文档修订记录

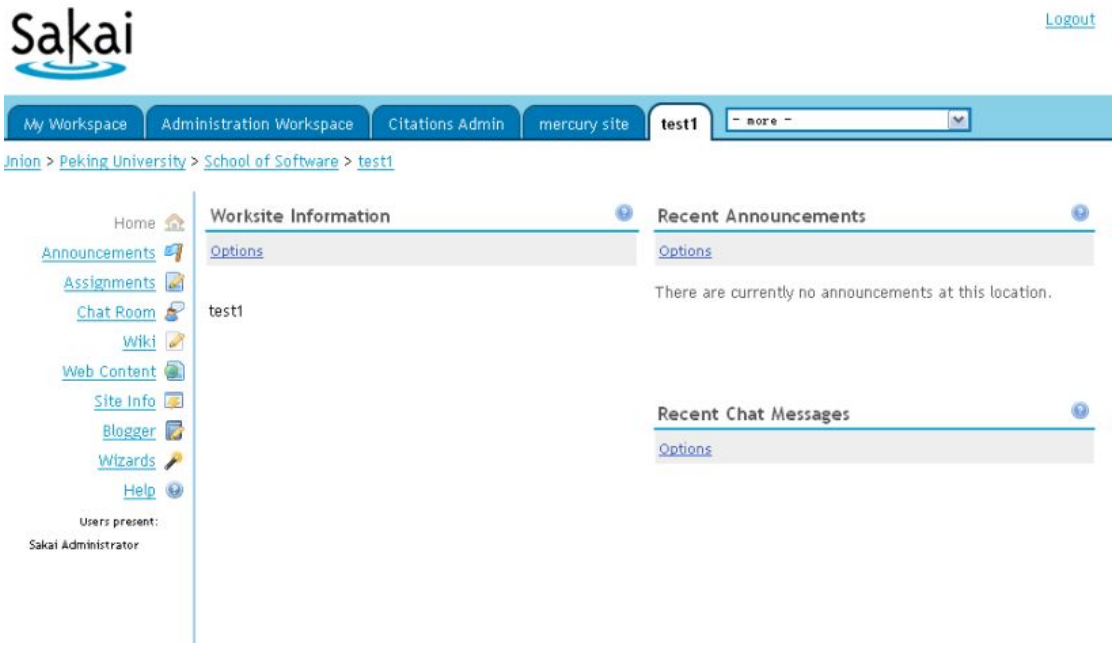
版本编号	说明：如形成文件、变更内容和变更范围	日期	变更人	批准日期	批准人
1．0.0	第一次编写文档	2007-11-10	陈俊年		
1.0.1	第二次修改文档	2008-01-09	陈俊年		
1.0.2	第三次修改文档	2008-03-04	陈俊年		
2.0	第四次修改文档	2008-09-01	陈俊年		

Sakai 开发说明

1 引言

Sakai 是由美国印地安那大学、密西根大学、斯坦福大学和麻省理工学院于 2004 年共同发起的一项开源 CMS 计划。Sakai 项目的主要目标是，开发 Sakai 程序的系统架构，将之与已有各种 CMS 工具和组件整合为一体。这些工具与组件既可用于课程管理，同时也可作为原有 CMS 模型的扩展插件。更为重要的是，Sakai 也将支持研究性协作学习。

Sakai 提供一组软件工具来帮助教师，研究员和学生创建一个用于交流、协作的站点。用户通过使用浏览器，就能从 Sakai 提供的工具中选择一个来创建符 合她们要求的站点（Site）。以下列出几个利用 Sakai 可能创建的例子：项目主管可以创建一个网站来发布通知与共享资源，比如文档或链接到其它网站的 资源。教师可以创建一个网站来让学生们可在线讨论。学生可以登录课程站点学习或者提交电子作业。



本文档作为前期工作的总结，编写的目的在于为后续开发工作的展开做好技术支持。本文档来源于前期工作的一些笔记和对英文资料的翻译。考虑到工作中可能的疏忽，文档将在后续的开发过程中不断补充和完善。

2 Sakai 安装和配置

2.1 Sakai 2.4 源代码编译

由于要在 sakai 的基础上做类似插件的二次开发。所以工作的第一步就是下载源代码，并在本机上编译通过。根据官方说明，环境采用 jdk1.5 和 maven1.0.2。下面对安装的步骤做简单的记录已备后查。

- 安装 maven1

1、解压程序包，配置环境变量%MAVEN_HOME%和把%MAVEN_HOME%\bin 加到 path 中去。

2、cmd 下运行 maven -v 查看版本号，检查是否安装成功。

3、运行 install_repo.bat %RepositoryRoot%\.maven\repository，把一些 jar 包拷入%RepositoryRoot%文件夹中（这个%RepositoryRoot%是你希望保存这些下载的 jar 的文件夹路径，通常都是 C:\Documents and Settings\UserName，UserName 当然就是你自己的账户名称了）。

- 编译

在%RepositoryRoot%下建立一个 build.properties 文件，把源和 tomcat 的位置加进去。

maven.repo.remote=http://source.sakaiproject.org/maven/

maven.tomcat.home=%TOMCAT_HOME%/（这里要把%TOMCAT_HOME%换成实际的路径。）

注意，如果是 windows 用户的话，一定要这样把“\”换成“/”。

安装 maven sakai plugin，在命令行下键入如下命令：

maven plugin:download -DgroupId=sakaiproject -DartifactId=sakai -Dversion=2.2 安装成功之后，就可以编译 sakai 了。

maven sakai

可能出现的问题是 maven 在自动编译并部署到服务器的时候，会做一些测试。如果发现本地设置不是英语，就可能会然后报错。解决的方法一是改本地设置为英语(如果是 windows XP 系统，即是在控制面板-区域和语言选项中把”区域选项“改成英语国家)，二是设置 -Dmaven.test.skip=true 即可。

- 数据库的配置

我采用的是 mysql，所以大致的步骤如下：

create database sakai default character set utf8;

grant all on sakai.* to sakaiuser@'localhost' identified by 'sakaipassword';

grant all on sakai.* to sakaiuser@'127.0.0.1' identified by 'sakaipassword';

即创建一个 sakai 数据库，把该数据库的所有权限都授予用户 sakaiuser。值得注意的是数据库 mysql 的版本最好是在 4.1.12 到 5.0 之间，尽量不要用 5.0。然后记得把 mysql-connector3.1.14.jar 复制到%TOMCAT_HOME%\common\lib 下。

- 把 sakai 源代码导入到 eclipse 中

由于我们要在此基础上作二次开发，所以导入代码到一个 IDE 中是必要的工作。

1、切换到一个新的工作空间，选择 Window -> Preferences -> Java -> Build Path -> Classpath Variables，把%RepositoryRoot%.maven\repository 加入到新建的 MAVEN_REPO classpath variable 中。

