Queries for Biostats integration

Vocabulary

Resource = a record of a "thing" in eagle-i. A resource is an instance of a class defined in the eagle-i ontology (e.g. "My nifty DNA Microarray" is an instance of the class "DNA MicroArray", which is a subclass of "Instrument")

A note of formats

See the MIME Type Section in the eagle-i repo API guide for the different formats that are supported by the SPARQL endpoint (all our queries are SELECT, so probably the best is SPARQL Result Set, which is XML representing a table)

Configuration

Organization Identifier

The organization to which biostats tools are attached in the eagle-i repository is:

http://harvard.eagle-i.net/i/0000012e-5946-2efe-55da-381e80000000

The same URI can be used in the dev environment, by changing the hostname as in the config property below.

Config File

```
# General
# What general type of resources we are dealing with

baseType=http://purl.obolibrary.org/obo/ERO_0000071

# What organization are the resources "attached" to (all resources in eagle-i need to belong to an organization)

organization=http://harvard.eagle-i.net/i/0000012e-5946-2efe-55da-381e80000000

# What properties to show in the overview record
# Literal properties (i.e. properties whose value is a simple string)
# Website
lp1=http://purl.obolibrary.org/obo/ERO_0000480
# Object properties (i.e. properties whose value is a link to another resource or ontology term)
# Not used for Biostats
# op1=
# op2=

#################
# Filters
#################
# Filter Group titles - for UI (do not grab these labels from ontology, they're usually not what you need)
fg1="Software Purpose"
fg2="Study Design"
fg3="Algorithm"
fg4="Data Type"
fg5="Measurement Scale"

# Properties for creating filters
# Each property corresponds to a filter group
# Direct filter properties
#has software purpose
fp1=http://purl.obolibrary.org/obo/ERO_0000078
#has related Study Design
fp2=http://eagle-i.org/ont/app/1.0/has_related_study_design
#Algorithm used
fp3=http://www.ebi.ac.uk/efo/swo/SWO_0000740

#Indirect filter properties (the property hangs off an embedded class) - need embedded property and indirect
property
#has data input
fp4=http://purl.obolibrary.org/obo/ERO_0000076
#data type
ip4=http://www.w3.org/1999/02/22-rdf-syntax-ns#type
#has data input
fp5=http://purl.obolibrary.org/obo/ERO_0000076
#has measurement scale
ip5=http://eagle-i.org/ont/app/1.0/has_measurement_scale
```

More advanced version with open-ended lists, potentially using yaml (but need to get the syntax right 😉 , for example:

```
# What properties to show in the overview record
# Literal properties (i.e. properties whose value is a simple string)
literalProperties:
   propertyUri: http://purl.obolibrary.org/obo/ERO_0000480

# Object properties (i.e. properties whose value is a link to another resource or ontology term)
objectProperties:
  propertyUri: http://purl.obolibrary.org/obo/ERO_0001719
  propertyUri: http://purl.obolibrary.org/obo/ERO_0000078
```

Base queries

Get base, unfiltered dataset

Find all instances of `baseType` affiliated with `organization`, returns a table with a column per query variable defined in property configuration above

Need to split in two queries (one for Ontology labels, one for instances). Note that if desired props don't have preferred label we'll need to alter query 1

Query 1 => to be executed only once (labels don't change)

```
PREFIX eiapp: <http://eagle-i.org/ont/app/1.0/>
SELECT

?lp1Label
?op1Label
?op2Label

 WHERE {
     OPTIONAL {
          <${lp1}> eiapp:preferredLabel ?lp1Label .
          <${op1}> eiapp:preferredLabel ?op1Label .
          <${op2}> eiapp:preferredLabel ?op2Label .
     }
}
```

Query 2
Note that if a property is not defined in the config file (e.g. op1), the variable in the SELECT clause, as well as the entire triple pattern/pattern group in the WHERE clause (i.e. the corresponding line) needs to be omitted. So there need to be some conditional logic in the query generation.

```
PREFIX ei: <http://purl.obolibrary.org/obo/>
PREFIX eiapp: <http://eagle-i.org/ont/app/1.0/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT
?resourceUri
?resourceLabel
?resourceDescription
?lp1Value
?op1ValueLabel
?op1ValueUri
?op2ValueLabel
?op2ValueUri

 WHERE {
     ?resourceUri a <${baseType}> .
     ?resourceUri ei:ERO_0000070 <${organization}> .
     ?resourceUri rdfs:label ?resourceLabel .
     OPTIONAL {
          ?resourceUri eiapp:resource_description ?resourceDescription .
          ?resourceUri <${lp1}> ?lp1Value .
          ?resourceUri <${op1}> ?op1ValueUri . ?op1ValueUri rdfs:label ?op1ValueLabel .
          ?resourceUri <${op2}> ?op2ValueUri . ?op2ValueUri rdfs:label ?op2ValueLabel .
     }
}
```

