Hooks POC (Proof of concept)

Hooks Proof of Concept

Hooks Introduction
Getting started with hooks
Hooks Example - Assign a Unix id to each new group

This document is about a proof of concept on hooks for a group change and a membership change.

The current progress is a working unit test for each, and a Grouper UI example for a group change (member change is in progress).

Grouper UI group example

I added a hook to the Grouper UI which is a veto hook which will not allow a group to be created which is not in the "penn" folder (which is a subfolder of root).

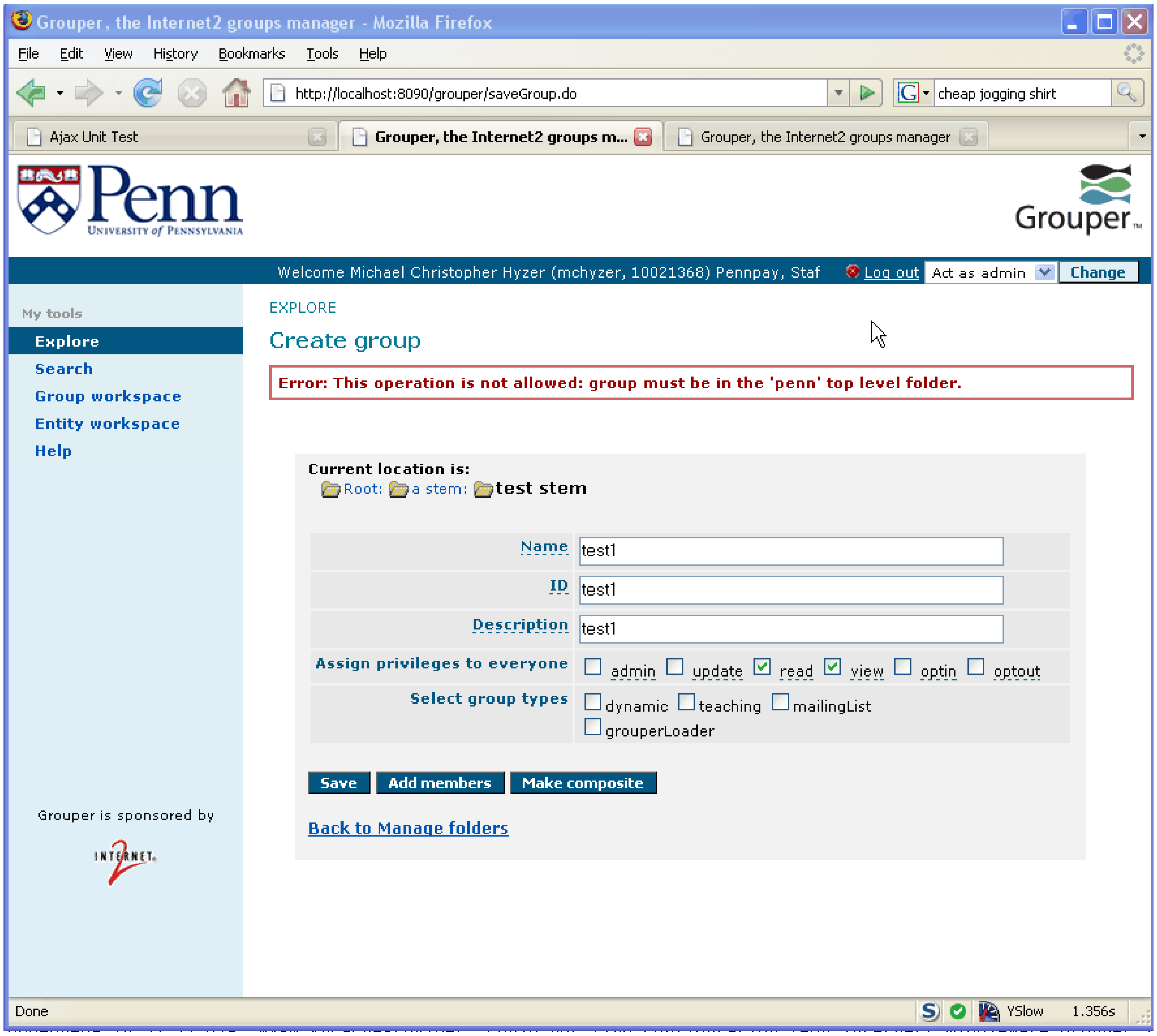
To implement this, it requires a group hook class:

```
/*
 * @author mchyzer
 * $Id: GroupHooksImplExample.java,v 1.1.2.1 2008/06/11 06:19:38 mchyzer Exp $
 */
package edu.internet2.middleware.grouper.ui.hooks;

import org.apache.commons.lang.StringUtils;

import edu.internet2.middleware.grouper.GrouperConfig;
import edu.internet2.middleware.grouper.hooks.GroupHooks;
import edu.internet2.middleware.grouper.hooks.beans.HooksGroupPreInsertBean;
import edu.internet2.middleware.grouper.hooks.HookVeto;
import edu.internet2.middleware.grouper.internal.dao.GroupDAO;


/**
 * test implementation of group hooks for test
 */
public class GroupHooksImplExample extends GroupHooks {

  /**
   * @see edu.internet2.middleware.grouper.hooks.GroupHooks#groupPreInsert(edu.internet2.middleware.grouper.
hooks.beans.HooksGroupPreInsertBean)
   */
  @Override
  public void groupPreInsert(HooksGroupPreInsertBean preInsertBean) {

    GroupDAO groupDAO = preInsertBean.getGroupDao();
    String name = StringUtils.defaultString((String)groupDAO.getAttributes().get(GrouperConfig.ATTR_NAME));
    if (!name.startsWith("penn:")) {
      throw new HookVeto("hook.veto.group.name.prefix", "group must be in the 'penn' top level folder");
    }
  }

}
```