2、切换到 Java perspective，采用 package explorer，取消 project 中的 Build automatically 选项。然后开始导入工程。具体要导入的工程可以参考官方网站的列表，不过我在导入的时候报错说 manage-api 需要 cmi-api，所以在官方推荐列表后我导入了 cmi-api。

然后 clean（需要选择 clean all project 和 build immediate），再不停地 build all，直到没有错误为止。

- 部署

在%TOMCAT_HOME%下创建 sakai 文件夹，然后从代码库中复制 sakai.properties 到下面，再根据你的配置和路径作适当修改。关于 sakai.properties 的说明，参见下文。

所有的 war 包和 jar 包都会在 maven 运行后自动复制到 tomcat 的相应位置，此时所需要做得只是键入 http://localhost:8080/portal，就可以开始 sakai 的享受之旅了。

2.2 Sakai 和 LDAP 的整合

现在需要 LDAP 来统一管理和认证用户的信息，所以需要 sakai 与 ldap 密切整合。我们选用的 ldap 版本是 openLDAP。

LDAP 安装

可以直接下载 windows 版的安装程序，默认安装就好，有些版本会自动地把 Berkeyle DB 也安装好。我安装版本是 openldap-2.2.29-db-4.3.29-openssl-0.9.8a-win32_Setup.exe，也就自动安装好了 Berkeyle DB。最主要的工作是配置 slapd.conf。

配置好后，运行 slapd -d 256 即可察看是否成功。256 是记录日志的等级(0 - 256),具体的说明可以参看官方网站。Sakai 和 LDAP 对应配置

我选用的是 Alternative JLDAP Providers，可以从 sakai 网站下载整个 project，然后替换掉以前的 providers 即可。

slapd.conf 需要根据 sakai 的需要做配置。主要配置如下所示：

slapd.conf 片断

```
suffix   "dc=nldap,dc=com"
rootdn  "cn=manager,dc=nldap,dc=com"
rootpw  {MD5}lUbTrx0LmzDXkyabcdeOA==
index   cn,sn,uid,mail,displayName eq
access to attrs=userPassword
by self write
by anonymous auth
by dn.base="cn=manager,dc=nldap,dc=com" write
by * none
access to *
    by self write
    by dn.base="cn=manager,dc=nldap,dc=com" write
    by * read
```

sakai-src\providers\component\src\webapp\WEB-INF\jldap-beans.xml 片断

```
<bean id="org.sakaiproject.user.api.UserDirectoryProvider"
  class="edu.amc.sakai.user.JLDAPDirectoryProvider"
init-method="init" destroy-method="destroy" singleton="true">
<property name="ldapHost">
<value>127.0.0.1</value>
</property>
<property name="ldapPort">
<value>389</value>
</property>
<property name="ldapUser">
<value>cn=manager,dc=nldap,dc=com</value>
</property>
<property name="ldapPassword">
<value>mypassword</value>
</property>
<property name="basePath">
<value>cn=manager,dc=nldap,dc=com</value>
```

```
</property>
<property name="caseSensitiveCacheKeys">
<value>false</value>
</property>
<property name="updateUserAfterAuthentication">
<value>true</value>
</property>
<property name="ldapAttributeMapper">
<ref bean="edu.amc.sakai.user.LdapAttributeMapper" />
</property>
</bean>
      <bean id="edu.amc.sakai.user.LdapAttributeMapper
   class="edu.amc.sakai.user.SimpleLdapAttributeMapper"
init-method="init" singleton="true">
<property name="attributeMappings">
<map>
<entry key="login"><value>cn</value></entry>
<entry key="firstName"><value>givenName</value></entry>
<entry key="lastName"><value>sn</value></entry>
<entry key="email"><value>mail</value></entry>
<entry key="groupMembership"><value>displayName</value></entry>
</map>
</property>
<property name="userTypeMapper">
<ref bean="edu.amc.sakai.user.EmptyStringUserTypeMapper" />
</property>
</bean>
```

sakai.properties 片断

```
ldapHost@org.sakaiproject.user.api.UserDirectoryProvider=127.0.0.1
ldapPort@org.sakaiproject.user.api.UserDirectoryProvider=389
basePath@org.sakaiproject.user.api.UserDirectoryProvider=dc=nldap,dc=com
ldapUser@org.sakaiproject.user.api.UserDirectoryProvider=cn=manager,dc=nldap,dc=com
ldapPassword@org.sakaiproject.user.api.UserDirectoryProvider=sakai
log.config.count=1
log.config.1=DEBUG.edu.amc.sakai.user.JLDAPDirectoryProvider
```

载入 entry

命令是 slapadd -f slapd.conf -l user.ldif

user.ldif 文件主要部分

```
dn: dc=nldap,dc=com
objectclass: top
objectclass: dcObject
```

```
objectclass: organization
dn: cn=manager,dc=nldap,dc=com
objectclass: organizationalRole
cn: manager
dn: uid=wangqian,dc=nldap,dc=com
uid: wangqian
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
mail: wangqian@sina.com
userPassword: test
givenName: Qian
displayName: student
sn: Qian
cn: wangqian
```

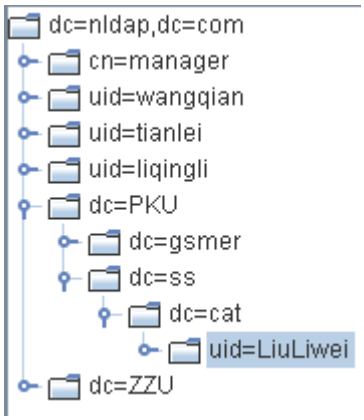
注意到前面的sladp.conf中的index行,可以发现为了查询entry需要把属性列在index中。然后可以用命令

ldapsearch -h localhost -p 389 -b "dc=nldap,dc=com" cn=wangqian 即可。如果不能工作，说明配置有误，需要检查更改。

然后再在 sakai-src\providers 下 maven sakai 即可。

与登录时用户名对应的属性(attribute)是 cn，登录密码对应的是 userPassword。其次，sakai 用户的 firstname,lastname 和 email 依次对应的 ldap entry 的 attribute 分布是 sn,givename 和 mail（这些对应信息可以在 jldap-beans.xml 做相应修改）。用户类型默认的是 student。

LDAP 目录结构和节点的数据结构如下：



Attribute	Value
mail	cjnajr@sina.com
labeledURI	student
userPassword	BINARY (4b)
uid	chenjunnian
givenName	junnian
objectClass	top
objectClass	person
objectClass	organizationalPerson
objectClass	inetOrgPerson
sn	junnian
cn	chen

