

<<代码的未来>>

图书基本信息

书名：<<代码的未来>>

13位ISBN编号：9787115317513

10位ISBN编号：7115317518

出版时间：2013-6

出版时间：人民邮电出版社

作者：[日] 松本行弘

译者：周自恒

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>

<<代码的未来>>

内容概要

《代码的未来》是Ruby之父松本行弘的又一力作。作者对云计算、大数据时代下的各种编程语言以及相关技术进行了剖析，并对编程语言的未来发展趋势做出预测，内容涉及Go、VoltDB、node.js、CoffeeScript、Dart、MongoDB、摩尔定律、编程语言、多核、NoSQL等当今备受关注的话题。

《代码的未来》面向各层次程序设计人员和编程爱好者，也可供相关技术人员参考。

<<代码的未来>>

作者简介

松本行弘 (Yukihiro Matsumoto) Ruby语言发明者，亦是亚洲首屈一指的编程语言发明者。现兼任网络应用通信研究所 (NaCl) 研究员、乐天技术研究所研究员、Heroku首席架构师等。昵称“Matz”。

讨厌东京，喜欢温泉。

周自恒 IT、编程爱好者，技术宅，初中时曾在NOI (国家信息学奥赛) 天津赛区获一等奖，大学毕业后曾任IT咨询顾问，精通英语和日语，译著有《30天自制操作系统》、《大数据的冲击》、《Android应用开发入门》。

