JCR 技术及其在数字图书馆中的应用

汤怡洁 杨 锐 刘 毅
(中国科学院国家科学图书馆武汉分馆, 湖北 武汉 430071)

〔摘 要〕 针对目前数字图书馆领域各种信息服务平台繁多, 各个异构系统之间信息资源相互利用、共享困难的问题, 提出了内容仓库(JCR)的概念, 研究内容仓库模型及其实现。利用内容仓库建设中国科学院国家科学图书馆学科馆员工作平台的信息资源管理模块并详细阐述其实现过程, 以达到平台信息资源灵活方便的提供给其他系统使用的目的。

〔关键词〕 Java 内容仓库; 学科化服务; 资源共享; JSR-170; Jackrabbit
〔中图分类号〕G250.76 〔文献标识码〕B 〔文章编号〕1008-0821(2008)11-0077-04

JCR Technology and Its Application in Digital Libraries

Tang Yijie Yang Rui Liu Yi
(The Wuhan Branch of National Science Library, CAS, Wuhan 430071, China)

〔**Abstract**〕 In view of the lots of information service systems in the library field and the problem of the resource sharing between the heterogeneous systems, this paper introduced the definition of java content repository (JCR), studied the JCR model and its implementation. Then taking the information resource management model of subject librarian platform which developed by national science library CAS as an example, this paper expatiated the design and implementation of this model using JCR technology.

〔**Key words**〕 JCR; subject service; resource sharing; JSR-170; jackrabbit

随着计算机与互联网等现代信息技术的飞速发展, 图书馆界正发生着翻天覆地的变化。传统图书馆面对面的服务模式已经不适应信息时代和知识经济发展的需求, 面临着向数字图书馆的转型。从传统图书馆到数字图书馆, 从Library1.0到Library2.0, 图书馆经历着一次又一次的变革和创新。在图书馆转型过程中, 应运而生了相当数量的数字图书馆系统。为了给用户提供各种电子化、数字化的服务, 每个图书馆都开发建设了一批信息服务平台。

与此同时, 随着信息服务获取需求的持续上升, 越来越多的用户希望共享数据, 通过集成数据源来获取更多更有价值的信息。然而图书馆中大量信息服务平台的涌现却无形的制约了数据共享的实现。由于各平台底层数据结构的异构性, 为平台间信息资源的相互利用与共享带来了困难。

Java 内容仓库(JCR)的提出在一定程度上解决了上述问题。利用 JCR 不绑定到任何特定的底层架构的特性, 可以实现异构数据结构的平台之间的资源共享。那么什么是JCR? 利用 JCR 解决上述问题具体是如何实现的? 本文拟就这些问题作一个简单的讨论, 并以学科馆员工作平台的信息资源管理模块为例阐述实现过程。

1 内容仓库 JCR

1.1 内容仓库概念

Content Repository for Java Technology (JCR) 规范由 Java Community Process 开发为 JSR-170[1], 提供统一的 API, 允许人们访问遵循该规范的开放内容仓库。JSR-170 把自己定义为一个能与底层数据存储互相访问的、独立的、标准的方式。同时也对内容仓库做出了自己的定义, 认为内容仓库是一个高级的信息管理系统, 该系统是传统的数据仓库的扩展, 提供了诸如版本控制、全文检索、访问控制、内容分类、内容事件监视等内容服务[2]。JCR 最显著的优点是不绑定到任何特定的底层架构, 利用 JCR 实现的后端数据存储可以是文件系统、WebDAV 仓库、支持 XML 的系统或者 SQL 数据库。

1.2 内容仓库模型

JSR-170 定义的内容仓库模型是一个树形层次结构, 由一个或者多个工作区(workspace)构成。内容仓库的每个工作区都是一个单根节点的树状结构, 树上的元素(Item)可以分为节点(node)和属性(property)两类。在工作区中, 除了根节点以外的每个节点元素都可以有一个

收稿日期: 2008-07-17
作者简介: 汤怡洁(1979-), 女, 助理研究员, 研究方向: 电子商务技术、数字图书馆技术、数据库技术, 发表论文数篇。
杨 锐(1977-), 男, 馆员, 研究方向: 数字图书馆技术, 发表论文数篇。
刘 毅(1972-), 男, 副研究馆员, 研究方向: 数字图书馆技术, 发表论文数篇。

