第 37 卷 第 21 期
Vol.37 No.21
计 算 机 工 程
Computer Engineering
2011 年 11 月
November 2011
·开发研究与设计技术· 文章编号：1000—3428(2011)21—0255—03 文献标识码：A 中图分类号：TP311.5

基于 MVC 模式的分页组件应用

张 俐
(江苏技术师范学院计算机工程学院，江苏 常州 213001)

摘 要：针对 Web 数据库系统中的数据分页问题，提出一种基于模型-视图-控制器(MVC)模式的三层分页组件。在数据层中引入 Java 持久化 API 和 Spring 技术，在业务逻辑层中引入泛型 DAO 模式和桥接模式，在显示层中引入 JSP 和 Struts2 标签，并给出部分程序源代码。应用结果表明，与其他分页方法相比，该模型的执行效率较高，稳定性和移植性较好。

关键词：分页组件；模型-视图-控制器模式；Struts2 框架；Spring 框架；Java 持久化 API

Application of Paging Component Based on Model-View-Controller Pattern

ZHANG Li
(College of Computer Engineering, Jiangsu Teachers College of Technology, Changzhou 213001, China)

【Abstract】Focusing on the question of magnanimous database pagination in Web-based system, this paper proposes a three layers of page component based on Model-View-Controller(MVC) pattern. The implementation includes Java Persistence API(JPA) and Spring as model, GerneticDAO and Bridge pattern as control, JSP and Struts2 tags are applied as view. It presents a short program source codes and the model is proved to be practicable and effective. The practice shows that the page component is easy to reuse, learn and deployment.

【Key words】paging component; Model-View-Controller(MVC) pattern; Struts2 framework; Spring framework; Java Persistence API(JPA)
DOI: 10.3969/j.issn.1000-3428.2011.21.087

1 概述

在基于 Web 企业级应用开发中，必须要涉及到查询数据的显示。通常做法是客户一次从服务器中查询他所要的数据并把它显示到客户端，但这种方法的问题是它减慢了 Web 服务器的响应速度和增加了服务器通信负载，不利于客户浏览效果。因此，有必要采用分页技术。文献[1]从分页架构的角度提出了分页模型的 3 种类型：上下型，窗口型和随意型，但它仅局限于理论上的探讨，并没有付诸于实施。文献[2]提出了基于 Struts 框架技术实现 Web 数据库查询分页显示处理一种解决方案，但该方案仍存在程序耦合度高、维护不便和可扩展性差等方面的不足。

针对上述问题，本文提出组件式的开发方式，利用模型-视图-控制器(Model-View-Controller, MVC)模式、泛型、反射技术、Struts2 框架、Spring 框架和 Java 持久化 API(Java Persistence API, JPA)规范实现一种简单高效的 Web 分页显示方法，从而满足分页组件的可扩展性和可移植性的要求，达到显示逻辑和业务逻辑的分离、代码重用率高和易于维护等的目的。

2 MVC 模式及相关技术

MVC 模式[3]是一种目前较为流行的设计模式，它包括 3 类组件：(1)View 组件表示数据输入或输出；(2)Model 组件描述数据业务逻辑和业务处理的过程；(3)Controller 组件则是表示定义 View 组件对调用 Model 组件的响应方式。在基于 Model 组件的开发中会大量使用泛型和反射的技术，因为泛型技术允许开发人员将一个实际的数据类型的规约延迟至泛型的实例被创建时才确定，它为开发者提供了一种高性能的编程方式，能够提高代码的重用性。反射技术允许程序能在运行时得到类的属性、方法的信息。这种特性使程序能随着需求变化而自我演化。

3 基于 MVC 模式和 S2SJ 的分页组件架构设计

文献[4]提出了一种基于 J2EE/JavaEE 平台的 RADF 多层设计框架，其存在的问题是在多层开发中，层与层之间的调用效率非常低效，而且学习难度高。因此，为了能尽可能地将分页组件做到易于复用，提高学习性和维护性，突出逻辑和业务逻辑之间连接简单并且耦合度低等特点，本文在基于 Web 企业级分页组件应用开发中，采用目前流行的 MVC 模式、Struts2、Spring 和 JPA 来构建三层分页组件框架。同时，把从数据库查找到数据封装成一个 Page 类，使其作为数据之间的传递对象。具体如图 1 所示。

