经济昆虫在线导购系统的设计与实现

雒动, 倪远平
(昆明理工大学 信息工程与自动化学院, 云南 昆明 650051)

摘要: 文章针对经济昆虫实现网上导购的项目要求, 利用 petstore框架构建了经济昆虫在线导购系统, 实现了商品动态更新、查询、购买, 身份验证、在线支付、系统管理等功能, 其中目录树的间接生成策略为需要频繁读取数据库的数据显示提供了一种高效可行的方法, 具有较高的实用价值.

关键词: 电子商务; 经济昆虫; 应用程序框架
中图分类号: TP312JA 文献标识码: A 文章编号: 1007-855X(2005)04-0043-05

Design and Realization of Economic Insect Online Shopping System

DING Wei-dong, NI Yuan-ping
(Faculty of Information Engineering and Automation, Kunming University of Science and Techology, Kunming 650051, China)

Abstract: To solve the inconvenience of buying economic insects caused by the area difference, aonline insect store is realized by JBuilder8.0 and SQLServer2000, according to the petstore application framework of J2EE. The store has the functions of automatic updating, querying and purchasing products, authentication and online-pay etc. The strategy of building catalog tree provides a practical and efficient method for displaying data which needs frequent access to database.

Key words: online shopping; economic insect; MVC

0 引言

在现有的自然资源中, 数量多、分布广、营养丰富、易于开发的资源首属经济昆虫资源. 据统计, 全世界约有3 650余种昆虫可供食用, 1 700多种昆虫具有药用价值, 还有大量的昆虫可用于农田、森林的天敌防治工作. 随着人们对经济昆虫营养、保健和生态价值认识的逐步深入, 已有越来越多的昆虫被大量进行人工养殖. 然而, 由于经济昆虫地区分布差异大, 传统的产供销模式存在着时效性差, 信息流动慢等诸多问题. 而电子商务作为新兴的购物方式, 以其特有的方便快捷性和潜在的市场生机而逐渐取得人们的认同和青睐. 但目前的电子商务主要用于日常生活用品、学习用品等方面, 用在经济昆虫领域的则几乎没有.

本文针对这一情况, 采用 MVC模式和 J2EE技术研究和开发了经济昆虫在线导购系统, 实现了经济昆虫在线查询、订购、更新, 身份验证、在线支付以及系统管理等功能, 解决了传统销售模式所存在的信息流动慢, 购销不便等问题, 填补了电子商务在经济昆虫应用方面的空白.

1 系统模型

考虑到开发的效率以及维护的可能性和扩展的方便性, 系统采用 Buschmann提出的 MVC标准模式, 这个模式经过多年的应用经验积累, 已经被广泛采纳接受. MVC模式由三个部分组成[1]:

- Model数据层: 应用系统的数据的存放.
- View 表示层: Model中的存储数据的可视化表示.
- Control控制层: 接受用户的输入, 通知 Model发生的事件.

收稿日期: 2004-07-05.

第一作者简介: 丁维动(1980~), 男, 在读硕士研究生. 主要研究方向: 人工智能和网络技术. E-mail: dingvdong@163.com

在本系统中用 J2EE的三个组件 Servlet, JavaBean/EJB, JSP分别对应 MVC的三个部分.

- Model数据层: 全部封装于 Enterprise JavaBean控件中.
- View 表示层: 由 JSP负责处理页面的表示. 比如: 数据格式, 翻页, 参照 /编辑模式等.
- Control控制层: Servlet接受用户在页面的输入以及提交动作, 并根据动作指示, 进 行相应的业务处理(调用相应的 EJB控件), 然后根据处理结 果交给相应的 View 表示层 JSP程序, 由他们负责表示.

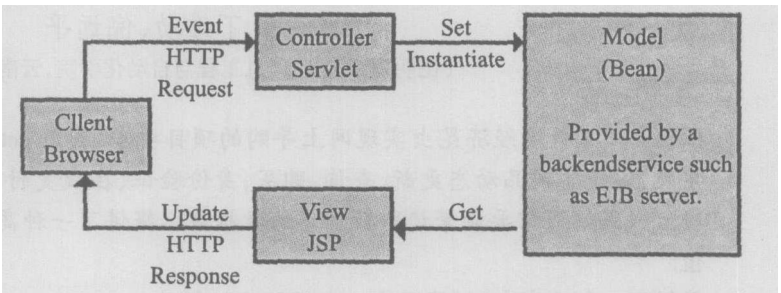

图1 MVC模式在J2EE编程中的体现

Fig.1 Implementation of MVC model in J2EE

图示说明:

1) Servlet接收 HTML Form 传送来的 Http Request请求, 并转换为 Event事件.

