fhwedel
UNIVERSITY OF APPLIED SCIENCES

Prof. Dr. Ulrich Hoffmann, Prof. Dr. Uwe Schmidt

Clojure

Lisp in der JVM

Michael Krüger (inf6065)

Übersicht

1. Hintergrund
2. Designziele
3. Spracheigenschaften
4. Klassenanalyse

fhwedel
UNIVERSITY OF APPLIED SCIENCES

1. Hintergrund

- Open Source Projekt, initiiert und betreut durch Rich Hickey
  - Common Lisp Nutzer
  - Hat vorher zwei CL-Java Bridges entwickelt
  - Praktiker, nicht Forscher
  - Broadcast Animation, Scheduling, Election → Nebenläufigkeit

fhwedel
UNIVERSITY OF APPLIED SCIENCES

2. Designziele

1. Ein Lisp
2. Unveränderliche Datenstrukturen (aus der funktionalen Programmierung)
3. Unterstützung von Nebenläufigkeit (Koordination von Zustandsübergängen auf Sprachebene)
4. Nutzung der Java Infrastruktur (JVM)

fhwedel UNIVERSITY OF APPLIED SCIENCES

2.1 Lisp

- REPL – Read Eval Print Loop
- Syntax (vereinfachte S-Expressions)
- Typenkonzept (dynamisch)
- Code as Data
- Makros

Allerdings: Namen haben teilweise neue Bedeutungen → keine Kompatibilität zu altem Lisp-Code

2.2 Datenstrukturen

fhwedel UNIVERSITY OF APPLIED SCIENCES

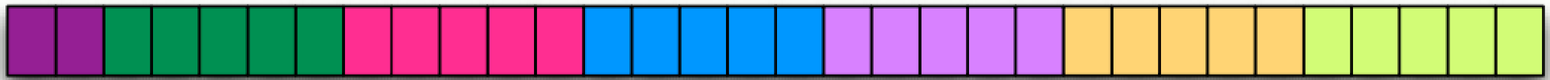
- Listen
- Vektoren
- Mengen
- Tabellen

(Alles was von der JVM bereitgestellt wird)

Eigenschaften

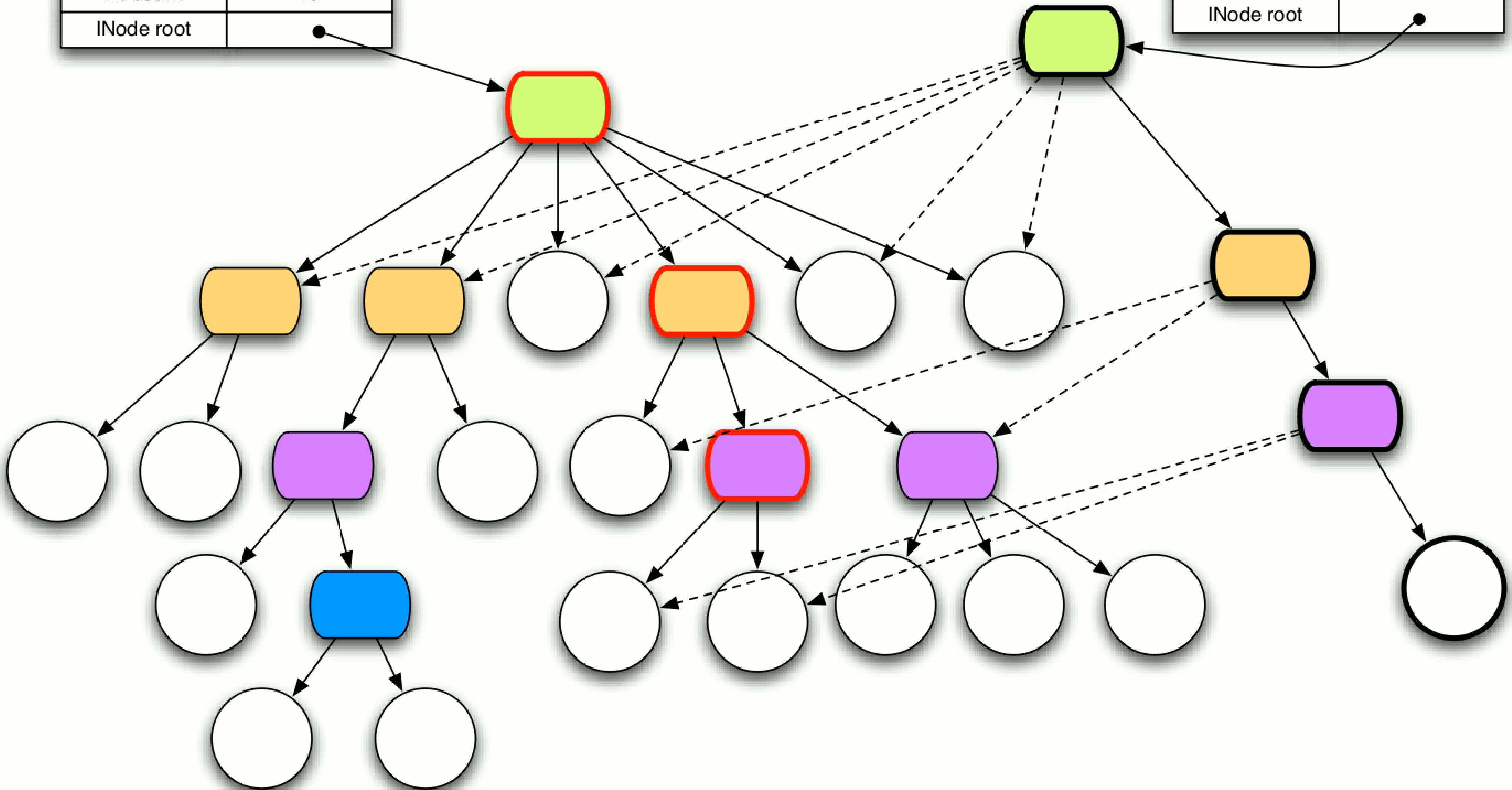
- Unveränderlich
- Persistent
- Lazy (wenn möglich)
- Vollständige Unterstützung durch die Sprache und den Reader

