

图书基本信息

## 内容概要

本书是微软DSL工具的软件开发指南。

本书主要讲解特定领域开发（DSD）、创建和使用DSL、域模型定义、界面表示、创建/删除/更新行为、序列化、约束与验证、生成工件、部署DSL、DSL高级定制功能以及设计DSL。

本书不仅介绍特定领域开发和模型驱动开发等概念，还注重理论与实例相结合，用一个具体DSL应用实例贯穿全书，来详细介绍如何使用DSL工具开发用户自己的特定领域语言，从而实现真正的模型驱动软件开发。

本书适合使用DSL工具的软件开发人员和 DSL 感兴趣的读者参考。

## 作者简介

Steve Cook微软公司企业框架和工具组的一名软件架构师。  
在加入微软之前，他是IBM的一名杰出工程师，在推出UML 2.0规范的过程中，他是IBM公司的代表。  
作为英国计算机协会的一名院士，他在早期参与了面向对象编程的推广。

## 书籍目录

译者序前言作者简介第1章 特定领域的开发 1.1 简介 1.2 特定领域开发 1.3 举例 1.3.1 软件定义电路 1.3.2 嵌入式系统 1.3.3 设备界面 1.3.4 软件开发过程定制 1.4 优点 1.5 语言 1.6 文本DSL 1.7 图形DSL 1.7.1 表示结构的规范 1.7.2 表示行为的规范 1.8 图形DSL的相关内容 1.8.1 符号 1.8.2 域模型 1.8.3 生成 1.8.4 序列化 1.8.5 工具集成 1.8.6 综合应用 1.9 Visual Studio中的DSL 1.10 定制化陷阱 1.11 UML 1.12 小结第2章 创建和使用DSL 2.1 简介 2.2 过程：DSL的递增开发 2.2.1 通用化应用程序：定位变化部分，发现DSL 2.2.2 自顶向下和自底向上 2.2.3 开发DSL：从草图到域模型 2.2.4 域模型和表示是分离的 2.2.5 改善DSL 2.2.6 由DSL驱动框架 2.2.7 使用DSL 2.2.8 DSL的发展 2.2.9 解析式框架 2.3 在Visual Studio中创建DSL 2.3.1 在Visual Studio中创建一个DSL authoring解决方案 2.3.2 尝试使用DSL解决方案 2.3.3 定义：DSL 2.3.4 生成设计器代码 2.3.5 向DSL中添加内容 2.3.6 约束 2.3.7 定制DSL资源管理器窗口 2.3.8 定制属性窗口 2.3.9 设计器自定义代码 2.3.10 DSL文件的序列化格式 2.3.11 由DSL驱动的应用程序 2.3.12 部署 2.4 第二个DSL例子：工程定义DSL 2.5 DSL工具的架构 2.5.1 生成的代码 2.5.2 DSL工具的架构分层 2.5.3 框架程序集 2.5.4 DSL项目的内容 2.5.5 DslPackage项目的内容 2.6 小结第3章 域模型定义 3.1 简介 3.2 域模型设计器 3.3 驻留内存中的Store 3.4 域类 3.5 域关系 3.5.1 嵌入关系 3.5.2 重数 3.5.3 引用关系 3.5.4 关系的派生 3.6 生成不包含任何形状的设计器“ 3.7 生成的代码 3.8 更多关于域类的讨论 3.9 更多关于域属性的讨论 3.9.1 自动计算的属性 3.9.2 DomainPropertyInfo 3.10 更多关于域关系和角色的讨论 3.10.1 访问链接 3.10.2 更多关于关系派生的讨论 3.10.3 DomainRelationshipInfo和DomainRoleInfo 3.11 更多关于Store的话题 3.11.1 查找元素 3.11.2 分区 (Partitions) 3.11.3 规则 3.11.4 DomainModelInfo 3.12 小结第4章 界面表示 4.1 简介 4.2 图形符号概述 4.3 图和编辑器 4.3.1 图表 4.3.2 编辑器 4.3.3 设计器 4.3.4 自定义编辑器 4.4 形状 4.4.1 形状分类 4.4.2 形状映射 4.5 连接器 4.5.1 连接器解析和外观 4.5.2 连接器和继承 4.5.3 连接器映射 4.5.4 高级连接器映射 4.6 装饰器 4.6.1 装饰器的类型 4.6.2 定位 4.6.3 装饰器映射 4.7 在代码中自定义图形符号 4.7.1 多行文本装饰器 4.7.2 图像形状变量 4.7.3 设置背景图片 4.7.4 设置自定义连接点 4.7.5 更改连接器的路线样式 4.8 浏览器 4.8.1 默认外观 4.8.2 更改窗口图标和标签 4.8.3 自定义的节点外观 4.8.4 隐藏节点 4.8.5 通过代码自定义浏览器 4.9 属性窗口 4.9.1 默认的属性窗口外观 4.9.2 类别、名称和说明 4.9.3 隐藏属性和使属性只读 4.9.4 属性的传递 4.9.5 通过代码自定义属性窗口 4.10 小结第5章 创建、删除和更新行为 5.1 简介 5.2 元素的创建 5.2.1 工具箱 5.2.2 元素合并指令 5.2.3 自定义元素合并指令 5.2.4 Re-Parenting与元素合并指令 5.2.5 自定义元素工具原型 5.3 连接构造器 5.3.1 多个源角色和目标角色的指令 5.3.2 多链接的连接指令 5.3.3 自定义连接构造器 5.4 元素的删除 5.4.1 默认的删除传播规则 5.4.2 控制删除传播 5.4.3 自定义删除传播 5.5 小结第6章 序列化 6.1 简介 6.2 保存和加载模型和图表 6.3 模型的XML文件格式 6.4 元素和属性 6.5 关系 6.6 交叉引用 6.6.1 使用Guid作为引用 6.6.2 使用qualified name作为引用 6.6.3 对链接引用 6.7 图表的XML文件格式 6.8 版本控制和迁移 6.9 XML模式文件 6.10 自定义序列化 6.10.1 修改XML元素名称 6.10.2 ElementData 6.10.3 实现你自己的序列化器 6.11 生成的序列化代码 6.11.1 自定义的序列化代码 6.11.2 自定义对模式文件的影响 6.12 小结第7章 约束与验证 7.1 简介 7.2 选择硬约束还是软约束 7.3 DSL工具中的软约束 7.3.1 验证方法 7.3.2 启用验证 7.3.3 触发验证 7.3.4 定制验证类别 7.3.5 验证行为的继承 7.3.6 验证输出 7.3.7 在Visual Studio IDE之外使用验证 7.3.8 针对外部数据的验证 7.4 DSL工具中的硬约束 7.5 规则 7.6 硬、软约束相结合 7.7 小结第8章 生成工件 8.1 简介 8.2 工件生成方式 8.2.1 扩展样式表转换语言 (XSLT) 8.2.2 使用特定领域API 8.2.3 一种基于模板的方法 8.3 复杂关系和同步 8.4 模板化过程 8.4.1 第一个模板 8.4.2 与生成代码相关的模型数据 8.4.3 开始创建模板库 8.5 文本模板的语法 8.5.1 指令 8.5.2 自定义指令 8.5.3 控制块的类型 8.6 实际应用中的大规模工件生成问题 8.7 高级自定义功能 8.7.1 文本模板的架构 8.7.2 自定义宿主 8.7.3 自定义指令处理器 8.7.4 自定义业务流程 8.8 小结第9章 部署DSL 9.1 简介 9.2 安装一个设计器所需要的文件 9.3 创建一个安装项目 9.4 安装项目内容 9.5 自定义安装程序 9.5.1 自定义InstallerDefinition.dslsetup 9.5.2 自定义settings.ini 9.5.3 自定义Strings.wxl 9.5.4 自定义Product.ico 9.6 dslsemp文件的格式 9.6.1 9.6.2 9.6.3 9.6.4 9.6.5 9.6.6 9.6.7 9.6.8 9.7 更新安装文件 9.8 包