或者多个父节点，同时也可以有任意多个子节点或者属性元素。不同于节点元素，属性元素类似于树状结构中的叶子，有且仅有一个父节点，并且不能有子元素，是真正存储实际内容的元素。具体内容仓库模型如图 1 所示[3]，其中圆形代表节点，矩形代表属性。

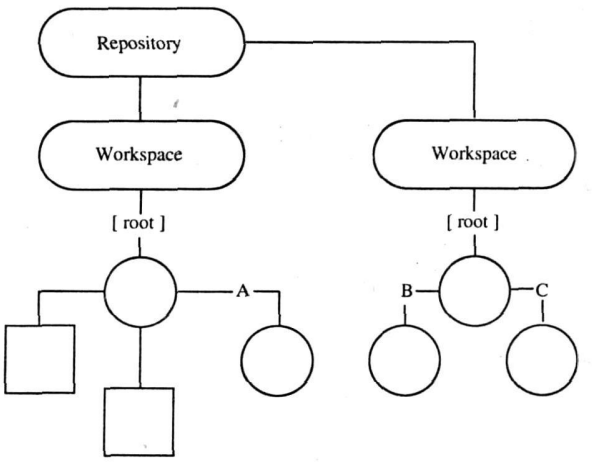

图 1 内容仓库模型

根据内容仓库实现的功能，JSR- 170 定义了三个级别的内容仓库[4]。Level 1: 定义了一个只读的内容仓库。功能包括读取内容，将内容导出为 XML 和查找内容。Level 2: 定义了可写的内容仓库，是 Level 1 的扩展，新增的功能包括往内容仓库里写入内容，和从 XML 导入数据到仓库。Optional Level 定义实现了一系列的高级功能，比如版本控制、内容锁定、允许用户监测仓库中发生的活动和 Java Transaction API。

1.3 内容仓库实现

现在支持 JCR 作为底层数据存储接口的内容管理系统以及 portal 门户系统越来越多，包括 IBM 公司的 WebSphere Portal、eXo Platform、magnolia、OpenCMS 等。其中有利用 Apache 公司的 Jackrabbit 开源项目来实现 JCR 的，也有自行编写实现的（本文以 eXo JCR 为例说明）。下面分别介绍 Jackrabbit 和 eXo JCR 的实现特性。

1.3.1 Jackrabbit

Apache Jackrabbit 是 Apache Software Foundation 开发的一个完全遵守 Java API 的内容存储规范（JCR）的实现，目前最新的 Jackrabbit 为 1.4 版本。JCR 的领导者之一，Day Software 选择了 Jackrabbit 作为商业产品的基础来实现内容仓库。Jackrabbit 提供的内容仓库是一个分等级的内容存储，支持结构化和非结构化内容的存储，全文检索，版本控制等。Jackrabbit 满足 JCR 定义的所有的 level 1，level 2 和 Optional Level 的功能，同时额外提供 session 监听，CustomNote 注册等功能。其中 jackrabbit- webapp 组件提供了友好的 Jackrabbit Web 应用；jackrabbit- ocm 组件形成对象/内容映射框架，将 Java 对象和 JCR 节点彼此映射；jackrabbit- spi 组件在 JCR 之下定义了一个架构层，为远程访问提供了一种方式；jackrabbit- core 组件支持对二进制内容的存储和查询引擎的处理[5]。

1.3.2 eXo JCR

eXo JCR 是 eXo Platform 的一部分，包含内容存储规范要求的所有强制特性和几个可选特性。eXo JCR 的实现包括以下 4 个部分：①eXo Container 继承于 org. exoplatform. container；②Repository Service 是一个标准的 eXo 服务；③Repository 是 javax. jcr. Repository 的实现；④Workspace 包含一个单一根节点的树，类似于 JCR 中的 workspace。其中 Repository 可以包含一个或多个的 workspace，Repository Service 可以管理多个 Repositories。同时 Repository Service 可以部署到多个 eXo Container 中。一个具体的 JCR 应用是通过仓库服务（Repository Service）从现有的 eXo 容器（eXo Container）中获取仓库对象。eXo JCR 还支持 JDBC 兼容数据库，如 MySQL、DB2 或 HSQL（缺省的）作为后端存储。同时提供建立在 JCR API 之上的框架，实现终端用户 JCR 应用开发的简化，帮助更好的设计应用程序，消除应用逻辑层和内容存储操作之间的耦合，允许在 JCR 之上使用一些数据交换协议[6]。

