

基于 JSP 技术的数据库查询分页显示

曹 晋, 胡谷雨

(解放军理工大学 指挥自动化学院, 江苏 南京 210007)

摘要:目前,数据库在 Web 服务中得到了广泛应用。针对 Web 系统中经常遇到的大量数据的多页查询问题,分析了运用 JSP 技术开发 Web 数据库应用,作为进行查询分页显示的主要技术手段。文中对目前广泛应用的多种分页显示的方法进行了对比介绍,然后以一实例讨论了 Web 数据库数据记录分页显示的程序设计方法,并结合实际工程给出了相关文件的部分程序源代码。结果表明,该方法可以达到显示逻辑和业务逻辑的分离、代码重用、与具体数据库无关和执行效率提高的目的。

关键词:查询;分页显示;JSP;Web 数据库;程序设计

中图分类号:TP393.09

文献标识码:A

文章编号:1673-629X(2007)05-0225-03

Pagination Display of Database Query Based on JSP

CAO Jin, HU Gu-yu

(Institute of Command Automation, PLA University of Science and Technology, Nanjing 210007, China)

Abstract: Nowadays, the databases are widely used in Web services. Focusing on the question of magnanimous database pagination query in Web-based system, analyzes the application developing of Web-based database with JSP and the main techniques for inquiry pagination display, and introduces some paging method which are widely used by contrast. Then discusses the programming method for pagination display of Web-based database data recording by an example. At last, presents a short program source codes. The result indicates that this realization method is a good way to make the separation of presentation from logic, reuse the code, be foreign to concrete databases and increase efficiency of application.

Key words: inquiry; pagination display; JSP; Web-based database; programming

0 引言

随着因特网的广泛应用,Web 应用程序开发已成为业界设计的主流,开发各种 Web 应用(如在电子商务系统中)必然要涉及到大量数据的查询显示,数据的多页显示是不可避免会遇到的问题。如果要显示的数据很多,用户往往希望能以分页的方式来显示。因为简单地将数据一次性地全部发往客户端,不仅不方便终端用户的浏览,而且影响了系统的性能:首先,大量数据显示在终端用户的浏览器窗口上使得用户只能通过窗口上的滚动条来帮助查看数据,很不方便,同时容易错过自己想要的信息;其次,一次性将大量的数据发往客户端,增加来往网络的负担,延缓了系统的反应速度,降低了系统的性能。所以采取分页技术非常必要,应用数据库记录分页技术,可以显著提高网站运行效

率,以及大幅度减轻数据库服务器的负担^[1]。

1 基于 JSP 技术的数据库查询分页显示的解决方案

1.1 JSP 分页技术简述

Java 语言具有平台无关性和强大的网络编程功能,因此 Java 已经成为网络应用开发中最常用编程语言。基于 Java 平台面向服务器端的 JSP/Servlet 技术秉承了完全面向对象的 Java 语言的优点,以其开放式、跨平台、移植性好、运行效率高、安全性好等优势已逐渐成为开发动态网站技术的主流。

JSP 分页技术^[2]是指基于 JSP 技术的 Web 网站针对用户的大批量数据查询请求情况,将用户所需数据信息分批取出、传送至客户浏览器的技术。分页的意义在于减少网络的流量和用户端的负荷。基于此目的,笔者采用 JSP/Servlet 技术来实现 Web 动态交互,其体系结构模型如图 1 所示,此模型利用 Oracle 或 Mysql 等数据库系统作为后台数据库,用 Servlet 等高

收稿日期:2006-07-25

作者简介:曹 晋(1981-),男,河北保定人,硕士研究生,研究方向为计算机网络及应用;胡谷雨,博士,教授,博士生导师,研究方向为网络管理、网络安全。

性能的服务器程序负责处理 HTTP 请求,包括对输入数据的检查和转换,通过 JavaBeans 访问数据库,初始化 JSP 中要用到的 Beans 或对象,调用不同的 JSP 程序向客户端反馈信息,JSP 程序在前台运行,主要负责生成交互后返回的界面。

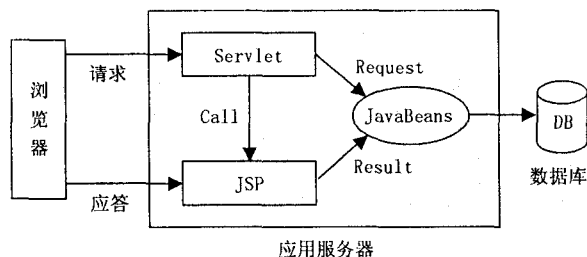


图 1 采用 JSP/Servlet 技术实现 Web 动态交互的体系结构模型

由查询接口向数据库服务器提供起始显示项号和最大显示项数,经服务器过滤,将可显示的结果信息返回给结果集,并根据服务器给出的结果总数和起始显示项号生成翻页链接。

1.2 基于 JSP 技术的分页显示的方法比较

据笔者了解,目前 JSP 技术有以下几种基本方法可实现分页显示:

1) 数据库提供的实现方法^[3]。

各种数据库本身都提供了各种函数或变量来控制数据库的分页,如 Mysql, Oracle, DB2 等有自己的分页方法, Mysql 可以使用 limit 子句, select * from tablename limit start, number (返回从第 start + 1 条记录开始的 number 条记录); Oracle 可以使用伪列 ROWNUM 来限制结果集的大小和起始位置, select * from tablename where ROWNUM <= endnumber minus select * from tablename where ROWNUM < startnumber, 返回 startnumber 和 endnumber 之间的所有记录,包括 startnumber 和 endnumber; 在 DB2 中可以使用 select * from tablename fetch first number row only, 来返回前 number 条记录等,但是另外像 MS SQL SERVER 等数据库没有提供此类的接口技术。因此,我们要设计一个通用的类需要适应不同的数据库,而基于这个类(库)的应用,也可能使用在不同的数据库或文件类型。

2) 用 ResultSet 移动游标实现分页。

即不使用任何封装,在需要分页的地方,直接操作 ResultSet 到相应的位置,再读取相应数量的记录。这种分页方法仅适用于较小数据量的情形,因为游标本身有缺点:游标是存放在内存中,很费内存。游标一建立,就将相关的记录锁住,直到取消游标。游标提供了对特定集合中逐行扫描的手段,一般使用游标来逐行遍历数据,根据取出数据条件的不同进行不同的操作。

而对于多表和大表中定义的游标循环很容易使程序进入一个漫长的等待甚至死机。更重要的是,对于非常大的数据模型而言,在分页检索时,如果按照传统的每次都加载整个数据源的方法是非常浪费资源的。另外,这种方法没有考虑到代码重用的问题,不仅代码数量巨大,而且在代码需要修改的情况下,将会无所适从。

3) 将查询结果缓存在 HttpSession 或有状态 bean 中,翻页的时候从缓存中取出一页数据显示。

这种方法目前使用比较多,其优点是减少了对数据库访问的次数,对数据库连接以及游标等访问资源占用也较少。但有两个主要的缺点:一是用户可能看到的是过期数据;二是如果数据量非常大时第一次查询遍历结果集会耗费很长时间,并且缓存的数据也会占用大量内存,效率明显下降。

4) 检索指定页数据,用 JavaBean 封装的分页方法。

这种方法虽然每次翻页都需要查询数据库,但查询出的记录数很少,网络传输数据量不大,如果使用连接池更可以略过最耗时的建立数据库连接过程。在数据库端有各种成熟的优化技术用于提高查询速度,比在应用服务器层做缓存有效许多,关于使用数据库连接缓冲池的技术很多,这里不赘述。

5) 使用标签库的方法。

将分页查询和显示做成 jsp taglib, 可以进一步简化 JSP 代码,屏蔽 Java Code。目前比较流行的数据分页显示的组件有 DisplayTag, ValueList 和 Pager - taglib^[4]等。其中 Pager - taglib 使用灵活简单,且可以较好地与数据库本身的分页查询语句结合,效率比较高。

2 分页显示方法的实现

2.1 设计思路

分页显示可以通过如下步骤实现:

- 1) 定义变量,为相应变量赋初值;
- 2) 创建数据库连接;
- 3) 从查询界面获得查询条件,产生操作数据库的 SQL 语句;
- 4) 执行查询语句,返回要查询的记录数;
- 5) 获得指定页面的数据;
- 6) 根据每一页能够显示的记录总数,计算符合查询条件的记录数需要显示的总页数;
- 7) 发送数据库数据,把当前记录传送到客户浏览器;
- 8) 关闭数据库连接;
- 9) 设置“首页”、“上一页”、“下一页”、“尾页”、“转

到第 n 页”选项,并用链接后添加参数(页码)的方式,发送新的请求,以便显示新的分页。

在具体的实现方案中,文中采用的是每次翻页的时候只从数据库里检索页面大小的块区的数据,然后利用 Vector 封装当前要显示的数据内容的方法,因为在当前项目系统中,对后台数据库最频繁的操作是查询,针对大批量数据查询,采用此方法让分页应用程序根据客户端的请求多次访问数据库,每次只获得当前页所需数据。

2.2 设计实现

下面是笔者用 JSP 技术进行数据分页显示的一个实现方案^[5]:

1) 构建一个 PageControl 对象将分页所涉及的一些关键的控制数据予以封装。

关于每页所要显示的资料数据的载体,其实现方式多种多样,可用 bean 的形式,这是一种面向对象的实现,比较简单的实现方法可用 `java.util.Vector`。

```
public PageControl(yourPersistenceLayer yourPL)
```

这是一个参数类型为 `yourPersistenceLayer` 的构造函数, `PersistenceLayer` 是直接同数据库打交道的一层,在应用系统中要实现数据库的连接、数据添加、查询、修改与更新等。不同的公司都有不同的实现,比如说 Microsoft 的 ADO 就可以看作是一 `PersistenceLayer`,一种简化的做法是不要 `PersistenceLayer`,或者可以说是淡化该层,这样做势必降低系统的稳定性、可重用性、可扩展性。我们的做法是用 `JavaBean` 封装 `JDBC` 代码,通过调用 `JavaBean` 来完成对数据库的各种操作。在这个构造函数中,有这样几个主要操作:

```
this.maxRowCount = yourPL.getAvailableCount(); //得到总行数
```

```
this.yourdata = yourPL.getResult(); //得到要显示于本页的数据
```

```
this.countMaxPage(); //计算总页数
```

这里 `this.yourdata` 是用 `PageData` 类来封装从数据库查找到的商业逻辑数据,在从数据库中获取商业逻辑数据时,通常有两种方式^[6]:

a. 第一次把所有的数据资料都查询出来,然后在每页中显示指定的资料;

b. 多次查询数据库,每次根据当前页号,只获取本页的数据,将其他数据予以抛弃。

考虑到数据往往是大量的,如果一次性地获取,那么这些数据必然大量占用服务器内存资源,使系统性能大大降低,因此这里使用第二种方式。在 `PageData` 中利用 `JDBC` 的 `ResultSet` 对象来得到所需要的分页后具体某一页的数据,通过 `Statement` 对象的 `set-`

`MaxRows(int max)` 方法来设置由该 `Statement` 对象所得到的 `ResultSet` 对象包含的记录条数,例如 `setMaxRows(10)`,则由该 `Statement` 得到的所有的 `ResultSet` 对象最多包含 10 条记录,其余的记录均被舍弃。然后通过 `ResultSet` 对象的 `absolute(int row)` 方法将游标定位到所需页记录的前一条记录上,最后从经过如上处理的结果集得到所需记录的内容,封装成 `Vector` 返回。通过这样的处理,才能真正得到所需页的数据,去除多余的数据,达到节省服务器内存资源,提高系统性能的效果。

在返回查询结果的 `yourBusinessObject` 中,通过判断当前页、判断起始页、循环取数据库值等步骤操作数据库,设一个 `Vector` 对象,用来返回获得指定页面数据。

2) 使用一个 `Servlet` 来处理、接收客户端的查询请求,查找数据,并将结果存储在 `JavaBean` 中,最后再将请求转发到相应的 JSP 页面提交。具体做法是:调用第一步中返回相关结果的 `yourBusinessObject` 的 `listData` 方法,并且获得 `PageControl` 对象,把它保存在 `Request` 中,在 `Servlet` 中只需进行如下操作:

```
PageControl pageCtl = yourBusinessObject.listData(req.getParameter("jumpPage"));
```

```
request.setAttribute("pageCtl", pageCtl);
```

3) 建立 JSP 页面实现可视化,用 `pageman.jsp` 实现分页显示的逻辑处理,该 `pageman.jsp` 可重用,实现了以下逻辑:

- a. 第一页时不能再向前翻;
- b. 最后一页时不能再向后翻;
- c. 同时能够进行任意页面跳转。

3 结束语

文中所讨论的基于 JSP 技术的数据库查询分页显示,源自实际开发中的需求,笔者对用户的需求进行了概括和整理,并给出了相应的解决方案,并已调试通过。该方法不与具体数据库相关,通用性好,能做到代码重用,执行效率较高。

参考文献:

- [1] 陈远平,及俊川. 基于 JAVA 的多种数据库分页方法研究[J]. 计算机系统应用, 2005(3): 65-69.
- [2] Sun Microsystems, Inc. Sun JSP Documentation[EB/OL]. 2002. <http://java.sun.com/products/jsp/docs.html>, Copyright 2002-2006.
- [3] 李安,刘晓东. 一个通用 JSP 数据库分页例程[J]. 微机

