ToiletPaper #144

jambit
WHERE INNOVATION WORKS

Architektur-Diagramme generieren

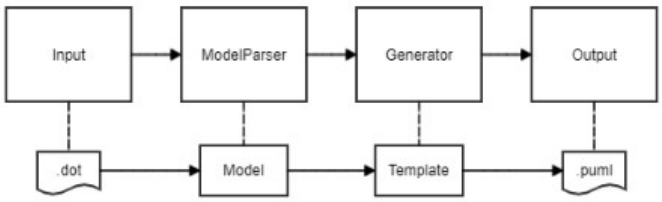
Autor: Vera Müller / Software Architect / New Business

✕ Problem

Bei neuen Projekten ist es hilfreich, ein Komponentendiagramm für die Package-Struktur anzulegen, um schnell einen Überblick zu bekommen wie die einzelnen Packages zusammenhängen. Allerdings ist eine manuelle Pflege bei vielen Änderungen sehr mühsam, vor allem bei refactorfreudigen Kollegen und ständig wechselnden Kundenvorstellungen.
Kann man das nicht automatisch erstellen lassen? Ja, aber sieht dann halt meistens nicht gut aus.
Was ist mit einem Bash-Skript, welches durch Suchen und Ersetzen die entstandene Datei zu verschönern? Ok, aber geht das nicht auch automatisch?

✔ Lösung

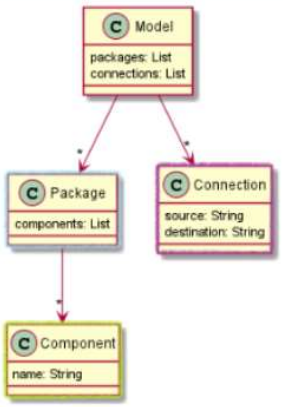
Man schreibt sich seinen eigenen Codegenerator. Dafür gibt es eine ganze Reihe von bestehenden Tools. Eine Möglichkeit ist *Apache Velocity*, eine auf Java basierende Template Engine. Mit ihr können automatisiert Textdateien und somit auch textbasierende UML-Diagramme, wie z. B. PlantUML, erzeugt werden.
Velocity bietet die Möglichkeit Templates anzulegen, sie mit Kontext zu füllen und anschließend als Datei zu speichern. Für den Kontext muss zunächst ein Model entwickelt werden, in dem alle Komponenten und die dazugehörigen Relationen gespeichert werden. Jetzt muss nur noch ein bisschen Code drumherum gebastelt werden, der die Informationen aus dem bestehenden Diagramm parst, im Model abspeichert und dieses anschließend dem Template übergibt. Beim Parsen kann festgelegt werden, ob bestimmte Pakete komplett ignoriert werden oder eventuell anders behandelt werden.



➔ Beispiel

Ein einfaches Template mit dazugehörigem Model könnte dann folgendermaßen aussehen:

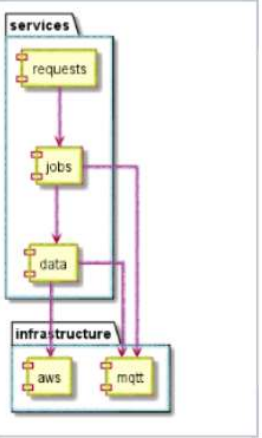
Model



Template

```
@startuml

#foreach($package in $packages)

package "$package.name" {
#foreach($component in $package.components)
    [$component.name]
#end
}
#end

#foreach($connection in $connections)
    [$connection.source] --> [$connection.destination]
#end

@enduml
```

Beispiel Output



Die farbigen Markierungen veranschaulichen, wie Model, Template und Output miteinander zusammenhängen.

+ Weiterführende Aspekte

- Dokumentation von Velocity: https://velocity.apache.org/
- Mehr Information zu PlantUML: https://plantuml.com/de/
- Java class dependency analyzer: https://docs.oracle.com/javase/9/tools/jdeps.htm#JSWOR690