San Diego, CA
1 June 2013
9:00 - noon

Aaron Zeckoski, Unicon
Robert Long, Unicon
Matt Jones, Longsight

Logistics

Wifi: ???

Wifi password: ???

You are on your own for lunch.

Introductions

Presenters

Who we are

You

- Name
- Affiliation
- Sakai Experience (rank yourself: newbie, learner, expert, master)
- Bootcamp expectations

Goals

Show you what is possible

Expose you to the concepts

Point you to online materials

Demonstrate best practices

Provide the tools you need to develop in Sakai

Encourage you to participate in the community

Topics

Developer Environment: SVN, Maven, Tomcat, Eclipse

Web server anatomy

Typical troubleshooting scenarios

Configuration

Administration: sites, realms, tools, users

Localization: skinning, default language

Integration

Web Services

Entity Broker (REST)

Kernel / Indie Release / Advanced Maven practices

bootcamp
2.x series

Sakai in a slide

Name: word play on CHEF project; inspired by TV personality, the Iron Chef, Hiroyuki Sakai.

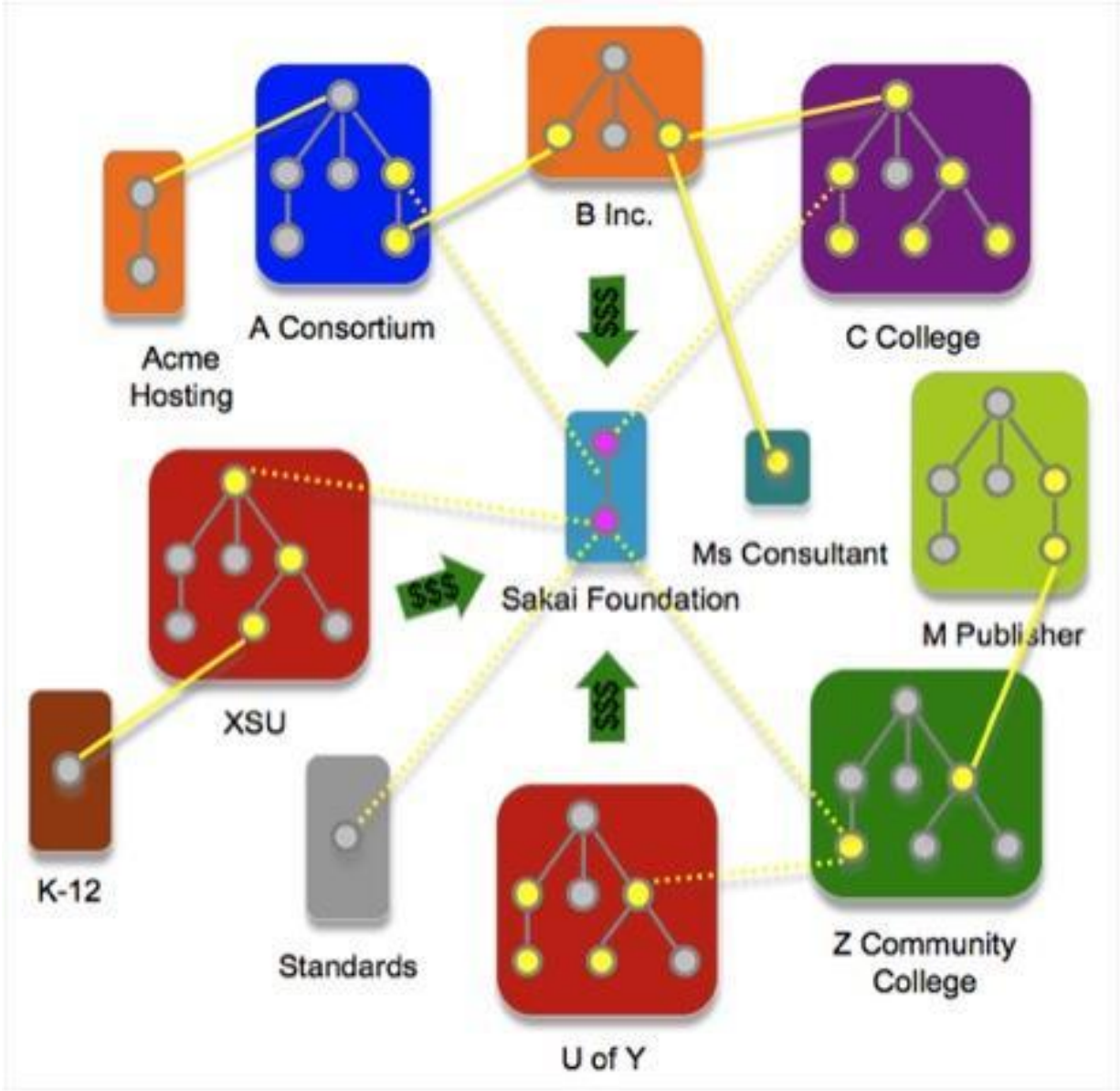
Software: open-source, enterprise-ready collaboration and learning environment; provides course, portfolio, library and ad-hoc project capabilities.

