Sakai User Id and EID

June 13, 2006

Please direct questions or comments about this document to:
Glenn R. Golden, ggolden@umich.edu

This document describes User ID and the EID separation in Sakai 2.2

For the last few releases, we've been working to introduce the concept of "internal" and "external" identifiers in Sakai in the User records. We started with a single identifier that had to play both of these roles, and have been tweaking the code to separate these out. For 2.2, we have completed this separation.

Internal and External ID

An "internal" ID is something that is unique within Sakai; something that never changes for the life of the User record; something that we can use to store references to User objects in other parts of Sakai. This is what the User.getId() returns. It is always used for inner-Sakai references, and should *never* be used when working with anyone or anything outside of Sakai. For instance, it is NOT the thing that we accept as a user ID when the user logs in; end-users have no clue what their internal Sakai identifier is for their User account. It should not be shown on user interfaces to disambiguate a display of an end user's name. It must not be used when talking to the enterprise systems that back our UserDirectory and Group providers. Id is used for things like storing "created by", user preferences, and group membership, and any other inner-Sakai user references.

An "external" ID is something that is unique within Sakai; something that has meaning to the end-users of Sakai; something that has meaning to the enterprise systems that back our UserDirectory and Group providers; and something that we might expect to see changed within the lifetime of our User objects in Sakai. This is what the User.getEid() returns. An external ID is also an "enterprise" ID. This is what the end-users know as their login ids. This is a value that we can use to identify our users with the enterprise systems that back our providers (although, there may be other enterprise-assigned identifiers for our end-users). Since an EID can change, it is never used store a reference to a User record in Sakai.

While EID is more appropriate for displaying in a user interface (along with the user name, to disambiguate the user) than ID, we provide a getDisplayId() method in the User object specifically for this purpose.

Changes

Basically, code that is using User.getId() inappropriately needs to be changed to use User.getEid() or User.getDisplayId(). The UserDirectoryService also has two new methods to map ids, in case you don't need the entire User object:

```
String getUserEid(String id) throws UserNotDefinedException
String getUserId(String eid) throws UserNotDefinedException
```

So, anytime you are displaying something to the end user, odds are very good that you need an EID, not an ID, and should use getDisplayId().

But don't go overboard and start using EID for ID purposes. If the user interface encodes a user ID, for example, for a selection of users from a list, the ID is appropriate, and required to find the user object.

The provider code, for both the UserDirectoryProviders and GroupProviders, needs special attention. These work exclusively with the EID, except for very special cases. So the "ID" sent to a provider from Sakai for authentication or getUser() or to find that user's group memberships is an EID, and any “ID” set back from the provider is an EID.

There are some "decorator" classes that wrap over User or some other data structure that has a user ID - and are used for Velocity or JSF interfaces. It might be valuable to add getEid() and getDisplayId() methods to these decorators to surface the EID and the display ID as well as the ID for UI needs. Related to this are some User-like implementations (Agent, Member, Participant) that have a userId - and will likely need to surface an EID as well.

Not changed

A user's "my workspace" site is still using the user ID, not EID. Using the EID here would be a mistake, as this is a user reference that we don't want to have to worry about changing later when the user's EID changes. But the portal URL to the site is based on the site's ID, which is less than ideal. To resolve this problem, the portal will be trained to use user EID as an alternate URL to the user's “MyWorkspace”. This is sort of like using a site's alias as well as the site ID in a portal URL to the site (which we can train the portal to do as well).

A user's "home" content area in ContentHosting is still /user/<user id>, again because it is a User object reference and should not get changed. Most of the user interface already hides this from the end user - but some areas such as the content URLs via Access and Web and WebDav could be further trained (if needed) to use the user's EID as an alias to this content area.

User references in session and event records are still using user ID. Scripts that query these need to have a way to get at users' actual ids - but since we have a nice table in the database to map ID and EID, this should be easy.

The "admin" and "postmaster" users are still called that - ID and EID. We could change this, but it's not worth it. To be less fragile, don't use the string "admin" in your code (mostly the test code has this), but use the values defined by the `UserDirectoryService` API:

```
ADMIN_ID
ADMIN_EID
```

That way, if we do every change the ID or EID of the admin, things will continue to work.

Implementation Details

We are not keeping a full User record for any user we ever see, as once we thought we might. This is misleading, because for external users (i.e. users defined by the provider), we really don't know anything about them inside Sakai except for their ID (which Sakai allocates) and the EID. So instead, we introduce a mapping table that maps ID and EID. This is the `SAKAI_USER_ID_MAP` table that maps the ID and EID fields. The main `SAKAI_USER` table is unchanged. When we have a record defined inside Sakai, there is a `SAKAI_USER` record and an entry in the `SAKAI_USER_ID_MAP` table. When we have a user who is defined solely by the provider, there's only an entry in the map table, which we make as soon as we first meet this user, by seeing the user's EID.

Provider Changes

Providers must be changed to work with the User object's EID field, and ignore the ID field.

We are also enhancing the UserDirectoryProvider, to allow it to optionally get involved with the formation of the user displayId(). Usually, the displayId() is the EID value. There are some special cases where the EID is not the best value to display in user interfaces to disambiguate the user's name display. In these cases, the UserDirectoryProvider will implement the new "display advisor" interface, and be called from the UserDirectoryService impl. whenever User.getDisplayId() is called.

The provider can use the User object passed in to get the EID, and lookup information based on that, or it can have already stashed away stuff in the User object's properties (either when the provider synthesized the User object, or when a bulk-load process created the internally kept User object), information it will use to form the displayId.

Internal Sakai User objects, and systems without providers or with providers that are not DisplayAdvisors will simply use the EID as the displayId.