Bit-partitioned hash tries



http://clojure.googlegroups.com/web/ClojureConcurrencyTalk.pdf

Path Copying



http://clojure.googlegroups.com/web/ClojureConcurrencyTalk.pdf

2.3 Unterstützung von Nebenläufigkeit

fhwedel UNIVERSITY OF APPLIED SCIENCES

- Ref
- Atom
- Agent
- Dynamic Var

- Indirekte Referenzen auf unveränderliche Datenstrukturen
- Koordinierter Zugriff auf Referenzen
  - Automatisch bzw. notwendig
  - Nicht lock-basiert

2.4 JVM

fhwedel
UNIVERSITY OF APPLIED SCIENCES

- Typsystem
- Garbage Collection
- Portabilität
- Nutzung der vorhandenen Infrastruktur in Unternehmen

„It's ok to hate Java, but love the JVM." Rich Hickey

Multimethoden

fhwedel UNIVERSITY OF APPLIED SCIENCES

- Dispatch über alle Parameter einer Methode
- Wahl der geeigneten Methode durch Auswertung einer Dispatch-Funktion
- Resultat der Dispatch-Funktion beliebig → Berücksichtigung der aktuellen Werte möglich

Java Interoperabilität

fhwedel UNIVERSITY OF APPLIED SCIENCES

- Spezielle Methoden für alle Konzepte, die durch JVM unterstützt werden, sind vorhanden:
  - Klassen
  - Objekte
  - Primitive Typen
  - Arrays ...
- Alle Java Bibliotheken nutzbar
- Clojure kompiliert zu nativem Java Bytecode → In Java nutzbar

Makros

fhwedel
UNIVERSITY OF APPLIED SCIENCES

- Auswertung durch Reader → zur Compilezeit
- Codetransformationen, dadurch Schaffung eigener Sprachstrukturen möglich
- Änderung der Reader-Makros nicht möglich (wird noch diskutiert)
- Veränderung der Sprache → extreme Sorgfalt erforderlich

Beispiel Makro

fhwedel UNIVERSITY OF APPLIED SCIENCES

```
(defn unless [pred body]
  (if pred nil body))
```

```
=> (unless false :t)
   :t

=> (unless true :t)
   nil
```

Aber:

```
=> (unless true (println :t))
   :t
   nil
```

```
(defmacro unless [pred body]
  (list 'if pred nil body))
```

Metadaten

fhwedel UNIVERSITY OF APPLIED SCIENCES

- Vollständig in Sprache integriert
- Als Tabelle an Symbolen, Sequenzen
  - Keywords als Schlüssel
- Beliebig verwendbar, z.B. für:
  - Dokumentation
  - Type Hints
  - Beschreibungen der Eigenschaften von Werten
  - Vor- & Nachbedingungen
  - Testfunktionen ...

Klassenanalyse

fhwedel
UNIVERSITY OF APPLIED SCIENCES

- Wie sieht der erzeugte Bytecode aus?

Quellen

fhwedel UNIVERSITY OF APPLIED SCIENCES

- http://clojure.org – Die Clojure Heimatseite
- http://github.com/richhickey/clojure – Git Repository
- http://clojure.blip.tv/ – Präsentationen
- http://www.infoq.com/author/Rich-Hickey – weitere Präsentationen
- Halloway, S. (2009): Programming Clojure. Raleigh, Dallas: Pragmatic Bookshelf
- Abelson, H. Sussman, G.J. (1996): Structure and Interpretation of Computer Programms. Cambridge, London: MIT Press

Fragen?