Eclipse GEF框架中命令模式的应用研究

刘绪斌[1],王 良[1],闫有朋[2]
(1.山东大学 信息工程学院,山东 威海 264209;
2.积成电子股份有限公司,山东 济南 250100)

摘 要:为了加快复杂面向对象图形编辑系统的开发速度,解决传统开发方法效率低、可重用性不强和部分功能难以实现的缺点,介绍开源项目Eclipse中的图形化编辑框架(GEF),并对该框架中的命令(Command)设计模式进行详细的研究。由于该框架是基于各种设计模式组合而成,可重用性非常好,并且是典型的MVC框架,因此,通过研究GEF中的命令模式,可以很容易地实现具有撤消/重做等功能的图形化编辑器系统。最后通过快速设计基于GEF框架的编辑器软件,加深了对基于MVC+Command模式系统开发的了解。

关键词:GEF;Command模式;MVC;设计模式
中图分类号:TP31 **文献标识码**:A **文章编号**:1673-629X(2010)08-0100-04

Application Studies on Command Pattern of Eclipse GEF

LIU Xu-bin[1],WANG Liang[1],YAN You-peng[2]
(1.School of Information Engineering,Shandong University,Weihai 264209,China;
2.Integrated Electronic Systems Lab Co.Ltd,Jinan 250100,China)

Abstract:In order to speed up the development of object-oriented graphic-editor system and solve the questions of low efficiency, lacking reusability and realizing difficult functions, a graphical editor framework which belongs to Eclipse, an open source project, is introduced. And the command pattern of this framework will be studied in detail. This framework is made up of many design patterns. And it is a typical and well reusable MVC framework. So it becomes very easy to finish an editor system with undo/redo functions. In the end, system developing based on MVC+Command pattern is well understood after quickly accomplishing a GEF editor software.

Key words:GEF;Command pattern;MVC;design pattern

0 引 言

GEF(Graphical Editor Framework)[1]是一个完善的图形编辑框架,是Eclipse的一个开源工具项目,它提供了图形编辑的解决方案。用户可以通过它迅速构建出功能强大的图形编辑器。它允许开发人员可以用图形化的方式编辑和展示模型。GEF已经广泛应用于UML类图编辑器、XML编辑器以及数据库结构设计工具等图形化系统当中。

GEF提供了标准的MVC(Model-View-Controller)结构[2],开发人员可以方便快捷地直接使用GEF提供的MVC框架来构建新的应用程序,而无需重新设计新的MVC框架。GEF所提供的MVC框架与其它一些MVC框架相比,GEF的MVC框架减少了模型和视图之间的依赖。因此,可以根据需要任意选择模型和视图的组合。同时,通过对GEF中的命令机制,即Command模式[3]的研究,可以很轻松地设计出具有撤销/重做功能[4]、符合MVC结构的复杂图形化编辑系统。因此,研究其命令机制对于掌握整个框架和设计开发MVC结构的图形编辑器软件有着很高的价值。

收稿日期:2009-12-13;**修回日期**:2010-02-28
作者简介:刘绪斌(1985-),男,硕士研究生,研究方向为智能测量与控制系统;王 良,副教授,硕士生导师,主要从事智能测量与控制系统与电力系统自动化的研究。

1 GEF中MVC模式简介

MVC(Model-View-Controller,模型-视图-控制器模式)是软件工程中的一种软件架构模式。它把软件系统分为三个基本部分:模型(Model)、视图(View)和控制器(Controller)。

GEF的MVC结构如图1所示。其模型、控制器和视图介绍如下:

模型:对应业务逻辑,实现相应的接口,并添加事件监听器,实现对模型事件的监听。它不知道任何与

视图相关的信息,它只与控制器打交道。为了在模型发生事件时,控制器能够捕获这一信息,于是在模型中添加了控制器为事件监听者。当模型发生事件时,控制器就会通知各个视图进行更新。

控制器:模型与视图之间的双向通道,也是整个 MVC 结构的核心。它不仅要监听模型事件,还要将视图编辑结果反映到模型上。GEF 中它由 EditPart[1,5] 实现,安装编辑策略,监听模型变化,处理模型变化请求,执行相应的 Command 命令完成相应的操作,绘制图形。

