

# Grouper Training Environment - text to copy and paste - 301.4 - GSH



If you mistype one or more lines in GSH, type :c to clear all the lines since the last executed command.

To exit GSH, type :q

## Reference Links

- Grouper wiki command reference for GSH
  - <https://spaces.internet2.edu/pages/viewpage.action?pageId=14517859>
- Groovy syntax: <https://groovy-lang.org/documentation.html>
  - For a quick language intro, see the [Style guide](#) page
  - Also useful is The [Groovy Development Kit](#) page
- Grouper API Javadoc: <https://software.internet2.edu/grouper/doc/>
  - 2.6.x (active release branch) -> Grouper API -> Project Reports -> Javadoc
- 3rd Party site Baeldung - Groovy blog posts: <https://www.baeldung.com/tag/groovy/>
- 3rd Party site MrHaki - Groovy blog posts: <https://blog.mrhaki.com/>

## Exercise 301.4.1 - Getting started

Start up your Grouper docker stack. You will want the UI running (log in as banderson) so you can see visually how the shell commands impact objects in Grouper.

With a container started and running, GSH can initiated by running:

```
gte-gsh
```

On start, you'll see some helpful information about your grouper environment that will look something like this:

```
Detected Grouper directory structure 'webapp' (valid is api, apiMvn, webapp)
Using GROUPER_HOME:           /opt/grouper/grouperWebapp/WEB-INF
Using GROUPER_CONF:           /opt/grouper/grouperWebapp/WEB-INF/classes
Using JAVA:                    /usr/lib/jvm/java-1.8.0-amazon-corretto/bin/java
Using CLASSPATH:               /opt/grouper/grouperWebapp/WEB-INF/classes:/opt/grouper/grouperWebapp/WEB-INF/lib
/*:/opt/tomee/lib/servlet-api.jar
using MEMORY:                  64m-750m
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class is `com.mysql.cj.jdbc.Driver'.
The driver is automatically registered via the SPI and manual loading of the driver class is generally
unnecessary.
Grouper starting up: version: 2.6.0, build date: 2021/09/17 09:14:52 +0000, env: <no label configured>
grouper.properties read from: /opt/grouper/grouperWebapp/WEB-INF/classes/grouper.properties
Grouper current directory is: /opt/grouper/grouperWebapp/WEB-INF
log4j.properties read from:   /opt/grouper/grouperWebapp/WEB-INF/classes/log4j.properties
Grouper is logging to file:    /tmp/logpipe, at min level WARN for package: edu.internet2.middleware.grouper,
based on log4j.properties
grouper.hibernate.properties: /opt/grouper/grouperWebapp/WEB-INF/classes/grouper.hibernate.properties
grouper.hibernate.properties: root@jdbc:mysql://localhost:3306/grouper?
Charset=utf8&useUnicode=true&characterEncoding=utf8

subject.properties read from: /opt/grouper/grouperWebapp/WEB-INF/classes/subject.properties
sources configured in:        subject.properties
subject.properties ldap source id:   eduLDAP: demo
subject.properties internalsource id:g:isa
subject.properties groupersource id: g:gsa
subject.properties groupersource id: grouperEntities
subject.properties jdbc2 source id:  grouperExternal: GrouperJdbcConnectionProvider
Type help() for instructions
Groovy Shell (2.5.0-beta-2, JVM: 1.8.0_302)
Type ':help' or ':h' for help.
-----
groovy:000> :load '/opt/grouper/grouperWebapp/WEB-INF/classes/groovysh.profile'
groovy:000>
```

Note that the Grouper version is indicated, information on what the connection string to the Grouper database is, where various configuration files are read from, and available subject sources. If your subject source is based on a database instead of ldap, you will see information on the connection string to that database too.