现在有两个问题：

1、Sakai Administrator 无法查询那些存储在 LDAP server 中的用户信息，也没法更改他们的权限。所有的外部用户的权限都是最初定义好的。

2、当要注册一个新的账户时，新账户会直接创建在 sakai DB 中，无法在 LDAP 中创建。

2.3 sakai 2.5 安装

sakai2.4 和 2.5 的不同确实很大。sakai2.4 采用了 spring 1 和 maven 1，而 2.5 不光使用了 spring2 和 maven2，一些其它的 jar 也更换了版本，甚至在处理和 Ldap 连接的时候，设置参数和类文件都不再相同了。

所以，从版本 2.4 升级到 2.5 实在是一件麻烦事。

1、 基本环境的改变

maven1 换成 maven 2：首先就是系统的环境变量得改变，其次是在.m2 文件夹下面增加 settings.xml,用于配置所有的改变。

数据库的变动：从官方网站下载一个 sql 文件，运行一下即可。

2、Maven 1 和 2 区别

可以参考 http://bugs.sakaiproject.org/confluence/display/ENC/Converting+to+Maven+2 的说明。首先把 project.xml 文件换成 pom.xml，其次除了 maven1、2 版本本身的不同需要改动一些地方外，里面有两个地方的改动值得注意：

（1）整个项目的部署属性，

<properties>

<deploy.type>war</deploy.type>

</properties>

改成<packaging>war</packaging>.这里的 war 属性根据部署的不同需要改动，包括 sakai-component jar war 和 pom。

（2）依赖的 jar 包或者工程的部署属性：

由于<dependency></dependency>中没有 properties、url 标签，所以把

<properties>

<war.bundle>true</war.bundle>

</properties>

改成<scope>provided</scope>或者干脆去掉这一属性。设置成 provided 后该 jar 包将不会添加到 wEB_INF\lib 中去。

3、maven repository

2.4 和 2.5 的 jar 库结构变动很大，只能重新下载。自己在 2.4 版本环境下开发的工程如果打算在 2.5 版本的环境下编译通过，就必须得在 pom.xml 文件中把所有版本变换的 jar 包

换成正确的版本号。

4、ldap 的改动

在 provider 的 component 的配置文件中修改设置。

5、导入到 eclipse

步骤基本一样，但是需要注意 maven_REOP 的变动和一些 jar 的目录改动（这就是由于 repository 结构变化引起的）。

2.4 sakai properties

sakai 采用 maven 来管理和部署，每一个功能模块都可以通过添加配置文件的方式部署。而部署完成后，sakai 系统的所有属性都可以动态地配置和更改。此时大多数的参数都通过文件 sakai.properties 来配置。下面对 sakai.properties 文件的主要配置项做简要说明。需要说明的是，本文档关于 properties 这部分的内容以 sakai 2.4 为准，2.5 版本会有一些变化，但是不会影响到主要配置，需要对这部分有更深入了解可以参考官方文档。

1、数据库相关配置

关于 demo 的数据库配置是采用的 Hsqldb，缺省值在/trunk/sakai/kernel/

db-components/src/webapp/WEB-INF/components.xml。其他数据库的配置 需要在 properties 里面改动参数设置。这些参数的设置包括 url，

defaultTransactionIsolationString，vendor。sakai 的持久层采用的是 hibernate，所以在配置文件里 hibernate 的参数配置包括 hibernate.dialect，hibernate.show_sql，validationQuery 和 auto.ddl。

defaultTransactionIsolationString 是确定数据库事务操作的等级。

auto.ddl 的设置是为了确认是否需要当 sakai 系统启动后在数据库中建立必要的表，设置为 true 即建立必要的表。

针对每一种不同的数据库，上述参数的配置都不一样。具体的不同配置请参见官方说明。

2、服务器参数的相关配置

serverId 用来配置你的应用服务器的名字，特别是在集群或多服务器的时候名字应该是唯一的。

serverUrl 用来配置服务器的 url，包括传输协议，dns 和端口。这些配置要和服务器的 server.xml 中的参数保持一致。

serverName 配置服务器的 DNS 名。

gatewaySiteId 用来配置网关站点（gateway site）。site 作为 sakai 的特有术语，类似于数页网页和一些工具的集合，为的是完成一种特定的功能和业务逻辑。站点通常可以由管理员添加，删除页面和工具来整合服务。网关站点作为实现登陆功能的站点，gatewaySiteId 用来设置该站点的 id。

loggedOutUrl 用来配置当用户退出时转向到的页面。

accessPath 用来配置访问路径的 url。该参数是为访问资源的相对路径而设置的。

portalPath 用作访问 portal 的相对路径的设置，登陆 sakai 通常都是通过 portal 来实现的。

top.login 确定是否要在 gateway site 的顶上方包括进用作登陆的模块。如果系统有其他外部工具来实现安全和认证功能，可以把该项参数设置成 false，那样登陆模块就不会再在 gateway site 出现。

container.login 确定是否需要由容器来操纵登陆。换句话说就是是否需要外部工具来管理认证，默认的设置是 false，即一旦登陆后就不需要再在其它的 site 登陆认证了。

xlogin.enable 决定是否需要为登陆单设专门的页面，即 gateway site 顶部不再有用

户名和密码的输入框而改做一个链接。该项设置还有两个子参数的设置，一是 xlogin.text，一是 xlogin.icon,分别是设置链接的显示的文字和图标。该两个参数不能同时设置成 true，否则只能显示图标。

3、日志的参数配置

enabled: 在 sakai2.2 日志控制是由实现 Log4jConfigurationManager 接口的 LogConfigurationManager 来实现的。enable 参数用来决定是否需要做日志控制，该 API 可以在运行时动态地改变设置。

log.config.count 和 log.config.n

以前日志的参数配置是在其他的配置文件中，参数配置也不同。现在改在 sakai.properties 中配置这些参数。log.config.count 是设置日志记录的个数，比如是 n，那么就有从 log.config.1 到 log.config.n 的参数设置用来配置不同的级别的日志信息。比如：

log.config.count=3

log.config.1 = ALL.org.sakaiproject.log.impl

log.config.2 = OFF.org.sakaiproject

log.config.3 = DEBUG.org.sakaiproject.db.impl

这些参数的形式是 LEVEL.logger，即日志信息级别.日志类。级别包括 OFF, TRACE, DEBUG, INFO, WARN, ERROR, FATAL, ALL。

4、页脚控制

bottomnav.count 和 bottomnav.n 在缺省安装下是没有设置的。bottomnav.count 设置在页脚会有多少链接，然后从 bottomnav.1 到 bottomnav.n 就会依次设置各链接的参数。

powered.url，powered.img，powered.alt，bottom.copyrighttext，version.service 和 version.sakai

