Documentação

1. O Esfinge$_{METADATA}$

O Esfinge$_{METADATA}$ é um meta-framework extensível para validação de metadados com transparência de localização, que visa a simplificar e auxiliar principalmente o desenvolvimento de frameworks baseados em metadados. A ideia principal desta abordagem é que as anotações devem descrever a forma em que elas devem ser validadas (por meio de suas restrições), expressa através de meta-anotações (validadores).

Os principais stakeholders do Esfinge$_{METADATA}$ são desenvolvedores de aplicações ou frameworks baseados em metadados (anotações de código em Java). Os desenvolvedores utilizarão o meta-framework para a validação das anotações configuradas na aplicação que está sendo desenvolvida.

A validação de metadados é realizada por meio de meta-anotações de validação, as quais possuem uma nomenclatura específica ao domínio a ser validado. Elas são classificadas em três tipos: de validação, de configuração de validação e de configuração de localização.

As meta-anotações de validação são classificadas em metadados para a validação de propriedades dos métodos e do contexto da anotação. Elas são aplicadas nas anotações e em seus métodos que precisam ser validados.

As meta-anotações de configuração de localização definem o local onde a anotação a ser validada poderá ser definida, como nos elementos que as envolve ou dentro de outras anotações. Estas configurações de localização indiretas são especificadas pelas meta-anotações @SearchInsideAnnotations (permite buscar dentro de outras anotações) e @SearchOnEnclosingElements (permite buscar no elementos envolventes em que a anotação foi configurada, e.g. se a anotação foi definida no método, a busca poderá ser feita no método, na classe e no pacote).

A Tabela 1 resume as meta-anotações do Esfinge$_{METADATA}$ para validação de metadados, seus elementos aplicáveis, o tipo de validação realizada e a sua descrição de aplicação.

Tabela 1: Meta-anotações do Esfinge$_{METADATA}$.

Meta-anotação	Elementos Aplicáveis	Tipo de Validação	Descrição
@Prohibits	Anotação	Contexto	estabelece que se um elemento de código é anotado com a meta-anotação atual, então, ela proibirá a utilização de outra anotação informada
@NeedsToHave	Anotação	Contexto	requer que a meta-anotação atual deve ocorrer em elementos que também são anotados com uma outra anotação específica
@SearchOnEnclosingElements	Anotação	Configuração de Localização	indica que a recuperação da anotação a ser validada pode ser buscada no(s) elemento(s) que envolve (níveis acima) o objeto informado como parâmetro, até atingir o ponto extremo da busca, no qual não há mais onde procurar
@SearchInsideAnnotations	Anotação	Configuração de Localização	designa que a busca de uma determinada anotação possa ocorrer dentro de uma outra anotação recursivamente.

1.1. Configuração de Metadados

Esta seção apresenta exemplos de configuração de metadados através de anotações de código, com base nos requisitos de utilização exigidos por elas.

Suponha-se uma aplicação que tenha como requisito realizar algumas operações em momentos específicos relacionados à execução do sistema. A configuração de quais métodos devem ser executados antes ou depois de outros métodos são definidas, respectivamente pelas anotações @Before e @After. Logo, os métodos anotados com @Before não podem ter a anotação @After configurada para o mesmo elemento e vice-versa.

A seguir são apresentados exemplos de código que ilustram os cenários válidos e inválidos de configuração das anotações supramencionadas.

```
1  public class Example{
2
3     //Invalido
4     @Before
      @After
5     public void createOutputFile(){
6     //implementacao do metodo
7     }
8
9     //valido
10    @Before
11    public void createOutputFile (){
12    //implementacao do metodo
13    }
14
15    //valido
16    @After
17    public void deleteOutputFile (){
18    //implementacao do metodo
19    }}
20
```

Figura 1: Cenários de configuração válidas e inválida das anotações @Before e @After.

1.2. Utilização das meta-anotações do Esfinge$_{METADATA}$

Para que o meta-framework de validação realize a verificação da conformidade da configuração das anotações, cada anotação a ser validada deve receber meta-anotações do Esfinge$_{METADATA}$ em sua definição, especificando quais são as restrições de uso, indicando os requisitos para a sua adequada configuração. Da mesma forma, a definição dos locais(e.g. na classe, no pacote ou dentro de outras anotações) onde essa configuração é válida também pode ser especificada.

Com base no requisito de utilização da anotação @Before, a qual não pode ser configurada junto com a anotação @After, para o mesmo elemento em uma aplicação, a Figura 2 apresenta a inserção da meta-anotação (@Prohibits) do Esfinge$_{METADATA}$, que irá encarregar-se de validar as regras de utilização de @Before.

As regras de validação das anotações não devem depender de sua localização de definição. Por exemplo, a anotação @Before não depende do local de definição da anotação @After, principalmente se ela foi definida nos elementos que envolvem a anotação @Before ou dentro de outras anotações.

A Figura 3 mostra a definição da anotação @After com as meta-anotações de configuração de localização: @SearchOnEnclosingElements (permite que ela possa ser definida nos elementos envolventes) e @SearchInsideAnnotations (permite que ela possa ser definida dentro de outras anotações).

```
1 @Prohibits(After.class)
2 @Target(ElementType.METHOD)
3 @Retention(RetentionPolicy.RUNTIME)

4 public @interface Before {

5 }
```

Figura 2: Definição de @Before para a validação com o Esfinge$_{METADATA}$, meta-anotação destacada em amarelo.

```
1 @SearchOnEnclosingElements
2 @SearchInsideAnnotations
3 @Retention(RetentionPolicy.RUNTIME)
  @Target(ElementType.METHOD)

4 public @interface After {

5

6 }
```

Figura 3: Definição da anotação @After com meta-anotações de configuração de localização.

1.3. Invocação do Esfinge$_{METADATA}$

A execução da funcionalidade de validação do Esfinge$_{METADATA}$ ocorre com a chamada do método estático validateMetadataOn(), da classe MetadataValidator, passando como parâmetro a classe em que as anotações precisam ser validadas. É recomendável que um framework invoque esta funcionalidade previamente ao processamento das anotações da classe, dado que isto pode ajudar a assegurar que todos as restrições de validação sejam implementados pelo código.

Para entender como invocar o meta-framework de validação de metadados, abaixo está um código que exemplifica o procedimento.

```
1   import org.esfinge.metadata.AnnotationValidationException;
2   import org.esfinge.metadata.validate.MetadataValidator;
3
4   public class ExemploMetadata {
5
6           public static void main(String[] args) {
7                   try {
8                           MetadataValidator.validateMetadataOn(Example.class);
9
10                  } catch (AnnotationValidationException e) {
11                          e.printStackTrace();
12                  }
13          }
14  }
```

Figura 4: Exemplo de invocação do Esfinge$_{METADATA}$.

Para validar o as anotações é necessário usar o método MetadataValidator.validateMetadataOn(Example.class), se as anotações estiverem com alguma inconsistência será executado a exceção AnnotationValidationException.