Grouper subject filter and attribute decorator

These topics are discussed in the "Grouper Subject API - Part 4" training video.

This Grouper enhancement facilitates secure subject attribute release. This could be used to protect FERPA information, create private subdomains of Grouper where users from one might not be able to see subjects in another, or attributes which can only be seen by certain subjects.

In Grouper 2.0 and previous there is no subject attribute security (unless it is baked into the custom subject source). This is only available in v2.1 and more recent.

This is an interface point that can be implemented in Java to retrieve more attributes about a subject (i.e. attributes that may require another query, or security, or attributes not always needed when subjects are resolved). When you implement this interface you should try to only use one (or few) queries, since this will be called a lot (for filters at least).

Implement the interface SubjectCustomizer by extending the class: edu.internet2.middleware.grouper.subj.SubjectCustomizerBase

```
/*******************************************************************************
 * Copyright 2012 Internet2
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *   http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 ******************************************************************************/
/**
 *
 */
package edu.internet2.middleware.grouper.subj;

import java.util.Collection;
import java.util.Set;

import edu.internet2.middleware.grouper.GrouperSession;
import edu.internet2.middleware.subject.Subject;


/**
 * add the ability to decorate a list of subjects with more attributes.
 * note, while you are decorating, you can check security to see if the
 * groupersession is allowed to see those attributes
 * @author mchyzer
 *
 */
public interface SubjectCustomizer {

  /**
   * decorate subjects based on attributes requested
   * @param grouperSession
   * @param subjects
   * @param attributeNamesRequested
   * @return the subjects if same set, or make a new set
   */
  public Set<Subject> decorateSubjects(GrouperSession grouperSession, Set<Subject> subjects, Collection<String>
attributeNamesRequested);

  /**
   * you can edit the subjects (or replace), but you shouldnt remove them
   * @param grouperSession
   * @param subjects
   * @param findSubjectsInStemName if this is a findSubjectsInStem call, this is the stem name.  This is useful
   * to filter when searching for subjects to add to a certain group
   * @return the subjects if same set, or make a new set
   */
  public Set<Subject> filterSubjects(GrouperSession grouperSession, Set<Subject> subjects, String
findSubjectsInStemName);


}
```

Configure this in the grouper.properties:

```
# customize subjects by implementing this interface: edu.internet2.middleware.grouper.subj.SubjectCustomizer
# or extending this class: edu.internet2.middleware.grouper.subj.SubjectCustomizerBase (recommended)
# note the instance will be reused to make sure it is threadsafe
subjects.customizer.className =
```

Note that the filter and decorator are batched for performance reasons, which is also why this is not configured in the subject source

The filterSubjects() method is called on each SubjectFinder call. The decorateSubjects() is called when web services resolve subjects

There are two ways to protect data.

1. You can change the data to protect it, the user will see the subject, but might only see the netId instead of the name. Or it could just say (protected-data)

