第16卷 第3期
2006年3月

计算机技术与发展
COMPUTER TECHNOLOGY AND DEVELOPMENT

Vol.16 No.3
Mar. 2006

基于Struts与SiteMesh框架的Web应用研究

薛 亮,葛 玮,郝克刚
(西北大学 计算机科学系 软件工程研究所,陕西 西安 710069)

摘 要:介绍了MVC模式与Decorator模式,以及基于这两种模式的两种应用框架Struts与SiteMesh。并从可扩展性、可复用性、可维护性、可控制性的角度出发具体研究了这两种框架。最终将这两个框架结合,形成一个典型的框架应用结构,为软件架构设计提供了很好的设计思路与参考模型。将此框架结构应用于实际项目的设计与开发中,收到了良好的效果。

关键词:设计模式;框架;MVC;Decorator;Struts;SiteMesh;体系架构
中图分类号:TP311.5 **文献标识码**:A **文章编号**:1005-3751(2006)03-0119-03

Web Research Based on Struts and SiteMesh Framworks

XUE Liang, GE Wei, HAO Ke-gang
(Inst. of Software Eng., Dept. of Computer Sci., Northwest University, Xi'an 710069, China)

Abstract: In this paper, introduce the MVC pattern and the Decorator pattern, and two frameworks - Struts and SiteMesh, which are based on above patterns. Study the two frameworks in the aspects of expansibility, reusability, maintenance and management. At last, incorporate the two frameworks and form a new representative framework, and this framework gives a good way and a good reference model to design the software architecture. Use this framework into practice on the design and programming of software and have got good result.

Key words: design pattern; framework; MVC; Decorator; Struts; SiteMesh; system frameworks

0 引 言

软件业的发展不仅要求软件有更高的生产率和可靠性,而且对软件的可复用性和可维护性也提出了更高的要求。设计模式以文档的形式把面向对象的软件设计经验记录下来,并予以系统的命名、解释和评价,使开发人员在进行系统的设计与开发时,可以使用已有的成功经验而不必重新设计解决方案,使设计者更容易理解其设计思路,能为自己的问题找到更合适的解决办法,帮助设计者更快更好地完成系统设计。正如Christopher Alexander所说的那样,"每一个模式描述了一个在我们周围不断重复发生的问题,以及该问题的解决方案的核心。这样,你就能一次又一次地使用该方案而不必做重复的劳动"。尽管Alexander所指的是城市和建筑模式,但他的思想也同样适用于面向对象设计模式[1]。

框架是一种面向对象的软件重用技术。基于模式的框架的应用,在软件的体系结构中起着重要的作用。在当前的B/S结构的软件开发中,软件的层次清晰、易于开发、易于维护的思想更是得到广泛应用。而现有的Web框架,基本上都可以看成是MVC模式的一个实现。

1 MVC模式

MVC模式是Xerox PARC在20世纪80年代为编程语言Smalltalk-80发明的一种软件设计模式[2]。MVC即Model-View-Controller的缩写,该模型将功能划分为相互关联的3个组件:模型(Model)、视图(View)和控制器(Controller)。

模型(Model):就是封装数据和所有基于对这些数据的操作,是应用程序的主体部分。

视图(View):就是封装的是对数据显示,显示从模型中提取的数据,即用户界面。

控制器(Controller):就是封装外界作用于模型的操作和对数据流向的控制等。

MVC模式是Web应用系统中一种常用的设计模式。它利用控制器来分离模型和视图,达到一种层间松散耦合的效果,从而减弱了业务逻辑接口和数据接口之间的耦合性,提高了系统灵活性、复用性和可维护性。MVC结构很好地实现了数据层与表示层的分离,在系统结构与开发效率两个方面都能取得较好的效果。

1.1 Struts框架

Struts是作为Apache Jakarta项目的组成部分,该项目的目标是为建立Java Web应用程序而提供的一个开源框架。Struts是MVC的一种实现,通过使用Struts框架

收稿日期:2005-06-26

作者简介:薛 亮(1980—),男,内蒙古鄂尔多斯人,硕士研究生,研究方向为软件工程、网络安全;葛 玮,副教授,硕士生导师,研究方向为软件工程、软件测试;郝克刚,教授,博士生导师,研究方向为软件工程、软件理论、形式化方法。