A friendly GroovyShell prompt greets us at the end (groovy:000>) and we are ready to start interacting with GSH. We'll start by looking up information about the subject banderson.

```
def subj = findSubject("banderson")
```

Result:

```
====> Subject id: 800001147, sourceId: eduLDAP, name: Bob Anderson
```

Groovy is Java-like syntax, and has strong variable type checking, but it is optional. We could have written "Subject subj = ..." and the interpreter would make sure it returns a value of type edu.internet2.middleware.subject.Subject. In this case, we are not worried about being strict, and we write "def" to indicate that the interpreter will figure out the type at runtime.

Also note that GSH will output information about the last run command. This is helpful in validating that the banderson we found is in fact Bob Anderson, but let's make sure that this is really Bob Anderson, the Grouper administrator, and not some other imposter Bob Anderson. Let's check the members of the sysadmin group, to check whether that group's Bob Anderson matches.

```
def admins = getMembers("etc:sysadmin")
```

When we run that command though, we see an error:

```
ERROR edu.internet2.middleware.grouper.exception.GroupNotFoundException:
Cannot find group with name: 'etc:sysadmin'
    at edu.internet2.middleware.grouper.internal.dao.hib3.Hib3GroupDAO.findByName (Hib3GroupDAO.java:1182)
    at edu.internet2.middleware.grouper.internal.dao.hib3.Hib3GroupDAO.findByName (Hib3GroupDAO.java:1144)
    at edu.internet2.middleware.grouper.GroupFinder.findByNameNoCache (GroupFinder.java:623)
    at edu.internet2.middleware.grouper.GroupFinder.findByName (GroupFinder.java:593)
    at edu.internet2.middleware.grouper.GroupFinder.findByName (GroupFinder.java:537)
    at edu.internet2.middleware.grouper.app.gsh.getMembers.invoke (getMembers.java:79)
    at edu.internet2.middleware.grouper.app.gsh.getMembers$invoke.call (Unknown Source)
    at groovysh_evaluate.getMembers (groovysh_evaluate:4)
```

GSH displays a friendly version of the problem (highlighted in red on most consoles) and we can quickly see that we just put in the wrong group name. The additional stack information is helpful if you ever need to report a problem out to [Grouper's bug reporting system](#).

Let's fix our mistake. Use the up arrow to go to your previous command to fix it up without having to retype the entire thing so it matches the command below:

```
def admins = getMembers("etc:sysadmingroup")
```

Result:

```
====> ['0903df94669042ef8ac9995ad9d61fbf'/'group'/'g:gsa', '80000026'/'person'/'eduLDAP', '800001026'/'person'/'eduLDAP', '800002061'/'person'/'eduLDAP', '800000252'/'person'/'eduLDAP', '800001147'/'person'/'eduLDAP']
```

As with all commands, the result of the evaluation is output to the console, even if it's not assigned to a variable. In this case, the result is a list of Member objects. The group is small enough that we can pick out that Bob Anderson's subject id 800001147 is in the list. A more precise way is to convert the Member objects in the list to Subject objects (the Member class has a method to do the conversion), and check whether our banderson Subject is in the list:

```
admins.collect { member -> member.getSubject() }.contains(subj)
```

```
==> true
```

The answer is yes, the subject in question, banderson, is one of the members of the sysadmin group. The command above introduces some idiomatic Groovy mixed with Java. The collect method is a Groovy method that takes a block of code, runs it over each member of a list, and returns a new list of the converted values. The contains() method is a Java method that checks whether an object is within a list. Note that we did not specify a variable to assign because we did not care to use the result for anything else. The console still output the result of the evaluation.

## Exercise 301.4.2 - Creating a folder/group structure

Most GSH commands run under the context of a specific user, in order to check appropriate privileges, and log audit actions. The default session in GSH is the root subject GrouperSystem. We want to run the commands as banderson, so we will start a new session as that user. The command needs a Subject variable to pass in, and we can use the subj variable that we already defined.

```
GrouperSession session = GrouperSession.start(subj)
```

```
==> 2b54a7f37d28408fa39ba3edf1e42818, '800001147', 'person'
```

Note that we used subj from the above exercise where we looked up and found banderson. Let's create a folder called tmp:

```
new StemSave().assignName("tmp").assignDisplayName("Temporary Folder at Root").save()  
// ==> Stem[displayName=Temporary Folder at Root,name=tmp,uuid=624bbb508b884d39993b0779bd64a9b2,  
creator=e77dbca323e94bf487041e638523d5b0]
```

This uses a chained builder class to set up how we want to create the folder, finally calling save() to run the action. We set up the internal path as "tmp", and a display name of "Temporary Folder at Root".