2) Servlet根据相应的 Event事件调用业务逻辑层的方法, 进行业务逻辑处理.

3) Model层次的 JavaBean/EJB通过 JDBC/EJB操作数据库, 将数据保存在 JavaBean中.

4) Servlet取得对数据存放的 JavaBean的参照以后, 传递给 JSP

5) JSP根据数据, 生成 HTML页面, 返回浏览器, 给进行页面表示.

2 系统的设计与实现

本系统共分为以下几个模块: 控制模块、购物车模块、登录模块和目录模块, 如图 2所示:

2.1 控制模块设计

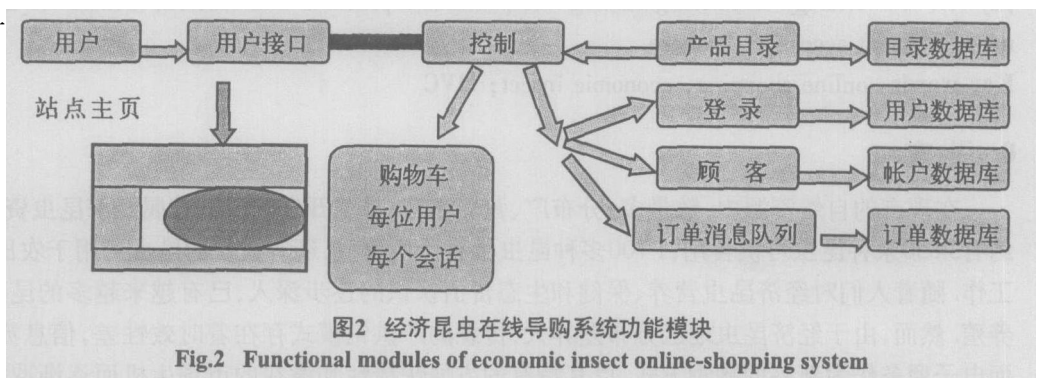

图2 经济昆虫在线导购系统功能模块

Fig.2 Functional modules of economic insect online-shopping system

控制模块是整个经济昆虫在线导购系统的管理中心, 我们将其它模块中除业务逻辑以外的任务(比如视图展现)都抽象到此模块中来, 从而实现集中调度. 它也是唯一的一个直接与用户交互的模块. 因为控制逻辑在实际应用中是经常改变的, 所以必须保证控制模块的扩展性和可维护性. 鉴于此, 我们采用了 sun公司提供的 petstore应用程序框架(application framework)[2]来进行开发, 其服务周期如图 3所示:

1) 请求过滤器: 我们开发了 SignOnFilter过滤器实现用户登录功能, 申请登录的 HTTP请求将被过滤器截获, 从中取出用户名和密码与数据库中数据进行比较验证. 由于过滤器独立于中央控制器之外, 可以很容易地在其中修改身份验证方法(如从基本身份验证改为 SSL)而不必修改中央控制器, 提高了系统的维护性.

2) 中央控制器: 我们利用 ControllerServlet作为作为整个系统的控制枢纽, 所有形如*.do的请求都将被之捕获, 进而调用相应的 Java类执行请求的转发、屏幕的生成等工作.

3) 利用 XML文件将请求映射到 HTML Action类并执行行为: 在 requesMap.xml中定义各个 URL与 Action类的映射关系, 比如:

```
<url-mapping url="order.do" screen="order_complete.screen">
  <web-action-class>dvd.actions.OrderHTMLAction</web-action-class>
</url-mapping>
```

在上述代码中指定了 order do映射到 ordeHTMIAction, 这样每当中央控制器捕获 order do请求时就调用 OrdeHTMLAction这一 java类. 每个 HTMLAction包括以下几个方法, 每个方法的功能标于其后的注释中:

public void doStart(HttpServletRequest request) //初始化行为

public void perform(HttpServlet request) throws HTMLActionException //执行行为的业务逻辑

Public void doEnd(HttpServletRequest request, EventResponse evr) //行为执行完毕后所要做的清理工作