这些参数都是在对"power by sakai"的 Log 做的设置。其中 powered.url，Powered.img 和 powered.alt 都分别有 count 和 n 的子参数设置。

5、缺省安装信息文件

server.info.url，myworkspace.info.url 和 webcontent.instructions.url，这些参数分别是 gateway site -> welcome，Home page of My Workspaces -> my workspaces information 和空白的工具（即没有实现或者没有指定功能）的链接，由 sakai.iframe.service 来控制显示。

6、用户展示列表

display.users.present 设置是否需要列出当前在线用户列表，presence.inchat.icon 用于展示用户聊天时的图标。

7、外观

wysiwyg.twinpeaks 设置是否需要添加 twinpeaks 工具到编辑器的工具栏中。

wysiwyg.editor 设置编辑器的类型，HTMLarea 或者是 FCKeditor。FCKeditor 的功能要比 HTMLarea 功能强大一些。

skin.default 和 skin.repo 分别设置外观类型和外观资源所在位置。

下面几条设置都是同外观设置有关，分别是

iconNames.count，iconNames.n，iconUrls.count 和 iconUrls.n 同界面图标的设置有关。

iconSkins.count

iconSkins.n 同更改外观主题（皮肤）有关。

disable.course.site.skin.selection 确定是否在建立站点的时候更改外观的功能是

可用的。

logout.icon 设置退出的图标的 url。

java.beep 在用户界面设置一些 beep(启动，声音)是可用的。

accessibility.url 在门户 portal 处设置一个链接，指向自定义的 url。

8、指派作业工具控制

配置的参数都可以在 assignment-impl/pack/src/webapp/WEB-INF/components.xml 中找到，但是可以在 sakai.properties 中重新设置。

allowGroupAssignments 设置是否可以把作业分派给 site 下的一个或多个组。

allowGroupAssignments@org.sakaiproject.assignment.api.AssignmentService=false

allowGroupAssignmentsInGradebook 设置是否可以单独给组作业给分（把分数直接加进 gradebook)。

allowGroupAssignmentsInGradebook@org.sakaiproject.assignment.api.AssignmentService=true

9、Help Tool control

帮助文档的一些设置

helpPath

display.help.menu

display.help.icon

help.location

help.welcomepage

10、资源管理

11、习题管理

12、webdav

该参数同样有 count 和 n 的子设置，负责设置什么类型的文件不会在 webdav 中存储。

13、wiki

experimental，comments 和 notification 分别负责设置 wiki 是否可用，可添加评论和可通知。

14、建立站点的管理

activeInactiveUser

wsetup.group.support

site.setup.import.file

titleEditableSiteType

courseSiteType

wsetup.disable.joinable.count

wsetup.disable.joinable.n

roster.available.weeks.before.term.start

termterm

新建站点的时候会要求你输入学期的信息，所有关于学期的信息都使用该参数来设置。

termterm.count termterm.n

termyear.count termyear.n

termlistabbr.count termlistabbr.n

termiscurrent.count termiscurrent.n

termstarttime.count termstarttime.n

termendtime.count termendtime.n

这些参数设置的是关于在 site info -〉add participants 的各项输入的提示信息的。

noEmailInIdAccountLabel

emailInIdAccountLabel

emailInIdAccountInstru

emailInIdAccount

notifyNewUserEmail

invalidEmailInIdAccountString

15、站点工具控制

sitebrowser.termsearch.type

sitebrowser.termsearch.property

sitesearch.noshow.sitetype

设置可以被搜索的站点的类型（是否是按学期分），显示的名称和种类（工程 project 或者课程 course）。

16、Email

News Tool

Automatic User Adds

Membership tool

GradTools

WebServices

其它各种设置

stealthTools

秘密工具，其实就是不会显示在 Worksite Setup's tool lists, 但是会显示在 Admin Site Editor 中的工具。一些更具体的设置需要参考 sakai 的官方文件说明，本文档不会一一说明。

3 Sakai 的框架

3.1 综述

sakai 可以看作是一种 j2ee 的框架，可以在上面做二次开发，扩充新的应用。这些应用在 sakai 环境里被称作工具，也可以看作是一些 插件（plugs）被添加进原有的系统中。

sakai 有一些 核心的服务（core service），用于整个应用的开发。比较常见的包括：

UserDirectoryService －处理用户信息的查询；

SessionManager － 处理用户的会话；

SecurityService － 权限检查验证；

SiteService － 允许把站点整合进 sakai 环境中；

ToolManager － 查找用户的位置和工具的信息；

FunctionManager － 为开发新功能，工具提供注册管理。

还有一些标准的服务，包括：

UserDirectoryProvider －把用户信息加入到 sakai 中；

GroupProvider －对群的增加用户等操作的管理；

CourseManagementProvider － 把课程，部门及其相关信息加入到 sakai 系统中。

并且,Sakai 也有一些特殊的整合功能的接口点(point),如实体支持器(Entity Broker / Provider)，即允许你的应用添加进 sakai 环境中。Entity Broker / Provider 会创建或

者探测到事件，控制实体的 url 并附上属性，导入和导出。

类加载器 classLoader

sakai 加载到标准的 j2ee 类加载器层的组件环境中，就像 servlet 使用 URLClassLoader 加载一样。一个 sakai 工具类似于一般的 servlet，只是 url 是被处理过的，不至于会通过直接键入 url 来获得信息。工具会在 toolsname/toolname.xml 中注册后会自动连接到 applicationContext.xml，即直接利用到 spring 的框架而没有通常的 servlet 重定向等，也不需要重新加载。

一个 sakai 的组件分成三个部分，包括 API 模块，实现（Implementation /IMPL)模块和测试模块。

API 模块包括一些 java 的接口定义和一些 util 类，也可能包括 hibernate 映射文件，他们共同组成一个 jar 包被配置部署到共享区域(可以看作是整个系统的其他模块和工具能够访问的区域)。

Implementation (IMPL) 模块包括对 API 模块中的接口的实现类和 spring 的配置文件共同组成 一个 war 包被配置部署到容器中。

测试模块则包括单元测试，整合测试。

由于 sakai 利用到 spring 的特性来创建 service beans 和本地资源，使用到的特性包括依赖注入，事务控制等。可以说 sakai 的组件管理是构建在 spring 的框架之上的。

sakai 的 api 文档可以直接从官方站点获取，由于有 maven 的管理，也可以通过编译源代码来获取（maven sakai:javadoc)，这样 javadoc 文件就在<sakai source>/target/

sakai-javadoc.zip 中了。

Sakai 文件系统设置

sakai content 工程组包括了 resources、dropbox 等 tools。而且很巧的是通过参数的设置，可以选择把上载的资源储存在关系数据库中还是建立新的文件系统中。