加载键 9.9 为生成代码部署文本模板 9.9.1 在Debugging项目中创建项目模板 9.9.2 使用文本模板包含文件 9.9.3 在VS项模板中包含文本模板 9.10 小结第10章 DSL高级定制功能 10.1 简介 10.2 定制工具 10.2.1 部分类 10.2.2 双重派生——The Generation Gap 10.2.3 自定义构造函数 10.2.4 自定义开关 10.2.5 自定义重载 10.3 对更改的响应 10.3.1 属性值变化处理“On Value Changed/Changing” 10.3.2 计算域属性 10.3.3 自定义存储域属性 10.3.4 值变化通知 10.3.5 把模型更改传递给形状：On AssociatedPropertyChanged 10.3.6 规则 10.3.7 Store事件 10.3.8 .NET事件处理程序 10.3.9 事件重载 10.3.10 边界规则 10.3.11 更改传递技术和约束技术的小结 10.4 DSL外壳程序体系架构 10.5 如何增加菜单命令 10.5.1 为每个命令增加一个命令标识 10.5.2 增量菜单资源索引 10.5.3 添加命令到命令集 10.5.4 定义命令处理程序 10.5.5 命令处理程序的较好实现 10.5.6 编译运行 10.5.7 为标准的命令提供处理器 10.6 在另一个界面中构建DSL图 10.7 实现复制粘贴 10.7.1 复制方法 10.7.2 粘贴方法 10.7.3 注册菜单处理程序 10.8 形状容器 10.8.1 子形状 10.8.2 使用内嵌子形状的DSL 10.8.3 使用规则实现形状包含 10.9 小结第11章 设计DSL 11.1 简介 11.2 识别可变性 11.2.1 自底向上还是自顶向下 11.2.2 特征树 11.2.3 特征树与DSL 11.3 开发域模型 11.3.1 拟订域快照 11.3.2 从快照中获得的域模型 11.4 开发标记法 11.4.1 项目定义标记法 11.4.2 问题状态标记法 11.4.3 常见的标记法 11.5 定义验证约束 11.5.1 内部一致性 11.5.2 外部数据及模型的一致性 11.6 开发和演化框架 11.6.1 比较生成型和解释型 11.6.2 演化一个通用框架 11.6.3 从DSL驱动生成框架 11.7 测试 11.7.1 验证约束 11.7.2 生成器模板 11.7.3 生成的代码 11.7.4 规则 11.7.5 语言定义 11.8 改进DSL 11.9 什么是一个好的DSL 11.9.1 适当的标记法：正则表达式的一个例子 11.9.2 候选的标记法 11.9.3 图不是语法树 11.10 小结 11.11 总结

