Grouper resource or permission picker

Starting with Grouper v2.4, the Lite UI is no longer available in the standard setup.

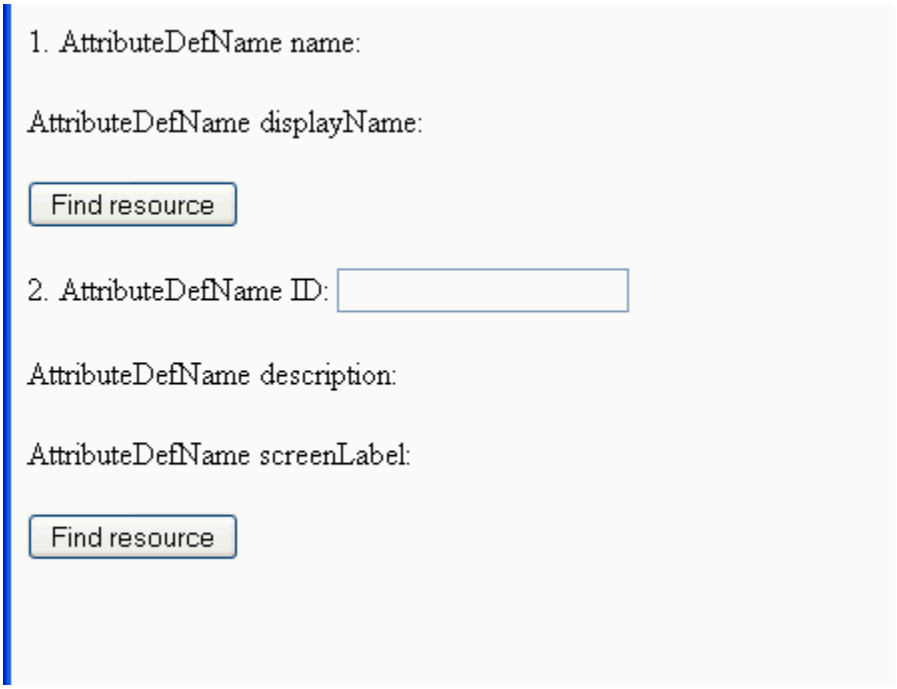
Resource or Permission Picker

Grouper LITE UIs

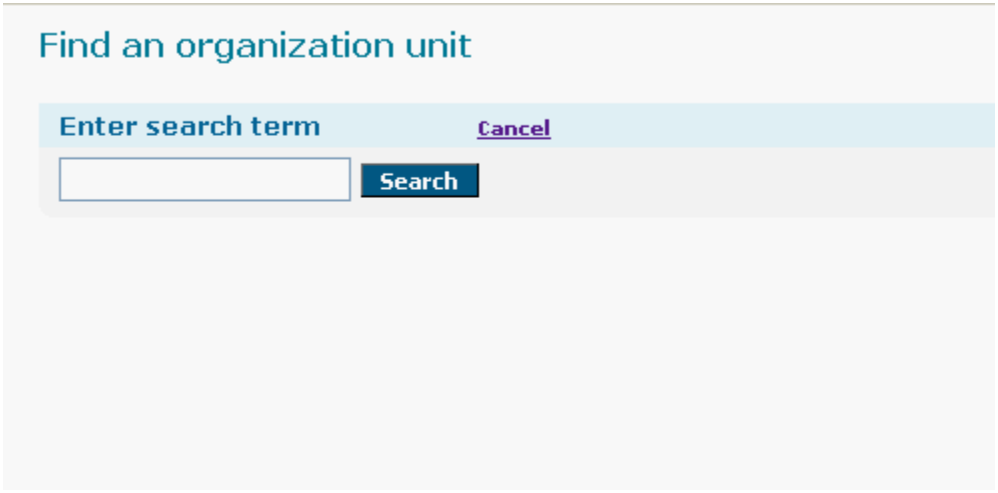
The Grouper attributeDefName resource picker is a lightweight embeddable UI component which can help external applications (or Grouper itself) to find attributes, resources, or permissions to put into textfields or hidden fields or drop downs or whatever. This is available for Grouper 1.6+. Here is a demo

Example

Here is a simple app screen. The URL says grouper, but it could be any URL.