视图:GEF 目前能够提供两种视图——图形(GraphicalViewer)视图和树状(TreeViewer)视图。视图任务同样也很繁重,除了显示模型外,还要提供编辑、工具提示(ToolTip)等功能。GEF 将使用 EditPartViewer[1,5]作为视图,其作用类似于 JFace 中的 Viewer。

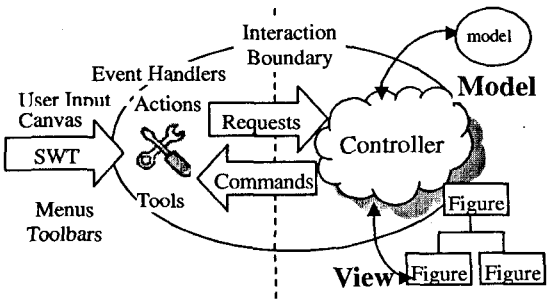

图 1　GEF 中 MVC 结构图

为了提高代码的复用性,以及实现 GEF 编辑器的功能,GEF 采用了 Command 模式提供了对模型的操作方式。因此,在使用 GEF 框架时,除了需要掌握模型、视图和控制器外,还应该了解请求和编辑策略以及命令模式。

2　Command 设计模式

要了解 GEF 中 Command 模式的使用,首先需要了解 GoF 设计模式[3]中的 Command 模式,掌握其意图、结构、参与者和效果。这是 GEF 中 Command 模式的原型。它对于研究 GEF 中 Command 模式有着基础性的作用。

2.1　GoF 设计模式中 Command 模式

设计模式(Design Pattern)是一套被反复使用、多人知晓、分类编目、代码设计经验的总结[3]。在 GoF 所著的 Design Pattern 一书中,一共总结了 23 种设计模式,共分为三大类。Command 模式属于其中的(对象)行为型模式。

Command 模式,又称命令模式。图 2 所示为其 UML 类图。其意图是把一个请求封装到一个对象中,从而使你可用不同的请求对客户进行参数化[3]。命令模式把发出命令的责任和执行命令的责任分割开,委派给不同的对象。允许请求的一方和发送的一方独立开来,使得请求的一方不必知道接收请求的一方的接口,更不必知道请求是怎么被接收,以及操作是否被执行,何时被执行以及是怎么被执行的。除此之外,支持可撤消的命令操作[6]。

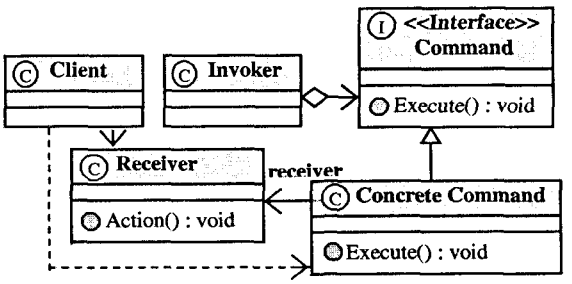

图 2　命令模式的 UML 类图

Command 模式降低 Client 和命令接收者的耦合,是命令请求和命令执行的对象分割便于修改和扩张,增加新的 Command 也无需改变已有的类,除此之外,还便于聚合多个命令(Composite 模式)。因此,Command 模式在许多面向对象系统中广为使用,尤其是在 MVC 体系结构中,MVC + Command[7] 已经成为解决这类复杂问题的模式组合。图 2 中各个参与者的介绍以及其协作详见文献[3]。

2.2　GEF 中 Command 模式实现

为了介绍 GEF 中 Command 模式是如何实现的,首先应该了解 GEF 应用程序的工作方式。

GEF 应用程序的工作方式:EditPartViewer 接受用户对模型对象的操作,而每个模型对象(Model)都对应一个 EditPart(控制器)对象。每一个 EditPart 对象又包含一组编辑策略(EditPolicy),其中,每一个编辑策略会对应一些 Command 对象。通过 Command 对象最终实现对模型的修改。

