(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 104008502 A
(43)申请公布日 2014.08.27

(21)申请号 201410270905.0

(22)申请日 2014.06.17

(71)申请人 大连大学
地址 116622 辽宁省大连市经济技术开发区学府大街10号

(72)发明人 汪祖民 王阳

(74)专利代理机构 大连智高专利事务所(特殊普通合伙) 21235
代理人 毕进

(51)Int.Cl.
G06Q 30/08(2012.01)
G06F 17/30(2006.01)

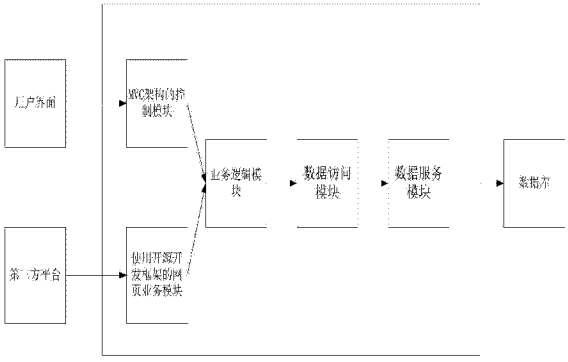
权利要求书1页 说明书16页 附图1页

(54)发明名称

电子拍卖系统

(57)摘要

本发明涉及一种对外开放的电子拍卖系统。包括:网页业务模块,控制模块、业务逻辑模块、数据访问模块和数据服务模块,其中,网页业务模块由开源框架开发得到,用于为第三方平台开放调用;控制模块由模型-视图-控制器MVC框架开发获得,用于控制业务逻辑模块和对外提供界面的交互;业务逻辑模块,用于提供业务规则,并根据业务规则进行逻辑处理,并对DAO对象进行证明模式的封装;数据访问模块,由Spring框架开发的,用于与数据库的持久化对象交互,封装对数据的原子操作;数据服务模块,采用Hibernate作为O/R Mapping框架开发,用于将关系型数据库的数据映射成对象,实现以面向对象方式操作数据库。



CN 104008502 A

1. 一种电子拍卖系统，其特征在于，包括：网页业务模块，控制模块、业务逻辑模块、数据访问模块和数据服务模块，其中，网页业务模块由开源框架开发得到，用于为第三方平台开放调用；控制模块由模型－视图－控制器 MVC 框架开发获得，用于控制业务逻辑模块和对外提供界面的交互；业务逻辑模块，用于提供业务规则，并根据业务规则进行逻辑处理，并对 DAO 对象进行证明模式的封装；数据访问模块，由 Spring 框架开发的，用于与数据库的持久化对象交互，封装对数据的原子操作；数据服务模块，采用 Hibernate 作为 O/RMapping 框架开发，用于将关系型数据库的数据映射成对象，实现以面向对象方式操作数据库。

2. 根据权利要求 1 所述的电子拍卖系统，其特征在于：所述开源框架为 CXF 框架。

电子拍卖系统

技术领域

[0001] 本发明涉及电子商务领域，尤其涉及一种电子拍卖系统。

背景技术

[0002] 目前，电子拍卖是电子商务的一个重要应用，是一个由拍卖群体决定价格及分配过程的特殊现货交易方式。随着网络技术的发展和保密系统的完善，拍卖交易逐渐从传统模式转向基于网络的电子模式。电子拍卖系统主要有两方面的优势：一是精心设计电子拍卖系统可以实现资源的最优分配，达到资源的合理利用；二是电子拍卖系统可以无需到拍卖现场就可以参与竞标，从而节约大量的人力物力，最小化交易成本，这个特点来源于其网络系统提供的强大功能。电子拍卖涉及到多种网络和信息技术，这些技术不仅可以用于电子拍卖，也可以用于其它电子商务和电子政务。

