Developing a dashboard for Sakai CLE

Jim Eng (jimeng@umich.edu)
Zhen Qian (zquian@umich.edu)
Gonzalo Silverio (gsilver@umich.edu)
Chris Kretler (ckretler@umich.edu)
The University of Michigan

Adrian Fish (a.fish@lancaster.ac.uk)
Lancaster University

jasig
Sakai
ATLANTA•2012

June 10-15, 2012

Growing Community:
Growing Possibilities

New dashboard tool

- Interested in adding dashboard to your Sakai instance?
  - Overview and brief demo
  - Brief technology design overview
  - Performance testing
  - Deploying dashboard in Sakai
- Interested in making other entities show up in dashboard?
  - Quick guide to adding entities to dashboard
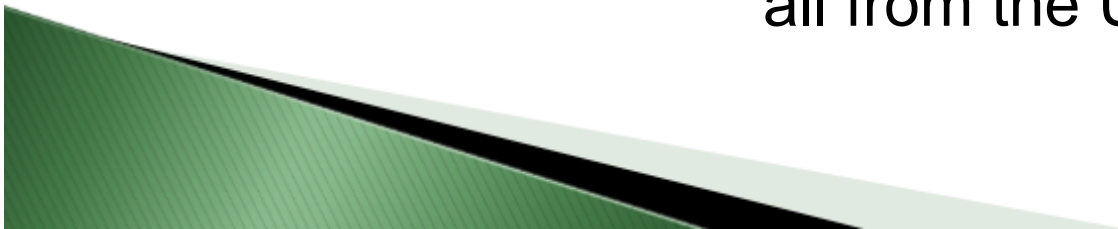 - Developer case study
- Summary
- Questions

See also:

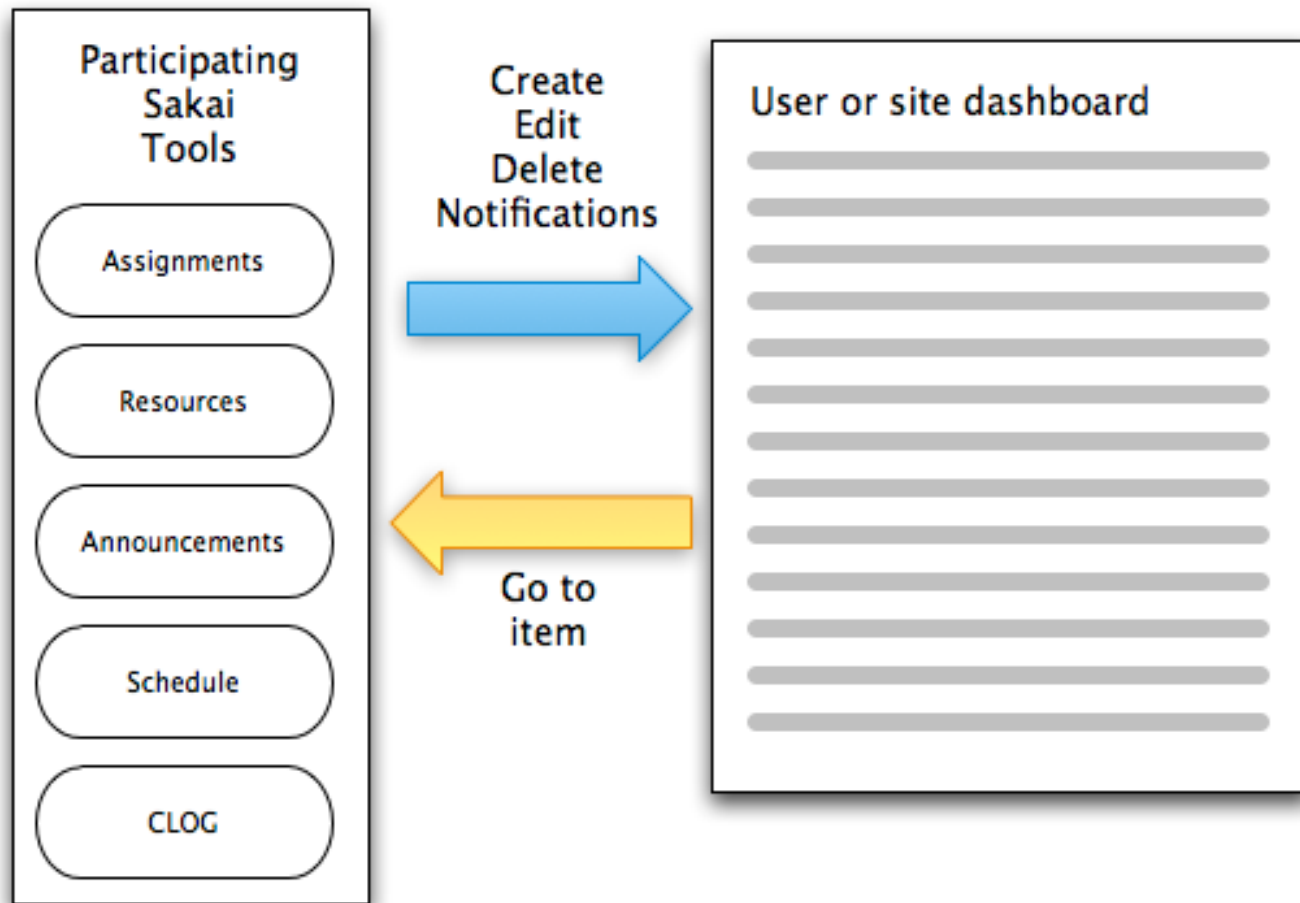*Leveraging Student Insights in Designing the Sakai Dashboard*

Thursday, 9 a.m. to 10 a.m. -- Tower 1402 (14th floor)

This presentation describes the initial design goals for the dashboard as developed by graduate students at the University of Michigan School of Information and the process of refining the design to meet user expectations. It also describes features of the current implementation.

