摘要

随着社会信息化的推进，信息量越来越庞大。如何保证这些信息被充分利用，实现信息价值的最大化，是一个非常迫切需要解决的问题。信息系统从数据管理、文档管理到内容管理的转变是解决这个问题的一条途径。

在上述背景下，本文首先介绍了有关内容管理的技术背景和相关理论，然后以作者负责设计和开发的 web 信息系统为背景，论述了运用内容管理的理论和相关技术、采用 J2EE 多层应用框架构建一个内容可管理、可定制的信息发布系统的详细过程。

文章详细介绍了该系统的主要设计目标以及围绕这些目标的具体设计思路，重点讨论了若干系统关键机制和相应的实现方案，并给出了其实现方法和过程。

关键词： 内容管理，模板，XML， J2EE

作　　者： 李国柱
指导老师： 吕　强

The Design and Implementation of web Information System Based on Content Management

ABSTRACT

With the development of Internet economy, the scale of data quantity has been much huger. How to take the full advantage of data and maximize the value of those digital properties is a very important problem and should be solved properly. So it is a good way that Information system is transformed from data management or document management to content management.

Under the background we mentioned above, we firstly introduce the background and fundamental t heory of Content Management. Secondly we take an actual information System of a University as an example to describe how these technologies have been used into the implement of a customizable Information system based on content management.

This paper discuss in detail on several crucial mechanisms implemented for the goals of designing the system, and gives out relevant methods and process of the implementation.

KEYWORDS: Content Management, Template, XML, J2EE

Written by Li Guozhu
Supervised by Lv Qiang

苏州大学学位论文独创性声明及使用授权的声明

学位论文独创性声明

本人郑重声明：所提交的学位论文是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含其他个人或集体已经发表或撰写过的研究成果，也不含为获得苏州大学或其它教育机构的学位证书而使用过的材料。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本人承担本声明的法律责任。

研究生签名：李国柱 日　期：04.4.30

学位论文使用授权声明

苏州大学、中国科学技术信息研究所、国家图书馆、清华大学论文合作部、中国社科院文献信息情报中心有权保留本人所送交学位论文的复印件和电子文档，可以采用影印、缩印或其他复制手段保存论文。本人电子文档的内容和纸质论文的内容相一致。除在保密期内的保密论文外，允许论文被查阅和借阅，可以公布（包括刊登）论文的全部或部分内容。论文的公布（包括刊登）授权苏州大学学位办办理。

研究生签名：李国柱 日　期：04.4.30

导 师 签 名：[signature] 日　期：04.4.30

第1章 绪论

1.1 引言

随着社会信息化的推进，各类企业、组织的业务活动纷纷加速向互联网渗透。企业通过互联网开展了多种业务，并且业务的种类和规模还在不断扩大之中。同时信息系统的规模也越来越大，信息量呈爆炸式增长，而且信息的类型和存储方式也五花八门，这使得系统的信息维护更加困难，迫切需要一种更加高效、平滑的管理方式来解决这个问题。另外，企业对通过互联网来收集、处理和传递数据的依赖性越来越强，不断有新的基于 web 方式的系统投入应用。同时，系统的拥有者希望自己的系统有很好的灵活性和扩展性，可以根据自己的需要进行定制。以上这些都使得传统的信息系统解决方案难以满足需要。信息系统需要实现从数据管理、文档管理到内容管理的转变，以满足日益复杂的 web 应用和急剧膨胀的信息量的需求。

1.2 课题的内容及意义

在这种形势下，我们为某大学开发信息发布系统时，引入了内容管理思想。内容管理作为一种组件级的信息管理模式，与传统的数据管理和文档管理不同，它把信息进行抽象和结构化，将信息划分成更小的单元（即内容组件）来进行管理。并且引入了元数据对信息进行描述。这样可以提高系统中信息的清晰度和透明度，并可实现基于内容的检索、加工、传递，为建立高层次、多样化、个性化的应用打下坚实的基础。同时，建立在内容管理基础上的信息发布可以保证发布过程的平滑和高效。另外，如果系统能提供给用户自己去抽象新内容的手段，那么将会大大增强系统的通用性。本课题涉及的系统正是按照这个想法来构建的。

要说明的是，我们没有去实现一个完整的内容管理系统，而是在内容组件化、结构化的基础上，提供从内容收集、创建到最终发布这一过

程所需要的基本功能。同时也力求使构建的系统能够具备以下两方面特征：

- 扩展性强

用户可以轻松地集成新的应用，从而可以不断扩展系统的功能，满足用户的需要。

- 通用性强

系统从功能到发布界面都给用户极大的定制自由，从而可以使用户根据自己的需要部署一个量身定做的系统。

信息系统中存有大量的内容信息，这些内容除了向外发布外，可能需要进一步的处理才能满足某些应用需求。本系统设计了较完善的模块部署规范，提供了便利的方式将符合部署规范的应用模块集成进来。这样在不改动系统原有代码的情况下，就可以集成新的应用到系统中。而且只要遵循了模块的部署规范，第三方开发的模块也可集成进来，从而使得系统的灵活性进一步增强。

另外，用户可以根据需要对系统进行功能上和界面上的定制。这就简化了建立一个符合自己需要系统的过程。用户可以根据情况定制系统发布界面，并且可以在系统功能上进行适度取舍。

系统所要管理的内容信息不仅仅包含那些已经结构化的数据，还有许多数据是半结构化和非结构化的。为了能将这部分异质的内容数据也管理起来，我们在系统异质数据集成方面作了一些尝试。在内容存储方面，我们尽量屏蔽了对数据操作的细节，提供了面向上层应用对象的统一数据存储接口。

本文详细介绍了某大学 web 信息发布系统的设计和实现。该系统是一个开放的基于特定应用，内容可管理的信息发布系统。以 J2EE 平台为基础，通过采用面向 WEB 应用程序开发的 struts 框架，以及 XML、XSLT 技术来建立一套模板机制，使得系统具有以下一些特点：

- 信息的组织是基于内容管理的

即按照内容管理的思想对信息进行抽象和结构化，并合理划分成内

容组件来进行管理。同时用元数据对信息进行描述，并对内容进行全文索引，以实现更高层次的信息服务，而不仅仅是简单的发布。

● 信息的逻辑类型是可扩展的

提供给用户针对某种信息类型自己创建模板的手段。使得用户可以自己对新的信息逻辑类型进行抽象。从而，创建自己的内容组件，集成新类型的数据。

● 系统的发布界面是可定制的

即可以根据需要对系统的界面布局和信息的显示方式进行设置，而不需要修改系统代码。

● 系统是可扩展的

系统管理的信息是经过组织和描述的，一旦有建立在这些信息之上的新的应用出现，系统允许在不修改任何代码的情况下将这些应用作为功能模块集成进来。

● 系统具备了一定的数据集成能力。

1.3 内容管理的研究和发展现状

由于内容管理思想的引入能够显著地提高企业、组织在信息时代的生存能力，内容管理得到人们的广泛重视。所谓内容是任何类型的数字信息的结合体，可以是文本、图形图像、web 页面、业务文档、数据库表单、视频和声音文件等。针对内容的管理就是施加在“内容”对象上的一系列处理过程，包括收集、确认、批准、整理、定位、转换、分发、更新和存档等，目的是为了使“内容”能够在正确的时间、以正确的形式传递到正确的地点和人[1]。内容管理是一个新兴的市场，其方案重点在于解决各种非结构化或半结构化数字资源的采集、管理、利用、传递和增值，并能有机集成到结构化数据的商业智能(BI)环境中，如 ERP,CRM 等。电子商务和 XML 是内容管理市场发展的源动力，内容管理解决方案的终极目标是实现内容价值链的最优化[1]。目前，与内容管理相关的基础技术和开发工具已日臻成熟，但内容管理本身还未形成统一的业界标准。

目前，已经有很多国内外公司推出了自己的内容管理产品。国外的公司如 Interwoven、Broadvision、Vignette 和 Documentum 等国际大厂商，它们进入这一领域的时间最早。 还有就是诸如 Oracle、Sybase、IBM，Microsoft 这样的拥有数据库技术的厂商。国外厂商固然在内容管理的影响力、技术、资金等方面具有一定优势，但是其内容管理产品存在面向的应用规模较大、功能大而全、价格昂贵、技术应用难度高和本地化支持不够等问题，因而目前较难适应中国的具体情况。国内内容管理厂商具有代表性的包括易宝北信、国信贝斯、联想等。但就目前而言，国内产品很多只是一些相关工具的组合。国内用户目前最需要的是针对用户的具体情况提供量身订制的解决方案。

1.4 论文结构

本文的内容是以如下方式组织的：

第一章绪论对论文内容作了概括性介绍。介绍了论文的内容、意义以及内容管理的研究和发展现状。

第二章技术背景对本文所要用到的技术作了详细介绍，指出了系统的开发和这些技术的关系。

第三章对系统的框架和一些重要机制的设计给予了详细说明。

第四章介绍了系统的一些重点难点问题的实现方法，并对系统某些部分的运行方式给出了说明。

第五章以实际的例子来说明系统是如何使用的。

第六章对系统的实现进行了回顾，并对实现过程进行了总结，最后对系统的未来发展进行了展望。

第2章 技术背景

2.1 内容管理

随着社会信息化的推进，内容信息量呈急速膨胀趋势。良好的内容管理解决方案是现代企业、组织实施信息化战略的一个重要保障。

2.1.1 什么是内容

要明白内容管理是什么首先要说明内容这一概念。内容是记录在介质上的意义，这种意义与记录它的媒体无关。

意义是人们对事物的共同理解，是包含在某些思想和度量中的。意义有四个基本特征:定义(definition)、状态(state)、上下文环境(context)和行为（behavior）[2]。

内容的成熟需要经过三个阶段，分别是数据、信息和知识。数据是值的简单描述或机械度量。数据本身难以解释，它必须借助相关的环境和关系，只有在特定的上下文环境中或是与其它数据的联系中才有意义。如果把数据置于具体的上下文环境中并和其它数据一起经过组织，数据就成熟为信息。信息不是孤立存在的，信息与信息之间有复杂的关系。当把信息加以组织然后以某种方式为了特定的目的发布出来后，信息就成为了内容[2]。

2.1.2 什么是内容管理

对于内容管理，目前业界还没有一个统一的定义。一般认为内容管理是：协助组织和个人，借助信息技术，实现内容的创建、储存、分享、应用和更新，并在企业个人、组织、业务、战略等诸方面产生价值的过程。内容管理最大的特点在于其管理的是内容而不是数据。因此内容管理与数据管理相比有很大的不同。

2.1.3 内容管理的其它相关概念

- 内容域

是指所要获取、管理、发布的信息的范围。内容域通常与内容管理系统的目标有直接联系。合理确定内容域是内容管理的第一步[3]。

- 内容组件

内容域确定下来后，就需要确定内容的类型，然后根据抽象好的模型将信息划分成一个个易于管理、方便使用的小单元，即内容组件。组件的创建、维护、分发可以自动化的实现。每种组件都有自己的属性，每个组件都是独立的，不局限于特定的上下文中，组件是内容管理的最小单元。任何内容都是由若干个内容组件组成，内容的创建、删除、维护都是以组件为基本单位来进行的。组件的划分并不是任意的，必须按照一定的原则来进行，这种划分也是与内容的具体应用领域相关的。正确的组件划分可以带来高效率的组件重用。

- 元数据

元数据就是关于数据的数据。在内容管理中，元数据给计算机提供了处理内容所需要的信息，从而使得计算机可以根据元数据进行内容的自动化处理。在内容管理中所涉及到的元数据主要包含以下几类：

- 拆分性元数据

计算机可以根据这类数据正确一致地把内容拆分成内容组件。

- 访问性元数据

访问性元数据包含了内容的位置信息。

- 管理性元数据

管理性元数据主要是指为了方便内容管理而附加的一些信息。

- 集成性元数据

这部分元数据规定了内容与内容继承的规则和方式。除了包括内容组件组合成复杂的内容实体的集成外还包括不同类型内容的集成，如关系型数据和非结构化文档的集成。

2.1.4 内容管理的应用体系结构

实施内容管理的目的就是为了使信息实现内容级的管理。这需要一个完善的应用体系结构支撑。和互联网的迅速发展一样，内容管理也是一个不断发展的技术领域，目前在这一领域已有很多产品面市，但这些产品分别有自己的侧重面。各厂商对内容管理究竟该包含那些功能还没有达成完全的共识。

一般认为，一个内容管理系统至少要包含四大部分：

● 内容收集系统（Collection System）

进行收集、获取、分发、编辑、整合及转换(如转换为 XML)内容等工作，并可加入元数据(metadata)以对内容组件进行定义及搜寻。

● 管理系统（Management System）

负责组件、内容及发布模板的存取管理，并可记录内容的版本、工作流程的状态、权限的设定及更新处理等等，也可说是保障内容从收集、创建到发布这一过程准确高效的进行的管理系统。

● 发布系统（Publishing System）

负责将内容从数据库中快速且自动的按照所建立的发布模板送至各种出版媒体上，如 web、电子出版品、PDA、WAP、印刷品、XML 数据交换等等。

● 工作流系统（workflow System）

确保整个内容从收集、储存及发布的整个流程可以有效及正确地运行的定制系统。

在以上几部分的基础上作进一步细化，一般认为内容管理的功能应该有：

● 内容采集和创建（Content Acquisition and Creation）

各种数据、信息、文档和程序的获取，创建内容的协作工具，如网络搜索机器人，文档和网页制作工具（包括协作创建），数据格式标准化和转换(Metadata and XML)等。

● 存贮和管理(Storage and Management)

高效、安全存贮和管理各种形式的内容。

● 版本控制和回滚(Version Control , tracking and rollback)

多版本控制，跟踪和回滚等功能。

● 先进工作流(Advanced Work flow management)

可以用户自定义的流程和基于角色的流程控制，审批流程等。

● 模板设计和管理(Template Design and Management)

可视化的内容模板设计，使得内容的表现和内容本身分开，兼容 XML。

● 内容复制(Content Replication)

包括多服务器镜像，跨平台内容自动同步更新等。

● 出版(Publishing)

动态和静态网页生成，和分发以及个性化密切相关，能够和其他外部数据很好的集成(Syndication)。

● 分发(Delivery & Distribution)

包括 Caching，负载平衡，流媒体的 Delivery 等，还自动推动等功能。

● 个性化(Personalization)

个性化是内容管理区别于传统的文档管理等类似系统的重要特点，个性化包括用户控制的个性化（用户喜好），站点控制的个性化（用户行为分析）和数据控制的个性化（内容相关性）。

● 自动归类(Categorization)

自动归类能大大提高在海量信息环境下，用户的检索和导航效果。

● 检索和导航(Search and Navigation)

先进智能化知识检索技术，基于内容整合(如相关新闻、相似性检索)的启发式信息导航，自然语言查询和对话，动态摘要生成，数字特征的提取和检索技术，跨语言检索和机器翻译（多语言应用环境）。

● 安全控制管理(Security and Access Rights)

除了一般性的数据库安全控制机制外，还包括加密，拷贝和传播限制，这些功能在电子商务环境中非常重要。

● 用户管理(User Management)

用户管理是个性化和电子商务的基础。[1]

2.2 XML 技术

XML（eXtensible Markup Language）是一种具有数据描述功能、高度结构性及可验证性的语言。和 HTML 一样，XML 同样适用了标记与属性，但和 HTML 最大的不同点则在于 XML 的标记与属性允许用户自定义，并可以依照所定义的标记与属性的语法来开发应用程序。在 XML 文件中，可以使用标记来描述数据，或配合属性来辅助描述数据。因此，XML 很适合用于作为对象或标准的描述语言。并且可以借助验证规则来规范一个 XML 文件的内容与结构，所以 XML 又很适合用做数据交换格式[4]。有许多 XML 的相关技术，如：XSL，验证规则（DTD 与 XML schema），DOM，SAX，xLink，XPath，XPointer，JDOM 等。这些技术都可以应用在基于内容管理的系统实现上。

2.3 J2EE 开发平台

J2EE 是一种利用 Java 2 平台来简化企业解决方案的开发、部署、管理等相关复杂问题的体系结构。J2EE 不仅具有 J2SE 中的许多优点，例如，"编写一次、随处运行"的特性、方便存取数据库的 JDBC API、CORBA 技术以及能够在 Internet 应用中保护数据的安全模式等等，同时还提供了对 EJB（Enterprise JavaBeans）、Java Servlet API、JSP（Java Server Pages）以及 XML 技术的全面支持。其最终目的就是成为一个能够使企业开发者大幅缩短投放市场时间的体系结构[5]。

J2EE 体系结构提供中间层集成框架用来满足需要高可用性、高可靠性以及可扩展性的应用需求。通过提供统一的开发平台，J2EE 降低了开发多层应用的费用和复杂性，同时提供对现有应用程序集成的强有力支持，完全支持 Enterprise JavaBeans，有良好的向导支持打包和部署应用，添加目录支持，增强了安全机制，提高了性能。J2EE 为搭建具有可

伸缩性、灵活性、易维护性的商务系统提供了良好的机制。

2.4 web 信息发布系统框架

web 信息发布系统是以 web 方式对信息进行创建、采集、加工等维护工作，并以适当的方式将信息发布给适当的受众的信息系统。web 信息发布系统的价值在于信息的加工和传播能力。作为信息的管理者，web 信息发布系统向一定范围内的用户提供各种信息服务，同时，信息的管理和信息服务的提供都是基于 web 的。结合 J2EE 平台对 web 应用的支持，web 信息发布系统一般采用多层 web 应用框架，而 J2EE 为系统提供了容器平台。多层 web 应用框架分为以下几层：

- 表现层

是系统界面的表现层。通常是 html/JSP 页面或 applet。

- 应用层

是系统的控制层，通常是以 servlet 来实现。

- 服务层

提供商业流程和逻辑运算，由 Session bean 来完成。

- 业务对象层

管理业务数据对象，即所有的实体 Bean。

- 持久层

即数据库或文件系统层。

其中，表现层和应用层由 web 容器来支持。服务层、业务对象层和持久层由 EJB 容器来支持，如多层 web 开发框架图 2-1 所示。

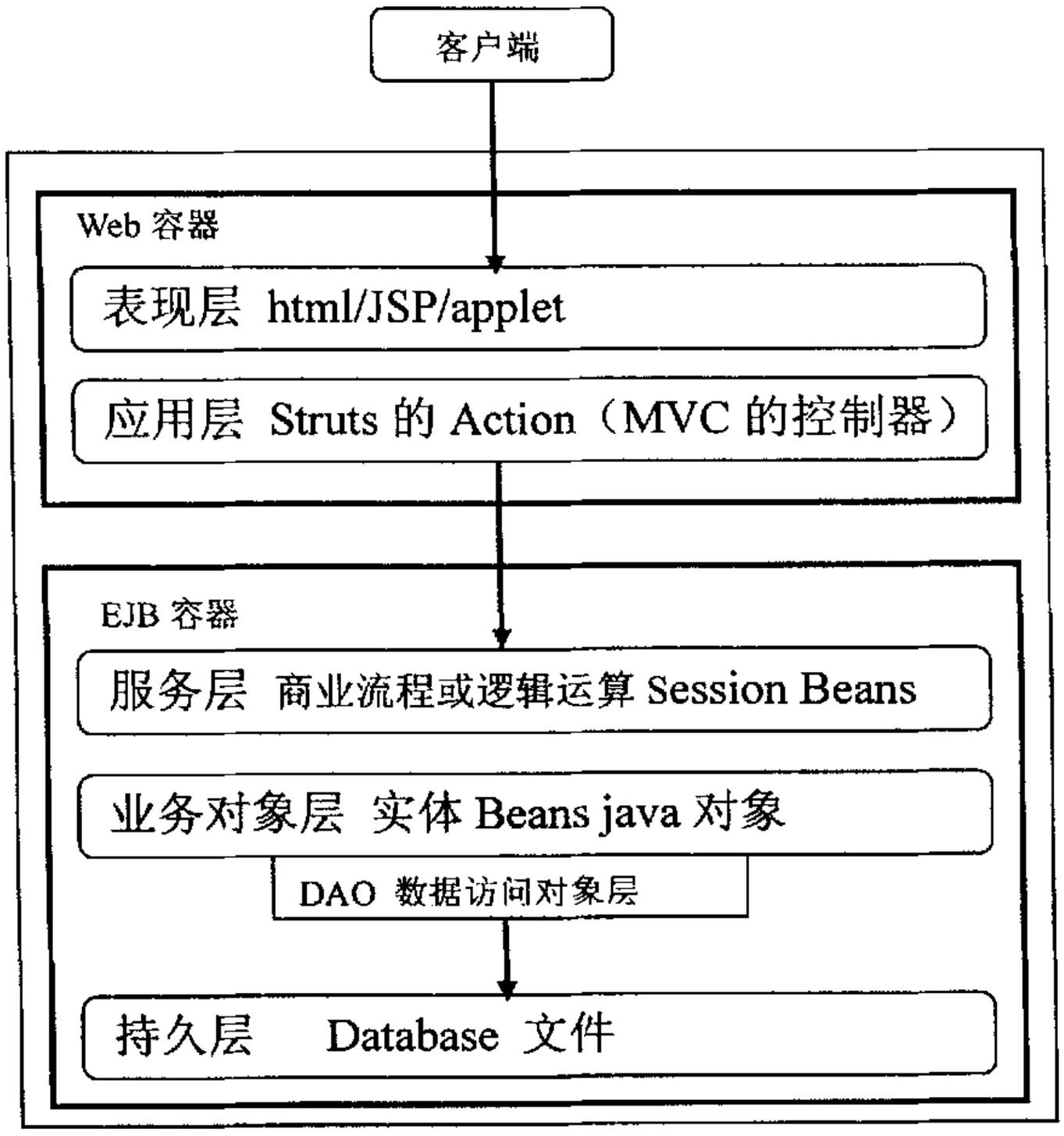


图 2-1 多层 web 开发框架

在客户端发出请求后，由实现 MVC 模式的 web 层负责处理客户的表单信息，通过事件触发 EJB 层的相应服务，EJB 层组件通过数据访问对象对数据进行存取。经过 EJB 层组件运算和处理，将结果返回 web 层，再送回客户端。