Now let's create a folder within tmp:

```
new StemSave().assignName("tmp:subfolder").assignDisplayExtension("Temp Subfolder").save()  
// ==> Stem[displayName=Temporary Folder at Root:Temp Subfolder,name=tmp:subfolder,  
uuid=10aca691666e4269b6f7ed224f46e65b,creator=e77dbca323e94bf487041e638523d5b0]
```

We specified the full path of the folder to make this folder in. In this case we set a display name for the folder, but we only needed the last part of it, the "extension", since the parent folder already has a name.

**IMPORTANT:** You are often acting as a super user in GSH, so do be careful with any commands to ensure you do not accidentally destroy any data in your Grouper environment. GSH will not prompt you if it is OK to delete something once you hit enter on the command.

Take a look in the UI (you will want to login as banderson for these exercises). We will see our new folder and subfolder created:

## Temporary Folder at Root

Edit folder

More actions ▾

Path:	Temporary Folder at Root
ID path:	tmp
Alternate ID path:	
ID:	tmp
Created:	2020/10/08 05:38:45.976
Creator:	Bob Anderson (banderson, )
Last edited:	
Last editor:	
ID index:	10068
UUID:	1bbb3572abe34e30bf1efcdf1842de5d

Less ^

Folder contents

Privileges

More ▾

Filter for:

Apply filter

Reset

Name ▾

^ Up one folder

Temp Subfolder

Show:

Showing 1-1 of 1 · First | Prev | Next | Last

Note that the creator of the folder is Bob Anderson, because we started the Grouper session as banderson. If you started a root session, the creator would instead be GrouperSysAdmin.

Click on More Actions -> View audit log.

Home > Root > Temporary Folder at Root

# Temporary Folder at Root

Edit folder

More actions ▼

More ▼

Folder contents
Privileges

The audit log displays all recent activity related to this folder.

Filter by date:

all ▼

yyyy/mm/dd

( and

yyyy/mm/dd

)
☐ Show extended results?
Find entries

Date ▼	Actor	Engine	Summary
2020/10/08 05:40 AM	Bob Anderson (banderson, )	GSH command	Added folder Temp Subfolder .
2020/10/08 05:38 AM	Bob Anderson (banderson, )	GSH command	Added folder Temporary Folder at Root .

Show:

50 ▼

Showing 1-2 of 2 · First | Prev | Next | Last

Note that the audit log shows that this folder was created through a GSH command. This can be very valuable in determining whether it was Bob Anderson logged in through the UI creating folders or someone acting as Bob Anderson via GSH that created the folder instead. We'll look at an example audit log with both when we create a group next.

Back in your GSH console, now create a group. Unlike the folder creation above, we want to save the result to a variable so we can reuse it.

```
Group group = new GroupSave().assignName("tmp:subfolder:test_group").assignDisplayExtension("Testing Group").
save()
// ==> Group[name=tmp:subfolder:test_group,uuid=14dee38631944cfca8390cf0323d7aa3]
```

This created the group, and the set that newly created group to a variable of type `edu.internet2.middleware.grouper.Group`. This Group class has a method to add a member.

