DSLs in Python

Marek Kubica

13. März 2008

Der Plan

- Dieser Vortrag lebt vom Feedback!
- daher: Fragen und Anmerkungen explizit erwünscht
- Chris Okazaki hat Recht: Vortrag in eine Richtung ist langweilig

Ganz kurze Einleitung

DSLs?

- Domänenspezifische Programmiersprachen – *Domain Specific Languages*
- Kleine Sprachen die für kleine, klar umrissene Dinge eingesetzt werden
- gegenteil wären General Purpose Sprachen(Python)
- Beispiele: GraphViz, sed, Vim script, Elisp, JavaScript
- letztere sind auch General Purpose Sprachen

Ganz kurze Einleitung

DSLs?

- Domänenspezifische Programmiersprachen – *Domain Specific Languages*
- Kleine Sprachen die für kleine, klar umrissene Dinge eingesetzt werden
- gegenteil wären General Purpose Sprachen(Python)
- Beispiele: GraphViz, sed, Vim script, Elisp, JavaScript
- letztere sind auch General Purpose Sprachen
- Werden aber für gewisse klar gegliederte Aufgabenbereiche eingesetzt
- etwa Editor-Skripting, DOM-Editieren

Zwei Arten von DSLs

Externe DSLs

- Sind komplette Sprachen, mit allem drum & dran
- Eigene Grammatik, eigene Token, Lexer, Parser, Verhalten
- Komplex zu erstellen, dafür aber viele Freiheiten in der Gestaltung

Zwei Arten von DSLs

Externe DSLs

- Sind komplette Sprachen, mit allem drum & dran
- Eigene Grammatik, eigene Token, Lexer, Parser, Verhalten
- Komplex zu erstellen, dafür aber viele Freiheiten in der Gestaltung
- Schließlich sind es eigenständige Sprachen

Interne DSLs

- Untermengen der sie umgebenden Sprache
- Also limitierter - aber simpler und einfacher zu verstehen
- Einfach zu implementieren - man wälzt alles auf die Hostsprache ab
- Oft in Ruby eingesetzt (Rails), aber auch in Python möglich
- (in einem anderen Umfang)

Und nun?

- Beschäftigen wir uns mit internen DSLs in Python, an einem einfachen Beispiel

Und nun?

- Beschäftigen wir uns mit internen DSLs in Python, an einem einfachen Beispiel
- Warum?

Und nun?

- Beschäftigen wir uns mit internen DSLs in Python, an einem einfachen Beispiel
- Warum?
- Weil man damit einige Probleme lesbarer lösen kann.
- lesbar = besser

Ohne DSLs

- Was macht der Code in “playfair-dsless.py”?
- Was ist gut, was ist schlecht?

Das Problem - Playfair implementieren

Wikipedia bietet ein gutes Beispiel dafür

Klartext

- Klartext wird in Großbuchstaben gewandelt
- Umlaute aufgelöst, J zu I, aufeinanderfolgende Buchstaben durch X getrennt
- in Buchstabenpaare geteilt

Schlüssel

- Alphabet mit 25 Buchstaben (ohne J)
- In 5x5-Matrix gesetzt
- Buchstaben des Schlüssels werden eingesetzt
- Rest wird alphabetisch aufgefüllt

Playfair-Algorithmus

Verfahren

Im großen und ganzen geht es um das tauschen von Buchstaben in der 5x5 Matrix.

Lass Code sprechen

Was brauchen wir so?

- Ein Alphabet ohne den Buchstaben J
- Einen Matrix-Datentyp
- Einen Matrix-Element-Datentyp
- etwas Glue-Code um das zu verbinden

Wie sieht das dann aus?

- “playfair.py”
- wie siehts im Vergleich aus?

Was hat uns das gebracht?

- Der Code ist länger, ja
- Der Code ist leichter zu verstehen
- Wir haben einige nette Python-Features genutzt :)

Kompliziertere DSLs

Interne

- Metaprogrammierung (siehe Elixir, Django)
- Bytecode-Hacks

Externe

- Parser (PLY, Pyparsing, Pysec)
- Andere Sprachen embedden (Lua, Guile)