Community: universities, colleges & commercial enterprises; distributed development model; non-profit Foundation; annual conferences, regional meetings, workshops; active lists; a do-ocracy.

History: grant phase, 2003-05; Sakai 1.0 released, Aug 2004; Sakai Foundation formed, Oct 2005; world-wide adoptions, 2005-present, 12th annual conference, Los Angeles, June 2011.

Production: 300+ institutions currently running or piloting Sakai.

Sakai ecosystem



Developer resources

- **Sakai Project**: http://www.sakaiproject.org/
  - *general info, events, downloads*
- **Project wiki**: http://confluence.sakaiproject.org/confluence/
  - *project wiki*
- **Issue Tracking**: http://jira.sakaiproject.org/jira/
  - *issue tracking and scheduling*

- Programmer's cafe: http://bugs.sakaiproject.org/confluence/display/BOOT/
  - *tutorials, tips, documentation, sample code*
- Release documentation: http://confluence.sakaiproject.org/display/DOC/
  - *release notes, install guides*
- PlanetSakai: http://www.planetsakai.org/
  - *community blog aggregator*
- CLE team: http://confluence.sakaiproject.org/display/MNT/
  - *focusing on defect reduction, issue tracking improvements, seeking new contributors*

Developer resources

Mailing Lists

Development (sakai-dev): sakai-dev@collab.sakaiproject.org

Production: production@collab.sakaiproject.org

QA/Release Management: sakai-qa@collab.sakaiproject.org

To join: http://collab.sakaiproject.org/mailman/listinfo/

Code repositories

Source code (Subversion repositories)

core: https://source.sakaiproject.org/svn/

projects included in community releases

contrib: https://source.sakaiproject.org/contrib/

projects in varying stages of development

Binaries (Maven repositories)

release:
http://repo2.maven.org/maven2/
snapshots: https://oss.sonatype.org/content/repositories/snapshots/

Configuration

- What types of users and sites do you require?
- Do you need to limit tool choices based on site type?
- Do you require a consistent tool (page) order for each site type?
- What roles and role permissions are required for each site type?
- What tools are required in a user's My Workspace?
- MOTD?
- What will your gateway site look like?
- Help
- Do you plan to enable user presence?
- What is your account creation policy?
- Do you need to specify any defaults for specific tools?

Default configurations

- OOTB: HSQLDB database (in memory), binary content stored in database (except archives), no incoming or outgoing mail.
- Kernel: /component-manager kernel.properties (2.6+)
- Sakai: /config default.sakai.properties (2.6+)
- Tool: component.xml
- Override order
  - ${sakai.home}local.properties (properties specific to a single machine in a cluster)
  - overrides ${sakai.home}sakai.properties
  - overrides default.sakai.properties
  - overrides kernel.properties
- You can also define ${sakai.security}security.properties that require tighter access permissions.

sakai.properties

Customization options: branding & localization settings, database settings, file system settings, mail settings and tool/service settings. 250+ settings in total.

- server name and URL
- login fields
- active/inactive user control
- “affiliate” participants
- help on/off, support email addresses
- skins location and default skin
- footer links and copyright notice
- resources copyright notices
- semester labels and dates (pre-2.5)
- presence on/off

Conventions: adheres to standard Java property files conventions.

Location: $CATALINA_HOME/sakai (default) or elsewhere by specifying a JAVA_OPTS sakai.home system property. In the latter case, your webapp server must have read/write access to the external location.

```
-Dsakai.home=/path/to/desired/sakai/home/
```