...and add a member to it:

```
group.addMember("jsmith")
```

```
ERROR groovy.lang.MissingMethodException:
No signature of method: edu.internet2.middleware.grouper.Group.addMember() is applicable for argument types:
(String) values: [jsmith]
Possible solutions: addMember(edu.internet2.middleware.subject.Subject), addMember(edu.internet2.middleware.
subject.Subject, boolean), addMember(edu.internet2.middleware.subject.Subject, edu.internet2.middleware.grouper.
Field), addMember(edu.internet2.middleware.subject.Subject, edu.internet2.middleware.grouper.Field, boolean),
toMember(), hasMember(edu.internet2.middleware.subject.Subject)
```

We guessed wrong at the method call, and there is no `Group.addMember()` method that takes a string. The interpreter can help with some suggestions of alternatives that would work. There is a version of `addMember()` that takes a Subject type. So we need to get the Subject for jsmith, similar to how we did for banderson.

```
Subject jsmith = findSubject("jsmith")
// ==> Subject id: 800001278, sourceId: eduLDAP, name: Joel Smith

group.addMember(jsmith)
// ==> null
```

The `addMember()` command didn't return any value. But we would have seen a Java exception if it failed. We can get an exception if we try to add the same subject twice:

```
group.addMember(jsmith)
```

```
ERROR edu.internet2.middleware.grouper.exception.MemberAddAlreadyExistsException:
membership already exists,
Problem in HibernateSession: HibernateSession (11d0bff5): notNew, notReadOnly, READ_WRITE_NEW,
activeTransaction, session (6c569caf), membership: group: tmp:subfolder:test_group, subject: 800001278, field:
members, uuid: null, startDate: null, endDate: null,
, group name: tmp:subfolder:test_group, subject: Subject id: 800001278, sourceId: eduLDAP, field: members,
Problem in HibernateSession: HibernateSession (233d2b76): new, notReadOnly, READ_WRITE_NEW,
notActiveTransaction, session (6c569caf)
    at edu.internet2.middleware.grouper.Membership.internal_addImmediateMembership (Membership.java:1296)
    at edu.internet2.middleware.grouper.Group$4.callback (Group.java:1621)
    at edu.internet2.middleware.grouper.hibernate.HibernateSession.callbackHibernateSession
(HibernateSession.java:703)
    at edu.internet2.middleware.grouper.Group.internal_addMember (Group.java:1593)
    at edu.internet2.middleware.grouper.Group.internal_addMember (Group.java:1543)
    at edu.internet2.middleware.grouper.Group.addMember (Group.java:1124)
    at edu.internet2.middleware.grouper.Group.addMember (Group.java:1043)
    at edu.internet2.middleware.grouper.Group.addMember (Group.java:1006)
    at edu.internet2.middleware.grouper.Group$addMember.call (Unknown Source)
```


Depending on our goal, we may want the API to quietly ignore attempts to add an existing member. There is a second `addMember()` method that takes a second parameter, `exceptionIfAlreadyMember`:

```
group.addMember(jsmith, false)
// ==> false
```

(As a reminder, the arguments for all of these commands are available on the [GSH wiki page](#))

Let's take a look in the UI at our new group we just created:

[Home](#) > [Root](#) > [Temporary Folder at Root](#) > [Temp Subfolder](#) > [Testing Group](#)

 **Testing Group**

+ Add members

More actions ▾

Members

Privileges

More ▾

The following table lists all entities which are members of this group.

Filter for: 

All members ▾


Member name

Apply filter

Reset

Advanced

Remove selected members

<input type="checkbox"/> Entity name ▾	Membership	Choose action
<input type="checkbox"/>  John Smith (jsmith, )	Direct	<div>Actions ▾</div>

Show: 

50 ▾

Showing 1-1 of 1 · [First](#) | [Prev](#) | [Next](#) | [Last](#)

While you are in the UI, go ahead and also add 'aadams' to the group too as a member. Once you have done that, take a look at the audit log for the group:

Home > Root > Temporary Folder at Root > subfolder > Testing Group

# Testing Group

More ▾

MembersPrivilegesMore ▾

+ Add members

More actions ▾

The audit log displays all recent activity related to this group.

Filter by date:  ( and  ) ☐ Show extended results?

