AspectJ

La programación orientada por aspectos (AOP) ha sido utilizada para resolver el problema de modularización de código, debido a preocupaciones transversales. Está metodología es aceptada porque que reduce la cantidad de código enredado y repetido, además de reducir los costos de introducir nuevo comportamiento en la aplicación base. Para lograr esto, AOP propone una nueva unidad de modularidad llamada aspecto, la cual se basa en tres conceptos importantes: join points, pointcuts y advice.

El modelo de join points define los lugares donde los advices van a ser ubicados en la ejecución de la aplicación base. Debido a que AspectJ es una extensión de Java, el modelo de join points define puntos dentro de la ejecución de un programa como lo son llamados a métodos, llamados a constructores, escritura y lectura de atributos, etc.

El lenguaje de pointcut es utilizado para seleccionar un conjunto específico de join points. Los pointcuts dentro de AspectJ se seleccionan gracias a las características que tienen los join points. Por ejemplo, es posible seleccionar puntos específicos dentro de la ejecución gracias a los tipos de parámetros que tenga un método, tipos de retorno o simplemente definiendo patrones en los nombres.

El lenguaje de advices define la funcionalidad que debe ser ejecutada en join points específicos. El advice es un fragmento de código que es ejecutado cuando se alcanza un join point identificado por el respectivo pointcut. El advice puede ser ejecutado antes, después, o en vez de los join points que son seleccionados por el pointcut, en AspectJ corresponde a los tipos de advices before, after o around.   
AspectJ también provee instrucciones que permiten al código definido en el advice a obtener información con respecto al contexto donde se está ejecutando. Por ejemplo, puede obtener quien es el que hace el llamado de un método, los parámetros que tiene ese llamado, etc.

|  |
| --- |
| **public** aspect Logging  {  //Donde?  pointcut loggableMethods(Object o): call(\_ bar (..)) && **this**(o);  //Cuando?  before(Object o): loggableMethods(o)  {  //Qué?  System.out. println ("bar called from object " + o.toString ());  }  } |

En el ejemplo anterior se define un aspecto de monitoreo usando AspectJ. Este aspecto define un pointcut llamado loggableMethods, el cual especifica donde se debe agregar la funcionalidad de esta preocupación transversal a la aplicación base. En este ejemplo los join points son los métodos llamados *bar*, sin importar en que clase están definidos, su tipo de retorno o su lista de parámetros. El advice también define cuando y cuál debe ser el comportamiento que debe ser ejecutado en los join points escogidos. La lógica del advice es imprimir un mensaje antes de que se ejecute los join points asociados. Dentro de la lógica del advice también se está haciendo uso de las instrucciones para poder obtener información del contexto donde se está ejecutando el advice, específicamente, poder llamar el método toString() sobre el objeto que va a llamar el método *bar*.