可以看到，系统使用了两种容器，每种容器都使用各种相关的 Java web 开发技术。这些技术包括两类：

- J2EE 各种不同的应用组件（如 Servlet，JSP，EJB），它们构成了应用的主体。
- J2EE 平台提供的应用服务（如 JDBC，JTA），这些服务保证并促进组件的良好运行。

我们在 J2EE 平台基础上，采用多层 web 架构开发了本系统。充分利用了 J2EE 的各种开发技术，为保证系统的可靠性、扩展性打下了坚实的

基础。

2.5 web与EJB接口框架

为了避免web层和EJB层过分耦合，提高web层和EJB层的开发效率，加强J2EE系统的可维护性和可拓展性，需要在web层和EJB层之间建立一个类似接口网关的框架系统，这就是系统中用到的web层和EJB层接口框架。在一般的J2EE开发项目中，控制层中对EJB的访问是直接嵌在Struts Action中的。这样做有很多缺点：

● 加重网络负担

如果在某个Action的一个方法体内分别调用多个不同的Façade类，就类似客户端直接调用实体Bean所面临的问题，大大加重网络传输负担。

● 可维护性差

如果在某个Action中再增加其它方法，这个Action将变得复杂，导致维护性和拓展性降低。

● 与web层过分耦合

造成整个系统严重依赖Struts，如果将来有新的客户端（非HTML等）加入，Struts框架不能再使用，则需要对系统web层进行较大改动。

● 与EJB层过分耦合

由于在Action代码中直接指定了EJB调用代码，而在一个大型系统中，这种情况随时会发生改变，一旦其它小组开发了新的客户EJB组件，就需要直接修改这个Action的代码。

针对这些缺点，web与EJB接口框架的解决思路是，将EJB访问调离Action类，这样在前台表现层和EJB层之间形成了一个新的Proxy层。然后采用EJB Command模式来实现事件的触发机制，使得前台Action可以透过Proxy触发相应的EJB 服务。遵循这样一个思路，该框架实现了web层和EJB层的解耦，通过配置文件，将两者具体Service对应了起来。并且提供了对有状态session bean的支持。

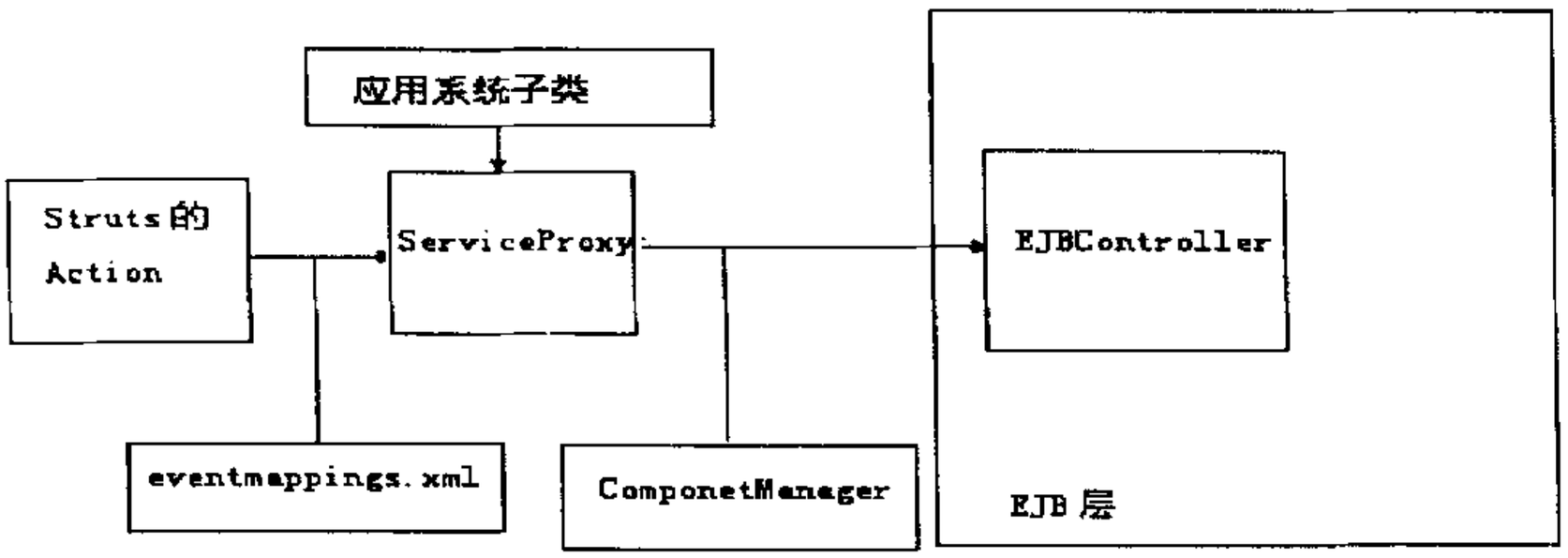


图 2-2 web 与 EJB 接口

从图 2-2 可看出，Struts 的 Action 通过配置文件找到对应的 ServiceProxy，然后由 ServiceProxy 通过 EJBController 接口的具体子类得到与 EJB 层的对应关系。同时，ServiceProxy 启动 ComponentManager 来实现对有状态行为的支持。该控制框架综合使用了 Command 模式、Proxy 模式以及工厂模式等，是一个可重用、可扩展、伸缩性极强的接口框架系统。

2.6 编程设计模式

上面提到，J2EE 通过 API 提供技术与服务的高层抽象，使企业开发得到简化。但是，仅仅知道 J2EE API 是不够的。要设计良好的体系结构，得到高质量的应用程序，就要知道何时如何正确使用 J2EE API。这就是设计模式所关心的问题。设计模式是情境中标准设计问题的重复性解决方案。设计模式可以帮助我们解决应用程序设计阶段的大多数常见问题。如：表示组件、组件内部结构及组件之间的关系，确定组件粒度及适当的交互，定义组件接口等等。而更具体地说，对于 J2EE 平台设计模式来说，可以解决使用 J2EE 服务与技术涉及的常见问题，包括：视图管理，请求处理，服务定位与激活，远程通信与层间通信，组件选择，持久状态、事务与安全性管理，EIS 集成。并且，设计模式还可以帮助我们进行程序的构架设计，使我们能够根据决策的准则进行适当的抽象，适当的一般化，使之更适合于复用，而且更健壮，从而大大提高设计的灵活性，更好的适应将来的改变[6]。在本系统的设计开发过程中，用到的一些比较

重要的J2EE模式有DAO模式、proxy模式、command模式、Façade模式、工厂模式等等。

2.7 Struts

在系统开发中，表现层会涉及很多用户界面的元素的使用，因此比较难以实现重用，但是，有一个宗旨是：不能将功能性的代码与显示性的代码混合在一起，否则，当需要更改页面，或者扩展新功能时，那么会带来很大的修改量，甚至破坏原有系统的稳定性。

因此，需要对表现层进行细化，可以将表现层分三个部分：

- 视图（View）

  负责显示功能。

- 控制器（Controller）

  根据Model处理结果，调节控制视图的输出。

- 业务对象模型（Business Object Model）

  是对真实世界实体的抽象，可以是一些数据、也可以是一些处理对象或事件对象，在本项目中，业务对象就是那些包含状态和行为的Javabeans。

这就是所谓的MVC模式，旨在将表现逻辑和业务逻辑相分离。从而有良好的重用性、可靠性，降低界面维护的成本。而struts就是这样一种基于MVC模式的开发框架。它是Apache Foundation发起的Jakarta开源项目的组成部分。 Struts 框架是结合 JSP、JSP 标签库以及Servlets的MVC模式实现，图2-3是struts实现的MVC流程图。

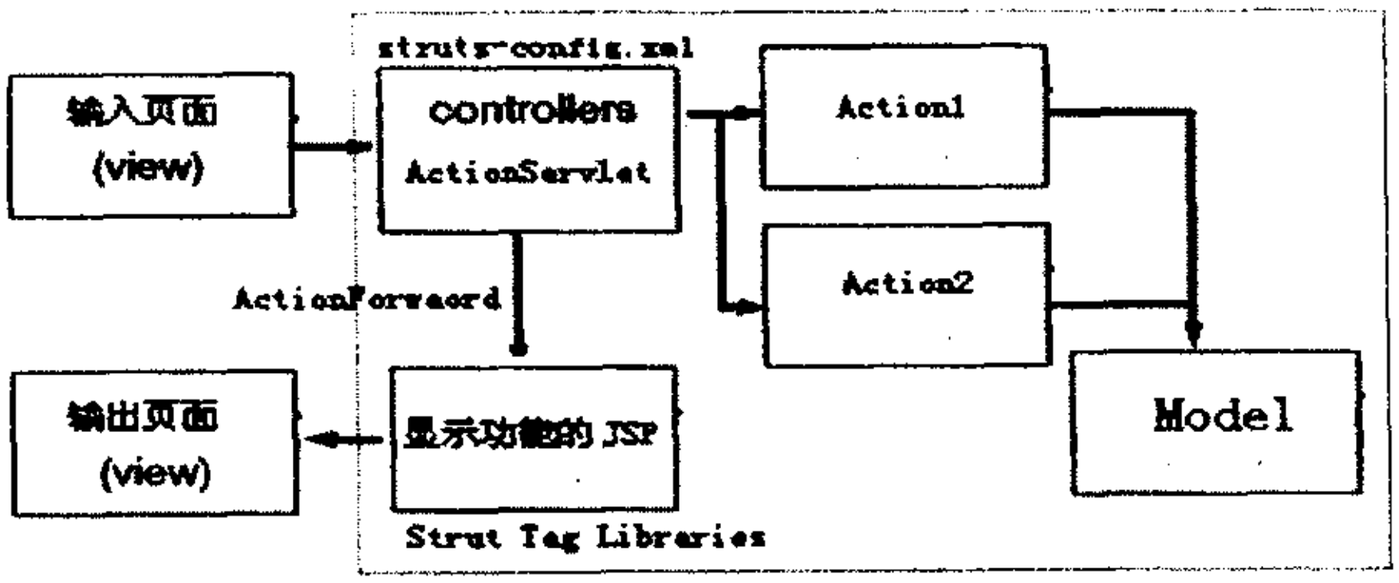


图2-3 struts框架流程图

第3章 系统设计

3.1 需求分析

本系统是为某大学开发的一个基于内容管理的校园 web 信息发布系统。目的是，将各类数据以内容组件的形式管理起来，使得内容能够以更灵活的方式被应用，并且系统提供多种手段保证从数据的创建到发布整个过程的平滑高效。除了有现成的针对学校的常用内容模型外，系统还提供给用户自己抽象并创建内容模型的手段。另外，系统可以在不修改原有代码的情况下，集成新的应用。这些使得系统具有一定的扩展性。此外，系统的外观布局和内容的发布形式也是可定制的。

从功能的角度来看，系统应分为以下几部分：

●内容采集和创建（工作台）

提供内容创建和添加元数据的工具。在这部分除了完成内容创建外，同时实现内容的组件化和描述。这部分是实现信息基于内容管理的基础。

●界面管理

对发布界面的布局和内容的表现形式进行定制。

●模板管理

这是系统对各类模板进行维护的功能模块。它实现对内容逻辑类型的抽象描述，新的内容组件的构造，以及对内容显示方式的定义。这些都通过对模板的操作来实现。特别是可以通过对内容模板的定制来实现内容模型的扩展。

●类别管理

首先是对系统的内容模型进行维护，即对系统内容树进行管理。具体到本项目就是对面向学校建立的内容模型的管理。其次是在已经抽象好的内容模型的基础上，进一步对内容进行逻辑上的分类管理。以便于内容的创建、审核和发布。同时，对工作流程的管理是基于类别的，所以类别管理中还包括了对工作流程的管理。

●存储管理

在这里要完成对数据源的管理，包括数据源的添加、删除。基于内容管理的系统查询机制要比一般的关系型数据库复杂不少。因为系统中还有许多非结构化数据，所以要将元数据查询和全文检索相结合。因为基于内容管理的系统所发布的内容都是以内容组件的形式存在的，在发布的时候要经过动态的组装和转换，所以还需要建立缓存机制来加速这一过程。

●应用模块管理

对集成进来的应用模块进行管理，包括模块文件的导入导出。通过引入应用模块实现应用的扩展。

此外，还有用户管理，版本跟踪与回滚等部分。系统整体功能如图3-1所示。

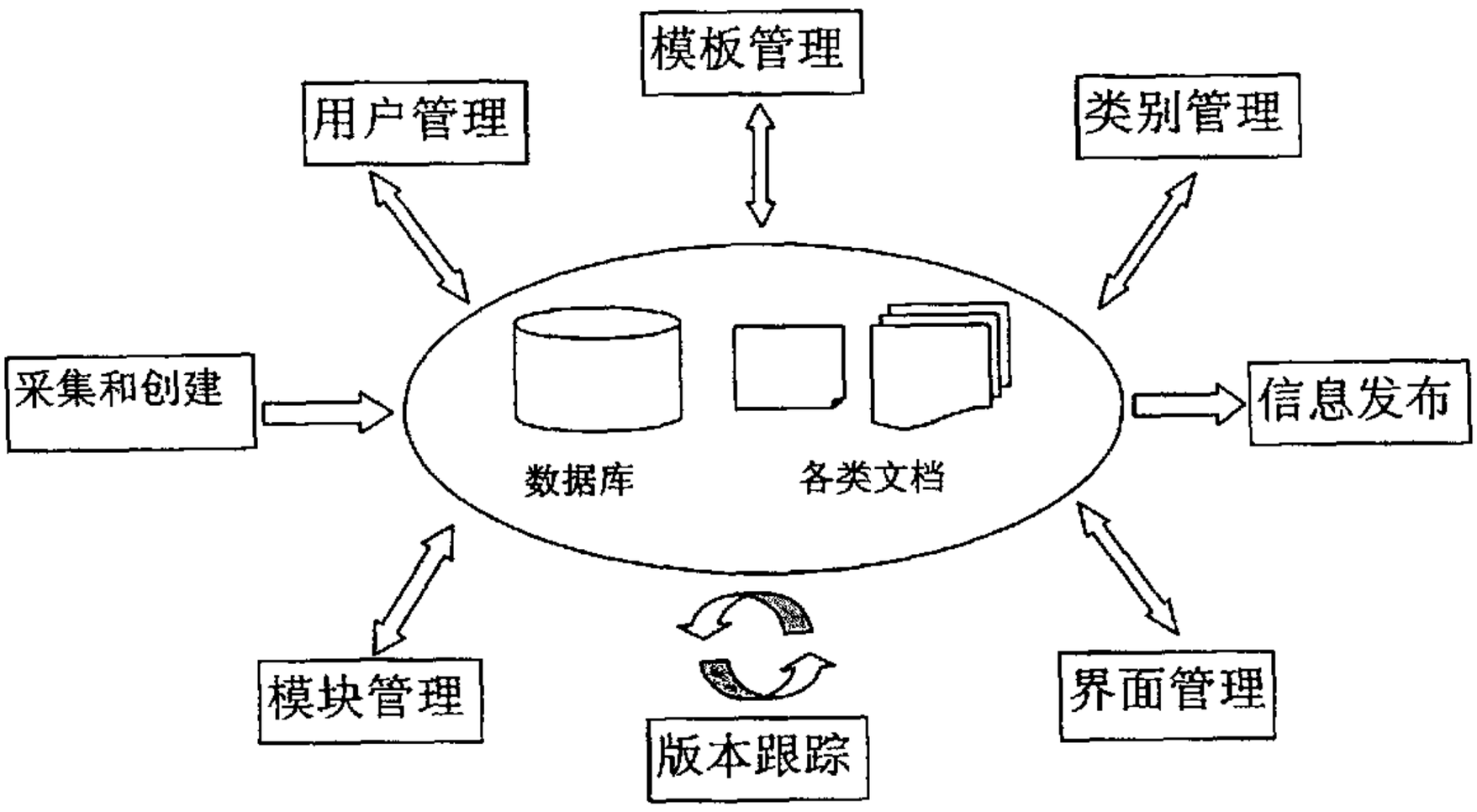


图3-1 系统功能

图3-2是系统最高抽象层次的部分用例分析图，从用例图中可以看到，系统将用户分成三类：

● 系统管理员

管理员用户可以通过各种管理手段完成对系统的维护工作。

● 内容发布者

主要负责各种内容的创建、收集和发布过程，并完成对内容信息的维护。

● 浏览者

即从发布系统获取信息的使用者。

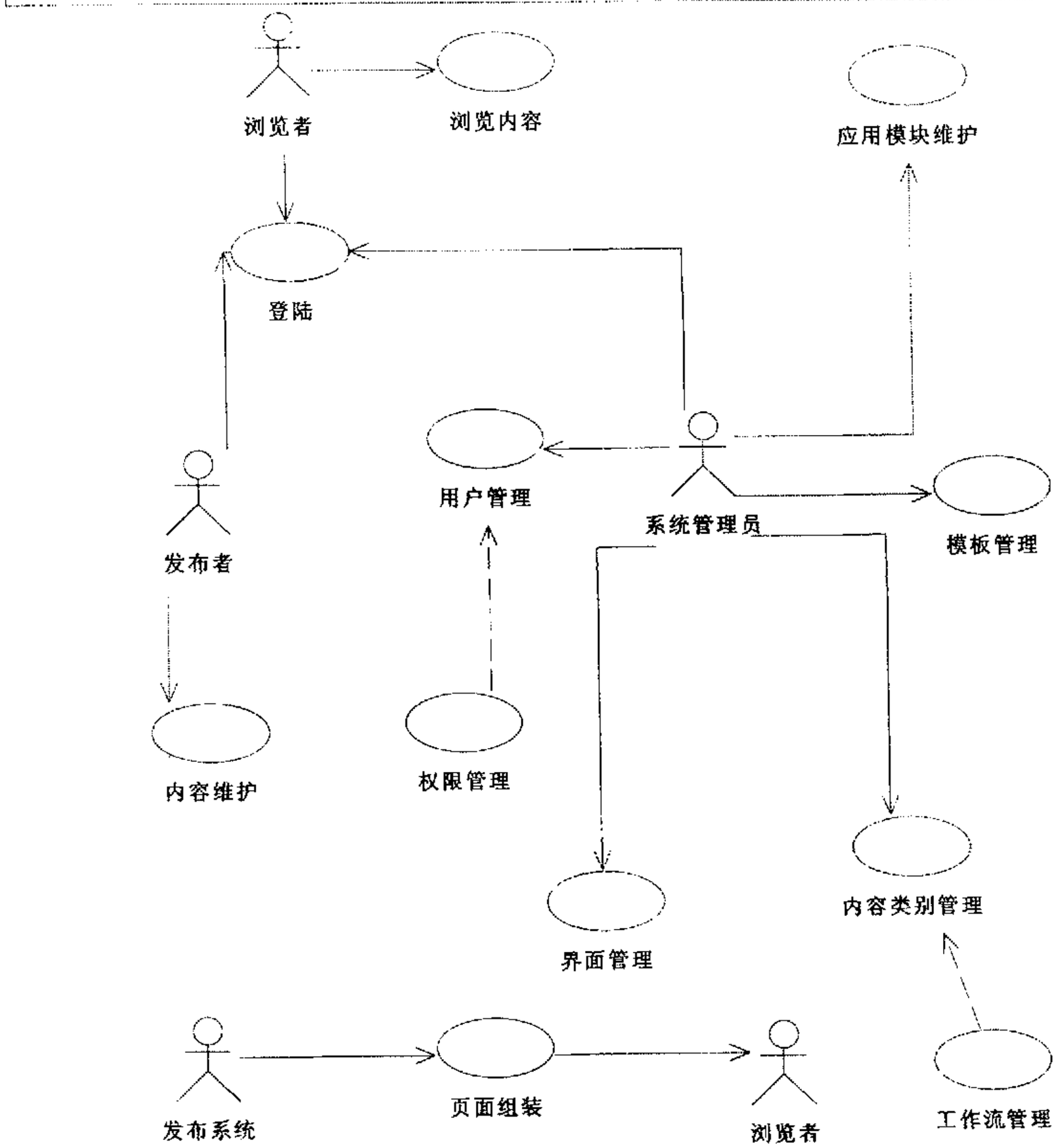


图 3-2 用例分析图

3.2开发环境

我们采用 Jbuilder9 作为系统开发工具，采用 Rational Rose 作为设计工具，同时还采用 Apache 和 Jboss3.0 作为应用服务器，后台数据库采用 oracle9i。

Jbuilder 作为编程工具提供了开发 EJB、web|、XML，以及数据库等各类应用程序的有力手段。籍由 Jbuilder 双向、可视化的设计工具，可以快速开发各种 J2EE 应用程序，并部署至多种应用服务器。

Rational Rose 是强大的系统建模工具，可以在系统的开发过程中帮助开发人员先进行建模，然后开发人员再写代码，从而在一开始就保证系统有合理的结构。利用模型可以更方便地捕获设计缺陷，从而以较低成本修正这些缺陷。

Apache 是由 Apache Group 主导开发的一个开放源码的 web 服务器。因其强大的功能和稳定性而得到广泛的应用。

Oracle 是目前应用最广泛的数据库之一，向业界提供了最佳的性能与最高的可靠性和安全性。

Jboss 是免费的 J2EE 服务器，并且是开放源代码的项目，遵循最新的 J2EE 规范。从 JBoss 项目开始至今，它已经从一个 EJB 容器发展成为一个基于的 J2EE 的一个 web 操作系统（operating system for web），它体现了 J2EE 规范中最新的技术。相比 webshere 和 weblogic 商业服务器，Jboss 也有很多特色和优势。

3.3系统规划与设计

3.3.1 框架设计

根据需求分析，校内各部门的工作人员都应该能以 web 方式，根据不同的权限进行内容管理和系统维护工作，而且校内外的内容浏览者也可以方便的享受到信息服务。系统以图 3-3 所示的方式来部署。