<<代码的未来>>

书籍目录

| | | | | | | | |
|-----|--------------|-----------------|---------------------------|---------------------------|--------------------|------------------|-------------------------|
| 目 录 | 第一章 编程的时间和空间 | 1.1 编程的本质 | 3 | 编程的本质是思考 | 4 | 创造世界的乐趣 | 4 |
| | | 快速提高的性能改变了社会 | 5 | 以不变应万变 | 8 | 摩尔定律的局限 | 9 |
| | | 社会变化与编程 | 10 | 1.2 未来预测 | 13 | 科学的未来预测 | 14 |
| | | IT未来预测 | 14 | 极限未来预测 | 16 | 从价格看未来 | 16 |
| | | 从性能看未来 | 17 | 从容量看未来 | 18 | 从带宽看未来 | 19 |
| | | 小结 | 20 | 第二章 编程语言的过去、现在和未来 | 2.1 | 编程语言的世界 | 23 |
| | | 被历史埋没的先驱 | 25 | 编程语言的历史 | 26 | 编程语言的进化方向 | 30 |
| | | 未来的编程语言 | 32 | 20年后的编程语言 | 34 | 学生们的想象 | 34 |
| | | 2.2 DSL(特定领域语言) | 36 | 外部DSL | 37 | 内部DSL | 38 |
| | | DSL的优势 | 39 | DSL的定义 | 39 | 适合内部DSL的语言 | 40 |
| | | 外部DSL实例 | 42 | DSL设计的构成要素 | 43 | Sinatra | 46 |
| | | 小结 | 47 | 2.3 元编程 | 48 | Meta, Reflection | 48 |
| | | 类对象 | 51 | 类的操作 | 52 | Lisp | 53 |
| | | 数据和程序 | 54 | Lisp程序 | 56 | 宏 | 56 |
| | | 宏的功与过 | 57 | 元编程的可能性与危险性 | 59 | 小结 | 60 |
| | | 2.4 内存管理 | 61 | 看似无限的内存 | 61 | GC的三种基本方式 | 62 |
| | | 术语定义 | 62 | 标记清除方式 | 63 | 复制收集方式 | 64 |
| | | 引用计数方式 | 65 | 引用计数方式的缺点 | 65 | 进一步改良的应用方式 | 66 |
| | | 分代回收 | 66 | 对来自老生代的引用进行记录 | 67 | 增量回收 | 68 |
| | | 并行回收 | 69 | GC大统一理论 | 69 | 2.5 异常处理 | 71 |
| | | “一定没问题的” | 71 | 用特殊返回值表示错误 | 72 | 容易忽略错误处理 | 72 |
| | | Ruby中的异常处理 | 73 | 产生异常 | 74 | 更高级的异常处理 | 75 |
| | | Ruby中的后处理保证 | 76 | 其他语言中的异常处理 | 77 | Java的检查型异常 | 77 |
| | | Icon的异常和真假值 | 78 | Eiffel的Design by Contract | 80 | 异常与错误值 | 80 |
| | | 小结 | 81 | 2.6 闭包 | 82 | 函数对象 | 82 |
| | | 高阶函数 | 83 | 用函数参数提高通用性 | 84 | 函数指针的局限 | 85 |
| | | 作用域：变量可见范围 | 87 | 生存周期：变量的存在范围 | 88 | 闭包与面向对象 | 89 |
| | | Ruby的函数对象 | 89 | Ruby与JavaScript的区别 | 90 | Lisp-1与Lisp-2 | 91 |
| | | 第三章 编程语言的新潮流 | 3.1 语言的设计 | 97 | 客户端与服务器端 | 97 | 向服务器端华丽转身 |
| | | 98 | 在服务器端获得成功的四大理由 | 99 | 客户端的JavaScript | 100 | 性能显著提升 |
| | | 101 | 服务器端的Ruby | 102 | Ruby on Rails带来的飞跃 | 102 | 服务器端的Go |
| | | 103 | 静态与动态 | 104 | 动态运行模式 | 105 | 何谓类型 |
| | | 105 | 静态类型的优点 | 106 | 动态类型的优点 | 106 | 有鸭子样的就是鸭子 |
| | | 107 | Structural Subtyping | 108 | 小结 | 108 | 3.2 Go |
| | | 109 | New(新的) | 109 | Experimental(实验性的) | 109 | Concurrent(并发的) |
| | | 110 | Garbage-collected(带垃圾回收的) | 110 | Systems(系统) | 111 | Go的创造者们 |
| | | 111 | Hello World | 112 | Go的控制结构 | 113 | 类型声明 |
| | | 116 | 无继承式面向对象 | 118 | 多值与多重赋值 | 120 | 并发编程 |
| | | 122 | 小结 | 124 | 3.3 Dart | 126 | 为什么要推出Dart? |
| | | 126 | Dart的设计目标 | 129 | 代码示例 | 130 | Dart的特征 |
| | | 132 | 基于类的对象系统 | 132 | 非强制性静态类型 | 133 | Dart的未来 |
| | | 134 | 3.4 CoffeeScript | 135 | 最普及的语言 | 135 | 被误解最多的语言 |
| | | 135 | 显著高速化的语言 | 136 | 对JavaScript的不满 | 138 | CoffeeScript |
| | | 138 | 安装方法 | 139 | 声明和作用域 | 139 | 分号和代码块 |
| | | 141 | 省略记法 | 142 | 字符串 | 143 | 数组和循环 |
| | | 143 | 类 | 145 | 小结 | 146 | 3.5 Lua |
| | | 148 | 示例程序 | 149 | 数据类型 | 149 | 函数 |
| | | 150 | 表 | 150 | 元表 | 151 | 方法调用的实现 |
| | | 153 | 基于原型编程 | 155 | 和Ruby的比较(语言篇) | 157 | 嵌入式语言Lua |
| | | 157 | 和Ruby的比较(实现篇) | 158 | 嵌入式Ruby | 159 | 第四章 云计算时代的编程 |
| | | 4.1 可扩展性 | 163 | 信息的尺度感 | 163 | 大量数据的查找 | 164 |
| | | 164 | 二分法查找 | 165 | 散列表 | 167 | 布隆过滤器 |
| | | 169 | 一台计算机的极限 | 170 | DHT(分布式散列表) | 171 | Roma |
| | | 172 | MapReduce | 173 | 小结 | 174 | 4.2 C10K问题 |
| | | 175 | C10K问题所引发的“想当然” | 177 | 使用epoll功能 | 180 | 使用libev框架 |
| | | 181 | 使用EventMachine | 183 | 小结 | 185 | 4.3 HashFold |
| | | 186 | HashFold库的实现(Level 1) | 187 | 运用多核的必要性 | 190 | 目前的Ruby实现所存在的问题 |
| | | 191 | 通过进程来实现HashFold(Level 2) | 191 | 抖动 | 193 | 运用进程池的HashFold(Level 3) |
| | | 194 | 小结 | 197 | 4.4 进程间通信 | 198 | 进程与线程 |
| | | 198 | 同一台计算机上的进程间通信 | 199 | TCP/IP协议 | 201 | 用C语言进行套接字编程 |
| | | 202 | 用Ruby进行套接字编程 | 204 | Ruby的套接字功能 | 205 | 用Ruby实现网络服务器 |
| | | 208 | 小结 | 209 | 4.5 Rack与Unicorn | 210 | Rack中间件 |
| | | 211 | 应用程序服务器的问题 | 212 | Unicorn的架构 | 215 | Unicorn的解决方案 |
| | | 215 | 性能 | 219 | 策略 | 220 | 小结 |
| | | 221 | 第五章 支撑大数据的数据存储技术 | 5.1 键-值存储 | 225 | Hash类 | 225 |
| | | 226 | 数据库的ACID特性 | 226 | CAP原理 | 227 | |