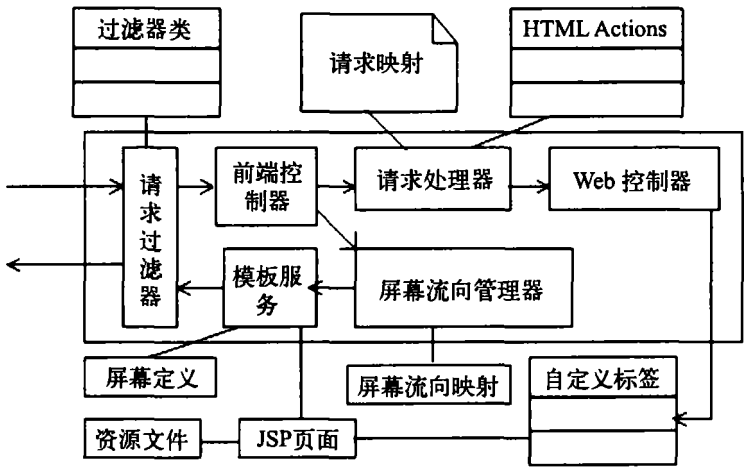

图3　Web应用程序框架服务周期

Fig.3　Service cycle of Web application framework

4) 屏幕 (screen)选择: 所谓屏幕, 是我们自定义的对应若干 JSP页面的一个集合名称. 执行完业务逻辑后, HTML Action类将读取 requesMap xml中定义的下一个屏幕. 例如由 3)中给出的代码可知, order do对应的 screen为 order_complete screen, 该屏幕对应的 JSP页面是在 screenDefinition xml文件中给出的, 节略如下:

```
< screen- definitions>
  < default- template> /template jsp< /default- template>
< screen name="order_complete">
  < parameter key="title" value="定单发送成功" direct="true" />
  < parameter key="top" value="/top jsp" />
  < parameter key="catalog" value="/catalog jsp" />
  < parameter key="body" value="/order_completed jsp" />
< /screen>
< /screen- definitions>
```

在 screenDefinition xml中还有 main screen, signon screen等屏幕的定义, 如果要更改用户界面, 只要在此 XML文件中更改各个 screen所对应的 JSP页面即可.

5) 视图的组装和响应的产生[3]: 我们开发了 AssembleServlet从上述 screendefinition. xml中读取所有屏幕所对应的 JSP页面, 存储到一个 java对象 Screens中, 当 AssembleServlet收到 HTML Action类所发来对某一屏幕的请求时, 从其中提取屏幕名称, 再调用 process( )方法将请求转发到在 screendefinition xml中定义的模板页面 Template jsp. Template jsp则利用< jsp include>指令将 AssembleServlet所获得的具体 JSP文件引入, 从而组装成一幅完整的画面呈现给客户.

下面各个模块的请求处理和视图展现都是用上述方法实现的.

2.2 购物车模块设计[4]

购物车模块要实现的功能有: 记录用户选择的多个商品; 存储每件商品的 ID、种类、名称和价格; 保证任何一件商品的数量不能为负; 用户可以增加、删除和查看商品, 提交购物车或放弃购物. 我们开发了一个 CartBean来代表购物车, 有相应于用户操作的 addProduct( ), deleteProduct( ), getTotal( ), empty( )等方法, 并在 JSP页面中将其作用域设为 session, 这表示它在整个用户登录过程中都有效, 从而可以记录用户购物车中的内容.

CartBean的部分代码如下:

```
private Vector cart = new Vector();
public Vector getCart() { return cart; }
public void addInsect(InsectBean insect) {
  if(insect.getId() != null && cart.indexOf(insect) == -1) { cart.addElement(insect);
  }
}
public void deleteInsect(String id) {
  Enumeration insects = cart.elements();
  while(insects.hasMoreElements()) {
    InsectBean insect = (InsectBean) insects.nextElement();
    if(insect.getId().equals(id)) {
      int index = cart.indexOf(insect);
      cart.removeElementAt(index);
    }
  }
}
public Enumeration getInsects() { return cart.elements(); }
public float getTotal() { }
```

2.3 登录模块设计

其原理如下:用户发出的每一个请求都被中央控制器 ControllerServlet截获检查,当顾客要访问购物车、帐户信息等页面时,中央控制器判定目标页面为受保护页面(也是在 XML文件中定义),于是检查请求中是否有登录标识 valid_user,找不到便将请求转到登录页面(signon.jsp)要求用户登录.与此同时,用户所要访问的页面 URL被保存到一个请求属性中以备后用.当用户成功登录后, ControllerServlet将为用户生成一个会话作用域的登录标识和存储个人信息的 JavaBean,并从前述的请求属性中取出所存页面的URL进行自动转发,此转发请求将再次被中央控制器捕获,因用户登录标识已经设置,所以此请求将成功通过检查.