[0003] WebService（网页业务）主要是为了使原来各孤立的站点之间的信息能够相互通信、共享。Web Service 所使用的是 Internet（因特网）上统一、开放的标准，如 HTTP(Hyper Text Transfer Protocol，超文本传输协议）、XML(Extensible Markup Language，可扩展标记语言）、SOAP(Simple Object Access Protocol，简单对象访问协议），WSDL(Web Services Description Language，Web Service 描述语言）等，所以 Web Service 可以在任何支持这些标准的环境（例如：Windows，Linux）中使用。Web Service 的出现满足了动态的商务合作所要求的基本功能和原则，从而使 Web 不仅成为信息共享的平台，而且成为服务共享的平台。但，不管是 SOAP 还是 WSDL，都比较复杂，如果开发者希望自己手动编写 WSDL 来开发 Web Service，难度是相当大的。Apache CXF 是一个开源的 Service 框架，可用于简化用户的 Service 开发，基于 CXF 开发的应用可提供 SOAP、XML/HTTP、RESTful HTTP 或 CORBA 等服务。CXF 底层也可以使用不同的传输协议，包括 HTTP、JMS 或 JBI 等。同时，CXF 部署灵活，支持多种编程语言。但是目前还没有开放和自动管理的的电 子商务的系统。

[0004] 有鉴于上述的缺陷，本设计人，积极加以研究创新，以期创设一种新型结构的电子拍卖系统，使其更具有产业上的利用价值。

发明内容

[0005] 为解决上述技术问题，本发明的目的是提供一种实现开放和自动管理的电子拍卖系统。

[0006] 本发明的电子拍卖系统，包括：网页业务模块，控制模块、业务逻辑模块、数据访问模块和数据服务模块，其中，网页业务模块由开源框架开发得到，用于为第三方平台开放调用；控制模块由模型－视图－控制器 MVC 框架开发获得，用于控制业务逻辑模块和对外提供界面的交互；业务逻辑模块，用于提供业务规则，并根据业务规则进行逻辑处理，并对 DAO 对象进行证明模式的封装；数据访问模块，由 Spring 框架开发的，用于与数据库的持久化对象交互，封装对数据的原子操作；数据服务模块，采用 Hibernate 作为 O/R Mapping 框架开发，用于将关系型数据库的数据映射成对象，实现以面向对象方式操作数据库。

[0007] 进一步的，开源框架为 CXF 框架。

[0008] 借由上述方案，本发明至少具有以下优点：

[0009] 在传统 JavaEE 项目的开发过程中，利用了 Spring 和 Hibernate 框架来实现 DAO 组件和业务逻辑组件，并利用了 Spring 框架的 IoC 容器来管理各组件之间的依赖关系，从而保证了整个应用具有良好的可扩展性和可维护性。而 CXF 框架的使用，不仅大大减轻开发者开发 Web Service 的过程，而且可以将业务逻辑方法暴露成 Web Service，从而允许其他平台、其他语言的程序来远程调用。

[0010] 上述说明仅是本发明技术方案的概述，为了能够更清楚了解本发明的技术手段，并可依照说明书的内容予以实施，以下以本发明的较佳实施例并配合附图详细说明如后。

附图说明

[0011] 图 1 是本发明电子拍卖的系统的结构示意图。

具体实施方式

[0012] 下面结合附图和实施例，对本发明的具体实施方式作进一步详细描述。以下实施例用于说明本发明，但不用来限制本发明的范围。

[0013] 基于上述考虑，本发明在传统 Java EE 项目—电子拍卖系统的基础上，添加 CXF 框架，将业务逻辑方法暴露成 Web Service，从而允许其他平台、其他语言的程序来远程调用。电子拍卖系统其实就是一个电子商务平台，只要将该系统部署在互联网上，全球的客户都可以在该系统上发布想售出的商品，也可以对拍卖中的商品参与竞价。整个过程无须任何人工干预，有系统自动完成。本发明提供与电子银行的接口，可以通过电子银行的操作，实现从买家到卖家的自动付款。一但付款成功，就可以利用全球物流供应系统将拍卖物品发送到买家手中。可见，本发明一种基于 CXF 的电子拍卖系统是一种开放式的，成本及其低廉的系统，大部分工作无须人工干预，系统自动完成管理。