Example with substituted values

```
PREFIX ei: <http://purl.obolibrary.org/obo/>
PREFIX eiapp: <http://eagle-i.org/ont/app/1.0/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT
?resourceUri
?resourceLabel
?resourceDescription
?lp1Label
?lp1Value

 WHERE {
     ?resourceUri a <http://purl.obolibrary.org/obo/ERO_0000071> .
     ?resourceUri ei:ERO_0000070 <http://harvard.dev.eagle-i.net/i/0000012e-5946-2efe-55da-381e80000000> .
     ?resourceUri rdfs:label ?resourceLabel .
     OPTIONAL {
          ?resourceUri eiapp:resource_description ?resourceDescription .
          ?resourceUri <http://purl.obolibrary.org/obo/ERO_0000480> ?lp1Value .
     }
}
```

Get detailed view of a resource

For the main resource, simply do a GET of its URI; the results can be in a variety of formats - see: eagle-i repo API guide

There is an additional parameter `forceXML`, which returns a format that may be the most useful (it's the XML from which the HTML version is generated) - pure RDF/XML is horrible

Example:

```
GET http://harvard.qa.eagle-i.net/i/0000012e-5480-32e6-55da-381e80000000?forceXML
```

After getting the main resource, the **embedded resources** need to be processed. Assume main resource has uri http://harvard.eagle-i.net/123

This query will return embedded resources:

```
PREFIX eiapp: <http://eagle-i.org/ont/app/1.0/>
SELECT
?embeddedResourceUri
WHERE {
<http://harvard.eagle-i.net/123> ?anyProperty ?embeddedResourceUri .
?embeddedResourceUri a ?type .
?type eiapp:inClassGroup eiapp:ClassGroup_EmbeddedResourceType .
}
```

To obtain the embedded resource's details, iterate over this list and do a GET as with the main resource.

Queries for filters

Get actual annotation values that have been used in the dataset

Do this for each direct filter property, e.g. fp1

```
PREFIX ei: <http://purl.obolibrary.org/obo/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT
?filterValueUri
?filterValueLabel

WHERE {
     ?resourceUri a <${baseType}> .
     ?resourceUri ei:ERO_0000070 <${organization}> .
     ?resource <${fp1}> ?filterValueUri .
     ?filterValueUri rdfs:label ?filterValueLabel .
}
```

Do this for each indirect filter property, e.g. ep4 and ip4

```
PREFIX ei: <http://purl.obolibrary.org/obo/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT
?filterValueUri
?filterValueLabel

WHERE {
?resourceUri a <${baseType}> .
?resourceUri ei:ERO_0000070 <${organization}> .
?resource <${ep4}> ?embeddedResource .
?embeddedResource <${ip4}> ?filterValueUri .
?filterValueUri rdfs:label ?filterValueLabel .
}
```

Get the filtered dataset

First part is identical to unfiltered query (remember to remove unset config params, e.g. op1), then conditions are added as triple patterns;
Assume filterValueUrix is the filter value selected by the user, for each filter property, e.g. filterValueUri1 for fp1
NOTE if only one filter is selected, add only that triple pattern/triple pattern group.

```
PREFIX ei: <http://purl.obolibrary.org/obo/>
PREFIX eiapp: <http://eagle-i.org/ont/app/1.0/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT
?resourceUri
?resourceLabel
?resourceDescription
?lp1Value
?op1ValueLabel
?op1ValueUri
?op2ValueLabel
?op2ValueUri

WHERE {
?resourceUri a <${baseType}> .
?resourceUri ei:ERO_0000070 <${organization}> .
?resourceUri rdfs:label ?resourceLabel .
OPTIONAL {
?resourceUri eiapp:resource_description ?resourceDescription .
?resourceUri <${lp1}> ?lp1Value .
?resourceUri <${op1}> ?op1ValueUri . ?op1ValueUri rdfs:label ?op1ValueLabel .
?resourceUri <${op2}> ?op2ValueUri . ?op2ValueUri rdfs:label ?op2ValueLabel .
}

?resourceUri <${fp1}> <${filterValueUri1}> .
?resourceUri <${fp2}> <${filterValueUri2}> .
?resourceUri <${fp3}> <${filterValueUri3}> .

?resourceUri <${ep4}> ?embeddedResource1 . ?embeddedResource1 <${ip4}> <${filterValueUri4}> .
?resourceUri <${ep5}> ?embeddedResource2 . ?embeddedResource2 <${ip5}> <${filterValueUri5}> .
}
```