And it requires a configuration in the grouper.properties:

```
#implement edu.internet2.middleware.grouper.hooks.GroupHooks
hooks.group.class=edu.internet2.middleware.grouper.ui.hooks.GroupHooksImplExample
```

The result when trying to add a group not in that folder, is:


Grouper, the Internet2 groups manager - Mozilla Firefox

http://localhost:8090/grouper/saveGroup.do

Welcome Michael Christopher Hyzer (mchyzer, 10021368) Pennpay, Staf Log out Act as admin Change

EXPLORE

Create group

Error: This operation is not allowed: group must be in the 'penn' top level folder.

Current location is:
Root: a stem: test stem

Name	test1
ID	test1
Description	test1
Assign privileges to everyone	☐ admin ☐ update ☑ read ☑ view ☐ optin ☐ optout
Select group types	☐ dynamic ☐ teaching ☐ mailingList ☐ grouperLoader

Grouper UI membership example

I added a hook to the Grouper UI (unfinished... was getting too fancy) for a membership veto that if the group type is grouperLoader, do not allow group memberships. However, if the user is a wheel group user, then allow it but put a warning on screen

Here is the starting point for the Java:

```
/*
 * @author mchyzer
 * $Id: MembershipHooksImplExample.java,v 1.1.2.1 2008/06/11 06:19:38 mchyzer Exp $
 */
package edu.internet2.middleware.grouper.ui.hooks;

import edu.internet2.middleware.grouper.Group;
import edu.internet2.middleware.grouper.GroupType;
import edu.internet2.middleware.grouper.GroupTypeFinder;
import edu.internet2.middleware.grouper.SchemaException;
import edu.internet2.middleware.grouper.hooks.MembershipHooks;
import edu.internet2.middleware.grouper.hooks.beans.GrouperBuiltinContextType;
import edu.internet2.middleware.grouper.hooks.beans.GrouperContextType;
import edu.internet2.middleware.grouper.hooks.beans.HooksContext;
import edu.internet2.middleware.grouper.hooks.beans.HooksMembershipPreAddMemberBean;
import edu.internet2.middleware.grouper.hooks.HookVeto;


/**
 * test implementation of group hooks for test
 */
public class MembershipHooksImplExample extends MembershipHooks {

  /**
   * @see edu.internet2.middleware.grouper.hooks.MembershipHooks#membershipPreAddMember(edu.internet2.
middleware.grouper.hooks.beans.HooksMembershipPreAddMemberBean)
   */
  @Override
  public void membershipPreAddMember(HooksMembershipPreAddMemberBean preAddMemberBean) {
    HooksContext hooksContext = preAddMemberBean.getHooksContext();
    GrouperContextType grouperContextType = hooksContext.getGrouperContextType();

    //only care about this if not grouper loader
    if (!grouperContextType.equals(GrouperBuiltinContextType.GROUPER_LOADER)) {

      //if the act as user is is in the wheel group, then just admonish
      if (hooksContext.isSubjectActAsInGroup("penn:etc:sysAdminGroup")) {

        //add warning to system

      } else {

        Group group = preAddMemberBean.getGroup();
        GroupType groupType = null;
        try {
          groupType = GroupTypeFinder.find("grouperLoader");
        } catch (SchemaException se) {
          throw new RuntimeException(se);
        }
        if (group.hasType(groupType)) {
          throw new HookVeto("hook.veto.loader.membership", "the membership of this group is automatically
managed and does not permit manual changes");
        }

      }



    }
  }

}
```

Here is the grouper.properties config:

```
#implement edu.internet2.middleware.grouper.hooks.MembershipHooks
hooks.membership.class=edu.internet2.middleware.grouper.ui.hooks.MembershipHooksImplExample
```

No screen shot yet

Issues

Everything went pretty smoothly, but...

1. To implement a hook, it will require some understanding about Grouper. We should post a bunch of examples. The problem is we have business objects, data transfer objects, and data access objects. Sometimes logic goes through any of these paths. So I added hooks to the data access layer (layer on top of hibernate) so that all operations can be hooked. Sometimes there will be a reference to the business object (e.g. Group), but it doesnt really make sense in some cases (like on an insert, there is no group uuid yet, so the Group object is not created yet, and even if it were, most methods would be invalid).
2. On memberships, the DAO hook will probably not be the useful one. Information like group name or subject id is not even available, it would have to be queried in a lot of cases. In some cases it is known, but it is weird to have it there sometimes and not others
3. I added a high level membership hook (addMember) which will give the information about what the group name is, subjectId, etc. This is most likely the one that can be used for veto operations. The low level one would most likely be used for auditing. The weird thing about the addMember hook is that some of the objects are business objects, and some are DTOs (see the BaseMemberOf object which encapsulates one member addition). I think people will figure it out... but for the record, I'm not completely bought in to the necessity of having so many data layers. 🙂

Implementation of grouper code details

This is implemented in a 1.4 hooks branch in cvs.