Date ▾	Actor	Engine	Summary
2021/09/30 4:44 AM	Bob Anderson	Web user interface	Added Ashley Adams as a member of the Testing Group group.
2021/09/30 4:43 AM	Bob Anderson	GSH command	Added Joel Smith as a member of the Testing Group group.
2021/09/30 4:43 AM	Bob Anderson	GSH command	Added group Testing Group .
<div> Show: <input type="text" value="50"/> </div> <div> Showing 1-3 of 3 · First   Prev   Next   Last </div>			

Note how even though both group member adds were done as Bob Anderson, we can see that the second add was down through the UI while the first was done as a GSH command.

### Exercise 301.4.3 - Working with the API

The first commands we ran – `findSubject("banderson")` and `getMembers("etc:sysadmin")` – are helper commands that often take strings, meant to simplify common operations. Other method calls like `Group.addMember()` tap into the Java API. All of the Java public methods in the Grouper jar -- plus methods from all the other jars in the `WEB-INF/lib` directory – can be called from within GSH.

The [GSH wiki](#) is the best starting point to search for ways of accomplishing GSH goals. The Groovy interpreter also can give hints via tab completion. For example, typing "subj." and hitting the tab key will list the public methods that can be called on it.

```
// type in subj. and hit tab

groovy:000> subj.
getAttributeValue(           getAttributeValueOrCommaSeparated(  getAttributeValueSingleValued
(       getAttributeValues(           getAttributes(
attributes                    description                    descriptionOverride        i
d                             name                               name                         t
nameOverride                 source                           sourceId
ranslationMap                type
typeName
```

For a deeper dive, the Javadoc documentation for callable methods is [available online](#).

### Exercise 301.4.4 - Automation

We have received a request from an application owner to:

- Create a folder `app:foo:userGroups`
- Given a list of user ids, create a group named as the id, but make the display name the user's first and last name (i.e., the LDAP "cn" attribute)
- Grant each user read and update permissions for their group

Depending on how many users are in the list, this is potentially tedious to do through the UI. All these steps can be automated with GSH. For the incoming ids, a data object can be created within the script and initialized. Depending on their needs, modifying this step to read data from a file can be a later enhancement.

The users' names don't need to be part of the incoming data. Because of how the LDAP subject source is configured, Grouper can access certain attributes from a subject. Let's figure out how to get the subject first and last name.

```
subj.getAttributes()
// ==> [sn:[Anderson], employeeNumber:[800001147], cn:[Bob Anderson], uid:[banderson], givenname:[Bob], mail:[Bob.Anderson@mock.edu.invalid]]

subj.getAttributeValue("cn")
// ==> Bob Anderson
```

First, set up the ids, and create the folder.

```
def ids = [
    "pharris",
    "ganderso",
    "ljacobso",
    "mroberts",
    "mhoward",
    "rharris",
    "cjohnson",
    "jhenders",
    "jbrown",
    "jsummers"
]

Stem stem = new StemSave().assignName("app:foo:userGroups").assignCreateParentStemsIfNotExist(true) .save()
```

Do a quick test to make sure the subjects all resolve, and have a name field we can use. This example uses idiomatic Groovy to loop through subjects. It is also fine to use Java syntax, i.e. "for (String id: ids) {...":

```
ids.each { id ->
    Subject subj = findSubject(id)
    def cn = subj.getAttributeValue("cn")
    println cn
}

/*
Patricia Harris
Gregory Anderson
Lisa Jacobson
Melissa Roberts
Megan Howard
Ronald Harris
Christopher Johnson
Jeremiah Henderson
James Brown
John Summers
*/
```

