Struts 中的 8 种 J2EE 核心模式

Eight Core J2EE Patterns in Struts

王 勇 (中国农业银行软件开发中心 100073)

摘 要: 本文首先介绍了 Struts 的基本概念,然后用 J2EE 核心模式的理念分析了 Struts 框架中的 8 种核心模式,最后说明了模式之间的关系。

关键词: Struts J2EE 核心模式

1 引言

Struts 是 Apache Software Foundation 支持 Jakarta 项目的一部分。Struts 是基于 MVC 模式的。JSP 处理表现层,ActionServlet 负责控制流程,Action 负责存取业务逻辑类。

Struts 基于以下特点:标签库提供了通用功能;模块可以被应用程序重用;本地化策略减少了重复的 JSP;具有开放的体系结构;是轻量级的;与标准兼容;开源软件;具有良好的文档;和持久层无关。

Struts 框架可以和其它层的框架彼此协作,获得较大复用粒度。Struts 是表示层框架,Spring 主要是业务层框架,Hibernate 和 ibatis 都是持久层框架。这些框架彼此合作,开发 WEB 应用可以采用 Struts + Spring + Hibernate 或者 Struts + Spring + ibatis 的框架组合方式。

2 Struts 中的 8 种 J2EE 核心模式

《J2EE 核心模式》一书对《设计模式》进行了扩充,更关注如何在 J2EE 平台构造一个具有健壮性和伸缩性应用程序的设计模式。J2EE 模式的一个关键之处就是着重说明了如何构造一个分层的应用程序,同时要尽量地抽象化,以及对于表现层、业务层和集成层的清楚地分离。《J2EE 核心模式》包括了三大类 21 个模式。它们分别是 8 个表现层模式,9 个业务层模式,4 个集成层模式。

用核心模式的理念分析 Struts,它主要使用了以下 8 个 J2EE 核心模式。

2.1 前端控制器(Front Controller)模式

前端控制器是一个容器,作为最初的接触点,用来处理所有相关的请求。前端控制器集中了控制逻辑,避免了逻辑的重复,完成了主要的请求处理操作。

前端控制器模式具有以下优点:集中控制;提供系统可维护性;增进了复用;增加了开发团队中职责之间的划分。

现实世界中的传达室就起到了前端控制器的作用。我们想进入一个机关大院,首先要在传达室填写会客单(发起请求),接着工作人员检查证件(验证和授权),然后我们才能被允许入内(分发)。

Struts 中的 ActionServlet 使用了前端处理器模式。它是对于 HTTP 请求进行处理的中心,接受用户请求以及状态改变。Struts 自己并不生成响应,而是使用请求的 URI 来决定是哪一个 Action 来处理该请求。当业务逻辑部分已经执行完后,Action 会选择一个 ActionForward,并将它返回给总控 ActionServlet。总控 ActionServlet 使用在 ActionForward 中存储的路径来调用对应的页面完成 HTTP 响应。这些配置信息都集中在 struts-config.xml 文件中,利于查看、修改系统的控制流程,不用在各个 JSP 页面中跳来跳去地跟踪。修改流程时只需修改配置文件,不需要修改程序本身。该模式可以保证那些和字符集转换、授权和验证、系统服务(例如应用级日志、调试信息)等相关的代码都是集中在一个地方。因此这些应用程序级别的代码就不会在应用程序中重

复或者与视图内容混合在一起。我们在 Windows 上开发应用,移植到 UNIX 生产机上时需要修改字符集,打开或关闭应用级日志、调试信息,只需在此一处修改即可。

数据库连接池技术就可以认为是一种拦截过滤器。数据库连接池拦截、控制了数据库连接的建立和释放,极大地提高了效率。

2.2 应用控制器 (Application Controller) 模式

应用控制器把请求处理组件(例如命令和视图)的获取和调用集中起来。

应用控制器模式具有以下优点:提高了模块化程度;提高了可重用性;提高了可扩展性。

Struts 中的 RequestProcessor 使用了应用控制器模式。Struts 通过在配置文件 struts – config. xml 中声明来实现映射,从而进行操作管理和视图管理。Struts 使用 RequestProcessor 来获取视图内容以及数据管理。ActionServlet 通过调用 Action 类中的一个方法来将 HTTP 请求传递进去,从而 Action 类可以代理生成 HTTP 响应的任务。ActionMapping 则完成了视图映射功能,它被实现为一个 ActionForward 类。这样就能够获取匹配的间接视图句柄,并用这个句柄来分派合适的视图。

起总控、分发作用的应用控制器模式可以应用于目前大行其道的 B/S 结构,也可以应用于银行传统业务在数据集中环境下使用的三层 C/S 结构。"农行集中式信贷管理系统"在信贷数据省域集中环境下使用三层 C/S 结构:Windows 操作系统下 Delphi 编写的客户端/建立在 IBM CICS 中间件基础上的应用服务器/Sybase 数据库服务器,总控就使用了应用控制器模式。总控统一进行授权和验证、交易分发、事务提交管理、记录日志,具体的交易只实现业务逻辑,不用关心这些底层服务。