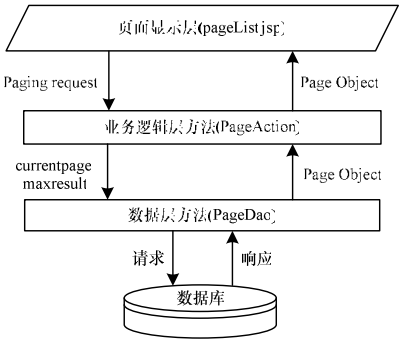

图 1 基于 MVC 模式的分页组件架构

以下按基于 MVC 模式的 3 层结构由上至下分别描述分

基金项目：江苏省高校自然科学基础研究基金资助项目(08KJD520005)
作者简介：张 俐(1977－)，男，讲师、硕士，主研方向：JavaEE 技术，软件架构分析
收稿日期：2011-05-11 **E-mail：**zhangli_3913@yahoo.com.cn

页组件中各层之间的功能。

(1)页面显示层是向业务层发出需要显示某页数据集请求，在该数据集请求中包括需要显示的总共记录数、共要显示多少页、每页的最大数以及当前位于第几页(包括上一页、下一页和最后一页等)，即要实现文献[1]中的上下型和随意型分页模型。为了增强显示层的通用性和可移植性，拟采用 JSP 规范中使用的标签库来包装数据集请求对象。因为，它可以使页面显示和业务逻辑彻底分离并且也使得页面显示功能不具体绑定在某个显示框架下面。最后，为了增加页面效果拟采用 JavaScript 完成页面组件动态的显示效果，好处是它可以简化了客户端负载重荷。

(2)业务逻辑层采用 Struts2 中的 Action 类进行业务组件的处理。因为在 Action 中，Page Object 属性可以直接被映射为一个 POJO 类，因此就不在需要像 Struts 中那样创建在一个单独的数据进行传输对象了，它的好处是维护工作简单高效，同时，移植性得到了大大的增强。最后，只需要在 Action 类中，定义一个方法去处理 Page Object 的请求，并把分页处理结果集返回给显示层。

(3)数据层主要是向数据库发送查询分页语句即就是完成 List 分页查询结果和查询总记录数，并把查询出的分页结果返回给业务逻辑层。

4 基于 MVC 模式的分页组件实现

4.1 传递数据 Page 类的定义与描述

用 Page 类来处理分页所需要的一些关键数据，即描述需要分页查询的数据。具体定义如下：

```
private int pageSize=10;   //每页显示的行数，默认是 10 条记录
private int curPage=1;   //当前页
private int totalPage=1;   //总页数
private int totalRecord=0;   //总记录数
private List<T> resultlist;   //返回 List 查询结果，同时生成以上
//属性的 get/set 方法
public Page getPage(String curPage, int totalRecord, int pageSize)
//返回从 Page 对象中得到的分页查询结果和总记录数，以及计算得
//出的总页数。
```

4.2 数据层中 JPA 和 Spring 分页方案的实现

4.2.1 选择组件化 JPA 和 Spring 的原因

因为传统的 O/R 映射解决方案过于太多，不方便开发人员的学习和使用。而新的 ORM 框架 JPA 提供了数据访问的统一规范和标准。JPA[5]包括以下优势：(1)标准化；(2)简单易用，集成方便；(3)可媲美 JDBC 的查询能力。使用 Spring 具有以下 3 个方面的好处：(1)Spring 可以帮助解决统一资源的管理，这样使可移植性优化变成了可能；(2)Spring 可以把 JPA 的数据访问异常统一纳入到 Spring 的异常层次体系中，使开发人员不需要过多的关注数据访问技术；(3)Spring 可以统一管理数据访问事务管理及其控制方式，可以避免需要人为因素的错误，重用性得到提高。因此，考虑到需保证分页组件的学习性、移植性和查询效率，本文采用 JPA+Spring 的分页方案。

4.2.2 组件化 JPA 和 Spring 的实现

JPA 和 Spring 的架构整合步骤如下[3]：

(1)使用 Spring 中 XML 配置的方式获取 EntityManager Factory 连接工厂资源的配置和创建，目的使它能够很好集成到 IOC 容器中，这就可以使用 JpaTemplate 模板方法类来进行数据访问的处理。

(2)在 Spring 配置文件中进行事务管理方面的配置，这里采用声明式事务进行管理，它可以最少影响应用代码的选择。一般是在 TransactionProxyFactoryBean 设置 Spring 事务代理，同时，选择一个目标对象并把它包装在这个事务代理中。而这个目标对象一般是一个普通 Javabean。具体事务属性定义如下：

```
<props>
<prop key="insert*">PROPAGATION_REQUIRED</prop>
<prop key="update*">PROPAGATION_REQUIRED</prop>
<prop key="*">PROPAGATION_REQUIRED,readOnly</prop>
</props>
```

