METAREP Programmatic Access

Listing 1: List METAREP Projects

```
use METAREP::MetarepDao;
use strict;
use warnings;

## create METAREP data access object
my $metarep = new METAREP::MetarepDao();

## get METAREP projects
my $projects = $metarep->getProjects();

## print results
print "Project ID\t Project Name\n";
foreach my $project(@$projects) {
  print "$project->{id}\t$project->{name}\n";
}

## example output

Project ID       Project Name
1       Human Microbiome Project
2       Australian Soil
3       Marine Virioplankton
4       ILMPVHIS Soil libraries
5       Natalies euk test set
7       Indoor and Outdoor Air Quality
9       Geobacter
10      Yellowstone Lake
...
```

Listing 2: List METAREP Project Libraries and Library Meta-Data

```
use METAREP::MetarepDao;
use strict;
use warnings;

## create METAREP data access object
my $metarep = new METAREP::MetarepDao();

## get GOS Phase I libraries
my $libraries = $metarep->getProjectLibraries(19);

## print results
foreach my $library(@$libraries) {
  print "-------------------------------------------------------\n";
  print("Library ID:\t                    $library->{id}\n");
  print("Library Label:\t                 $library->{label}\n");
  print("Library Description:\t\t         $library->{description}\n");
  print("Sample Date:\t                   $library->{date}\n");
  print("Sample Longitude:\t              $library->{longitude}\n");
  print("Sample Latitude:\t               $library->{latitude}\n");
  print("Sample Habitat:\t\t              $library->{habitat}\n");
  print("Sample Altitude:\t               $library->{altitude}\n");
  print("Sample Depth:\t                  $library->{depth}\n");
  print("Sample Filter:\t                 $library->{filter}\n");
  print("Annotation Pipeline:\t\t         $library->{pipeline}\n");
}
```

```
## example output

...
-----------------------------------------------------
Library ID:                          GS-11-01-01-1P3-1P8KB
Library Label:                       NA
Library Description:                 Next to Nuclear Power Plant, Delaware Bay, NJ, USA
Sample Date:                         2003-11-18
Sample Longitude:                    39.426667
Sample Latitude:                      -75.504167
Sample Habitat:                      estuary
Sample Altitude:                     NA
Sample Depth:                        1
Sample Filter:                       0.1
Annotation Pipeline:                 prok
-----------------------------------------------------
...
```

Listing 3: Query METAREP dataset (COUNT)

```
use METAREP::MetarepDao;
use strict;
use warnings;

#########################################################
# Query Syntax: {field_name}:{value} see Table 1 for fields
# http://lucene.apache.org/java/2_3_2/queryparsersyntax.html
#
# SQL:     Select count(*) from dataset where field1=a AND field2=b
# METAREP: metarep->count(dataset,'field1:a AND field2:b')
#
# Default field: com_name_txt Operators: AND OR NOT
#########################################################

## create METAREP data access object
my $metarep = new METAREP::MetarepDao();

my $dataset = 'GS321-0p8um-DNA-fragment-quarter-MIDpool-FRHLGZN02';

## query examples
my $query1  = 'kinase';

## oxidoreductases in non-bacterial species (BLAST)
my $query2  = 'NOT blast_tree:2 AND ec_id:1.*';

## non bacterial oxidoreductases (APIS)
my $query3  = 'NOT apis_tree:2 AND ec_id:1.*';

## PFAM HMMs that overlap with final clusters
my $query4  = 'hmm_id:PF* AND cluster_id:CAM_CL_*';

## Proteobacteria with high confidence BLAST hits 10^-50 and lower
my $query5  = 'blast_tree:1224 AND blast_evalue_exp:{50 TO *}';

## execute count query
my $count = $metarep->count($dataset,$query1);

print "Found $count hits in $dataset dataset for '$query1'\n";

## example output

Found 3240 hits in GS321-0p8um-DNA-fragment-quarter-MIDpool-FRHLGZN02 dataset

## search multiple datasets
```

```
my $dataset1 = '03-GS114-G-1p6-2kb';
my $dataset2 = '02-GS114-G-2-3kb';
my $dataset3 = '01-GS122-G-4-6kb';

my @datasets = ($dataset1,$dataset2,$dataset3);

## use array ref to search multiple datasets
my $count = $metarep->count(\@datasets,$query1);
```

