加微信：642945106 发送“ 赠送 ”领取赠送精品课程
发数字“ 2 ”获取众筹列表

11 | DSL：你也可以设计一门自己的语言

2020-06-17 郑晔

软件设计之美

进入课程 >

讲述：郑晔

时长 11:47　大小 10.81M

你好！我是郑晔。

在前面，我们花了三讲的篇幅探讨程序设计语言，一方面是为了增进我们对程序设计语言的理解，另一方面，也希望从中学习到软件设计方面做得好的地方。除了借鉴一些语言特性之外，我们还能怎样应用程序语言，来帮我们做设计呢？

讲到程序设计语言模型时，我说过，程序设计语言的发展趋势，就是离计算机本身越来越远，而离要解决的问题越来越近。但通用程序设计语言无论怎样逼近要解决的问题，它都不可能走得离问题特别近，因为通用程序设计语言不可能知道具体的问题是什么。

这给具体的问题留下了一个空间，**如果我们能把设计做到极致，它就能成为一门语言**，填补这个空间。注意，我这里用的并不是比喻，而是真的成为一门语言，一门解决一个特定问题的语言。

这种语言就是领域特定语言（Domain Specific Language，简称 DSL），它是一种用于某个特定领域的程序设计语言。这种特定于某个领域是相对于通用语言而言的，通用语言可以横跨各个领域，我们熟悉的大多数程序设计语言都是通用语言。

我在第 8 讲说过，它们都是图灵完备的，但 DSL 不必做到图灵完备，它只要做到满足特定领域的业务需求，就足以缩短问题和解决方案之间的距离，降低理解的门槛。

虽然大多数程序员并不会真正地实现一个通用程序设计语言，但实现一个 DSL，我们还是有机会的。这一讲我们就来谈谈 DSL，看看我们可以怎样设计自己的语言。

领域特定语言

不过，一说起设计一门语言，很多人直觉上会有畏惧心理。但实际上，你可能已经在各种场合接触过一些不同的 DSL 了。程序员最熟悉的一种 DSL 就是正则表达式了，没错，也许已经习惯使用正则表达式的你都不知道，但它确实就是一种 DSL，一种用于文本处理这个特定领域的 DSL。

如果你觉得正则表达式有点复杂，还有一种更简单的 DSL，就是配置文件。你可能真的不把配置文件当作一种 DSL，但它确实是在实现某个特定领域的需求，而且可以根据你的需求对软件的行为进行定制。

一个典型的例子是 Ngnix。无论你是用它单独做 Web 服务器也好，做反向代理也罢，抑或是做负载均衡，只要通过 Ngnix 的配置文件，你都能实现。配合 OpenResty，你甚至可以完成一些业务功能。

这么一说，你是不是觉得 DSL 的门槛不像听上去那么高了。

经过前面几讲的学习，你应该知道了，语法只是一种接口。很多人说到设计 DSL，脑子里实际想的也只是设计一种语法。所以，从软件设计的角度看，DSL 最终呈现出来的语法只是一种接口，但最重要的是它包裹的模型。

Martin Fowler 在他的《领域特定语言》这本书中，将这个模型称为语义模型（Semantic Model）。不过，在我看来，Martin Fowler 起这个名字是站在语言开发的角度，毕竟语义这个词，只有学过编译原理的人才好理解。所以，这里真正的重点是模型。

想要实现一个 DSL，可以这么说，DSL 的语法本身都是次要的，模型才是第一位的。当你有了模型之后，所谓的构建 DSL，就相当于设计一个接口，将模型的能力暴露出来。

当把 DSL 理解成接口，我们接受 DSL 的心理负担就小了很多。你可以想一想，它和你熟悉的 REST API 其实没有什么本质的不同。

既然是接口，形式就可以有很多种，我们经常能接触到的 DSL 主要有两种：外部 DSL 和内部 DSL。Martin Fowler 在他的书中还提到了语言工作台（Language Workbench），不过，这种做法在实际工作中用到的不多，我们暂且忽略。

