



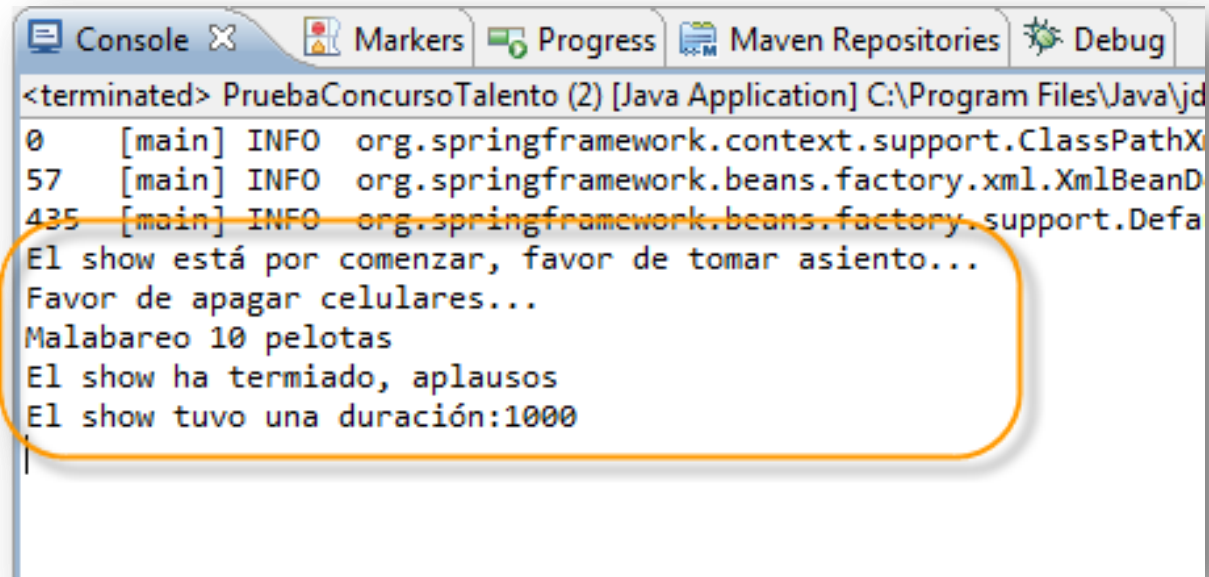
Ejercicio

Proyecto Concurso de Talentos v6

Objetivo del Ejercicio

- El objetivo del ejercicio es agregar funcionalidad AOP con el uso de around Advice a nuestro proyecto de Concurso de Talentos.

Al finalizar deberemos ver la siguiente salida, resultado de aplicar los conceptos de AOP descritos:



```
<terminated> PruebaConcursoTalentos (2) [Java Application] C:\Program Files\Java\jdk
0    [main] INFO    org.springframework.context.support.ClassPathX
57   [main] INFO    org.springframework.beans.factory.xml.XmlBeanD
435  [main] INFO    org.springframework.beans.factory.support.Defa
El show está por comenzar, favor de tomar asiento...
Favor de apagar celulares...
Malabareo 10 pelotas
El show ha terminado, aplausos
El show tuvo una duración:1000
```



Paso 1. Modificar el Aspecto Audiencia

Debemos modificar la definición del Aspecto en el archivo applicationContext.xml, ya que ahora en un solo método tendremos toda la funcionalidad de AOP.

```
<!-- configuracion de AOP con Aspect Around -->
<aop:config>
    <aop:aspect ref="audiencia">

        <aop:pointcut expression="execution(* concursantes.Concursante.ejecutar(..))" id="show" />

        <aop:around pointcut-ref="show" method="monitorearShow" />

    </aop:aspect>
</aop:config>
```