<<代码的未来>>

CAP解决方案——BASE 228 不能舍弃可用性 229 大规模环境下的键-值存储 230 访问键-值存储 230 键-值存储的节点处理 231 存储器 232 写入和读取 233 节点追加 233 故障应对 233 终止处理 235 其他机制 235 性能与应用实例 236 小结 2365.2 NoSQL 237 RDB的极限 237 NoSQL数据库的解决方案 238 形形色色的NoSQL数据库 239 面向文档数据库 240 MongoDB的安装 241 启动数据库服务器 243 MongoDB的数据库结构 244 数据的插入和查询 244 用JavaScript进行查询 245 高级查询 246 数据的更新和删除 249 乐观并发控制 2505.3 用Ruby来操作MongoDB 251 使用Ruby驱动 251 对数据库进行操作 253 数据的插入 253 数据的查询 253 高级查询 254 find方法的选项 256 原子操作 257 ActiveRecord 259 OD Mapper 2605.4 SQL数据库的反击 264 “云”的定义 264 SQL数据库的极限 264 存储引擎Spider 265 SQL数据库之父的反驳 265 SQL数据库VoltDB 268 VoltDB的架构 269 VoltDB中的编程 270 Hello VoltDB! 271 性能测试 273 小结 2755.5 memcached和它的伙伴们 276 用于高速访问的缓存 276 memcached 277 示例程序 278 对memcached的不满 279 memcached替代服务器 280 另一种键-值存储Redis 282 Redis的数据类型 284 Redis的命令与示例 285 小结 289第六章 多核时代的编程6.1 摩尔定律 293 呈几何级数增长 293 摩尔定律的内涵 294 摩尔定律的结果 295 摩尔定律所带来的可能性 296 为了提高性能 297 摩尔定律的极限 302 超越极限 303 不再有免费的午餐 3046.2 UNIX管道 305 管道编程 306 多核时代的管道 308 xargs——另一种运用核心的方式 309 注意瓶颈 311 阿姆达尔定律 311 多核编译 312 ccache 313 distcc 313 编译性能测试 314 小结 3156.3 非阻塞I/O 316 何为非阻塞I/O 316 使用read(2)的方法 317 边沿触发与电平触发 319 使用read(2)+select的方法 319 使用read+O_NONBLOCK标志 321 Ruby的非阻塞I/O 322 使用aio_read的方法 3236.4 node.js 330 减负 330 拖延 331 委派 332 非阻塞编程 333 node.js框架 333 事件驱动编程 334 事件循环的利弊 335 node.js编程 335 node.js网络编程 337 node.js回调风格 339 node.js的优越性 340 EventMachine与Rev 3416.5 ZeroMQ 342 多CPU的必要性 342 阿姆达尔定律 343 多CPU的运用方法 343 进程间通信 345 管道 345 SysV IPC 346 套接字 347 UNIX套接字 349 ZeroMQ 349 ZeroMQ的连接模型 350 ZeroMQ的安装 352 ZeroMQ示例程序 352 小结 354版权声明 356

<<代码的未来>>

编辑推荐

《代码的未来》编辑推荐：20年后、100年后的编程语言会是什么样？

《代码的未来》中Ruby之父松本行弘剖析云计算、大数据时代下的技术：Lisp会是未来的发展趋势吗？

Go和Dart能取代C和JavaScript吗？

关系型数据库已经走到穷途末路了吗？

Go、VoltDB、node.js、CoffeeScript、Dart、MongoDB……云计算、大数据时代下谁主沉浮？

作者在《代码的未来》中一一剖析。

<<代码的未来>>

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>