(3)查询数据。该过程较为复杂，通常是对于某个实体类采用循环迭代的方式进行，代码要求过于复杂，而且，一次从数据库检索所有实体，又有可能消耗大量内存。因此，JPA 提供在结果中集合分页的功能，使用 setFirstResult 方法设置结果集中第一个结果的位置，同时，使用 setMaxResults 方法获得最大实体数量，这样即可获得该类型实体的分页结果列表。考虑到每个实体都要进行分页和代码重用的考虑，本文采用泛型和反射技术加已解决。具体步骤如下：

1)为减少应用分页组件 DAO 模式编程的代码量。定义一个泛型分页 Dao 模板类，方法如下：

```
public <T> Page<T> getListData(Class<T> entityClass, Page
pager);
```

2)编写泛型分页 DAO 的抽象实现类 GenericDAO 类，方法如下：

```
public abstract class GenericDAO implements DAO{
//动态获取实体类名
        String entityname=entityClass.getAnnotation(Entity.class);
//获取分页查询到的结果集
Query query=em.createQuery("select o from"+entityname+"o");
query.setFirstResult((page.getPage()-1) * page.getPageSize()+1)
         .setMaxResults(page.getPageSize());
        qr.setResultlist(query.getResultList());
        //记录获取总记录数
query=em.createQuery("select  count("+getCountField(entityClass)+")
from "+entityname+"o");
        qr.setTotalrecord((Long)query.getSingleResult());
        return qr;
```

将结果集封装成 List 对象进行返回，因为它能够节省服务器内存资源，从而达到提高系统的性能，并且过滤多余的数据，得到真正意义上所需的数据。

4.3 业务逻辑层的桥梁模式

为了将 Action 类中分页信息与当前查询实体类相关的某个具体 DAO 类分离，本文提出“泛型+反射+桥梁”模式，它不同于文献[2]提出的基于“IDAO+DAO”的实现模式。因为该模式灵活性不够，它一开始便把抽象化角色和实现角色的关系给绑定了，使得 IDAO 和 DAO 之间相互限制。而本文提出来的模式是一种弱耦合的方式，它在 GerneticDAO 和某个具体实现的 DAO 类之间实现了动态的绑定，它们之间的联系关系可以在运行时期动态地进行确定和关联，具体如图 2 所示。可以看出，对于某个具体的 Action 类来说，仅仅只需要将当前要显示 Page 类以及要查询的实体对象作为参数，并且以 Request 方式进行传入，然后即可在具体的 Action 类中进行处理。

当处理完成后，就可以把一个已经处理好的 Page 对象，传入显示层进行显示。接下来仅需机械重复地在不同的 Action 中定义相同方法 loadPageList 方法，并将对相应过程

进行简单机械的实现即可。

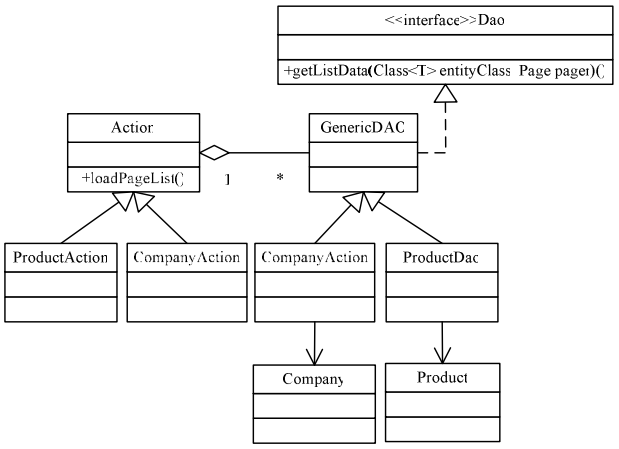

图 2　在 Action 类中实现的桥梁模式

简易示例如下：

```
//获取分页的客户信息列表
HttpSession session=request.getSession();
//从需要分页的页面中接收当前 Page 对象
Page page=new Page().getPage(String curPage,int totalRecord,int pageSize);
//计算当前分页的索引位置
int firstindex=(page.getCurpage()-1)* page.getPageSize();
//设置需要的分页结果集，并把分页对象返回给显示页面
page.setQueryResult(CompanyDao. getListData (Company.class, page));
request.setAttribute("page", page);
return SUCCESS;
```