外部 DSL 和内部 DSL 的区别就在于，DSL 采用的是不是宿主语言（Host Language）。你可以这么理解，假设你的模型主要是用 Java 写的，如果 DSL 用的就是 Java 语言，它就是内部 DSL，如果 DSL 用的不是 Java，比如，你自己设计了一种语法，那它就是外部 DSL。

防止断更 请务必加
首发微信：17161436665

把概念说清楚了，一些问题便迎刃而解了。这也可以解释为什么 DSL 让有些人畏惧了，原因就是说起 DSL，这些人想到的就是自己设计语法的外部的 DSL。其实，即便是外部 DSL，也不一定要设计一门语法，我们甚至可以借助已有的语法来完成。比如，很多程序员熟悉的一种语法：XML。

如果你是一个 Java 程序员，XML 就再熟悉不过了。从 Ant 到 Maven，从 Servlet 到 Spring，曾经的 XML 几乎是无处不在的。如果你有兴趣，可以去找一些使用 Ant 做构建工具的项目，项目规模稍微大一点，其 XML 配置文件的复杂程度就不亚于普通的源代码。

因为它本质上就是一种用于构建领域的 DSL，只不过，它的语法是 XML 而已。正是因为这种 DSL 越来越复杂，后来，一种新的趋势渐渐兴起，就是用全功能语言（也就是真正的程序设计语言）做 DSL，这是后来像 Gradle 这种构建工具逐渐流行的原因，它们只是用内部 DSL 替换了外部 DSL。

从复杂度而言，自己设计一种外部 DSL 语法，大于利用一种现有语法做外部 DSL，二者之间的差别在于谁来开发解析器。而外部 DSL 的复杂度要大于内部 DSL，因为内部 DSL 连解析的过程都省略了。从实用性的角度，更好地挖掘内部 DSL 的潜力对我们的实际工作助益更多。

代码的表达性

你或许会有一个疑问，内部 DSL 听上去就是一个程序库啊！你这个理解是没错的。我们前面说过，语言设计就是程序库设计，程序库设计就是语言设计。当一个程序库只能用在某个特定领域时，它就是一个内部 DSL，这个内部 DSL 的语法就是这个程序库的用法。

我先用一个例子让你感受一下内部 DSL，它来自 Martin Fowler 的《领域特定语言》。我们要创建一个 Computer 的实例，如果用普通风格的代码写出来，应该是这个样子：

```
Processor p = new Processor(2, 2500, Processor.Type.i386); Disk d1 = new Disk(
Disk d2 = new Disk(75, 7200, Disk.Interface.SATA);
return new Computer(p, d1, d2);
```

而用内部 DSL 写出来，则是这种风格：

```
computer()
  .processor()
    .cores(2)
    .speed(2500)
    .i386()
  .disk()
    .size(150)
  .disk()
   .size(75)
   .speed(7200)
   .sata()
.end();
```

如果这是一段普通的 Java 代码，我们看到一连串的方法调用，一定会说，这段代码糟糕至极！但在这个场景下，和前面的代码相比，这段代码省去了好多变量，反而是清晰了。这其中的差别在哪里呢？

之所以我们会觉得这种一连串的方法调用可以接受，一个重要的原因是，这段代码并不是在做动作，而是在进行声明。做动作是在说明怎么做（How），而声明的代码则是在说做什么（What）。

二者的抽象级别是不同的，“怎么做”是一种实现，而“做什么”则体现着意图。**将意图与实现分离开来**，是内部 DSL 与普通的程序代码一个重要的区别，同样，这也是一个好设计的考虑因素。

Martin Fowler 在讨论 DSL 定义时，提到了 DSL 的 4 个关键元素：

计算机程序设计语言（Computer programming language）；

语言性（Language nature）；

受限的表达性（Limited expressiveness）；

针对领域（Domain focus）。

其中，语言性强调的就是 DSL 要有连贯的表达能力。也就是说，你设计自己的 DSL 时，重点是要体现出意图。抛开是否要实现一个 DSL 不说，的确，**程序员在写代码时应该关注代码的表达能力**，而这也恰恰是很多程序员忽略的，同时也是优秀程序员与普通程序员拉开差距的地方。