Presenters: John Johnston, Caitlin Holman,
                      Shwetangi Savant and Tiffany Chow,
                         all from the University of Michigan

Overview and brief demo

Participating Sakai Tools

- Assignments
- Resources
- Announcements
- Schedule
- CLOG

Create
Edit
Delete
Notifications

Go to item

User or site dashboard

Dashboard placement: My Workspace

Dashboard placement: a site

Dashboard structure: future and past

Dashboard filters

Dashboard filters - details

| | | << | < | 1-5 | 6-10 | 11-14 | > | >> |

Tomorrow 10:00 AM	🏃 Lecture - Activity	ENTOM 245 001 Fall 2012	⭐	
Jun 10 06:00 AM	🏃 Spider Field Trip Cancelled - Activity	ENTOM 245 001 Fall 2012	☆	❌
Jun 11 10:00 AM	🏃 Lecture - Activity	ENTOM 245 001 Fall 2012	☆	❌

| Tomorrow 10:00 AM | 🏃 Lecture - Activity | ENTOM 245 001 Fall 2012 | ⭐ |

| Jun 09 06:00 AM | 🏃 Optional Beetle Study Group Meeting - Activity | ENTOM 245 001 Fall 2012 | ✅ |

| Jun 05 07:00 AM | 🏃 Spider Field Trip! - Activity | ENTOM 245 001 Fall 2012 | ☆ |

Dashboard information

Dashboard context and navigation

Dashboard item grouping
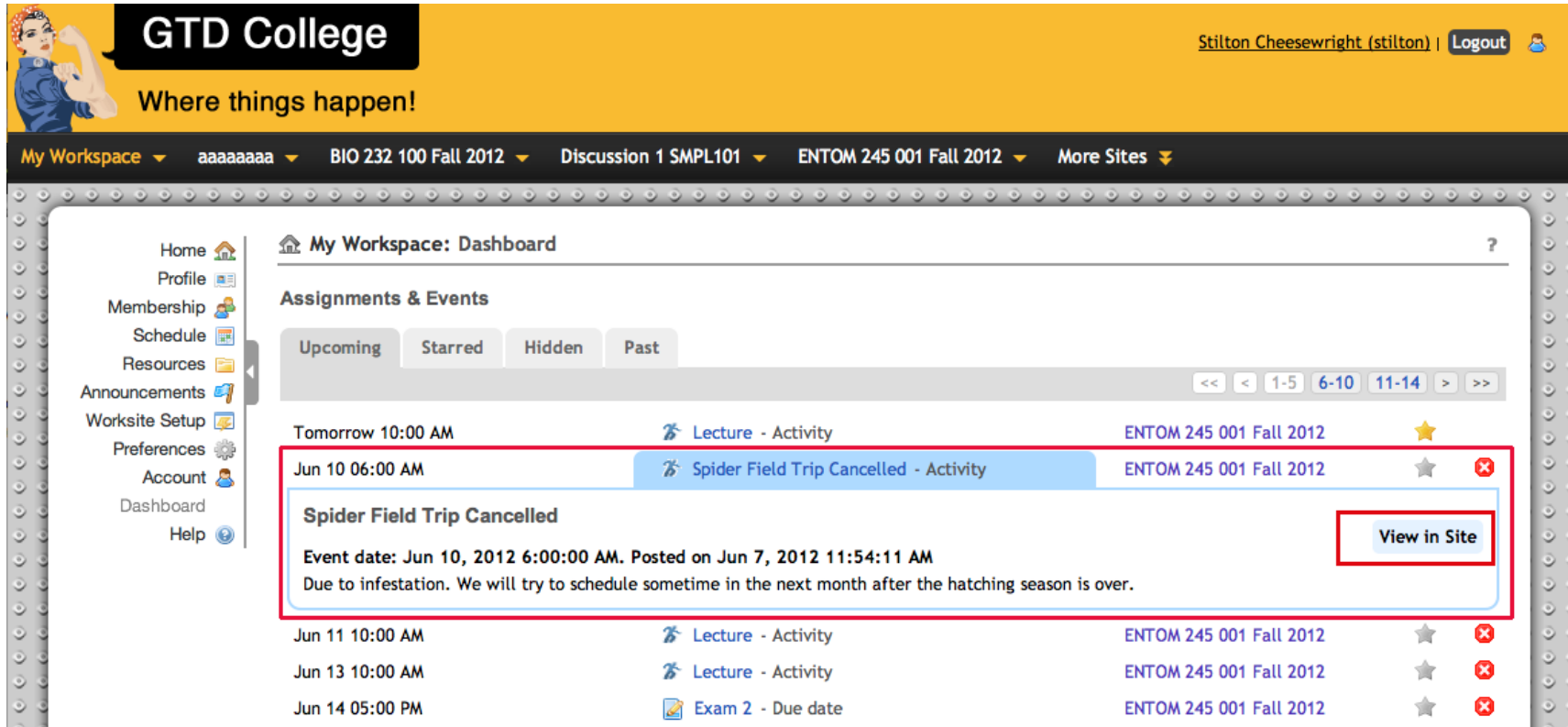
Recent Activity

| Updates | Starred | Hidden |

<< < 1-5 6 > >>

18 minutes ago		Stamp Exchange Location Change - added	Philately Society	☆	⊗
22 minutes ago		Exam 2 - revised	ENTOM 245 001 Fall 2012	☆	⊗
23 minutes ago		3 resources added to ENTOM 245 001 Fall 2012	ENTOM 245 001 Fall 2012		

	brownrecluse.gif - added		☆	⊗
	huntsmanani.gif - added		☆	⊗
	mousespiderfemale150.gif - added		☆	⊗

Technology design overview

- Challenges -- performance impacts
- Our approach -- precompute dashboard lists
- Constraints -- dashboard rendering should be lightning fast, and precomputing lists for users should not impact performance
- Our implementation -- new database tables, event observers
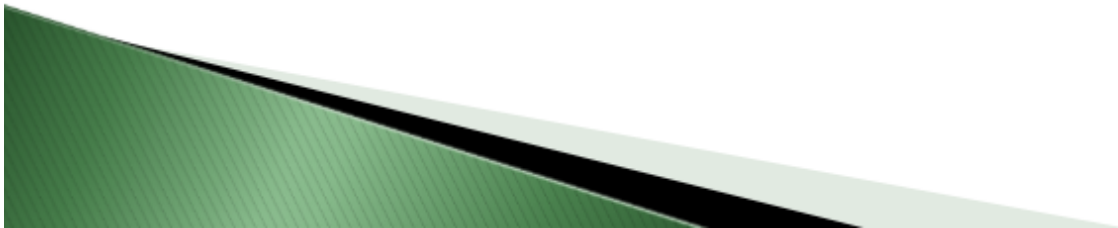
Tech design challenges -- performance

- It's hard to build a page listing items relevant to a particular user in Sakai. Here's why:
  - Items come from many different tools and services
  - No central place to get all items relevant to a user
  - Would need to check permissions on each item
- That's why synoptic tools have impacted performance (they also failed to deliver useful links)
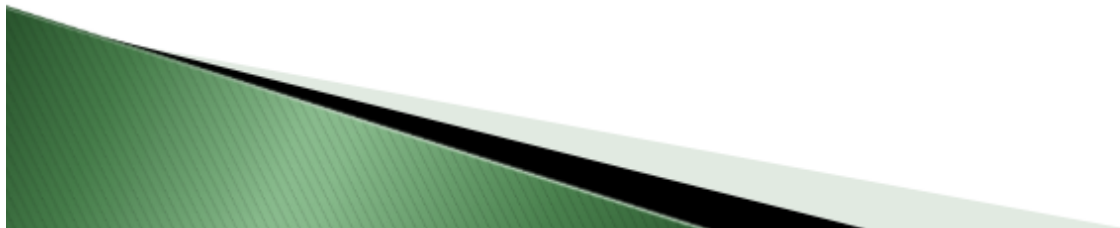
Tech design approach -- precomputing

What if we precomputed
a list of items for each user and
kept it up-to-date as things were
added, deleted and modified?