Read: /reference/docs/architecture/sakai_properties.doc

sakai.properties

getString(): string that can include spaces and delimiters but no carriage return.

```
# Gateway site id
gatewaySiteId=!gateway
```

getBoolean(): boolean property (true/false)

```
# Let Sakai generate database objects on startup.
auto.ddl=true
# Enable/disable presence display in the portal: always / never / true
/ false
display.users.present=false
```

getStrings(): list or hierarchy of properties with the item count specified.

```
# Supported language locales for user preferences.
locales = en_US, es_ES, fr_FR, pt_PT
# Links placed on the bottom nav - set the .count to the number of
items, then add each item
bottomnav.count=2
bottomnav.1=<a href="/portal/site/!gateway">Gateway</a>
bottomnav.2=<a href="http://www.sakaiproject.org/" target="_blank">The
Sakai Project</a>
```

component property: add key/value pairs that takes the form property@bean=value

```
<bean id="org.sakaiproject.email.api.EmailService" class="org.
sakaiproject.email.impl.BasicEmailService"
    init-method="init" destroy-method="destroy" singleton="true">
        <property name="smtp"><null/></property>
</bean>
smtp@org.sakaiproject.email.impl.BasicEmailService=214.233.26.119
```

Configuration: tool component.xml

Kernel: kernel-component/src/main/webapp/WEB-INF/email-components.xml

excerpt:

```
<bean id="org.sakaiproject.email.api.EmailService"
class="org.sakaiproject.email.impl.BasicEmailService"
init-method="init" destroy-method="destroy" singleton="true">
  <property name="serverConfigurationService"  ref="org.
sakaiproject.component.api.ServerConfigurationService"/>
  <property name="smtp"><null/></property>
  <property name="smtpPort"><null/></property>
  <property name="smtpFrom"><null/></property>
  <property name="maxRecipients">
    <value>100</value>
  </property>
  <property     name="oneMessagePerConnection">
    <value>false</value>
  </property>
  <property name="testMode"><value>false</value></property>
</bean>
```

sakai-configuration.xml

COMPLEX PROPERTY HANDLING: 2.6+ now checks the sakai.home folder for sakai-configuration.xml, a Spring bean definition file. The file is read after all Sakai component definitions have been loaded but before the system starts. Use cases include (in ascending level of risk):

1. defining custom properties files
2. handling complex configuration that cannot be expressed in a string property
3. overriding or disabling standard service implementations

Use case: an institution maintains three server clusters, one for development, one for QA, and one for production. The three deployments use different database tables and each machine in a cluster needs to set its own "serverId" property, but otherwise it's important that they share the same configuration. The sakai-configuration.xml file example code will simply property organization and maintenance.

Example: custom properties files

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://www.springframework.org/schema/beans
     http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">
     <!-- Change the list of local properties files. -->
     <bean id="org.sakaiproject.component.SakaiProperties"
      parent="org.sakaiproject.component.DefaultSakaiProperties">
         <property name="locations">
             <list merge="true">
                 <value>file:${sakai.home}standard.properties</value>
                 <value>file:${sakai.home}database.properties</value>
                 <value>file:${sakai.home}server.properties</value>
             </list>
         </property>
     </bean>
</beans>
```

More info: SAK-8315, SAK-12236, SAK-12237, More flexible Sakai Configurations

Configuring academic terms

OOTB: CM_ACADEMIC_SESSION_T table is created with zero records.

Demo: add JAVA_OPTS environment setting -Dsakai.demo=true in order to generate the CM_ACADEMIC_SESSION_T table and a set of demo records.

Custom: create CM_ACADEMIC_SESSION_T with a SQL script and populate with custom values. Sample code:

https://source.sakaiproject.org/svn/msub/umich.edu/ctools/ctools-providers/trunk/component/impl/src/sql/

```
<query name="findCurrentAcademicSessions" >
   <![CDATA[
     from AcademicSessionCmImpl as term where
     (term.startDate is null or term.startDate <= current_date()) and
     (term.endDate is null or term.endDate >= current_date())
     order by term.startDate
   ]]>
</query>
```

Gotcha: since 2.5 worksite setup displays only academic terms that are current by date. To display older items do the following:
2.6+: populate CM_ACADEMIC_SESSION_T.IS_CURRENT (values 0/1)

Site setup

Type: specified when the worksite is created. Create new types by using the Sites tool.

List view: you can add additional site types to Worksite Setup by modifying sakai.sitesetup.xml (site-manage-tool).
Tool availability: specified in each tool's registry file.
Stealth: set in sakai.properties

```
stealthTools@org.sakaiproject.tool.api.ActiveToolManager= sakai.mailtool,
sakai.presentation,sakai.profile
```

also