Everything looks good, so do the loop again, but now create the groups and assign the privileges. The grantPriv helper command takes Privilege Java objects, in this case Privilege.READ and Privilege.UPDATE.




```
ids.each { id ->
  Subject subj = findSubject(id)
  def cn = subj.getAttributeValue("cn")
  Group g = new GroupSave().assignName("app:foo:userGroups:${id}").assignDisplayExtension(cn).save()
  grantPriv(g.getName(), id, Privilege.READ)
  grantPriv(g.getName(), id, Privilege.UPDATE)
  println "Done with ${g.getName()}"
}

/*
Done with app:foo:userGroups:pharris
Done with app:foo:userGroups:ganderso
Done with app:foo:userGroups:ljacobso
Done with app:foo:userGroups:mroberts
Done with app:foo:userGroups:mhoward
Done with app:foo:userGroups:rharris
Done with app:foo:userGroups:cjohnson
Done with app:foo:userGroups:jhenders
Done with app:foo:userGroups:jbrown
Done with app:foo:userGroups:jsummers
==> [pharris, ganderso, ljacobso, mroberts, mhoward, rharris, cjohnson, jhenders, jbrown, jsummers]
*/
```

In the UI, verify the groups were created. In the Privileges tab, spot check that read/update privileges have been created for the user matching the group name.

Home > Root > app > foo > userGroups > Christopher Johnson



# Christopher Johnson

+ Add members
Group actions


Show details

MembersPrivilegesMore

The following table lists all entities with privileges on this group.

Filter for:

Update:

<input type="checkbox"/> Entity name	Admin	Read	Update	OptIn	OptOut	Attribute read	Attribute update	View	Choose action
<input type="checkbox"/>  Christopher Johnson		✓	✓	✓	✓			✓	<input type="button" value="Actions"/>

Show:

Showing 1-1 of 1 · First | Prev | Next | Last

## (Extra) Exercise 301.4.5 - Build a Report (making a GSH script)

In this exercise, we are going create an automatically generated report of all memberships for something like a daily report to a security officer around campus who is interested in who has access to what for auditing purposes. GSH can help us make that pretty simple without having to write any sort of database query. We will be generating that report using GSH, but also looking at how to run a script in GSH without having to copy and paste all of the lines of the script.

Open your favorite text editor and copy in the following script:

```

def session = GrouperSession.startRootSession();
def group = GroupFinder.findByName(session, "basis:sis:prog_status:all:xo", true);
def effectiveMembers = group.getEffectiveMembers();
def immediateMembers = group.getImmediateMembers();

def writer = new File('/tmp/out.txt').newWriter('UTF-8')
writer.println(String.join("\t", "id", "name", "Effective", "Immediate"));
for (Member m: group.getMembers()) {
    writer.print(m.getSubject().getId() + "\t" + m.getSubject().getName() + "\t");
    writer.print(effectiveMembers.contains(m).toString() + "\t");
    writer.println(immediateMembers.contains(m).toString() + "\t");
}
writer.close();

```

Save your script (in this case called report.gsh) and here are two ways we can run the gsh script:

Copy the script in to the container and then run it:

```
$ docker cp report.gsh 101.1.1:/tmp/report.gsh
```

Open a shell into the container and switch to the tomcat user:

```
$ ./gte-shell
```

```
$ sudo -u tomcat /bin/bash
```

Run the report

```
$ bin/gsh.sh /tmp/report.gsh
```

Note that data in containers is not persistent and it would be better to mount in a directory where you keep your GSH scripts.

The output shows very little other than the output for each command run:

```

groovy:000> :load '/opt/grouper/grouperWebapp/WEB-INF/classes/groovysh.profile'
groovy:000> :gshFileLoad '/tmp/report.gsh'
====> 93b3848ffecf418584fd24fa0e3c9405,'GrouperSystem','application'
====> Group[name=basis:sis:prog_status:all:xo,uuid=807ef842a7a64928b3fb64f3fff343ae]
====> []
====> ['800000785'/'person'/'eduLDAP', '800001652'/'person'/'eduLDAP', '800002031'/'person'/'eduLDAP',
'800000236'/'person'/'eduLDAP', '800000107'/'person'/'eduLDAP', '800002227'/'person'/'eduLDAP', '800002752'/'person'/'eduLDAP',
'800001853'/'person'/'eduLDAP', '800001200'/'person'/'eduLDAP', '800001673'/'person'/'eduLDAP',
'800002431'/'person'/'eduLDAP', '800002675'/'person'/'eduLDAP', '800002581'/'person'/'eduLDAP',
'800002426'/'person'/'eduLDAP']
====> groovy.io.EncodingAwareBufferedWriter@61b76a7d
====> null
====> null
====> null
groovy:000> :exit