## 章节摘录

第3章 域模型定义3.1 简介第2章介绍了开发一个DSL需要创建的不同组成部分：域模型，图形符号和工具箱，资源管理器和属性窗口，验证，序列化和部署。

同时第2章还介绍了DSL设计器，DSL设计者可以运用这一工具来定义新语言的不同组件。

本章将描述怎样定义域模型，并从域模型生成的DSL工具的角度来解释域模型的含义。

每一个DSL的核心都是一个域模型。

它定义了这一语言所代表的各种概念，这些概念的属性，以及它们之间的关系。

所有的DSL用户都必须对这些有一定程度的了解，因为用户在使用DSL时创建和操作的每一个元素都是用域模型来描述的。

域模型就像DSL的语法；它定义了组成模型的要素并给出了将这些要素互相联系在一起的规则。

域模型还为语言的其他方面的建立提供了基础。

图形符号、工具箱、资源管理器、属性窗口、验证、序列化和部署的定义都建立在域模型上。

域模型还用来生成可用于编程的API，用户可以用这些API来定制和扩展语言，或在模板中用这些API来生成代码或其他文本工件。

如果你熟悉面向对象设计或者面向对象编程的话，那么域建模（Domain Modeling）的基本概念对你来说就显得非常简单了。

本章将用第2章中介绍的Issue State示例来介绍域建模中所有的基本思想。

同时，为了更加细致地探讨域建模中的部分细节，通过对Issue State示例进行改进来介绍域建模中的一些关键问题。

## 编辑推荐

《Visual Studio DSL工具特定领域开发指南》深入地探讨了以下内容：判断DSL是否适合您。

对DSL和其他模型驱动的开发方法进行对比。

DSL的定义、调优和演化：模型、界面表示、创建、更新、序列化、限制条件、验证以及其他内容。

在编写很少代码、甚至不编写代码的情况下，为新定义的DSL生成可视化设计器。

利用简单的文本模板，为您的模型生成应用程序代码，极大地提高生产效率。

自动生成配置文件、资源和其他工件。

快速简单地在不同组织中部署可视化设计器。

为特定的过程需求定制可视化设计器。

作为一种为某些特定(横向或纵向)领域而定制的语言，特定领域语言(DSL)正在软件工程师和架构师之间孕育一个不断增长的兴奋因子。

DSL给软件的创建和演化带来了新的敏捷性。

与标准的程序代码相比，DSL使得设计考虑的不同方面都能够以更加贴近系统需求的形式表示出来，从而极大地减少大规模项目和生产线的开发成本。

在这本具有突破性的著作中，4名杰出的专家向读者介绍了DSL的使用方式，以及在您的环境中怎样最好地利用它们。

通过阅读《Visual Studio DSL工具特定领域开发指南》，首先您可以掌握可应用于所有平台的DSL概念和技术。

然后，您将学会怎样通过微软公司新推出的功能强大的DSL工具(这个工具集正是由《Visual Studio DSL工具特定领域开发指南》作者所设计的)来创建和使用DSL，以及怎样定义DSL并用Visual Studio内建的建模技术来生成可视化设计器。

#### 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>