eXo Platform——企业信息门户的实现*

罗军刚[1]，解建仓[1]，张永进[2]，李 弘[2]

(1.西安理工大学 水利水电学院，陕西 西安 710048；2.西安理工大学 管理学院，陕西 西安 710048)

摘 要：介绍了一种企业信息门户的实现技术 eXo Platform，它基于 JSR 168 和 WSRP 标准，以 JavaServer Faces 作为用户界面的开发框架，提供了丰富的个性化定制特征，支持多种客户端设备，并具有安全控制等功能。在分析了 eXo Platform 体系结构和组成模块的基础上，详细分析了 JavaServer Faces 框架的体系结构及处理流程。

关键词：eXo Platform；Portlet；JavaServer Faces；框架；MVC

中图法分类号：TP391 **文献标识码**：A **文章编号**：1001-3695(2006)02-0142-04

eXo Platform: Implementation of Enterprise Information Portal

LUO Jun-gang[1], XIE Jian-cang[1], ZHANG Yong-jin[2], LI Hong[2]

(1. *School of Water Resource & Hydro-electric Engineering, Xi'an University of Technology, Xi'an Shanxi* 710048, *China*; 2. *School of Administration, Xi'an University of Technology, Xi'an Shanxi* 710048, *China*)

Abstract: Introduces a new technology eXo platform which is an implementation of enterprise information portal. EXo platform based on standards such as JSR 168 and WSRP. It adopts JavaServer Faces framework for development of user interface. Based on the analysis of the architecture and composition modules of eXo platform, analyses the architecture and processing flow of JavaServer Faces framework.

Key words: eXo Platform; Portlet; JavaServer Faces; Framework; MVC

在新经济时代，信息的重要性日益突现出来。人们对于信息的集成程度要求越来越高，企业竞争力的提高越来越多地依赖于企业信息的采集和处理能力。随着 Internet/Intranet 技术的迅速发展，通过统一的 Web 界面，提供企业的信息、业务、组织结构以及相应授权的统一入口、整体形象和集成平台，也就是企业信息门户(Enterprise Information Portal, EIP)，是企业信息化建设的大趋势，也是企业信息化工程的重要组成部分。EIP 的作用就是整合各种信息和应用，同时实现对知识的创作、获取、传递和应用。它为企业提供单一的访问各种信息资源的入口，企业的员工、客户、合作伙伴和供应商都可以通过这个入口获得个性化的信息和服务。EIP 通过及时地向用户提供准确的信息来优化企业运作，成为企业信息管理的重要平台。现在，门户不仅充当人们获取信息的入口点，同时也是一种集成企业应用的基础框架。eXo Platform 就是这样一种企业信息门户的实现技术。

1 eXo Platform 概述

eXo Platform 是一个由多个模块构建而成的遵循 JSR 168 标准的开源企业门户[1]。它基于业界革新工具、API 和框架(如 JavaServer Faces, PicoContainer, JBossMX 和 AspectJ)，包含内容管理系统(CMS)、Portlet 容器、Portal、热部署服务、Portlet 框架、定制工具、工作流服务、增强的通信工具和许多绑定的 Portlets，是 J2EE 平台上领先的开源 Web 门户解决方案之一。

eXo Platform 支持多种终端设备，如计算机上的 HTTP 浏览器，手机上的 WAP 浏览器等。终端用户看到的是一个个性化十足的页面，所有信息都集成在这个页面上，用户可以任意地设定哪些信息显示，哪些信息不显示，以及这些信息在页面中的布局和显示方式等。每个这样的信息在页面中都占据一个小的内嵌窗口，在这些窗口中运行的实际上是一个个 Portlet。Portlet 是一种在 Portal 中运行的小应用程序，Portal 中的信息都是通过这些 Portlet 展现给用户的。所以门户站点开发者所做的主要工作是写这些 Portlet，然后将它们部署到 eXo 中。尽管 eXo 支持 Struts 和 Cocoon 框架，但是 eXo 更信赖功能强大的 JavaServer Faces(JSF)框架。JavaServer Faces 是一个开发 Java Web 应用程序的用户界面(UI)框架[2]，它为构建 Java 应用服务的用户接口进行了简化，可以使 Java 开发者迅速构建安全的 Web 应用程序。eXo Portal 就是构建在 JSF 框架之上的。