Listing 4: Query METAREP dataset (GROUP BY)

```
use METAREP::MetarepDao;
use strict;
use warnings;

#########################################################
# Query Syntax: {field_name}:{value} see Table 1 for fields
# http://lucene.apache.org/java/2_3_2/queryparsersyntax.html
#
# SQL:     select count(*) as c,category from dataset where
#          field1=a group by category having c >= x limit y
# METAREP: metarep->groupBy(dataset,'field1:a',category,x,y);
#
# Default field: com_name_txt Operators: AND OR NOT
#########################################################

## create METAREP data access object
my $metarep = new METAREP::MetarepDao();

my $dataset = 'GS321-0p8um-DNA-fragment-quarter-MIDpool-FRHLGZN02';

## list all entries with cluster_id:CAM_CL_399
my $query = 'cluster_id:CAM_CL_399';

## group by category examples

## group by BLAST species
my $groupBy1 = 'blast_species';

## group by enzyme IDs
my $groupBy2 = 'ec_id';

## group by common name
my $groupBy3 = 'com_name';

## specify number of groups to return
my $limit   = 20;

## specify the minimum count for a group by category
my $minCount = 1;

## optional prefix to filter the specified category
## counts for categories that start with the prefix
my $prefix = 'CAM_CL_';

## execute group by query (prefix can be omitted);
my $groupByCounts = $metarep->groupBy($dataset,$query,$groupBy1,$minCount,$limit,$prefix);

## sort group by result by descending count
my @uniqueValues = (sort { $groupByCounts->{$b} <=> $groupByCounts->{$a}} keys( %$groupByCounts));

## print results
```

Table 1: METAREP Search Fields

Field Name	Description	Type/Range	Example
Core Annotation Fields			
peptide_id	Peptide ID	text	**peptide_id:1120333534885** *retrieve hit with the specified peptide id*
com_name_txt	Common Name (default field)	text	**com_name_txt:phage** *all hits containing the word phage*
com_name_src	Common Name Source	text	**com_name_src:PRIAM** *all hits having names assigned based on PRIAM*
go_id	Gene Ontology ID	text	**go_id:GO:0000160** *all hits with GO:0000160*
go_tree	Gene Ontology Tree	integer portion of ID	**go_tree:160** *all hits with GO:0000160 or lower (including all hits with GO IDs that are lower (more specific) in the GO hierarchy)*
go_src	Gene Ontology Source	text	**go_src:PF00204** *all hits that have GO terms assigned based on PF00204*
ec_id	Enzyme ID	text	**ec_id:5.99.1.3** *all hits with Enzyme ID 5.99.1.3*
ec_src	Enzyme Source	text	**ec_src:PRIAM** *all hits that have EC IDs assigned based on PRIAM*
hmm_id	HMM ID	text	**hmm_id:PF00204** *all hits that have a PF00204 HMM assignment*
library_id	Library ID	text	**library_id:GS-00a-01-01-2P5KB** *all hits that belong to library GS-00a-01-01-2P5KB (helpful to search for library entries within populations)*
filter	any filter tag (e.g. sequence duplicates)	text	**filter:duplicate** *all hits with filter tagged with duplicate* **-filter:duplicate** *exclude entries with filter tag duplicate*
Best BLAST Hit Fields			
blast_species	Species	text	**blast_species:Chlamydia*** *all Chlamydia species*
blast_tree	Taxonomy	integer (NCBI Taxon ID)	**blast_tree:2** *all bacteria* **-blast_tree:2** *exclude all bacteria*
blast_evalue_exp	Negative E-Value Exponent	positive integer	**blast_evalue_exp:{20 TO *}** *all hits with Blast E-value* $\leq 10^{-20}$ **blast_evalue_exp:{10 TO 20}** *all hits with* $10^{-20} \leq E-value \leq 10^{-10}$
blast_pid	Percent Identity	float between 0-1	**blast_pid:{0.9 TO *}** *all hits with Blast percent identity* $\geq 90\%$ **blast_pid:{0.6 TO 0.8}** *all hits with* $60\% \leq percent\ identity \leq 80\%$
blast_cov	Percent Sequence Coverage	float between 0-1	**blast_cov:{0.8 TO *}** *all hits with Blast percent sequence coverage* $\geq 80\%$ **blast_cov:{0.2 TO 0.3}** *all hits with* $20\% \leq sequence\ coverage \leq 30\%$
Optional Fields			
apis_tree	Taxonomy	integer (NCBI Taxon ID)	**apis_tree:2** *all bacteria* **-apis_tree:2** *exclude all bacteria*
env_lib	Environmental Library	text	**env_lib:Whale*** *all hits that match the whale fall metagenome*
cluster_id	Cluster ID	text	**cluster_id:CAM_CL_399** *all hits that match with cluster CAM_CL_399*

```
print "Species (BLAST)\tCount\n";
foreach my $uniqeValue(@uniqueValues) {
  print $uniqeValue."\t".$groupByCounts->{$uniqeValue} . "\n" ;
}

## example output (species breakdown for cluster CAM_CL_399)

Species (BLAST) Count
unresolved      103
Ruegeria sp. TM1040     32
Hyphomonas neptunium    21
Pseudoalteromonas atlantica     18
Parabacteroides distasonis      16
Colwellia psychrerythraea       15
Shewanella woodyi       15
Parvibaculum lavamentivorans    14
Ruegeria pomeroyi       12
Bacteroides thetaiotaomicron    10
unassigned      8
Bradyrhizobium japonicum        8
Roseobacter denitrificans       8
Bacteroides vulgatus    7
Aspergillus niger       6
Bacteroides fragilis    6
Jannaschia sp. CCS1     6
Sinorhizobium meliloti  6
Sinorhizobium medicae   5
Nematostella vectensis  5


## group by search using multiple datasets

my $dataset1 = '03-GS114-G-1p6-2kb';
my $dataset2 = '02-GS114-G-2-3kb';
my $dataset3 = '01-GS122-G-4-6kb';

my @datasets = ($dataset1,$dataset2,$dataset3);

## use array ref to search multiple datasets
my $groupByCounts = $metarep->groupBy(\@datasets,$query,$groupBy1,$minCount,$limit,$prefix);
```