这一点可以在 sakai.properties 中改动设置。具体参数如下：

bodyPath@org.sakaiproject.content.api.ContentHostingService=/content/sakai

bodyVolumes@org.sakaiproject.content.api.ContentHostingService=vol1,vol2,vol3

bodyPath 是文件系统的根目录,bodyVolumes 则是文件系统的容积,如果指定了 bodyPath 则至少要设定一个容积。

如果该两项参数为空的话，则默认储存先前已经指定好的关系数据库中。

3.2 开发的规范

防止命名的冲突

因为 sakai 是组件式的开发，其本身可算为一个框架，所以为避免组件之间发生命名的冲突，就有必要制定一些命名规范了。

工具的 id 必须是唯一的，所以规范的命名方式是 sakai.toolname，该文件会在 tool/src/webapp/tools/，命名为 sakai.toolname.xml；Sakai project.xml 的命名也必须是唯一的，id 的标准命名方式是 sakai-toolname-location。component.xml 位于 impl/pack/src/webapp/WEB-INF/，文件中所有的 bean id 也都是唯一的。

数据库中表，所有的 hibernate 配置文件 he 和所有的持久层 bean 的 命名规范都是加上 toolname 的前缀，即分别是 TOOLNAME_TABLEname，ToolnameItem.hbm.xml 和 ToolnameItem.java。

在 web.xml 文件中的 servlet-name 必须和 tool 的 id 相同(即和注册文件 sakai.toolname.xml 中的 id 相同)，web.xml 位于 tool/src/webapp/WEB-INF。

3.3 增加一个新的工具(tool)到 sakai 系统

由 portal 来管理授权和认证，通过 sakai 框架的功能来浏览，添加和配置工具。这样做也方便了用户的使用。

由于 sakai 的开发是插件式的开发，而且不只是支持一般的 servlet。sakai 还作为一种框架，还支持 jsf，rsf，所以扩展新的工具也至少有几种方法。下面的例子是官方网站上的例子。

1、一般的 servlet 方式开发 tool

目录结构：

tool/src/java 存放 java 源代码

tool/src/webapp/WEB-INF 存放 web.xml 文件

tool/src/webapp/tools 存放 sakai 的注册文件 sakai.cafe.tool.xml

tool/src 存放 maven 的编译文件 project.xml

这样就只需要在 servlet 中写逻辑代码,如果需要数据库的话,代码也只有写在 servlet 中。

2、加入 spring 的 IoC 容器

sakai 采用了 spring 作为 Ioc 的容器，但是并没有采用 spring MVC 的框架。所以写代码的时候还是会使用 servlet。

此时，servlet 会在 init 方法中调用

org.sakaiproject.component.cover.ComponentManager 得到容器管理器，在从容器中得到 TaskListService。通过 servce 的实例，在 doGet 或 doPost 方法中实现业务逻辑。org.sakaiproject.tool.tasklist.api 包下定义了接口，

org.sakaiproject.tool.tasklist.impl 包中的类实现了接口。

3、采用框架的开发

显然在开发的时候，我们不可能再采用上面的方法，也就是说我们不能不停的重新发明车轮。我们下面的开发将更多的注意需求分析和业务逻辑上面。所以，下面简要介绍一下采用 spring,jsf 和 hibernate 框架的开发方式。

值得说明的是，由于 sakai 采用了 jsf 来开发，可以说 sakai 是具有两套结构体系的，或者说是有两种 API。一种是处理各种业务逻辑的 class 体系，一种是关于自定义的各种标签（taglib)。

关于新的 sakai tool 的代码框架：

tool/src/java 存放 java 源代码

tool/src/webapp/WEB-INF 存放 web.xml，faces-config.xml 和

applicationContext.xml 文件。

tool/src/webapp/tasklist 存放 jsp 和 html 等网页文件。

tool/src/webapp/css 存放 css 文件。

tool/src 存放 maven 的编译文件 project.xml

tool/src/bundle/org/sakaiproject/tool/tasklist/bundle 用于存放 messages.

properties 等本地化和资源文件。

tool/src/webapp/tools 存放 sakai 的注册文件 sakai.cafe.tool.xml。

注册文件的内容包括说明该 tool 在工具列表中的 id、名称和说明（description)，还指定了该工具使用的范围，如<category name="course" />即在 course site 下可以添加，删除和使用该 tool。