We could just keep a list in
memory for each user.

Tech design constraints

- Rendering the dashboard for each and every user should be very fast. Simple, fast database queries. No render-time permission checks
- We achieve that by precomputing the lists of items for each user, as Dr Chuck suggested (but in the database)
- Even then, permission checks must be done. We can't show things to users who lack permission to see them
- We do permission checks and/or revise the lists as items are added to sakai, as site membership or group membership changes, as permissions change, as items are deleted or modified. ...
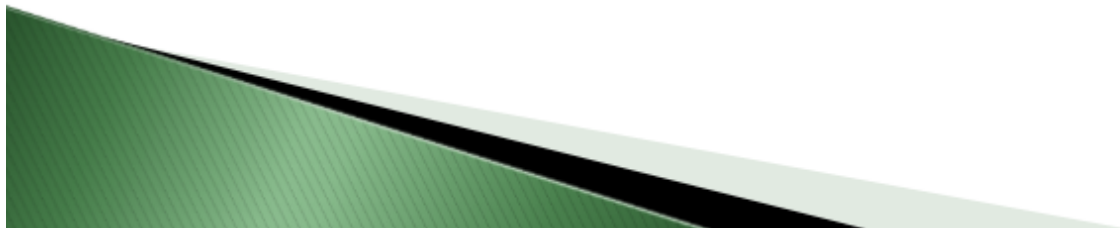
This design requires new database tables

- Needing to get two lists of items for each user very fast
- The lists must be precomputed
- We will have several new tables, including:
  - one containing entries for all items in the "Assignments & Events" section of dashboard
  - one containing entries for each item in the "Recent Activity" section.
  - one with links between people and entries in the table listing items in the "Assignments & Events" section
  - one containing links between people and entries in the table listing items in the "Recent Activity" section of dashboard

# We use event observers to track changes in Sakai entities

- Consider dashboard support for "resources"
- Every time a new resource becomes available to a person, we want to add a dashboard item reporting that fact to that person
  - We watch Sakai events and add dashboard items when we see new things created
  - We watch Sakai events and remove items from a person's dashboard when that item is deleted or no longer available to a person for whatever reason
- To avoid impacting the user who created the resource, we make a copy of the event and put it in a queue. It will be processed in a separate thread, outside any user's thread

System Load: A Dialogue

Optimists said that the dashboard would improve the overall performance of Sakai because it would reduce the number of pages we would be rendering to get users from the "home" page in MyWorkspace to the new assignment or resource they needed to see.

That *might* be true.
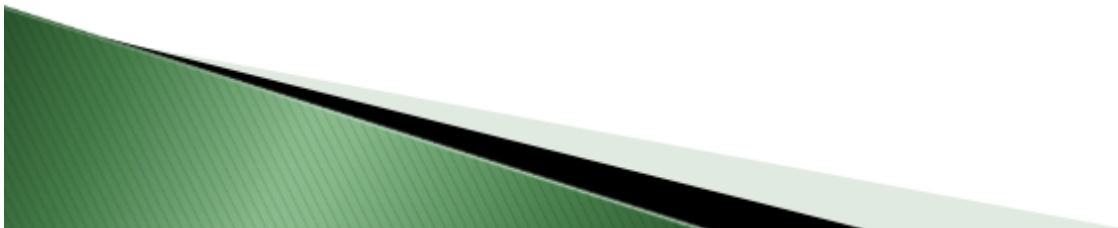
System Load: A Dialogue

Others--slightly less optimistic --pointed out we were adding extra work: event observers and new database activity to pre-compute lists of items and render the dashboard.

More efficient configuration may depend on an institution's infrastructure configuration

The answer could only be found through performance testing. Unfortunately, we never had adequate time on our load test environment to do this.

Performance Testing

Limited Concurrency Testing during Functional Test Cycles
- Not Load Tested.

Proposed Approach:
- Create Test Environment
- Standard User Navigation, no Dashboard Event Processing
- Standard User Navigation, with Event Processing
- Dashboard Navigation, with Event Processing
- Stepped Application of Virtual Traffic to Find Bottlenecks
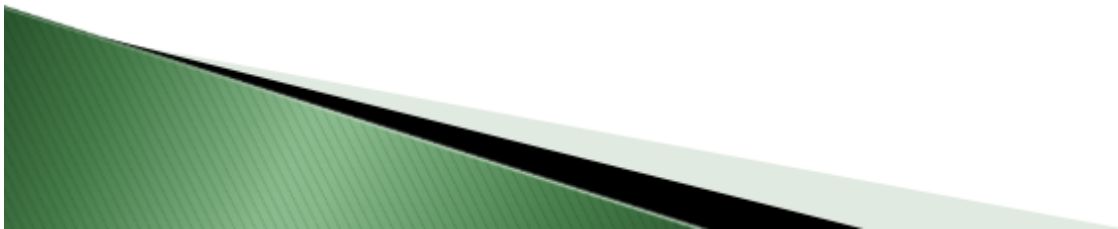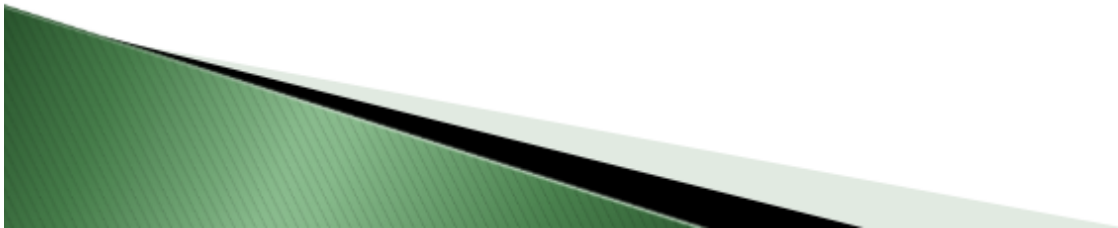
Performance Testing

Call for Assistance
- Dev team wants performance feedback

Available Resources:
- Data Files
- Load Runner Test Scripts and Scenarios
  - Creating test environment
  - Generating load