```
visibleTools@org.sakaiproject.tool.api.ActiveToolManager=
hiddenTools@org.sakaiproject.tool.api.ActiveToolManager=
```

Tool (page) order: default specified in toolOrder.xml (kernel). Override locally with custom $CATALINA_HOME/sakai/toolOrder.xml. If site type is not defined, the page order is alpha. If a page features multiple tools, the first tool on the list will determine order. If no local toolOrder.xml new tools selected are added to bottom of list. Can also use PageOrderHelper tool.

Tool Choices: to require or pre-select tools in worksite setup tool list, add <tool> attributes selected="true" or required="true" in toolOrder.xml.

Worksite setup: site list

Location: site-manage/site-manage-tool/tool/src/webapp/tools/sakai.sitesetup.xml

```
<?xml version="1.0"?>
<registration>
     <tool
          id="sakai.sitesetup"
          title="Worksite Setup"
          description="Modify your sites and create new ones.">
          <!-- types are separated by , -->
          <!-- Steps for adding new site type: -->
          <!-- 1. add the site type into the following siteTypes value -->
          <!-- 2. add the site type as category into related tool reg files-->
          <!-- 3. if the site title is editable, add the site type into
titleEditableSiteType in
         sakai.properties file -->
          <!-- 4. if specific tool order for the site type is needed, specify it inside
toolOrder.xml file -->
     <configuration name="siteTypes" value="course,project,portfolio,bootcamp" />
          <!-- default site type -->
          <configuration name="defaultSiteType" value="project" />
          <!-- types of sites that can either be public or private -->
          <configuration name="publicChangeableSiteTypes" value="project" />
          <!-- types of sites that are always public -->
          <configuration name="publicSiteTypes" value="course" />
          <!-- types of sites that are always private -->
          <configuration name="privateSiteTypes" value="" />
          <configuration name="site_mode" value="sitesetup" type="final" />
          <category name="myworkspace" />
     </tool>
</registration>
```

Worksite setup: tool order

Kernel: component-manager/src/main/bundle/org/sakaiproject/config/toolOrder.xml

```
<?xml version="1.0"?>
<toolOrder>
     <category name="myworkspace">
          <tool id = "home" selected = "true" />
          <tool id = "sakai.sitesetup" required = "true" />
          <tool id = "sakai.membership" required="true" />
          <tool id = "sakai.preferences" required="true" />
     </category>
     <category name="course">
          <tool id = "sakai.iframe.site" />
          <tool id = "sakai.synoptic.chat" />
          <tool id = "sakai.synoptic.discussion" />
          <tool id = "sakai.synoptic.announcement" />
          <tool id = "home" selected = "true" />
          <tool id = "sakai.syllabus" />
          <tool id = "sakai.schedule" />
          <tool id = "sakai.announcements" selected = "true" />
          <tool id = "sakai.resources" />
          <tool id = "sakai.assignment" />
          <tool id = "sakai.assignment.grades" />
          <tool id = "sakai.samigo" />
          <tool id = "sakai.gradebook.tool" />
          <tool id = "sakai.dropbox" />
          <tool id = "sakai.chat" />
          . . .
          <tool id = "sakai.siteinfo" required = "true" />
     </category>
     <category name="project">
      . . .
     <category name="portfolio">
     . . .
</toolOrder>
```

Worksite setup: tool availability

Profile2: profile2/tool/src/webapp/tools/sakai.profile2.xml

```
<?xml version="1.0"?>
<registration>
     <tool id="sakai.profile2" title="Profile"
          description="Edit your profile, post status updates, search for people with
common interests, view their profile and add them as connections">
          <category name="myworkspace" />
          <configuration name="functions.require" />
     </tool>
</registration>
```

Sitestats: sitestats/sitestats-tool/src/webapp/tools/sakai.sitestats.xml

```
<?xml version="1.0"?>
<registration>
     <tool id="sakai.sitestats"
     title="Site Stats"
        description="For showing site statistics by user, event, or resource.">
          <category name="course" />
          <category name="project" />
          <!-- Sakai 2.2.x: For use with custom SAK-4120 patch -->
          <configuration name="roles.allow" value="maintain,Instructor" />
          <!-- <configuration name="roles.order" value="allow,deny" />
          <configuration name="roles.deny" value="access,guest" /> -->
          <!-- Sakai 2.3.0, up -->
          <configuration name="functions.require" value="sitestats.view" />
     </tool>
```

Site roles and permissions

Realms: sites inherit a realm template by type. Think of a realm as a collection of role-based permissions. The admin realms tool is used to add/edit/delete realms.