每一个 Web 应用程序都有一个 Web.xml 配置文件，它存放在 Web-INF/conf 目录下，Web.xml 文件配置 Web 应用程序中所有 Servlet 对应的类，以及各 Servlet 对应的 URL 等。eXo Platform 中的 Web 应用程序也有一个 Web.xml 配置文件，在该文件中配置了一个 Servlet 应用程序，它对应的 Java 类为 javax.faces.webapp.FacesServlet，该类是所有 JSF 应用程序的引擎，它将所有对 URL 地址为/faces/private/* 和/faces/public/* 的请求都映射到该 Servlet 上。所以，所有对 eXo Platform 的请求都首先传递给 JSF 来处理，而后再由 JSF 传给 eXo。可见，JSF 在用户和 eXo Platform 之间起着桥梁作用。

收稿日期：2005-03-04；修返日期：2005-04-18

基金项目：国家“863”计划资助项目(2002AA113150)；国家自然科学基金资助项目(50279041)

2　Portlet

Portlet是一个基于 Java 技术的 Web 组件。它由 Portlet 容器管理, 可以处理请求并产生动态内容。Portal 使用 Portlet 作为可插入的用户界面组件, 为信息系统提供一个展现层[3]。Portlet 是 Portal Server 的核心组成部分, 它是运行在 Portal Server 上的 Portal 应用。

Portlet 是遵循模型 - 视图 - 控制器(Model-View-Controller, MVC) 设计模式完整地显示页面内容的应用程序。模型为 Portlet 检索的数据源。Portlet 的模型数据通常是从外部数据源检索并装入 JavaBean, 或者在数据到达时就以 XML 文档格式化了。视图作为 Portlet 显示数据的输出机制。显示视图通常是作为 JSP 或 XSLT 样式表来实现的, 使用 JavaBean 实现数据模型时通常较多使用前者, 将数据格式化为 XML 文档时, 较多使用后者。控制器用于连接选定的视图到数据并指定 Portlet 的操作。控制器根据目标设备或浏览器选择要显示的视图, 然后将数据模型传递给视图。视图抽取特定的显示数据并格式化为浏览器所支持的格式, 然后将其输出呈现给浏览器, 作为 Portlet 输出的门户网站聚集的一部分。

在 Portlet 的作用域内, Portlet 充当了控制器, JavaBean 作为模型和 JSP 作为视图。无论是在 Portlet 内部或是使用了一个助手, 控制器都会通过访问企业信息系统(EJB 层、数据库、ERP 等) 创建一个模型对象(JavaBeans) , 然后 Portlet 发出请求给当前的视图。JSP 页面捕获先前创建的对象并且使用它返回输出的标记语言。

Portlet 的开发、部署、管理、显示相对于其他的 Portlet 都是相互独立的。Portlet 不仅有多种状态和实现模式, 还具有事件和消息处理能力。Portlet 容器提供了 Portlet 初始化、使用和最后销毁的运行时环境。Portlet 依靠门户网站的基础结构来访问用户信息、参与窗口和操作事件和其他 Portlet 的通信、访问远程内容和存储持久数据。门户系统中 Portlet 的请求/响应模式如图 1 所示。与 Servlet 的响应模式不同, 用户发出的 Request 被交由 Portlet Container 进行处理, 并被封装为 Action Request/Render Request 对象, Portlet 据此产生各种动态信息内容。这些由 Portlet 产生的内容也被称为片段(Fragment) , 而片段通常是由标记语言(HTML, XHTML, WML) 描述, 而且可以和其他的片段组合而成一个复杂的文件, 即门户页面。

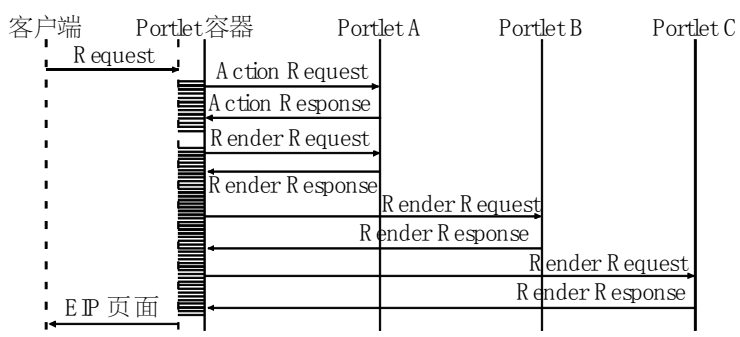

图 1　Portlet的请求/响应模式

在 eXo 中, 每个 Portlet 被分为两个部分[4]: ①包装器(Decorator) 。它是用来包装 Portlet 主体的组件, 包含窗口状态和 Portlet 的样式图标。② Portlet 主体。它是包含 Portlet 的组件。通过在 Web-INF 目录下给 Portlet 的 WAR 文件添加一个 Skin-config. xml 文件, 就可以定义一个 CSS 文件来使用它了。