4.4　显示层的高效移植方案

为保证显示层高效移植，在本文方案中只需要写一个单独的 pageList.jsp 来描述分页页面即可。它可以轻松地实现页面组件的移植，并保证在需要分页的页面中以<jsp:include>的方式将 pageList.jsp 包含进去，而在 pageList.jsp 中已经完全实现了它与 Action 类之间耦合性的分离。2 种分页显示模型的实现代码[1]如下：

(1)上下型分页组件的实现

```
当前页:第${page.curpage}页 | 总记录数:${page.totalrecord}条 | 每页显示:${page.maxresult}条 | 总页数:${page.totalpage}页|上一页：${page.currentpage-1} |下一页：${page.currentpage+1} |最后页：${page. totalPage }
```

(2)随意型分页组件的实现

在随意型分页组件中，需要随意改变的是随机跳转的页面。该实现需要采用<select>标签加上 Struts2 中<s:bean>标签中的 org.apache.struts2.util.Counter 完成。

```
<select name="pageNum" onchange="javaScript:toUrl();" id="_pageNum">
 <s:bean name="org.apache.struts2.util.Counter" id="counter">
 <s:param name="first" value="0" />
 <s:param name="last" value="#request.page.totalPage-1" />
 <s:iterator>
 <option value="${counter.current}"> 第 ${counter.current} 页 </option>
 </s:iterator>
 </s:bean>
</select>
```

其中，toUrl()，用来指定跳转到的某个分页页面上。

5　结束语

本文针对 Web 系统的分页查询问题，结合 MVC 模式和 S2SJ 框架技术实现一种分页组件模型。该模型的耦合性弱、复用性高、可读性强，且与数据库相关性小，易于复用、学习和部署。目前，该分页组件已在某电信公司和下属公司中使用，获得了较高的用户满意度。

参考文献

[1] 顾志峰, 李娟子. Web 应用程序分页策略的研究[J]. 计算机工程, 2005, 31(21): 60-62.

[2] 何玲娟, 蚁　龙, 刘连臣. 一种松耦合高复用 MVC 模式的 Web 分页实现[J]. 计算机工程与应用, 2007, 43(15): 95-97.

[3] 张　俐. MVC 模式在数据中间件中的应用[J]. 计算机工程, 2010, 36(9): 70-72.

[4] 俞东进, 阮红勇. 支持快速开发的行业软件通用框架[J]. 计算机工程, 2009, 35(20): 47-49.

[5] 欧黎源, 邱会中, 白亚茹. 基于 JPA 的数据持久化模型设计与实现[J]. 计算机工程, 2009, 35(20): 76-77, 80.

编辑　金胡考

(上接第 245 页)

数，可满足不同负载特性传动系统的实验要求。基于此实验系统还可开展港口变频电源的特性、调整、效率等研究。

6　结束语

本文提出传动实验平台电力测功机监控系统。该系统集成现场总线技术、OPC 通信技术和计算机控制技术，实现数据采集的实时性、转矩控制的准确性和转矩模拟的多样性，具有结构简单、通信可靠和操作方便等特点，对类似的交流电力测功机具有推广应用价值。

伴随着电力电子技术的发展和完善，对电力测功机智能控制和预测方法提出了越来越高的要求，即希望能够把各种现在或未来的因素考虑进去进行控制、调整和优化。但运算和模型的复杂性使其仅处于理论研究阶段，如果能够设计出相应的芯片，就能更好地解决实际应用问题，这将是以后研究的重点。

参考文献

[1] 李宗帅, 董　春, 刘　颜. 国内外电力测功机发展现状[J]. 电机与控制应用, 2007, 34(5): 1-4.

[2] Williamson A C, Alkhalidi K M S. An Improved Engine-testing Dynamometer[C]//Proc. of the 4th International Conference on Electrical Machines and Drives. [S. l.]: IEEE Press, 1989.

[3] Collins E R, Hung Y. A Programmable Dynamometer for Testing Rotating Machinery Using a Three-phase Induction Machine[J]. IEEE Transactions on Energy Conversion, 1994, 9(3): 521-527.

[4] 李　强, 方一鸣, 陈　刚, 等. 五机架冷连轧机监控系统的设计与实现[J]. 计算机工程, 2009, 35(2): 245-246.

[5] 袁裕辉. Delphi 多线程数据库应用程序编程技术[J]. 计算机工程, 2001, 27(1): 162-163.

编辑　刘　冰