Visualización interactiva de código Java mediante anotaciones

Guia del programador

1. Estructura de la librería

1.1. Procesador raíz (RootProcessor)

1.1.1. Funcionalidad

Este procesador es la base del sistema de procesamiento. Es el procesador que está registrado como un procesador válido de Java y es el que se encarga de recopilar toda la información.

1.1.2. Datos almacenados

La clase almacena un Map en el que por cada anotación guarda una lista de todos los usos que ha tenido esa anotación (AnnotationInfo).

1.1.3. Versión soportada

Se ha hecho override de la función de getSupportedSourceVersion para que siempre reporte al compilador que es válido para la máxima versión soportada.

1.1.4. Función process

La función oficial a la que llama la librería de Java tras encontrar cada anotación. Se encarga de convertir y almacenar toda la información sobre las anotaciones y de llamar a processData cuando se termine el procesamiento por parte de Java.

1.1.5. Función processData

Esta función es la que se encarga de la parte del trabajo que ya gestiona nuestra librería. Utilizando la información acumulada y los parámetros del paquete de anotaciones que haya configurado en ese momento, lanza los distintos procesadores de anotaciones sobre la información recogida. Finalmente, si todos ellos procesan correctamente, llama al procesador final.

1.1.6. Función getInfo

Función auxiliar que se encarga de convertir los datos que ofrece Java en nuestra clase de almacenamiento de información AnnotationInfo.

1.2. AnnotationInfo

Como ya se ha mencionado, la información que ofrece Java sobre las anotaciones está dentro de un objeto con muchos parámetros, y resulta difícil navegar por ella. Es por esto que se ha creado la clase AnnotationInfo, que almacena toda la información pertinente a un uso concreto de una anotación. Tiene los siguientes campos:

name: Nombre del elemento anotado

kind: Tipo del elemento anotado (si es una función, un parámetro, un campo, etc)

type: Clase del elemento anotado (String, List, etc)

modifiers: Modificadores del elemento anotado (public, abstract, etc)

parentName: Nombre del elemento superior al elemento anotado

values: Valores de los parámetros que tiene la anotación

En el caso de necesitarse más párametros la clase puede ampliarse. Para consultar la información completa que proporciona Java, se puede consultar la propia documentación[1]

1.3. Processor

Clase base de la que se conforman todos los procesadores. Se resume en una función que recibe una lista de usos de la anotación y devuelve un booleano para indicar si el procesamiento ha resultado correcto.

1.4. Final Processor

Clase base en la que se basan los procesadores finales. Se resume en una función sin argumentos que devuelve un booleano para indicar si el procesamiento ha resultado correcto.

[1] Documentación Java: https://docs.oracle.com/en/java/javase/17/docs/api//java.compiler/javax/annotation/processing/RoundEnvironment.html Es especialmente util el retorno de la funcion getElementsAnnotatedWith y la clase Element

2. Creación de un paquete de anotaciones

2.1. Anotaciones

Cómo primer paso para diseñar el paquete de anotaciones, es necesario dar de alta las anotaciones siguiendo la sintaxis oficial de Java. Es muy recomendable el uso de la meta-anotacion de @Target, que permite especificar a qué elementos se les puede poner la anotación, para facilitar su futuro uso. También se debe prestar especial atención a qué parámetros deben ser opcionales y cuáles no.

2.2. Configuración

Una vez creadas las anotaciones, se debe pensar sobre las relaciones de dependencia que tienen y cómo van a procesarse. Para empezar, se debe dar de alta un procesador por anotación (es posible compartir procesadores entre anotaciones) y un procesador final

2.2.1. Dependencias (DependencyTree)

Con las anotaciones y los procesadores (vacíos) creados, la primera variable a configurar es el DependencyTree. En esta variable se almacena, para cada anotación, las anotaciones de las que depende. Si una anotación no tiene dependencias, puede incluirse con una lista vacía o no incluirse. Es importante tener en cuenta que las dependencias no deben ser circulares (el sistema avisará de ello si hay dependencias circulares)

2.2.2. Procesadores (Processor)

En la variable de Processors se debe asociar cada anotación a un procesador. Varias anotaciones pueden compartir el mismo procesador, pero no es lo recomendado.

2.2.3. Procesador Final (MainProcessor)

Finalmente, en esta variable se indicará cuál es el procesador principal.

2.3. Procesadores

El objetivo de cada procesador es recibir un listado de los usos que ha tenido su anotación asociada y, o bien tomar la acción pertinente en ese momento, o bien almacenar la información para su acceso en el procesador final. Se pueden ver ejemplos de todos los puntos que se van a tratar en los procesadores del código de prueba.

2.3.1. Almacenamiento de datos

Cada procesador, como se ha comentado, debe realizar la acción que le corresponda. Cada procesador recibe información sobre sus anotaciones. En los casos como los ejemplificados en los que el objetivo es convertir las anotaciones en texto, los procesadores deben almacenar o bien la cadena de texto final, o bien información estructurada con la que luego pueda generarse.

2.3.2. Interacción entre procesadores

El almacenamiento de datos de los procesadores puede ser público. Esto implica que no solo el procesador final podrá ver los datos, si no también los otros procesadores. Es muy importante que, si un procesador accede a la información recopilada por otro, esto se añada como una dependencia en las anotaciones. En caso contrario, es posible que el otro procesador no se haya ejecutado.

2.3.3. Manejo de errores y otros requisitos

Es responsabilidad de cada procesador el finalizar con una información correcta y completa y devolver “true” o dar alguna clase de feedback sobre el error ocurrido y requisito incumplido y devolver “false”. También es posible emitir avisos aunque se devuelva “true”

2.4. Procesador Final

El procesador final es un procesador especial que no recibe ningún argumento. Sólo se le llama cuando todas las anotaciones se han procesado correctamente y han devuelto todas “true”

2.4.1. Objetivo y configuración

El objetivo del procesador final es utilizar toda la información del resto de procesadores para hacer un último procesamiento y así obtener el resultado final.

3. Adaptación de la herramienta de compilación

Junto con el código Java también se adjunta en la carpeta de Tools un pequeño archivo .bat que se encarga de la compilación. Para utilizarlo, basta con incluirlo en la carpeta src del proyecto y modificarlo en el caso de que los nombres de los paquetes sean distintos.

Tiene dos modos de ejecución:

- Modo Estándar: Utiliza archivos fuente
- Modo Librería: Utiliza .jar librería

Hay una línea cuyo objetivo es la generación del fichero jar de la librería. Este paso es totalmente opcional y está pensado para que, cuando se tenga desarrollado un paquete de anotaciones, se pueda usar este jar en lugar de tener que compartir todos los ficheros fuente.

4. Herramienta de refresco en tiempo real

Hay una segunda herramienta por encima de la herramienta de compilación, y es esta herramienta de refresco en tiempo real.

Es un bucle infinito en el que el compilador se ejecuta cada vez que detecta cambios en algún fichero fuente o de anotación. Esto hace que el proceso de visualización sea mucho más fluido e interactivo.

Para una experiencia totalmente en tiempo real se recomienda usar un visor de imágenes que las refresque de forma automática (la mayoría de los visores de imágenes no tienen esta funcionalidad y, o no refrescan la imagen nunca, o se puede refrescar de forma manual [como es el caso del visor de Eclipse])