- Here is the start of a groups hook class (you override this to add a hook)
- Here is the start of a membership hook class (notice high and low level hook methods)
- All DB calls have been refactored to go through grouper's HibernateSession API. The transaction implementation in 1.3 brought us more than halfway there, and this seals the deal. The HibernateSession API will allow events to be registered on each hibernate action in a safe way so that the transaction can still be used (differentiates from the built in hibernate interceptors and I think events though events are not documented well)

e.g. the delete and load methods are just wrappers around the ones in hibernate of the same name. ByObject just separates up the namespace a bit (there are also ByHql, and ByCriteria)

```
HibernateSession.callbackHibernateSession(GrouperTransactionType.READ_WRITE_OR_USE_EXISTING,
        new HibernateHandler() {

          public Object callback(HibernateSession hibernateSession) {
            ByObject byObject = hibernateSession.byObject();
            byObject.delete( byObject.load( Hib3GrouperSessionDAO.class, _s.getId() ) );
            return null;
          }

    });
```

* Each hook will have its own bean (these arent finished by any means, but look at add member for decent example). This is to encapsulate the params passed to the hook (and to facilitate an easy way to get data back from a hook if needbe). This way if any params change, existing implementors will be less likely to have to change their code

- Notice the reference in the hook beans to the hook context bean. This holds information about the current user, and gives utility methods (e.g. is the current user in a certain group). This bean also holds attributes which can be set by the current application. e.g. the UI and WS might give a reference to the HttpServletRequest. These attributes can be set as threadsafe or not, which means when the asynchronous callback is called, all the data will be handled correctly (e.g. in a new thread there is no HttpServletRequest object since that is a weak reference and the request will be over before thread is)
- For vetos, there is one exception for various types, and it indicates which type of veto it is (set automatically if not manually). This is going to require a little bit of work on the implementors (e.g. WS, UI, etc) to handle the vetos gracefully. In the UI it was a matter of adding this code (the three lines starting with catch HookVeto). Note this will have to be done in all places where the API is used if it cant be done centrally (hopefully it can be added to filter or something...). 🙂

```
try{
  group = parent.addChildGroup(extension,displayExtension );

} catch(HookVeto hookVeto) {

  //this action was vetoed, put explanation on screen, and go back
  Message.addVetoMessageToScreen(request, hookVeto);
  return mapping.findForward(FORWARD_CreateAgain);

} catch(GroupAddException e) {
  String name = parent.getName() + GrouperHelper.HIER_DELIM + extension;
  request.setAttribute("message", new Message(
    "groups.message.error.add-problem",new String[] {e.getMessage()}, true));
  return mapping.findForward(FORWARD_CreateAgain);
}
```

* There was an issue of setters affecting the API, and I think we can fix that. e.g. for group I added a save() method which needs to now be called after setting the description, name, etc. However, there are methods which arent affected (e.g. Grouper.addMember() does not require a save).   Only javabean setters.

- Note that in the UI you can specify a message in the nav.properties for each hook veto, or just use the standard one as a default to reduce the number of steps required to get working...
- Its pretty easy/lightweight to add a hook invocation to the grouper code (I will be adding these)... e.g. here is the one for the addMember:

```
//see if there is a hook class
MembershipHooks membershipHooks = (MembershipHooks)GrouperHookType.MEMBERSHIP.hooksInstance();

if (membershipHooks != null) {
  HooksMembershipPreAddMemberBean hooksMembershipPreUpdateHighLevelBean =
    new HooksMembershipPreAddMemberBean(new HooksContext(), mof);

  membershipHooks.membershipPreAddMember(hooksMembershipPreUpdateHighLevelBean);
}
```

* For the low level hooks, the implementation is similar but even easier (since it is central)... each DAO implements this interface, so those methods just need to be implemented. e.g. in Hib3GroupDAO

```
/**
   * @see edu.internet2.middleware.grouper.internal.dao.hib3.Hib3DAO#onPreSave(edu.internet2.middleware.grouper.
hibernate.HibernateSession)
   */
  @Override
  public void onPreSave(HibernateSession hibernateSession) {
    super.onPreSave(hibernateSession);

    //see if there is a hook class
    GroupHooks groupHooks = (GroupHooks)GrouperHookType.GROUP.hooksInstance();

    if (groupHooks != null) {
      HooksGroupPreInsertBean hooksGroupPreInsertBean = new HooksGroupPreInsertBean(new HooksContext(), this);
      groupHooks.groupPreInsert(hooksGroupPreInsertBean);
    }

  }
```

This will kick in wherever a group is saved

- Group / member hooks are unit tested, and all existing unit tests still pass
- sdf