

<<领域专用语言实战>>

图书基本信息

书名：<<领域专用语言实战>>

13位ISBN编号：9787115331748

10位ISBN编号：711533174X

出版时间：2013-11

出版单位：人民邮电出版社

作者：[美] Debasish Ghosh

译者：郭晓刚

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>

<<领域专用语言实战>>

内容概要

本书是第一本领域专用语言（DSL）的实战类图书，它面向真正的实践者，是一本让奋战在开发第一线的实干家备感实用的专著！

《领域专用语言实战》基于一系列JVM语言（Java、Ruby、Groovy、Scala和Clojure），分析了它们在实现DSL中的优劣势，同时也给出了丰富的示例。

如果你是一位架构师，希望设计出极具表现力的API，让它既能满足领域用户的需要，又能达到高效开发的要求；如果你是一位有技术背景的特定领域用户，期待着改善与开发团队的沟通效果；如果你是一名程序员，正为如何与领域用户对接业务规则而苦恼……那么，请看这本书。

本书分三部分，以金融中介系统为例全面细致介绍从头设计、实现与使用领域专用语言的方方面面。它不仅讲解了DSL的实现手法，而且从最基本的技术入手，逐渐深入到高级技术，如元编程、解析器组合子，以及ANTLR、Xtext等开发框架。

看完这本书，你将全面、深入地理解领域专用语言的定位、规范、术语，同时还可以把它作为具体开发工作的指导，学以致用，提高工作效率和质量。

<<领域专用语言实战>>

作者简介

作者简介：

Debasish Ghosh

Anshinsoft公司 (<http://www.anshinsoft.com>) 首席技术布道师，开发者推荐博客“Ruminations of a Programmer”的作者，ACM协会高级会员。

他拥有跨国IT企业20余年工作经验，擅长为各种客户（无论是中小型公司还是财富500强企业）交付企业级解决方案，对自己将软件设计和编程最佳实践制度化而引以为傲，热爱Java、Ruby、Scala、OO及函数式编程，关注DSL和NoSQL数据库。

电子邮件：dghosh@acm.org。

Twitter账号：[@debasishg](https://twitter.com/debasishg)。

译者简介：

郭晓刚

大学肄业，有过两次创业和创业失败的经验，从嵌入式硬件到企业软件开发皆无所成。

作为爱好的翻译反倒坚持不辍，积攒了五六本独译、合译的作品。

长期在InfoQ中文站从事编辑工作，顺带磨练了技术触觉和翻译、写作的技艺。

现在家照顾本书拖稿期间出生的儿子。

<<领域专用语言实战>>

书籍目录

第一部分 领域专用语言入门	
第1章 初识DSL	2
1.1 问题域与解答域	2
1.1.1 问题域	3
1.1.2 解答域	3
1.2 领域建模：确立共通的语汇	4
1.3 初窥DSL	6
1.3.1 何为DSL	7
1.3.2 流行的几种DSL	8
1.3.3 DSL 的结构	10
1.4 DSL 的执行模型	11
1.5 DSL 的分类	13
1.5.1 内部DSL	13
1.5.2 外部DSL	14
1.5.3 非文本DSL	15
1.6 何时需要DSL	15
1.6.1 优点	16
1.6.2 缺点	16
1.7 DSL 与抽象设计	17
1.8 小结	18
1.9 参考文献	18
第2章 现实中的DSL	19
2.1 打造首个Java DSL	20
2.1.1 确立共通语汇	21
2.1.2 用Java 完成的首个实现	21
2.2 创造更友好的DSL	24
2.2.1 用XML 实现领域的外部化	25
2.2.2 Groovy：更具表现力的实现语言	25
2.2.3 执行Groovy DSL	27
2.3 DSL 实现模式	28
2.3.1 内部DSL 模式：共性与差异性	29
2.3.2 外部DSL 模式：共性与差异性	35
2.4 选择DSL 的实现方式	39
2.5 小结	41
2.6 参考文献	42
第3章 DSL 驱动的应用程序开发	43
3.1 探索DSL 集成	44
3.2 内部DSL 的集成模式	47
3.2.1 通过Java 6 的脚本引擎进行集成	48
3.2.2 通过DSL 包装器集成	52
3.2.3 语言特有的集成功能	59
3.2.4 基于Spring 的集成	61
3.3 外部DSL 集成模式	62
3.4 处理错误和异常	64
3.4.1 给异常命名	64