[0014] 参见图 1 所示，一种电子拍卖的系统，包括：网页业务模块，控制模块、业务逻辑模块、数据访问模块和数据服务模块，其中，网页业务模块由开源 (Celtix+XFire，CXF) 框架开发得到，用于为第三方平台开放调用，这样可将应用中的业务方法暴露成 Web Service 操作，允许其他平台、其他语言的应用来调用；控制模块由模型 - 视图 - 控制器 (Model View Controller，MVC) 框架开发，用于控制业务逻辑模块和对外提供界面的交互。业务逻辑模块，用于提供业务规则，并根据业务规则进行逻辑处理，并对 DAO 对象进行证明模式的封装。数据访问模块，由 Spring 框架开发的，用于与数据库的持久化对象交互，封装对数据的增、删、改、查等原子操作。数据服务模块，用于通过实体 / 关系映射工具将关系型数据库的数据映射成对象，实现以面向对象方式操作数据库，数据服务模块采用 Hibernate 作为 O/R Mapping 框架开发。

[0015] 本发明的效果和益处是：本发明在传统 JavaEE 项目的开发过程中，利用了 Spring 和 Hibernate 框架来实现 DAO 组件和业务逻辑组件，并利用了 Spring 框架的 IoC 容器来管理各组件之间的依赖关系，从而保证了整个应用具有良好的可扩展性和可维护性。而 CXF 框架的使用，不仅大大减轻开发者开发 Web Service 的过程，而且可以将业务逻辑方法暴露成 Web Service，从而允许其他平台、其他语言的程序来远程调用。

[0016] 下面介绍一个具体的例子，来说明上述电子拍卖系统的开发过程包括：1，设计系统数据库，2，开发系统 DAO 层组件，3，开发业务逻辑层组件，4，CXF 框架的添加。

[0017] 现就 CXF 框架对 Web Service 的调用做详细阐述。

[0018] 为了在该应用中启动 CXF 支持，首先应该将 CXF 的核心 JAR 包 csf-2.2.2.jar 复制到 Web 应用的 WEB-INF/lib 路径下。除此之外，还应将 CXF 所依赖的第三方类库复制到 Web 应用的 WEB-INF/lib 路径下。为了在应用中整合 CXF 和 Spring，首先应该在 web.xml 文件中启动 Spring 容器，然后在 CXF 中添加核心 Serlet，负责处理 Web Service 客户端请求。代码如下：

[0019]

```
<?xml version="1.0" encoding="GBK"?>
<!-- 配置Web应用配置文件的根元素，并指定配置文件的Schema信息
-->
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
version="2.5">
  <!-- 指定Spring配置文件的位置 -->
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/applicationContext.xml,
      /WEB-INF/daoContext.xml</param-value>
  </context-param>

  <!-- 配置Web应用启动时候加载Spring容器 -->
  <listener>
```

[0020]

```
    <listener-class>org.springframework.web.context.ContextLoaderLi
stener
        </listener-class>
    </listener>

    <!-- 定义ActionContextCleanUp过滤器 -->
    <filter>
        <filter-name>struts-cleanup</filter-name>

    <filter-class>org.apache.struts2.dispatcher.ActionContextCleanU
p</filter-class>
    </filter>
    <!-- 定义SiteMesh的核心过滤器 -->
    <filter>
        <filter-name>sitemesh</filter-name>

    <filter-class>com.opensymphony.module.sitemesh.filter.PageFilte
r</filter-class>
    </filter>
    <!-- 定义Struts 2的核心控制器：FilterDispatcher -->
    <filter>
        <!-- 定义核心Filter的名字 -->
        <filter-name>struts2</filter-name>
        <!-- 定义核心Filter的实现类 -->

    <filter-class>org.apache.struts2.dispatcher.FilterDispatcher</f
ilter-class>
```

[0021]

```
    </filter>

    <!-- 定义过滤器链 -->
    <!-- 排在第一位的过滤器是：ActionContextCleanUp过滤器 -->
    <filter-mapping>
        <filter-name>struts-cleanup</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
    <!-- 排在第二位的过滤器是：SiteMesh的核心过滤器 -->
    <filter-mapping>
        <filter-name>sitemesh</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
    <!-- FilterDispatcher用来初始化Struts 2并且处理所有的HTTP请求
-->
    <filter-mapping>
        <filter-name>struts2</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
    <!-- 配置图形验证码Servlet -->
    <servlet>
        <servlet-name>img</servlet-name>
        <servlet-class>org.wy.auction.web.AuthImg</servlet-class>
    </servlet>
    <!-- 为图形验证码Servlet指定URL -->
    <servlet-mapping>
        <servlet-name>img</servlet-name>
        <url-pattern>/authImg.jpg</url-pattern>
```

