Sakai Security Function Registration

October 24, 2005

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

Sakai 2.1 changes the way that applications register their security functions with the Sakai framework. It improves the 2.0 method, which was to modify a single file to contain all the security functions for all applications, buy distributing the registration out to the applications. This restores functionality that we had in 1.5, and extends it to make it even easier to register your security functions

Security Functions

Sakai applications that use the Sakai framework security service need to register the security function strings, those names given to the fine-grained security points of the application that are checked when users attempt access and attempt to perform application functions.

The security function registrations are used by various tools and tool helpers in Sakai to present to the end user; the admin authorization group editor and the permissions helper are the primary users. These let Sakai users customize the security definitions of their system by assigning different sets of security functions to various users and roles.

Security functions are currently just strings. To use the permissions helper, you should have a common prefix for your strings that identify the function as part of your application (for example, all security functions for the Announcement application use the "annc." Prefix).

Registration

Security function registration is required of all Sakai applications that participate in the security system. Registration is transient to each run of Sakai; it is done at server startup, stored in memory, and lost when the server shuts down. There is no persistence (such as a database table) behind registration. Like tool registration, it is something that the application must make sure is done each time Sakai starts up.

There are two ways to register; in java, making calls to the `FunctionManager`, or in .xml along with you tool registration, using the Sakai web context listener.

Registration / Java

The best way for your application to register security functions is to do it in java code in one of your API components, that of your service or manager. This is usually a singleton that starts up at server startup and contains your applications storage and business logic. This is loaded by our component manager, and can be designated as having an `init()` method to call on startup.

The best place to define your security functions is in your service or manager API. The really really best place to enforce security is in your manager API's implementation components. It's in there, in the `init()` method, that you can register your functions.

To register security functions from your `init()` method of your service or manager component, call the Sakai kernel's `FunctionManager`. You can inject this or call the cover. The examples below use the cover:

```
import org.sakaiproject.api.kernel.function.cover.FunctionManager;
```

In your manager's `init()` method:

```
/**
* Final initialization, once all dependencies are set.
*/
public void init()
{
```

simply call the `registerFunction()` method, like this (from content hosting):

```
// register functions
FunctionManager.registerFunction(EVENT_RESOURCE_ADD);
FunctionManager.registerFunction(EVENT_RESOURCE_READ);
FunctionManager.registerFunction(EVENT_RESOURCE_WRITE);
FunctionManager.registerFunction(EVENT_RESOURCE_REMOVE);
FunctionManager.registerFunction("dropbox.own");
...
```

Register each of your functions in this way.

If you have defined constants for your security function in your API, as ContentHosting has done and is show in the above code (all except for the dropbox permission), then you can use them here in the registration.

The dependency for Maven, to include in your `project.xml`, for code using the function manager, is:

```
<dependency>
     <groupId>sakaiproject</groupId>
     <artifactId>sakai-function</artifactId>
```

```
            <version>${sakai.version}</version>
        </dependency>
```

Registration / XML

The other, less preferable way, to register an application's functions is to include it with one of your tool registration XML files in one of your application's webapps. The reason this is not the best way is because security function registration is an application feature, not a tool feature. If you can do it in you manager or service component, that is a better expression of separation of concerns for your application.

In some cases, registering security functions in a tool registration file can be useful. This would be good when integrating some external application, or when you have built an application that does not follow the Sakai best-practice of separating persistence and business logic into a separate singleton manager or service. If there is no convenient singleton from which to call `registerFunction()`, you can (once again) register the function in the tool registration .xml file. This is a return of the feature we had in Sakai 1.5.

Alongside of the `<tool>` elements in a tool registration.xml file, simply add some `<function>` ones:

```
        ...
        </tool>

        <function name="myapp.function" />
        <function name="myapp.other.function" />
        ...
```

(this is from a bogus example, since the Sakai code will not be using this feature).