Deploying the dashboard in Sakai

- Dashboard is a contrib project:
    - https://source.sakaiproject.org/contrib/dashboard/trunk/trunk
    - Latest tag: https://source.sakaiproject.org/contrib/dashboard/tags/1.0.2_RC01/
- Dashboard uses new event types introduced in 2.9. Build with Sakai 2.9 onward.
- Nightly trunk is on nightly experimental build server:
    - http://nightly2.sakaiproject.org:8085/portal
- Conversion script is needed for new dashboard tables: dash_user, dash_context, dash_sourcetype, etc.
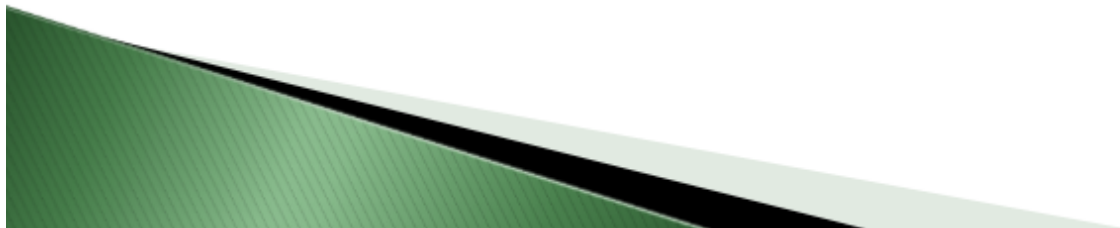
New dashboard tool

- Interested in adding dashboard to your Sakai instance?
  - o Overview and brief demo
  - o Brief technology design overview
  - o Performance testing
  - o Deploying dashboard in Sakai
- Interested in making other entities show up in dashboard?
  - o Quick guide to adding entities to dashboard
  - o Developer case study
- Summary
- Questions

Basic core entities show up in dashboard, and others can be added
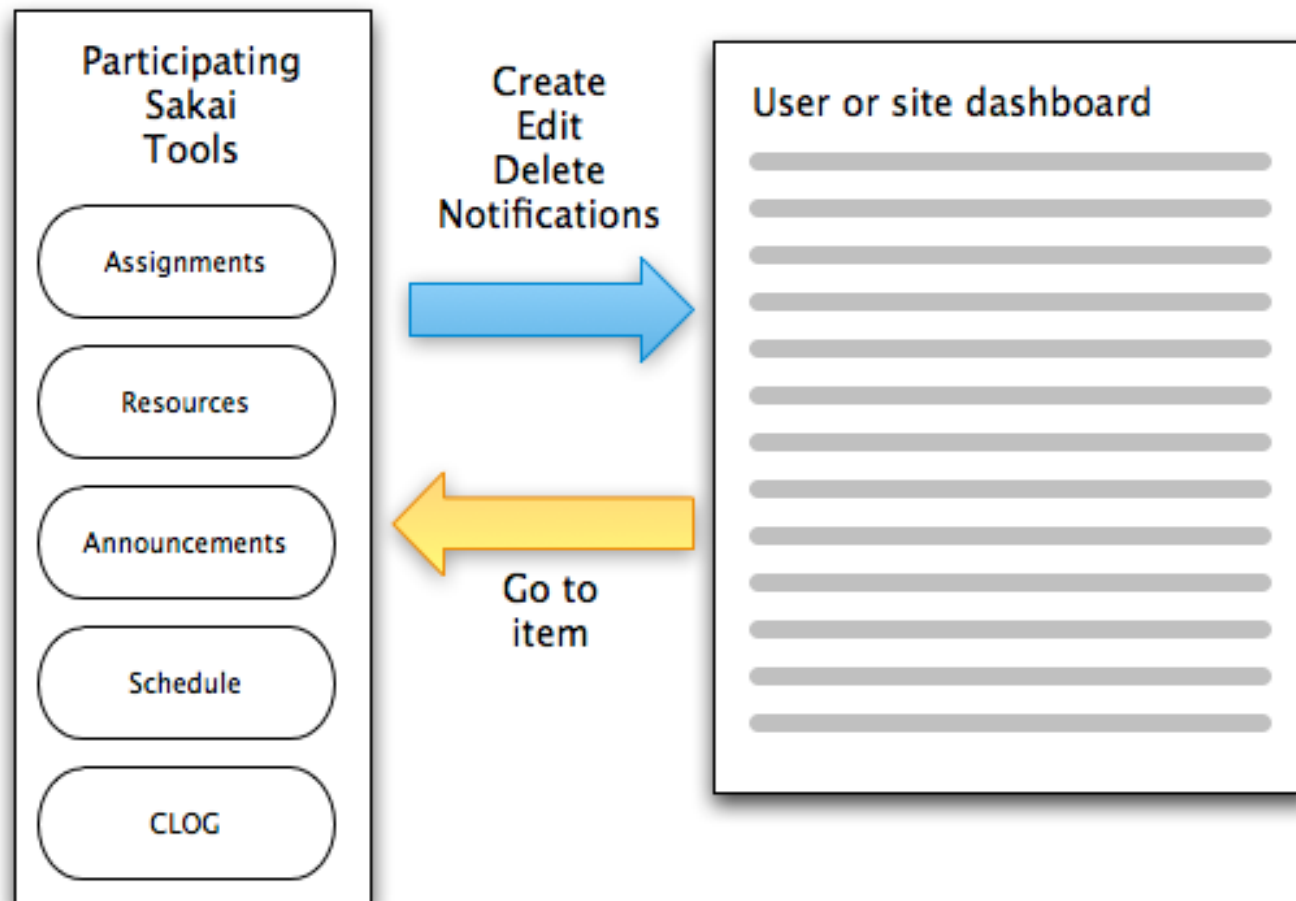
Quick guide to making entities show up in dashboard

1. Figure out which Sakai events affect what should be shown in dashboard
2. Implement an "EventProcessor" for each relevant Sakai event
3. Implement the "EntityType" interface to answer specific questions about this new type of entity
4. Register instances of your EventProcessor and EntityType classes with DashboardLogic
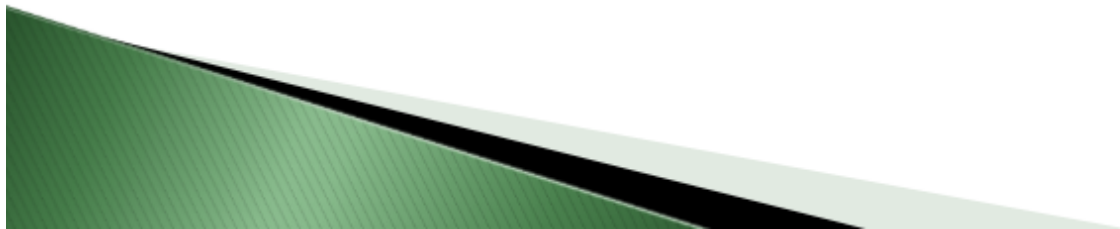
1. Figure out which Sakai events affect what should be shown in dashboard