[0022]

```
    </servlet-mapping>
    <!-- 配置CXF的核心Servlet -->
    <servlet>
        <servlet-name>CXFServlet</servlet-name>

  <servlet-class>org.apache.cxf.transport.servlet.CXFServlet</ser
vlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <!-- 为CXF的核心Servlet配置URL -->
    <servlet-mapping>
        <servlet-name>CXFServlet</servlet-name>
        <url-pattern>/services/*</url-pattern>
    </servlet-mapping>
  </web-app>
```

[0023] 为了将 Spring 容器中已有的 Bean 暴露成 Web Service，为所有 Web Service 操作定义一个接口，在该借口中定义所有希望暴露成 Web Service 的方法，接口代码如下：

[0024]

```
package org.wy.auction.ws;
import java.util.List;

import org.wy.auction.business.*;
import org.wy.auction.exception.AuctionException;
import javax.jws.*;
//以@WebService Annotation标注，表明该接口将对应一个Web Services
@WebService
public interface AuctionWs
{
```

[0025]

```
/**
 * 根据赢取者查询物品
 * @param name 赢取者的用户名
 * @param pass 赢取者的密码
 * @return 赢取者获得的全部物品
 */
List<ItemBean> getItemByWiner(String name , String pass)
    throws AuctionException;

/**
 * 查询流拍的全部物品
 * @return 全部流拍物品
 */
List<ItemBean> getFailItems()throws AuctionException;

/**
 * 根据用户名，密码验证登录是否成功
 * @param username 登录的用户名
 * @param pass 登录的密码
 * @return 登录成功返回用户ID，否则返回-1
 */
int validLogin(String username , String pass)
    throws AuctionException;

/**
 * 查询用户的全部出价
 * @param name 竞价者的用户名
 * @param pass 竞价者的密码
```

[0026]

```
     * @return 指定用户的全部竞价
     */
    List<BidBean> getBidByUser(String name , String pass)
        throws AuctionException;

    /**
     * 根据用户查找目前仍在拍卖中的全部物品
     * @param name 所属者的用户名
     * @param pass 所属者的密码
     * @return 属于当前用户的、处于拍卖中的全部物品。
     */
    List<ItemBean> getItemsByOwner(String name , String pass)
        throws AuctionException;

    /**
     * 查询全部种类
     * @return 系统中全部全部种类
     */
    List<KindBean> getAllKind() throws AuctionException;

    /**
     * 添加物品
     * @param name 物品名称
     * @param desc 物品描述
     * @param remark 物品备注
     * @param initPrice 起拍价格
     * @param avail 有效天数
     * @param kind 物品种类
```

[0027]

```
     * @param username 添加物品用户的用户名
     * @param pass 添加物品用户的密码
     * @return 新增物品的主键
     */
    int addItem(String name , String desc , String remark
      , double initPrice , int avail , int kind , String username
      , String pass) throws AuctionException;

    /**
     * 添加种类
     * @param name 种类名称
     * @param desc 种类描述
     * @return 新增种类的主键
     */
    int addKind(String name , String desc)throws AuctionException;

    /**
     * 根据产品分类，获取处于拍卖中的全部物品
     * @param kindId 种类id;
     * @return 该种类下的全部产品
     */
    List<ItemBean> getItemsByKind(int kindId) throws
AuctionException;

    /**
     * 根据物品id，获取物品
     * @param itemId 物品id;
     * @return 指定id对应的物品
```

[0028]

```
     */
    ItemBean getItem(int itemId) throws AuctionException;

    /**
     * 增加新的竞价，并对竞价用户发邮件通知
     * @param itemId 物品id;
     * @param bidPrice 竞价价格
     * @param name 参加竞价的用户的用户名
     * @param pass 参加竞价的用户的密码
     * @return 返回新增竞价记录的ID
     */
    int addBid(int itemId , double bidPrice , String name , String
pass)
  throws AuctionException;
  }
  Web Service实现类代码：
  //定义Web Services接口的实现类，Web Services的名字
  @WebService(endpointInterface = "org.wy.auction.ws.AuctionWs",
              serviceName = "auctionWs")
  public class AuctionWsImpl implements AuctionWs
  {
    private AuctionManager manager;
    public void setAuctionManager(AuctionManager manager)
    {
       this.manager = manager;
    }

    /**
```