```
!site.template.<site> (e.g., !site.template.course)
!site.template (default)
```

Roles: site roles are defined in its associated realm template. A site maintainer (e.g. owner) can also be is also specified. Example realms include:

```
!site.template.course = instructor (maintain), student, teaching assistant
!site.template = access, maintain
```

UMich

```
!site.template.project = owner, organizer, member, observer
!site.template.course = owner, instructor, assistant, student, observer, affiliate
```

Permissions: use the realms tool to edit each site template's set of default role permissions. Many tools provide a UI for editing context-specific permissions (e.g., announcements, assignments, polls, resources, schedule, etc.).

Helper: use !site.helper to add role permission settings in all existing sites that include the target role. This simplifies the process of retrofitting old sites with updated role permissions.

My Workspace (!user)

Special case: when a user logs in for the first time their personal My Workspace site is created. My Workspace pages and tools are determined by the user's account type.

Site template: a site template is used to specify My Workspace pages and tools. Create !user.<type> templates if you want to provision My Workspaces differently for particular user types.

```
!user.<type> (e.g., !user.registered, !user.guest, !user.student)
!user (default)
```

Template editing: Use the admin sites tool to add/edit/delete !user.<type> templates. Simply copy the !user template to !user.<newtype> and then edit.

Ownership: the default (and recommended) approach is to grant the user both the maintain role and full tool permissions in their My Workspace.

Dropbox: check dropbox.maintain if a user is having problems with dropboxes in course sites.

Group roles and permissions

Group templates: enforce permissions for group-aware tools. The roles specified should match the associated site templates.

```
!group.template.course roles = !site.template.course roles
```

Group creation: use the worksite setup tool to add/edit/delete groups.

Group aware tools: announcements, assignments, forums, jForum gradebook, messages, OSP matrix & wizards, Podcasts and resources.

User (account) handling

Types: you can define user types that are scoped for the installation. Such types should not be confused with users roles which are scoped at the site level. User types can be specified during account creation.

Realms: default user (account) permissions are defined in user realm templates.

!user.template.<type> (e.g., !user.template.maintain, !user.template.guest, !user.template.registered)
!user.template (default)

Roles: each !user.template's .auth role defines a user type's application scope permission set.

realm	site.add
!user.template	N
!user.template.maintain	Y
!user.template.registered	Y
!user.template.sample	Y

Gateway site (!gateway)

Site template: !gateway.

Welcome message: the welcome message is displayed in a specially configured web content tool (sakai.iframe.service). The OOTB configuration displays /library/content/server_info.html.

Admin MOTD: create a message of the day using the Admin site's MOTD tool. This tool is a specially configured announcement's tool that publishes announcements to /announcement/channel/!site/motd.

Gateway MOTD: special display tool (sakai.motd) that displays the announcement channel /announcement/channel/!site/motd.

Multiple gateways

Public sites: installations are not limited to a single gateway. Multiple sites can be exposed to anonymous users in order to make available public capabilities and content.
SiteId: use the sites tool rather than worksite setup to create your sites as it allows for the creation of semantically meaningful siteIds.
Site List: select the sites to expose to anonymous users. Add the property gatewaySiteList and provide an comma-delimited list of ordered sites The !gateway site is not required.

```
#gatewaySiteId=!gateway
gatewaySiteList=!gateway,support,library,bootcamp
gatewaySiteListDisplayCount=8
```

Role Permissions: add the .anon role to each site's realm template and then grant site.visit to .anon together with any other tool *.read permissions you wish to expose.
Tool access: use the PageOrderHelper or add a permission to a tool's function.require setting (e.g. site.upd or *.new) not granted to .anon in order to hide tools such as Site Info. Tools such as the wiki or resources can be hidden from view yet their content made available publicly to anonymous users using the iframe tool.

Misc. info

Gateway: !gateway includes a tool (sakai.iframe.service) for displaying general info. The gateway site includes additional static content at library/content/gateway (e.g., about, acknowledgements, features, training) that should be customized.
My Workspace: !user includes a tool (sakai.iframe.myworkspace) for displaying general info. The default content location can be overridden in sakai.properties:
Web Content tool: includes instructions that appear when a URL is not specified. The default instructions can be overridden in sakai.properties:
Errors: remember to customize the !error (site unavailable) and !urlError (invalid URL) site descriptions.
Others: webdav and accessibility info files.
Editing: avoid editing the default files. Instead, copy the originals, edit them and store the new files either in Admin resources or externally. Then specify the new locations in sakai.properties:

```
server.info.url=/content/public/my_server_info.html
#server.info.url=/library/content/server_info.html
myworkspace.info.url=http://school.edu/somepath/webcontent_instructions.
html
#myworkspace.info.url=/library/content/myworkspace_info.html

webdav.instructions.url=/library/content/webdav_instructions.html
```

Skins

Location: $CATALINA_HOME/webapps/library/skin/

Layout

skin/
- tool_base.css
- myskin01/
  - images/
  - pda.css
  - portal.css
  - tool.css
- myskin02/

Types

- portal skin (portal.css)
- tool skins (tool_base.css + tool.css)
- pda skin (pda.css)

Usage

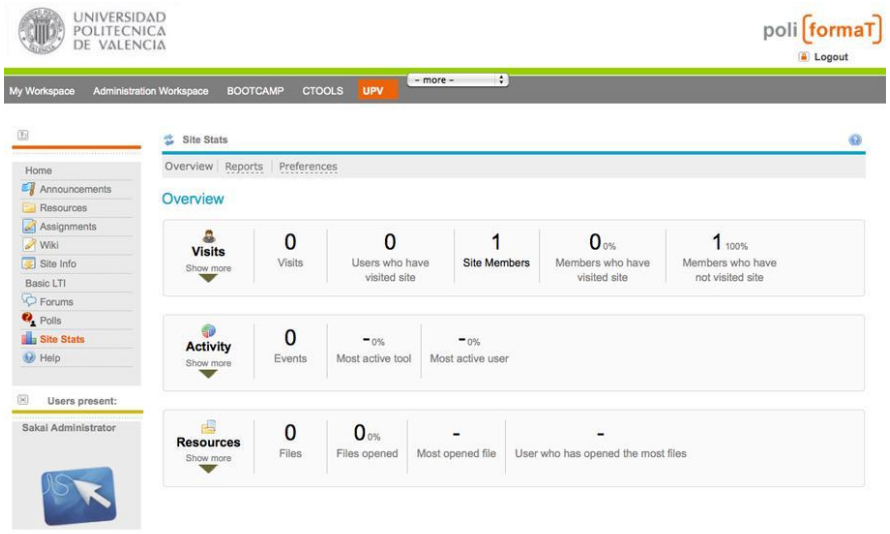
- tool_base.css for institutional styles.
- tool.css for unit, sub-group styles.

Defaults: set in sakai.properties.

skin.default=default
skin.repo=/library/skin

Skin overrides: use the sites tool.

Logos/Banners: use background images.



bootcamp 2.x series

Skins

Site type decorations: the site type is emitted as a class in the portal markup in certain top level .css blocks:

You can leverage this feature to get at anything in the portal and decorate it via a contextual selector in order to distinguish visually each site type.

```
.workspace #toolMenuWrap:hover{
     border-right:2px solid #aaa;
}
.course #toolMenuWrap:hover{
     border-right:2px solid #FC6;
}
.project #toolMenuWrap:hover{
     border-right:2px solid #9CF;
}
```

Skins: PDA (Mobile) portal

Forget me not: Sakai includes a simple portal for small devices. You should consider providing a skin for it.

Features: simplified UI for the small screen; flattened tool/site hierarchy (simplifies breadcrumbing)

Sample: $CATALINA_HOME/webapps/library/skin/default/pda.css

Default path: http://localhost:8080/portal/pda/

UMich skin: https://ctools.umich.edu/portal/pda/

More examples and skin guide:
https://confluence.sakaiproject.org/display/DOC/Sakai+2.9+skin+guide

Skins exercise

Replace the default Sakai masthead logo with a new image. In Tomcat, edit #mastLogo, #mastBanner and #mastLogo Id attributes in skin/default/portal.css to implement the change:

1. Hide the portal-provisioned logo and banner:

```
/*the portal outputs an image that can be treated here or hidden with display:none*/
#mastLogo img{
    /* margin: 1em .5em; */
    display:none;
}
/*the portal outputs an image that can be treated here or hidden with display:none*/
#mastBanner img{
     /* margin: 1em .5em; */
     display:none;
}
```

2. Specify a background image property for #mastLogo. You must also define a width and height properties that matches your image size.