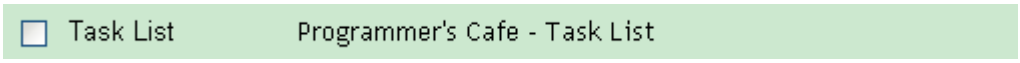
Sakai.cafe.tool.xml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<registration>
 <tool id="sakai.tasklist"
      title="Task List"
      description="Programmer's Cafe - Task List">
      <category name="course" />
      <category name="project" />
 </tool>
</registration>
```

注册文件的内容包括说明该 tool 在工具列表中的 id、名称和说明（description），还指定了该工具使用的范围，如<category name="course" />即在 course site 下可以添加，删除和使用该 tool。

在管理站点的工具页面中就会出现如下图所示的一列，分别是工具的名称和说明，同注册文件中的 title 和 description 一一对应。



下面在代码层次上对整个的开发做一下说明。

Task.hbm.xml 负责管理数据持久层和数据库的交互，在数据库中对应的表是 Tasklist_tasks,对应的 bean 是 TaskImpl,该类实现了 Task 的接口。接口 TaskListManager 的实现类 TaskListDaoImpl 负责数据持久层的操作。

TaskListService 作为一个 Service 来处理所有的业务逻辑。这个简单的 service 有三个属性，包括 TaskListManager，ToolManager 和 UserDirectoryService。其中后两者都是 sakai 已经实现的类，分别负责管理对工具和用户的逻辑操作。

对 hibernate 的管理和 service 的配置都在 spring 的配置文件中作了一一部署。

TaskListBean 作为 jsf 框架下的一个 bean，其主要属性包括一个 TaskListService 和一个 DataModel。Jsf 的配置文件则对 jsp 和对应的响应作了配置，不同的响应对应该 bean 的不同的方法，然后不同的方法又分别调用 TaskListService 的不同方法。

TaskWrapper 是一个 DataModel，作为一个包装的数据结构而存在。jsp 的每一次请求都封装成 TaskWrapper 传递给 TaskListBean，该 bean 负责解析然后调用不同的方法处理请求。

3.4 数据库 sakai

表名： sakai_user_id_map

名称	类型	空
USER_ID	varchar(99)	no
EID	varchar(255)	no

所有注册在 LDAP 中的用户和所有直接在数据库中注册的用户都在该表中保存了用户的登录名称(EID)和用户 id(USER_ID)。

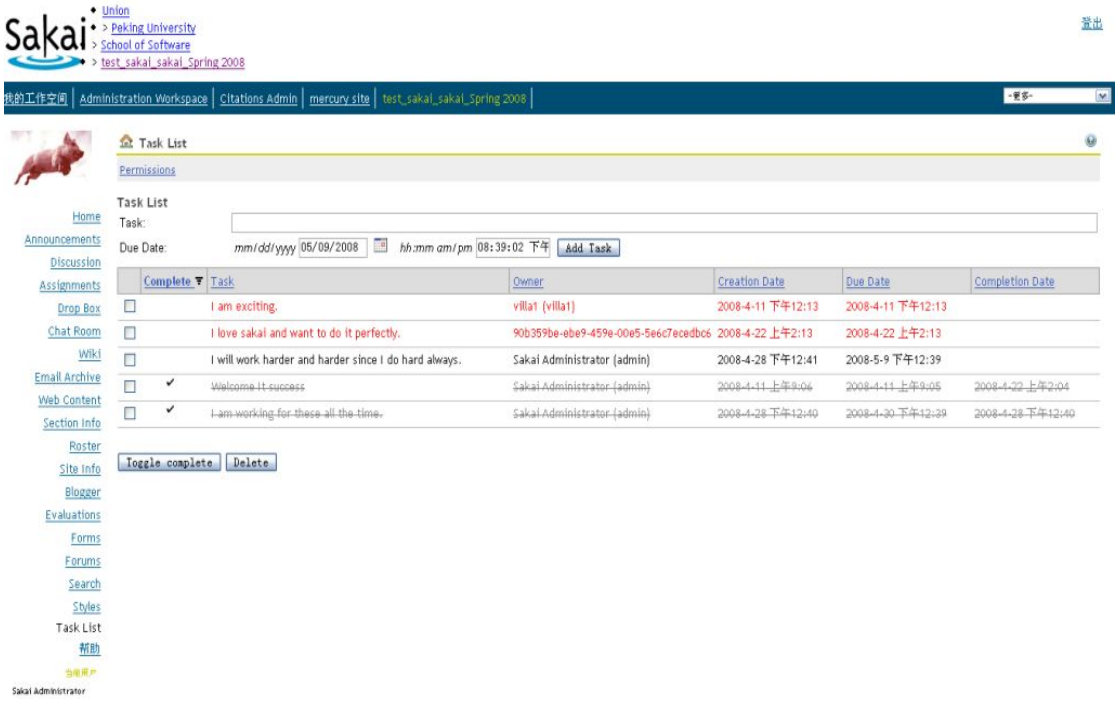
表名：sakai_user

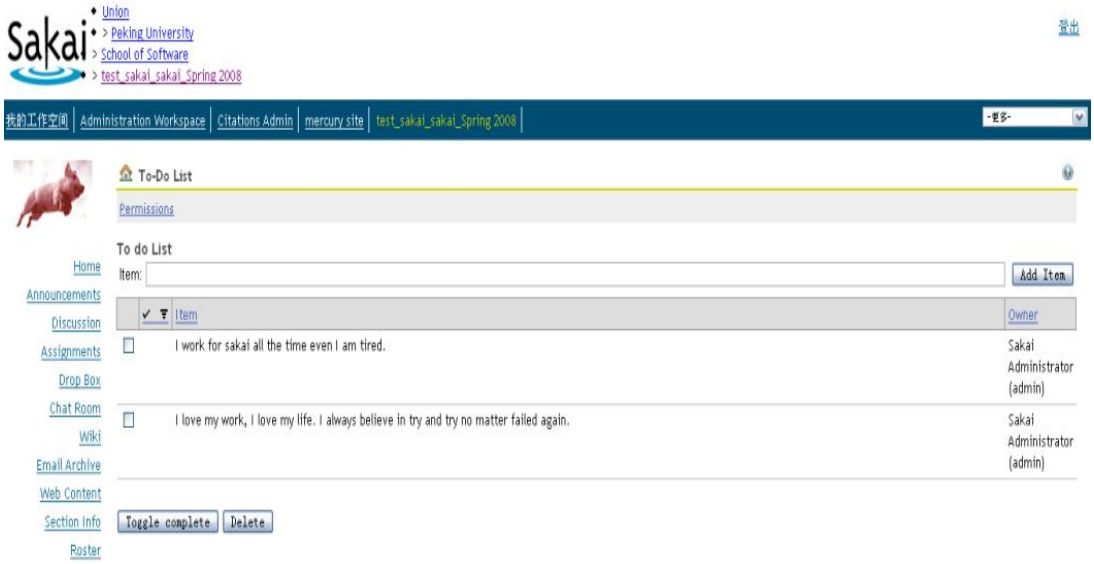
名称	类型	空
USER_ID	varchar(99)	no
EMAIL	varchar(255)	yes
EMAIL_LC	varchar(255)	yes
FIRST_NAME	varchar(255)	yes
LAST_NAME	varchar(255)	yes
TYPE	varchar(255)	yes
PW	varchar(255)	yes
CREATEDBY	varchar(99)	no
MODIFIEDBY	varchar(99)	no
CREATEDON	timestamp	no
MODIFIEDON	datetime	no

该表中存储的是所有直接注册在数据库中的用户账号信息，储存在 LDAP 服务器中的用户账号信息不在该表中。USER_ID 是用户的 ID 号，TYPE 是 sakai LMS 系统中已有的用户类型，PW 是登录密码信息。

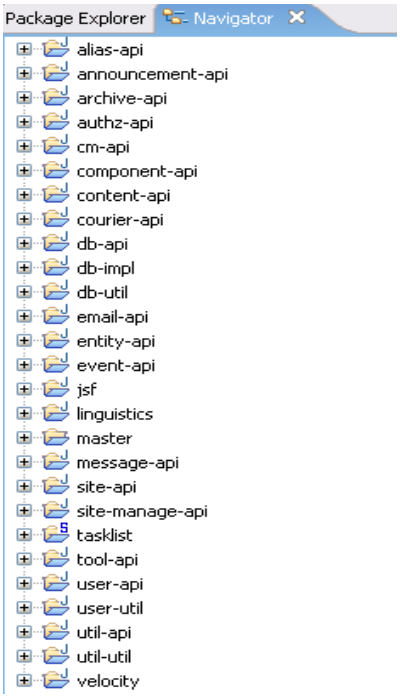
3.5 实例

sakai 的网站有一个例子，演示了怎样在 sakai 的基础上开发自己的插件。这则经典的例子由于 sakai2.4 升级到 sakai2.5，也作了实时更新。代码整体质量和实现功能都比以前版本有所提高。下面两图是该例子 tasklist 编译安装成功后的截图。

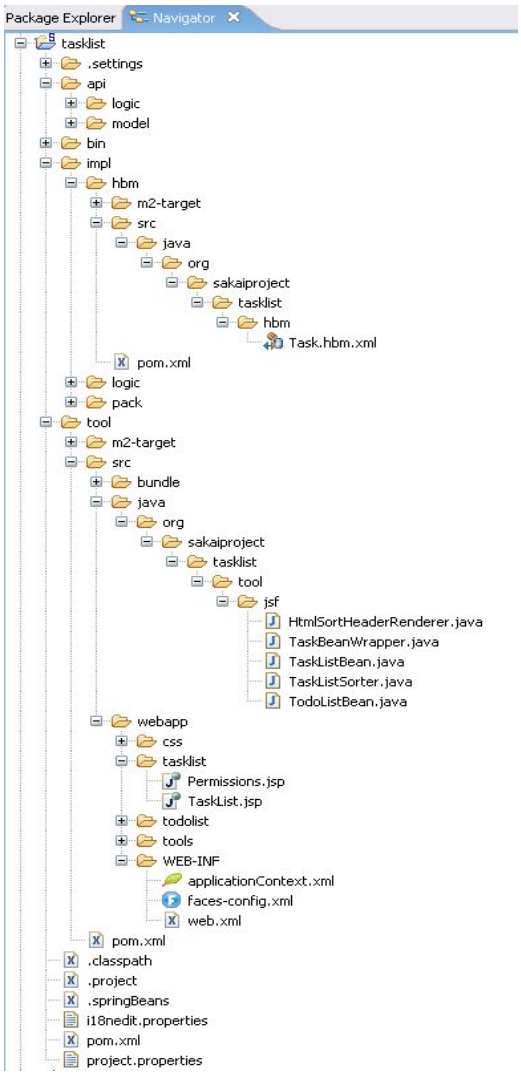




导入到 eclipse 的过程前面的已有说明，所以这里只关注整个例子的结构。下面就是 eclipse 中的文件目录结构。第一张图是 sakai 的基本 projects，开发一个新的 project 都需要这些 projects 的 jar 包和 Class。



这一张就是 tasklist 的目录结构了。api/logic，api/model，impl/hbm，impl/logic，impl/pack 和 tool 都是 tasklist 下的 project，共同组成了 tasklist project。每一个工程都有一个 pom.xml 文件组织所有相关的工程和 jar 包，所以 pom.xml 的配置很重要。



sakai tasklist 的层次结构也很清楚。tool 会作为 war 文件部署到 wepapps 中，pack 会将 tool 用到的 POJO 和 Logic Implementation 部署到 component 中。

以各自的 pom.xml 文件片断来说明：

tasklist 根目录下 pom.xml

