Sintaxis del DSL

Indice

Diseño del sistema

Se proveerá un mecanismo en el cual el usuario pueda definir de una forma simple y legible que aspectos actuarán en el sistema y qué funcionalidad se ejecutará y en qué momento de la intercepción se desea realizar.

Se utilizará la arquitectura desarrollada en la parte 2 del TP, el DSL expondrá una forma de utilización sencilla para el usuario y se implementará la lógica correspondiente para poder configurar y agregar aspectos con lo ya existente, siendo esto último transparente para el usuario final.

Ejemplo de la definición de un aspecto mediante nuestro DSL:

```
module Aspects
    aspect do
        before do
            puts Time.now
        end
        cuando do
            expresion_regular metodo es /.*withLogTime/
        end
    end
end
```

Definición de los aspectos

El usuario define los varios aspectos que desea aplicar de la siguiente forma:

```
module Aspects

    aspect do

    end

    aspect do

    end
end
```

Bloque “aspect”

Dentro de este bloque se define la operación a realizar, el momento de realizarla y bajo que condiciones (“cuando”):

Operaciones (el usuario define que desea ejecutar):

```
instead_of | after | before | on_exception do |context|
    ejecucion del usuario
end
cuando do
end
```

Cuando (cuando se cumplen qué condiciones): Este bloque se describe a continuación

Bloque “cuando”

Acepta varios tipos de join points:

1. **Clase especifica:**
   clase es MiClase
2. **Nombre de una clase con expresión regular:**
   expresion_regular clase es /ClaseBuscada/
3. **Clases que provienen de una jerarquía**
   jerarquia clase pertenece EstaClase
4. **Nombre de metodo**
   simbolo metodo es :simbolo_metodo
5. **Expresiones regulares de metodos:**
   expresion_regular metodo es /metodo_buscado/
6. **Metodos que son accessors:**
   accessors
7. **Aridad de un metodo:**
   aridad metodo es desde..hasta
8. **Nombre de parametros:**
   parametro metodo es :simbolo_metodo
9. **Tipo de parametros:**
   parametro tipo es :req
10. **Custom (este hay que ver como se puede hacer):**

```
joinpoint do |clase, metodo|
    #Retornar true si se debe aplicar el aspecto a la clase y a ese metodo
end
```