

## <<Clojure编程乐趣>>

### 图书基本信息

书名：<<Clojure编程乐趣>>

13位ISBN编号：9787115319494

10位ISBN编号：7115319499

出版时间：2013-11-1

出版时间：人民邮电出版社

作者：Michael Fogus,Chris Houser

译者：郑晔

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

## <<Clojure编程乐趣>>

### 内容概要

Clojure是一门Lisp方言。

它通过函数式编程技术，直接支持并发软件开发，得到众多开发人员的欢迎。

《Clojure编程乐趣》并非Clojure初学指南，也不是一本Clojure的编程操作手册，而是通过对Clojure详尽地探究，教授函数式的程序设计方式，帮助读者理解和体会Clojure编程的乐趣，进而开发出优美的软件。

全书分为5个部分共13章。

第1部分是基础，包括第1章到第3章，从Clojure背后的思想开始，介绍了Clojure的基础知识，并带领读者初步尝试Clojure编程。

第2部分包括第4章和第5章，介绍了Clojure的各种数据类型。

第3部分是第6章和第7章，介绍了函数式编程的特性。

第4部分包括第8章到第11章，分别介绍了宏、组合数据域代码、Clojure对Java的调用，以及并发编程等较为高级的话题。

第5部分为第12章和第13章，探讨了Clojure的性能问题及其带给我们的思考。

《Clojure编程乐趣》适合有一定基础的Clojure程序员阅读，进而掌握函数编程的思考方法和程序设计方法，也可以作为读者学习函数式编程的参考资料。

## <<Clojure编程乐趣>>

### 作者简介

<http://blog.fogus.me/>

## <<Clojure编程乐趣>>

### 书籍目录

#### 第1部分 基础

##### 第1章 Clojure哲学

###### 1.1 Clojure之道

###### 1.1.1 简单

###### 1.1.2 专注

###### 1.1.3 实用

###### 1.1.4 清晰

###### 1.1.5 一致

###### 1.2 为何（又一种）Lisp

###### 1.2.1 优美

###### 1.2.2 极度灵活

###### 1.2.3 代码即数据

###### 1.3 函数式编程

###### 1.3.1 一个可行的函数式编程定义

###### 1.3.2 函数式编程的内涵

###### 1.4 Clojure为何不是面向对象的

###### 1.4.1 定义术语

###### 1.4.2 命令式“烘烤”

###### 1.4.3 OOP提供的大多数东西，Clojure也有

###### 1.5 小结

##### 第2章 Clojure疾风式教程

###### 2.1 标量

###### 2.1.1 数字

###### 2.1.2 整数

###### 2.1.3 浮点数

###### 2.1.4 有理数

###### 2.1.5 符号

###### 2.1.6 关键字

###### 2.1.7 字符串

###### 2.1.8 字符

###### 2.2 组合起来：集合

###### 2.2.1 list

###### 2.2.2 vector

###### 2.2.3 map

###### 2.2.4 set

###### 2.3 付诸实现：函数

###### 2.3.1 调用函数

###### 2.3.2 定义函数

###### 2.3.3 用def和defn简化函数定义

###### 2.3.4 以#（）定义原位（in—place）函数

###### 2.4 var

###### 2.5 局部量、循环和block

###### 2.5.1 block

###### 2.5.2 局部量

###### 2.5.3 循环

## <<Clojure编程乐趣>>

### 2.6 防止发生：quote

#### 2.6.1 求值

#### 2.6.2 quote

#### 2.6.3 反quote

#### 2.6.4 反quote拼接

#### 2.6.5 auto—gensym

### 2.7 与Java互操作

#### 2.7.1 访问静态类成员

#### 2.7.2 创建Java实例

#### 2.7.3 用运算符访问Java实例成员

#### 2.7.4 设置Java实例属性

#### 2.7.5 ...