Listing 5: Query METAREP dataset (SELECT)

```
use METAREP::MetarepDao;
use strict;
use warnings;

#########################################################
# Query Syntax: {field_name}:{value} see Table 1 for fields
# http://lucene.apache.org/java/2_3_2/queryparsersyntax.html
#
# SQL:      select * from dataset where field1=a and field2=b
#           limit x,y
# METAREP:  metarep->select(dataset,'field1:a AND field2:b',x,y);
#
# Default field: com_name_txt Operators: AND OR NOT
#########################################################

## create METAREP data access object
my $metarep = new METAREP::MetarepDao();

my $dataset = 'GS327-3p0um-DNA-fragment-quarter-MIDpool-FZWZ24A02';

## fetch all non-bacterial hits with cluster_id:CAM_CL_399
```

```
my $query1 = 'NOT blast_tree:2 AND cluster_id:CAM_CL_399';

## fetch entry for peptide JCVI_PEP_metagenomic.orf.1120624357994.1
my $query2 = 'peptide_id:JCVI_PEP_metagenomic.orf.1120624357994.1';

## specify the row number to start selecting data
my $start = 0;

## specify how many results to return
my $rows = 10;

## execute select query
my $rows = $metarep->select($dataset,$query2,$start,$rows);

## delimiter to use for fields with multiple values
my $multValDlm = '||';

## print results
for my $row(@$rows) {
  print "-----------------------------------------------------\n";
  print "Peptide ID:\t".      $row->{peptide_id}."\n";
  print "Common Name(s):\t".  join($multValDlm,@{$row->{com_name}}) ."\n";
  print "Common Name Source(s):\t".join($multValDlm,@{$row->{com_name_src}})."\n";
  print "EC ID(s):\t".        join($multValDlm,@{$row->{ec_id}})."\n";
  print "EC Source(s):\t".    join($multValDlm,@{$row->{ec_src}})."\n";
  print "GO ID(s):\t".        join($multValDlm,@{$row->{go_id}})."\n";
  print "GO Source(s):\t".    join($multValDlm,@{$row->{go_src}})."\n";
  print "HMM ID(s):\t".       join($multValDlm,@{$row->{hmm_id}})."\n";
  print "Cluster ID(s):\t".   join($multValDlm,@{$row->{cluster_id}})."\n";
  print "Filter:\t".          join($multValDlm,@{$row->{filter}})."\n";
  print "BLAST species:\t".   $row->{blast_species}."\n";
  print "BLAST E-Value:\t".   $row->{blast_evalue}."\n";
}

## example output

-----------------------------------------------------
Peptide ID:     JCVI_PEP_metagenomic.orf.1120624380876.1
Common Name(s): N-acetylgalactosamine 6-sulfate sulfatase (galns)
Common Name Source(s):  RF|NP_867447.1|32474453|NC_005027
EC ID(s):       3.1.6.4
EC Source(s):   RF|NP_867447.1|32474453|NC_005027
GO ID(s):       unassigned
GO Source(s):   unassigned
HMM ID(s):      unassigned
Cluster ID(s):  CAM_CRCL_55837||CAM_CL_399
Filter:
BLAST species:  unresolved
BLAST E-Value:  4.68245E-21
-----------------------------------------------------
Peptide ID:     JCVI_PEP_metagenomic.orf.1120624398456.1
Common Name(s): sulfatase||Steryl-sulfatase
Common Name Source(s):  RF|YP_614107.1|99081953|NC_008044||PRIAM
EC ID(s):       3.1.6.2
EC Source(s):   PRIAM
GO ID(s):       unassigned
GO Source(s):   unassigned
HMM ID(s):      unassigned
Cluster ID(s):  CAM_CRCL_47481||CAM_CL_399
Filter:
BLAST species:  Ruegeria sp. TM1040
BLAST E-Value:  9.32648E-46
-----------------------------------------------------
Peptide ID:     JCVI_PEP_metagenomic.orf.1120624404688.1
Common Name(s): N-acetylgalactosamine 6-sulfatase (galns)||Steryl-sulfatase
Common Name Source(s):  RF|NP_866352.1|32473358|NC_005027||PRIAM
```

```
EC ID(s):        3.1.6.2
EC Source(s):    PRIAM
GO ID(s):        unassigned
GO Source(s):    unassigned
HMM ID(s):       unassigned
Cluster ID(s):   CAM_CRCL_4012959||CAM_CL_399
Filter:
BLAST species:   unresolved
BLAST E-Value:   2.82248E-26

## select using multiple datasets

my $dataset1 = '03-GS114-G-1p6-2kb';
my $dataset2 = '02-GS114-G-2-3kb';
my $dataset3 = '01-GS122-G-4-6kb';

my @datasets = ($dataset1,$dataset2,$dataset3);

## use array ref to select from multiple datasets
my $rows = $metarep->select(\@datasets,$query2,$start,$rows);
```