2. You could remove the subject from the results and the user will not know the subject exists

Here is an example of hiding names of students to users who aren't allowed to see them (note, you can still search by private information potentially, though you will not see the result. If you want something more complex you might need a custom source, or only have public information in the search field)

```
/*******************************************************************************
 * Copyright 2012 Internet2
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *   http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 ******************************************************************************/
package edu.internet2.middleware.grouper.subj.decoratorExamples;

import java.util.LinkedHashSet;
import java.util.Set;

import org.apache.commons.lang.StringUtils;

import edu.internet2.middleware.grouper.GrouperSession;
import edu.internet2.middleware.grouper.MembershipFinder;
import edu.internet2.middleware.grouper.membership.MembershipResult;
import edu.internet2.middleware.grouper.subj.SubjectCustomizer;
import edu.internet2.middleware.grouper.subj.SubjectCustomizerBase;
import edu.internet2.middleware.grouper.util.GrouperUtil;
import edu.internet2.middleware.subject.Subject;
import edu.internet2.middleware.subject.provider.SubjectImpl;

/**
 * filter students private information out from people who cant see them
 * @author mchyzer
 *
 */
public class SubjectCustomizerForDecoratorTestingHideStudentData extends SubjectCustomizerBase {

  /** student (protected data) group name */
  public static final String STUDENT_GROUP_NAME = "apps:subjectSecurity:groups:student";
  /** privileged employee group name */
  public static final String PRIVILEGED_EMPLOYEE_GROUP_NAME = "apps:subjectSecurity:groups:privilegedEmployee";

  /** source id we care about */
  private static final String SOURCE_ID = "jdbc";

  /**
   * @see SubjectCustomizer#filterSubjects(GrouperSession, Set, String)
   */
  @Override
```

```
  public Set<Subject> filterSubjects(GrouperSession grouperSession, Set<Subject> subjects, String
findSubjectsInStemName) {

    //nothing to do if no results
    if (GrouperUtil.length(subjects) == 0) {
      return subjects;
    }

    //get results in one query
    MembershipResult groupMembershipResult = new MembershipFinder().assignCheckSecurity(false).addGroup
(STUDENT_GROUP_NAME)
        .addGroup(PRIVILEGED_EMPLOYEE_GROUP_NAME).addSubjects(subjects).addSubject(grouperSession.getSubject())
        .findMembershipResult();

    //see if the user is privileged
    boolean grouperSessionIsPrivileged = groupMembershipResult.hasGroupMembership
(PRIVILEGED_EMPLOYEE_GROUP_NAME, grouperSession.getSubject());

    //if so, we are done, they can see stuff
    if (grouperSessionIsPrivileged) {
      return subjects;
    }

    //loop through the subjects and see which are students, change their name and description to be their
netId, with no other attributes
    Set<Subject> results = new LinkedHashSet<Subject>();
    for (Subject subject : subjects) {
      if (StringUtils.equals(SOURCE_ID, subject.getSourceId()) && groupMembershipResult.hasGroupMembership
(STUDENT_GROUP_NAME, subject)) {
        String loginid = subject.getAttributeValue("loginid");
        Subject replacementSubject = new SubjectImpl(subject.getId(), loginid, loginid, subject.getTypeName(),
subject.getSourceId());
        results.add(replacementSubject);
      } else {
        results.add(subject);
      }
    }
    return results;
  }
}
```

API access

In order to use the decorator part, you need to make an extra call to the SubjectFinder for your subjects

```
SubjectFinder.decorateSubjects(GrouperSession.staticGrouperSession(), subjects,
subjectAttributesForDecoratorSet);
```

WS access

You can use this via WS by configuring which subject attributes will trigger a call to the SubjectFinder.decorateSubjects() method in the grouper-ws. properties:

```
# customize subjects by implementing this interface: edu.internet2.middleware.grouper.subj.SubjectCustomizer
# or extending this class: edu.internet2.middleware.grouper.subj.SubjectCustomizerBase (recommended)
# note the instance will be reused to make sure it is threadsafe
subjects.customizer.className = edu.internet2.middleware.grouper.subj.decoratorExamples.
SubjectCustomizerForDecoratorExtraAttributes
```

Then when the user requests certain attributes for subjects, the subject customizer will fire and can resolve the attributes. Here is an example client call:

```
C:\mchyzer\grouper\trunk\grouperClient\dist>java -jar grouperClient.jar --operation=getSubjectsWs --
subjectIds=test.subject.1 --subjectAttributeNames=title --outputTemplate="Index: ${index}: success: ${success},
code: ${wsSubject.resultCode}, subject:"${wsSubject.id}, title: ${wsSubject.attributeValues[0]}$newline$"
Index: 0: success: T, code: SUCCESS, subject: test.subject.1, title: title1
```

Here is the XML (POX) request:

```
```

And here is the response (the title for this subject is "title1"

```
<WsRestGetSubjectsRequest>
    <subjectAttributeNames>
        <string>title</string>
    </subjectAttributeNames>
    <wsSubjectLookups>
        <WsSubjectLookup>
            <subjectId>test.subject.1</subjectId>
        </WsSubjectLookup>
    </wsSubjectLookups>
</WsRestGetSubjectsRequest>
```