```
/*container wrapping branding images - portal outputs one /library/skin/<skin
name>/images/logo_inst.gif that can be used or hidden, can also use a background:url
(images/<image name>.<ext>) in this block or both superimposing them*/

#mastLogo{
     float: left;
     margin-left: 0.2em;
     /* width: auto; */
     background: #fff url(images/bootcamp-logo-30h131w.png) top left no-repeat;
     width: 131px;
     height: 30px;
```

Sakai anatomy

- Look at the basics of Sakai webapps
- Go over some Sakai app file structure conventions
- Talk about some package naming conventions

Tomcat deployment

/shared/lib: Tomcat shared library space. More stuff than you might think will have to go here for your app to work.

- Spring framework
- Hibernate
- Some commons libraries
- Almost all APIs

/components: Sakai application context. This is where Sakai maintains its collection of beans.

- Framework
- Services
- All other service-level libraries

/webapps: Anything specific to your app gets deployed the same way it would if it were outside Sakai.

- Presentation framework (JSP, Spring MVC, RSF, JSF, Wicket, Velocity, Grails, etc.)
- No direct access to the Sakai database (use logic/dao layer)
- No business logic here (locate it in the logic service layer)

Sakai app structure

/api (interfaces)

logic - business logic and dao apis
model - POJOs (value/data objects)
public - Service API (if you have one)
hbm - Hibernate HBM files (if using hibernate)

/impl (implementations)

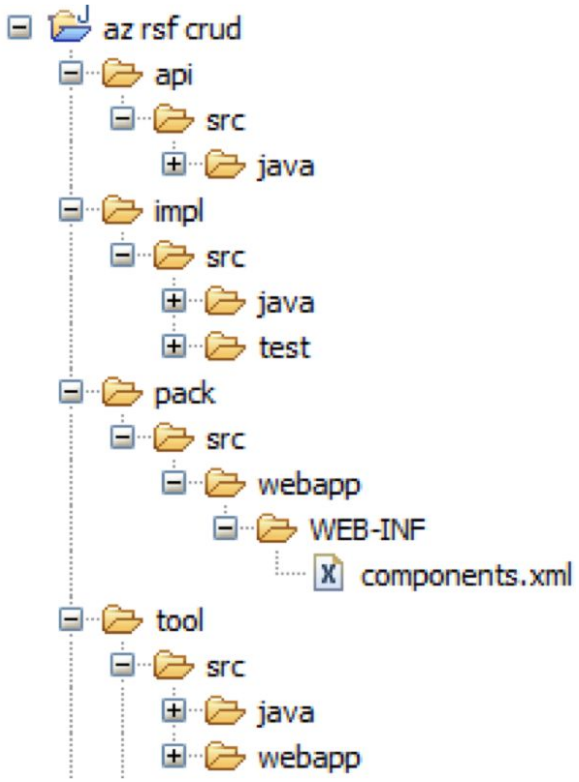
dao - data access implementation
logic - business logic implementation
tests - programmatic tests (unit/integration)

/pack (component definitions)

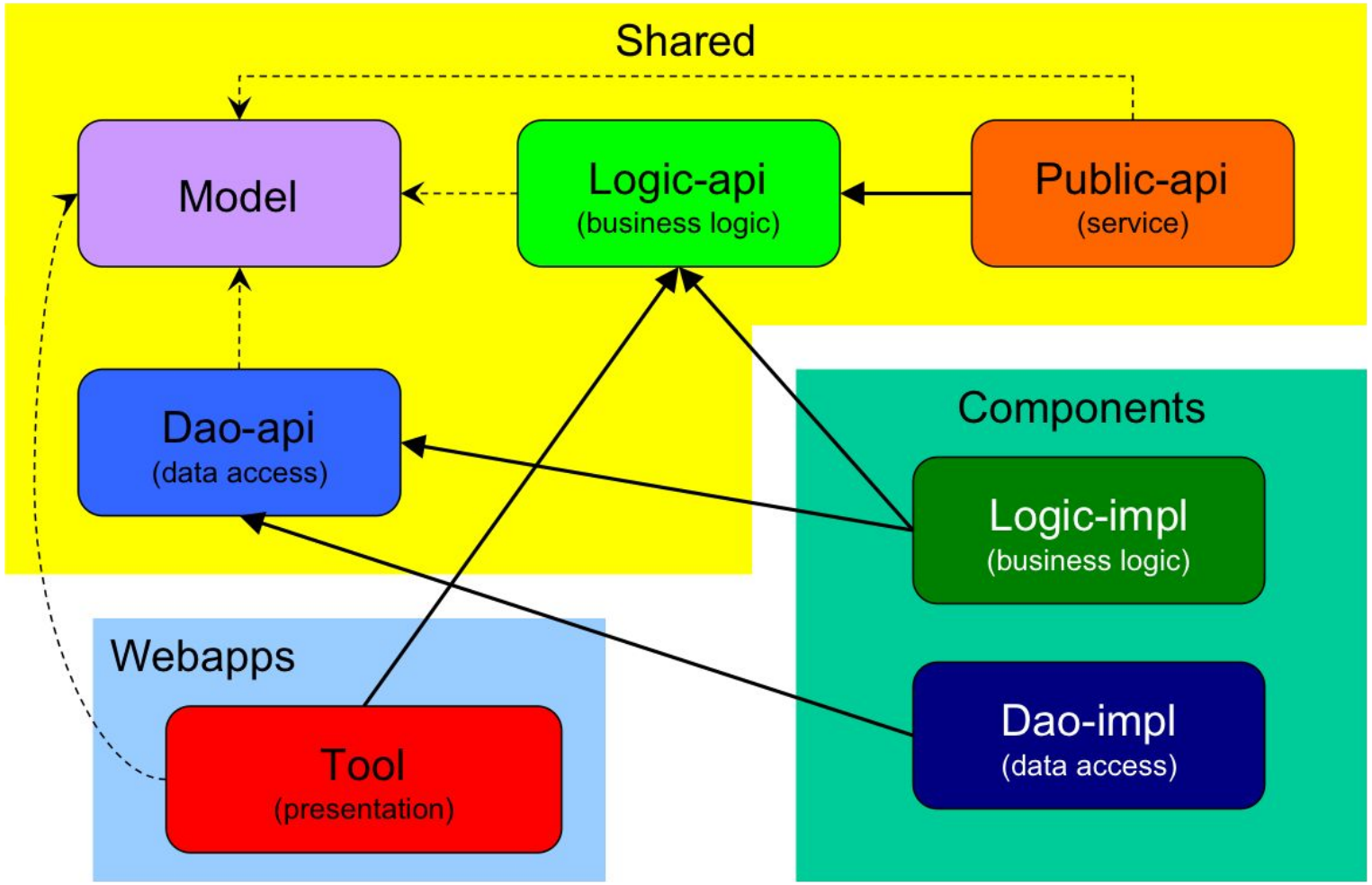
spring config files (Sakai components.xml)

/tool (webapp)

src/java - java classes specific to your tool
src/webapp - xml, jsp, html, other meta file



Application structure



Naming practices (unique ids)

Tool Id (tool/src/webapp/tools)

convention: id = sakai.<toolname>

