Domain-specific Languages

Języki dziedzinowe

Adam Robaszyński-Janiec
a.r.janiec@world-loom.com

Czym są DSL?

- (często) prosty język przewidziany do rozwiązania konkretnego problemu
- Wysokopoziomowy (wyżej niż Java/C++)
- Nie muszą być kompletne w sensie Turinga (sic!)
- Łatwy do połączenia z językiem ogólnego zastosowania
- Posiada ograniczone możliwości
- Wykorzystywany podobnie jak biblioteki (API)
- Rzadko kompilują się do bytecode'u – ale potrafią do innych formatów – np. JPEG, PNG

Przykłady DSL

- Składnia grep'a, sed'a
- Język opisu grafów Graphviz
- Makra w procesorach tekstu
- Skrypty w GIMP'ie

- XSLT
- SQL

Sposoby używania

- Samodzielne przetwarzanie – np. z konsoli
- Makra w językach ogólnego zastosowania – rozwijają się do właściwego języka
- Wywoływane w trakcie wykonanie („run-time”)
- Osadzone w aplikacjach użytkownika (makra w programach biurowych)

Cele projektowe

- Mniejsze skomplikowanie
- Lepszy opis dziedziny, w której się porusza
- Zminimalizowana „nadmiarowość” - jak najmniejszy nakład prac potrzebny w przypadku zmiany wymagań

Przykładowe dane wejściowe

```
#12345678901234567890123456789012345678901234567890123456789012345678901234567890
SVCLFOWLER         10101MS0120050313.....
SVCLHOHPE          10201DX0320050315.....
SVCLTWO          x10301MRP220050329.....
USGE10301TWO         x50214..7050329.....
```

Trochę kodu – Java/C#

```
public void Configure(Reader target) {
    target.AddStrategy(ConfigureServiceCall());
    target.AddStrategy(ConfigureUsage());
  }
  private ReaderStrategy ConfigureServiceCall() {
    ReaderStrategy result = new ReaderStrategy("SVCL", typeof (ServiceCall));
    result.AddFieldExtractor(4, 18, "CustomerName");
    result.AddFieldExtractor(19, 23, "CustomerID");
    result.AddFieldExtractor(24, 27, "CallTypeCode");
    result.AddFieldExtractor(28, 35, "DateOfCallString");
    return result;
  }
  private ReaderStrategy ConfigureUsage() {
    ReaderStrategy result = new ReaderStrategy("USGE", typeof (Usage));
    result.AddFieldExtractor(4, 8, "CustomerID");
    result.AddFieldExtractor(9, 22, "CustomerName");
    result.AddFieldExtractor(30, 30, "Cycle");
    result.AddFieldExtractor(31, 36, "ReadDate");
    return result;
  }
```

Trochę inaczej - XML

```
<ReaderConfiguration>
  <Mapping Code = "SVCL" TargetClass = "dsl.ServiceCall">
    <Field name = "CustomerName" start = "4" end = "18"/>
    <Field name = "CustomerID" start = "19" end = "23"/>
    <Field name = "CallTypeCode" start = "24" end = "27"/>
    <Field name = "DateOfCallString" start = "28" end = "35"/>
  </Mapping>
  <Mapping Code = "USGE" TargetClass = "dsl.Usage">
    <Field name = "CustomerID" start = "4" end = "8"/>
    <Field name = "CustomerName" start = "9" end = "22"/>
    <Field name = "Cycle" start = "30" end = "30"/>
    <Field name = "ReadDate" start = "31" end = "36"/>
  </Mapping>
</ReaderConfiguration>
```

Jeszcze inaczej

```
mapping SVCL dsl.ServiceCall
  4-18: CustomerName
  19-23: CustomerID
  24-27 : CallTypeCode
  28-35 : DateOfCallString

mapping  USGE dsl.Usage
  4-8 : CustomerID
  9-22: CustomerName
  30-30: Cycle
  31-36: ReadDate
```

Kolejna wersja - Ruby

```
mapping('SVCL', ServiceCall) do
	extract 4..18, 'customer_name'
	extract 19..23, 'customer_ID'
	extract 24..27, 'call_type_code'
	extract 28..35, 'date_of_call_string'
end
mapping('USGE', Usage) do
	extract 9..22, 'customer_name'
	extract 4..8, 'customer_ID'
	extract 30..30, 'cycle'
	extract 31..36, 'read_date'
end
```

Podział DSL

- Zewnętrzne („external”) - napisane w innym języku niż język podstawowy – używane z wykorzystanie interpreterów, samodzielne języki (wyrażenia regularne), plik konfiguracyjne
- Wewnętrzne („internal”) - składnia taka sama jak języka podstawowego, ograniczona część możliwości

Zalety i wady ex-DSL

- Dowolność składni – łatwiejsze powiązanie z dziedziną
- Konieczność stworzenia interpretera
- Problemy w komunikacji z językiem podstawowym (oddzielne symbole)
- Brak środowiska – IDE, debugger
- (Kakofonia języków)

Zalety i wady in-DSL

- Dostępność środowisk
- Brak problemów komunikacyjnych z językiem podstawowym
- Ograniczone możliwości dostosowania do dziedziny
- Skomplikowane dla „nie-programistów”

Skąd pomysł?

- Tworzenie gier komputerowych
- Duża ilość „powtarzalnych” plansz
- 35 plansz na początek a będzie więcej
- Proste zasady, łatwe do opisania
- Koszt pracy programisty a koszt pracy designer'a plansz
- Możliwość ograniczenia „widoczności”

Bibliografia

- Wikipedia

  http://en.wikipedia.org http://pl.wikipedia.org
- Martin Fowler „Language Workbenches”
  http://www.martinfowler.com/articles/languageWorkbench.html
- GameMakers

  http://www.gamemakers.pl/home/meeting4