Command 模式实现:当用户操作时,用户的操作转换为 Request 分配给适当的编辑策略,由编辑策略根据不同的 Request 创建适当的 Command 对象。在 EditPolicy 中存在一个接口:getCommand(Request request),该接口根据不同的请求产生不同的 Command 对象。由于 EditPolicy 是注册到 EditPart 中的,而作为控制器的 EditPart 又是一一对应模型的,因此,模型是作为 Receiver 的,与图 2 所示一致。

在 GEF 中 Command 只是一个接口,其中包含了三个重要的接口函数:execute()、undo()和 redo()。在实现过程中,只需要根据应用,实现接口,覆盖三个函数即可。其对应图 2 中的 Command 接口。所产生的 Command 对象都会存储在 EditDomain(用于维护视

图、命令等对象，一般每个 Editor 对应一个该对象）的命令堆栈里。命令堆栈的弹出，压入用于实现撤消/重做功能，起到命令管理器的作用。

图 1 中 Commands 连接着 MVC 的 EditPart 与 EventHandlers。这里 EventHandlers 中的 Tools 对应于图 2 中的 Invoker。GEF 的 Tool 接口用于解析来自于编辑域和视图中的鼠标键盘消息。其通过使用 EditDomain 中的命令堆栈，来执行相应的命令。从而完整地实现了 Command 模式，完美地将 MVC 与 Command 模式结合在一起。

3 GEF 系统实例设计

在 GEF 的基础上，设计开发了一款基于设计模式的电力系统无功展示编辑软件[8,9]（基于已有的潮流数据）。采用 MVC + Command 的软件体系结构。能够显示变电站模型、拓扑链接；能够对变电站模型进行各种基本操作（创建、删除、移动、链接等）；能够实现操作的撤销/重做，以及层次化元件模型[10]等。

在设计中，将介绍编辑器软件的部分设计类图，包括模型、控制器以及 Command 模式。

3.1 系统 UML 类图

（1）模型与控制器类图。

如图 3 所示，由于变电站、发电机、链接等模型都具有类似的特征，所以不妨将其抽象为 NodeModel 类。由于模型发生变化时，必须将变化信息通知给控制器，由它来负责视图的刷新，因此，可以把侦听器侦听属性的行为，提升到父类中。因此，定义一个实现 PropertyChangeListener 接口的 AbstractNodeModel 类。

在控制器方面，采用了 GEF 已有的控制器 AbstractGraphicalEditPart，对其进行扩展从而得到自己需要的 SustationEditPart，ConnectionEditPart 等多种控制器。在多种控制器中，根据对应的模型，创建返回对应的 IFigure 对象，用于显示模型。

（2）策略与命令类图。

如图 4 所示，对于注册于控制器的策略类，需要将其在 ComponentEditPolicy 类的基础上按照不同模型需要的命令操作，进行不同的覆盖实现，得到各种包含不同 Command 的策略。