2.3 服务定位器 (Service Locator) 模式

服务定位器实现、封装对服务/组件的寻址。服务定位器能够隐藏寻址机制的实现细节,封装这一机制对不同实现的依赖。

服务定位器模式具有以下优点:封装了服务寻址、创建过程的复杂性,并对客户端隐藏了这种复杂性;为客户端提供了统一的服务访问方式。

许多要将数据进行持久化的模块都依赖于标准的 DataSource 对象来获取访问后台存储系统的方法。后台的存储系统通常是 JDBC 数据库,但是一个 DataSource 对象可以连接上任何其他类型的存储系统。ActionServlet 中保持着在 Struts 配置文件中定义的各个 DataSource 对象的键值(逻辑名)。因此,其他的对象可以通过使用该逻辑名来获取一个 DataSource 对象,而不用了解如何其他的实现细节。当 DataSource 的物理路径发生变化时,保持逻辑名不变,只需修改 Struts 配置文件即可。以上过程就是服务定位器模式的具体应用。使用服务定位器模式使得应用在开发、生产数据库之间的移植、部署带来了极大的便利。

我们可以设想发明一种多功能定位仪,把现有的定位仪功能集成起来。这种定位仪能够使用美国的全球定位系统(GPS)、俄罗斯的全球导航卫星系统(GLONASS)、欧洲的伽利略系统(Galileo)和中国的"北斗"导航卫星系统(COMPASS)。当某种定位系统的信号出现故障或被关闭,定位仪可以无缝切换到另一种定位系统。只要定位仪能够提供精确定位,用户不需要知道使用的究竟是哪一种定位系统、它的工作原理等细节。这种多功能定位仪可以看作是服务定位器模式的假想应用。

2.4 视图助手(View Helper)模式

视图助手把用于表现格式的代码和其它业务逻辑分离开。它要求助手组件来封装与初始化内容获取、验证、模型的转换和格式化相关的逻辑。

视图助手模式具有以下优点:明确了应用系统各部分的分隔,增进了可重用性和可维护性;明确了开发团队中的角色分工;简化了测试。

Struts 中的 JavaBean 使用了视图助手模式。当 HTML 的表单通过 HTTP 进行提交时,其中的每一个数据都被转换成了文本格式。Web 服务器收到的表单元素变成了名字/值对。ActionForm 中包含了要改变系统状态的数据。当 ActionForm 的某个属性和请求中的某个参数名一致时,Struts 将 HTTP 请求中的参数值赋给该属性。这样,进入运行系统的数据都是被某个 JavaBean 封装了。因此,程序员就可以专心关注 Jav-

aBean,而将和 HTTP 请求打交道的工作交给 Struts 框架结构。

在相反的方向上,使用 JavaBean 将业务数据传递给视图层的模式也使用了视图助手模式。

2.5 会话门面(Session Facade)模式

会话门面封装业务层组件,对远程客户端暴露粗粒度服务。客户端不用直接访问业务组件,而是访问会话门面。

会话门面模式具有以下优点:减少了各层次之间的耦合;明确的层次划分增进了灵活性和可维护性;降低了复杂度;集中了安全管理和事务控制。

模式层和应用程序其它部分的交换可能是相当复杂的。如果将这些细节信息封装到一个简单的对象中,并且该对象仅仅暴露出一个简单的接口,则可以使得一个应用程序更加容易编写和维护。会话门面像负责招商引资的服务中心,给客户端提供了"一站式服务"。

在现实世界中,我们可以观察到相似的场景。中关村科技园区管委会和海淀区政府共同组建了中关村科技园区服务中心——"一站式"办公服务大厅,主要为企业提供注册登记、专项审批和社会公共事务服务事项。服务中心目前有北京市和海淀区共 23 个单位 28 个部门合署办公。中心还引进了代理服务站、公证处、银行、会计师、税务师事务所等 10 个中介机构,形成了全方位的服务体系。企业和个人就可以在服务大厅获取各种服务,而不用在全市各个机构之间跑来跑去。这是政府在公共服务职能方面提供会话门面模式服务的典型案例。

Struts 中的 Action 类使用了会话门面模式。Struts 将服务的细节信息封装到 Action 类中。一个 Action 类具有一个定义良好的接口以及一些通用的任务。Action 类主要是一个白盒类,提供的核心责任是:访问业务层;为表现层准备数据对象;处理业务层和表现层之间的任何错误。Action 类抽象了后台业务对象的交互过程,并提供了一个服务层,在其中仅仅暴露自己需要的接口。

Struts 在消息资源模块中同样也使用了会话门面模式。农行外网门户网站(www. abchina. com)有简体中文、繁体中文、英语三个语言版本。后台的内容管理系统将常见的消息和标签分组存放在资源文件中,这样就可以对它们集中检查和修改。模块仅仅通过一个键值来查询一个消息和标签。消息和标签需要在不同语言、字符集之间移植时,只需修改资源文件即可。

