2023-12-11 - DSL, Domain Specific Languages, DSL interni, DSL esterni

Domain Specific Languages

Un DSL è un **linguaggio di programmazione** che imita i termini, gli idiomi e le espressioni usate all'interno di uno **specifico contesto**. *Idealmente un esperto del contesto, senza alcuna esperienza di programmazione, dovrebbe riuscire a leggere e capire il codice (scritto nel DSL).*

Sono l'**opposto** dei linguaggi **general purpose**, sono utili solo in un certo contesto, non è possibile realizzare tutto *(spesso non sono nemmeno Touring-complete)*. Alcuni esempi possono essere *HTML* per lo sviluppo web o *Make(file)* per buildare librerie ed eseguibili.

Alcuni DSL hanno bisogno di un **linguaggio host**, come ad esempio *SQL* per l'interazione con database, non è Touring-complete e non può essere eseguito da solo, ma deve essere "inserito" in un linguaggio host.

Esempio: meccanismo di calcolo stipendio dipendenti

Questo strumento verrà usato da personale NON-informatico, con nessuna base di programmazione, ma esperto del contesto (ovvero del calcolo degli stipendi).

quello che scriveremmo da informatici è:

```
import payroll.api._
import payroll.api.DeductionsCalculator._
import payroll._
import payroll.Type2Money._

val buck = Employee(Name("Buck", "Trends"), Money(80000))
val jane = Employee(Name("Jane", "Doe"), Money(90000))

List(buck, jane).foreach { employee =>
    val biweeklyGross = employee.annualGrossSalary / 26.
    val deductions = federalIncomeTax(employee, biweeklyGross) +
                     stateIncomeTax(employee, biweeklyGross) +
                     insurancePremiums(employee, biweeklyGross) +
                     retirementFundContributions(employee, biweeklyGross)
    val check = Paycheck(biweeklyGross, biweeklyGross - deductions, deductions)
    print(format("%s %s: %s\n", employee.name.first, employee.name.last, check))
}
```

quello che vorremmo realizzare tramite un DSL è:

```
Rules to calculate an employee's paycheck:
    employee's gross salary for 2 weeks
    minus deductions for
        federalIncomeTax,            which is 25% of gross
        stateIncomeTax,              which is 5% of gross
        insurancePremiums,           which are 500. in gross's currency
        retirementFundContributions, which are 10% of gross
```

Vantaggi e Svantaggi DSL

Vantaggi:

- **incapsulamento**: vengono nascosti i dettagli implementativi
- **produttività**: viene semplificato lo sviluppo, utilizzando il gergo del contesto
- **comunicazione**: utenti non-programmatori possono essere coinvolti nello sviluppo
- **qualità**: è più facile realizzare esattamente quello che viene richiesto dagli esperti del contesto *(si assottiglia il gap tra quello che "capisce" il programmatore e quello che "vuole" il cliente)*

Svantaggi:

- **progettazione**: progettare un buon DSL è difficile, è necessario un buon tradeoff tra cosa *"mettere"* e cosa *"non mettere"*, ovvero quali costrutti includere nel linguaggio e quali no: il DSL non deve essere né *troppo poco potente* né *troppo potente ("bloated")*

- **realizzazione**: capacità di sviluppare un parser (compilatore)
- **manutenzione**: può essere difficile *(e costoso)* aggiungere o modificare parti del lingaggio

DSL interni (embedded)

I DSL interni (o embedded) sono una maniera idiomatica di scrivere del codice all'interno di un linguaggio general purpose. Non è necessario alcun parser o compilatore, viene compilato come qualsiasi altro codice di quel linguaggio.

Sono più **facili** da realizzare (non è necessario scrivere il parser), ma sono **limitati alla sintassi** del linguaggio in cui sono scritti.

Scala è un linguaggio particolarmente adatto a scrivere DSL interni grazie alla sua sintassi molto flessibile (omettere parentesi, conversione grazie agli `apply`*, ...).*

DSL esterni

I DSL esterni sono indipendenti da qualsiasi linguaggio, ma sono basati su una loro grammatica ed hanno un proprio parser (compilatore) scritto ad hoc.

Non hanno alcun **limite sintattico**, ma sono più **difficili** e costosi da realizzare.