- Events that create new entities
- Events that delete existing entities
- Events that change who has read-access to an entity
  - Changes in permissions
  - Changes in access (group restrictions)
  - Changes in availability (hidden/shown, release date, retract date, open-date of an assignment, etc)
  - Changes in site membership or group membership
- Events that modify the info displayed in dashboard (e.g. "display name" of a resource, date of a calendar event)
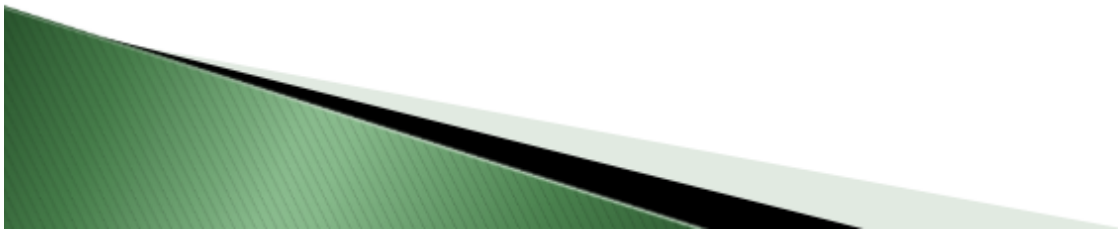
2. Implement an "EventProcessor" for each relevant Sakai event

```
public String getEventIdentifer();
```

Get the unique identifier for the events that will be handled by this processor (e.g. "content.new", "asn.delete.assignment")

```
public void processEvent(Event event);
```

Deal with one event -- adding, updating or deleting dashboard items as appropriate. The parameter is a copy of the event object (type "org.sakaiproject.event.api.Event")
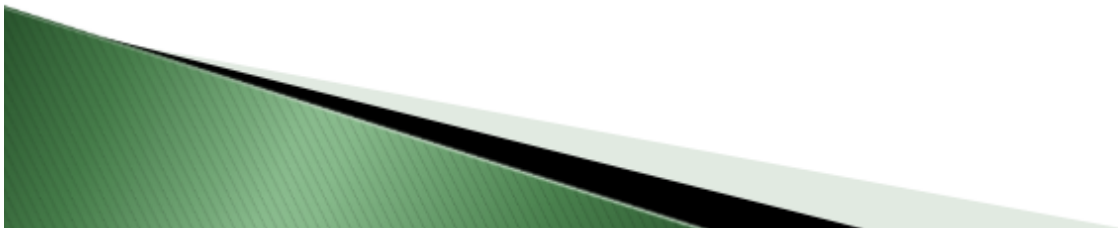
3. Implement the "EntityType" interface

```
public List<String> getUsersWithAccess(String
entityReference);
```

Get a list of users with access to an entity. This lets your code figure out which users can see an entity, rather than requiring dashboard to know how to calculate permissions, access, availability, etc., for every type of entity.

```
public boolean isUserPermitted(String sakaiUserId,
String entityReference, String contextId);
```

Does a user has permission to access an entity?
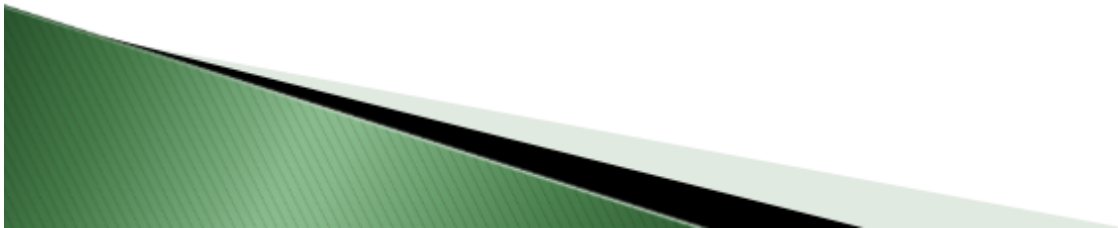
3. Implement "EntityType" (continued)

```
public Map<String, Object> getValues(String
entityReference, String localeCode);
```

Retrieve a mapping of key-value pairs related to a specific
entity for a specific locale

```
public Map<String,String> getProperties(String
entityReference, String localeCode);
```
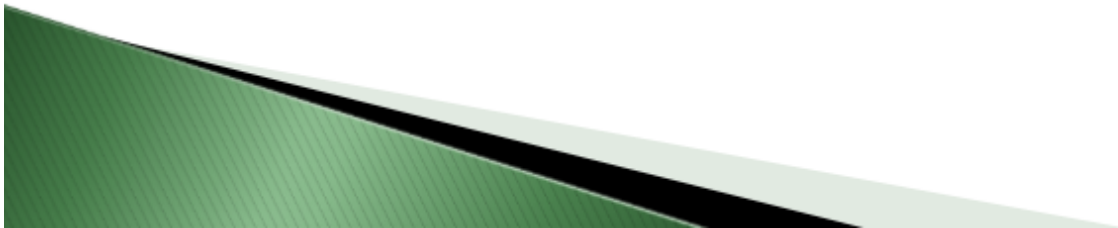
Retrieve a mapping of strings suitable for use to a user. The
strings will be localized for the locale and may be specific to the
entity identified by the entityReference parameter.

3. Implement "EntityType" (continued)

```
public List<List<String>> getOrder(String
entityReference, String localeCode);
```

Retrieve a list of lists. Each of the inner lists represents a section (or "line") of the disclosure for the entity. The strings within each of those lists represents a value (along with any labels for that value). So the list of lists of strings returned by this method should specify the order of the sections and the order of the information within each section. This may be the same for all entities of a particular type, or it may be specific to the entity and/or the user's locale.

3. Implement "EntityType" (continued)

```
public String getGroupTitle(int numberOfItems,
String contextTitle, String labelKey);
```

Retrieve a localized string suitable for use as the title of a grouped news item representing some number of individual items of this entity type within the context.

```
public String getIconUrl(String subtype);
```

Determine the correct icon to use for a particular entity in the dashboard list based on its "subtype".