<<领域专用语言实战>>

3.4.2	处理输入错误	65
3.4.3	处理异常的业务状态	66
3.5	管理性能表现	67
3.6	小结	68
3.7	参考文献	68
第二部分 实现DSL		
第4章	内部DSL实现模式	70
4.1	充实DSL“工具箱”	71
4.2	内嵌式DSL：元编程模式	72
4.2.1	隐式上下文和灵巧API	73
4.2.2	利用动态装饰器的反射式元编程	78
4.2.3	利用builder的反射式元编程	83
4.2.4	经验总结：元编程模式	85
4.3	内嵌式DSL：类型化抽象模式	86
4.3.1	运用高阶函数使抽象泛化	86
4.3.2	运用显式类型约束建模领域逻辑	93
4.3.3	经验总结：类型思维	95
4.4	生成式DSL：通过模板进行运行时代码生成	96
4.4.1	生成式DSL的工作原理	97
4.4.2	利用Ruby元编程实现简洁的DSL设计	97
4.5	生成式DSL：通过宏进行编译时代码生成	100
4.5.1	开展Clojure元编程	100
4.5.2	实现领域模型	102
4.5.3	Clojure宏之美	103
4.6	小结	104
4.7	参考文献	105
第5章	Ruby、Groovy、Clojure语言中的内部DSL设计	106
5.1	动态类型成就简洁的DSL	107
5.1.1	易读	107
5.1.2	鸭子类型	108
5.1.3	元编程——又碰面了	110
5.1.4	为何选择Ruby、Groovy、Clojure	111
5.2	Ruby语言实现的交易处理DSL	112
5.2.1	从API开始	113
5.2.2	来点猴子补丁	115
5.2.3	设立DSL解释器	116
5.2.4	以装饰器的形式添加领域规则	119
5.3	指令处理DSL：精益求精的Groovy实现	123
5.3.1	指令处理DSL的现状	123
5.3.2	控制元编程的作用域	124
5.3.3	收尾工作	127
5.4	思路迥异的Clojure实现	128
5.4.1	建立领域对象	129
5.4.2	通过装饰器充实领域对象	130
5.4.3	通过REPL进行的DSL会话	134
5.5	告诫	135
5.5.1	遵从最低复杂度原则	135

<<领域专用语言实战>>

5.5.2	追求适度的表现力	135	
5.5.3	坚持优秀抽象设计的各项原则	136	
5.5.4	避免语言间的摩擦	136	
5.6	小结	137	
5.7	参考文献	138	
第6章	Scala 语言中的内部DSL设计	139	
6.1	为何选择Scala	140	
6.2	迈向Scala DSL 的第一步	141	
6.2.1	通过Scala DSL 测试Java对象	142	
6.2.2	用Scala DSL 作为对Java 对象的包装	142	
6.2.3	将非关键功能建模为ScalaDSL	142	
6.3	正式启程	142	
6.3.1	语法层面的表现力	143	
6.3.2	建立领域抽象	144	
6.4	制作一种创建交易的DSL	147	
6.4.1	实现细节	148	
6.4.2	DSL 实现模式的变化	152	
6.5	用DSL 建模业务规则	153	
6.5.1	模式匹配如同可扩展的Visitor模式	153	
6.5.2	充实领域模型	155	
6.5.3	用DSL 表达税费计算的规则	157	
6.6	把组件装配起来	160	
6.6.1	用trait 和类型组合出更多的抽象	160	
6.6.2	使领域组件具体化	161	
6.7	组合多种DSL	162	
6.7.1	扩展关系的组合方式	163	
6.7.2	层级关系的组合方式	167	
6.8	DSL 中的Monad 化结构	171	
6.9	小结	175	
6.10	参考文献	176	
第7章	外部DSL 的实现载体	178	
7.1	解剖外部DSL	179	
7.1.1	最简单的实现形式	179	
7.1.2	对领域模型进行抽象	179	
7.2	语法分析器在外部DSL 设计中的作用	182	
7.2.1	语法分析器、语法分析器生成器	183	
7.2.2	语法制导翻译	184	
7.3	语法分析器的分类	190	
7.3.1	简单的自顶向下语法分析器	191	
7.3.2	高级的自顶向下语法分析器	192	
7.3.3	自底向上语法分析器	193	
7.4	工具支持下的DSL 开发——Xtext	194	
7.4.1	文法规则和大纲视图	195	
7.4.2	文法的元模型	197	
7.4.3	为语义模型生成代码	198	
7.5	小结	201	
7.6	参考文献	202	