2.4 目录模块设计

2.4.1 目录模块的主要需求

- 根据目、亚目、科、亚科、属的层次组成树型目录结构;
- 当某一级为叶子节点时,在屏幕的 body部分显示其详细信息;
- 每件商品都要有唯一的标识和描述信息;
- 目录树与库存商品保持一致;
- 作为本系统中最常用的组件,目录树必须具有良好的响应性能.

2.4.2 目录模块的实现

对应目、亚目、科、亚科,我们开发了 mu.jsp, yamu.jsp, ke.jsp, yake.jsp等页面,当要显示目录树的下一级目录时,用 <jsp:include> 指令引入相应页面并以当前物种名称为参数查询数据库,如果查询结果集不为空,则遍历结果集进而将物种名称显示出来;如果结果集为空,则将当前物种名称传递给 details_screen所对应的 JSP页面,在其中查询数据库并显示详细信息.与库存商品保持一致的初步实现方案是用无状态会话 Bean通过数据访问对象(DAO)动态访问数据库,功能上得到了实现,但性能未达最佳,为了提高系统的响应性能,我们设计了目录树的间接生成策略.

2.4.3 目录树间接生成策略

如前所述,商品目录必须要与数据库同步,但与数据库进行交互是一项很耗资源的工作.为了既满足同步的要求又不影响性能,我们设计了如下方法:

当服务器启动时, 读取数据库数据生成一个 CatalogBean, 其作用域为 application scope, 这意味着所有访问此系统的用户都可以直接利用它来生成目录树, 而不必再次读取数据库. 但既然目录树不是实时生成, 那又如何做到与数据库的同步呢? 解决方法是一旦数据库数据发生变化, 即用户购买商品或供货商更新商品时, 便由中央控制器销毁原有的 CatalogBean并读取数据库信息重新生成, 代码如下:

```
//读取数据库信息生成 CatalogBean
mus = loadMus(conn);    //CatalogBean中的目
for(int i = 0; i < mus.length; i++) {
  YaMu[] yamus = loadYaMus(conn, mus[i].getName()); //亚目
  if(yamus.length != 0){
  mus[i].setYaMus(yamus);
  for(int j = 0; j < yamus.length; j++) {
  Ke[] kes = loadKesOfYaMu(conn, yamus[j].getName()); //科
  yamus[j].setKes(kes);
  for(int k = 0; k < kes.length; k++) {
    YaKe[] yakes = loadYaKes(conn, kes[k].getName()); //亚科
    kes[k].setYaKes(yakes); }.....
  }
//当数据库数据发生变化时销毁 CatalogBean
catalogBean = null;
//重新读取数据库信息生成 CatalogBean, 代码同前.
```

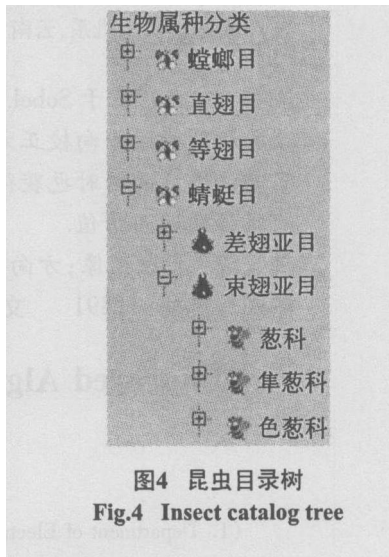

图4　昆虫目录树
Fig.4　Insect catalog tree

目录树的间接生成策略既保证了与数据库的同步, 又实现了商品目录的快速生成, 极较好地满足了系统需求.

3 结束语

文章针对经济昆虫购销所存在的区域性问题, 设计并开发了经济昆虫在线导购系统, 实现了商品动态更新、查询、购买, 身份验证、在线支付等功能, 具有较高的实用价值. 其中所提出的目录树间接生成策略, 为需要频繁读取数据库的数据显示提供了一种可行而高效的方法, 对实现类似的在线购物系统具有一定的参考意义.

参考文献:

[1] Nadir Gulzar 实用 J2EE应用程序体系结构 [M]. 陈晓燕, 丁炎炎, 译. 北京: 清华大学出版社, 2003. 37~40.
[2] Sun corporation. J2EE blueprints[EB/OL]. http://java.sun.com. 2004.
[3] Andrew Harbourne-Thomas. Servlet2.3 programming[M]. O'Reilly and Associates, Inc. 2002. 18~96, 156~208.
[4] Hans Bergsten. Jsp Design[M]. O'reilly and Associates, Inc. 2000. 158~302.