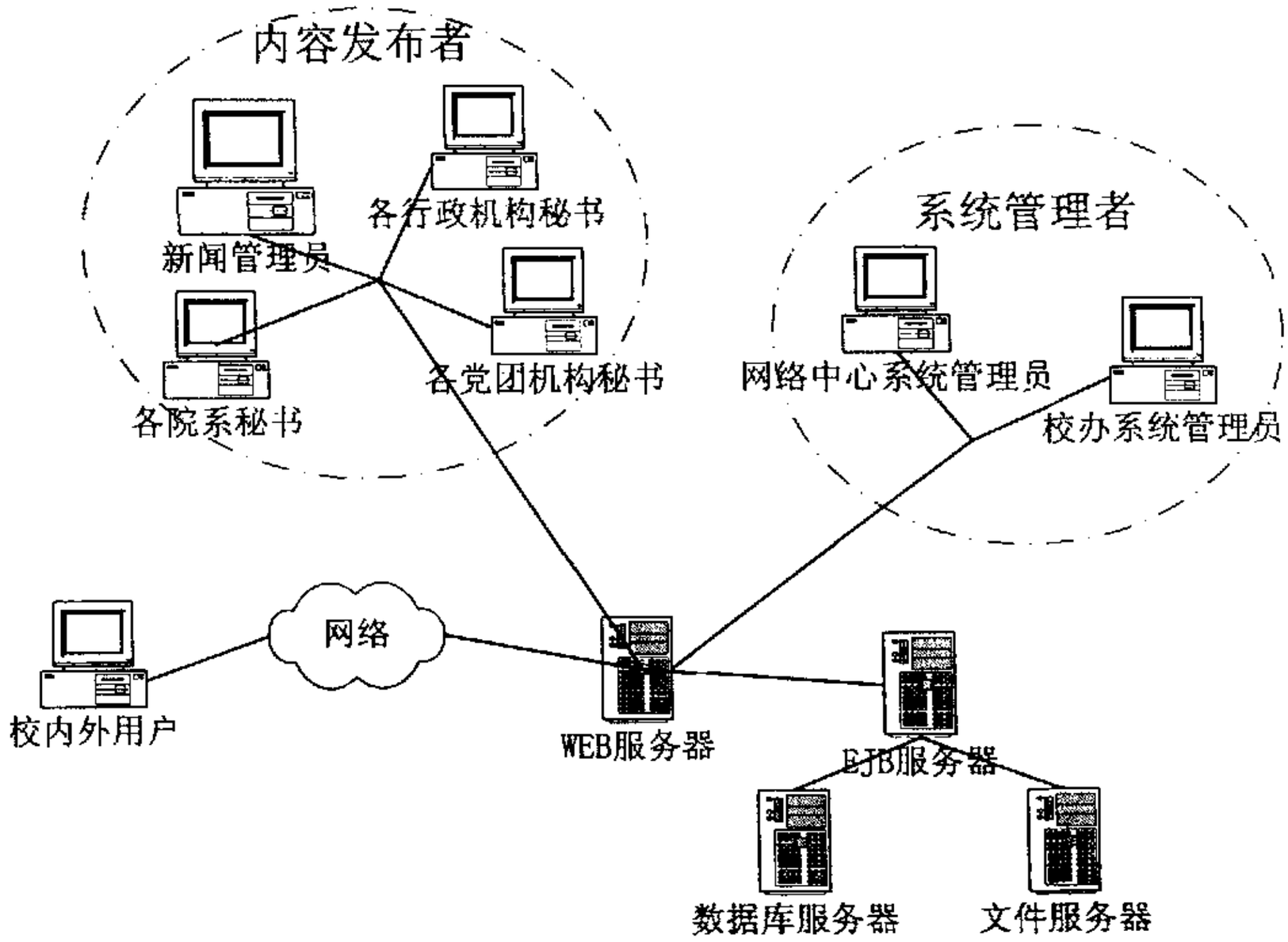


图 3-3 系统网络拓扑结构图

系统的设计遵循了 J2EE 框架。在进行框架设计时，要确保系统各模块的松散耦合性、重用性以及各部分的关系清晰有条理。

我们结合系统需求分析对系统控制层的 Action 进行了划分，其中有一些是最主要的，这些 Action 分别负责处理不同类型的请求：

- WorkplaceAction

  处理对内容的维护请求。

- ViewAction

  处理一般用户对信息的浏览请求。

- CategoryAction

  处理对内容类别的管理请求。

- TemplateAction

  处理对各类模板的维护请求。

- ModuleAction

  处理对应用模块的管理请求。

- InterfaceAction

处理对系统界面的定制请求。

- UserAcion

处理和用户管理有关的所有请求。包括对用户组的管理和用户权限管理。

控制层的 Action 在接到用户的请求后，通过 web 层和 EJB 层的接口访问相应的 session bean，再由 session bean 访问相应的实体 bean，而实体 bean 通过 DAO 来访问数据源，得到数据后返回。Session bean 在得到返回数据后，经过加工处理返回给上一层，经过上一层的进一步加工、组合后返回给用户。图 3-4 是细化的系统功能框架图。

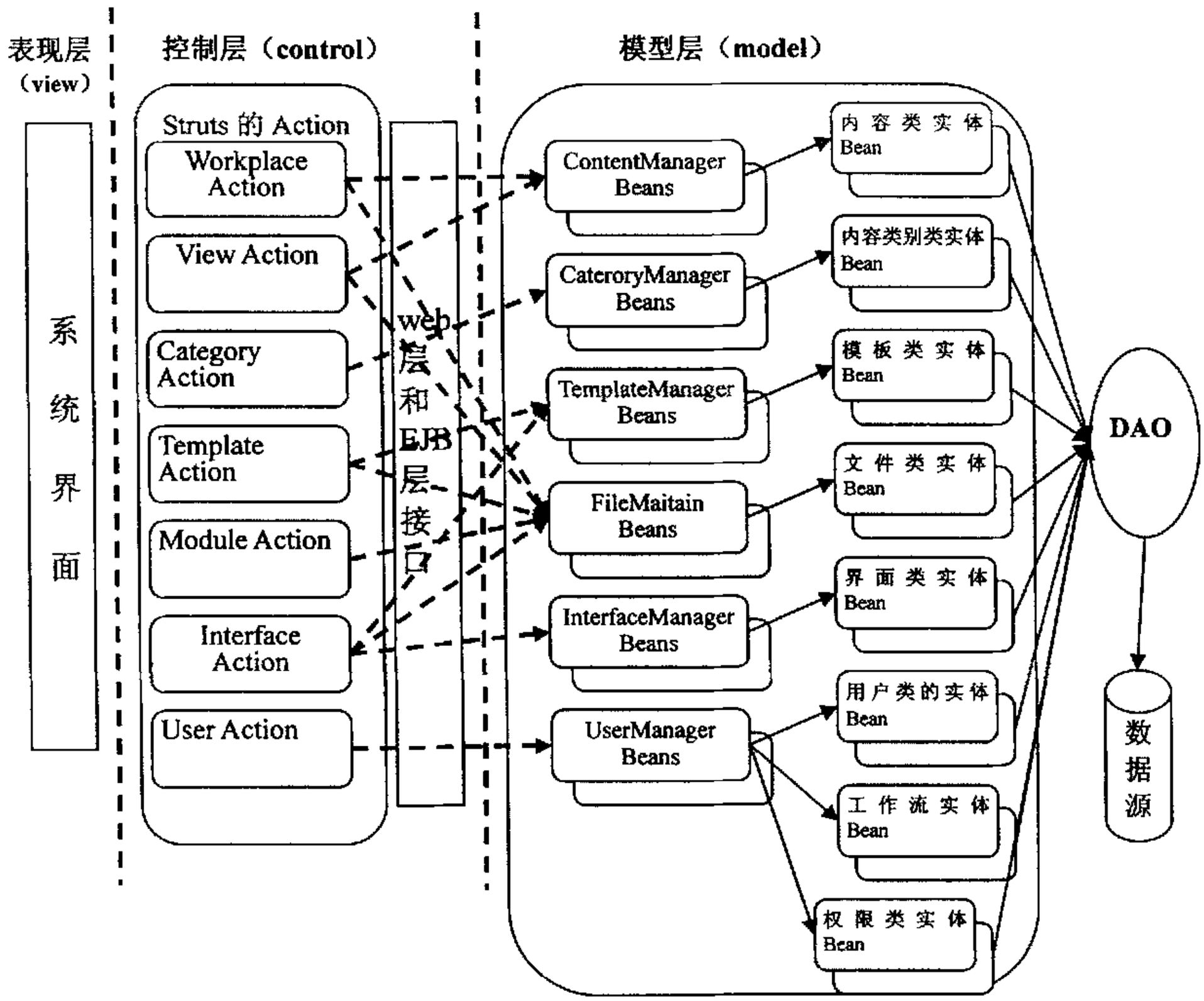


图 3-4 系统构架图

系统框架中使用了 web 与 EJB 层接口框架来实现 web 层和 EJB 层的解耦，以提高 web 层和 EJB 层的开发效率，加强 J2EE 系统的可维护性和可拓展性。此外，为降低实体 bean 和它们的控制层使用者间的耦合度，系统中还使用了 Façade 模式。从而在客户端和 EJB 之间建立了一个统一接口，客户端只要直接和 Façade 类交互操作，通过 Façade 类再操作那些实体 Bean。在系统中，根据不同的需要建立了多个 Façade 接口，其中比较重要的有以下几个：

- ContentManager

负责所有与内容有关的实体 bean 的访问。

- CateroryManager

负责所有与内容类别有关的实体 bean 的访问。

- TemplateManager

负责所有与模板有关的实体 bean 的访问。

- FileMaintain

负责与文件系统有关的访问。

- InterfaceManager

负责所有与界面设置有关的实体 bean 的访问。

- UserManager

负责所有与用户有关的实体 bean 的访问。

3.3.2功能设计

在进行系统设计时，如果要确保设计目标的成功实现还要解决以下问题：

首先，要实现基于内容的管理，必须能够将内容结构化、组件化并进行描述。这样做的前提是要建立相关的内容模型，并采用一种合适的方式来描述抽象的内容模型，从而将其具体化。

其次，如果要实现界面的定制，必须将内容本身与表现相分离。即需要采用不同的方式分别对内容本身和对内容显示方式进行描述。同时，

还要对界面元素进行抽象和描述。

再次，要实现系统目标所要达到的应用集成能力，必须设计比较完整的应用模块部署规范。

为此系统通过建立模板机制和应用模块机制来解决以上问题。首先定义了各种模板专门对内容模型、界面框架进行描述，其次定义了完整的应用模块部署规范，规定了模块的结构和各种属性以及与系统的交互方式。但如果构建一个符合项目实现目标的系统，还要建立一些辅助机制以完成一些重要环节。

- 光有描述内容模型的模板是不够的，用户还需要相应的内容创建方式。所以系统实现了根据模板自动生成内容创建界面的机制。而且界面的结构能够确保顺利地将内容分割为内容组件。
- 上传的内容是包含多个内容组件的复杂类型，系统还实现了相应的机制将内容拆分成单个组件，映射到数据库并分开索引、存储，同时生成相应的XML描述文档。
- 定制好的界面是以XML的形式存在的，系统要自动解析填充好的框架模板并生成相应的JSP页面，应用到系统中。
- 在应用模块导入系统后，系统要自动完成从注册到初始化的一系列工作。
- 所有的定制过程都涉及到对配置文件进行操作的问题。系统要有相关的机制来完成这个工作。

在以上各种机制的相互合作下，系统将按照如下方式运行：

- 用户通过模板管理部分对模板进行定制，实现对内容模型的抽象。定制的同时，系统自动生成相应的内容创建界面。
- 用户创建内容类别，并为新类别选择模板和设定各类参数。在这里要完成基于类别的工作流设定和栏目模块的设定。
- 用户可通过工作台对内容进行基于类别和组件的管理。根据用户要创建的内容，工作台自动提供相应的创建界面。在这里可以对内容进行检索，并完成内容的创建、编辑、审核、校对等工作。

- 用户对系统发布界面进行定制，选择要使用的栏目模块和应用模块加入界面，定制行为转化为对框架模板的操作。定制完毕后，系统自动将填充好的框架模板转为 JSP 页面放入相应的文件夹，然后新页面付诸使用。
- 用户将新的应用模块包载入系统，系统自动完成对模块的部署，用户在界面设置时可以选择是否启用新模块。
- 当客户端发来内容浏览请求时，系统找到所请求的内容，然后将内容组装并转换为适当的表现形式，返回给客户端。

以上这些机制的实现在第 4 章有详细描述。

3.4 学校内容分类模型

如果要对内容实施全面有效的管理，必须针对所要实施内容管理的内容域建立完善的组织框架，即内容分类模型。本系统采用树型结构来建立针对学校的内容分类模型。内容分类模型包括以下信息：

- 学校的信息主题分类。
- 主题间的层次关系。
- 主题的属性描述。

系统的内容信息都是按照内容分类模型来组织的，如图 3-5 所示。

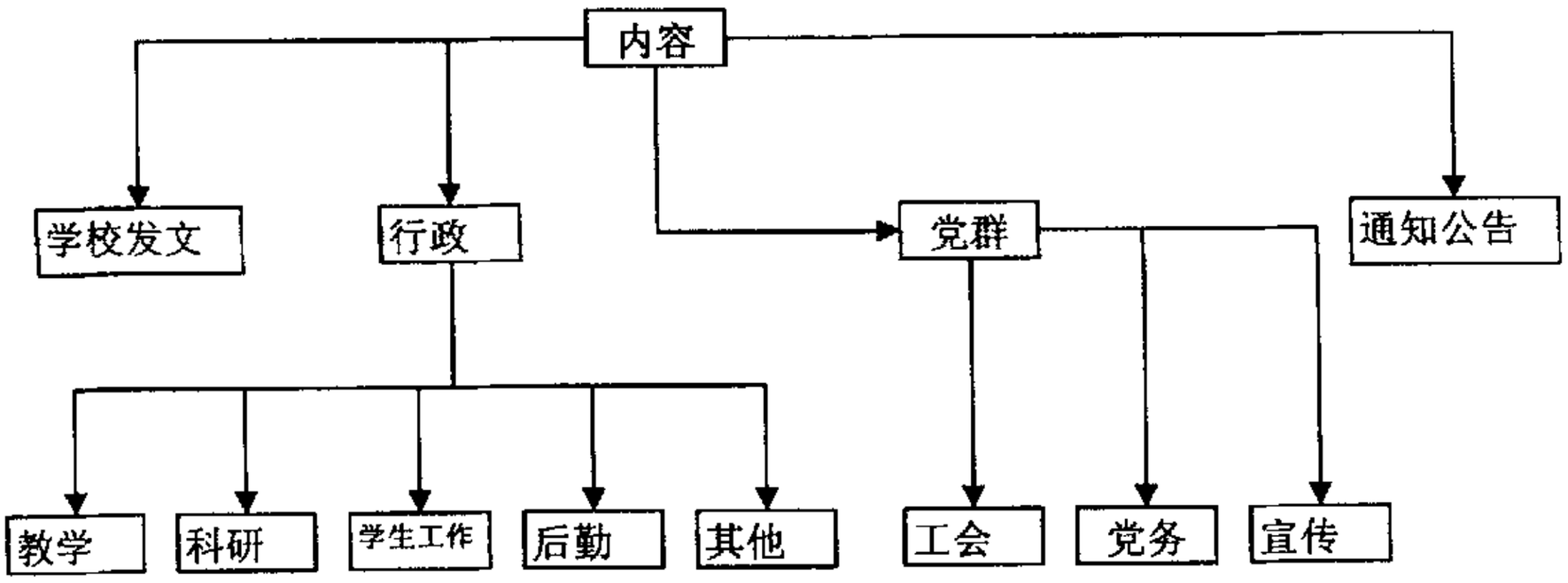


图 3-5 系统信息分类模型

面向学校的内容分类模型有序地组织学校各部门的信息，加快搜索和查询的速度。系统提供了对内容分类模型操作的方式，可以对其它上

层应用提供支持，而上层应用系统是和具体领域相关的，不同的领域其内容组织结构也不相同，可以利用系统提供的方式根据需要进行内容组织结构的定制。

3.5 内容逻辑模型

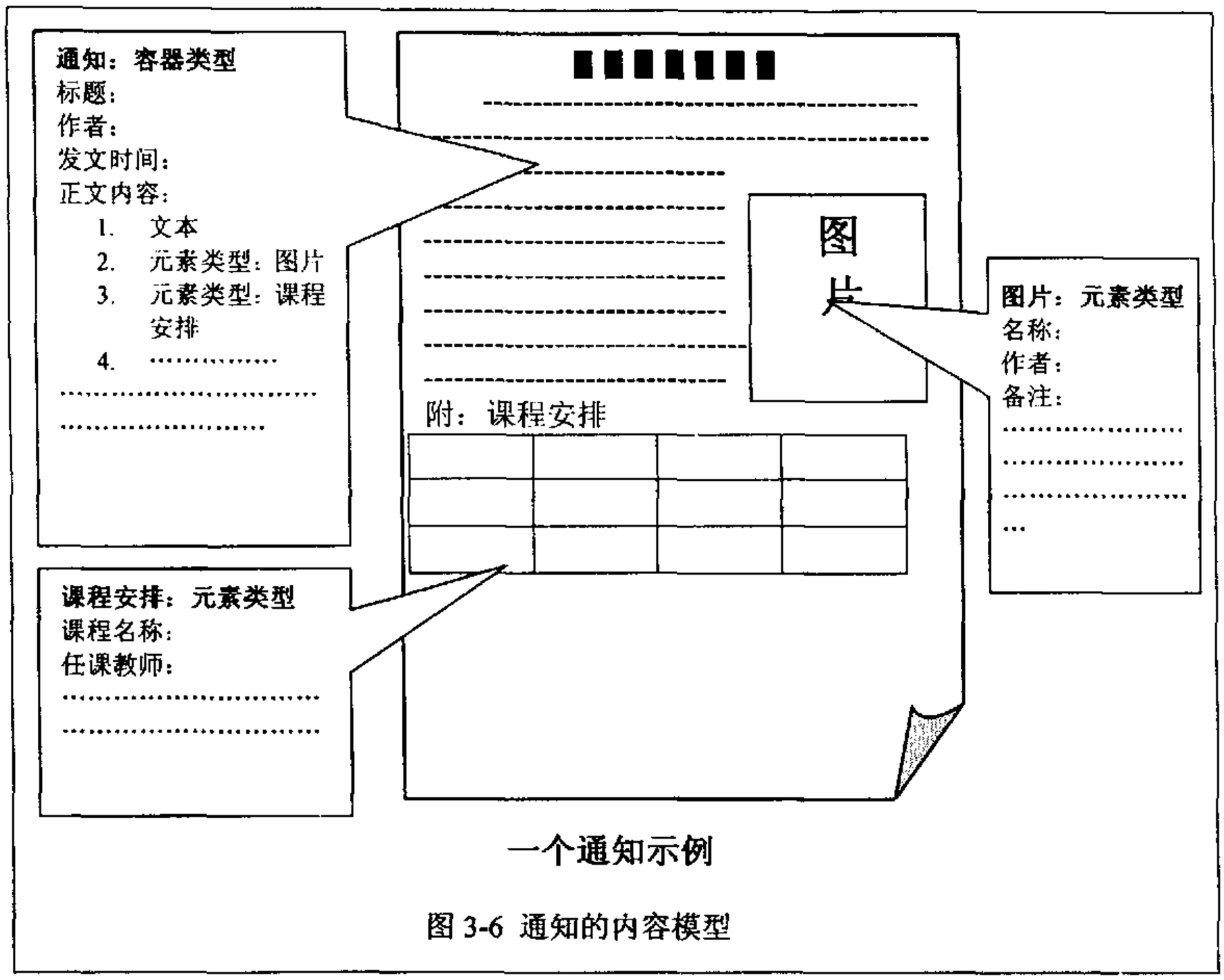
对内容管理来说，最重要的就是根据需要对内容信息进行抽象建模。不但要将相对独立的大的内容体如新闻、公告抽象建模，还要将大的内容模型分割成可重用的内容组件级模型。我们用 XML 作为内容模型的描述语言，并建立一整套机制，以实现内容在创建时按照模型进行分割、描述，形成相应的内容组件这一过程。

为能够将复杂的内容合理的分解，系统将内容逻辑模型分为两个层次：一是容器类型，一般是完整的发布单位，例如通知公告。另一类是元素类型，即内容组件的抽象模型。元素类型常常作为完整的发布信息的一部分，并且有一定的重用性。容器类型具有容器的作用，可以放入元素类型的信息。

系统提供一些常见信息类型的抽象模型，即所谓的内容模板。我们可以应用这些现成的模板来实现对常见内容的结构化、组件化和描述。

同时我们也提供给用户自己抽象并建立新的信息模型的手段。因为在某些具体的应用中，还需要用户根据需要自己抽象信息模型并创建新的类型模板。这就大大增强了系统的扩展性。

图 3-6 是一个具体的内容实体的抽象分解示例，其中的通知是属于容器类型的。同时，通知中还包含了两种元素类型的内容组件，即课程安排和图片。通知的模板除了描述通知的组成外，还指出到哪里找到这些组成部分。



一个通知示例

图 3-6 通知的内容模型

本系统已对一些学校常用内容模型进行了抽象，建立了相应的模板。容器类包括新闻、通知、公告、学校公文等等。元素类包括课程表、图片、考试安排、通讯录、成绩单等。 以学校发文为例，其内容组成为：发文单位、标题、内容、主题词、发文时间、抄报、报送、印发单位、校对、附件、发文编号、收文单位。其中内容和附件中又可能包括文本、图片、表格等。这些都通过相应的内容模板来精确描述。

