Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

8. Mai 2020

Modern programming languages: Clojure

Christian Wolff

Für das Proseminar
„Softwareentwicklung in der Wissenschaft“

Outline

- Clojure
  - What is Clojure?
  - What is Lisp?
  - Why Clojure?
- Features of Clojure explained
  - Functional programming
  - Concurrency
  - Dynamic/ dynamic type structure
- Dynamic development with REPL
  - Examples/ pmap
- Conclusion/ Sources

Clojure

Clojure:
What is Clojure?

- Modern Lisp-Dialect
  - **CL**o**J**ure (C#, Lisp, Java -> Closure)

- Rich Hickey (BDFL)
  - Self-funded development
  - First release in 2007
  - Community-driven development (clojure.org)

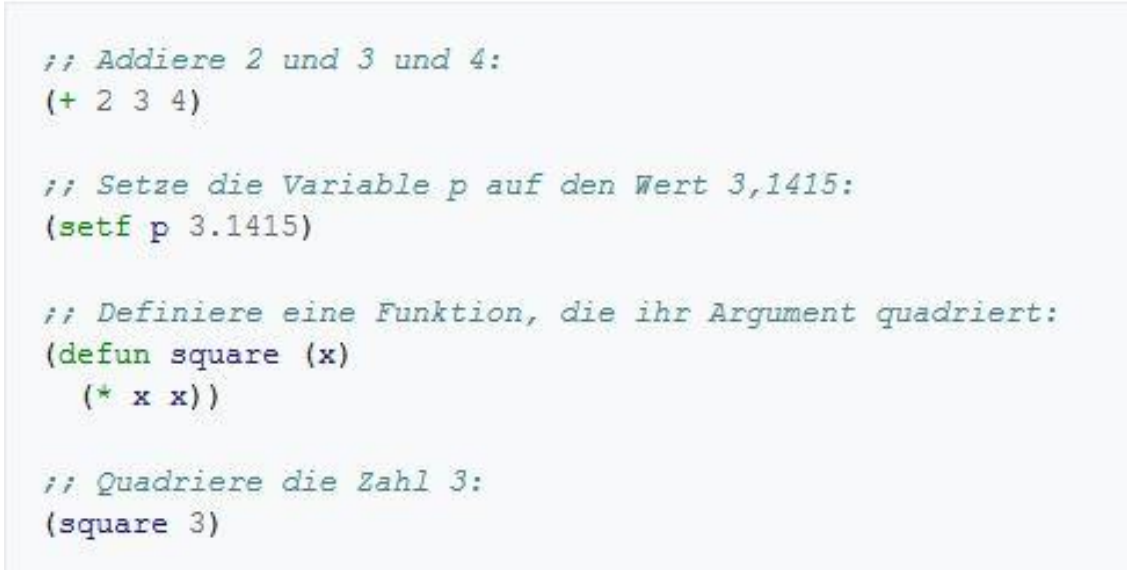
- Commercial support provided by Cognitect
  - Global conferences every year

Clojure:
What is Lisp?

- LISP
  - „**LIS**t **P**rocessing“
  - complex lists
- Massachusetts Institute of Technology (MIT) (1958)
- „Common Lisp“, „Scheme“, „Clojure“
- Homoiconic
- Macros
- Dynamic

Clojure:
What is Lisp?

Common Lisp



```
;; Addiere 2 und 3 und 4:
(+ 2 3 4)

;; Setze die Variable p auf den Wert 3,1415:
(setf p 3.1415)

;; Definiere eine Funktion, die ihr Argument quadriert:
(defun square (x)
  (* x x))

;; Quadriere die Zahl 3:
(square 3)
```

illus. 1: https://de.wikipedia.org/wiki/Lisp

Clojure:
Why Clojure?

- Runs on the JVM
- Functional programming
  - First-class-functions
- Dynamic/ dynamic type system
- Concurrency
- dynamic compilation/ REPL
  - Read-eval-print-loop

Clojure:
Why Clojure?

- Lazy sequences
  - „call-by-need“
  - Delays the evaluation of an expression until its value is needed
  - Increases performance

- Persistent data structures
  - Immutable -> no change, but update
  - „modify“

- No text-based syntax
  - „just“ lists
  - Data structures are the code

Clojure:
Why Clojure?