2 JCR 在数字图书馆中的应用及实现实例

随着 Library2. 0 时代的到来，对图书馆馆员的要求也越来越高。传统图书馆要求馆员是博闻强记、具备一定检索能力的杂家，为读者用户提供面对面的服务；现代化的数字图书馆要求馆员在此基础上还是某一学科领域的专家，通过网络为读者用户提供个性化服务，因此创建了图书馆学科馆员制度。学科馆员在各种各样的信息服务平台中获取资源，进行加工处理，最终提供给用户使用[7]。利用 JCR 进行异构平台资源共享，可以提高学科馆员的工作效率与服务质量，更好更快的满足用户需求。

2.1 学科馆员工作平台简介

为了适应中国科学院知识创新工程三期的需要，满足院科研用户的学科信息需求，2006 年中国科学院国家科学图书馆建立了学科馆员制度，并面向院所开展了学科化的文献信息服务。学科馆员在学科化服务中的主要工作包括资源与服务的推广、解决科研用户的信息需求问题、为用户提供个性化服务等[8]。

学科馆员工作平台是中国科学院国家科学图书馆 2006 - 2007 年信息系统建设优先发展的 9 个项目之一，为学科馆员提供网络化的个人空间，促进个人知识的组织和隐性知识的显性化，进行业务协作，提交、发布业务信息。平台采用 B/S 结构，分为五大模块：个人工作管理模块、信息资源管理模块、虚拟团队管理模块，交互协同功能模块、系统管理模块。各模块间松散耦合，大大降低了某个模块变化对系统造成的影响。平台的具体框架结构如图 2 所示。

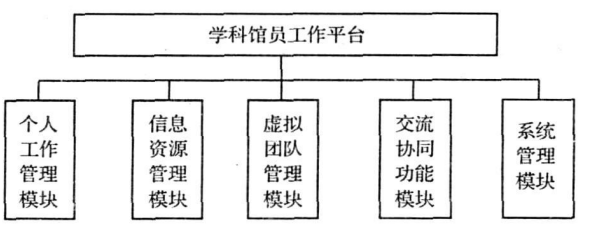

图 2 学科馆员工作平台框架结构

在学科馆员工作平台中开发信息资源管理模块的目的有两个，一方面是为学科馆员提供一个私有的空间，对自

己收集的文献资料进行上传下载以及分类整理；另一方面是将个人收集的文献资料共享给其他的学科馆员或者科研院所的科研人员与学生。模块利用内容仓库技术进行设计开发，使得信息资源不仅仅局限于学科馆员自身的知识积累，同时可以方便的跨平台提供给科研用户共享，提高学科化服务质量与效率。

2.2 模块设计

信息资源管理模块主要分为个人文档资源管理和公共文档资源管理两个部分。个人文档资源管理主要针对学科馆员个人文献资料的收集积累；公共文档资源管理针对整个平台的文献资源共享，提供一个公共的空间允许任意登录系统的学科馆员上传资料共享给其他学科馆员，同时也可以下载其他学科馆员上传的文献。通过公共文档资源的空间实现平台内部的文献资料共享机制。利用 JCR 创建内容仓库存储这些资料文献，同时提供标准接口供其他系统调用，其他平台只需要通过 JSR- 170 访问其仓库，就可以很容易的展示该仓库的内容，共享给科研用户。具体的功能结构如图 3 所示。

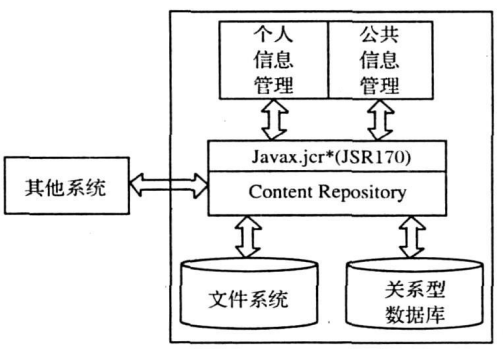

图 3 信息资源管理模块功能结构

根据 JCR 自身的特点结合实际需求，后端数据存储考虑使用文件系统和关系型数据库相结合的方式来实现。文献资料的一般说明性属性以及具体文件大小存放于关系型数据库中，具体的文献资料以二进制流的形式存放于文件系统中。