宏

#### 2.7.6 doto宏

#### 2.7.7 定义类

### 2.8 异常环境

### 2.9 命名空间

#### 2.9.1 用ns创建命名空间

#### 2.9.2 用：require加载其他命名空间

#### 2.9.3 用：use加载和创建映射

#### 2.9.4 用：refer创建映射

#### 2.9.5 用：import加载Java类

### 2.10 小结

## 第3章 小试牛刀

### 3.1 真值

#### 3.1.1 什么是真

#### 3.1.2 不要创建布尔对象

#### 3.1.3 nil vs. false

### 3.2 小心翼翼nil双关

### 3.3 解构

#### 3.3.1 你的任务，你应该选择接受

#### 3.3.2 解构vector

#### 3.3.3 解构map

#### 3.3.4 解构函数参数

#### 3.3.5 解构vs.访问器方法

### 3.4 用REPL做试验

#### 3.4.1 试验seq

#### 3.4.2 试验图形化

#### 3.4.3 知识汇总

#### 3.4.4 出错之时

#### 3.4.5 只为乐趣

### 3.5 小结

## 第2部分 数据类型

## 第4章 标量

### 4.1 理解精度

#### 4.1.1 截断（Truncation）

#### 4.1.2 提升（Promotion）

#### 4.1.3 上溢（Overflow）

## <<Clojure编程乐趣>>

- 4.1.4 下溢 ( Underflow )
- 4.1.5 舍入错误 ( Rounding errors )
- 4.2 有理数
  - 4.2.1 为什么是有理数
  - 4.2.2 怎样才是有理数
  - 4.2.3 有理数的合理性
- 4.3 使用关键字的时机
  - 4.3.1 关键字与符号有怎样的差别
  - 4.3.2 限定关键字
- 4.4 符号解析
  - 4.4.1 元数据
  - 4.4.2 符号与命名空间
  - 4.4.3 Lisp—1
- 4.5 正则表达式——第二个问题
  - 4.5.1 语法
  - 4.5.2 函数
  - 4.5.3 小心可变匹配器 ( matcher )
- 4.6 总结
- 第5章 组合数据类型
  - 5.1 持久化、序列和复杂度
    - 5.1.1 “ 你一直用着这个词。  
我认为，这并不意味着它就是你以为的含义 ”
    - 5.1.2 序列术语及其含义
    - 5.1.3 大O
  - 5.2 vector：创建和使用其各种变体
    - 5.2.1 构建vector
    - 5.2.2 大vector
    - 5.2.3 vector当做栈
    - 5.2.4 使用vector而非reverse
    - 5.2.5 子vector
    - 5.2.6 vector当做MapEntry
    - 5.2.7 vector不是什么
  - 5.3 list：Clojure代码form的数据结构
    - 5.3.1 像Lisp那样的list
    - 5.3.2 list当做栈
    - 5.3.3 list不是什么
  - 5.4 如何使用持久化队列
    - 5.4.1 什么都没有的队列
    - 5.4.2 入队
    - 5.4.3 获取
    - 5.4.4 出队
  - 5.5 持久化set
    - 5.5.1 Clojure set的基本属性
    - 5.5.2 用sorted—set保持set的顺序
    - 5.5.3 contains？
    - 5.5.4 clojure.set