(下转第 230 页)

```

status = WdfDeviceCreateSymbolicLink(device, L"\\Dos-
Devices\\WDFDEMO");
if (! NT_SUCCESS(status)) return(status);
// 创建和初始化请求队列
WDF_IO_QUEUE_CONFIG_INIT(&ioCallbacks, WdfIo-
QueueDispatchSerial, WDF_NO_EVENT_CALLBACK, WDF-
NO_EVENT_CALLBACK);
ioCallbacks.EvtIoDeviceControl = MyEvtDeviceControlIoctl;
status = WdfDeviceCreateDefaultQueue(device, &ioCallbacks,
WDF_NO_OBJECT_ATTRIBUTES, NULL);
if (! NT_SUCCESS(status)) return(status);
// 创建和初始化中断对象, MyIsr 和 MyDpc 分别是中断服
务例程和 DPC 例程
WDF_INTERRUPT_CONFIG_INIT(&interrupt
Config, FALSE, MyIsr, MyDpc);
interruptConfig.EvtInterruptEnable = MyEvtInterrupt En-
able; // 中断使能
interruptConfig.EvtInterruptDisable = MyEvtInterrupt Dis-
able; // 中断禁止
status = WdfInterruptCreate(device, &interruptConfig,
&objAttributes, &devContext -> WdfInterrupt);
// 一些其他初始化操作(略)
return(status);
}

```

开发 WDF 驱动程序下一步的工作就是编写各事件处理回调函数, 当相应事件发生时, WDF 框架会自动调用指定的回调函数进行处理。其中 EvtDevicePrepareHardware 回调函数在分配资源的时候被调用, 框架将分配给设备的资源传递给回调函数, 回调函数保存需要的资源, 将共享内存映射到内核虚拟地址空间。与此对应的是 EvtDeviceReleaseHardware 回调函数, 每当设备释放所占用的资源时, 框架都将调用它。

EvtDeviceD0Entry 和 EvtDeviceD0Exit 事件 callbacks 则分别在设备即将进入和离开 D0 电源状态时调用。EvtIoDeviceControl, EvtIoRead, EvtIoWrite 等回调函数分别用来处理 DeviceControl, Read, Write I/O 请求。当框架获得一个 I/O 请求时, 它首先确定该请求应该放入哪个请求队列。如果驱动程序没有提供指定的队列, WDF 框架默认将请求放入缺省请求队列会自

动调用对应的回调函数。然后, 框架寻找处理该请求的回调函数, 如果驱动程序提供了相应的 callback, 则调用它处理请求。对于没有指定回调函数的 I/O 请求, WDF 调用 EvtIoStart 回调函数处理。如果 EvtIoStart callback 也不存在, 框架将返回 STATUS_NOT_SUPPORTED。

设备中断的管理由 EvtInterruptEnable callback、EvtInterruptDisable callback、中断服务例程 (ISR) 和 DpcForIsr 例程实现。WDF 框架在调用 EvtDeviceD0Entry callback 和注册 ISR 后, 通过调用 EvtInterruptEnable 回调函数使能设备中断; 而 EvtInterruptDisable 回调函数则在设备离开 D0 状态, EvtDeviceD0Exit callback 调用前获得调用, 完成禁止设备中断的工作。此外, 中断服务例程和 DpcForIsr 例程具体完成中断服务的功能, 与 WDM 驱动程序相似。

3 结束语

WDF 驱动程序模型是 Microsoft 下一代驱动程序模型, 它提供的 KMDF 框架为开发内核模式驱动程序提供了一个面向对象、事件驱动的开发框架, 极大地简化了设备驱动程序的开发, 避免了由于驱动程序的错误导致整个操作系统崩溃的现象。文中分析了 WDF 的对象模型和驱动程序结构, 并给出了基本的实例。WDF 驱动程序有很多值得深入研究的地方, 今后还要对其作进一步的探讨。

参考文献:

- [1] Microsoft Corporation. Architecture of the Windows Driver Foundation [EB/OL]. 2005. <http://www.microsoft.com/whdc/driver/wdf/default.aspx>.
- [2] OSR online. The Future Is Now—The WDF Kernel Mode Framework [EB/OL]. 2004. <http://www.osronline.com/article.cfm?article=287>.
- [3] Cant C. Windows WDM 设备驱动程序开发指南 [M]. 北京: 机械工业出版社, 2000.
- [4] OSR online. A New Framework [EB/OL]. 2004. <http://www.osronline.com/article.cfm?id=289>.
- [5] Microsoft Corporation. Microsoft Corporation Windows DDK document [M]. USA: Microsoft Corporation, 2002.

(上接第 227 页)

发展, 2002, 12(4): 53-54.

- [4] Tag Libraries: Pager Tag Library [EB/OL]. 2004-01-24. <http://jsptags.com/tags/navigation/pager/index.jsp>.
- [5] 飞思科技产品研发中心. JSP 应用开发详解 [M]. 第 2 版.

北京: 电子工业出版社, 2005: 310-316.

- [6] Ayers D, Bell J, Calvert Bettis C. Java 数据编程指南 [M]. 戴英, 等译. 北京: 电子工业出版社, 2002.