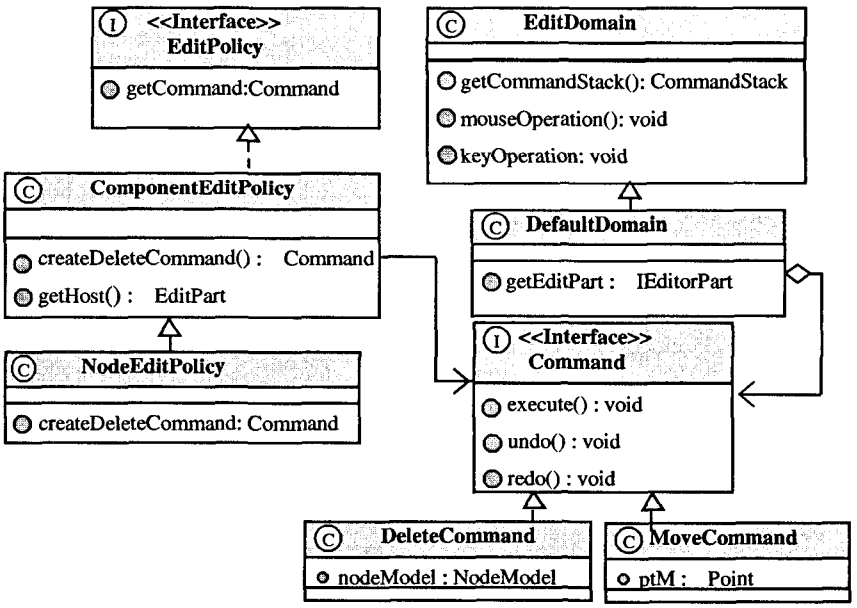

图 4 GEF 中 Command 模式 UML 类图

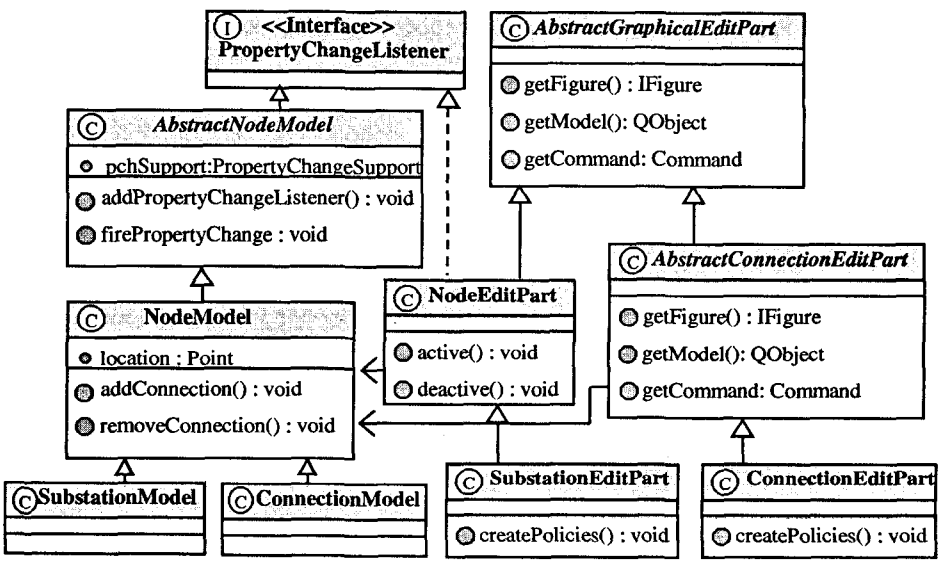

图 3 模型（Model）与控制器（Controller）UML 类图

策略负责全部 EditPart 控制器的编辑工作，其最重要的函数就是 getCommand（Request request），根据请求，产生不同的命令。在 AbstractGraphicalEditPart 接口中存在一个 installEditPolicy（Object key，EditPolicy editPolicy）函数，这个函数根据特定的角色（Role）安装相应的编辑策略。

对于 Command 模式，只需要实现 Command 接口里的三个主要函数就可以，如何使用框架 GEF 已经给

出,无需自己设计。在整个应用程序中包含一个 EditDomain 命令管理器对象,负责管理 Command 对象,实现撤销、重做等操作。

3.2 部分代码示例

(1) Command 模式部分代码(省略连接操作)。

```
public class DeleteCommand extends Command { //删除命令
//DiagramModel 作为 NodeModel 父类容器
private final DiagramModel parent;
private final NodeModel child;
private boolean isRemoved;
public DeleteCommand(DiagramModel parent, //构造函数
NodeModel child) {
if (parent = = null||child = = null) {
throw new IllegalArgumentException();}
this.child = child; this.parent = parent;
}
public boolean canUndo() { //判断能否撤销
return isRemoved;
}
public void execute() { redo();} //执行操作
public void redo() { //重做
if(isRemoved) {
removeConnections();} //删除时,要除去与其相连的链接
public void undo() { //撤销
if (parent.addChild(child)) { addConnections();}}
}
```

(2) 连接策略类代码。

```
public class DeleteNodeEditPolicy extends ComponentEditPolicy { //删除节点策略
protected Command createDeleteCommand(
GroupRequest deleteReq) //创建删除命令
{
if(deleteReq.getType = = DELETECMD){
return new DeleteCommand();}} // 根据类型返回删除对象
```