eXo Platform 是一个遵循 JSR168 和 WSRP 标准的门户解决方案, 遵循 JSR168 标准的优势在于可以将自己的门户构建在 JSF 之上和使用 Velocity 与 Cocoon 开发 Portlet。就像 Servlet 是运行在 Tomcat 等容器中一样, Portlet 是运行在 Portal 中的小应用程序。eXo 的 Portal 框架提供了一套 Portlet API 来插入这些 Portlet。每一个 Portlet 对应的类都必须直接或间接地实现 javax. portlet. Portlet 接口。通常一个 Portlet 只需直接或间接地继承 javax. portlet. GenericPortlet 类, 该类实现了 javax. portlet. Portlet 接口, 并提供了 Portlet 的一些基本功能和许多用于处理请求的方法(如 doView( ) , doHelp( ) 等) 。GenericPortlet 抽象类的 render( ) 方法负责设置 Portlet 的标题和调用doDispatch( ) 方法, doDispatch( ) 方法根据 Portlet 的当前模式来调用 doView( ) , doEdit( ) 和 doHelp( ) 方法。另外 init( ) 也是一个很重要的方法, 可以在 Portlet 初始化时加载一些配置信息和执行其他一些一次性动作。

运行 Portlet 需要做两件事: ①将 Portlet 所对应的类放到 Web-INF \classes 目录中; ②在 Portlet 的注册文件中对 Portlet 进行注册。

具体的注册是在 Web-INF 目录下的 Portlet. xml 文件中添加一个注册项, 它包含一些初始化参数和 Portlet 的类名等信息。下面就是一个注册项的例子。

```
< portlet >
  < description lang = 'EN' > My First Portlet < /description >
  < portlet-name > HelloWorld < /portlet-name >
  < display-name lang = 'EN' > Hello World < / display-name >
  < portlet-class > HelloWorldPortlet < /portlet-class >
  < init-param >
    < description > something to describe < /description >
    < name > initName < /name >
    < value > initValue < /value >
  < / init-param >
  < expiration-cache > -1 < /expiration-cache >
  < supports >
    < mime-type > text/html < / mime-type >
    < portlet-mode > edit < /portlet-mode >
    < portlet-mode > help < / portlet-mode >
  < / supports >
  < supported-locale > en < /supported-locale >
  < portlet-info >
    < title > Hello World < /title >
    < short-title > Hello < / short-title >
    < keywords > hello < /keywords >
  < / portlet-info >
< /portlet >
```

同时, 还需要在 Web. xml 文件中添加一些 Portlet 的基本信息(Portlet 的名字等) , 如下所示:

```
< display-name > HelloWorld < / display-name >
  < description >  This is my first portlet. < / description >
```

最后, 将这个 Portlet 部署到应用服务器(Tomcat, JBoss 等) 的 webapps/目录下。

3　eXo Platform 的体系结构及组成模块

eXo Platform 有 eXo-express 和 eXo-enterprise 两个版本。这两个版本都是基于相同的体系结构, 如图 2 所示。eXo-express 版本不包含 Enterprise JavaBean(EJB) 容器, 它的组成模块是使用 Hibernate 构建成的。这个版本适用于开发个人和团体站点。eXo-enterprise 版是为了满足企业需要而设计的。它

基于 eXo-express 之上,提供了 EJB 容器和工作流系统。它的组成模块也是使用 Hibernate 实现的,但是也可以实现 EJB 版本。eXo-enterprise 版是一个企业级的门户,能够在任何 Intranet 上使用。

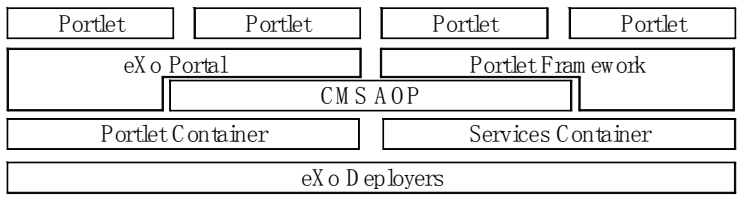

图 2 eXo Platform 体系结构

eXo Platform 的核心基于以下五个模块[5],这五个模块都独立于应用服务器。

(1) Services 容器。Services 容器基于 PicoContainer 的,以便于提供了一个反转控制(IoC)服务管理器。因此,服务不必对它们所依赖的组件实例负责。对象的创建被授权给一个单态对象 ServiceManager。每一个中间件组件,如数据库和 EJB 访问服务,都被抽象出来并且与它们的实现相分离。这个体系结构允许一个组件的松散耦合堆栈,这样依赖服务在运行时间可以被明显地交换。