可以改进和提高 Java Server Pages (JSPs)、Servlet、标签库以及面向对象的技术在 Web 应用程序中的应用[3]。

1.2 Struts 框架的工作原理

Struts 框架通过相互关联的 3 个组件:模型(Model)、视图(View)和控制器(Controller)协调工作,提高了系统的可维护性和可复用性。具体功能分配如下:

模型:在 Struts 中,主要存在三种 bean,分别是 Action,ActionForm,EJB 或者 Java Bean。ActionForm 用来封装客户请求信息,Action 取得 ActionForm 中的数据,再由 EJB 或者 Java Bean 进行处理,最后将处理结果放入 FormBean 中,用视图来显示。

视图:Struts 应用中的 View 部分是通过 JSP 技术实现的。Struts 提供了自定义标记库可以使用,通过这些自定义标记可以很好地和系统的 Model 部分交互,通过这些自定义标记创建的 JSP 表单,可以实现和 Model 部分中的 ActionForm 的映射,完成对用户数据的封装,同时这些自定义标记还提供了像模板定制等多种显示功能[2]。

控制器:在 Struts 中,ActionServlet 起着一个控制器(Controller)的作用。ActionServlet 是一个通用的控制组件。这个控制组件提供了处理所有发送到 Struts 的 HTTP 请求的入口点,它截取和分发这些请求到相应的动作类(这些动作类都是 Action 类的子类)。另外控制组件也负责用相应的请求参数填充 ActionForm(通常称之为 FromBean),并传给动作类(通常称之为 ActionBean)。动作类实现核心商业逻辑,它可以访问 JavaBean 或调用 EJB 等。所有这些控制逻辑利用 Struts - config. xml 文件来配置。

2 Decorator 模式

Decorator 即装饰,Decorator 模式即装饰器模式。Decorator 模式是对象结构型模式的一种。结构型模式采用继承机制来组合接口或实现。Decorator 的定义即动态给一个对象添加一些额外的职责[1]。

2.1 为什么使用 Decorator

人们通常可以使用继承来实现功能的拓展,但如果这些需要拓展的功能的种类很繁多,那么会生成很多子类,增加了系统的复杂性。同时,使用继承实现功能拓展,必须可预见这些拓展功能,而这些功能是编译时就确定了,是静态的。当需要在运行期间添加功能,并且由客户程序来决定何时添加何种功能,那么单纯的扩展子类对象是很难办到的。Decorator 提供了"即插即用"的方法,在运行期间决定何时增加何种功能,扩展的功能由用户动态决定加入的方式和时机。所以,使用 Decorator 模式相比用生成子类方式达到功能的扩展更为灵活。

2.2 SiteMesh 框架

SiteMesh 框架[3]是 Decorator 模式的一个很好的应用与实现,它将资源文件(HTML,JSP 等)抽象为被装饰的对象,然后用定义好的装饰器页面去对其进行装饰即添加新的功能。其中装饰器页面是在配置文件中配置的,是根据需求自己定义的,所以是动态的。

在开发 Web 应用特别是 J2EE 应用的时候,由于 Web 页面是由不同的人所开发,所以开发出来的界面通常是各种各样,难以形成一个统一的界面风格,随着项目的进一步的开发,要求统一的界面风格的紧迫性逐渐浮现了出来。

当要建立可复用的 Web 应用程序时,一个通用的方法是建立一个分层系统,如同下面一个普通的 Web 应用:

1)前端,front - end:JSP 和 Servlets,或 jakarta 的 velocity;

2)控制层框架 Controller:Struts;

3)业务逻辑 Business:主要业务逻辑;

4)持久化框架:hibernate。

但是前端的页面逻辑很难被复用,当你在每一个页面中用数之不尽的 include 来复用公共的 header,stylesheet,scripts,footer 时,一个问题出现了——重复代码。

SiteMesh 是 opensymphony 团队开发的 Web 应用框架之一,旨在提高页面的可维护性和复用性,动态地进行功能扩展。它是由一个基于 Web 页面布局、装饰以及与现存 Web 应用整合的框架。它能在由大量页面构成的项目中创建一致的页面布局和外观,对通用的导航条、版权进行复用。使用它能够帮助开发人员较容易实现页面中动态内容和静态装饰外观的分离。所有这些,都是 GOF 的 Decorator 模式的最生动的实现。其数据流图[4]见图 1。