(3) Undo/Redo 功能注册到工具条。

```
public class EditorActionBarContributor extends ActionBarContributor {
protected void buildActions() {
addRetargetAction(new UndoRetargetAction());
addRetargetAction(new RedoRetargetAction());
}
public void contributeToToolBar(IToolBarManager toolbar Manager) {
super.contributeToToolBar(toolBarManager);
//安装 undo/redo 命令
toolBarManager.add(getAction(ActionFactory.UNDO.getId()));
toolBarManager.add(getAction(ActionFactory.REDO.getId()));
}
protected void declareGlobalActionKeys() {
addGlobalActionKey (ActionFactory.SELECT_ALL.getId());}
}
```

4 结束语

文中通过对 Eclipse GEF 中 Command 命令模式的研究,展现了 GEF 框架下命令模式是如何与 MVC 模式相组合使用的,也展现了设计模式在各种面向对象系统设计中的重要作用[11,12]。最后,通过使用 MVC + Command 组合模式,实现了软件的撤销、重做等复杂功能,也充分地体现了 GEF 的便捷、快速等特点。通过对 GEF 框架和命令模式的应用研究,使得普通开发人员认识到轻松设计出可复用系统已经成为现实。

参考文献:

[1] 张文毅,尤晋元. 自动生成 GEF 图形编辑器的研究与实现[J]. 计算机工程,2006,32(16):31 - 33.

[2] 冯相忠 . 基于 MVC 设计模式的 Struts 框架及其应用的研究[J]. 计算机技术与发展,2006,16(8):131 - 136.

[3] Gamma E, Helm R, Johnson R, et al. Design Patterns: Elements of Reusable Object - Oriented Software[M]. [s. l.]: Addison Wesley,1995.

[4] 黄逸民,袁　繁,王　树. 利用设计模式实现电力图形编辑系统(iSee3.0)中的 Undo/Redo 功能[J]. 计算机工程与应用,2003,39(11): 127 - 131.

[5] 黄　瑛. 基于 GEF 的统一用户界面插件设计[J]. 现代电力,2008,25(4):71 - 74.

[6] Wei Gang, Zhao Weidong, Li Qiyan. Command Design in the CAD Software[C]//11th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2007). Melbourne, Australia: [s. n.],2007.

[7] 杨小妹,高　娇,杨志东. MVC 与 COMMAND 设计模式的应用[J]. 河北北方学院学报:自然科学版,2006,22(4):59 - 63.

[8] 汪小寒,项响琴,陈　洁,等. 基于 PowerBuilder 的运费编辑器的设计与实现[J]. 计算机技术与发展,2006,16(4):135

(下转第 127 页)

中指定先进行 IPv6 地址解析,当 IPv6 解析失败后再进行 IPv4 地址解析。这样就可以达到在 IPv6 资源存在的情况下被优先使用的目的。

(4)丰富 IPv6 网络上的资源,吸引大家来使用 IPv6 网络。对于校园网络来说,学生和老师是主要的使用者,学生们精力充沛,喜欢尝试新事物,可以搭建一个互动平台,让同学们来丰富 IPv6 网络上的资源,让他们上传资源供大家分享,对于他们上传的资源如果有人下载就给予上传者一定的回报(比如上网流量赠送等),下载使用的人越多,上传者得到的回报就越多,调动他们的积极性,这样大家逐渐接触、了解、使用 IPv6 网络资源,无形中就培育出了使用群体,推动 IPv6 网络的建设和发展。

推动 IPv4 向 IPv6 过渡将是一个长期艰巨的任务,还有很多问题等待探索与解决。比如:

网络性能问题:重庆大学正在进行试商用,IPv6 和 IPv4 要同时跑大量数据,对核心和汇聚交换机性能是一个严峻考验。要有一个良好的体系架构,一般采用大汇聚结构。IPv6 一定用硬件方式实现,软件方式只能做试验。IPv6 试商用首先要保证 V4 网的正常使用。