(2) CMS AOP。ExoCMS 模块是一个内容管理模块,它是基于组合(Composite)模式设计的。ExoCMS 模块可以被看作一个低级别的内容管理框架,它提供了一个二进制对象的分级组织,可以被存储在任何仅需要少量简单的代码就可以实现的数据库或者文件系统内。换句话说,要存储 XML 文档,可以使用一个本地的 XML 数据库,而对象既可以存储在一个关系数据库中,也可以存储在一个对象数据库中。在 eXo 的 CMS 中,目录可以被看作节点,目录下的文件是叶子。

(3) Portlet 容器。它是一个通过认证的 Portlet API 规范(JSR 168)的开源实现,用于管理 Portlets 的生命周期以及和遵循 JSR 168 的 Portlets 进行交互。Portlet 容器和 CMS 模块进行交互的目的是为了实现 Portlet API 规范要求的不变部分。这个模块是基于 PicoContainer 的,以便于管理 Portlet 组件的生命周期和延迟实例化。Portlet 容器经过了缓存,池化的优化,并且共享 Session 的特性和高效的监视能力来提供一个强大的开发环境。

(4) Portal。eXo Portal 基于 JavaServer Faces(JSR-127),提供一个完全的 MVC 参考结构,允许开发者以处理多 Swing 客户端的方式来处理 Web 层上的组件和事件。Portal 通过与 Portlet 容器交互来得到 Portlet 的内容和从 CMS AOP 检索用户的参数,然后负责聚合页面中的 Web 组件。

(5) Portlet 框架。eXo Portlet 框架能够帮助 Portlet 开发者构建 Portlet 应用程序。它是一个 Portlet 专用设计框架,为开发者提供一种在单个 XML 文件中管理 Portlet 模式的所有行为映射的途径。这个框架是一个完全的 IoC 框架,它允许行为类使用它们自己本身还没有创建的服务。

4 JavaServer Faces

4.1 JavaServer Faces 的体系结构

JSF 的体系结构由六个部分组成[6],分别是 UI 组件模型、Rendering 模型、事件模型、验证框架、页面导航框架和国际化框架。

(1) UI 组件模型。该模型主要定义了组件的功能,由许多 JavaBean 组件组成,这些 JavaBean 具有控制各种输出、选择和分组方式的权能。所以 UI 组件的父类是一个抽象类,这个抽象类定义了状态信息和与之相关的行为。每个组件都有一个类型、一个标识符(ID)、一些局部值和类属性。UI 组件模型是可扩展的,通过继承或聚合已有的组件,可以很容易创建新的组件。

(2) Rendering 模型。该模型定义了 UI 组件的显示外表,通过创建多个展现方式,同一个 UI 组件可以不同的方式展现出来。这样做的好处是可以将组件的显示从功能中分离出来。例如,可以很容易地在 HTML 客户端(即 Web 浏览器)或 WML 客户端(即移动电话)把同一个组件表现出来。Render Kit 是一组标准的标签库(Tag Library),Render 可以将 Server 端的 UI 组件转换成任何一种用户界面标准(任何一种标记语言)。表现模型使用了 Render Kits,所以内容的显示与设备无关。

(3) 事件模型。JSF 的事件模型类似于 JavaBeans 的事件模型,Web 应用程序可以使用它定义的监听器和事件类来处理 UI 组件产生的事件。事件对象识别产生事件和存储事件信息的组件,为了将这个事件对象通知给事件,应用程序必须提供一个监听器类的实现类并且把它注册到产生事件的组件。当用户激活一个组件时,如单击一个按钮,就触发了一个事件,这将促使 JSF 调用监听器的处理事件的方法。

(4) 验证框架。该框架提供了一种机制,通过这种机制,验证器可以被注册到 UI 组件上。验证器是一些简单的类,这些类可以定义了数据类型验证、范围验证或请求字段验证。在这个形式最简单的验证框架中,一个类充当了一个验证器,这个类必须实现 Valitator 接口。