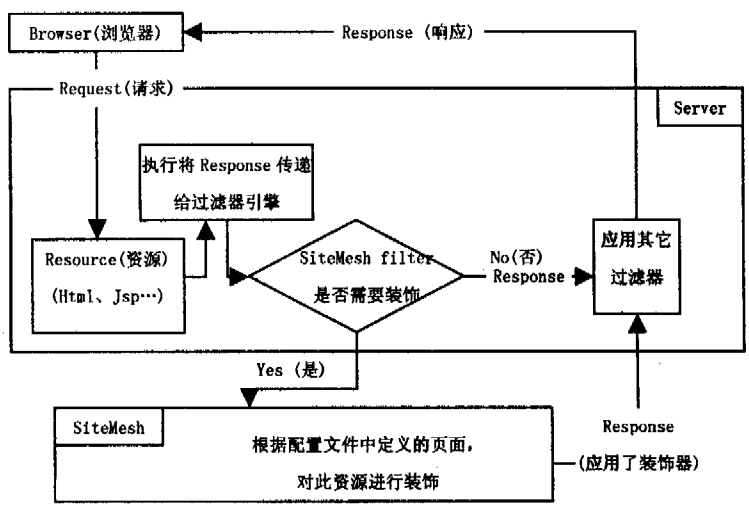

图 1 数据流图

SiteMesh 的原理很简单,它使用 filter 拦截请求,一个请求到服务器后,如果该请求需要 SiteMesh 装饰,服务器先解释被请求的资源,然后根据配置文件获得用于该请求的装饰器,即根据不同的请求根据配置文件获取不同的装饰器模板。最后用装饰器装饰被请求资源,将结果一同返回给客户端浏览器。SiteMesh 这里的分离,使得内容页面,不知道自己将怎么样地被"装饰",而这个装饰由容器

通过配置文件来控制。

用SiteMesh带来的是不仅仅是页面结构问题,它的出现让人们有更多的时间去关注底层业务逻辑,而不是整个页面的风格和结构。它让人们摆脱了大量用include方式复用页面尴尬局面,它也提供了很大的灵活性以及提供了整合异构Web系统页面的一种方案,真正实现页面显示view的内容与框架(或者说布局、导航)的分离。

3　Struts和SiteMesh框架的Web应用

在项目开发实践中,整个系统采用B/S结构,在系统中使用了Struts框架、SiteMesh框架以及JDBC技术。其体系结构主体采用了MVC架构,由Struts来实现这个架构,充分利用了Struts中的Servlet来控制整个业务流程,使用FormBean做为业务数据的载体,而JSP页面作为数据的输出。在这个基础上同时使用SiteMesh框架,由SiteMesh来对JSP页面进行统一管理,对不同的页面进行不同的装饰,最终形成一致的页面布局和外观。系统的体系架构图见图2。

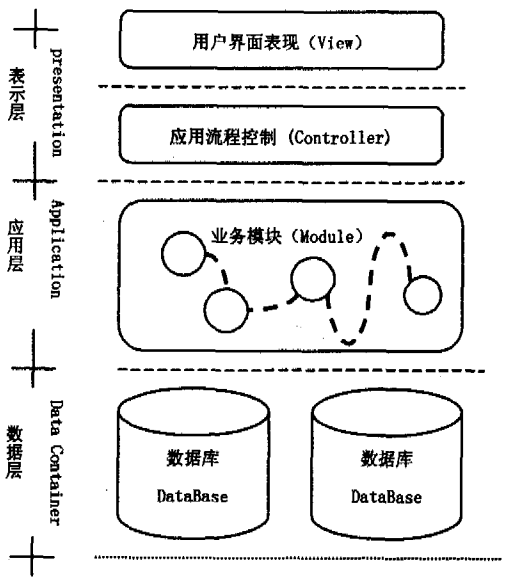

图2　系统体系架构图

View是指用户界面,也就是面向用户的数据表示,在系统中由JSP来实现;Model是指对业务数据/信息的处理模块,包括对业务数据的存取、加工、综合等,在系统中加工处理由Action来完成,数据的保存由ActionForm来完成;Controller则负责View和Module之间的流程控制,也就是完成两个方向的动作:

1)将用户界面(View)的操作映射到具体的Model,以完成具体的业务逻辑;

2)将通过Model处理完的业务数据及时反应到用户界面(View)上。

这个架构被分为三层:表示层、应用层与数据层。数据层主要是用来存储数据,对数据进行操作,使用JDBC来完成;应用层由Struts来完成;在表示层,在Struts框架中使用了SiteMesh框架,由SiteMesh根据配置文件来对JSP进行装饰、统一管理。系统的框架应用结构图见图3。