Example: collaboration groups

Here is an example where subjects not in collab groups are filtered from users not in that group

```
package edu.internet2.middleware.grouper.subj;

import java.util.LinkedHashSet;
import java.util.Set;

import org.apache.commons.collections.CollectionUtils;
import org.apache.commons.lang.StringUtils;

import edu.internet2.middleware.grouper.GrouperSession;
import edu.internet2.middleware.grouper.membership.GroupMembershipResult;
import edu.internet2.middleware.grouper.util.GrouperUtil;
import edu.internet2.middleware.subject.Subject;

/**
 * filter out subjects not in the collaboration group
 * @author mchyzer
 *
 */
public class SubjectCustomizerForDecoratorTesting2 extends SubjectCustomizerBase {

  /** student (protected data) group name */
  private static final String COLLAB_STEM_NAME = "collaboration:collabGroups";

  /** privileged employee group name */
  private static final String PRIVILEGED_ADMIN_GROUP_NAME = "collaboration:etc:privilegedAdmin";

  /** source id we care about */
  private static final String SOURCE_ID = "jdbc";

  /**
   * @see SubjectCustomizer#filterSubjects(GrouperSession, Set)
   */
  @Override
  public Set<Subject> filterSubjects(GrouperSession grouperSession, Set<Subject> subjects) {

    //nothing to do if no results
    if (GrouperUtil.length(subjects) == 0) {
      return subjects;
```

```
    }

    //get results in one query
    GroupMembershipResult groupMembershipResult = calculateMembershipsInStems(subjects,
IncludeGrouperSessionSubject.TRUE,
        GrouperUtil.toSet(PRIVILEGED_ADMIN_GROUP_NAME), GrouperUtil.toSet(COLLAB_STEM_NAME));

    //see if the user is privileged
    boolean grouperSessionIsPrivileged = groupMembershipResult.hasMembership(PRIVILEGED_ADMIN_GROUP_NAME,
grouperSession.getSubject());

    //if so, we are done, they can see stuff
    if (grouperSessionIsPrivileged) {      return subjects;    }

    //get the group names that the grouper session subject is in
    Set<String> grouperSessionGroupNames = groupMembershipResult.groupNamesInStem(grouperSession.getSubject(),
COLLAB_STEM_NAME);

    //loop through the subjects and see which collab groups the users are in
    Set<Subject> results = new LinkedHashSet<Subject>();
    for (Subject subject : subjects) {
      //only protect one source
      if (!StringUtils.equals(SOURCE_ID, subject.getSourceId())) {
        results.add(subject);
      } else {
        Set<String> subjectGroupNames = groupMembershipResult.groupNamesInStem(subject, COLLAB_STEM_NAME);
        if (CollectionUtils.containsAny(grouperSessionGroupNames, subjectGroupNames)) {
          results.add(subject);
        }
      }
    }
    return results;
  }



}
```

Example: hide student data to most users

Here is an example where student data is hidden to most users

```
package edu.internet2.middleware.grouper.subj.decoratorExamples;

import java.util.LinkedHashSet;
import java.util.Set;

import org.apache.commons.lang.StringUtils;

import edu.internet2.middleware.grouper.GrouperSession;
import edu.internet2.middleware.grouper.MembershipFinder;
import edu.internet2.middleware.grouper.membership.MembershipResult;
import edu.internet2.middleware.grouper.subj.SubjectCustomizer;
import edu.internet2.middleware.grouper.subj.SubjectCustomizerBase;
import edu.internet2.middleware.grouper.util.GrouperUtil;
import edu.internet2.middleware.subject.Subject;
import edu.internet2.middleware.subject.provider.SubjectImpl;

/**
 * filter students private information out from people who cant see them
 * @author mchyzer
 *
 */
public class SubjectCustomizerForDecoratorTestingHideStudentData extends SubjectCustomizerBase {

  /** student (protected data) group name */
  public static final String STUDENT_GROUP_NAME = "apps:subjectSecurity:groups:student";
  /** privileged employee group name */
```

```
  public static final String PRIVILEGED_EMPLOYEE_GROUP_NAME = "apps:subjectSecurity:groups:privilegedEmployee";

  /** source id we care about */
  private static final String SOURCE_ID = "jdbc";

  /**
   * @see SubjectCustomizer#filterSubjects(GrouperSession, Set, String)
   */
  @Override
  public Set<Subject> filterSubjects(GrouperSession grouperSession, Set<Subject> subjects, String
findSubjectsInStemName) {

    //nothing to do if no results
    if (GrouperUtil.length(subjects) == 0) {
      return subjects;
    }

    //get results in one query
    MembershipResult groupMembershipResult = new MembershipFinder().assignCheckSecurity(false).addGroup
(STUDENT_GROUP_NAME)
        .addGroup(PRIVILEGED_EMPLOYEE_GROUP_NAME).addSubjects(subjects).addSubject(grouperSession.getSubject())
        .findMembershipResult();

    //see if the user is privileged
    boolean grouperSessionIsPrivileged = groupMembershipResult.hasGroupMembership
(PRIVILEGED_EMPLOYEE_GROUP_NAME, grouperSession.getSubject());

    //if so, we are done, they can see stuff
    if (grouperSessionIsPrivileged) {      return subjects;    }

    //loop through the subjects and see which are students, change their name and description to be their
netId, with no other attributes
    Set<Subject> results = new LinkedHashSet<Subject>();
    for (Subject subject : subjects) {
      if (StringUtils.equals(SOURCE_ID, subject.getSourceId()) && groupMembershipResult.hasGroupMembership
(STUDENT_GROUP_NAME, subject)) {
        String loginid = subject.getAttributeValue("loginid");
        Subject replacementSubject = new SubjectImpl(subject.getId(), loginid, loginid, subject.getTypeName(),
subject.getSourceId());
        results.add(replacementSubject);
      } else {
        results.add(subject);
      }
    }
    return results;
  }
}
```