4. Register instances of your EventProcessors and EntityTypes

Examples from the Schedule support class:

```
dashboardLogic.registerEntityType(new ScheduleEntityType());


dashboardLogic.registerEventProcessor(new ScheduleNewEventProcessor());
dashboardLogic.registerEventProcessor(new ScheduleRemoveEventProcessor());
dashboardLogic.registerEventProcessor(new ScheduleUpdateTimeEventProcessor());
dashboardLogic.registerEventProcessor(new ScheduleUpdateTitleEventProcessor());
dashboardLogic.registerEventProcessor(new ScheduleUpdateTypeEventProcessor());
dashboardLogic.registerEventProcessor(new ScheduleReviseEventProcessor());
    ...
```

4. Register instances of your EventProcessors and EntityTypes

Examples from the Schedule support class:

```
dashboardLogic.registerEntityType(new ScheduleEntityType());


dashboardLogic.registerEventProcessor(new ScheduleNewEventProcessor());
dashboardLogic.registerEventProcessor(new ScheduleRemoveEventProcessor());
dashboardLogic.registerEventProcessor(new ScheduleUpdateTimeEventProcessor());
dashboardLogic.registerEventProcessor(new ScheduleUpdateTitleEventProcessor());
dashboardLogic.registerEventProcessor(new ScheduleUpdateTypeEventProcessor());
dashboardLogic.registerEventProcessor(new ScheduleReviseEventProcessor());
    ...
```

4. Register instances of your EventProcessors and EntityTypes

Examples from the Schedule support class:

```
dashboardLogic.registerEntityType(new ScheduleEntityType());


dashboardLogic.registerEventProcessor(new ScheduleNewEventProcessor());
dashboardLogic.registerEventProcessor(new ScheduleRemoveEventProcessor());
dashboardLogic.registerEventProcessor(new ScheduleUpdateTimeEventProcessor());
dashboardLogic.registerEventProcessor(new ScheduleUpdateTitleEventProcessor());
dashboardLogic.registerEventProcessor(new ScheduleUpdateTypeEventProcessor());
dashboardLogic.registerEventProcessor(new ScheduleReviseEventProcessor());
    ...
```

Quick guide to making entities show up in dashboard (continued)

- Dashboard code should not depend on Sakai EntityProducer or EntityProvider implementations
- Also best that Sakai EntityProducer and EntityProvider implementations should not depend on Dashboard

- Strange example: Suppose we want to add cats to the dashboard
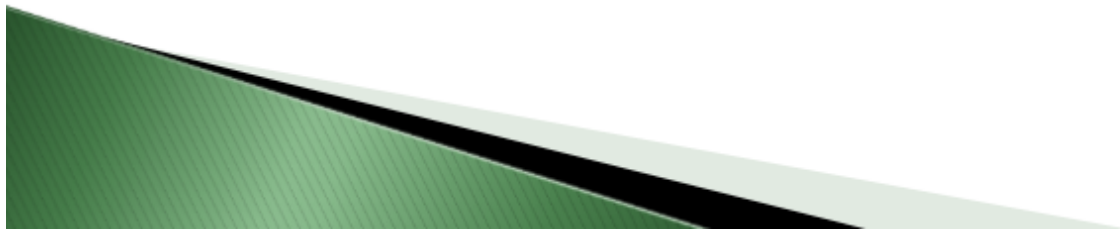
Quick guide to making entities show up in dashboard (continued)

- Dashboard code should not depend on Sakai EntityProducer or EntityProvider implementations
- Also best that Sakai EntityProducer and EntityProvider implementations should not depend on Dashboard

- Strange example: Suppose we want to add cats to the dashboard

feline-support jar

Quick guide to making entities show up in dashboard (continued)

- Dashboard code should not depend on Sakai EntityProducer or EntityProvider implementations
- Also best that Sakai EntityProducer and EntityProvider implementations should not depend on Dashboard

feline-api jar

feline-support jar
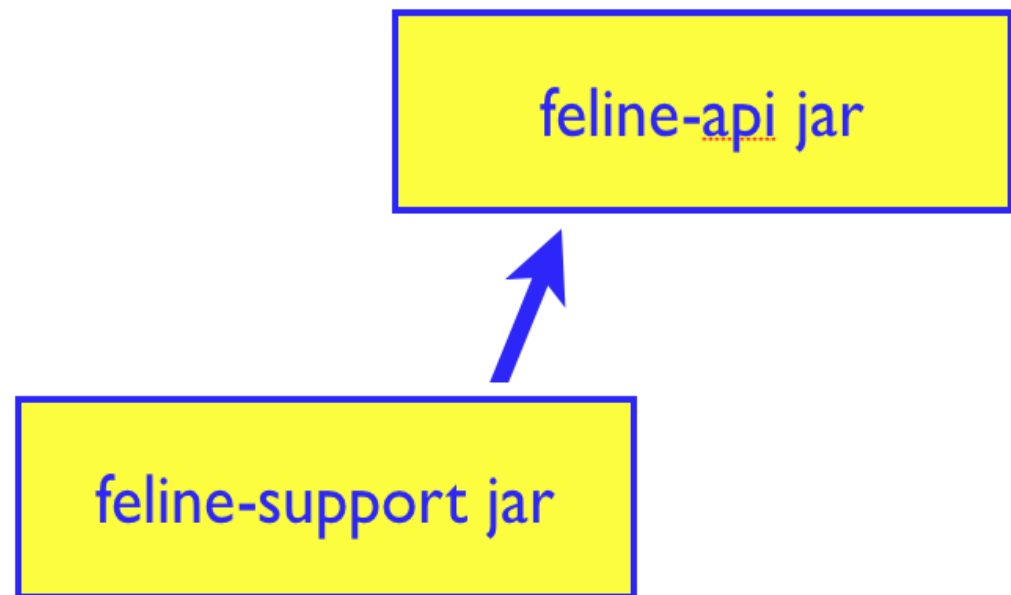
Quick guide to making entities show up in dashboard (continued)

- Dashboard code should not depend on Sakai EntityProducer or EntityProvider implementations
- Also best that Sakai EntityProducer and EntityProvider implementations should not depend on Dashboard
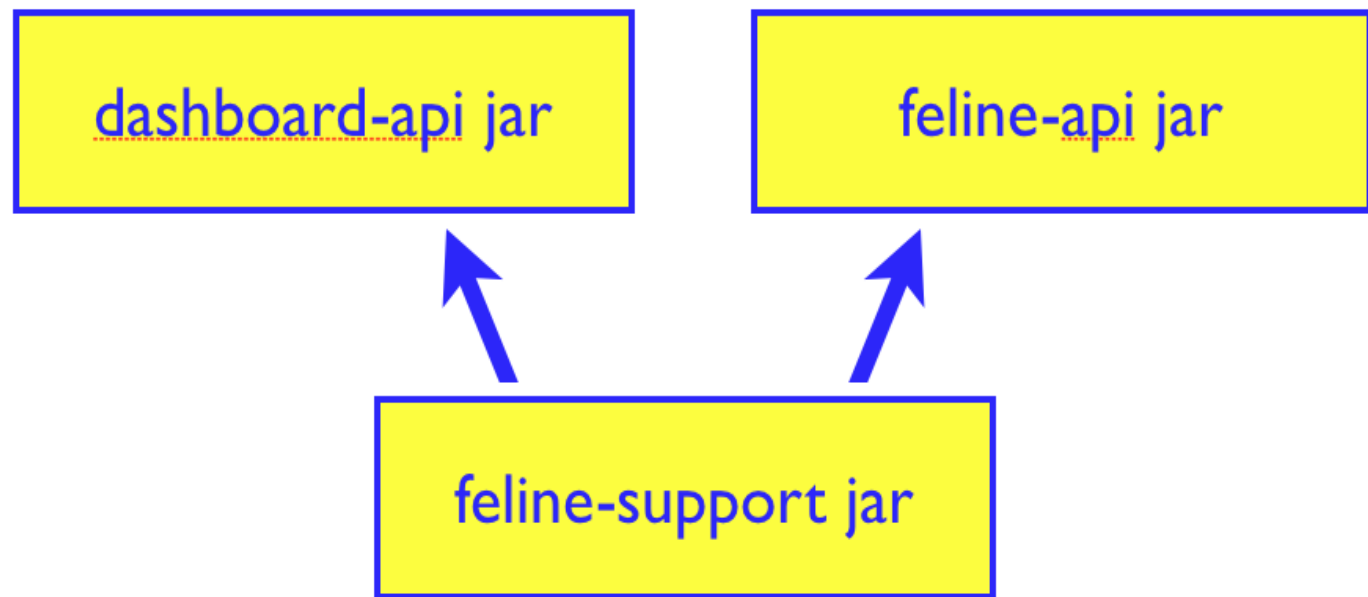
Developer case study