关系型数据库中创建两张数据表，其中 Folder 表主要存储文件夹的名称，创建时间，创建人，是否有上层的文件夹等基本信息；Files 表主要存储文件的显示名称，后缀名，大小，下载次数以及所属文件夹等。可以看出，在数据表的设计时，并没有为文件的内容预留属性字段，真正的文件则转化为二进制流存入文件系统中。文件的具体存储位置是根据 Jackrabbit 的配置文件 Repository. xml 中的具体配置决定的。

2.3 模块实现步骤

系统利用 Jackrabbit 开源项目作为内容仓库的实现来完成信息资源管理模块的建设。此处以文件写入内容仓库为例，阐述利用内容仓库实现信息资源管理模块的流程。

2.3.1 Jackrabbit 的配置

首先在系统的 properties 文件中定义了jackrabbit. repository. root 的路径，然后为 jackrabbit 创建了配置文件 repository. xml。这个配置文件为该内容仓库指定了底层的数据存储方式，可用的工作区以及版本控制系统等。配置文件如下所示:

```
< Repository>
    < FileSystem class=" org. apache. jackrabbit. core. fs. local. LocalFileSystem" >
        < param name=" path" value=" ${ rep. home}/ repository" />
    < /FileSystem>
    < Security appName=" Jackrabbit" >
        …
    < /Security>
    < Workspaces rootPath=" ${ rep. home}/ workspaces" defaultWorkspace=" xkgy" />
    < Workspace name=" ${ wsp. name}" >
        < FileSystem class=" org. apache. jackrabbit. core. fs. local. LocalFileSystem" >
            < param name=" path" value=" ${ wsp. home}" />
        < /FileSystem>
    < /Workspace>
    < Versioning rootPath=" ${ rep. home}/ version" >
        < FileSystem class=" org. apache. jackrabbit. core. fs. local. LocalFileSystem" >
            < param name=" path" value=" ${ rep. home}/ version" />
        < /FileSystem>
    < /Versioning>
< /Repository>
```

根据上述配置文件所示，平台的内容仓库指定底层数据存储为本地文件系统，存储路径是以系统 properties 文件中定义的 jackrabbit. repository 的根路径为基础，默认工作区设置为 xkgy。配置文件创建完毕后就可以具体进行模块开发工作。

2.3.2 JSP 页面

在完成了 Jackrabbit 的配置后，开始进行 UI 界面的开发设计。界面提供信息资源名称、描述的填写以及具体资源的上传。考虑到用户操作的方便性和平台的实用性，在页面做一个循环，将界面设计成为允许一次进行多个资源的同时上传。由于该页面为资源上传页面，Form 的定义与其他一般页面的 Form 定义稍微有些差别，主要是要将 enetype 定义为 multipart/ form- data 类型以支持文件的上传操作。

2.3.3 Struts Action 的处理[9]

当点击上传页面的上传按钮向系统发出 Action 请求后，ActionServlet 会通过查找 struts 的配置文件 struts- config. xml 来实现到具体业务逻辑 Action 的映射。此处相应的配置如下:

```
< action path=" /resource/ edit_ file"
        type=" com. xkgy. action. resource. EditFileAction" >
```

```
<forward name="resource/edit_file" path="resource/edit_file.jsp" />
<forward name="resource/error" path="resource/error.jsp" />
</action>
```

在具体的业务逻辑 EditFileAction 中，首先根据操作状态变量来判断选择进入相应的具体业务流程，然后将上传的文件内容用一个循环读取出来，放入一个 List 中保存，关键代码如下：

```
List files= new ArrayList();
for(int i= 1; i<= 3; i++)
{
    File file= uploadReq.getFile("resFile"+ i);
    String fileName= uploadReq.getFileName("resFile"+ i);
    String fileDescription= uploadReq.getFileDescription("resFile"+ i);
    byte[]bytes= FileUtil.getBytes(file);
    if((bytes!= null) && (bytes.length> 0))
    {
        ObjectValuePair ovp= new ObjectValuePair(fileName, fileDescription, bytes);
        files.add(ovp);
    }
}
```