```
<packaging>pom</packaging>
<modules>
  <module>api/logic</module>
  <module>api/model</module>
  <module>impl/hbm</module>
  <module>impl/logic</module>
  <module>impl/pack</module>
  <module>tool</module>
</modules>
```

可见作为一级 project，下有 6 个 sub-project。

tasklist/api/model/pom.xml：

```
<packaging>jar</packaging>
<properties>
   <deploy.target>shared</deploy.target>
```

```
</properties>
```

其他的几个 sub-project 部署方式也类似于 api/model，只是有些需要引用该 sub-project。这样就需要在其 pom.xml 中添加

```
<dependency>
   <groupId>org.sakaiproject</groupId>
   <artifactId>sakai-tasklist-model-api</artifactId>
   <version>${sakai.version}</version>
</dependency>
```

pack project 的作用是负责把 bean 和 dao 和一些逻辑操作的类打包部署到 sakai 的容器中，所以 pom.xml 也不同于其他 sub-project。tasklist/impl/pack/pom.xml：

```
<packaging>sakai-component</packaging>
<properties>
  <deploy.target>components</deploy.target>
</properties>
<dependencies>
   <dependency>
       <groupId>org.sakaiproject</groupId>
       <artifactId>sakai-tasklist-logic-impl</artifactId>
       <version>${sakai.version}</version>
    </dependency>
```

tool 会作为 war 部署到 tomcat 中。

```
<packaging>war</packaging>
    <dependencies>
       <dependency>
            <groupId>org.sakaiproject</groupId>
            <artifactId>sakai-tasklist-model-api</artifactId>
            <version>${sakai.version}</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>org.sakaiproject</groupId>
            <artifactId>sakai-tasklist-logic-api</artifactId>
            <version>${sakai.version}</version>
            <scope>provided</scope>
        </dependency>
```

再就是整个工程都需要的**依赖包**（通常是把他们写在 tool 和 impl、pack 的 pom 里面）：

```
        <dependency>
            <groupId>org.sakaiproject</groupId>
            <artifactId>sakai-util</artifactId>
            <version>${sakai.version}</version>
        </dependency>
        <dependency>
            <groupId>org.sakaiproject</groupId>
            <artifactId>sakai-site-api</artifactId>
```

```
    <version>${sakai.version}</version>
</dependency>
<dependency>
    <groupId>org.sakaiproject</groupId>
    <artifactId>sakai-tool-api</artifactId>
    <version>${sakai.version}</version>
</dependency>
<dependency>
    <groupId>org.sakaiproject</groupId>
    <artifactId>sakai-authz-api</artifactId>
    <version>${sakai.version}</version>
</dependency>
<dependency>
    <groupId>org.sakaiproject</groupId>
    <artifactId>sakai-user-api</artifactId>
    <version>${sakai.version}</version>
</dependency>
<dependency>
    <groupId>org.sakaiproject</groupId>
    <artifactId>sakai-entity-api</artifactId>
    <version>${sakai.version}</version>
</dependency>

<!-- JSF -->
<dependency>
    <groupId>org.sakaiproject</groupId>
    <artifactId>sakai-depend-jsf-widgets-myfaces</artifactId>
    <version>${sakai.version}</version>
    <type>pom</type>
</dependency>
<dependency>
    <groupId>org.sakaiproject</groupId>
    <artifactId>sakai-jsf-tool</artifactId>
    <version>${sakai.version}</version>
</dependency>
<dependency>
      <groupId>org.sakaiproject</groupId>
      <artifactId>sakai-jsf-app</artifactId>
      <version>${sakai.version}</version>
      <exclusions>
        <exclusion>
                <groupId>jsf</groupId>
                <artifactId>jsf-api</artifactId>
        </exclusion>
```