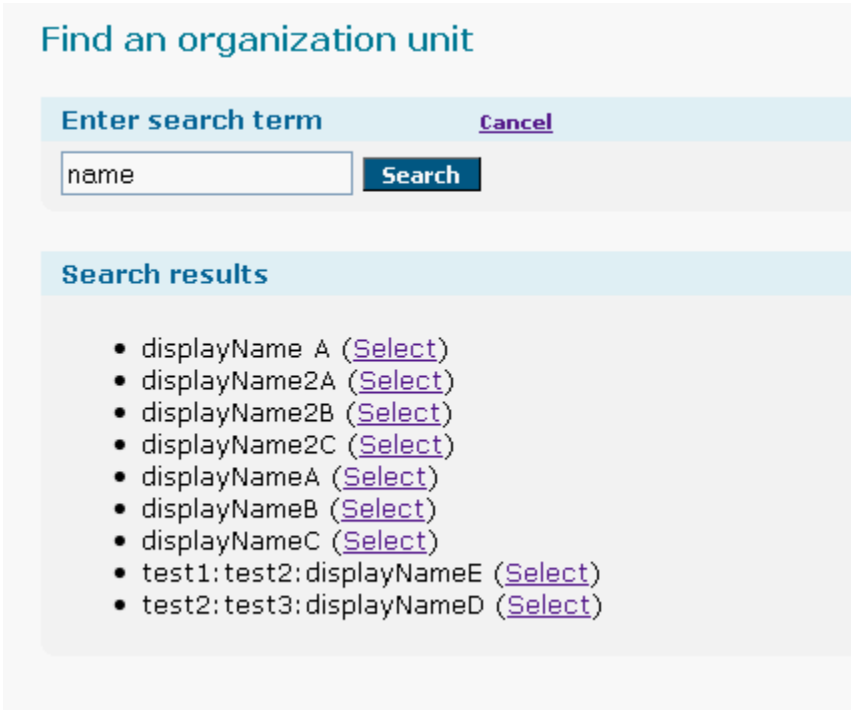


There are two fields which need resources selected (could be attributes, resources, permissions, attirbuteDefNames, etc), one just goes to a label on the screen, one to a textfield. When clicking on find resource, this screen appears as a popup



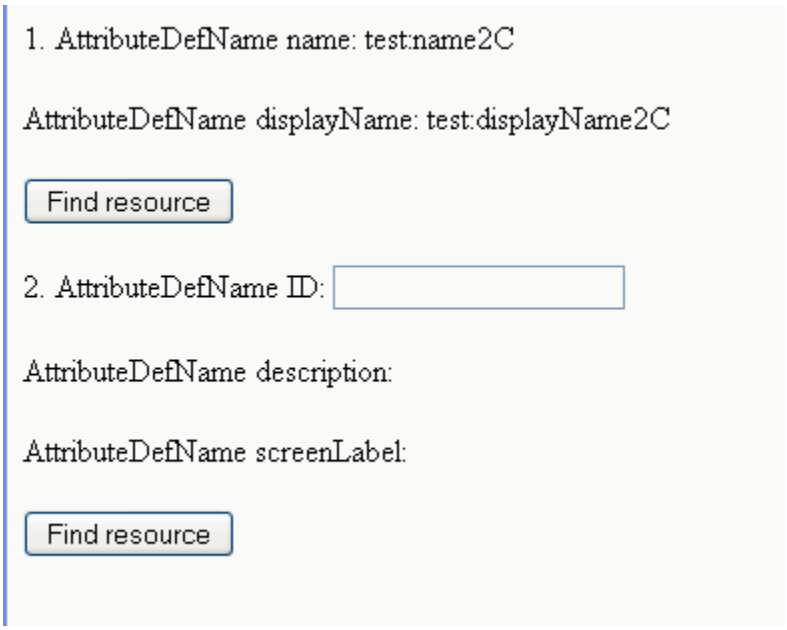
This screen above is hosted in the Grouper UI. It is assumed there is singlesign on between the two applications for good usability. If there isnt, then the user would need to login to Grouper (I dont think the resource picker can be used anonymously). Since it is a popup there should be fewer problems with cross site browser plugin blockers (e.g. for ajax)

Now a resource can be searched for. Its results will populate with ajax. Note: there is a resource picker "name" that links the picker with a config file. That config file specifies which attribute definitions are to be used for searching, and other configs, css to use, customizable text, etc



Now another search can take place, or one of the resources can be selected. If the resource is selected, then the window will close, and the calling window will get a javascript call with the resource chosen (the label, the id, and the name, display name, description of the resource. The calling page can easily take that and put it in a hidden field, a textfield, a label, etc with a little javascript. We can give examples of any javascript people will need.