<<领域专用语言实战>>

第8章	用Scala 语法分析器组合子设计外部DSL	203
8.1	分析器组合子	204
8.1.1	什么是分析器组合子	205
8.1.2	按照分析器组合子的方式设计DSL	206
8.2	Scala 的分析器组合子库	207
8.2.1	分析器组合子库中的基本抽象	208
8.2.2	把分析器连接起来的组合子	209
8.2.3	用Monad 组合DSL 分析器	213
8.2.4	左递归DSL 语法的packrat分析	214
8.3	用分析器组合子设计DSL 的步骤	217
8.3.1	第一步：执行文法	218
8.3.2	第二步：建立DSL 的语义模型	219
8.3.3	第三步：设计Order 抽象	220
8.3.4	第四步：通过函数施用组合子生成AST	221
8.4	一个需要packrat 分析器的DSL实例	223
8.4.1	待解决的领域问题	223
8.4.2	定义文法	225
8.4.3	设计语义模型	227
8.4.4	通过分析器的组合来扩展DSL语义	229
8.5	小结	231
8.6	参考文献	231
第三部分	DSL开发的未来趋势	
第9章	展望DSL 设计的未来	234
9.1	语言层面对DSL 设计的支持越来越充分	235
9.1.1	对表现力的不懈追求	235
9.1.2	元编程的能力越来越强	237
9.1.3	S 表达式取代XML 充当载体	237
9.1.4	分析器组合子越来越流行	238
9.2	DSL 工作台	238
9.2.1	DSL 工作台的原理	239
9.2.2	使用DSL 工作台的好处	240
9.3	其他方面的工具支持	241
9.4	DSL 的成长和演化	242
9.4.1	DSL 的版本化	242
9.4.2	DSL 平稳演化的最佳实践	242
9.5	小结	244
9.6	参考文献	244
附录A	抽象在领域建模中的角色	246
A.1	设计得当的抽象应具备的特质	246
A.1.1	极简	247
A.1.2	精炼	247
A.1.3	扩展性和组合性	247
A.2	极简，只公开对外承诺的	247
A.2.1	用泛化来保留演化余地	248
A.2.2	用子类型化防止实现的泄露	248
A.2.3	正确实施实现继承	249
A.3	精炼，只保留自身需要的	250

<<领域专用语言实战>>

A.3.1	什么是非本质的	250
A.3.2	非本质复杂性	250
A.3.3	撇除杂质	251
A.3.4	用DI隐藏实现细节	252
A.4	扩展性提供成长的空间	253
A.4.1	什么是扩展性	253
A.4.2	mixin：满足扩展性的一种设计模式	254
A.4.3	用mixin扩展Map	255
A.4.4	函数式的扩展性	256
A.4.5	扩展性也可以临时抱佛脚	256
A.5	组合性，源自纯粹	257
A.5.1	用设计模式满足组合性	257
A.5.2	回归语言	259
A.5.3	副作用和组合性	260
A.5.4	组合性与并发	262
A.6	参考文献	262
附录B	元编程与DSL设计	263
B.1	DSL中的元编程	263
B.1.1	DSL实现中的运行时元编程	264
B.1.2	DSL实现中的编译时元编程	265
B.2	作为DSL载体的Lisp	268
B.2.1	Lisp的特殊之处	268
B.2.2	代码等同于数据	269
B.2.3	数据等同于代码	269
B.2.4	简单到只分析列表结构的语法分析器	270
B.3	参考文献	271
附录C	Ruby语言的DSL相关特性	272
C.1	Ruby语言的DSL相关特性	272
C.2	参考文献	275
附录D	Scala语言的DSL相关特性	276
D.1	Scala语言的DSL相关特性	276
D.2	参考文献	279
附录E	Groovy语言的DSL相关特性	280
E.1	Groovy语言的DSL相关特性	280
E.2	参考文献	282
附录F	Clojure语言的DSL相关特性	283
F.1	Clojure语言的DSL相关特性	283
F.2	参考文献	285
附录G	多语言开发	286
G.1	对IDE的特性要求	287
G.2	搭建Java和Groovy的混合开发环境	287
G.3	搭建Java和Scala的混合开发环境	288
G.4	常见的多语言开发IDE	288
索引		290

<<领域专用语言实战>>

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>