普通程序员的关注点只在于功能如何实现，而优秀的程序员会懂得将不同层次的代码分离开来，将意图和实现分离开来，而实现可以替换。

说到这里，你就不难理解学习内部 DSL 的价值了，退一步说，你不一定真的要自己设计一个内部 DSL，但学会将意图与实现分离开，这件事对日常写代码也是有极大价值的。

有了这个意识，你就可以很好地理解程序设计语言的一个重要发展趋势：声明式编程。现在一些程序设计语言的语法就是为了方便进行声明式编程，典型的例子就是 Java 的 Annotation。正是它的出现，Spring 原来基于 XML 的外部 DSL 就逐步转向了今天常用的内部 DSL 了，也就是很多人熟悉的 Java Config。

你会发现，虽然我在这说的是写代码，但分离意图和实现其实也是一个重要的设计原则，是的，**想写好代码，一定要懂得设计**。

总结时刻

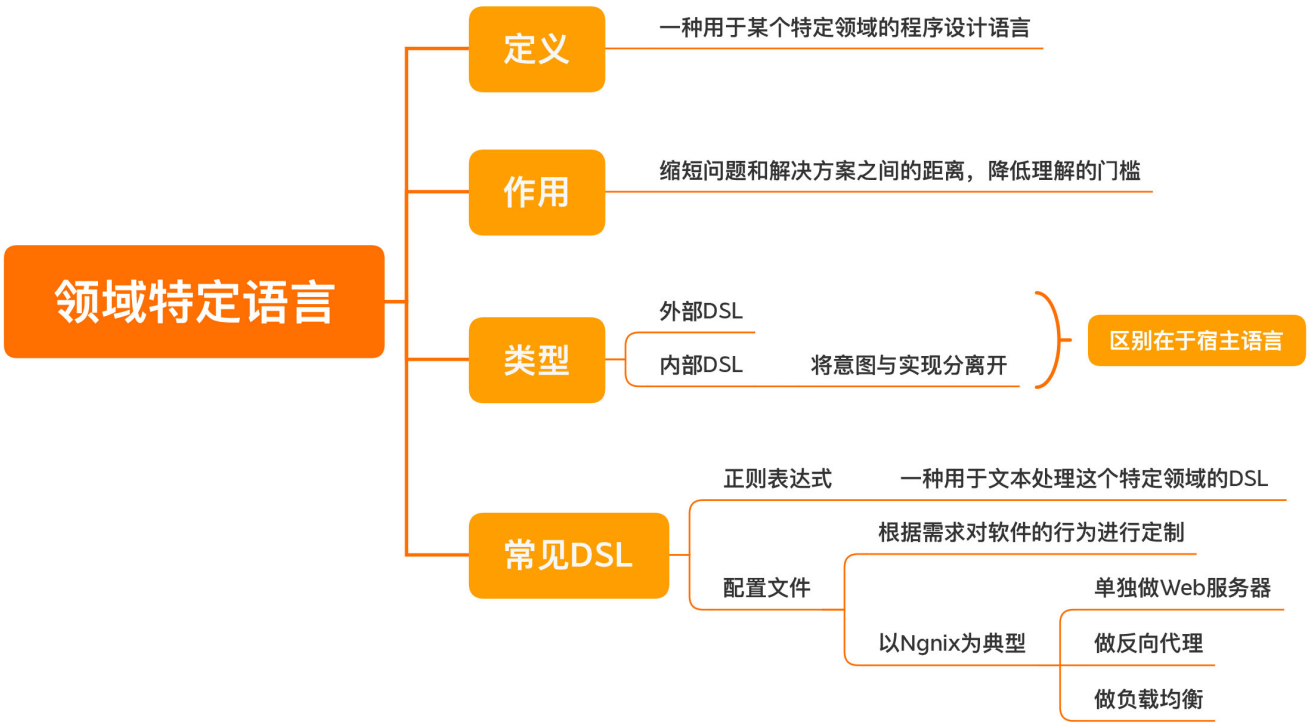
今天，我们讨论了领域特定语言，这是针对某个特定领域的程序设计语言。DSL 在软件开发领域中得到了广泛的应用。要实现一个 DSL，首先要构建好模型。

