

设计模式在光传输网管系统中的应用研究

沈建¹, 雷航¹, 石浩鸿²

(1. 电子科技大学 计算机科学与工程学院, 四川 成都 610054;

2. 深圳中兴通讯股份有限公司本部事业部, 广东 深圳 518004)

摘 要:设计模式是经过验证可复用的成功设计和体系结构,文中详细地分析了设计模式中的 Builder 模式、Bridge 模式和 Command 模式在光传输网管系统中的应用,证明了在光传输网管系统中运用设计模式能够有效地提高系统质量、优化系统结构。同时也可以为其他的网管系统所借鉴。

关键词:设计模式;光传输网管系统;生成器模式;桥接模式;命令模式

中图分类号:TP393.07

文献标识码:A

文章编号:1673-629X(2007)03-0231-02

Design Patterns Applied in Optical Transport Integrated NMS

SHEN Jian¹, LEI Hang¹, SHI Hao-hong²

(1. School of Computer Science & Engineering, University of Electronic

Science and Technology of China, Chengdu 610054, China;

2. Headquarter Division of Shenzhen ZTE Corporation, Shenzhen 518004, China)

Abstract: The design patterns are a set of successful designs and system structure. Detailed analyses the application of Builder pattern, Bridge pattern and Command pattern in optical transport integrated NMS. It has been proved that the application of design patterns can enhance the quality and optimize the system structure.

Key words: design pattern; NMS; builder pattern; bridge pattern; command pattern

0 引言

光传输网管系统是一个功能强大,且结构相当复杂的光网络管理系统。因此,为了搭建一个功能完备、易于维护的系统结构,在它的系统设计中引入了大量的设计模式。设计模式使开发人员可以简单方便地复用成功的设计和体系结构,同时将已证实的技术表述成设计模式也会使新系统开发者更加容易理解其设计思路^[1]。设计模式的核心是把软件设计中不断重复发生的问题以及解决问题的优良方案抽象出来形成模式,可以说是软件人员在面向对象软件设计中,经过验证的成功解的记录和提炼。

1 设计模式

设计模式是对被用来在特定场景下解决一般设计问题的类和相互通信的对象的描述。模式依据其目的

可分为创建型(Creational)、结构型(Structural)或行为型(Behavioral)三种^[1]。创建型模式与对象的创建有关,它的意图都是将创建细节隐藏,使客户代码不受制于创建过程;结构型模式主要是处理类或对象的组合;行为型模式是用来对类或对象怎样交互和怎样分配职责进行描述。

设计模式使使用者可以在比设计表示或者编程语言更高的抽象级别上谈论一个系统,从而降低了其复杂度。同时结合现有的其它面向对象开发工具,如对象建模技术(OMT)、形式化语言等^[2],还可以有效地展示如何使用诸如对象、继承和多态等基本技术。

2 光传输网管中的设计模式

光传输网管系统是基于面向对象技术设计的,在开发过程中应用了多种设计模式。下面就生成器(Builder)模式、桥接(Bridge)模式、命令(Command)模式分析它们在光传输网管系统中的应用。

2.1 生成器(Builder)模式

在光传输网管系统的客户端中经常会使用到一种“分页表格”控件,在其他网管软件中也经常会用到,这

收稿日期:2006-05-30

作者简介:沈建(1979-),男,江苏盐城人,硕士研究生,研究方向为计算机通信与网络管理;雷航,博士,副教授,研究方向为嵌入式软件技术、软件可靠性测试和评价技术等。

是一个复杂的公用控件,它在 JTable 的基础上提供分页、排序、列定制等常用功能。像很多 Java swing 控件一样,JTable 通过 Model 管理数据(即表格的列、行数据),通过渲染器 Renderer 定制显示,通过 Editor 实现单元格编辑^[3]。分页表控件也继承了这一机制。实际使用中,每一个表格使用者的表格样式都是完全不同的,主要体现在列的数目和种类、单元格的显示格式等。作为公共控件的分页表应当尽可能多地封装分页、排序等功能,同时又不受限于丰富多变的表格样式。对于一个完整分页表的创建,必不可少地由翻面板创建、可排序的表头创建、列样式确定等一系列复杂步骤组成,对于列样式的确定,在基类中无法完成,必须延迟到实现时再确定具体的列格式。由此考虑将列格式的确定这一步骤分离出来,由单独的类负责完成。这就引入 Builder 模式。