3.6 模板机制

模板在系统中扮演着重要的角色，利用模板我们可以完成对内容的结构化、组件化和描述，还可以确定发布界面的布局和内容的发布形式。

3.6.1 模板的分类和作用

按照用途的不同，模板可分为三类：

● 内容模板

即对内容模型的抽象描述。表明内容的结构，指出内容是由什么元素组成的。利用内容模板可以在内容创建时将其结构化，并分割成内容组件。内容模板是内容分类模型的具体描述形式。一个内容模板可以由多个其它内容模板组成。对于半结构化和非结构化的数据，我们可以利用内容模板来添加元数据，并利用一个开放源码的全文检索引擎 Lucene 来建立全文索引。内容模板使用 XML Schema 来定义和描述。一个内容模板要定义的信息有：本内容模型中有哪些元素，各元素分别是什么类型，映射到数据库中各字段的大小、类型、名称以及映射生成 html 内容创建界面所需的必要信息等等。下面是新闻类型的内容模板片段：

```
.....
<xsd:element name="news" tag="新闻">
 <xsd:element  name="title"  tag="标 题 "  htmlTag="title"  type="xsd:string"
DBAttribute="title" size="64"/>
 <xsd:element name="content" type="ContentType"/>
 <xsd:element  name="author"  tag="作 者 "  htmlTag="author"  type="xsd:string"
input="off"/>
 .....
</xsd:element >
 <xsd:complexType name="ContentType">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="picture" type="picType" ProcessType="component"/>
        <xsd:element name="text" tag="正文" htmlTag="text" type="xsd:string"/
DBAttribute="text" size="8192">
      </xsd:sequence>
      <xsd:attribute name="type" type="xsd:string"/>
 </xsd:complexType>
.....
```

以其中的名为 title 的元素为例，模板定义了元素的类型（type），在数据库表中的字段名（DBAttribute），生成相应的创建界面时的 html 标签名（htmlTag）等等。

● 框架模板。

对界面的布局进行描述。并可指定界面的每一部分放置什么功能模块。可以通过对框架模板操作改变界面布局。对界面的定制就是建立在这一基础上的。框架模板使用 XML 定义和描述。下面是一个三列布局的

框架模板的例子。

```
.....
<FrameTemplate ID="237">
    <frameset rows="*,86%">
      <frameset cols="*">
        <frame name="top" elementType="pic" elementNumber="1">
        </frame>
      </frameset>
      <frameset cols="20%,30%,30%">
        <frame name="left" elementType="any" elementNumber="*">NULL </frame>
        <frame name="middle" elementType="any" elementNumber="*"> NULL </frame>
        <frame name="right" elementType="any" elementNumber="*"> NULL </frame>
      </frameset>
    </frameset>
</FrameTemplate>
.....
```

● 表现模板。

表现模板对应于特定的内容模板，每一个表现模板都是对它所对应的内容模板所描述内容的显示方式的规定。一个内容模板可以对应多个表现模板，而一个表现模板只对应一个内容模板。表现模板使用 XSLT 来进行描述和转换。下面是一个新闻的表现模板片段：

```
.....
<xsl:output method="html"/>
<xsl:template match="news-in-school">
<body bgcolor="#FFDDFF">
  <title><xsl:value-of select="title"/></title>
  <h1><xsl:value-of select="title"></h1>
  <xsl:apply-templates select="section"/>
  <hr/>
  <xsl:if test="author">
    <hr/>
    <h2>author:</h2>
    <xsl:apply-templates select="author"/>
</xsl:if>
.....
```

3.6.2 模板的工作方式

从内容的创建到最后发布，各类模板分别发挥着不同的作用。在内容创建时，系统根据特定的内容模板生成特定的输入界面。内容输入后即被分割为内容组件，结构化后存入数据库，同时以数据库记录和 XML

文档的形式存在。在发布时，将内容通过其内容模板所对应的表示模板转化为某种表现形式，再由框架模板按照某种布局，将内容和其他应用模块组织成为完整的页面提交给使用者。如图 3-7 所示。

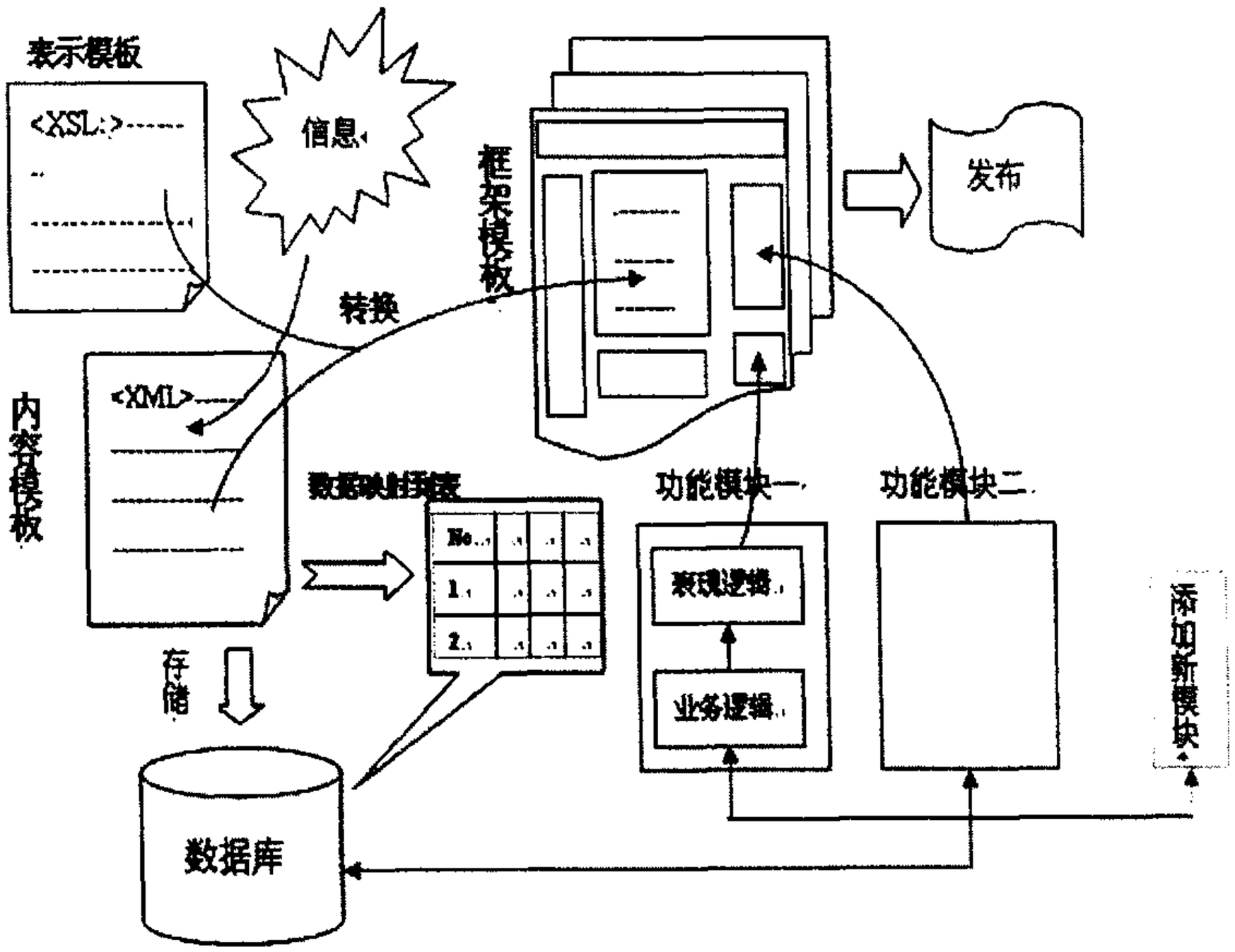


图 3-7 模板工作方式图

3.6.3 模板与界面定制

界面定制有两种，一种是对系统各类发布界面的定制，另一种是对内容发布样式的定制。与界面定制有关的模板是框架模板和表现模板。框架模板负责界面元素的描述，即要说明界面分为哪几部分，每部分都包含什么元素。表现模板对内容的表现形式进行描述。界面元素的类型有三种：

● 栏目模块元素

即某类内容的栏目模块，用于在有限的发布版面范围内将此类内容条目提供给大家，也是此类内容的入口。栏目模块和内容的类别是一一

对应的。

● 应用模块元素

即利用内容来实现某些应用的功能模块。和栏目模块比起来，应用模块不是简单的内容展示，而是要针对相应的应用，通过对内容的加工来提供给用户有用的结果。比如像学生考试情况分析模块就属于这一类。应用模块的界面本身又作为界面的元素显示在系统界面上。

● 内容类型元素

这类元素其实是比较完整的内容实体，如通知公告、学校发文等等。

一般来说，一个完整的页面就是由这三类元素组成。所谓界面定制就是要系统提供能够改变界面框架布局和选择各类元素的手段。每个系统发布界面都有自己已填充好的框架模板，发布时系统首先解析已填充好的框架模板内容，取得各元素信息。然后根据这些信息生成 JSP 页面。JSP 页面中已指定了生成页面所需各种资源的位置。最后取得所有资源生成 html 提交给用户。所以，对发布界面的定制就是对框架模板的定制。系统提供了对框架模板进行操纵的 web 界面，可以很方便的对框架模板进行各种操作。因而实现了对发布界面的定制。而要实现对内容发布样式的定制，就要针对每种内容模板设计一些以 XSLT 描述的样式转换文件。可以针对不同内容选择不同的样式转换方式。同时系统也给用户提供了自己创建样式转换文件的手段。

3.6.4 模板与内容模型

内容模板是内容模型的载体，系统通过内容模板来描述内容模型。XML Schema 以其相比 DTD 的强大优势成为模板描述语言的首选。之所以选择直接用 XML Schema 而不是用 XML 来定义内容模板中的元素，是因为这样可以直接利用 XML Schema 严谨的元素类型定义，而且 XML Schema 本身也符合 XML 规范，可以很容易的处理。在建模过程中，内容模型存在嵌套的情况。即一个内容模型可能由很多更小的内容模型组成，这些内容模型可以看作是一个个的内容组件。而模板对内容的描述应该体现

这一点。我们采用了一种模块化的方法，把模型分解成多个模块。然后再使用 include 方式将所有的模块都组合在同一模板中。在组合时，这些模块提供了该内容模型的完整框架。这种方式利于使用现有内容模板针对新内容模型的生成新的内容模板。也就是说一个模板可以由多个其它模板组成。另外，我们在内容组件化的过程中还应用到了 XML 实体的概念。但因为实体是 XML DTD 中的概念，而 XML Schema 中没有定义对应的等效物。所以我们采取了变通的方式，采用 XML Schema 中的 group 元素来模仿实体参数的行为。

在系统开发过程中，我们对学校中的一些常见内容类型（见 3.5）进行了建模。所有的模板标记都建立在同一命名空间中。为了确保内容模型的可扩充性，我们抽取各类内容的共性定义了一个公共的词汇表。各内容模型在公共词汇表的基础上还拥有自己的词汇表，这样在建立新的内容模型时，只要建立其特有的词汇表就可以了。公共内容模型 XML Schema 中定义了最高抽象层次公共内容模型的元素及其属性。

需要特别指出的是，对于半结构化、非结构化的内容数据，内容的主体不在相应的 XML 描述文档中。XML 描述文档只包含指向内容真正载体的 URI。

内容模型的模块化设计，保证了内容组件化的顺利实现。

3.7 系统应用的模块化和可定制

要想使系统在不改动原有代码的情况下的添加新功能，必须建立一定的功能模块部署规范，并且建立一定的模块导入机制，使之很好的与系统融合。因为本系统是基于 web 的系统，所以所谓应用模块还应是一组针对某种应用的文件的组合。本质上仍然是服务器端程序。为了使应用模块能够很好的集成进系统，并且有一定的通用性，则必须制定相应的约定来对模块的结构、模块与系统的交互、模块的行为做细致的规定。

3.7.1 功能模块的结构

一个功能模块应该由以下几部分组成：

● 模块自身描述部分

对模块的自身信息进行描述，通常是一个叫做 model.xml 的 XML 文件。这部分非常重要，系统就是从这部分取得模块的所有信息并决定如何集成模块代码的。

● 初始化逻辑

在第一次安装模块时，要做一些初始化的工作，如建立数据库表、建立安装目录等等。这些工作由初始化逻辑来完成。

● 模块表现逻辑

模块也有可视化外观，直接表现为系统界面上的模块操作界面块。负责界面显示的服务器脚本就属于模块表现逻辑部分。

● 模块业务逻辑

针对某项应用对内容进行加工处理的逻辑部分就是模块业务逻辑。整个应用过程就是由这部分完成的。

作为 web 应用程序，模块应遵循 MVC 模式。具体地说模块本身也要使用 struts 框架进行开发，以便更好的和系统融合，同时实现表现逻辑和业务逻辑相分离。

3.7.2 模块与系统的交互

模块与系统之间必须进行交互才能很好的协调工作。模块必须告知系统自己的存在和其它自身的必要信息。系统则要告知模块系统的数据源等信息，以提供数据服务。这些交互工作都是通过各类配置文件来完成的。

模块载入系统时，运行系统的脚本。脚本执行时首先分配一个系统 ID，然后读取 module.xml 的内容。将其中的部分信息和分配的 ID、数据源等信息写入系统的模块配置文件。

然后，系统再调用模块的初始化代码进行进一步的初始化。初始化

时一般要建立模块自己的数据库表，并从其它配置文件中读取一些必要信息。

此外，作为模块与系统交互的一种方式，模块有时会使用系统的一些类。为了能够方便的实现这一交互，模块的业务逻辑和表现逻辑都是分开放在不同的系统目录下的。

第4章 系统实现

4.1 工作台

工作台是内容编辑和创建的工作平台。内容的结构化、组件化和描述是由工作台完成的。用户可以利用工作台提供的各种功能对内容进行管理。具有不同权限的工作人员在这里分工合作，确保高质量、高效率地完成内容的维护工作。

4.1.1 内容的创建和编辑

内容的结构化、组件化和描述是通过工作台中的内容编辑界面完成的，内容编辑界面是相应的内容模板在创建时自动生成的 html 页面，是系统表现层根据要创建的内容类型自动选取的。编辑界面里表单中的元素与模板所描述的元素有一一对应关系，填写表单就相当于为内容模板所描述的元素赋值。

内容提交到服务器端后，系统根据模板生成相应的 XML 文档，并且将内容分割为内容组件存储在数据库或文件系统中。系统对内容的管理是组件级的，所以内容提交后服务器端程序要根据内容模板将内容拆分为内容组件分开存放。每一种组件都单独存放并索引，可以单独查询和编辑。比如说新闻中的图片既可在对新闻的浏览中看到，又可单独进行查询、浏览。

很多类型的内容结构复杂，由各类内容组件组成，而且既包含文本又包含二进制文件，这样的内容创建后，如何简便的提交到服务器是一个比较麻烦的问题，提交到服务器端后如何恰当的分割为组件并分别存储也是比较棘手的。系统采用两种方式相结合来解决这个问题。一种方式是采用内嵌网页的方式分表单、分次提交内容。这主要是针对含有多个二进制文件的情况，这样的话，因为组件是分次提交所以可以很容易的将各个组件分开管理。另一种方式是在表单中加入隐藏域，专门用来存储必要的信息，以便于内容组件的分离。

下面以校内新闻的创建为例，详细介绍一下这一过程的实现。新闻的内容中除了文本外还可能包含多张图片。图片是与内容的其它部分分次提交的，数据到服务器端后，再根据提交的表单隐藏域中的信息将内容个部分联系起来，所以我们将文本部分的提交和图片的提交分开介绍。

当内容被创建时，部分文本内容创建界面如下：

```
.....
<input type="text" name="title" size="64">
<textarea name="Content" wrap="VIRTUAL"></textarea>
<input type="hidden" name="author" value="Liu Ting">
.....
```

当文本内容提交到服务器端时，Action 类 WorkplaceAction 通过 Request 对象得到 title 的值：

```
.....
title=req.getParameter("title");
content=req.getParameter("content");
author=req.gerParameter("author");
.....
```

然后，WorkplaceAction 调用 JDOM 的 API 来操作 XML 文档：

```
.....
element.addChild(new Element("title").setContent(title));
element.addChild(new Element("content").setContent(content));
element.addChild(new Element("author").setContent(author));
.....
```

这样 title、content、author 等元素就被加入了相应的 XML 文档。

```
.....
<title>国际免疫学研讨会在我校召开</title>
<content> 12 月 8 日，国际免疫学研讨会在我校召开，来自美国哈佛大学医学院、斯坦福大学医学院和中国科学院的多名院士及众多免疫学专家出席了研讨会。
.....
</content>
<author> Liu Ting </author>
.....
```

因为在创建相应的内容类别时，已经根据所使用的模板创建了相应的数据库表，所以只要把表单数据交给 ContentManager，由 ContentManager 再将数据映射到数据库中就可以了。

新闻中的图片作为内容组件是与其它部分分开提交的。我们通过创建界面中的内嵌网页来实现多张图片及其原数据的分次提交：

```
<IFRAME src="ContentAction.do?TID=0312&pic-create=9320055" frameBorder=0 width="100%" scrolling=no height=30> </IFRAME>
```

对应的 html 如下：

```
.....
<form enctype=multipart/form-data method=post action=TemplateAction.do?type=pic>
<table cellpadding=0 cellspacing=0 width=100%>
<tr><td><IMG src=images/affix.gif>上传图片</td></tr>
<tr><td><input type="text" name="title" size="32"></td></tr>
<tr><td><input type="hidden" name="size" size=""></td></tr>
<input type="hidden" name="TcontentID" value=" 9320055">
<tr><td><textarea name="description" wrap="VIRTUAL"></textarea></td></tr>
<tr><td align=right><input type=file name=file size="30">
<input type="submit" value="上传" name=Submit>
.....
```

图片上传后，元数据存入专门的数据库表，图片放入相应的目录。可以看到，页面中有一个隐藏域：

```
<input type="hidden" name="TcontentID" value="9320055">
```

因为内容中的组件是与内容主体分开来上传的，所以必须先生成一个编号作为隐藏域加入创建界面。等所有组件都上传后，根据编号建立各组件间的联系。所以当包括图片在内的所有内容组件都上传后，新闻内容的 XML 文档中会加入：

```
<picture ProcessType="component">
  <title>计算机系元旦师生联欢会——交谊舞 1</title>
  <size>137</size>
  <description>数据结构老师和同学跳慢四</description>
  <uri>./pic/news/campus/9320055.jpg</uri>
  .....
</picture>
.....
```

同时，图片的元数据被映射入相应的数据库表中。

可以看到内容模板中有的元素含有 processType 属性。其实内容模板所描述的 ComplexType 类型的元素，只要其 processType 属性指定为‘component’，则这类元素就会被作为内容组件来处理。系统会将其单独存放，并为之建立与其它组件的联系。

通过上面的方式，新闻被分割为不同的内容组件。新闻图片和元数据与新闻的其它信息分开存储。在内容创建好后，结构化、组件化和描述的过程也完成了。通过将内容分割为组件分开存储，使得我们可以实现组件级的内容利用。比如说可以对图片实现单独索引和查找，进行图

片的单独管理。这样我们在新闻中用到的图片，也可以使用在学校发文中，只要在起草学校发文时检索到相应的图片并引用过来就可以了。通过这样的管理方式可以使内容组件达到很高的复用性。同时，利用 XML 对内容的完整描述可以达到两个目的：一个是实现内容与表现的分离。我们可以很容易地将以 XML 来描述的数据以多种形式表现出来。另一个是建立内容组件间的联系。一个内容体的完整的 XML 文档将描述内容体的内容和结构，同时将表明内容体中各组件间的联系。

对于结构化的内容信息，在编辑时直接填写表单，提交后直接映射到数据库。对于半结构化、非结构化内容信息，除了填写元数据表单外，还要上传内容文件。如果文件是文本类的，而非图片、声音等二进制文件，那么如果要对这部分内容进行有效管理，除了添加元数据外系统还要根据文件内容生成内容索引。生成索引的过程是系统采用一个开放源码的全文检索引擎 Lucene 来完成的。系统可以通过引擎的外部入口调用诸如 Field.Text(String name, String value)的方法实现切分词索引并存储。

为了建立一个通用的数据输出接口，以支持多种类型文件内容索引，如 html 文件、文本文件、pdf 文件等等，我们采用了另一个基于 Lucene 的开源项目中的基于 XML 数据源的索引器。项目名称为 WebLucene。这样我们以标准的 XML 为描述语言的中间格式作为 Lucene 的数据导入接口，然后其他数据（如 PDF）只需要通过相应的解析器转换成标准的中间格式就可以进行数据索引了。建立全文索引后，系统就可以利用引擎的检索入口 API 实现快速的全文检索。这样用户就可以通过元数据检索和全文检索相结合的方式来查找内容信息，从而为半结构化、非结构化内容数据管理提供了有力的手段。

4.1.2 内容访问控制

系统的权限管理是基于用户组的，同一用户组的数个用户可能对同一类内容都具有编辑权限。当一个用户对数据进行操作时，另一个用户

可能正在对相同的内容进行操作。这时为了维护数据的一致性，在用户进行操作时需要首先检查是否有其他人在对相关数据进行操作。如果客户端请求的数据操作和其他人的操作有冲突，系统还要反馈回信息通知用户。即不但要维护数据一致性，还要保证不同用户间的协调合作。 所以内容的维护要解决访问控制的问题。

系统管理的内容分为两种：已发布内容和未发布内容。内容从创建、采集到最后发布要经过一个流程。这个流程中有很多结点，分别负责对内容的编辑、校对、审查等工作。经过以上种种工序后，内容到达流程的最后一个结点——发布，即内容达到可以正式对外发布的状态，这时内容即为已发布内容。而在达到最后一个结点前，内容的状态为未发布。系统规定只有处在未发布状态下的内容才是可编辑的。如果想要对已发布内容进行编辑，则首先要将其重新设为未发布状态。所以，访问控制是针对未发布状态内容的。

为了实现前面所说的目标，系统在对内容存取时实现了一种访问控制器，这种模式可以很好地完成上述目标。控制器包括以下组件：

- CheckManager

主要负责接受用户的访问请求，并调用 ConceptStrChecker 来检查用户的读写权限，然后将适当的权限赋予当前用户。

- AccessTree

是根据当前系统内数据状态为每一条未发布内容生成的权限树，每个节点对应每个所管理的内容体。比如一棵权限树对应一条未发布新闻，而树中的每个叶子结点代表新闻的一项不可再分的数据，每个非叶子结点代表一项复杂类型数据或内容组件，由于这棵权限树是在数据被改动后的系统空闲期间生成的，所以它不会对系统的实时反应和整体性能产生不良的影响。另外，因为未发布内容通常数量较少，所以生成的权限树数量不会很多。

