Ejercicios de AOP en Spring

Índice

Continuamos en esta sesión con la aplicación AmigosSpring, añadiéndole ahora aspectos para poder realizar diversas tareas

1. Seguridad básica con AOP

Nota:
El objetivo del ejercicio no es tanto implementar un sistema completo de seguridad como hacer un ejercicio interesante de AOP que nos permita vislumbrar sus posibilidades. Para un sistema potente y flexible de seguridad AOP, veremos Spring Security en la última sesión

Queremos incorporar control de acceso basado en AOP a AmigosSpring. Esto nos permitirá despreocuparnos de chequear de forma programativa los permisos, evitando descuidos y sin necesidad de tocar para nada el código original.

Antes que nada, hay que incorporar las librerías de AOP al proyecto:

- Librerías de AspectJ: *runtime* y *weaver*
- Enlace de AspectJ con Spring

Aviso:
Para poder comprobar la sintaxis AOP y visualizar gráficamente los puntos de corte y los advices, se puede activar el soporte AJDT en Eclipse. Pulsar con el botón derecho sobre el proyecto y seleccionar "Convert to AspectJ Project". Este soporte es de gran ayuda a la hora de editar y compilar. El problema es que dicha opción también activa el "weaving" de las clases en tiempo de compilación de AspectJ, lo que **causará problemas al ejecutar la aplicación**. La opción más sencilla (aunque no muy elegante) es desactivar el soporte AspectJ para el proyecto cuando queramos ejecutar la aplicación.

1. Crear la clase `es.ua.jtech.amigosSpring.aspects.SeguridadAOP` y convertirla en un "Aspect"
2. Debemos interceptar el código existente en dos puntos. Se recomienda definir un pointcut con nombre para cada uno y combinarlos si es necesario.
   - **Cuando el usuario haga login**, para poder guardar sus datos en algún sitio. Los guardaremos en un bean con ámbito de sesión dentro del propio `Aspect`. Tendrás que definir por tanto una clase `es.ua.jtech.amigosSpring.aspects.UsuarioActual` que sea un bean de Spring con ámbito de sesión, e inyectarlo en SeguridadAOP. Para UsuarioActual únicamente es necesaria una propiedad `String login`, accesible a través de get/set.
   - **Cuando alguien intente hacer una operación**, para comprobar si se ha logueado (si el getLogin() del UsuarioActual devuelve algo o devuelve null). En caso de no estar logueado, se deberá abortar la operación. Para no interferir demasiado con el código de los servlets, en los que ya hay seguridad declarativa, **proteged los métodos de la capa de negocio**. Eso sí, **el propio método login no debe estar**

protegido.