## <<Clojure编程乐趣>>

### 5.6 思考map

#### 5.6.1 hash map

#### 5.6.2 以有序map保持键值的顺序

#### 5.6.3 用数组map保持插入顺序

### 5.7 知识汇总：在序列里查找某项的位置

### 5.8 小结

## 第3部分 函数式编程

## 第6章 惰性与不变性

### 6.1 关于不变性

#### 6.1.1 定义不变性

#### 6.1.2 固步自封—不变性

### 6.2 设计一个持久化玩具

### 6.3 惰性

#### 6.3.1 以“逻辑与”熟悉惰性

#### 6.3.2 理解lazy—seq的秘诀

#### 6.3.3 丢掉头

#### 6.3.4 采用无限序列

#### 6.3.5 delay和force宏

#### 6.3.6 知识汇总：一个惰性的快速排序程序

### 6.4 小结

## 第7章 函数式编程

### 7.1 各种形式的函数

#### 7.1.1 一等函数

#### 7.1.2 高阶函数

#### 7.1.3 纯函数

#### 7.1.4 命名实参

#### 7.1.5 使用前置条件和后置条件约束函数

### 7.2 闭包

### 7.3 递归思考

#### 7.3.1 普通递归

#### 7.3.2 尾递归和recur

#### 7.3.3 勿忘trampoline

#### 7.3.4 延续传递风格

### 7.4 知识汇总：A\*寻路

#### 7.4.1 A\*实现

#### 7.4.2 A\*实现的笔记

### 7.5 小结

## 第4部分 大规模设计

## 第8章 宏

### 8.1 数据即代码即数据

#### 8.1.1 语法quote、反quote和拼接

#### 8.1.2 宏之经验谈

### 8.2 定义控制结构

#### 8.2.1 不用语法quote定义控制结构

#### 8.2.2 使用语法quote和反quote定义控制结构

### 8.3 组合form的宏

### 8.4 使用宏改变form

## <<Clojure编程乐趣>>

### 8.5 使用宏控制符号解析

#### 8.5.1 回指

#### 8.5.2 (具有争议地) 有用的选择性名字捕获

### 8.6 使用宏管理资源

### 8.7 知识汇总：返回函数的宏

### 8.8 小结

## 第9章 组合数据与代码

### 9.1 命名空间

#### 9.1.1 创建命名空间

#### 9.1.2 只暴露所需

#### 9.1.3 声明性包含和排除

### 9.2 以通用设计模式探索Clojure多重方法

#### 9.2.1 组成部分

#### 9.2.2 用法

#### 9.2.3 以多重方法拯救

#### 9.2.4 处理继承行为的特别继承

#### 9.2.5 解析层次中的冲突

#### 9.2.6 真正的最大功率任意分发

### 9.3 类型、协议和记录

#### 9.3.1 记录

#### 9.3.2 协议

#### 9.3.3 用deftype从更原始的基础开始构建

### 9.4 知识汇总：国际象棋移动的流畅构建器

#### 9.4.1 Java实现

#### 9.4.2 Clojure实现

### 9.5 小结

## 第10章 Java.next

### 10.1 使用proxy动态生成对象

### 10.2 Clojure gen—class和GUI程序设计

#### 10.2.1 命名空间作为类的规范

#### 10.2.2 以Clojure探索用户界面设计与开发

### 10.3 Clojure同Java数组的关系

#### 10.3.1 数组的类型：原生与引用

#### 10.3.2 数组可变性

#### 10.3.3 那个不幸的命名约定

#### 10.3.4 多维数组

#### 10.3.5 调用可变方法 / 构造函数

### 10.4 所有Clojure函数都实现

#### 10.4.1 java.util.Comparator

#### 10.4.2 java.lang Runnable

#### 10.4.3 java.util.concurrent.Callable

### 10.5 在Java API里使用Clojure数据结构

#### 10.5.1 java.util.List

#### 10.5.2 java.lang.Comparable

#### 10.5.3 java.util.RandomAccess

#### 10.5.4 java.util.Collection

#### 10.5.5 java.util.Set



## <<Clojure编程乐趣>>

10.6 definterface

10.7 慎用异常

10.7.1 一点异常的背景

10.7.2 运行时异常vs.编译时异常

10.7.3 处理异常

10.7.4 定制异常

10.8 小结

第11章 变化

11.1 软件事务性内存，包括多版本并发控制和快照隔离

11.1.1 事务

11.1.2 嵌入式事务

11.1.3 STM使其简单的事情

11.1.4 潜在缺陷

11.1.5 让STM不高兴的事

11.2 使用Ref的时机

11.2.1 使用alter进行可协调的、同步的改变

11.2.2 以commute进行可交换的改变

11.2.3 以ref—set进行普通改变

11.2.4 用ensure修正写入偏差

11.2.5 压力之下的Ref

11.3 使用Agent的时机

11.3.1 进程内并发模型vs分布式并发模型

11.3.2 用Agent控制I / O

11.3.3 send和send—off之间的差异

11.3.4 错误处理

11.3.5 何时不用Agent

11.4 使用Atom的时机

11.4.1 跨线程共享

11.4.2 在事务里使用Atom

11.5 使用lock的时机

11.5.1 使用锁进行安全变化

11.5.2 使用Java的显式锁

11.6 使用future的时机

11.7 使用promise的时机

11.7.1 以promise进行并行任务

11.7.2 回调API到阻塞API

11.7.3 确定性死锁

11.8 并行

11.8.1 pvalues

11.8.2 pmap

11.8.3 pcalls

11.9 var和动态绑定

11.9.1 binding宏

11.9.2 创建命名var

11.9.3 创建匿名var

11.9.4 动态作用域

11.10 小结

## <<Clojure编程乐趣>>

.....

### 第5部分 杂项考量

## <<Clojure编程乐趣>>

### 版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>