参考：<http://blog.csdn.net/chgaowei/article/details/21296483>

**第一部分：叙述**

DSL（Domain Specified Language）编程：有人将DSL编程称之为声明式编程。

DSL是在模型之上建立的一种更加灵活的对 模型化的理解和使用方式。

语义模型是DSL的核心。

内部DSL：用通用语言的语法表示DSL，需要安装某种风格使用这种语言。

外部DSL：在主程序设计语言之外，用一种单独的语言表示领域专有语言。可以是定制语法，或者遵循另外一种语法，如XML。

DSL定义：针对某一**领域**，具有**受限表达性**的一种计算机程序设计**语言**。

外部DSL：不同于应用系统主要使用语言（c，java，c++，c#）的语言。

内部DSL：通用语言的特定用法。内部DSL通常是一段合法的程序，但是具有特定的风格。而且只用到了语言一部分特性。Ruby形成了显著的DSL文化。后面好好看看。

语言工作平台：专用IDE，用于构建和定义DSL。

DSL伴随语义模型出现，语义模型会表现为程序库或者框架，对于构建DSL而言，语义模型不可或缺。DSL只是位于其上的一层而已。

命令查询时api定义了抽象领域的词汇，而内部DSL则在此基础上添加了语法。

内部DSL给人的感觉是一个整句，而并非一个无关命令。

语言性：领域词汇的连贯性。

片段DSL（SQL，正则表达式），独立DSL。

DSL的受限表达性可以使DSL语言不易出错，即便出错，也易于发现。这是受限表达性的意义。

声明式编程：定义做什么，而不是用一堆命令语句来描述怎么做。这一点很重要。DSL就是用声明式编程。

防止DSL逐渐演变为一种通用语言，要受限表达。目的防止DSL过于复杂，可维护性降低，学习成本提升，偏离方向。

不要让DSL读起来像自然语言。它是程序语言，比自然语言更加准确和简洁。

语义模型位于语言和DSL之间，为二者解耦。

DSL脚本，解析器，语义模型，模型——DSL自上而下几个层次。

内部DSL和外部DSL区别：是否使用通用语言写。

DSL的执行有两种方式，执行语义模型与代码生成。

可以考虑从语义模型生成代码。

DSL---->解析器----->语义模型----->执行

                 |                  |

                 |                  |

                 |                  V

                 |             代码生成

                 V

           代码生成