常见的 DSL 主要是外部 DSL 和内部 DSL。二者的主要区别在于，DSL 采用的是不是宿主语言。相对于外部 DSL，内部 DSL 的开发成本更低，与我们的日常工作结合得更加紧密。

内部 DSL 体现更多的是表达能力，相对于传统的代码编写方法而言，这种做法很好地将作者的意图体现了出来。即便我们不去设计一个内部 DSL，这种写代码的方式也会对我们代码质量的提高大有助益。

关于语言，已经讲了四讲，我们先告一段落。下一讲，我们要来讨论编程范式，也就是做设计的时候，我们可以利用的元素有哪些。

如果今天的内容你只能记住一件事，那请记住：**好的设计要迈向 DSL，我们可以从编写有表达性的代码起步**。



极客时间

思考题

最后，我想请你分享一下，你还能举出哪些 DSL 的例子呢？欢迎在留言区分享你的想法。

感谢阅读，如果你觉得这一讲的内容对你有帮助的话，也欢迎把它分享给你的朋友。

更多课程推荐

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇　10 | 语言的实现：运行时，软件设计的地基

下一篇　加餐 | 再八卦几门语言！

精选留言 (12)

写留言

蓝士钦
2020-06-17

SQL也是一种DSL，他屏蔽了计算机存储的底层实现，提供了易于操作数据的接口。
一些ORM框架对SQL这些DSL进行了进一步的封装提供了声明式注解，相当于构建在DSL之上的DSL翻译器。面向对象编程将面向关系的DSL进行更高层次的封装，使得在编程这个特定领域更加易于使用。

展开

作者回复: 很好的分享！

8

泡泡龙
2020-06-17

我觉得markdown应该算一个

展开

作者回复: 嗯，好有趣的角度！

5

Jxin
2020-06-17

1.k8s和docker-compose的yml文件，就是声明试编程。算是外部dsl。

2.本章疑问： dsl和接口有何异同点?

首先dsl和接口都做了一件事，就是意图和实现的分离。但是dsl的语义（意图）是可以灵...

展开

作者回复: 后面我们就来谈结构化编程和面向对象编程。

5

被雨水过滤的空气
2020-06-18

高级编程语言是低级编程语言的DSL。

展开

作者回复: 哈哈，确实可以这么理解。

3

happychap
2020-06-17

drone.io用的.drone.yml，jenkins用的pipefile都算是轻量级的dsl。uml算不算是一种重量级的dsl呢?

作者回复: UML可能不算，因为它不能执行。

3

阳仔
2020-06-17

DSL是为了解决某个特定领域的程序设计语言。
作为一个客户端APP开发者，最常用到的莫过于gradle。
现在JAVA后端程序主要是通过pom配置构建，它其实就是通过xml来实现DSL，
我觉得后端程序通过gradle构建也将会成为主流。它比xml更加灵活，表达性更强
要设计一个DSL就要构建一个模型，通过接口将能力暴露出来。...
展开

作者回复: 非常好的总结！

3

escray
2020-06-18

我还没有到“设计一个 DSL”的高度，而且可能日常工作中遇到的问题也没有需要一门新的 DSL 来解决。

正则表达式、配置文件和 SQL 都可以算作 DSL，这些都是受众比较广泛的，如果是自己设计一个，使用的人没有那么多，还有意义么？...
展开

作者回复: Markdown 并不能执行。

2

再来二两杜康酒
2020-06-17

lambda，网络协议描述算不算是dsl呢?
展开

作者回复: DSL 首先是一门语言，能够执行的那种。单独的 Lambda 和协议都是不可执行的。

2

PM2
2020-06-22

1、redis的指令应该不算是dsl，而只是接口吧
2、linux的awk应该算是一种dsl。

作者回复: AWK已经是一门正经的语言了。

1

春去春又来
2020-06-19

全文读完就感觉 DSL 其实就是个间接层，为什么要有这个间接层，为了就是能更简单的更快捷的解决问题。

作者回复: 是的，接口也很重要。

1

NIU
2020-06-17

如果这么说，移动端开发常用的Cocoapods也是一种DSL。

作者回复: 嗯，也算。

1

董松涛
2020-06-23

ansible-playbooks的yml文件应该也能看成dsl

展开