```
<registration>
   <tool id="sakai.podcasts" title="Podcasts"
      description="For managing individual podcast and podcast feed information.">
      <category name="course" />
      <category name="project" />
      <configuration name="functions.require" value="content.read" />
      <configuration name="help.id" value="sakai.podcasts" />
   </tool>
</registration>
```

web.xml servlet (tool/src/webapp/WEB-INF)

convention: name=sakai.<toolname> (must match tool id)

```
<servlet>
   <servlet-name>sakai.poll</servlet-name>
   . . .
</servlet>
```

Spring bean (pack/src/webapp/WEB-INF)

convention: id = fully qualified classpath of the class interface

```
<bean id="org.sakaiproject.assignment.api.AssignmentEntityProvider"
    class="org.sakaiproject.assignment.impl.AssignmentEntityProviderImpl">
    <property name="assignmentService" ref="org.sakaiproject.assignment.api.
AssignmentService" />
    <property name="siteService" ref="org.sakaiproject.site.api.SiteService" />
    <property name="entityBroker" ref="org.sakaiproject.entitybroker.EntityBroker" />
</bean>
```

Naming practices (unique ids)

Hibernate hbm mapping files

convention: prefix hbm filename with tool name.

AssignmentSupplementItem.hbm.xml
ChatImpl.hbm.xml
SyllabusItemImpl.hbm.xml

Hibernate: Sakai uses common Hbm session factory so hbm file names must be unique.

Hibernate persistent classnames

convention: prefix classname with tool name.

Maven artifact coordinates

convention: groupId=org.sakaiproject.[project]; artifactid=[project]-[xxx]

```
<dependency>
   <groupId>org.sakaiproject.basiclti</groupId>
   <artifactId>basiclti-impl</artifactId>
   <version>1.1.0</version>
</dependency>
```

Sakai Component (Service) Injection

- Via Spring (preferred):
  - Service wiring in `TOMCAT/components/myapp-pack/WEB-INF/components.xml`
  - setters (getters) in the impl class
  - Create a bean to set the values

- Via the ComponentManager static cover
  - No Spring wiring needed
  - Useful upgrade for legacy code
    - Directly in class, in init/constructor

```
userDirectoryService = (UserDirectoryService)
ComponentManager.get(UserDirectoryService.class);
```

Questions?

More stuff (external integrations) next