- Some tools have already been hooked up to the Dashboard
- We're going to have a look at one of them, CLOG ...
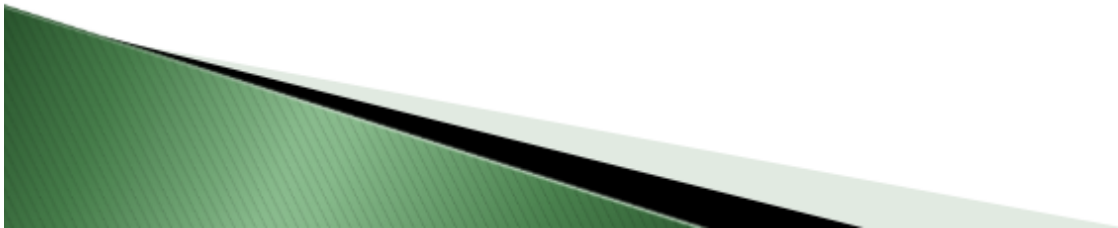
Dashboarding the CLOG

- CLOG (Collaborative Log) is a blogging tool for Sakai.
- It fires events when authors create/update/delete entries.
- It makes a great deal of sense to expose these CLOG events in the Dashboard.

- I have and here is the hard evidence to prove it …
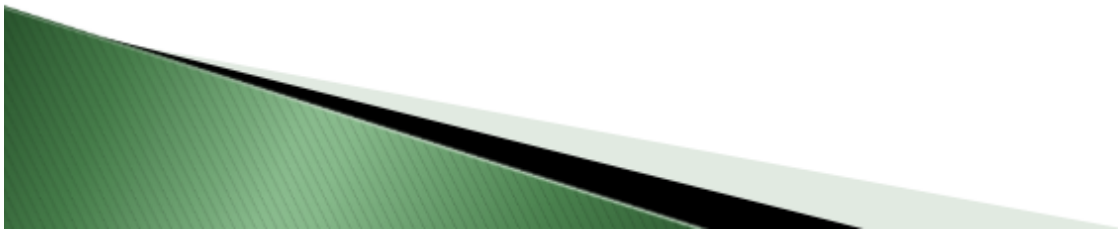
Create a CLOG post ...

Playpen: Blog

| Home | View authors | View my blog | Create new post | Permissions | Recycle Bin |

Post Editor

Title *: Free range eggs

Content:

How come free range eggs are so expensive nowadays?

body p

☑ Allow comments

◯ Only site administrators and I can see this post
◉ All members of this site can see this post

Publish   Save as Draft   Cancel

… and it appears in the Dashboard!

How it works

- The CLOG integration is implemented as an add on component which can be deployed as an assembly or built using Maven.
- There is no dependency between CLOG and Dashboard, the coupling happens through observers and events.
- Your event entities, CLOG posts in this case, need to be accessible via the EntityManager so that the dashboard code can retrieve them. You have to implement EntityProducer in your tool (your tool needs to be 'entified').
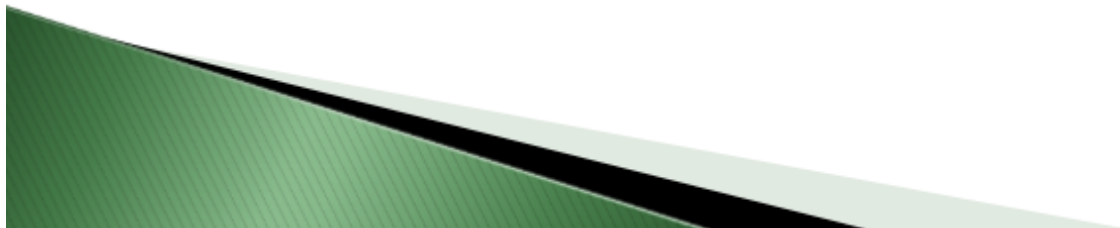
EventProcessor.getEventIdentifier

Simply return your event name from this. It has to be the same one your tool passes to the EventTrackingService when it fires its events, so Dashboard knows to look out for it.

```
public String getEventIdentifer() {
        return "clog.post.created";
    }
```
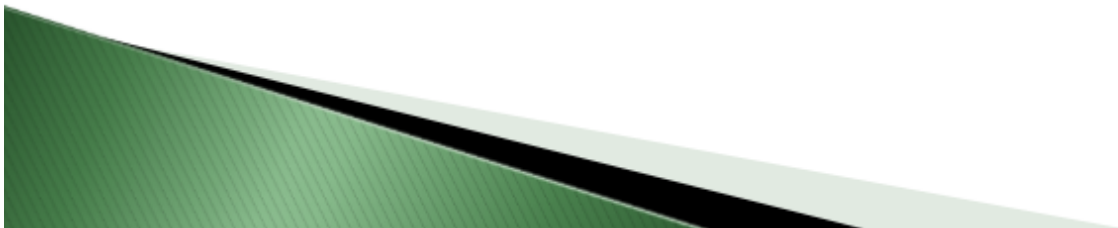
EventProcessor.processEvent

```java
public void processEvent(Event event) {
    String resource = event.getResource();
    String postId = resource.substring(resource.lastIndexOf("/") + 1);
    Post post;
    try {
        post = clogManager.getPostHeader(postId);
    } catch (Exception e) {
        e.printStackTrace();
        return;
    }
    NewsItem newsItem = dashboardLogic.createNewsItem(post.getTitle()
                            ,event.getEventTime()
                            ,ClogManager.CLOG_POST_CREATED
                            ,resource
                            ,dashboardLogic.getContext(event.getContext())
                            ,sourceType
                            ,null);
    dashboardLogic.createNewsLinks(newsItem);
```
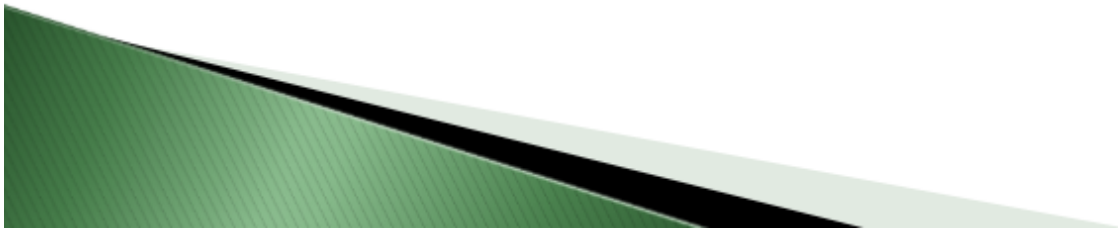
Java files