[0029]

```
     * 根据赢取者查询物品
     * @param name 赢取者的用户名
     * @param pass 赢取者的密码
     * @return 赢取者获得的全部物品
     */
    public List<ItemBean> getItemByWiner(String name , String pass)
        throws AuctionException
    {
        int userId = validLogin(name , pass);
        if (userId > 0)
        {
            return manager.getItemByWiner(userId);
        }
        throw new AuctionException("用户名、密码不正确，访问该服务!
");
    }

    /**
     * 查询流拍的全部物品
     * @return 全部流拍物品
     */
    public List<ItemBean> getFailItems()throws AuctionException
    {
        return manager.getFailItems();
    }

    /**
     * 根据用户名，密码验证登录是否成功
```

[0030]

```
     * @param username 登录的用户名
     * @param pass 登录的密码
     * @return 登录成功返回用户ID，否则返回-1
     */
    public int validLogin(String username , String pass)
        throws AuctionException
    {
        return manager.validLogin(username , pass);
    }

    /**
     * 查询用户的全部出价
     * @param name 竞价者的用户名
     * @param pass 竞价者的密码
     * @return 指定用户的全部竞价
     */
    public List<BidBean> getBidByUser(String name , String pass)
        throws AuctionException
    {
        int userId = validLogin(name , pass);
        if (userId > 0)
        {
            return manager.getBidByUser(userId);
        }
        throw new AuctionException("用户名、密码不正确，访问该服务！
");
    }
```

[0031]

```
        /**
         * 根据用户查找目前仍在拍卖中的全部物品
         * @param name 所属者的用户名
         * @param pass 所属者的密码
         * @return 属于当前用户的、处于拍卖中的全部物品。
         */
        public List<ItemBean> getItemsByOwner(String name , String
pass)
            throws AuctionException
        {
            int userId = validLogin(name , pass);
            if (userId > 0)
            {
                return manager.getItemsByOwner(userId);
            }
            throw new AuctionException("用户名、密码不正确，访问该服务！
");
        }


        /**
         * 查询全部种类
         * @return 系统中全部全部种类
         */
        public List<KindBean> getAllKind() throws AuctionException
        {
            return manager.getAllKind();
        }
```

[0032]

```
    /**
     * 添加物品
     * @param name 物品名称
     * @param desc 物品描述
     * @param remark 物品备注
     * @param initPrice 起拍价格
     * @param avail 有效天数
     * @param kind 物品种类
     * @param username 添加物品用户的用户名
     * @param pass 添加物品用户的密码
     * @return 新增物品的主键
     */
    public int addItem(String name , String desc , String remark
        , double initPrice , int avail , int kind , String username
        , String pass) throws AuctionException
    {
        int userId = validLogin(username , pass);
        if (userId > 0)
        {
            return manager.addItem(name , desc , remark
                , initPrice , avail , kind , userId);
        }
        throw new AuctionException("用户名、密码不正确，访问该服务！
");
    }

    /**
```

[0033]

```
     * 添加种类
     * @param name 种类名称
     * @param desc 种类描述
     * @return 新增种类的主键
     */
    public int addKind(String name , String desc)throws
AuctionException
    {
        return manager.addKind(name , desc);
    }

    /**
     * 根据产品分类，获取处于拍卖中的全部物品
     * @param kindId 种类id;
     * @return 该种类下的全部产品
     */
    public List<ItemBean> getItemsByKind(int kindId) throws
AuctionException
    {
       return manager.getItemsByKind(kindId);
    }

    /**
     * 根据物品id，获取物品
     * @param itemId 物品id;
     * @return 指定id对应的物品
     */
    public ItemBean getItem(int itemId) throws AuctionException
```