Builder 模式是一种常用的创建型模式,它是将一个复杂对象的构建与它的表示分离,使得同样的构建过程可以创建不同的表示^[4]。如图 1 所示,在使用分页表控件时,首先需要派生自己的 TNMSTableModel,以确定表格样式,然后在实例化分页表时,将自己的 Model 传入,以创建完整的符合期望的分页表。

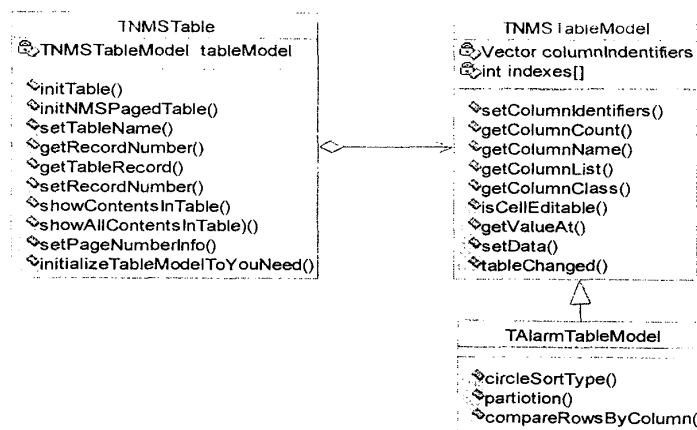


图 1 Builder 模式的类结构图

2.2 桥接(Bridge)模式

在软件开发过程中,人们经常希望能够把功能模块的使用者和功能模块的开发者之间的耦合分开。对于功能模块的使用者,希望无论自己怎样设计自己的程序,在调用功能模块时,都会得到正确的、满意的结果;对于功能模块开发者,则希望无论自己怎样实现自己的模块,绝不会影响、耽误功能模块使用者的设计和进度。这也就是设计模式中的 Bridge 模式,这种模式能够极大地简化类结构的设计,提高代码的复用性和可维护性。Bridge 模式是一种常用的结构型模式,它主要是为了实现将抽象部分与实现部分分离,使它们

都可以独立地变化^[4],它最能体现设计模式的“针对接口进行编程”和“使用聚合不使用继承”两个原则。

在光传输网管系统的告警管理模块中,获取当前告警时,界面显示描述每一条告警的各个方面的信息,如告警源、严重等级、产生时间等。为了支持告警信息项的变化,如新增了描述项 - 告警厂商类型,则需要把告警信息项看作是易变化的实现线路的内容。那么当前告警模块就是主控制线路的内容,如图 2 所示。

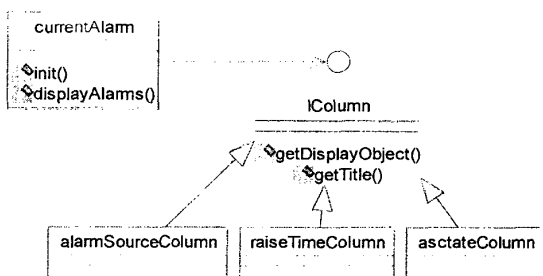


图 2 Bridge 模式的类结构图

当前告警模块 CurAlarm 在 init 函数中,会从配置文件中读取所有告警显示项(表格区中的列)的实现类,然后把这些实现类实例化并保持对它们的引用。然后,在有告警需要显示时,调用 displayAlarms 方法,

在此方法内,会依次遍历对每个告警显示项的引用,然后调用它的 getDisplayObject 方法,以便获得输送给表格进行显示的对象。这样每个告警显示项实现类可以自行改变自己的实现逻辑,还可以增加或减少显示项个数(在配置文件中配置);同时,当前告警模块可以根据自己的需要而增加、删除函数声明,或改变处理逻辑。双方完全解耦。

2.3 命令(Command)模式

Command 模式是一种常用的行为型模式,它将一个请求封装为一个对象,从而使你可以用不同的请求对客户进行参数化;

对请求排队或记录请求日志,以及支持可撤消的操作^[4]。在光传输网管的业务处理模块中,需要将界面下发的命令进行拆分之后下发到设备。但拆分的规则和报文对不同的命令可能不同,而中间的处理流程可能有相同的。所以处理流程和报文的拆分应该分开处理,也就是将业务相关的和业务无关的处理分开。业务相关的部分可以由 Command 来实现,业务无关的部分另外定义一个类来实现,多条命令可以共享业务无关的实现。

如图 3 所示,定义了一个命令处理类 CCmdProces-

(下转第 235 页)



