Clojure

Ein Lisp für die Java-VM

Entwicklungsprozesse in Open Source Projekten

Gliederung

1. Ziele
2. Designentscheidungen
    1. Warum Lisp
    2. Warum die JVM
    3. Warum nicht Common Lisp oder Scheme
3. Sprachfeatures
4. Demo von Sprachfeatures
    1. Unveränderliche Datenstrukturen
    2. Nebenläufigkeit mit Refs und STM
    3. Interoperabilität mit Java
5. Das Clojure-Projekt und mein Beitrag
6. Ressourcen
7. Quellen

1. Ziele

- Ausdrucksstarke Programmiersprache
- Interoperabilität mit Java-Programmen
- Unterstützung von Nebenläufigkeit

2.1 Warum Lisp

- Homoikonisch
  - Erweiterung der Sprache durch Makros möglich
- Read-Evaluate-Print-Loop
  - Schrittweise Softwareentwicklung
  - Einfaches Testen neuer Ideen
- Funktionale Programmierung
  - First-Class-Funktionen und Funktionen höherer Ordnung
- Dynamische Typisierung

2.2 Warum die JVM

- Interoperabilität mit Java
  - viele Libraries verfügbar
- Interoperabilität mit Scala, Groovy, JRuby ...
- Viele Tools verfügbar
- Infrastruktur vorhanden
  - Hotspot-VM
  - ASM Bytecode framework

2.3 Warum nicht CL oder Scheme

- Scheme
  - Tail Calls und Continuations inkompatibel mit Java
  - Wenige portable Libraries
- Common Lisp
  - Eigene Plattform: inkompatibel zu Java
  - Keine Threads im Common-Lisp-Standard
  - Nicht funktional genug
- Kein bestehendes Lisp erfüllt die Anforderungen
- Rich Hickey entwickelt Clojure
  - Ab Oktober 2007 Open-Source-Projekt

3. Sprachfeatures

- Standardbibliothek mit unveränderlichen Datenstrukturen
  - „sharing structure“
- Unterstützung für Nebenläufigkeit durch
  - Software Transactional Memory
  - Atome, Refs, Agenten
- Interoperabilität mit Java
  - Kompromiss: nur explizite Tail Calls (mit loop und recur)
  - Aufrufen von Java-Code aus Clojure
  - Compiler erzeugt Klassendateien aus Clojure-Code
- Generische Methoden mit eigener Dispatch-Funktion

4. Demo von Sprachfeatures

- Unveränderliche Datenstrukturen
- Nebenläufigkeit mit Refs und STM
- Interoperabilität mit Java

5. Das Clojure-Projekt und mein Beitrag

- Lizenz: EPL
  - Open Source
- SVN-Repository öffentlich
  - Open Process?
- Rechtliche Absicherung: „Rich Hickey Contributor Agreement“
- Kein Bugtracker, aber To-Do-Liste
  - „Arbitrary Symbols between ||“
- Nachfrage in #clojure auf freenode.org

5. Das Clojure-Projekt und mein Beitrag

- Mein Patch wurde abgelehnt
  - Und zwar vom BDFL persönlich

- Rich Hickey:

„[…] it would be better not to replicate the logic for macro characters etc that already exists in the reader. That's not to say the reader stuff is ready for reuse as is, but making it so is the right thing. I haven't fully thought out how this ought to work (there's likely a speed/space tradeoff somewhere), and it is pretty low priority.“

http://groups.google.com/group/clojure/browse_thread/thread/aad6aa5468ea1d55#

- Die To-Do-Liste jetzt offline, dafür gibt es einen Google Code Bugtracker ohne dieses Feature

6. Ressourcen

- Projektseite
  - http://clojure.org/
- Google Group (für Clojure-Entwickler und Anwender)
  - http://groups.google.com/group/clojure
- Google-Code-Projekt (mit SVN und Bugtracker)
  - http://code.google.com/p/clojure/
- #clojure auf freenode.org
- Wikibook „Clojure Programming“
  - http://en.wikibooks.org/wiki/Clojure_Programming

7. Quellen

- Clojure for Lisp Programmers
  - Rich Hickey, Boston Lisp Meeting October 2008
  - Part 1: http://clojure.blip.tv/#1319826
  - Part 2: http://clojure.blip.tv/#1319721
- Clojure: API documentation
  - http://clojure.org/api
- Clojure Concurrency
  - Rich Hickey, Western Massachusetts Developers Group Meeting 2008
  - http://clojure.blip.tv/#819147
- Clojure - A Dynamic Programming Language for the JVM
  - Rich Hickey, JVM Summit 2008
  - http://www.infoq.com/presentations/hickey-clojure