Source: Clojure, Made Simple

„Clojure shrinks our code base to about one-fifth
the size it would be if we had written in Java“

Anthony Marcar –
Senior Architect, WalmartLabs

Features of Clojure explained: Functional programming

Features of Clojure explained

Features of Clojure explained: Functional programming

- „first-class-citizen/ -function“ (Treated as any other data)
  - Bound to names
  - Passed as arguments
  - Returned from other functions

- Not a sequence of instructions, but complex functions

- Better understanding e.g. for calculations

Features of Clojure explained: Functional programming

Sourcecode by G. Mania
and https://gist.github.com/akonring/7804273

```
(defn fact [n]
  (loop [i n result 1]
    (if (zero? i)
      result
      (recur (dec i)
             (* result i)))))
```

```
(defn fac [n]
   (if (zero? n) 1
        (* n (fac (dec n)))))
```

Features of Clojure explained: Dynamic/ dynamic type structure

- No manual reservation for space needed

- No declaration during compiling time
  - Java (int i = 5)
  - Clojure (def i 5)

- Declaration is made during runtime

Features of Clojure explained: Concurrency

- Using multiple threads

- Immutable Data Structures
  - Copy of the object for each thread
    - No conflicts
    - No synchronisation

- Software transactional memory
  - Preventing deadlocks and inconsistencies

Dynamic development with REPL

Dynamic development with REPL

Summary/ Conclusion

- Size!
- „Unknown“
  - NASA, Apple, Netflix, Walmart (data management system)
- Experimental implementations
  - Perl
  - Python
  - C++

Sources

- Clojure (reading):
  - www.clojure.org
  - https://www.braveclojure.com/introduction/
  - https://de.wikipedia.org/wiki/Clojure
  - https://en.wikipedia.org/wiki/Clojure

Sources

- Clojure (video):
  - Why Clojure? - Derek Slager
  - https://www.youtube.com/watch?v=BThkk5zv0DE&t=

  - Clojure for Java Programmers Part 1 - Rich Hickey
  - https://www.youtube.com/watch?v=P76Vbsk_3J0&t=

  - Clojure - what's so great about it?
  - https://www.youtube.com/watch?v=vfnL5Dai77Q&t=s

  - Expert to Expert: Rich Hickey and Brian Beckman - Inside Clojure
  - https://www.youtube.com/watch?v=wASCH_gPnDw&t=

  - Clojure Made Simple
  - https://www.youtube.com/watch?v=VSdnJDO-xdg

Sources

- Others:
  - Lisp
    - https://de.wikipedia.org/wiki/Lisp
    - https://en.wikipedia.org/wiki/Lisp_(programming_language)
    - https://de.wikipedia.org/wiki/Lambda-Kalk%C3%BCl
  - Functional Programming
    - https://en.wikipedia.org/wiki/Functional_programming
    - https://de.wikipedia.org/wiki/Funktionale_Programmierung
    - https://de.wikipedia.org/wiki/Closure_(Funktion)

Sources

- Others:
  - Concurrency
    - https://de.wikipedia.org/wiki/Parallele_Programmierung
    - https://en.wikipedia.org/wiki/Concurrency_(computer_science)
    - https://de.wikipedia.org/wiki/Nebenl%C3%A4ufigkeit
    - https://en.wikipedia.org/wiki/MapReduce
    - https://en.wikipedia.org/wiki/Software_transactional_memory
    - https://de.wikipedia.org/wiki/Transaktionaler_Speicher

Sources

- Others- other:
  - https://de.wikipedia.org/wiki/Programmiersprache
  - https://en.wikipedia.org/wiki/Persistent_data_structure
  - https://en.wikipedia.org/wiki/Macro_(computer_science)#Syntactic_macros
  - https://en.wikipedia.org/wiki/Lazy_evaluation
  - https://en.wikipedia.org/wiki/Benevolent_dictator_for_life
  - https://en.wikipedia.org/wiki/Immutable_object#Immutable_variables
  - https://en.wikipedia.org/wiki/Thread_safety
  - https://de.wikipedia.org/wiki/First-Class-Funktion
  - https://de.wikipedia.org/wiki/Homoikonizit%C3%A4t
  - https://en.wikipedia.org/wiki/Homoiconicity
  - https://en.wikipedia.org/wiki/First-class_citizen
  - https://en.wikipedia.org/wiki/First-class_function

Sources

- Sourcecodes:
  - Factorial loop by G. Mania
  - Factorial redundancy
  - https://gist.github.com/akonring/7804273
  - https://clojuredocs.org/clojure.core/pmap
- Tools:
  - REPL
    - https://repl.it/languages/clojure