clogdashboardintegration/impl/src/java/org/
sakaiproject/clogdashboard/
- **ClogDashboardIntegration.java**
- **ClogPostCreatedEventProcessor.java**
- **ClogPostRestoredEventProcessor.java**
- **ClogPostWithdrawnEventProcessor.java**
- **ClogPostRecycledEventProcessor.java**
- **ClogPostEntityType.java**
- **ClogDashboardEntityType.java**
- **ClogDashboardEventProcessor.java**

Other files

A source of localized strings:

```
clogdashboardintegration/impl/src/bundle/
dashboard.properties
```

Assemble, deploy and run:

```
clogdashboardintegration/assembly/src/main/assembly/
deploy.xml
clogdashboardintegration/pack/src/webapp/WEB-INF/
components.xml
```

```
https://source.sakaiproject.org/contrib/
lancaster.ac.uk/clogdashboardintegration/trunk
```
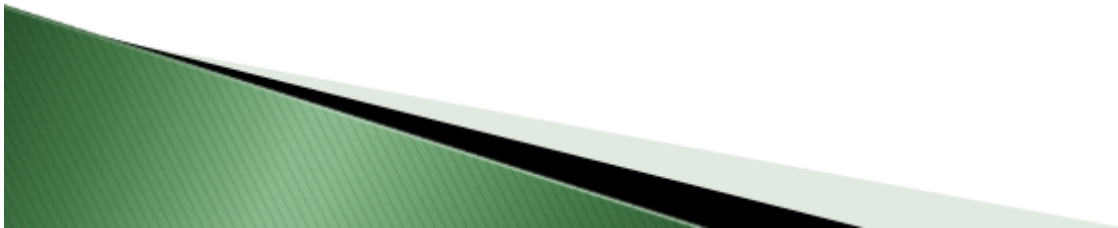
Summary

- If you are interested in adding dashboard to your sakai instance ...
  - o Quick access to the most current stuff in Sakai CLE
  - o Builds and deploys with Sakai 2.9 and later
  - o Can be backported to Sakai 2.7 or 2.8 with patches
  - o Keeps a list of items for each user
  - o Performance is TBD
- If you are interested in making another type of entity show up in dashboard
  - o Need to implement several EventProcessor and one EntityType class
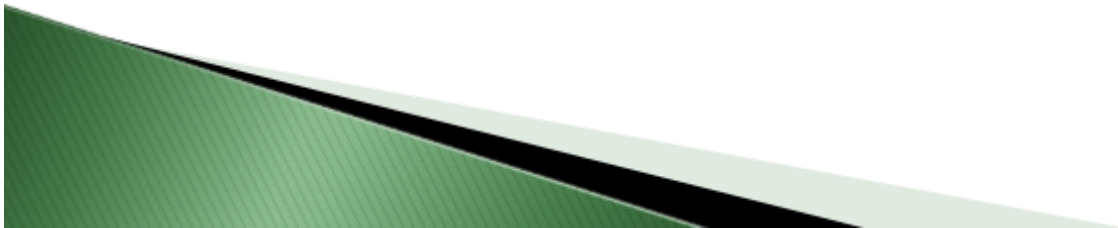
See also:

Leveraging Student Insights in Designing the Sakai Dashboard

Thursday, 9 a.m. to 10 a.m. -- Tower 1402 (14th floor)

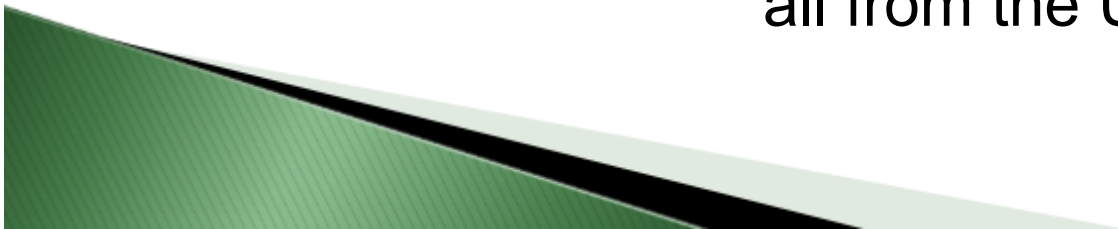This presentation describes the initial design goals for the dashboard as developed by graduate students at the University of Michigan School of Information and the process of refining the design to meet user expectations. It also describes features of the current implementation.

Presenters: John Johnston, Catlin Holman,
               Shwetangi Savant and Tiffany Chow,
                  all from the University of Michigan

More information

Proposal: **https://confluence.sakaiproject.org/x/zIWCB**
Report issues: **https://jira.sakaiproject.org/browse/DASH**
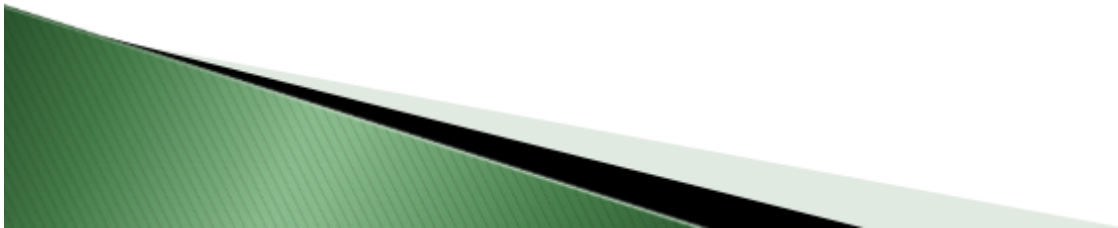
Jim Eng (jimeng@umich.edu)
Zhen Qian (zquian@umich.edu)
Gonzalo Silverio (gsilver@umich.edu)
Chris Kretler (ckretler@umich.edu)

CLOG:
Adrian Fish (a.fish@lancaster.ac.uk)

Questions?