- AccessRule

AccessRule 是事先由开发人员写进系统的，决定用户访问数据的规

则。

● ConceptStrChecker

根据来自 CheckerManager 的事件而检查权限树，同时参照系统内的 AccessRule，之后向 CheckerManager 返回检查结果。

图 4-1 说明了访问控制器的运行原理。理论上，在 web 层而不是 EJB 层就阻挡了无效操作（一般事务管理在 EJB 层进行），这样就大大减少了用户不必要的等待和系统计算时间。

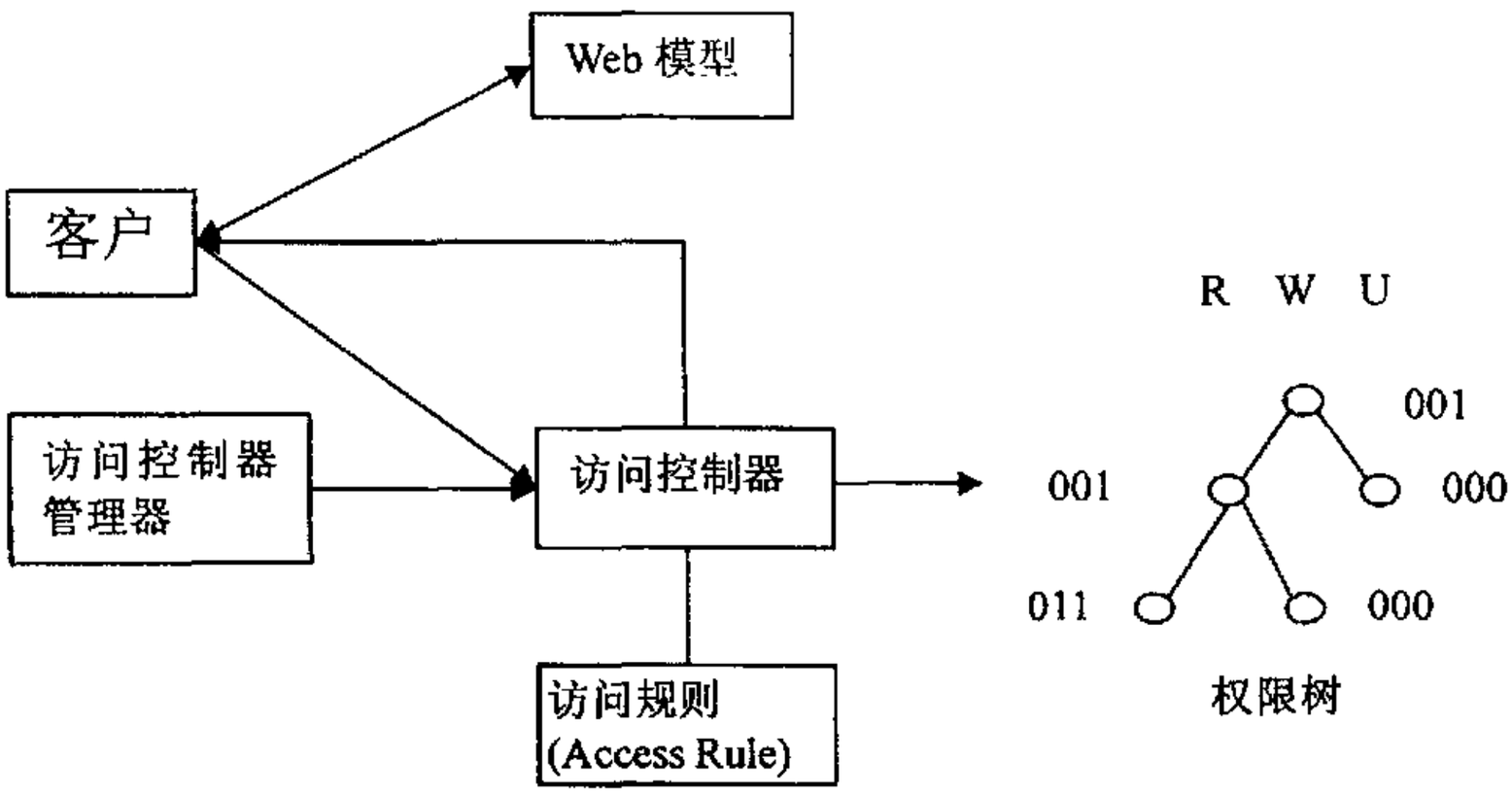


图 4-1 控制器原理图

在这个结构中，组件之间主要传递三位请求状态信息，即 R，W，U。R 代表读（read），W 代表写（write），而 U 代表向上检查位，它主要是用于检查当前请求数据的下层数据（所有直接或间接子结点）是否正在被改动。比如，如果一个数据正在被某些用户读取，而且正在被某些用户改动，它的所有子结点都没有正在被改动，那么它的状态为“110”。在 CheckManager 收到用户的请求之后，他将调用 ConceptStrChecker，后者将根据 AccessRule 定义具体的访问规则，如果一个结点正在被更新，那么它的所有父结点（直接或间接）都不能被删除。基于一定的权限，用户可以访问相关的数据模型。在 ConceptStrChecker 中对客户端的数据请求事件进行分析，通过 AccessRule 的规定和当前树状结构的状态值确定是否允许客户进一步使用 EJB。

4.2 界面定制

界面定制是指对包括主页在内的各种发布界面的定制。通过界面定制，用户可以对发布界面的布局、界面上的元素进行规定。界面定制过程中系统要做的工作主要有两部分，一部分是把用户通过 web 页发来的定制请求转化为对框架模板的操作，从而形成一个填写好的框架模板。另一部分是将设置好的框架模板根据专门的 XSLT 所规定的转换方式转换成 JSP 文件。因为界面一经定制就不会经常改变，所以不必在每个用户请求到来时都将界面重新生成，而只在对界面进行重新设置时再重新生成界面的 JSP 文件。下面我们以主页的定制为例，来对界面定制的实现进行说明。

对于界面布局和元素的定制，系统提供了一种比较直观的方式来进行。当用户选好定制界面所要使用的框架模板时，系统会根据框架模板的描述生成空白的框架页面，然后用户通过框架页面上的链接来指定空白框架页面每一部分要显示的元素。

首先，InterfaceAction 从管理模板的 Façade 接口得到要使用的框架模板，并根据模板生成相应的定制界面。下面是一个两列布局的框架模板：

```
<?xml version="1.0"?>
<FrameTemplate ID="237">
    <frameset rows="*,86%">
      <frameset cols="*">
        <frame name="top" elementType="pic" elementNumber="1">
        </frame>
      </frameset>
      <frameset cols="*,66%">
        <frame name="left" elementType="any" elementNumber="*">NULL</frame>
        <frame name="right" elementType="any" elementNumber="*">NULL</frame>
      </frameset>
    </frameset>
</FrameTemplate>
```

可以看到框架将界面分为三部分，最上方是放置 logo 的区域。下面的区域分为两列，可添加各种元素。系统将这个模板转换为 html 的定制界面：

```
<table border="1" cellpadding="0" cellspacing="0" style="border-collapse: collapse"
```

```
bordercolor="#111111" width="101%" id="AutoNumber1" height="100%">
  <tr>
    <td width="100%" height="108" align="left" valign="top">
       本部分为 logo 区域，可放置图片
       <a href="interfaceAction.do?fname=top">点击这里添加界面元素</a>
    </td>
  </tr>
<tr>
  <td width="100%" height="18" align="left" valign="top">
   <table border="1" cellpadding="0" cellspacing="0" style="border-collapse: collapse"
       bordercolor="#111111" width="100%" id="AutoNumber2" height="100%">
    <tr>
     <td      width="34%"      height="473"><a      href="interfaceAction.do?action=
          InterfaceCustomize&FtemplateID=237&fname=left"> 点击这里添加界面元
          素</a>
     </td>
     <td     width="66%"     height="473"><a     href="interfaceAction.do?     action=
          InterfaceCustomize&FtemplateID=237& fname=right"> 点击这里添加界面元
          素</a>
        </td>
    </tr>
   </table>
    </td>
  </tr>
</table>
```

这是一个用表格嵌套来进行页面布局的 html 页面，其中加入了用来处理界面定制请求的服务器端 servlet 链接。可以看到<table>标记按嵌套逻辑顺序从外到内与模板中的<FrameSet>标记一一对应。这个转换定义在 interface-customize.xslt 文件中，由 customizeProcessor 类来完成，下面是转换的过程：

```
StreamSource contentSource =new StreamSource(Curi);
StreamSource styleSource =
      new StreamSource(./transform/style/interface-customize.xslt);

TransformerFactory transformerFactory = TransformerFactory.newInstance();
Transformer transformer = transformerFactory.newTransformer(styleSource);
transformer.transform(contentSource, result);
```

通过以上方式，系统提供给用户一个可视化的创建界面。

下一步，系统把用户通过创建界面发来的请求转变为对框架模板的操作。下面是常见的定制请求：

```
action=InterfaceCustomize&FtemplateID=237&fname=left
```

请求表明定制要使用编号为 237 的框架模板，要对模板的名为 left 的部分进行设置。然后 InterfaceAction 去执行元素选取界面 elementlist.jsp。elementlist.jsp 从 element-config.xml 中读取所有模块信息并生成列表提供给用户。用户选好要添加的元素后将选择结果提交，系统就会得到相应元素的 ID。根据以上定制信息，系统对框架模板作相应的操作。我们采用 JDOM 的 API 来实现对模板的各种操作。下面是系统用界面元素对框架模板进行填充的过程：

```
.....
//找到要添加元素的位置
Element father=doc.getElement("frame",framename);
Namespace ns=Namespace.getNamespace("javaXML",URL);
//建立元素和相应的属性
Element element= new Element("element").addAttribute("ID",number);
element.addAttribute("type", EType);
.....
//将元素加入相应的位置
father.addChild(element);
.....
//输出
XMLOutputter fmt=new XMLOutputter();
fmt.output(doc,out)
.....
```

可以看到，利用 Element 类的方法，我们把界面元素添加到<frame>标记中。最后还需要用 XMLOutputter 将这些改动写到附加的 OutputStream，确保修改被映射到 XML 文件中。这时我们得到填充好的模板：

```
<?xml version="1.0"?>
<FrameTemplate ID="237">
    <frameset rows="*,86%">
      <frameset cols="*">
        <frame name="top" elementType="pic" elementNumber="1">
         <element ID="2743352" type="picture">
         <uri>./pic/logo/应用技术学院 logo1.jpg</uri>
         </element>
        </frame>
    </frameset>
    <frameset cols="*,66%">
      <frame name="left" elementType="any" elementNumber="*">
        <element ID="03376" type="categoryModule">校内新闻<element/>
        <element ID="01137" type="categoryModule">通知公告<element/>
      </frame>
```

```
      <frame name="right" elementType="any" elementNumber="*">
        <element ID="02191" type="categoryModule">学校发文<element/>
        <element ID="01285" type="applicationModule">考分查询系统<element/>
      </frame>
    </frameset>
  </frameset>
</FrameTemplate>
```

然后，系统将填充好的框架模板转化为相应的 JSP 文件。转变规则定义在 XML2JSP.xslt 中。利用 java 类库中的 transform 类就可以实现从 XML 到 JSP 的转变。转变过程与定制界面的生成过程类似，这里不再赘述。在转化过程中，除了将界面布局信息加入界面脚本外，所有界面元素都将利用 Struts 框架中的 tiles 标记包含到界面中来。下面是一个简要的框架模板的 XML 标记片段：

```
<Frame name="left" elementType="any" elementNumber="*" >
  <element ID="03376" type="categoryModule">校内新闻<element/>
  <element ID="01137" type="categoryModule">通知公告<element/>
  <element ID="02191" type="categoryModule">学校发文<element/>
  <element ID="01285" type="applicationModule">考分查询系统<element/>
</Frame>
```

它被转换为如下形式：

```
<tiles:insert page="./template/FrameTemplate/filed/552.jsp" flush="true">
  <tiles:put name=" 校内新闻" value="./jsp/module/category/03376/03376.jsp"/>
  <tiles:put name="通知公告" value="./jsp/module/category/01137/01137.jsp"/>
  <tiles:put name="学校发文" value="./jsp/module/category/02191/02191.jsp"/>
  <tiles:put name=" 考 分 查 询 系 统 "
value="./jsp/module/application/01285/first.jsp"/>
</tiles:insert>
```

Transform 类首先根据框架模板的内容生成框架布局的 JSP 页面（*./template/FrameTemplate/filed/552.jsp*）。即第一行中参数 page 的值所指向的页面。然后将其它界面元素对应转化为相应的 tiles:put 标记。其作用是将所指向的模块的 JSP 入口页面包含进来。像*<tiles:put name=" 校内新闻" value="./category/entrance/03376.jsp"/>*这样的脚本就是指定了校内新闻栏目模块的入口 JSP 页面。在界面脚本运行时 552.jsp 负责整个界面的布局显示。而其他的模块入口页面如 03376.jsp 将被包含到总的界面中来。

最后，客户端发来确认使用这个新定制界面的请求，InterfaceAction 对象利用 FileMaintain 类的实例将旧的主页用新的

JSP 文件代替。其中所有以前定制的界面模板都作为存档保留。这样就很好的利用 Struts 框架的现成标记实现了界面的定制。

参与这个过程最重要的几个类有：

- InterfaceAction

负责根据请求来采取相应的动作。是界面定制过程的组织者。

- CustomizeProcessor

负责根据请求对框架模板进行填充、修改等操作，可以根据要求将元素放置在界面的某些位置。

- InterfaceTransformer

将填充好的模板转化为 JSP 页面。

- FileMaintain 是一个 Session bean

作为 Façade 类，文件的维护操作，是访问系统文件的统一接口。在这里 FileMaintain 负责将新生成的 JSP 文件代替当前正在使用的文件。

以刚才所举的定制过程为例，具体的实现如图 4-2 所示。

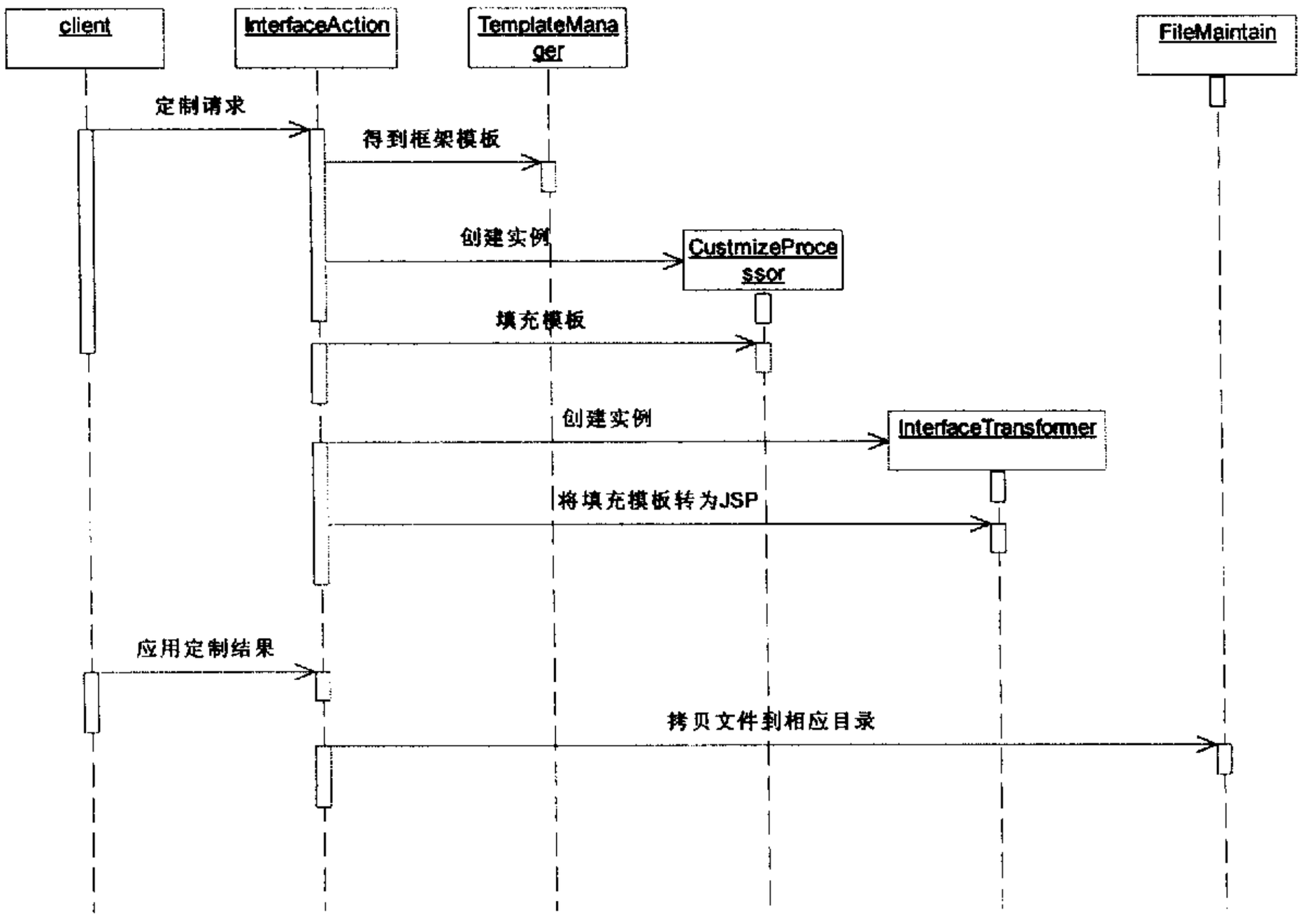


图 4-2 界面定制时序图

下面对这一过程作一个简述。首先，InterfaceAction 从管理模板的 Façade 接口得到要使用的框架模板，生成定制界面。然后客户端发出的设置界面的请求。InterfaceAction 对请求进行分析后，确定设置的是模板的哪个部分，使用的是什么元素。紧接着 InterfaceAction 得到一个 FTemplateProcessor 对象，再利用这个对象根据提供的信息对框架模板进行填充。然后再利用 InterfaceTransformer 对象的实例将填充好的框架模板转化为相应的 JSP 文件。最后，当客户端发来确认使用这个新定制的界面的请求时，InterfaceAction 对象利用 FileMaintain 类的实例将旧的主页用新的 JSP 文件代替。

和系统界面的定制比较相似的是内容显示方式的定制。系统在显示内容信息时，先取出内容信息的 XML 文档，同样是使用 transform 类，按照事先指定的表现模板所描述的方式将内容转化为 html 显示出来。

4.3 模板管理

模板管理部分要完成对各类模板的维护工作，特别是提供给用户自己抽象内容模型并将其描述出来的方式，即对内容模板进行创建。模板的创建方式有两种，一种是利用其他编辑工具将模板文件编辑好，再上传到系统中。另一种是采用系统自己的编辑界面来创建。这里特别要指出的是内容模板的创建。通过创建内容模板，用户可以自己抽象内容类型，从而使系统可以处理更多类型的内容信息。在内容模板创建的同时，相应的内容创建界面也根据模板的结构自动生成，从而为相应内容的结构化，组件化和描述做好准备。系统为内容模板的创建提供了可视化的界面。即采用模板结构树与表单相结合的方式来创建内容模板。因为内容模板是 XML Schema 描述，很适合用树型结构来表示，所以我们通过表单来添加模板树上的结点。因为我们采用树型结构来表示模板，所以可以很容易地对添加的元素位置进行定位。下面以新闻内容模板的创建为例来说明模板创建的实现过程。

首先，这些工作主要由以下几个类完成：

● TemplateAction

是 Struts 框架中的控制类。负责接收客户请求，并将请求分析后，交给相应的类来处理。

● TemplateManager

是一个 Session bean。作为 Façade 类，他封装了对各类模板的实体 bean 的创建、修改、删除操作，是访问各类模板的统一接口。

● CTInit

是负责在内容模板创建、修改时生成相应的内容创建界面的类。这些界面是以 html 形式存在的。创建界面主要是以表单的形式存在的。在进行内容维护时通过对这些表单的填写完成内容的结构化、组件化。在 CTInit 类中，要对内容模板进行解析，根据解析结果来生成相应的表单。内容模板每次被修改后都要重新生成相应的创建界面。

● TemplateCreator

在用户采用系统的编辑界面来创建模板时，进行模板创建的后台工作。

通过图 4-3 我们可以清楚地看到一个内容模板的创建过程。

第一步，TemplateAction 收到客户端发来的向未完成内容模板添加新元素的请求，生成 TemplateCreator 类的实例。实例在创建时载入指定的未完成内容模板临时文件。用户提交的元素信息中除了结点的各项属性外还有其父结点的类型和值，用以确定添加元素的位置。这些信息可以通过内置 request 对象得到。如下所示：

```
.....
fatherName= req.getParameter("fname");
fatherValue= req.getParameter("fvalue");
nodename=req.getParameter("Nname");
nodeType=req.getParameter("nType");
AttributeInDB=req.gerParameter("AttributeInDB");
AttributeSize= req.gerParameter("AttributeSize");
date=session.getAttribute("date");
.....
```