认证计费问题:目前各学校已经建立了认证计费系统,并有自己的一套管理收费办法,数字化校园统一身份认证,IPv6 技术升级后,能否替换原有的计费系统,存在风险。

地址管理问题:IPv6 地址自动分配适应前期,后期如何合理有效管理分配 IPv6 地址,如何实现 IPv6 源地址认证与校园统一身份认证、计费认证结合实现单点登录是一个很大的难题。

防火墙问题:目前虽然有支持 IPv6 的防火墙,一般是包过滤防火墙,与 V4 防火墙相比有很大差距,一般学校防火墙不支持 IPv6,对 IPv6 可以采用跳接方式跨越方式跳过防火墙。

3　结束语

较详细地介绍了 IPv4/IPv6 的常用互通技术,给出了一些鼓励引导大家使用 IPv6 的思路并分析了可能存在的问题。IPv4 地址即将耗尽,推动部署 IPv6 既是形势所迫也是战略考虑,但是只有较好地解决 IPv4/IPv6 互通的问题,使 IPv4 和 IPv6 能够容易方便地互访各自的资源,才能有力地推动 IPv6 的部署与试商用。同时,在部署 IPv6 的时候采取一些方法有意引导大家接触、了解、使用 IPv6 网络资源,从而加速 IPv4/IPv6 的过渡进程。

参考文献:

[1] 马　严,赵晓宇. IPv4 向 IPv6 过渡技术综述[J]. 北京邮电大学学报,2002,25(4):2-5.

[2] 杨惠仁,吕　波,谢晓尧. IPv6 驻地网部署方案研究[J]. 计算机技术与发展,2007,17(11):60-63.

[3] 张光妲. 基于 IPv6 的下一代校园网的过渡研究[D]. 哈尔滨:哈尔滨工程大学,2007.

[4] 马军锋. IPv4/IPv6 过渡技术及其标准化进展[J]. 电信网技术,2009(9):35-38.

[5] Chen J, Chang Y, Lin C. Performance Investigation of IPv4/IPv6 Transition Mechanisms[J]. Journal of Internet Technology, 2004, 5(2): 163-169.

[6] 庄正松,吴家皋,吴清亮,等. 互联网基本服务 IPv4/ IPv6 过渡的研究与实现[J]. 计算机技术与发展,2006,16(8):13-16.

[7] 陈志亮. IPv6 访问 IPv4 的过渡技术研究与实践[D]. 天津:天津大学,2007.

[8] 刘利强,吴永英,王勇智. IPv6 下 Socket 网络编程的研究与实现[J]. 计算机技术与发展,2006,16(6):201-203.

[9] 郭晓冬,郭汝廷. 实践 IVI[J]. 中国教育网络,2009(4):12-14.

[10] Govil J, Govil J. On the Investigation of Transactional and Interoperability Issues between IPv4 and IPv6[C]//2007 IEEE Electro/ Information Technology Conference (EIT 2007). Chicago, USA: [s. n.], 2007.

[11] Bi J, Wu J, Leng X. IPv4/IPv6 Transition Technologies and Universal Architecture[J]. International Journal of Computer Science and Network Security, 2007, 7(1): 232-242.

[12] Abidah Hj Mat Taib, UiTM Perlis, Rahmat Budiarto, USM. Security Mechanisms for the IPv4 to IPv6 Transition[C]// The 5th Student Conference on Research and Development (SCOReD 2007). [s. l.]: [s. n.], 2007: 11-12.

(上接第 103 页)

-137.

[9] 董朝霞,陈青华,范　斗. 电力仿真系统图形编辑器面向对象的设计与实现[J]. 继电器,2002,30(6):40-42.

[10] Chen Liyan. Application Research of Using Design Pattern to Improve Layered Architecture[C]//IITA International Conference on Control, Automation and Systems Engineering. [s. l.]: [s. n.], 2009.

[11] 沈　建,雷　航,石浩鸿. 设计模式在光传输网管系统中的应用研究[J]. 计算机技术与发展,2007,17(3):232-235.

[12] 刘海岩,锁志海,吕　青,等. 设计模式及其在软件设计中的应用研究[J]. 西安交通大学学报,2005,39(10):1043-1049.