Example: securely protect extra subject attributes based on Grouper permissions

This example has extra subject attributes (title and major) protected by Grouper permissions. This is an extra call, not populated by default, for performance reasons (note the WS examples above, they use this implementation)

```
package edu.internet2.middleware.grouper.subj.decoratorExamples;

import java.util.ArrayList;
import java.util.Collection;
import java.util.HashMap;
import java.util.HashSet;
import java.util.LinkedHashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;

import org.apache.commons.lang.StringUtils;

import edu.internet2.middleware.grouper.GrouperSession;
import edu.internet2.middleware.grouper.MembershipFinder;
```

```
import edu.internet2.middleware.grouper.Stem;
import edu.internet2.middleware.grouper.StemFinder;
import edu.internet2.middleware.grouper.Stem.Scope;
import edu.internet2.middleware.grouper.exception.GrouperSessionException;
import edu.internet2.middleware.grouper.hibernate.HibUtils;
import edu.internet2.middleware.grouper.hibernate.HibernateSession;
import edu.internet2.middleware.grouper.misc.GrouperSessionHandler;
import edu.internet2.middleware.grouper.permissions.PermissionFinder;
import edu.internet2.middleware.grouper.permissions.PermissionResult;
import edu.internet2.middleware.grouper.subj.SubjectCustomizer;
import edu.internet2.middleware.grouper.subj.SubjectCustomizerBase;
import edu.internet2.middleware.grouper.util.GrouperUtil;
import edu.internet2.middleware.subject.Subject;
import edu.internet2.middleware.subject.provider.SubjectImpl;

/**
 * add attributes securely to the subject
 * @author mchyzer
 *
 */
public class SubjectCustomizerForDecoratorExtraAttributes extends SubjectCustomizerBase {

  /** stem name of the permission resources which represent columns in the attribute table */
  public static final String PERMISSIONS_STEM_NAME = "subjectAttributes:permissions:columnNames";

  /** privileged employee group name */
  public static final String PRIVILEGED_ADMIN_GROUP_NAME = "etc:privilegedAdmin";

  /** source id we care about */
  private static final String SOURCE_ID = "jdbc";

  /** subjectAttributes:permissions */
  public static final String SUBJECT_ATTRIBUTES_PERMISSIONS_ATTRIBUTE_DEF = "subjectAttributes:permissions";


  /**
   * @see SubjectCustomizer#decorateSubjects(GrouperSession, Set, Collection)
   */
  @Override
  public Set<Subject> decorateSubjects(GrouperSession grouperSession,
      final Set<Subject> subjects, final Collection<String> attributeNamesRequested) {

    //nothing to do if no results or no attributes
    if (GrouperUtil.length(subjects) == 0 || GrouperUtil.length(attributeNamesRequested) == 0) {
      return subjects;
    }

    final Subject subjectChecking = grouperSession.getSubject();

    //do all this as admin
    GrouperSession.callbackGrouperSession(grouperSession.internal_getRootSession(), new GrouperSessionHandler()
{

      public Object callback(GrouperSession theGrouperSession) throws GrouperSessionException {

        //see if admin
        boolean isAdmin = new MembershipFinder().assignCheckSecurity(false).addGroup
(PRIVILEGED_ADMIN_GROUP_NAME)
          .addSubject(subjectChecking).hasMembership();

        //see which attributes the user has access to based on permissions
        PermissionResult permissionResult = null;

        //only need to check permissions if not admin
        if (!isAdmin) {

          Stem permissionsStem = StemFinder.findByName(theGrouperSession, PERMISSIONS_STEM_NAME, true);

          permissionResult = new PermissionFinder().addAction("read").addPermissionDef
(SUBJECT_ATTRIBUTES_PERMISSIONS_ATTRIBUTE_DEF)
            .addSubject(subjectChecking).assignPermissionNameFolder(permissionsStem)