由上面的代码可以得到新添加结点的父结点的位置及各项属性值。

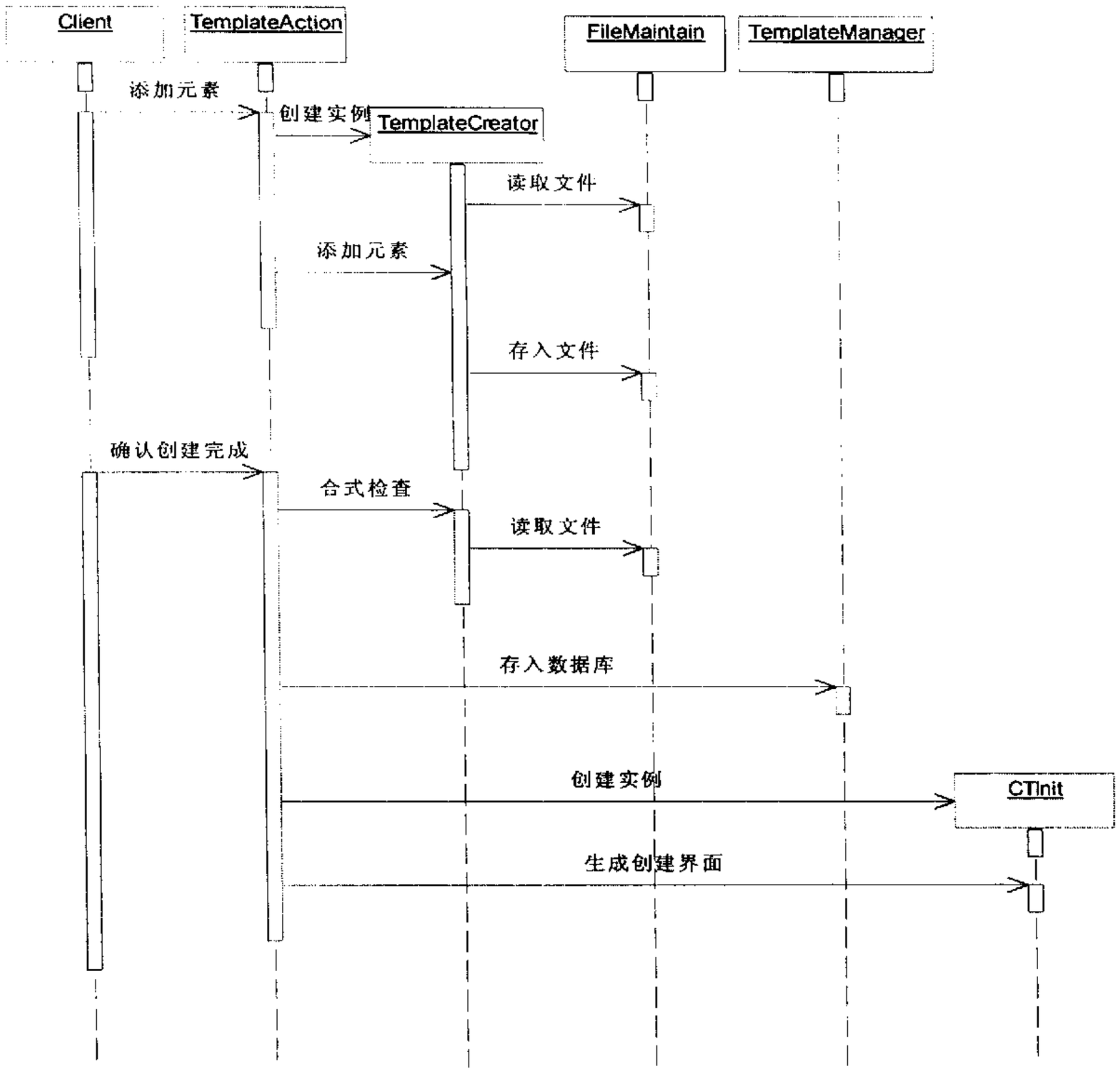


图 4-3 模板定制时序图

第二步，TemplateAction 通过调用 TemplateCreator 实例的方法来加入新元素。对模板临时文件的操作是通过 JDOM 来实现的。下面的代码将新的元素加入了内容模板：

```
//得到根结点
Element root=doc.getRootElement();
//建立新元素
Element element= new Element(nodename).addAttribute(nodetype, "xsd:string");
//加入各种属性
element.addAttribute("DBAttribute", AttributeInDB);
element.addAttribute("size", AttributeSize);
.....................
//作为子结点
root.addChild(element);
.................
```

然后通过 XMLOutputter 类将改动写回模板临时文件。

第三步，模板修改完毕后，客户端发出确认请求。TemplateAction 首先通过 TemplateCreator 对新生成的模板进行进行合式检查，确认模板没有错误。然后通过 Façade 入口 TemplateManager 将模板存入数据库。

最后，TemplateAction 创建一个 CTint 类的实例，通过 CTint 类生成相应内容模板的 html 创建界面。下面我们简要介绍一下这一生成过程。

这里是新闻内容模板的一部分：

```
.....
<xsd:element name="news" tag="新闻">
.....
 <xsd:element  name="title"  tag=" 标 题 "  htmlTag="title"  type="xsd:string"
DBAttribute="title" size="64"/>
 <xsd:element name="content" type="ContentType"/>
 <xsd:element  name="author"  tag=" 作 者 "  htmlTag="author"  type="xsd:string"
input="off"/>
 .....
</xsd:element >
 <xsd:complexType name="ContentType">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="picture" type="picType"/ processType="component">
        <xsd:element name="text" tag="正文" htmlTag="text" type="xsd:string"/
DBAttribute="text" size="8192">
      </xsd:sequence>
      <xsd:attribute name="type" type="xsd:string"/>
 </xsd:complexType>
 <xsd:complexType name="PicType">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="title" tag="图片标题" htmlTag="title" type=" xsd:string "/
size="32">
        <xsd:element name="size" htmlTag="size" tag="图片大小"type=" xsd:string
"/size="10">
        <xsd:element name="description" tag="描述" htmlTag="description"
type="xsd:string"/ DBAttribute="description" size="512">
    <xsd:element name="uri" type="xsd:string" create="server">
      </xsd:sequence>
 </xsd:complexType>
.....
.....
```

模板上有多种元素，有的是简单类型，如：标题，只需要一个文本框输入就可以了。有的很复杂，像图片类型，它有很多文本域需要输入，

还需要上传图片。有的元素不需要用户输入，而是系统通过其它方式取得，比如说作者，它是系统通过会话取得的。以上这些信息都包含在模板中，是系统生成输入界面的参考依据。通过转换，将产生如下的 html：

```
.....
<table width="100%">
<Form method="post" action="./action/TemplateAction.do?CTemplate=news">
.....
<tr><td>
标题: <input type="text" name="title" size="64">
</td></tr>
<tr><td>
正文: <textarea name="Content" wrap="VIRTUAL"></textarea>
</td></tr>
<input type="hidden" name="author" value="Liu Ting">
<tr><td>
<IFRAME src="upfileAction.do" frameBorder=0 width="100%" scrolling=no
  height=30></IFRAME>
</td></tr>
.....
.....
<tr><td>
<div align="left"><input type="submit" value="提交">  <input type="reset"
value="重置"></div>
  </td></tr>
</Form>
</table>
.....
```

相应的转换规则定义在 Tcreate-html.xslt 文件中。转换过程与界面定制一节有关定制界面生成中用到的转换过程相同。可以看到，需要用户输入的元素都被转成了文本域。不需要用户输入的元素转为了隐藏域。其中<IFRAME>标签所指向的是一个专门负责文件上传的 html 页面。此页面除了上传文件外还要负责元数据的填写，也是根据文件的类型不同而生成的。下面是自动生成的图片上传页面：

```
.....
<form enctype=multipart/form-data method=post action=TemplateAction.do?type=pic>
<table cellpadding=0 cellspacing=0 width=100%>
<tr><td><IMG src=images/affix.gif>上传图片</td></tr>
<tr><td><input type="text" name="title" size="32"></td></tr>
<tr><td><input type="hidden" name="size" size=""></td></tr>
<tr><td><textarea name="description" wrap="VIRTUAL"></textarea></td></tr>
<tr><td align=right><input type=file name=file size="30">
<input type="submit" value="上传" name=Submit>
```

```
</td></tr></table>
.....
.....
```

可以看到，生成的图片上传页面还包括了图片元数据的填写界面。图片上传后，将作为一种内容组件单独存放。

用户也可以使用多个现成的内容模板来组成自己需要的内容模板。系统使用 include 方式将多个模板添加入一个模板：

```
<xsd:include schemaLocation="./template/0091522.xsd"/>
```

这样模板创建就更灵活、更高效，并且重用性更好，有助于内容组件化的实现。

通过上述机制，可以保证用户通过定制模板来实现新类型内容的结构化、组件化，同时也为界面定制提供保障。

因为系统中三种模板都是以 XML 及其兼容方式描述，所以原则上都可以采用这种上面介绍的方式来创建，但是对框架模板和表现模板来说，这种创建方式并不直观，所以系统也提供了其它方式来创建这两种模板。比如，用户可以使用一些可视化工具，如 frontpage，来编辑模板，同时加入一些识别标签，然后以文件的形式导入系统，系统根据识别标签将模板转为系统可用的格式。

4.4内容类别管理

系统内容是按照一定的内容分类模型组织的，但这样的分类主要是为了方便的对内容进行管理，如果为了内容更好的发布，还要能够动态的对发布内容进行分类，并以内容栏目的方式展现在用户面前，同时还要对每一类内容的发布流程、发布格式等进行相应的定制，这就是内容类别管理所要做的工作。

每一个内容类别在系统中也被称作一个栏目。即每个内容类别对应着一个栏目模块。栏目模块和应用模块一样，可以作为界面元素布置在发布界面上。它也是对应内容类别的入口。对内容的发布权限、发布流程和发布格式的管理应以栏目模块为单位。每个栏目模块的信息都记录在 element-cofig.xml 文件中。下面是一个校内公告栏目模块的信息：

```
.....................
.....
<category>
<id>0013</id>
<name>校内公告</name>
<ContentTemplate id="023" name="notice11" />
<!—栏目所用到的内容模板-->
<PresentTemplate id="02301" name="ForNotice1"/>
<!—内容模板对应的内容模板-->
<ModelInterface>
<!—栏目模块界面信息-->
<type>static-list</type>
<bkcolor>#ff0066</bkcolor>
<fontcolor>#012237</ fontcolor >
<font>14</font>
<related-category-list>Yes</related-category-list>
    <uri>./jsp/module/category/interface/notice1.jsp</uri>
    </ModelInterface>
    <!—数据源-->
    <DataSource TemplateID="02301">校办服1</DataSource>
    <DataSource TemplateID="00701">校办服2</DataSource>
    <DataSource TemplateID="01301">校办服2</DataSource>
    <workflow step="3">
    <!—栏目所对应的工作流-->
    <node>021</node>
    <node>023</node>
    <node>041</node>
    </workflow>
    <BelongTo>通知公告</BelongTo>
    <!—栏目所属的内容分类模型类别-->
    </category>
    .....................
```

可以看到，文件中记录了栏目所使用的模板，以及模板所对应的内容所使用的数据源，还有栏目所对应的工作流等等。

在创建一个类别时，需要指定采用哪一种内容模板来对本类别内容进行结构化、组件化。同时还要指定相应的表现模板来决定以什么样的格式来显示此类内容。

另外，此类内容的工作流程定制也要在这时完成。工作流程是基于内容类别的，系统需要针对每个内容类别制定一个工作流程。在工作流程中，每一个用户组都是一个节点。内容从创建到最终发布这一流程中，要经历若干个节点。这些节点行使着自己的创建、编辑、校对、审核等

工作，以确保把正确的、恰当的内容提供给用户。系统为每个用户组都维护着一个内容列表，其中列出了各个流程中正好流经这个用户组节点的内容。这些内容也是本节点用户所要完成的工作。

栏目模块界面的显示方式也要在这时指定。参照常见情况，系统为栏目模块界面规定了四种显示方式：单列滚动方式，双列滚动方式，单列静止方式，双列静止方式。用户可以从中进行选择，并且还可以选择模块下方是否列出其他相关类别的链接。另外，模块的字体和颜色也是可以设置的。用户对界面的设置都会写入模块配置文件，然后系统会根据配置文件的相关内容生成栏目模块界面的 JSP 页面，放入./jsp/module/category/interface/目录下。类似的实现前文有详细介绍，这里从略。特别说明的是，系统只实现了栏目模块界面的简单定制，即提供几种常用显示方式供用户选择。如果用户希望有更多样的显示方式，则需要对系统生成的 JSP 页面进行修改。

接下来，要指定本类别内容的各内容组件存储所用的数据源：

```
<!—数据源-->
<DataSource TemplateID="02301">校办服 1</DataSource>
<DataSource TemplateID="00701">校办服 2</DataSource>
<DataSource TemplateID="01301">校办服 2</DataSource>
```

因为内容组件和内容模板是一一对应的，所以只要为每种组件所对应的内容模板指定数据源就可以了。接着根据需要还会按照所用内容模板生成相应数据库表。模板元素中 DBAttribute、 size 和 type 这三个属性提供了生成数据表所需要的信息。这样就确定了本类别内容各种内容组件的存储位置和方式。

最后，将栏目模块信息注册到 element-config.xml 配置文件中。这样模块信息就会出现在栏目列表中，在进行界面定制时可供选择。

可以看出，以上工作主要就是将用户请求转化为对配置文件的操作。这类操作的实现在前几节已有详细论述。

针对每个内容类别的工作流程是在创建栏目时由用户定制的，关于工作流程的信息记录在相关栏目模块信息中。每当一个节点的用户将经过加工的内容提交后，系统自动将相关信息加入到下一个节点的用户内

容列表中。内容列表是放在数据库中的，在需要时会将列表提交给相关联的用户。

此外，系统还维护着一个内容树，每个内容类别都是这个内容树上的一个结点，可以通过对相关配置文件的操作来改变内容树。

可以看出，在内容类别管理的实现中，一个任务就是将各类类别定制请求转化为对各类相关配置文件的操作。另一个就是根据应用的内容模板生成相应的数据库表。还有就是，根据设置生成栏目模块界面。在这里有三个最主要的类负责完成以上任务：

- CategoryConfig

负责完成对各类相关的配置文件的操作。这类操作主要是利用 JDOM 的 API 来实现。类似的应用在前面其他部分已有详细论述，此处从略。

- CInterfaceCreate

负责按照用户的设置生成栏目模块界面的 JSP 文件。

- DBMap

负责根据本类别所应用的内容模板生成相应的数据库表。在内容模板中，每个元素所对应的数据库字段名、类型、大小都已有明确说明，只要根据模板中的规定一一映射到数据库表中就可以了。

4.5 内容发布

系统管理的内容要通过内容发布机制来发布。系统的发布界面包括主页和各内容类别的总览页面，这些页面都是由系统根据用户的设定自动生成的，并放在特定的目录中，这在前面已有详细介绍。用户通过发布界面对象要浏览的内容进行定位，然后通过点击链接的方式发出请求。系统根据用户的请求取得相应内容的 XML 文档，然后从 element-cofig.xml 文件中读取相应内容类别的信息，从而得到显示此类内容需要的表现模板 ID。如下列所示：

```
<PresentTemplate id="02301" name="ForNoticel"/>
```

根据 ID 可以取得相应的表现模板。然后根据表现模板设定的格式将其转为 html，并嵌在发布界面中提供给用户。将 XML 根据 XSLT 的描述转

为 html 格式的方法在前面各节已多次用到，这里从略。对于转化为 html 的内容，可以使用 response 对象的 write()方法将其嵌在发布界面中。

4.6 应用模块管理

模块管理负责应用模块的加载、卸载等维护操作。系统有一个专门的模块元素配置文件 element-config.xml，这个文件记录了系统中所有栏目模块和应用模块的信息。下面是一个考试成绩查询模块的配置信息：

```
.....
<model>
<id>0309102018</id>
<name>考试成绩查询系统</name>
<entrance>./jsp/index.jsp</entrance>
<DataSource  type=" RDBMS" >
   <DBDriver>
   oracle.jdbc.driver.OracleDriver
   </DBDriver>
  <ConnString>
   jdbc:oracle:thin:zhhz/zhhz@192.168.140.18:1521:ora9ipro
   </ConnString>
</DataSource>
<description>null</description>
.....
```

配置信息中包括模块的入口页面，使用的数据库类型、驱动，以及连接串等等。

在应用模块导入时，系统会经历如下动作：

1. 将应用模块软件包解压缩。

2. 读取模块的描述文件 module.xml，并将信息写入系统模块配置文件。

3. 将模块软件包按照系统约定和描述文件提供的信息部署。

4. 进行模块初始化

模块导入后就可以在模块列表里看到，进行界面设置时就可以从列表中选择。

以上过程主要由以下几个类合作完成：

- ModuleAction

分析客户端请求并组织各类动作的实施。

- ModuleRegist

负责模块导入系统时的注册工作。

- Moduledeploy

负责模块软件包的解压缩和部署。

- FileMaintain

是一个 Session bean。作为 Façade 类，在前面各部分的实现中已经多次用到。它封装了对各类文件的维护操作，是对文件进行操作的统一接口。

- UpLoad

负责文件的上传。

- Init

是模块内的类，负责部署后的初始化。

首先，得到客户端请求后，ModuleAction 先创建一个 UpLoad 类的实例，再利用其完成模块文件包的上载：

```
SmartUpload mySmartUpload=new SmartUpload();
mySmartUpload.initialize(config,request,response);
mySmartUpload.upload();
```

然后，生成 Moduledeploy 类的实例，并通过其对软件包解压缩，放入新的文件夹。我们使用了一个叫做 DETZipInputStream 的第三方类来解压缩软件包。

```
DETZipInputStream zip = new DETZipInputStream(
                                    new FileInputStream(zipFileName),directory);
```

接下来，利用 ModuleRegist 类的实例完成将模块注册入系统的工作。从技术的角度讲，要完成的工作就是对 XML 文档的读写、修改的操作。和以前一样，主要用 JDOM 的 API 来完成这类操作。系统首先为新的应用模块生成一个系统 ID，然后把模块描述文件中的元素解析出来，再将这些元素写入系统模块配置文件。这部分的实现与前述的对系统配置文件的操作是类似的，这里不再赘述。

下面， Moduledeploy 根据 Module.xml 文件的描述将模块的文件分别放在系统目录的不同地方。系统的表现逻辑和业务逻辑是分开的，所

有和界面有关的文件都放在./Jsp 目录下，而所有与业务逻辑相关的文件都放在./WEB-INF/classes 目录下，所以模块也要按照这一方式来部署。例如，考试成绩查询系统的所有界面文件都要放在./jsp/module/application/01285/目录下，01285 是系统为模块分配的 ID，在为模块建立相应目录时我们使用 ID 作为目录名，以保证其唯一性。而类文件都要放在./WEB-INF/classes/ 01285/目录下，在Module.xml 文件中是这样描述的：

```
<present-logic-directory>./jsp/module/application/SID< /present-logic-directory >
<business-logic-directory>./WEB-INF/classes/SID</ business-logic-directory >
```

在部署时 SID 将被替换为系统分配的 ID。建立目录和拷贝文件的动作由我们通过 FileMaintain 完成。FileMaintain 是一个对文件操作进行了完整包装的 session bean，我们使用其 MakeDirectory()方法来建立目录。方法实现如下：

```
String path=request.getRealPath("");
path=path +subDirectory;//将要建立的目录路径
File d=new File(path);//建立代表 Sub 目录的 File 对象，并得到它的一个引用
if(d.exists()){//检查 Sub 目录是否存在
 return error;
}else{
 d.mkdir();//建立 Sub 目录
```

文件拷贝采用 Copy()方法，如下方式：

```
FileInputStream fi = new FileInputStream(orgFile);
FileOutputStream fo = new FileOutputStream(copyFile);
int content;
while((content=fi.read())!=-1){
   fo.write(content);
}
fi.close();
fo.close();
```

可以看到系统使用了 FileInputStream 和 FileOutputStream 复制文件。

最后，由 ModuleAction 调用模块中 Init 类实例的 initiate()方法来完成模块的初始化工作。比如：建立新的文件目录，建立新的关系表，等等。Init 类固定放在 initialize 目录中，系统会自动到这个目录下将类加载并实例化。图 4-4 表明了模块导入系统的实现。

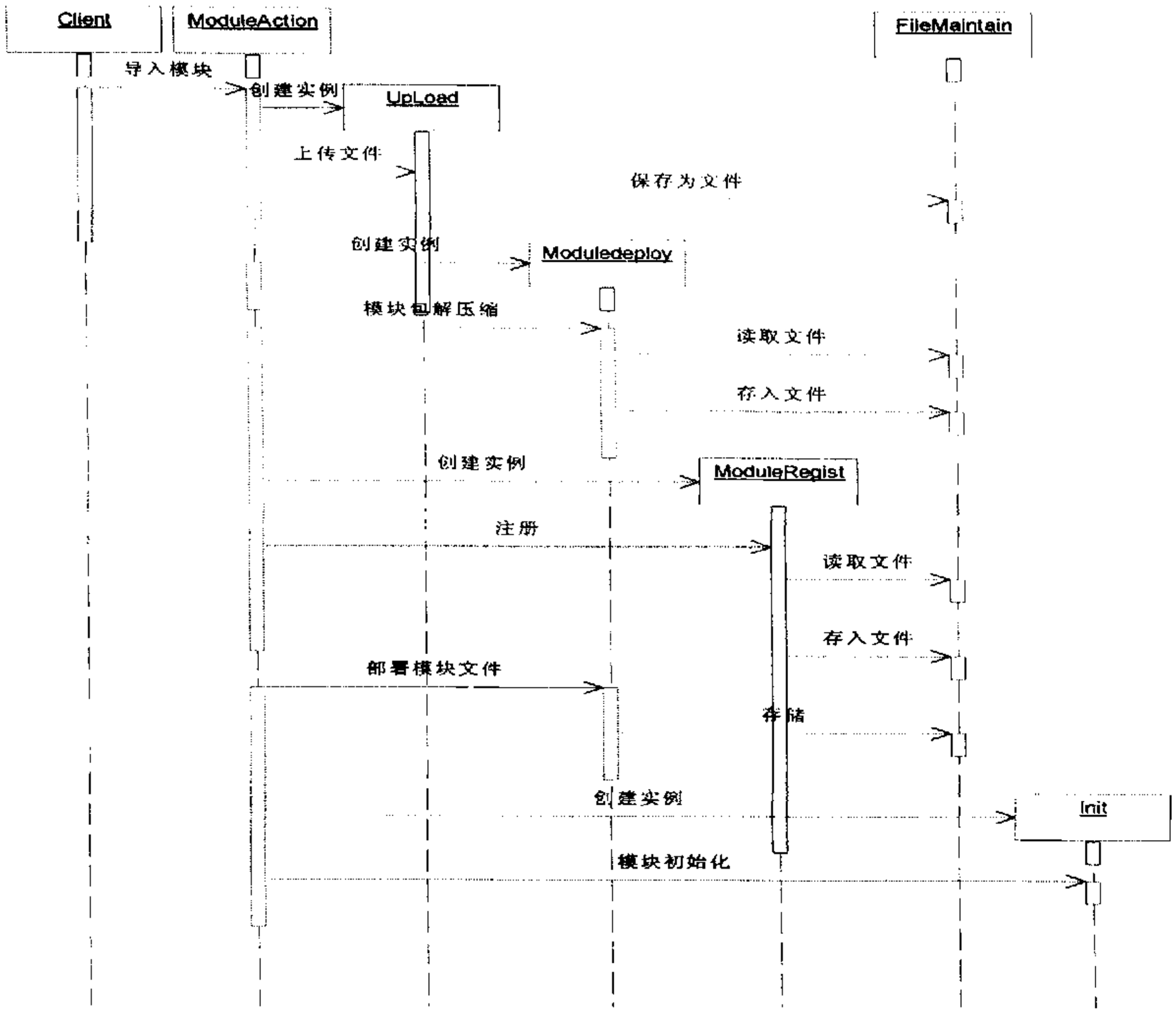


图 4-4 模块管理的时序图

应用模块卸载的实现也与之类似。系统先通过模块注册信息得到所有模块文件和目录的位置，然后将这些文件和目录删除。如果应用模块在系统中建立了自己的数据库表，则这些表单数据将作为备份留在系统中。如果管理员认为其中的数据没有价值，可以选择彻底清除。最后，还要将模块配置文件中相关信息全部删除，同时相关信息也被存储到历史记录中供查阅。