```
            </exclusions>
        </dependency>
        <!-- MyFaces JSF -->
        <dependency>
            <groupId>org.apache.myfaces.core</groupId>
            <artifactId>myfaces-api</artifactId>
            <version>1.1.5</version>
        </dependency>
        <dependency>
            <groupId>org.apache.myfaces.core</groupId>
            <artifactId>myfaces-impl</artifactId>
            <version>1.1.5</version>
        </dependency>
        <dependency>
            <groupId>org.apache.myfaces.tomahawk</groupId>
            <artifactId>tomahawk</artifactId>
            <version>1.1.6</version>
            <exclusions>
                <exclusion>
                    <!-- shared -->
                    <groupId>antlr</groupId>
                    <artifactId>antlr</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
        <dependency>
            <groupId>commons-logging</groupId>
            <artifactId>commons-logging</artifactId>
            <version>1.0.4</version>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
        <artifactId>servlet-api</artifactId>
        <version>${sakai.servletapi.version}</version>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jsp-api</artifactId>
            <version>2.0</version>
            <scope>provided</scope>
        </dependency>
```

明白了整个 tasklist 的结构，了解了每一个 sub-project 的作用，那么就很清楚了在每一个 sub 里面会实现一些什么类，写一些什么配置了。 所以 hbm 肯定是会写 hbm.xml 文件了，model 就会把所有的 bean 写好，pack 需要写 spring 的配置文件 components.xml (这

是 sakai 的框架所决定的)，tool 里就会实现 JSF 的功能，web.xml 和 faces-config.xml 就需要在此配置好了。

当然，在开发自己的 project 的时候，没有必要写这么多的 sub-project。只要 pom.xml 配置恰当的话就可以，不过一定把所有的逻辑都打包到 sakai 容器中，war 里面只能有 jsf 和视图层的东西。

4 sakai troubleshoot

4.1 关于虚拟机内存的溢出问题

更换 tomcat 5.5.20 的 bin 版本，即便是在 windows 下也最好不用安装版本。

参数的设置

JAVA_OPTS=-server -Xmx768m -XX:MaxNewSize=128m -XX:MaxPermSize=128m

CATALINA_OPTS=-server -Xmx768m -XX:MaxNewSize=128m -XX:MaxPermSize=128m

或者

-Xmx256m -Xms256m -XX:PermSize=16m -XX:MaxPermSize=128m -XX:NewSize=128m -XX:+UseConcMarkSweepGC -XX:+UseParNewGC

4.2 关于无法创建 course site 的问题

在 JAVA_OPTS 中添加参数 -Dsakai.demo=true，这样就可以默认启动 sakai OOTB 来配置学期的信息，进而就可以创建课程了。cm_academic_session_t 表中存储所有学期时间信息。

但是，仍然可能出现无法创建 site 的问题。有一种很笨但是易操作的方法来帮助解决该问题。

表 cm_academic_session_t 保存的数据即是站点可以创建的时间，即只有在该学期时间有效的情况下才可以创建对应的课程。比如 08 年暑假有小学期，如果打算在小学期开设一门课程，那么该课程站点只有在该学期的起始时间以内才可以创建。

名称	类型	空	默认值	属性
PRIMARY	ACADEMIC_SESSION_ID			unique
ENTERPRISE_ID	ENTERPRISE_ID			unique
ACADEMIC_SESSION_ID	bigint(20)	no		auto_increment
VERSION	int(11)	no	0	
LAST_MODIFIED_BY	varchar(255)	yes		
LAST_MODIFIED_DATE	date	yes		
CREATED_BY	varchar(255)	yes		
CREATED_DATE	date	yes		
ENTERPRISE_ID	varchar(255)	no		
TITLE	varchar(255)	no		
DESCRIPTION	varchar(255)	no		
START_DATE	date	yes		
END_DATE	date	yes		

VERSION	LAST_MODIFIED_BY	LAST_MODIFIED_DATE	CREATED_BY	CREATED_DATE	ENTERPRISE_ID	TITLE	DESCRIPTION	START_DATE	END_DATE
0	<NULL>	<NULL>	admin	2008-02-13	Winter 2008	Winter 2008	Winter 2008	2008-01-01	2008-04-01
0	<NULL>	<NULL>	admin	2008-02-13	Spring 2008	Spring 2008	Spring 2008	2008-04-01	2008-06-01
0	<NULL>	<NULL>	admin	2008-02-13	Summer 2008	Summer 2008	Summer 2008	2008-06-01	2008-09-01
0	<NULL>	<NULL>	admin	2008-02-13	Fall 2007	Fall 2007	Fall 2007	2007-09-01	2008-01-01
0	<NULL>	<NULL>	admin	2008-02-13	Fall 2008	Fall 2008	Fall	2008-09-01	2009-01-01
0	<NULL>	<NULL>	admin	2008-02-13	NewSpring 2008	NewSpring 2008	NewSpring 2008	2008-01-01	2008-02-29

下面就是创建站点的页面。

4.3 关于一些以 dw_something 命名的表

这些以 dw_something 命名的表都是作为数据仓库的作用而存在的。在启动 sakai 的时候，可能是由于外键冲突的原因导致建表 dw_matrix_cell 失败。解决的办法是在其他数据库中创建一张同样的 dw_matrix_cell，再把 dw_matrix_cell.frm 文件复制回 sakai 数据库中。

4.4 添加新的用户类型

当注册一个新的用户时，系统默认的类型是 registered，尽管该类型的权限不多，但是我们还是不希望新注册者具有创建站点等权限，所以我们需要新创建一种用户类型，当注册新的用户帐号时，默认的用户类型设置为 student。

先以管理员的身份登陆系统，在管理员的工作室（administrator workspace）中选择 Realm->new realms。在各个输入栏中对应的输入各项，如用户类型名为 user.template.student，然后再选择各种权限。其中，site.add 属性决定了用户是否有创建站点的权限。

再在 user mode 中选择 sakai.createuser.xml 文件，将<configuration name="create-type" value="registered" />中的 registered 改成 student 即可。然后在 user 文件夹下输入 maven sakai，重新编译部署即可。

4.5 添加子站点

一个学校会有很多不同的院系，不同的院系还会有不同的班级和课程，所以站点下面还有子站点的设计比较符合教学管理的现实。

要在 sakai.proterty 文件中设置 portal.includesubsites=true。其次是要在创建新站点的时候添加属性名 sakai:parent-id 和该属性值（即父站点的 id），那么新创建的站点就是原站点的子站点了。

参考文献

[1] Aaron Zeckoski. Introduction to Sakai and Sakai Services.
[2] Aaron Zeckoski. Sakai development tips
[3] Aaron Zeckoski. Sakai Tool Naming Tips
http://www.sakaiproject.org
http://mguessan.free.fr/nt/openldap_en.html