar correctamente).

3.- SessionManager

Propósito: Clase final que **gestiona todas las sesiones activas** , controla su tiempo de inactividad y las cierra automáticamente si es necesario.

Atributos estáticos:

- Map<String, SesionActiva> sesiones: Mapa donde se almacenan todas las sesiones activas por usuario.
- long TIEMPO_INACTIVIDAD: Tiempo máximo permitido sin actividad (10 segundos en el código).
- Timer timer: Temporizador que revisa periódicamente la actividad.

Métodos:

- iniciarSesion(String usuario):
    - Crea una nueva sesión activa para un usuario.
    - **Validación:** El usuario no puede estar null ni vacío.
    - Llama a programarCierrePorInactividad()para verificar su tiempo de inactividad.

- registrarActividad(String usuario):
    - Actualiza el tiempo de la última actividad del usuario.
    - **Verifica si el usuario existe antes de continuar.**
- sesionActiva(String usuario):
    - Retorna true si la sesión del usuario sigue activa.
- cerrarSesion(String usuario):
    - Elimina la sesión del mapa y muestra mensaje en consola.
- programarCierrePorInactividad(String usuario):
    - Crea una tarea repetitiva (cada segundo) que verifica si el usuario lleva más del tiempo límite inactivo.
    - Si es así, cierra la sesión automáticamente.

INSTRUCCIONES PARA IPORTAR EL .JAR EN OTRO PROYECTO

1. Crear el archivo .jar de la librería (si aún no lo has hecho):

Paso 1: Abre el proyecto de la librería en NetBeans.

Paso 2: Haz clic derecho sobre el proyecto en el panel izquierdo (Projects).

Paso 3: Selecciona **“Limpiar y construir”**.

Paso 4: NetBeans generará el .jar en la carpeta, por ejemplo:

NombreProyecto/dist/NombreProyecto.jar

2. Importar el .jar en otro proyecto:

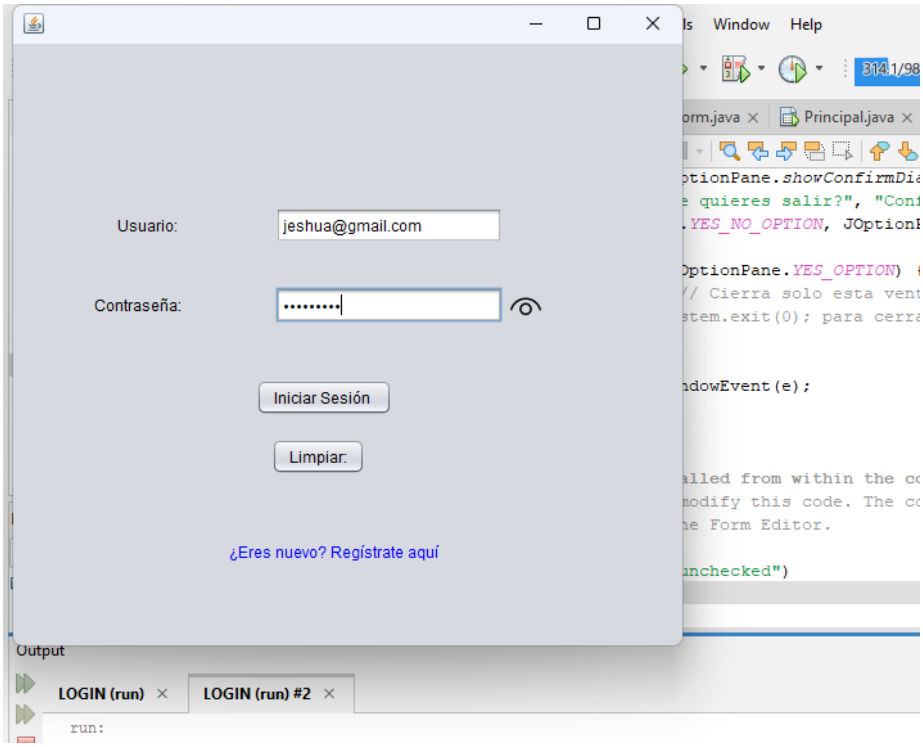
Paso 1: Abre el proyecto en el que quieres usar la librería.
Paso 2: Haz clic derecho sobre el nombre del proyecto (en el panel "Projects").
Paso 3: Selecciona "Propiedades" (Propiedades).
Paso 4: En el menú de la izquierda, selecciona "Librerías" (Librerías).
Paso 5: Haz clic en el botón "Add JAR/Folder..." (Agregar JAR/Carpeta...).
Paso 6: Busca la ubicación del .jar que generaste (usualmente está en la carpeta dist del proyecto original).
Paso 7: Selecciona el archivo .jar y haz clic en "Abrir".
Paso 8: Haz clic en "OK" para cerrar la ventana de propiedades.

3. Usar las clases de la librería:
Ahora ya se puede importar y utilizar las clases de nuestra librería en cualquier archivo del nuevo proyecto. Por ejemplo:

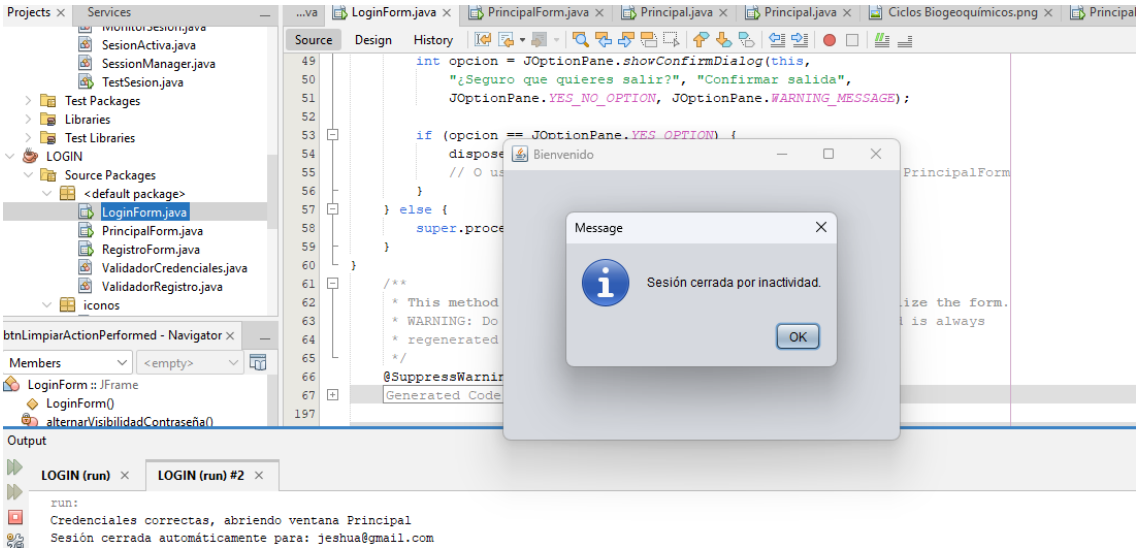
```
import mi.libreria.SesionActiva;

import mi.libreria.SessionManager;
```

CAPTURAS

Se inicia sesión y empieza a analizar cada segundo si esta activo o si se hace click en alguna parte de la pantalla (entiéndase por pantalla el JFrame al que nos redirige una vez iniciada sesión).



Una vez pasado el tiempo máximo de inactividad permitida, cierra la sesión.



LINK YOUTUBE

https://youtu.be/RjkmQMBGqsM