ISEL-LEIC

Automatic DSL extension for idiomatic Kotlin

ISEL
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

INTRODUÇÃO

O objetivo deste trabalho é utilizar geração automática de código para gerar propriedades e funções de extensão no processo de build para a DSL de HTML, HTMLFlow, para tornar o uso da DSL em Kotlin mais fluído e natural. Outro objetivo é migrar a biblioteca de geração de código para o KotlinPoet, desta forma gerar ficheiros de código fonte em vez de gerar classes, facilitando assim a utilização de extensões à biblioteca usando Kotlin.

DSL

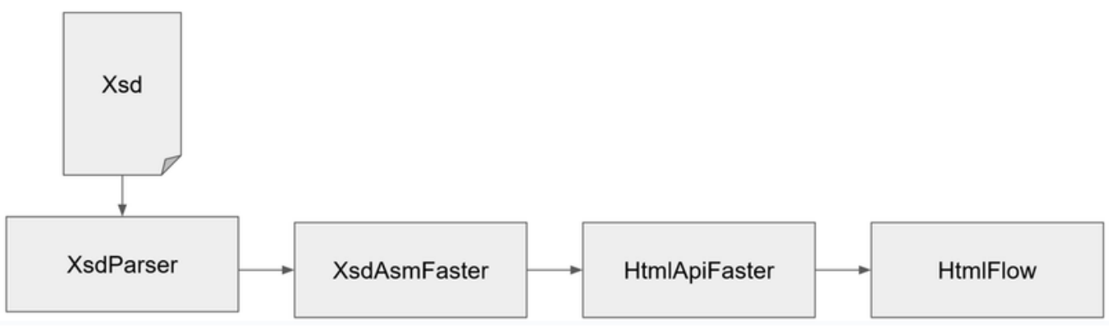
Domain Specific Language é um tipo de linguagem especifica para um domínio em particular. As DSL's são criadas com o intuito de abstrair o utilizador da necessidade de aprender as especificidades do problema em questão e focar-se em compor uma solução. Temos como exemplo de uma DSL o SQL e o Kotlin Compose.

DSL'S PARA HTML

Existem várias DSL's criadas para HTML ao longo dos anos para várias linguagens. Uma das características que distingue as várias DSL's de HTML é se são safe ou não.
Uma DSL que produz HTML safe implica que se a DSL compila isso quer dizer que o HTML gerado é válido em termos de sintaxe gerada.

HTMLFLOW

É uma DSL para Java que é HTML safe. Esta DSL tem um ficheiro de configuração (.xsd), que tem a configuração de todos os elementos HTML. Este ficheiro é lido e é automaticamente gerado o bytecode das classes necessárias para esta biblioteca.



KOTLINPOET

KotlinPoet foi a biblioteca usada para a geração automática de código. Esta biblioteca ao contrario da biblioteca usada no HTMLFlow (ASM) gera ficheiros .kt em vez de gerar ficheiros .class usando bytecode.

```
FunSpec.builder( name: "myName")
  .returns(Int::class.java)
  .addModifiers(
    KModifier.PUBLIC,
    KModifier.INLINE)
  .addParameter(
    name: "myParameter",
    Int::class.java)
  .addStatement("println(%S)", "MyString")
  .addStatement("return 10 + myParameter")
```

```
public inline fun myName(myParameter: Int): Int {
println("MyString")
return 10 + myParameter
}
```

HTMLFLOW SINTAXE

A escrita desta DSL requer sempre a chamada de duas funções para criar um elemento. Uma função de começo (Ex: body()) e outro de término do elemento, __().

```
html()
  .head()
    .title().text("string").__()
  .__()
  .body()
    .p()
      .text("other string")
    .__()
  .__()
.__();
```

OBJETIVOS

O 1º e 2º objetivos eram criar propriedades e funções de extensão por cada elemento HTML no processo build da biblioteca para tornar a escrita da DSL mais fluída e natural.

```
html
  .head
    .title.text("string").l
  .l
  .body
    .p
      .text("other string")
    .l
  .l
.l
```

```
html {
  head {
    title.text("string").l
  }
  body {
    p.text("other string").l
  }
}
```

Este objetivo foi alcançado usando reflexão para obter os elementos HTML e usando KotlinPoet foi gerado o código para as propriedades e funções de extensão para cada elemento. Por fim este procedimento foi adicionado ao processo de build Gradle.
O último objetivo forçou um estudo do funcionamento do processo de geração de código automático do HTMLFlow. Posto isto, foi usado o JavaPoet para substituir a geração de código antiga pela nova. Pelo facto do JavaPoet poder gerar código que não compila, tornou a geração de código muito mais rápida e eficiente já que as várias classes que estão a ser geradas têm dependências de outras classes que ainda não foram geradas.