(5) 页面导航框架。该框架提供了一种简易说明的机制,这种机制不需要请求应用程序中任何特定的代码就可以指定被加载页面的顺序。导航可以完全定义在应用程序的源文件中,这个源文件是一个简单的 XML 文件,它包含一些导航规则。如果应用程序使用导航规则,不需要修改应用程序代码就可以变更页面导航。

(6) 国际化框架。这个框架提供了一种本地化应用程序中静态数据、动态数据和信息的简易机制。标准标签库可以国际化标签、提供资源绑定以及把 JSP 页面中特殊的数据和关键字关联起来,通过使用标准标签库可以本地化静态数据。由于只有在运行期才能获得数据,所以国际化动态数据是一个模型 Bean 的职责。

4.2 JavaServer Faces 的处理流程

JSF 通过六个步骤来处理一个 HTTP 请求,如图 3 所示。正常的处理流程通过实线表示,虚线表示一些诸如刷新显示、验证错误、转换错误等特殊情况的可选的处理流程。

当客户端发出的 HTTP 请求到达之后,系统会根据 JSF 网页的内容,在服务器端重新构建一个组件树,这个组件树以 UIForm 类为根,组件树的叶子是 UIComponent 的子类。在构建组件树的过程中,系统根据每个组件的设定,将对应的转换器(Converter)、验证器(Valitator)以及事件处理函数(Event Handler)全部和某个特定组件绑定起来。最后将这个组件树保存到 javax.faces.context.FacesContext 类中。

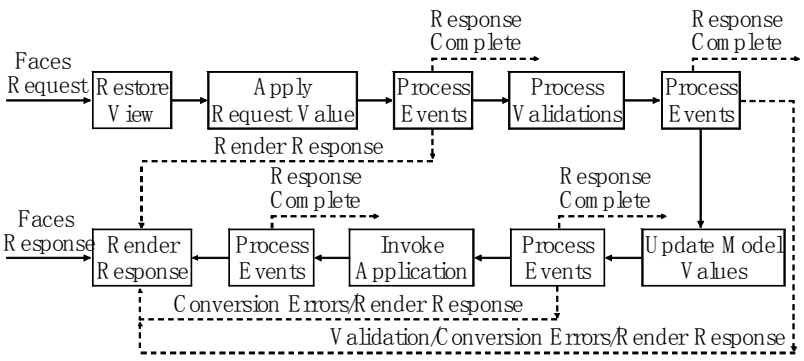

图 3　JSF 处理请求流程

在组件树构建好之后,系统会根据当前的 HTTP 请求中的信息(参数、报头、Cookies 等),通过调用组件树中每一个组件的 decode( ) 方法来获得组件的属性值。如果组件有注册转换器,那么系统会调用转换器的 getAsObject( ) 方法,此方法会返回一个对象型的参考并被保存到数据模型之中。接着系统会检查有没有注册过的需要处理的事件处理函数,如果有,系统就一一调用每一个事件处理函数。在上述整个过程中,如果任何一个 decode( ) 方法或事件监听器调用了 FacesContext 类的 responseComplete( ) 方法,那么当前的请求就立即中止。如果调用了 FacesContext 类的 renderResponse( ) 方法,那么流程将立刻跳至 Render Response 阶段。

接下来,系统会检查组件树中每个组件是否有注册验证器,如果有,就一一调用验证器的 Validate( ) 方法。在验证过程中,如果有任何验证或转换信息不合法,那么将会调用 FacesContext 的 addMessage( ) 方法,流程也将立刻跳至 Render Response 阶段。如果没有验证或转换错误,系统会检查有没有注册过的要在此阶段执行的事件处理函数,如果有,就一一调用所有的事件处理函数。在上述整个过程中,如果任何一个 validate( ) 方法或事件监听器调用了 responseComplete( ) 方法,那么当前的请求就立即中止。如果调用了 renderResponse( ) 方法,那么流程将立刻跳至 Render Response 阶段。

在验证结束后,系统将找出组件树中所有具有 valueRef 属性的组件并调用其 updateModel( ) 方法来更新数据模型的内容。接下来系统会检查有没有注册过的要在此阶段执行的事件处理函数,如果有,就一一调用所有的事件处理函数。在上述整个过程中,如果任何一个 updateModel( ) 方法或事件监听器调用了 responseComplete( ) 方法,那么当前的请求就立即中止。如果调用了 renderResponse( ) 方法,那么流程将立刻跳至 Render Response 阶段。