[0034]

```
        {
            return manager.getItem(itemId);
        }

        /**
         * 增加新的竞价，并对竞价用户发邮件通知
         * @param itemId 物品id;
         * @param bidPrice 竞价价格
         * @param name 参加竞价的用户的用户名
         * @param pass 参加竞价的用户的密码
         * @return 返回新增竞价记录的ID
         */
        public int addBid(int itemId , double bidPrice
            , String name , String pass) throws AuctionException
        {
            int userId = validLogin(name , pass);
            if (userId > 0)
            {
                return manager.addBid(itemId , bidPrice , userId);
            }
            throw new AuctionException("用户名、密码不正确，访问该服务！
");
        }
    }
```

[0035] 以上所述仅是本发明的优选实施方式，并不用于限制本发明，应当指出，对于本技术领域的普通技术人员来说，在不脱离本发明技术原理的前提下，还可以做出若干改进和变型，这些改进和变型也应视为本发明的保护范围。

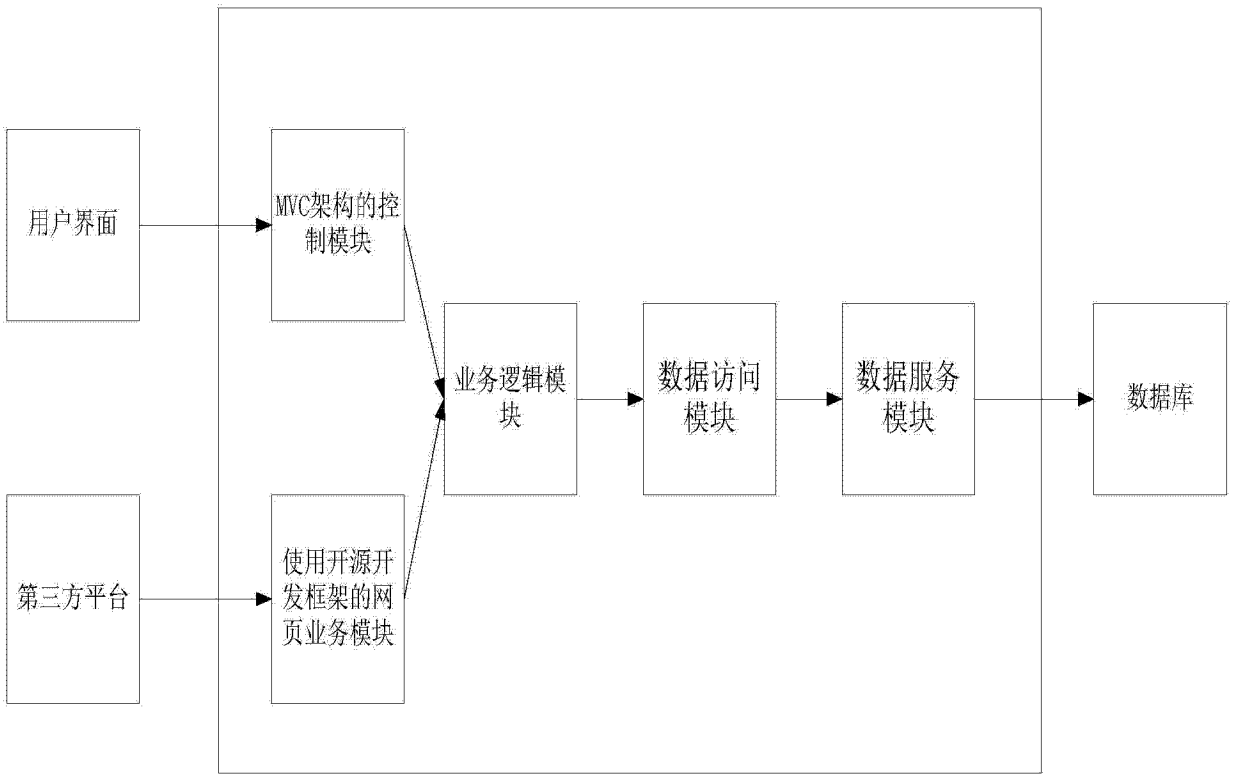

用户界面
MVC架构的控制模块
第三方平台
使用开源开发框架的网页业务模块
业务逻辑模块
数据访问模块
数据服务模块
数据库

图 1