2.6 传输对象(Transfer Object)模式

传输对象,也叫做值对象。它通过传递大粒度的数据视图来有效地进行小粒度的数据通讯。一个值对象常常将一系列相关的属性合并到一个组中,以便它们被序列化并且一次性地发送到远程服务器上。这通常用于分布式应用程序。

传输对象模式具有以下优点:简化了远程对象和接口;减少了代码重复;降低了网络负载。

Struts 中的 ActionForm 使用了传输对象模式。和传统的传输对象不同的是,ActionForm 的属性是可以改变的。它一次性地从一个 HTML 表单中获取需要的全部数据,因此这些数据可以一次性地进行校验,然后或者发送回客户进行错误处理,或者向后发送给 Action 类进行处理。

在使用校验的时候,可能会产生多个错误,校验方法需要将错误信息传回给输入表存放多个相关的错误信息。同样,ActionErrors 类的增强版 ActionMessages 类也是一个传输对象。

我们在网站注册用户时,需要遵守一定的校验信息。例如,用户 ID 不能重复,必输项必须填写,密码强度要求等。我们希望系统返回错误信息时,最好一次全部告知,而不用反复多次。使用传输对象模式,可以很好地解决这个问题。

打个比方,客运火车在一条线路上行驶,带动的不是一节客车车厢,而是十几节性质相同的客车车厢,这可以看成一种传输对象模式。

2.7 传输对象组装器(Value Object Assemble)模式

传输对象组装器:顾名思义,就是以复合传输对象的形式构建应用模型。传输对象组装器从各种不同的业务组件和业务服务中聚合多个传输对象,并且最后把复合传输对象返回给客户端。使用其他的值对象来构造一个值对象,因此这些数据可以作为一个整体进行处理。这种聚合的方法用于传输对象,和下面讲到的在表现层使用"聚合"的复合视图模式相类似。

传输对象组装器模式具有以下优点:减少了客户端和应用程序模型之间的耦合;提高了系统的网络性能;提高了客户端性能。

Struts 中的 ActionForm 使用了传输对象组装器模式。它可以使用内嵌的 JavaBean。这样程序员就可以将自己需要的属性存放在不同的 JavaBean 中，而这些 JavaBean 也被模式层的不同部分所使用。

还是用客运火车的比喻。客运火车不仅带动十几节性质相同的客车车厢，还有性质不同的餐车车厢、邮件车厢。这可以看成一种传输对象组装器模式。

2.8 复合视图（Composite View）模式

复合视图使用多个原子化的子视图构成。整个模板中的每个子视图都可以动态地纳入整个整体，页面布局的管理可以独立于页面的内容。

复合视图模式具有以下优点：增进了模块化和重用；添加基于角色或安全策略的访问控制；提高了可维护性。

Struts 中的模板标签使用了复合视图模式，并利用一些标准页面来改造一个 JSP 页面。Tiles 是表示层建筑构件。Definition 存储了一组 attribute，把屏幕描述分离成一个具有独立标志的对象。声明一个基础屏幕 Definition，然后从这个基础 Definition 继承，创建其他 Definition，使之成为扩展类。如果改变了基础屏幕 Definition，继承自它的 Definition 都同样改变，这样就把继承和封装的面向对象原则引入了动态页面中。

3 模式之间的关系

最后我们简单讨论一下模式之间的关系：

有些模式经常会被绑在一起使用，例如，前端控制器通常会使用应用控制器。有些模式是可替代的，例如，一些控制逻辑（例如日志和调试信息）既可以由拦截过滤器也可以由前端控制器实现，但这两种模式实际上相辅相成，可以协同使用。数据库连接池技术既可以认为是拦截过滤器，也可以认为是前端控制器。数据库连接池拦截、控制了数据库连接的建立和释放，极大地提高了效率。

模式不是一个个孤立的个体，它们之间是分层和协作的。每个模式都嵌入到粒度更大的模式里，被同样粒度的模式环绕，并且还有更小粒度的模式嵌入在它的内部。例如，粗粒度的服务到工作者模式实际是一个宏模式，它是由细粒度的前端控制器、应用控制器和视图助手一起构成的，以实现集中控制、请求处理和视图创建的功能。Struts 中的 8 种 J2EE 核心模式既有分工，又互相协作，构成了 Struts 这个优秀的 Web 应用程序框架。

参考文献

1 《实战 STRUTS》机械工业出版社，2005 年 5 月第 1 版.

2 《设计模式：可复用面向对象软件的基础》机械工业出版社，2005 年 6 月第 1 版.

3 http://hillside.net/patterns/DPBook/DPBook.html

4 《J2EE 核心模式》机械工业出版社，2005 年 3 月第 1 版.

5 http://www.corej2eepatterns.com

更 正 声 明

由于本刊编排、校对工作的失误，将《计算机系统应用》杂志 2007 年第 11 期第 36 页文章《基于 PMI 的 ERP 系统安全基础设施研究》的唯一作者钟文龙前面错加蒋泰。本文确属钟文龙一人所作。

特此更正声明。并向钟文龙先生深表歉意！还望广大读者谅解！

《计算机系统应用》编辑部

2007 年 11 月 14 日