So the bottom line is that with some Javascript, and some configuring of properties files in the Grouper UI, any web application can have a customized resource picker. Here is the calling page with the subject selected



Default settings

Resource pickers (you can have multiple at your site, customized for each application), have a lot of settings which are available. So to avoid having to set common settings in each resource picker instance, there are default settings. In the media.properties you can configure defaults for any of the resource picker settings. Note, since this is the media.properties, Grouper has defaults built in, and you can override those defaults:

```
##################################
## Attribute def name picker

# if putting config files on file system and not classpath, then put the files here
attributeDefNamePicker.confDir =


## attributeDefName picker config defaults

#comma separated css urls (relative or absolute) for skinning this attributeDefName picker
attributeDefNamePicker.defaultSettings.extraCss =

# names of attribute defs where the attribute def names should be searched from
attributeDefNamePicker.defaultSettings.searchInAttributeDefNames =

## You can configure per source how the attributeDefNames appear on screen, and customize per attributeDefName
picker instance as well
#this is the expression language of how the attributeDefName result should appear on screen
attributeDefNamePicker.defaultSettings.attributeDefNameNameEl =

# max results that can be retrieved
attributeDefNamePicker.defaultSettings.maxAttributeDefNamesResults = 200

# is an actas should be applied for group operations.  Generally this is GrouperSystem, though could be anyone,
or blank
# to act as the logged in user
attributeDefNamePicker.defaultSettings.actAsSourceId = g:isa
attributeDefNamePicker.defaultSettings.actAsSubjectId = GrouperSystem

# put a URL here where the result (attributeDefNameId, displayName, name, description) will be submitted back
# blank if same domain and just call opener directly
attributeDefNamePicker.defaultSettings.submitResultToUrl =
```

Then for each resource picker, assign a resource picker name, some alphanumeric or underscore that will tie the resource picker in the URL to the property file. Make a file in the directory above or on classpath: /attributeDefNamePicker/resourcePickerName.properties

This file can be blank, or could have any of the default settings above overridden:

```
##################################
## AttributeDefName picker
## http://localhost:8090/grouper/grouperUi/appHtml/grouper.html?operation=AttributeDefNamePicker.
index&attributeDefNamePickerName=attributeDefNamePickerExample&attributeDefNamePickerElementName=subject1

#comma separated css urls (relative or absolute) for skinning this attributeDefName picker
extraCss = ../public/assets/css/attributeDefNamePickerExample.css
#extraCss = http://localhost:8091/grouper/grouperUi/public/assets/css/subbjectPickerExample.css

# put attributeDefNames to search in
searchInAttributeDefNames = test:testAttributeDefName1Def, test:testAttributeDefName2Def

#this is the expression language of how the subject result should appear on screen
#attributeDefNameNameEl = ${attributeDefName.displayName}
attributeDefNameNameEl = ${pickerResultAttributeDefName.grandParentAndDisplayName}


# max results that can be retrieved
maxAttributeDefNamesResults = 800

# put a URL here where the result (attributeDefNameId, displayName, name, description) will be submitted back
# blank if same domain and just call opener directly
submitResultToUrl = http://localhost:8089/grouper/grouperUi/appHtml/attributeDefNamePickerTestSubmit.html

#if the resource is far down the folder structure, you can remove part of it
removePrefixOnUi = test:
```

Not only the settings can be configured, but also the text. This is in the nav.properties so it can in different languages or locales based on browser preferences. Also this means Grouper has defaults, institutions can override those defaults (globally), and also further customize per subjectPicker. Here are the defaults in the nav.properties:

```
#####################################
## AttributeDefName picker defaults
#####################################
attributeDefNamePickerDefault.header = Find a resource
attributeDefNamePickerDefault.title = Resource picker
attributeDefNamePickerDefault.searchSectionTitle = Enter search term
attributeDefNamePickerDefault.searchButtonText = Search

attributeDefNamePickerDefault.resultsSectionTitle = Search results

attributeDefNamePickerDefault.noSearchTerm = Enter a search term

attributeDefNamePickerDefault.noResultsFound = No results found

attributeDefNamePickerDefault.tooManyResults = Too many results, please narrow your search.  Note, a partial
listing might still display.

attributeDefNamePickerDefault.cancelText = Cancel
```