```

We can see our output report by running the following:

```
$ docker exec -i 101.1.1 cat /tmp/out.txt
```

id	name	Effective	Immediate
800000785	Dawn Stewart	false	true
800001652	Brianna Ballard	false	true
800002031	Kristen Taylor	false	true
800000236	Jennifer Hunt	false	true
800000107	Christopher Johnson	false	true
800002227	Jeffrey Freeman	false	true
800002752	Jesse Stafford	false	true
800001853	Kevin Mendoza	false	true

800001200	Laura Villa	false	true
800001673	Ray Miranda	false	true
800002431	Ryan Bennett	false	true
800002675	Sara Jones	false	true
800002581	Brian Hernandez	false	true
800002426	Gerald Perkins	false	true

...and we see our tab separated report we can now ship off to whoever requested it.

You could execute your recurring GSH scripts either by calling the docker commands from cron, or perhaps creating a container based on tier-grouper that runs crond instead of any of the grouper components. Your container would be built with the crontab and you would be able to schedule any functionality you need with grouper components like GSH. But that is an exercise left to the reader (for now).

## (Extra) Exercise 301.4.6 - Creating and Running a Grouper Loader Job

For this exercise, we are going to create a very simple loader job that loads all subjects from the grouper members database table and run it. While the UI does provide a very easy to use mechanism to create and run loader jobs, there are times where you may find it convenient to use GSH instead. GSH can be helpful if there are a bunch of loader groups you need to create and do not want to click through the UI for each one. Much of how to create Grouper Loader jobs in GSH is documented on the [Grouper Loader wiki page](#).

First, create the group.

```
session = GrouperSession.startRootSession();
addStem("test", "subfolder", "subfolder")
addGroup("test:subfolder", "loader_test", "Test Loader Group")
```

Now assign it the attributes needed to make it a loader job. This time, instead of writing one line at a time, try copying the entire block below in to GSH:

```
session = GrouperSession.startRootSession();
groupAddType("test:subfolder:loader_test", "grouperLoader");
setGroupAttr("test:subfolder:loader_test", "grouperLoaderDbName", "grouper");
setGroupAttr("test:subfolder:loader_test", "grouperLoaderType", "SQL_SIMPLE");
setGroupAttr("test:subfolder:loader_test", "grouperLoaderScheduleType", "CRON");
setGroupAttr("test:subfolder:loader_test", "grouperLoaderQuartzCron", "0 * * * * ?");
setGroupAttr("test:subfolder:loader_test", "grouperLoaderQuery", "select distinct subject_id as SUBJECT_ID,
subject_source SUBJECT_SOURCE_ID from grouper.grouper_members where subject_source = 'eduLDAP'");
```

GSH handled the line feeds between commands and you should see each command run followed by 'true' printed after each line. In the next exercise we will show you how to run a block of commands as a script, which is the preferred way over copy/paste many lines at a time.

Now let's dry run the job to see what it would do. Because dry run requires a group object and not the group id as a string we are going to put a function to find the group in as the argument to dry run command:

```

session = GrouperSession.startRootSession();
loaderDryRunOneJob(GroupFinder.findByName(session, 'test:subfolder:loader_test'), null)

/*
Group: test:subfolder:loader_test add Subject id: 800001356, sourceId: eduLDAP
Group: test:subfolder:loader_test add Subject id: 800002613, sourceId: eduLDAP
Group: test:subfolder:loader_test add Subject id: 800001651, sourceId: eduLDAP
Group: test:subfolder:loader_test add Subject id: 800000099, sourceId: eduLDAP
...
==> loader dry ran successfully, would have inserted 2884 memberships, would have deleted 0 memberships, total
membership count: 2884, unresolvable subjects: 0
*/