4.7内容存储

内容存储部分负责系统内容和元数据的存储，并对数据源进行管理，对整个系统的成功实现有重要影响。与传统的关系型数据库相比，基于

内容管理的系统的存储管理有自身的特点：

- 内容存储不仅要访问结构化数据，还要管理半结构化和非结构化的数据，这就给数据处理带来了困难。
- 内容存储应当将异质数据源按照统一的标准和规范进行管理，从而使异质数据的存取和共享成为可能。
- 对内容存储的管理来说，内容大多分布在不同的信息系统中，因此无法像数据库一样对数据进行集中存储。
- 内容存储管理的是内容而不仅仅是数据，因此还需要管理大量的元数据。

内容存储管理使用内容模型来对现实世界进行抽象。内容模型就是内容存储中用于提供信息表示和操作手段的构架。本系统采用树型结构的内容模型。有了内容模型后就可以以此为基础来设计内容模式。内容模式的概念与数据库系统的模式概念类似，是内容在逻辑级的视图，是内容存储系统中全部内容的逻辑表示或描述。内容模式是对内容存储结构的一种描述，而不是内容存储系统本身，它是存放内容的一个框架。对于本系统的内容模式，在前面章节已有论述，我们下面将对存储系统的数据源管理、异质数据集成和事务处理作简单介绍。

4.7.1 数据源管理

系统中的内容通常存储于不同的数据源。系统要对内容进行存取操作，首先要将这些数据源管理起来。系统中所有数据源的信息都放在一个叫做 db-config.xml 的配置文件中，文件记录了每个数据源的名称、类型、IP 地址、端口号、数据库连接串、驱动等等信息。系统在对内容进行访问时首先要确定相应的数据源。通过这样的数据源管理方式，我们就可以将内容以内容组件为单位存储在不同的服务器上，提高了内容存储的灵活性。

通过数据源管理的 web 界面，用户可以进行数据源的维护操作。实现这部分功能所要做的就是将用户请求变为对配置文件的操作。这个工

作由DBConfig类来完成。像系统其它部分的实现一样，这里也是采用JDOM的 API 实现对配置文件的操作，这里不再赘述。另外，DBConfig 还有一个 Test()方法，专门用来对数据源进行连接测试。以确认数据源是否可用。

4.7.2 异质数据的集成

系统中的内容存在形式是多样的，既可能是储存在数据库中的结构化数据，又可能是以文件等其他形式存在的半结构化、非结构化数据。即使是同样种类的结构化数据，也可能存储在不同类型的数据库中。在这种情况下，为了确保系统能对内容进行有效的管理，应面向上层访问对象屏蔽数据访问细节，以实现数据的集成。我们在对数据的底层访问中采用了 DAO 模式，从而面向上层实现统一的数据访问接口，实现了一定程度上的异质数据集成。在具体的 DAO 模式应用中，系统使用了抽象代理的方法来实现数据访问对象。抽象代理的方法会创建各种类型的虚拟数据访问对象代理和实际数据访问对象代理，每种对象对应一种持久性存储介质的实现。只要上层对象得到这些代理，就可以利用它们创建所需要的数据访问对象。

首先系统要实现一个最高层次的抽象数据访问对象代理，然后由此抽象类继承出针对系统不同类型数据存取的下一层次的抽象代理。
在系统实现过程中，我们将系统要访问的数据类型从逻辑上进行了归类，提供了针对这些类型数据的抽象代理，如图 4-5 所示。

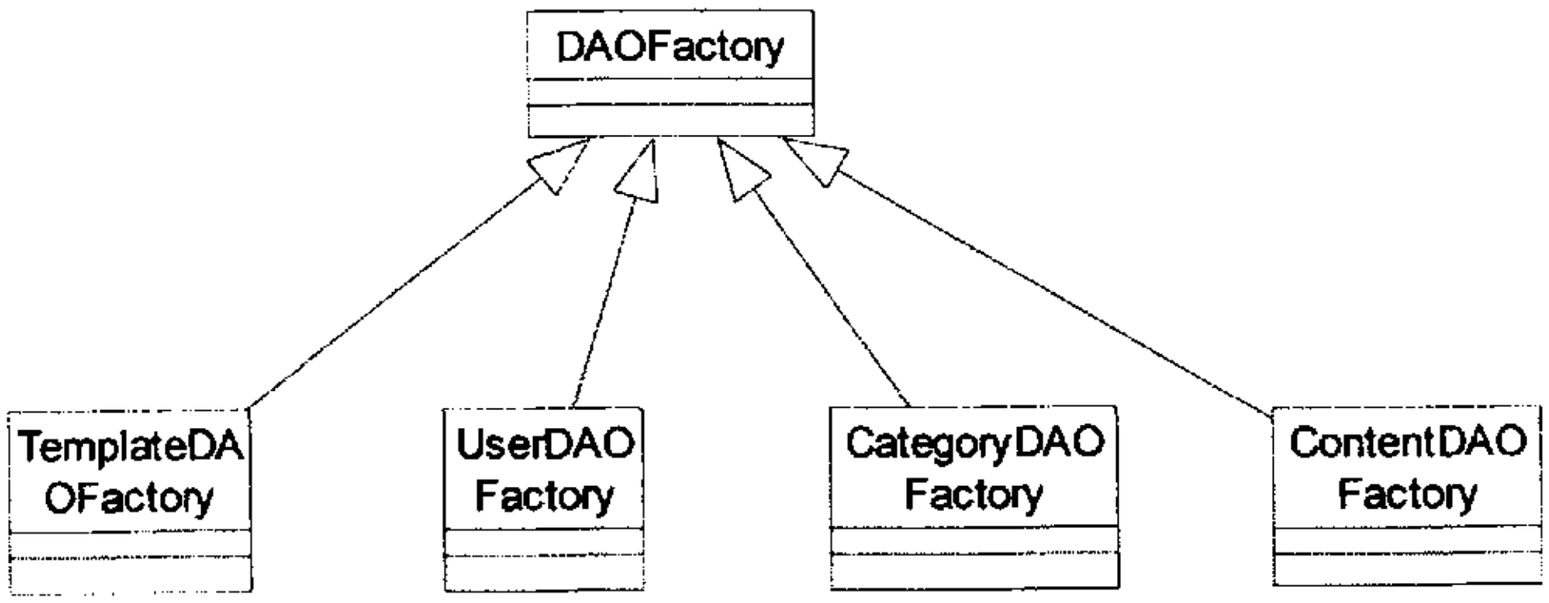


图 4-5 DAO 抽象代理

- TemplateDAOFactory

  模板抽象数据访问对象代理。

- UserDAOFactory

  用户信息抽象数据访问对象代理。

- CategoryDAOFactory

  内容类别抽象数据访问对象代理。

- ContentDAOFactory

  一般内容抽象数据访问对象代理。

图 4-6 给出了系统中有关模板存取的 DAO（TemplateDAOFactory）实现类图。该类图表示了一个基础的模板数据访问对象代理，它是一个抽象类，被其它一些实际的数据访问对象代理继承，以支持特定的模板数据访问函数。用户可以得到一个实际的模板数据访问对象，并利用它来创建需要的数据访问对象而访问相关数据。每个实际的数据访问对象都负责建立数据源连接，得到并管理所支持的业务数据。

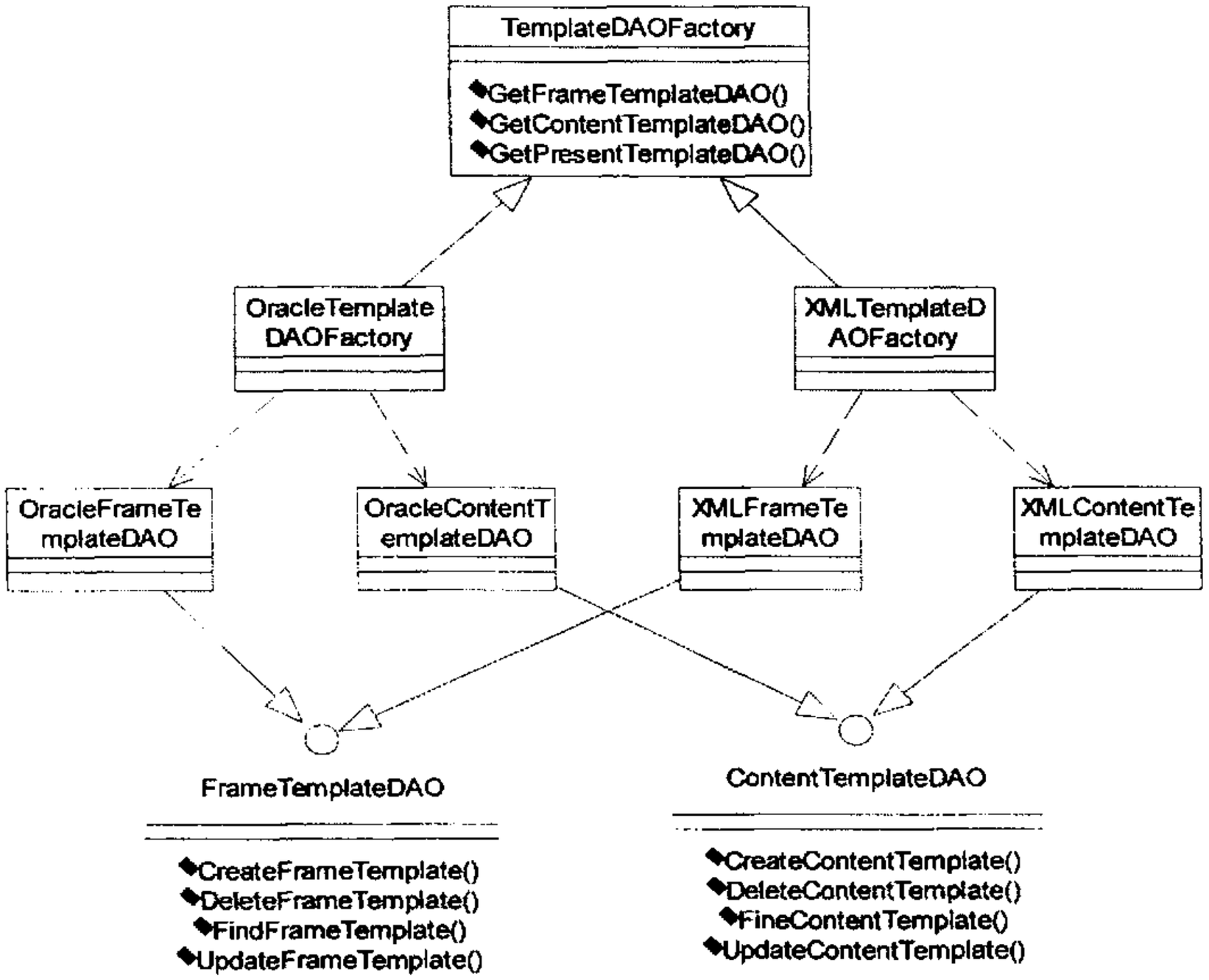


图 4-6 DAO 模板存取实现类图

下面来介绍一下有关模板存取的 DAO 实现。对于模板的存取来说，我们主要针对 oracle 数据库、XML 文档实现了相应的 DAO。如有需要也可以很容易的实现针对其它类型数据源的 DAO，并加入进来。

首先是实现有关模板操作的抽象类：

```
//抽象类 Template DAO Factory
public abstract class TemplateDAOFactory {
  // 代理支持的数据源类型
  public static final int ORACLE = 1;
  public static final int XML = 2;
  ...
  // 下面是所有继承 TemplateDAOFactory 类的子类都要实现的方法
  public abstract FrameTemplateDAO getFrameTemplateDAO();
  public abstract ContentTemplateDAO getContentTemplateDAO();
  public abstract PresentTemplate getPresentTemplateDAO();
.....
```

这个抽象类针对三种模板的存取分别建立了三种抽象方法。下面是针对 Oracle 数据库的工厂类 OracleTemplateDAOFactory 的实现，其他诸如 XMLTemplateDAOFactory 类的实现都是类似的.

```
// OracleTemplateDAOFactory 类的实现
import java.sql.*;
public class OracleTemplateDAOFactory extends TemplateDAOFactory {
  public static final String DRIVER=
    "oracle.jdbc.driver.OracleDriver";
  public static final String DBURL=
    "jdbc:oracle:thin:zhhz/zhhz@192.168.140.18:1521:ora9ipro";
  // 创建 oracle 连接
  public static Statement createConnection() {
    Class.forName(DRIVER);
          mabenBase=DriverManager.getConnection(DBURL);
          statement=mabenBase.createStatement();
         return statement;
  }
public FrameTemplateDAO getFrameTemplateDAO() {
    // OracleFrameTemplateDAO 实现 FrameTemplateDAO
    return new OracleFrameTemplateDAO();
  }
//下面其他类型模板 DAO 的 get 方法都略去
.....
```

在这个类中，getFrameTemplateDAO()、getContentTemplateDAO()等方法被实现，返回 oracleFrameTemplateDAO 等对模板操作的具体类的实例。createConnection()用来做数据库连接。

然后为模板操作建立 DAO 接口：

```
// 所有对框架模板操作必须实现的接口
public interface FrameTemplateDAO {
  public int CreateFrameTemplate();
  public boolean DeleteFrameTemplate();
  public Customer FindFrameTemplate();
  public boolean UpdateFrameTemplate ();
  ...
}
```

接口中的方法包括模板的创建、修改和删除等维护操作。

接下来是 OracleFrameTemplateDAO 类，这个类实现了 FrameTemplateDAO 接口，是负责直接对数据源进行操作的类。这个类封装了所有对 oracle 数据库的操作，对上层应用来说数据访问的细节被屏蔽掉了。

```
import java.sql.*;
public class OracleFrameTemplateDAO implements
    FrameTemplateDAO {
  public OracleFrameTemplateDAO () {
    // 初始化
  }
  // 下面的方法可以使用 OracleTemplateDAOFactory.createConnection()
  // 方法来得到一个数据库连接
  public FrameTemplate FindFrameTemplate(FID) {
      Statement statement;
      Statement=OracleTemplateDAOFactory.createConnection();
      String findStr="select * from FrameTemplate where fid = '"+FID+"'";
     ResultSet results=statement.executeQuery(findStr);
//下面的其他对框架模板进行操作的方法略去
  ..................
```

其中 FindFrameTemplate(FID)方法是用来查找框架模板的。可以看到方法中包含了对数据库操作的实现细节。

最后我们再看一下框架模板的传输对象：

```
public class FrameTemplate implements java.io.Serializable {
  // 成员变量
  int FTemplateID;
  String FTname;
  String FTbody;
  String description;
  ...
  // 下面是取值和设值的函数从略...
  ...
}
```

这里传输对象被用来作为传送框架模板数据的载体，包括框架模板的各种属性。

图 4-7 显示了存取框架模板和内容模板的过程。

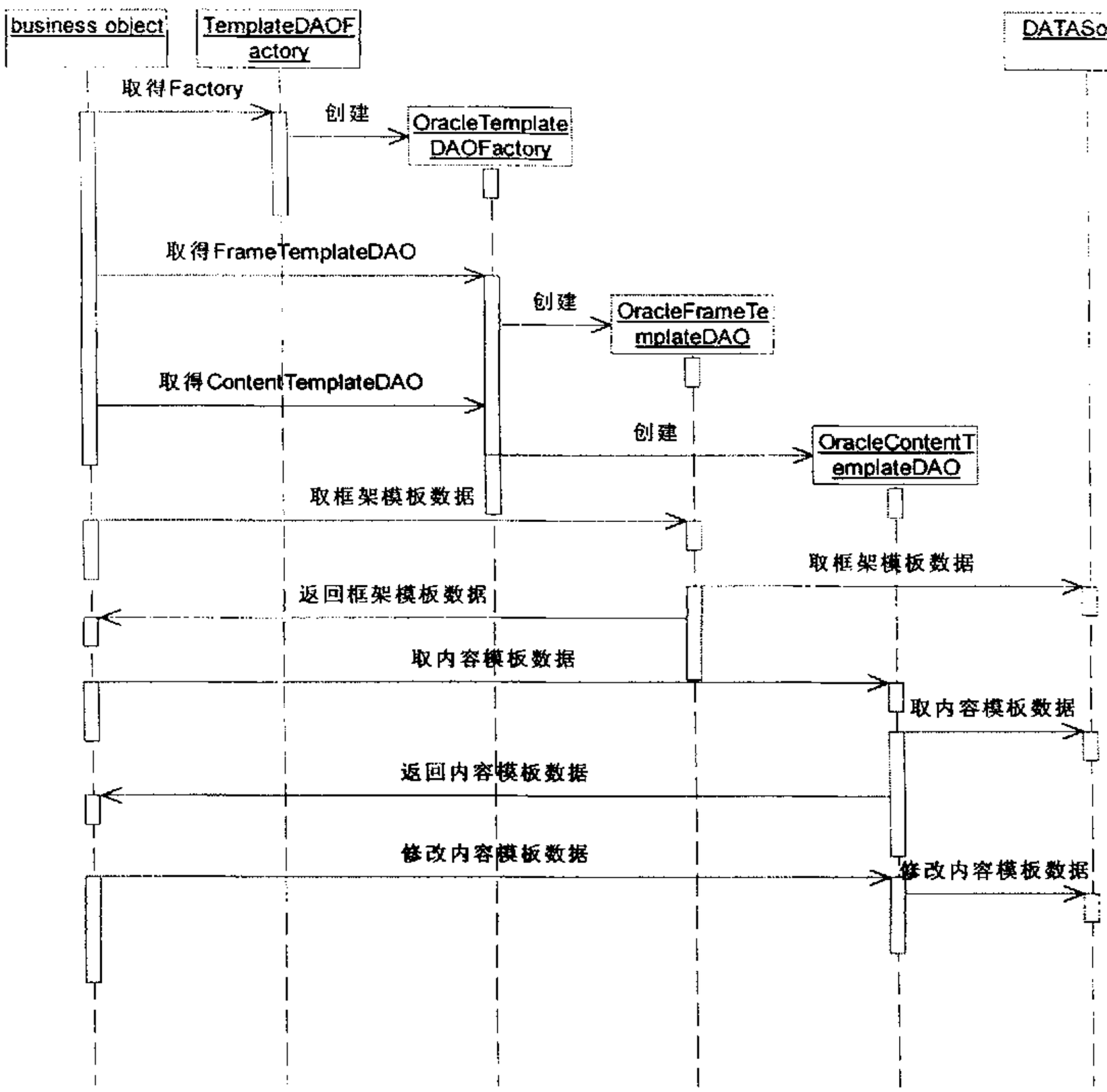


图 4-7 DAO 模板存取实现时序图

4.7.3 事务管理

系统运行时常会碰到因断电或磁盘崩溃等造成的非正常中断，或者很多请求同时访问同一组数据的情况。在这种情况下，要保持数据的完整性必须引入事务管理。事务有两种管理方式，一种是容器管理方式，即指其事物边界由 EJB 容器来划定。另一种是 bean 管理方式，即指由程序员来写事务管理逻辑。容器管理事务的方式可以减少一定的代码量，

但是它也具有一定的局限性，当一个方法执行时，他将只与单一的事务相关联，或与任何事务均不相联。这使得开发人员很难设计相关 bean 的事务模式。而我们在开发系统时，希望事务能跨越多个 DAO 和多个数据库。这样我们就要选择 bean 管理方式自己来写事务管理逻辑。

DAO 是事务性对象。一个典型的 DAO 执行创建、更新和删除之类的事务性操作。在 DAO 中有两种主要的界定事务的策略。一种方式是让 DAO 负责界定事务，另一种将事务界定交给调用这个 DAO 方法的对象处理。如果选择了前一种方式，那么就将事务代码嵌入到 DAO 中。如果选择后一种方式，那么事务界定代码就是在 DAO 类外面。如果需要在一个事务中访问多个 DAO，后一种方式最合适。

在设计有关事务界定部分时，我们遵循了以下思路：

- 事务何时开始
- 事务何时结束
- 哪一个对象将负责开始一个事务
- 哪一个对象将负责结束一个事务
- DAO 是否要负责事务的开始和结束
- 应用程序是否需要通过多个 DAO 访问数据
- 一个 DAO 是否调用另一个 DAO 的方法
- 是否有涉及到多个 DAO 的事务操作