最后将包含所有文件内容，名称和描述的 List 作为参数传入持久层方法以供进行文件内容保存的具体操作。

2.3.4 持久层的实现

根据从业务逻辑层传过来的参数，这里要对具体的内容存储进行操作。一方面是对数据库的操作，采用了 Hibernate 对象关系映射（ORM）框架[10]将文件名称，描述等基本信息保存在数据库中，此处与一般的 Hibernate 操作一致，不再重复叙述。另一方面是对内容仓库的操作[11]，将文件的名称和内容保存在文件系统中。要应用 JCR 进行操作，必须首先导入 JCR 的包，这次需要用到的主要有以下这些：

```
import javax.jcr.Node;
import javax.jcr.NodeIterator;
import javax.jcr.PathNotFoundException;
import javax.jcr.RepositoryException;
import javax.jcr.Session;
```

利用 JCR 对文件进行操作保存到本地文件系统中，其关键代码如下：

```
Node fileNode= repositoryNode.addNode(fileName, JCRConstants.NT_FILE);
Node contentNode= fileNode.addNode(JCRConstants.JCR_CONTENT, JCRConstants.NT_RESOURCE);
contentNode.addMixin(JCRConstants.MIX_VERSIONABLE);
contentNode.setProperty(JCRConstants.JCR_MIME_TYPE,"text/plain");
contentNode.setProperty(JCRConstants.JCR_DATA, is);
Version version= contentNode.checkin();
contentNode.getVersionHistory().addVersionLabel(version.getName(), Double.toString(DEFAULT_VERSION), false);
```

根据代码可以看出，内容仓库为文件名称和文件内容分别建立了一个节点，然后往内容节点中添加了类型以及二进制流形式的具体内容属性等。

到此为止，整个文件写入内容仓库的流程已经完整的实现。利用内容仓库同样可以很容易实现文件的读取，这里不再阐述。这种利用 JCR 实现的信息资源管理模块提供标准接口供其他系统的调用。

3 结 语

针对未来数字图书馆的大力发展，资源整合与资源共享势必成为发展趋势。JCR 技术作为一种异构数据结构平台信息资源共享的解决方案必然会在数字图书馆领域得到广泛应用。本文利用 Java 内容仓库的特性，开发了学科馆员工作平台的信息资源管理模块，使其具有标准接口供其他系统调用。Java 内容仓库还有许多的高级应用，如全文检索、访问控制等都值得我们深入的研究和探讨，以便于更好地提高学科馆员进行学科化服务时的服务能力以及数字图书馆的服务效率。

参 考 文 献

[1] JSR 170: Content Repository for JavaTM technology API [EB]. http://jcp.org/en/jsr/detail?id=170, 2008-04-01.

[2] What is Java Content Repository [EB]. http://www.onjava.com/pub/a/onjava/2006/10/04/what-is-java-content-repository.html?page=1, 2008-04-05.

[3] Java Content Repository API 简介 [EB]. http://www.ibm.com/developerworks/cn/java/j-jcr/, 2008-04-13.

[4] JSR-170 Levels [EB]. http://jackrabbit.apache.org/jcr-api.html, 2008-02-15.

[5] Apache Jackrabbit 官方网站 [EB]. http://jackrabbit.apache.org/, 2008-04-22.

[6] eXo JCR 官方网站 [EB]. http://docs.exoplatform.org/exo-documents/exo-jcr.site/, 2008-04-25.

[7] 李春旺. 学科化服务模式研究 [J]. 图书情报工作, 2006, (10): 14-18.

[8] 初景利. 试论新| 代学科馆员的角色定位 [J]. 图书馆理论与实践, 2007, (3): 1-3.

[9] Jreg Barish. J2EE Web 应用高级编程 [M]. 北京: 北京清华大学出版社, 2002, 10.

[10] Hibernate Reference Documentation [EB]. http://www.hibernate.org/hib_docs/v3/reference/en/html/, 2008-05-04.

[11] JSR-170 API Documentation [EB]. http://www.day.com/maven/jsr170/javadocs/jcr-1.0/, 2008-05-08.