Paso 2. Modificar la clase Audiencia

Modificar la clase Audiencia.java por el siguiente código:

```
package concursantes;

import org.aspectj.lang.ProceedingJoinPoint;
import org.springframework.stereotype.Component;

@Component
public class Audiencia {

    public void monitorearShow(ProceedingJoinPoint joinpoint) {
        try {
            System.out.println("El show está por comenzar, favor de tomar asiento...");
            System.out.println("Favor de apagar celulares...");

            //Anotamos la hora de inicio
            long horaInicio = System.currentTimeMillis();

            //Se llama al método de negocio (método objetivo)
            joinpoint.proceed();

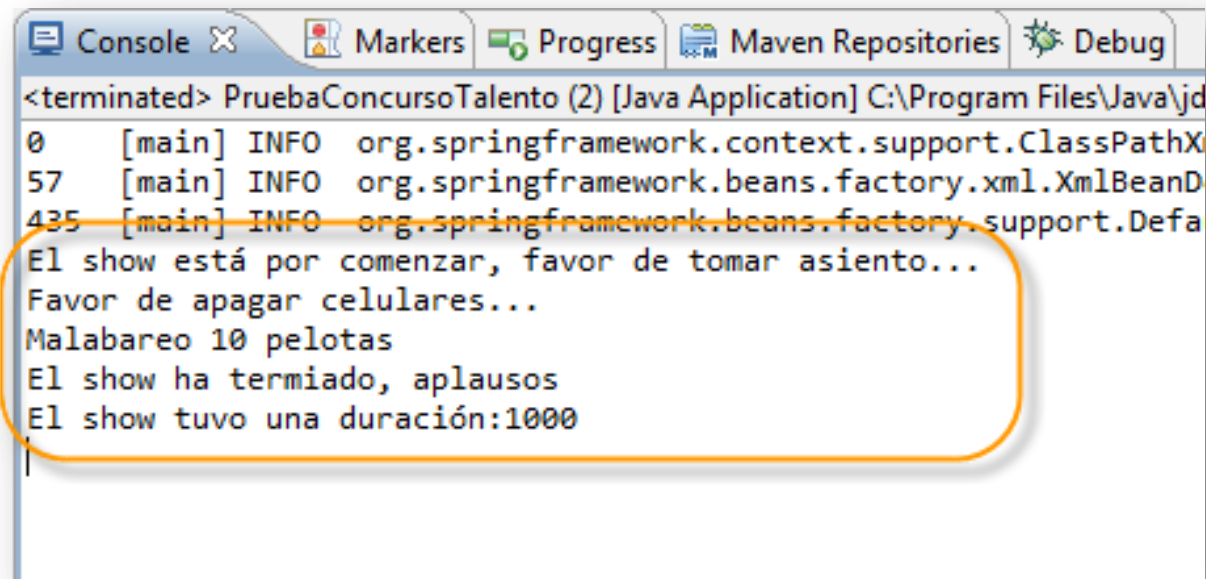
            Thread.sleep(1000); //1 segundo
            //Este delay en milisegundos es opcional y se puede poner en los métodos
            //de negocio para simular la duración del método

            long horaTermino = System.currentTimeMillis();

            System.out.println("El show ha terminado, aplausos");
            System.out.println("El show tuvo una duración:" + (horaTermino - horaInicio));
        } catch (Throwable t) {
            System.out.println("El show fue terrible, se devolverán las entradas");
        }
    }
}
```

Paso 3: Ejecutar la Prueba

- Ejecutamos la clase de prueba `PruebaConcursoTalentos.java`. Como podemos observar, podemos obtener todos el comportamiento del ejercicio anterior, pero más simplificado.



```
<terminated> PruebaConcursoTalentos (2) [Java Application] C:\Program Files\Java\jdk
0    [main] INFO    org.springframework.context.support.ClassPathX
57   [main] INFO    org.springframework.beans.factory.xml.XmlBeanD
435  [main] INFO    org.springframework.beans.factory.support.Defa
El show está por comenzar, favor de tomar asiento...
Favor de apagar celulares...
Malabareo 10 pelotas
El show ha terminado, aplausos
El show tuvo una duración:1000
```