图 2 扩展菜单功能后的提示信息

a. 声明包类型和内容。

```
< RDF:Seq about = "chrome://navigator/content/navigator. xul"
>
  < RDF:li > chrome://myself/content/myself. xul < /RDF:li >
//描述包中每个资源
< /RDF:Seq>
```

b. 描述包中的一些信息。

```
< RDF:Description RDF:about = "urn:mozilla:package: myself"
  chrome:displayName = "my first extension application" //显示给
  用户的包的标题
  chrome:author = "qi yanjun " //包作者
  chrome:name = "myself" //包名称,应与 about 中包名称一致
  chrome:extension = "true" //是一个扩展,在扩展窗口可视
  chrome:description = "Displays Tools menu." > //包描述信息
< /RDF:Description>
```

最后,注册此应用,即用户新的扩展。在 installed - chrome. txt 的最后加入应用的入口: content, install, url, resource: /chrome/ myself /content/

要使一个扩展浏览器的功能基本完成,还可以加

(上接第 232 页)

sor_ T 和一个命令基类 CCommand。CCmdProcessor_ T 负责命令的调用,以及命令共同的处理,如命令回调接点的注册、命令的下发等。所有具体的命令都从 CCommand 派生, CGetMultiNeCurPerf_ G2M 是实现多网元性能查询命令报文的处理。通过 Command 模式,实现了命令执行和调用的解耦,而新增命令对命令的调用没有任何影响,这使得系统结构更加优化,维护更加容易了。

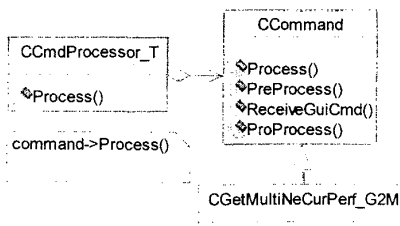


图 3 Command 模式的类结构图

3 结束语

文中按设计模式中的三类模式:对象创建型模式、对象结构型模式、对象行为型模式,分别取一种简单介

入相关的步骤,此略。启动 Mozilla,使用:chrome://myself /content/ myself. xul 就可以访问定义好的 XUL 文件,在浏览器菜单 Tools 中出现定义的功能(见图 1),当选此菜单项会出现如图 2 的信息框。

3 结束语

XUL 以其强大的用户界面开发功能使得用户界面得到了较好的扩展,使复杂的界面可以通过标记语言编写,因此比传统的编程语言开发用户界面要简单、易写,并且可以根据用户需要扩展浏览器的功能,快速地定制用户界面。

参考文献:

- [1] Limodou. XUL Tutorial[EB/OL]. 2005 - 09. <http://www.xulplanet.com/>.
- [2] Mozilla. XML User Interface Language[EB/OL]. 2005 - 01. <http://www.mozilla.org/projects/xul/>.
- [3] 朱先忠. 基于 AJAX 和 JSF 打造丰富的因特网组件之 XUL 篇[EB/OL]. 2006 - 05 - 10. <http://tech. 51cto. com/art/200605/26200. htm>.
- [4] Gross C. Ajax Patterns and Best Practices[M]. [s. l.]: A-press, 2006.
- [5] Asleson R, Schutta N T. Ajax 基础教程[M]. 金 灵等译. 北京:人民邮电出版社, 2006: 42 - 45.

绍在光传输网管系统中的应用。设计模式带来了很多好处,增强了程序的复用能力和可扩展能力及某些非功能需求,但是有时也可能会使得软件变得复杂,效率下降,使用设计模式需要根据实际情况要求,进行综合考虑和折中,一味套用模式并不可取^[5]。但经过分析表明,合理地应用设计模式能够极大地改善系统设计,优化系统结构,提高系统的设计质量。

参考文献:

- [1] Gamma E, Helm R, Johnson R, et al. Design Patterns Elements of Reusable Object - Oriented Software[M]. 北京:机械工业出版社, 2002.
- [2] 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 - 1047.
- [3] Eckel B. Java 编程思想[M]. 第 2 版. 侯 捷译. 北京:机械工业出版社, 2002.
- [4] Loway A S, Trott J. 设计模式精解:面向对象设计的新视角[M]. 透 明译. 北京:清华大学出版社, 2002.
- [5] 马 争, 周 艳, 谢世波. 设计模式在网管系统中的设计与实现[J]. 电子科技大学学报, 2004, 33(5): 523 - 526.