```

Note that variable called 'session' that we set back in the first exercise is needed for the GroupFinder functions. It seems that what this loader job will do is fine, so let's run the job:

```

session = GrouperSession.startRootSession();
loaderRunOneJob(GroupFinder.findByName(session, 'test:subfolder:loader_test'))

/*
==> loader ran successfully, inserted 2884 memberships, deleted 0 memberships, total membership count: 2884,
unresolvable subjects: 0
*/

```

Verify in the UI that the membership count matches what GSH reported. Also check out under More->Loader that the settings are the same as if you were to configure a loader job in the UI

## (Extra) Exercise 301.4.7 - Burn it Down

Sometimes there are folders in Grouper that you simply want to destroy everything under including all history that it ever existed. A good example of this are course enrollment groups for a particular semester that are likely a lot of data and that changed often during the semester. After a certain amount of time, these groups will be meaningless and can be destroyed (including all history) in order to free up some storage. We do not have any course enrollments in this exercise, so let's destroy all of our hard work from Exercise 301 instead.

If you do not have GSH open, fire it up again and run the following:

```

obliterateStem("test", true, true);

/*
Would obliterate stem: test
Would obliterate stem: test:subfolder
Would be done deleting group: test:subfolder:loader_test
Would be done deleting group: test:subfolder:test_group
Would be done obliterating stem: test:subfolder
Would be done obliterating stem: test
==> true
*/

```

The first argument specifies what stem we want to destroy. The second argument is set to true as a test only mode. The third argument specifies whether you want this also destroyed from all point in time data too. We see above that it would destroy all of our work today as expected. Let's do it:

```
obliterateStem("test", false, true);
/*
Obliterating stem: test
Obliterating stem: test:subfolder
Done deleting group: test:subfolder:loader_test
Done deleting group: test:subfolder:test_group
Done obliterating stem: test:subfolder
Done obliterating stem: test
Waiting for Grouper Daemon to process before obliterating from point in time data. This is expected to take a
few minutes. Be sure the Grouper Daemon is running.
Obliterating stem from point in time: test, ID=b8b1cbecf1bb4919822d0dd412c86c4a
Done deleting group from point in time: test:test, ID=fdbdb9881c28e43f8b0d1756481d6651f
Obliterating stem from point in time: test:subfolder, ID=ac9a6978f1a94849a3760eb1bd606824
Done deleting group from point in time: test:subfolder:loader_test, ID=4597c45d2d9f454ca56793f6f6d56ddc
Done deleting group from point in time: test:subfolder:test_group, ID=9ca911618ff74ad2ac0d954fff8fa3c9
Done obliterating stem from point in time: test:subfolder, ID=ac9a6978f1a94849a3760eb1bd606824
Done obliterating stem from point in time: test, ID=b8b1cbecf1bb4919822d0dd412c86c4a
====> true*/
```

This may take a while if you have a very large set of folders/groups to delete. This can also be very dangerous since, as you can see, there was no confirmation asked when we ran the command above before it destroyed everything. We can look in the Grouper UI to verify that our work is no longer there:

## Conclusion

You have now used GSH to manipulate folders, groups, and members in Grouper and even create a report that could be automated by running script on a regular basis through cron. Finally, we used GSH to efficiently destroy everything from a given folder. These were all basic examples, but we hope that you can begin to see the potential power for GSH commands in creating things like templates for new services, bulk adding/removing members, writing a script a script to bootstrap your development environment, etc. All of these training exercises are built out using GSH scripts.

Here is an example from the training environment that bootstraps the environment by building out a complete tree of folders and groups, adds members, grants privileges, creates a loader job, add attributes, creates a composite group and assigns Grouper rules.

[https://github.internet2.edu/docker/grouper\\_training/blob/master/full-demo/container\\_files/demo.gsh](https://github.internet2.edu/docker/grouper_training/blob/master/full-demo/container_files/demo.gsh)