DSLs and Code Generation

Scott Stanchfield

http://javadude.com

Data Entry

- Programmers
  - Code
  - Binary
  - CSV
  - XML – “human-readable”
    - yeah… right…
  - JSON
- Non-Programmers / Subject-Matter Experts
  - Say “ewwwwwww”
  - Spreadsheets -> CSV
    - “Bad” characters (<-- like those quotes)
  - XML, JSON
    - Easy to make syntax mistakes

Domain-Specific Languages (DSLs)

- A “little language”
- Simpler data entry
- Less code (depends)
- More terse
- Specific to task
- Terms possibly more familiar to SME (than code)
- Better validation

Internal DSLs
(Uses programming language directly)

- Fluent API

```
// Java example
new Robot()
        .turnLeft(90)
        .move(100)
        .shoot();
```

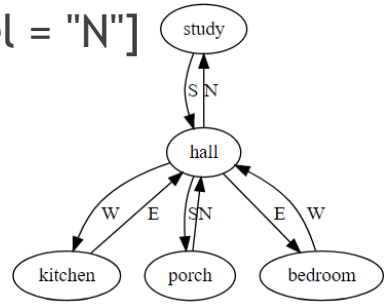
- More Complex

```
// Groovy example
Email.make {
        to “Luke"
        from “Han"
        subject “Don’t Get Cocky, Kid"
        body “…”
        send
}
```

External DSLs (Input parsed by language)

// GraphViz dot example

```
digraph map {
        study -> hall [label = "S"]
        kitchen -> hall [label = "E"]
        hall -> porch [label = "S"]
        hall -> bedroom [label = "E"]
        hall -> study [label = "N"]
        hall -> kitchen [label = "W"]
        bedroom -> hall [label = "W"]
        porch -> hall [label = "N"]
}
```



// Custom DSL example

```
carryable item key "It is a shiny brass key"

carryable item letter "It reads You win!"

fixed item safe
    "It is a very heavy locked box. There is a keyhole
on it"
  opens with key locked closed
  contains letter

room bedroom "This is where you sleep"
  contains key
  exit west hall

start in porch
```

Today's Examples

Text Adventure Game

- Two DSLs
  - Commands (ANTLR)
  - Data (xText)

ANother Tool for Language Recognition

- Run actions when input matched

xText

- Model classes (Build)
- Generated IDE (Build)
- Read Model Instance (Runtime)

Object Model Generation

- One DSL
  - Object Model (xText)

xText

- Model classes (Build)
- Generated IDE (Build)