Based on the attributeDefName picker name, you can customize per resource picker in your local nav.properties

```
#####################################
## AttributeDefName picker test for attributeDefName picker with name attributeDefNamePickerExample
#####################################

attributeDefNamePicker.attributeDefNamePickerExample.title = Resource picker
```

Note that the same rules apply in the nav.properties as other apps, you can use the same syntax to make infodot help icons appear

Using the Picker

Once you have the resource picker configured in the Grouper UI, you can use it from an application. Note, with this setup with Grouper as an external application, there is no way to tell that the user is coming from the application you think it is coming from. This is one reason why authentication is required, and you can clamp down on who can use the service (e.g. active members of the community). If we want more security we could pass tokens with web services or something. In the HTML of the web app that needs a resource picker, make a button with an onclick that opens the picker. You also need to pass in the element name on the screen which had its button pressed (since multiple elements could need a resource picker).

```
<button onclick="theWindow = window.open('http://localhost:8091/grouper/grouperUi/appHtml/grouper.html?
operation=AttributeDefNamePicker.
index&attributeDefNamePickerName=attributeDefNamePickerExample&attributeDefNamePickerElementName=attributeDefNam
e2','newWindow', 'scrollbars,resizable,width=600,height=500'); theWindow.focus(); return false;"
>Find resource</button>
```

This will popup the resource picker. Note this is not a modal popup, it is assumed the user will use it or close it. If they forget it and use it later, that is another reason the element name is returned (so it doesnt mangle another screen). Anyways, have a javascript in the calling page for the callback for when a resource in the popup is selected. Here is an example that handles two resource pickers on one page, and escapes HTML on the screen

```
<script>

      function grouperAttributeDefNameSelected(elementName,attributeDefNameId,screenLabel,
          attributeDefNameName, attributeDefNameDisplayName, attributeDefNameDescription) {

        screenLabel = escapeHtml(screenLabel);
        attributeDefNameId = escapeHtml(attributeDefNameId);
        attributeDefNameName = escapeHtml(attributeDefNameName);
        attributeDefNameDisplayName = escapeHtml(attributeDefNameDisplayName);
        attributeDefNameDescription = escapeHtml(attributeDefNameDescription);

        var theElement = document.getElementById(elementName + "IdSpanId");
        if (theElement.value) {
          theElement.value = attributeDefNameId;
        }
        if (theElement.innerHTML) {
          theElement.innerHTML = screenLabel;
        }

      }

      /** convert input into a non-null string */
      function escapeHtml(input) {
        if (!input) {
          return input;
        }
        input = input.replace(/&/g,  "&amp;");
        input = input.replace(/</g,  "&lt;");
        input = input.replace(/>/g,  "&gt;");
        return input;
      }
    </script>
```

Note, you could use a library like Jquery to make this easier. Also, it is nice if the elements you are putting the resources into have id's (not just names), since it is more browser neutral (IE has issues with names).

Algorithm

There is a max number of resources to display, which is configurable. Regardless if you search for a resource's extension or display extension, that result will be at the top. So if the extension is "a", and that throws an exception in the source for too many results, the resource with extension "a" will still be in the results to choose. The results are sorted by display name.

Any exact matches of extension, and displayExtension will be brought to the top fo the list and not counted in the max count.

Security

Note, there are browser security restrictions that prohibit applications with different domain names or ports from accessing other windows of the browser. In this case the resource picker is a popup, and it needs to call a javascript function in its "opener". So if Grouper and the application have different domain names or ports, then you need to workaround this. Set this in the settings to an HTML file which resides in the application:# put a URL here where the result (attributeDefNameId, screenLabel, displayName, name, description) will be submitted back

1. blank if same domain and just call opener directly
   submitResultToUrl =

That HTML page (or servlet or whatever) will be submitted to with an HTTP GET with the results of the picker. An example of an HTML page is attached, it just gets the args from the URL, and calls the opener function which will work since it is the same domain and port, and closes itself.

To do

- Should add paging here.

asdf