接下来的阶段是实现应用级的事件,如提交一个 Form 表单或连接到另一个页面。在这个阶段,系统会找出组件树中所有具有 Action 属性的 UICommand 组件。在初始化时,系统会为具有 Action 属性的 UICommand 组件注册一个预设的事件处理函数并调用此事件处理函数的 processAction( ) 方法。如果应用必须重定向到不同的 Web 应用资源,产生一个不包含任何 JSF 组件的响应,那么 processAction( ) 方法就必须调用 responseComplete( ) 方法。

最后一个阶段是对客户端作出响应和为下一个处理请求保存响应状态。在此阶段,系统会调用每一个 UIComponent 组件的 decode( ) 方法,同时将每个组件转换成适当的展示标签(HTML 标签或 WAP 标签),然后将这些标签返回到客户端。JSF 通常会以 Session 或 HTML 隐藏标签的方式保存组件树的状态。这个功能相当于在 JSF 网页中做了缓冲,可以提高请求的响应速度。

5　结束语

eXo Platform 是业界领先的开源 Portal 之一,也是符合 JSR 168 规范的 Portal 产品。它全面采用了基于 AOP 的设计思路,因此具有很好的灵活性和可配置性,深得 J2EE 开发者的认可。eXo Platform 与其他 EIP 的实现技术相比有许多优点,特别是在布局和配置方面,GUI-Enabled Web 布局、革新的 Skins 工具、灵活的导航树和 Eclipse 插件使其更具竞争优势。在 eXo Platform 的企业信息门户的实现中,JSF 为之提供了一个稳健的、可伸缩的 Web 应用架构,eXo Platform 则在 JSF 之上构筑了一个 Portal 运行框架。整个系统层次清晰,具有很好的伸缩性和可移植性。

参考文献:

[1] enjamin Mestrallet. Exo: New Open Source JSR 168 Compliant Portal and More [EB/OL]. http://www.theserverside.com/news/thread.tss? thread_id=21102, 2003-08-25.

[2] Craig McClanahan, Ed Burns, *et al*. JavaServer™ Faces Specification [EB/OL]. http://java.sun.com/j2ee/javaserverfaces/download.html, 2004-02.

[3] Java™ Portlet Specification [EB/OL]. http://jcp.org/en/jsr/detail? Id=168, 2003-10-07.

[4] Benjamin Mestrallet, Tuan Nguyen, *et al*. eXo Platform, Reload... [EB/OL]. http://www.theserverside.com/articles/ar-ticle.tss? l=eXoReloaded, 2004-08.

[5] Benjamin Mestrallet, Tuan Nguyen. Introducing the eXo Platform [EB/OL]. http://www.theserverside.com/articles/article.tss? l=eXo, 2003-09.

[6] Prithpal S Bhogill. An Introduction to Java Server Faces [EB/OL]. http://www.ociweb.com/jnb/jnbAug2003.html, 2003-08.

作者简介:

罗军刚(1981-),男,陕西人,硕士研究生,研究方向为企业信息门户、组件技术;解建仓(1963-),男,陕西人,教授,博士生导师,研究方向为决策支持系统;张永进(1970-),男,陕西人,副教授,硕士生导师,研究方向为电子商务、企业信息化;李弘(1980-),女,陕西人,硕士研究生,研究方向为电子商务、信息管理与信息系统。

(上接第 141 页)

[2] 冯旭东,陈方. 病虫害诊断的神经网络专家系统的设计与实现[J]. 计算机应用,1997,17(6):42-44.

[3] 刘振凯,贵忠华. 基于人工神经网络的知识获取方法[J]. 计算机应用研究,1999,16(5):7-9.

[4] 张晓光,李浴,徐健健. 基于神经网络的焊缝缺陷识别专家系统[J]. 计算机工程,2003,29(17):22-23.

[5] 田东,傅泽田,李道亮,等. 网络化淡水虾养殖专家系统的设计[J]. 计算机应用研究,2001,18(6):24-25 .

[6] 张映梅,李修炼,赵惠燕. 人工神经网络及其在小麦等作物病虫害预测中的应用[J]. 麦类作物学报,2002,22(4):84-87.

[7] 李军,阮晓钢. 一种基于神经网络的专家系统设计[J]. 北京工业大学学报,2003,29(2):171-174.

作者简介:

徐胜祥(1979-),男,湖北武汉人,硕士研究生,主要研究方向为资源环境信息工程;贺立源(1951-),男,湖北武汉人,教授、博士生导师;黄魏,男,湖北武汉人,讲师;陈杰(1980-),男,湖北宜昌人,硕士研究生,主要研究方向为资源环境信息工程。