在具体实现过程中我们使用了 Java 事务应用程序接口（JTA)。要用 JTA 进行事务界定，应用程序要调用 javax.transaction.UserTransaction 接口中的方法。首先要对 UserTransaction 对象进行 JNDI 查询，当应用程序找到了 UserTransaction 对象后，就可以开始事务了。应用程序调用 begin() 开始事务，调用 commit() 或者 rollback() 结束事务。

4.7.4 Cache 的实现

在系统中，内容是由 XML 来描述的，XML 读入内存进行处理时，要进

行实体的替换和有效性检查等工作。而且还要进行格式转换。在这个过程中所用到的以 XML Schema 为描述语言的内容模板和 XSLT 表现模板需要被频繁调用，但这类数据变动并不频繁。如果每次用到时，都要重复加载并完成上面提到的一系列动作，这样会大大降低系统性能。所以必须要有一定的缓存机制来支持。

本系统采用了一种较简单的方式来实现缓存机制。首先要建立 hashMap 来作为分类存放数据对象的数据结构。存放的数据从逻辑上来说其实是一个个四元组：（资源 ID, 资源类型, 数据对象, 更新时间）。

然后我们采用了一个简单的算法来确保缓存中的数据始终是最新，图 4-8 是实现算法的流程图。

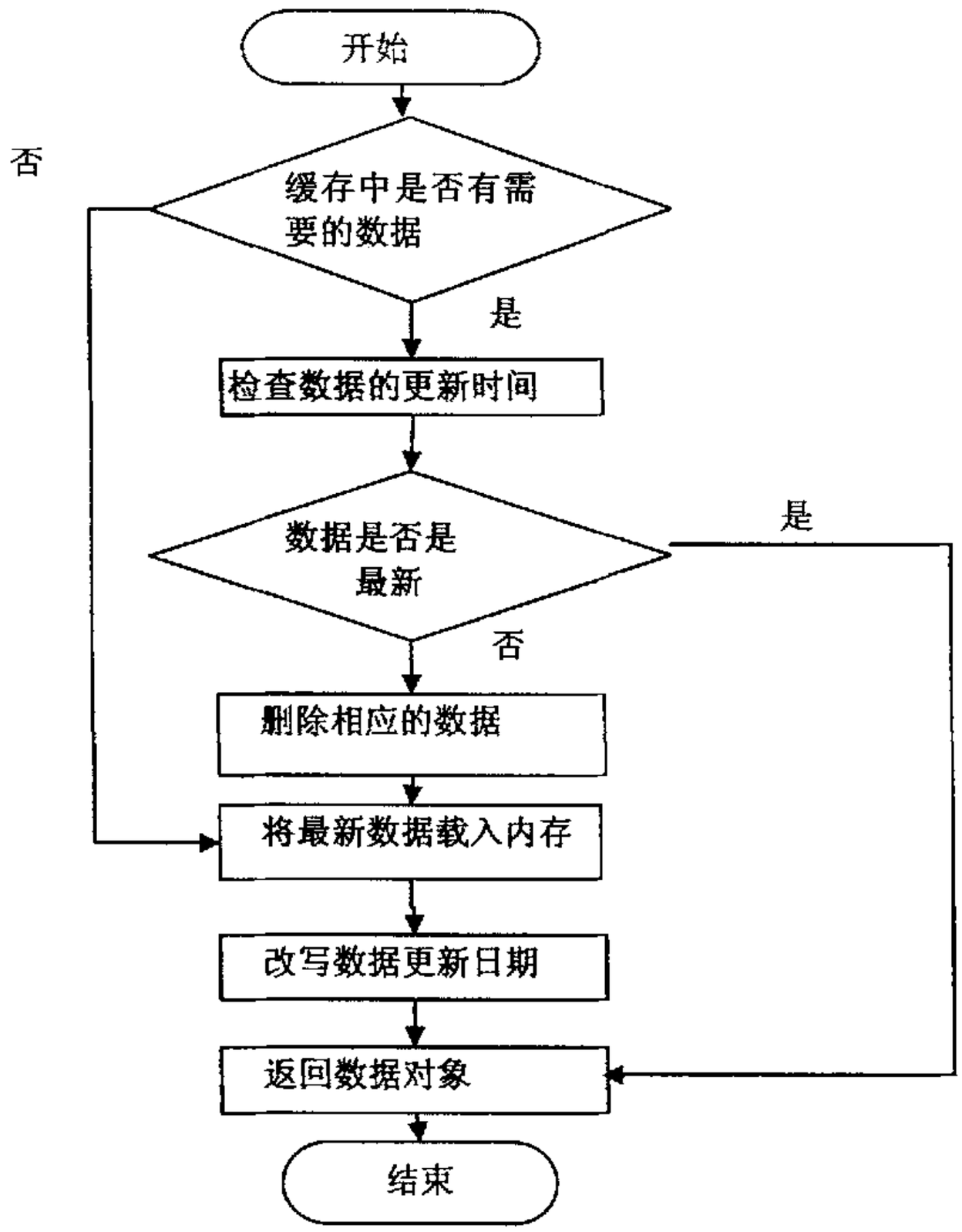


图 4-8 Cache 机制的算法流程图

第5章 系统应用举例

5.1 界面定制

界面设置方式非常简单，点击添加元素的链接会弹出新的窗口来选择加入什么界面元素。选定后新的模块就加入框架模板中了。下面介绍一下主页定制过程。

首先，选择要使用的框架模板并提交，然后会弹出相应的设置界面。在本例中，选择了两列式框架模板。如图 5-1 所示。

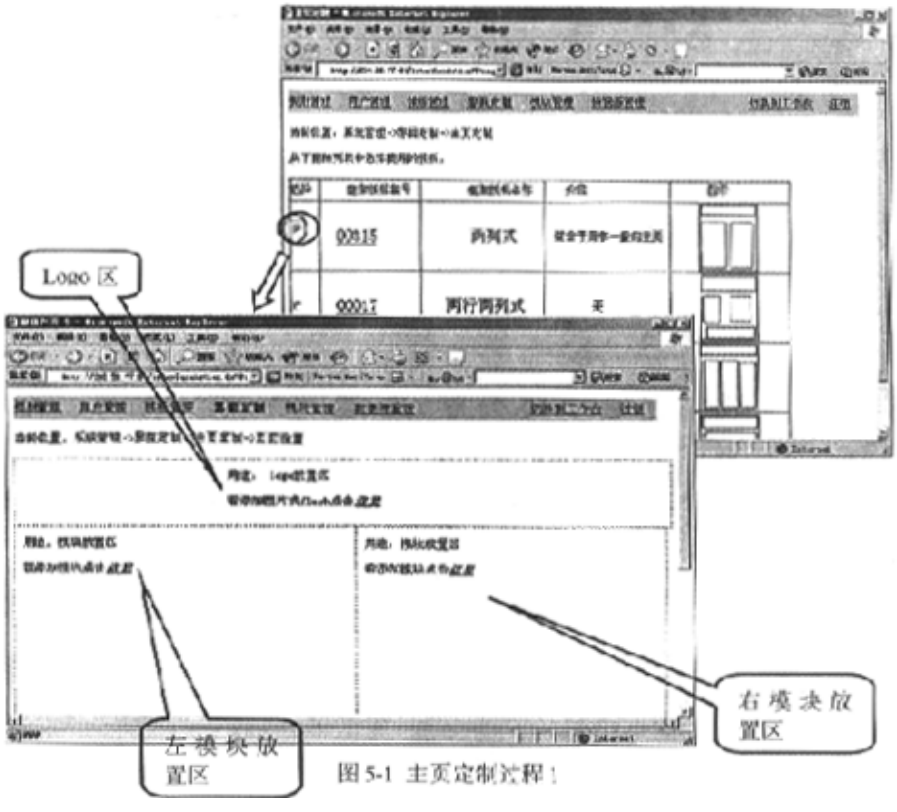


图 5-1 主页定制过程

然后，点击要设置区域的链接会弹出模块选择界面。首先选择元素类型，然后从相应类型的模块列表中选择。如图 5-2 所示。

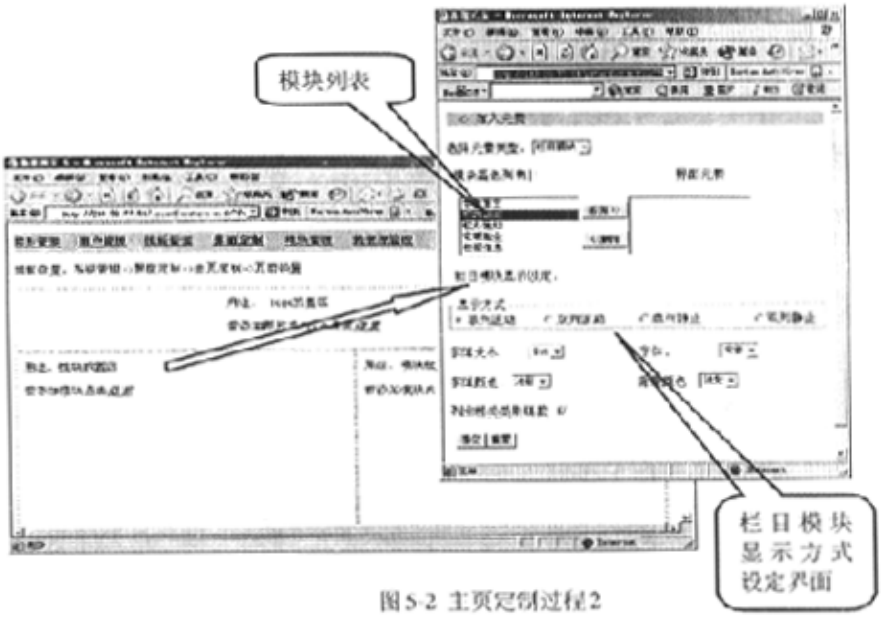


图 5-2 主页定制过程2

下一步，选择好所要加入的模块并提交，模块会出现在相应的位置上。如图 5-3 所示。

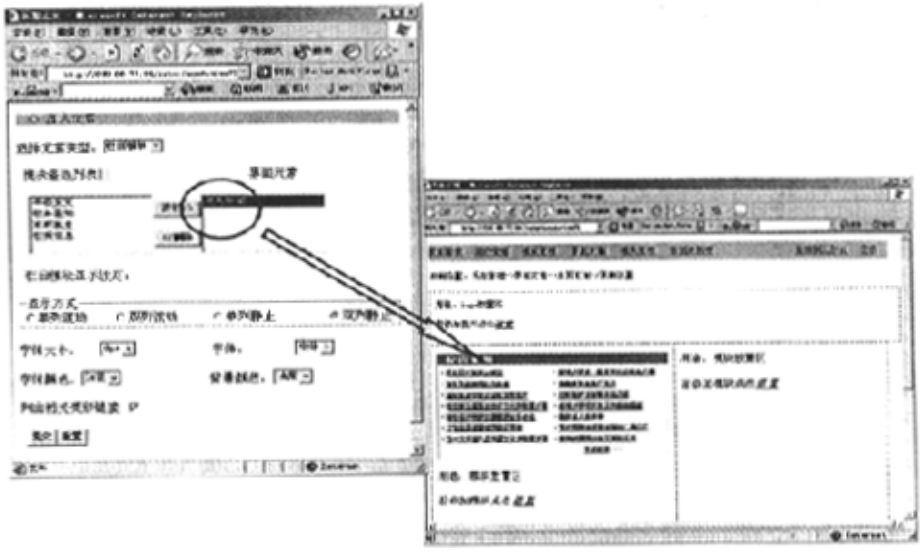


图 5-3 主页定制过程 3

以这样的方式将界面的所有区域都设置好，然后点击保存，界面就摄制完毕了。最后点击“应用”按钮，界面就正式使用了。主页将以新的面貌展现在用户面前。如图 5-4 所示。

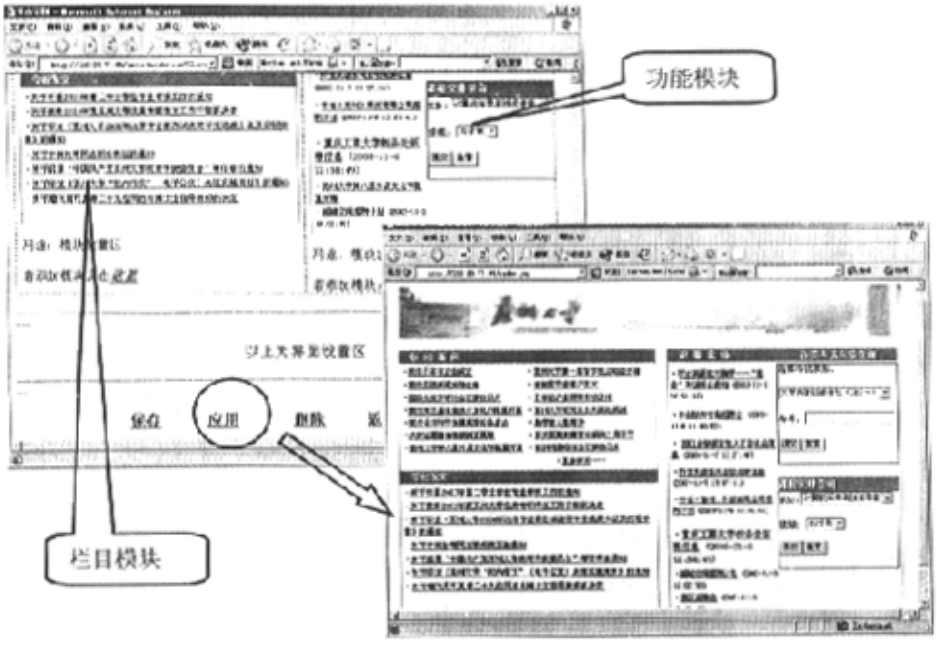


图 5-4 主页定制过程

5.2 模板管理

对模板管理功能来说，最重要的是模板创建功能。其中比较复杂的是内容模板的创建。内容模板本身是 XML Schema 描述的。而 XML Schema 本身也是 XML。所以以树型结构来表示内容模板是再合适不过了。图 是内容模板的初始创建界面。一开始只有一个根节点，我们可以通过向模板树中添加各类结点来完成模板的创建。从内容模板创建界面可以看出，我们可以通过创建内容模板中的结点并添加属性及其类型的方式来创建自己的模板。同时，我们还可以利用现成的内容模块模板来组合成自己想要的模板。在创建模板的同时也提供给系统创建模板对应数据库表的必要信息。这样，在需要的时候，可以将内容信息映射到数据库中存起

来。图 5-5 是内容模板创建的初始界面。

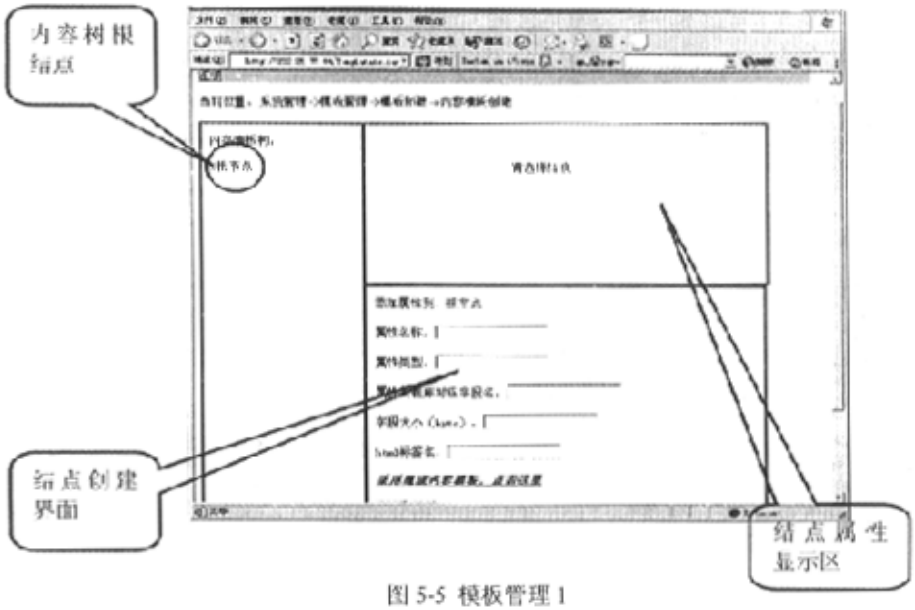


图 5-5 模板管理 1

图 5-6 是模板树创建完成的状态，只要点击“保存”就可以使用了。

图 5-6 模板管理 2

第6章 结束语

本文详细介绍了某大学基于内容管理的web信息发布系统的设计与实现。所有的工作都是围绕以下几点来进行的：

首先，内容信息是以基于一定内容模型的组件化形式来管理，并且要建立对内容进行描述的原数据框架，以充分实现细粒度的内容重用和保证内容检索、加工、传递的自动化进行。

其次，系统应用应该模块化，并辅以一定的集成机制，以保证系统的扩展性。

再次，包括系统界面在内的许多系统属性都应是可定制的，以使得系统具有高度适应性。

最后，系统必须能将各类异质数据集成起来以确保对各种半结构化、非结构化内容的管理。

这几点也是系统实现中的重点和难点。我们在进行系统开发的过程中，对学校常见内容类型进行了抽象，建立了一整套内容模型。并且利用模板机制实现了内容的创建、组件化和描述。另外，利用模板机制并结合应用模块机制我们还实现了系统发布界面和系统应用的可定制。而且我们建立了较好的内容存储机制，不但通过应用 DAO 模式实现了异质数据集成，而且还通过应用 JTA 实现了跨越多个 DAO 和数据库的事务处理。

目前，这一系统已经应用在某大学校园网中，并且在学校的日常运作中发挥了很大的作用。但同时，我们也感到本系统还有很多需要完善的地方：

首先，本系统不是一个完整的内容管理平台。它并没有实现第二章所谈到的内容管理应用体系结构中所有功能。虽然它可以通过添加新应用模块的方式来进一步完善。但对于复杂的、跨模块的、需要协同工作的应用来说，在目前的应用模块机制下还难以支持。

其次，在实现应用集成方面做得还不够。目前本系统还没有实现对

其他 EIS 的集成。

再次，系统的通用性还需要进一步增强。虽然系统的很多方面都可以由用户来定制，但不得不承认它还是面向某种应用的。对于内容管理方式和应用方式与目前有较大差异的情形，本系统可能不能很好的满足需要。

最后，系统还不能提供比较直观的方式来创建表现模板，目前只能以文本输入的方式或借助可视化网页编辑工具来创建。

虽然系统还有很多需要完善之处，但系统的设计思路和方法是有特色的，作进一步改良后可以达到更好的效果。

参考文献

[1]中文内容管理白皮书，http://www.trs.com.cn/，2001 年 9 月

[2]企业门户关键技术研究—内容管理系统的设计与实现，北京航空航天大学研究生毕业论文，2001 年 2 月

[3]企业信息门户中内容管理的关键实现技术研究，北京航空航天大学研究生毕业论文，2001 年 2 月

[4]《XML 技术手册》，elloitte rusty 著，中国电力出版社，2001 年

[5]http://www-900.ibm.com/developerWorks/cn/java/j2ee/index.shtml，J2EE 简介，刘湛，2001 年 7 月

[6]《Java 网络程序设计：J2EE》，蔡剑、景楠著，清华大学出版社[北京]，2003 年 6 月

[7]《J2EE 核心模式》，Deepak Alur 等著，牛志奇等译，机械工业出版社[北京]，2002 年 1 月

[8]《J2EE 编程指南》，Subrahmanyam Allamaraju 等著，马树奇译，电子工业出版社[北京]，2002 年 3 月

[9]《Java 与 XML》，Brett McLaughlin 著，孙兆林、汪东、王鹏译，中国电力出版社[北京]，2001 年 4 月

[10]《XML Schema 数据库编程指南》，Chelsea Valemtine，Lucinda Dykes，Ed Tittel 著，毛选、魏海萍等译，电子工业出版社[北京]，2002 年 5 月

[11]《XSLT 从入门到精通》，Chuck White 著，王建、王军等译，电子工业出版社[北京]，2003 年 1 月

[12] Step two Designs 网站中关于 CMS 的所有短文。

[13]《Enterprise java with UML 中文版》，CT Arrington 著，机械工业出版社 2003 年

[14]《Java 与模式》，阎宏著，电子工业出版社[北京]，2002 年 10 月

[15] 《oracle 与 Java: 从客户/服务器到电子商务》, Elio Bonazzi 等著, 贺民等译, 清华大学出版社[北京], 2003 年 8 月

[16] 《Java2 核心技术卷 I》, Cay S. Horstmann 等著, 程峰等译, 机械工业出版社[北京], 2003 年 10 月

[17] 《Java2 核心技术卷 II》, Cay S. Horstmann 等著, 程峰等译, 机械工业出版社[北京], 2003 年 1 月

攻读学位期间公开发表的论文

[1] 李国柱 吕强 杨季文，网络数据库应用程序的多线程解决方案，《计算机工程与科学》 已录用

致谢

写到这里，意味着三年学习生涯即将画上圆满的句号。回想起几年来在这里生活学习的点点滴滴，不禁感慨万千。我是幸运的，是周围很多人的关心和帮助让我顺利地走到了今天。我将带着获得的满足感离开，同时带走所学的知识和老师的教诲，还有的就是沉甸甸的友谊……

感谢我的导师吕强教授几年来对我的关心和指导，您的很多话让我刻骨铭心，您对我们的期望是我今后继续努力奋斗的动力。

感谢李培峰老师的悉心指导。您指导论文时的耐心和精益求精让我感动。

感谢杨季文教授，您既严厉又富有人情味的风格，让我难以忘怀。

感谢查伟忠老师对我的热心帮助和指导。

感谢师姐 Phoenix 和吴娴，你们在方方面面都给了我很多的关心和帮助，我永远不会忘记。

感谢我的同学瓜瓜、李浩、TDC、侯靖、郁春江、郭胜超、老段、宝雷、晓静、皮皮、朱麟和高洁羽，还有各位师弟师妹，难忘大家在一起的日日夜夜。

最后特别感谢我的父母，是你们给了我学习和生活上莫大的支持和鼓励。