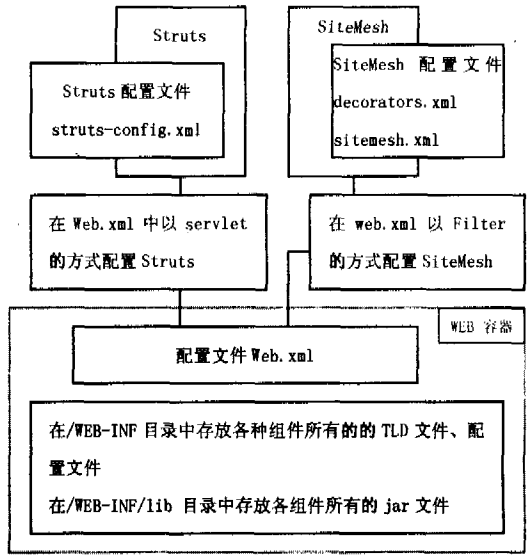

图3　框架应用结构

整个主体框架采用了Struts与SiteMesh两种框架相结合。首先搭建Web服务器,建立Web应用。在Web.xml中以Servlet的方式配置用Struts,其主要用来实现应用流程控制(Controller)的功能;在Struts-config.xml中配置应用中需要的Action与ActionForm来完成应用层的相关工作;配置JSP页面来完成表示层的显示。其次以Filter的方式在Web.xml中配置SiteMeth框架,这里配置url-pattern项,系统将根据该属性项来判断哪些请求将被装饰。并在SiteMesh框架的配置文件decorators.xml中配置装饰器,且可以配置多个装饰器。业务请求根据配置文件中的url-pattern项来匹配使用哪一个装饰器来装饰此页面资源。

4　结束语

基于MVC设计模式的框架软件Struts来实现Web开发应用,充分利用Java的强大功能和平台无关性的特点。Struts是一种优秀的MVC架构框架,可以使开发人员集中精力编写JSP和业务逻辑及与外部数据库进行交互的代码而不用考虑基础框架问题;基于Decorator模式的SiteMesh框架,完全实现了Decorator模式的思想,使页面的管理变得更加简单、方便,最终形成统一的页面。两种框架结合使用,发挥了各自的优点,将是Web系统开发的一个典型开发框架。用户界面与业务逻辑分离,使得开发人员与页面设计人员分工更加明确,更好地协调工作。同时增强了系统的可维护性与可控制性,从而提高应用程序的柔韧性,使整个系统的开发有条不紊,容易集成,便于维护升级。

参考文献:

[1] Gamma E,Helm R,Johnson R,et al.设计模式:可复用面向对象软件的基础[M].李英军,马晓星,蔡　敏,等译.北京:

(下转第126页)

(在寄存器A,B,C,D,E中)。压缩函数逻辑结构对其进行四轮,每轮20个步骤的运算,每一步骤只处理一个字 W_t。全部 L 个分组处理完毕后输出160 bit散列值。表2中列出了依照下式运算后头5个步骤寄存器的缓存值。

表1 改进后的逻辑函数表达式

轮数	逻辑函数	函数表达式
第一轮($0 \leqslant t \leqslant 19$)	f_1(B,C,D)	(B AND C) OR (NOT B AND D)
第二轮($20 \leqslant t \leqslant 39$)	f_2(B,C,D)	B XOR C XOR D
第三轮($40 \leqslant t \leqslant 59$)	f_3(B,C,D)	(B AND C) OR (B AND D) OR (C AND D)
第四轮($60 \leqslant t \leqslant 79$)	f_4(B,C,D)	(B AND NOT C) XOR D

$A,B,C,D,E \leftarrow (E + f_t(B,C,D) + S^5(A) + W_t + K_t), A, S^{30}(B), C, D$

表2 寄存器缓存值

步骤	寄存器缓存值				
0	A	B	C	D	E
1	E_1	A	B^1	C	D
2	D_1	E_1	A^1	B^1	C
3	C_1	D_1	$E_1{}^1$	A^1	B^1
4	$B_1{}^1$	C_1	$D_1{}^1$	$E_1{}^1$	A^1
5	$A_1{}^1$	$B_1{}^1$	$C_1{}^1$	$D_1{}^1$	$E_1{}^1$

上式中,W_t 分别是 t 步骤处理的字 W_t,+ 表示 $mod2^{32}$ 加法。A,B,C,D,E是处理 q 分组时寄存器的起始值。表2中各种符号代表值如下:

$A^1 = S^{30}(A)$ $B^1 = S^{30}(B)$

$C_1 = C + f_1(E_1, A^1, B^1) + S^5(D_1) + W_3 + k_1$

$D_1 = D + f_1(A, B^1, C) + S^5(E_1) + W_2 + k_1$

$E_1 = E + f_1(B,C,D) + S^5(A) + W_1 + k_1$

$A_1{}^1 = A^1 + f_1(C_1, D_1{}^1, E_1{}^1) + S^5(B_1{}^1) + W_5 + k_1$

$B_1{}^1 = B^1 + f_1(D_1, E_1{}^1, A^1) + S^5(C_1) + W_4 + k_1$

$C_1{}^1 = S^5(C_1)$ $D_1{}^1 = S^5(D_1)$ $E_1{}^1 = S^5(E_1)$

据此,可以写出步骤10,15,20以及第二、三、四轮循环模块运算完毕后5个寄存器的缓存值。由此可见算法中上述规律性的存在是一个安全隐患。

SHA-1算法中分组 q 经四轮逻辑运算产生的输出结果$(A,B,C,D,E)_q$,尚需和5个寄存器起始缓存值 CV_q 按 $mod2^{32}$ 相加后,才产生分组 q 的最终的输出 CV_{q+1}。可表示成:

$CV_0 = CV_0$

$CV_{q+1} = SUM_{32}(CV_q, (A,B,C,D,E)_q)$

CV_L = 散列值 MD

原SHA—1算法只是二个32bit字的输入,产生一个32bit字的输出。如果先对每个寄存器进行一次模加法,再进行SHA—1规定的其余的运算,那么对抵抗强抗冲突将会更加有效,更好地隐藏了摘要和明文之间的潜在规律。也即采用三个32bit字的输入产生一个32bit字的输出。

具体说,将分组 q 寄存器的缓存起始值,按顺序每两个进行 $mod2^{32}$ 加法运算后得出 CV_q(AA,BB,CC,DD,EE),再和分组 q 经四轮循环运算输出$(A,B,C,D,E)_q$,相应作 $mod2^{32}$ 加法运算后的输出,得出最终的输出值 CV_{q+1}。全部过程可表示为:

$CV_0 = CV_0$

$CV_{q+1} = SUM_{32}(CV_q(AA,BB,CC,DD,EE), (A,B,C,D,E)_q)$

$CV_L = MD$

$AA = B + C$ $BB = C + D$ $CC = D + E$

$DD = E + A$ $EE = A + B$

可见上述对压缩函数逻辑结构所做的变换在基本不增加计算量的同时,却提高了抵抗强无碰撞的能力。

4 结束语

文中从安全性及运算效率方面对散列算法SHA—1作了深入分析,并由此提出若干改进方法。改进后的算法在安全性及运算效率方面均较原算法有所提高,因此具有更好的可用性。文中还提出了一种安全散列值的计算方法,该方法产生的散列值不仅具有更高的安全性,而且还具有序列号或时间戳的功能。

参考文献:

[1] Rivest R. The MD5 Message - Digest Algorithm[S]. RFC 1321,1992.

[2] Eastlake D, Jones P. US Secure Hash Algorithm1 (SHA1)[S]. RFC 3174,2001.

[3] Dobbertin H, Bosselaers A, Preneel B. RIPMEMD - 160: A strengthened version of RIPMMD[Z]. Fast Software Encryption,1996,LNCS 1039:71 - 82.

[4] National Institute of Standards and Technology. Secure hash standard. NISTFIPS PUB 180 - 1, Washington D C: Department of Commerce, NIST, 1995[EB/OL]. http://csrc.nist.gov/cryptval/shs.html.1995.

[5] 王小云,张金清. MD5报文摘要算法的各圈函数碰撞分析[J]. 计算机工程与科学,1996,18(2):15 - 22.

(上接第121页)

机械工业出版社,2004.

[2] BUILDER COM. MVC设计模式带来更好的软件结构和代码重用[DB/OL]. http://www.zdnet.com.cn/developer/tech/story/0,2000081602,39098006,00.htm,2002 - 11.

[3] The Apache Software Foundation: Struts[DB/OL]. http://struts.apache.org/,2005 - 03.

[4] OPENSYMPHONY Quality Components[EB/OL]. http://www.opensymphony.com/sitemesh/,2005 - 03.