```

```
            .assignPermissionNameFolderScope(Scope.ONE).findPermissionResult();

        }

        //see which columns the user can see
        Set<String> columnsSet = isAdmin ? new HashSet<String>(attributeNamesRequested)
            : permissionResult.permissionNameExtensions();

        //intersect the columns the user can see with the ones requested
        columnsSet.retainAll(attributeNamesRequested);

        if (GrouperUtil.length(columnsSet) == 0) {
          return null;
        }

        List<String> columns = new ArrayList<String>(columnsSet);

        List<Subject> subjectList = new ArrayList<Subject>(subjects);

        int batchSize = 180;
        int numberOfBatches = GrouperUtil.batchNumberOfBatches(subjectList, batchSize);

        subjects.clear();

        //batch these since you dont know if there will be more than 180 to resolve
        for (int i=0; i<numberOfBatches; i++) {

          List<Subject> subjectsBatch = GrouperUtil.batchList(subjectList, batchSize, i);

          //get the list of subject ids
          Set<String> subjectIds = new LinkedHashSet<String>();
          for (Subject subject : subjectsBatch) {
            if (StringUtils.equals(SOURCE_ID, subject.getSourceId())) {
              subjectIds.add(subject.getId());
            }
          }

          //get the results of these columns for these subjects (by id)
          //make query
          StringBuilder sql = new StringBuilder("select subject_id, ");
          sql.append(GrouperUtil.join(columns.iterator(), ','));
          sql.append(" from testgrouper_subj_attr where subject_id in( ");
          sql.append(HibUtils.convertToInClauseForSqlStatic(subjectIds));
          sql.append(")");

          //get the results from the DB
          List<String[]> dbResults = HibernateSession.bySqlStatic().listSelect(String[].class, sql.toString(),
GrouperUtil.toListObject(subjectIds.toArray()));

          //index the results by id of row
          Map<String, String[]> dbResultLookup = new HashMap<String, String[]>();

          Set<String> subjectIdsRelated = new HashSet<String>();

          for(String[] row : dbResults) {
            dbResultLookup.put(row[0], row);
            subjectIdsRelated.add(row[0]);
          }

          {
            //copy over the subjects since we need new objects...
            for (Subject subject : subjectsBatch) {

              //if it is a subject impl or doesnt have new attributes, we are all good...
              if (StringUtils.equals("jdbc", subject.getSourceId()) && subjectIdsRelated.contains(subject.
getId())) {

                if (!(subject instanceof SubjectImpl)) {
                  subject = new SubjectImpl(subject.getId(), subject.getName(),
                      subject.getDescription(), subject.getTypeName(), subject.getSourceId(),
                      subject.getAttributes(false));
```

```
                }
                String[] row = dbResultLookup.get(subject.getId());
                //shouldnt be null at this point
                if (row != null) {
                  //look through the attributes
                  for (int j=0;j<columns.size();j++) {

                    //add one to row index since first is id.  add if null or not, we need the attribute set
                    String columnName = columns.get(j);
                    String value = row[j+1];

                    //this should return a modifiable map of attributes for us to work with
                    subject.getAttributes(false).put(columnName, GrouperUtil.toSet(value));
                  }
                }
              }

              subjects.add(subject);

            }

          }

        }
        return null;
      }
    });

    return subjects;
  }

}
```