Ejercicios de acceso remoto a componentes y transaccionalidad

Índice

1. Acceso remoto a componentes

Hacer accesible el GestorOfertas de la sesión anterior como componente remoto usando todas las opciones vistas en el tema: RMI, hessian y HTTP invoker (burlap no es necesario ya que es prácticamente idéntico a Hessian). Comprobar que funciona usando un cliente simple en línea de comandos que aceptará como parámetro un 0 si hay que acceder por RMI, 1 por hessian, o 2 por HTTP invoker.

2. Eficiencia en el acceso remoto (*)

Añadir una opción al cliente anterior que permita, pasando el parámetro con valor 3, probar una tras otra todas las opciones. Para comparar el tiempo que tarda cada una se puede usar la clase `org.springframework.util.StopWatch`. Simplemente usando dicha clase hay que llamar a `start(nombre_de_tarea)` justo antes de llamar al método que queremos cronometrar y a `stop()` justo después. El nombre_de_tarea es arbitrario. Al llamar a `prettyPrint` se obtendrá un String con un informe de los tiempos de ejecución de todos los métodos cronometrados hasta el momento. ¿Cuáles son las conclusiones?. incluirlas en un fichero llamado "tiempos.txt" que debéis poner en el directorio raíz del proyecto de Eclipse de esta sesión.

3. Transaccionalidad declarativa en Spring

Hacer transaccional el método `enviarMensaje` de la fachada. Debéis modificar la versión actual de Fachada.enviarMensaje y GestorUsuariosImpl.cobrar por las siguientes, ya que la versión anterior tenía un bug que impedía anular el cobro si fallaba la transacción.

Cambiar el método "enviarMensaje" de la clase Fachada por este:

```
	public void enviarMensaje(Usuario remitente, Mensaje mensaje)
throws SinPermisoException, UsuarioNoExisteException {
		//cobrar por el envio del mensaje
		gu.cobrar(remitente, costeMensaje);
		//realizar el envio
		gm.enviar(mensaje);

	}
```

Cambiar el método "cobrar" de la clase GestorUsuariosImpl por este:

```
	public void cobrar(Usuario u, int cantidad) throws
SinPermisoException {
```

```
                int credActual = u.getCredito()-cantidad;
                if (credActual<0)
                        throw new SinPermisoException("No tienes crédito
suficiente");
                else {
                        u.setCredito(credActual);
                        usuarioDAO.actualizar(u);
                }
        }
```