Listing 6: METAREP Programmatic Access Exception

```
#########################################################
# Example METAREP Programmatic Access Exception
#
# The first section describes the request parameters including
# the method and query that was executed as well as dataset(s)
# used. The second section, shows the response information
# including HTTP status and error message and HTML body. Here
# the query 'kina:se' contained a Lucene special character  ':'
# that is used to separate fields from values. As there is no
# field kina, it throws the exception shown below.
#########################################################


METAREP Programmatic Access Exception
Please check the method, dataset and query:

-----------------------------------------------
REQUEST
METHOD:          COUNT
DATASET(S):      03-GS114-G-1p6-2kb,02-GS114-G-2-3kb,01-GS122-G-4-6kb
QUERY:           kina:se
-----------------------------------------------
RESPONSE
STATUS:          400
MESSAGE:         undefined_field_kina
STATUS:          400
MESSAGE:         undefined_field_kina
CONTENT:

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1"/> <title>Error 400 </title> </head>
    <body><h2>HTTP ERROR: 400</h2><pre>undefined field kina</pre>
<p>RequestURI=/solr/03-GS114-G-1p6-2kb/select</p><p><i><small><a href="http://jetty.mortbay.org/">Powered by
     Jetty://</a></small></i></p><br/>
<br/>
...
</body>
</html>
```

Listing 7: Get Project Populations

```
use METAREP::MetarepDao;
use strict;
use warnings;

## create METAREP data access object
my $metarep = new METAREP::MetarepDao();

## get GOS Phase I populations
my $populations = $metarep->getProjectPopulations(19);

## print results
print "Population ID\tPopulation Description\tPipeline\n";
foreach my $population(@$populations) {
  print "$population->{id}\t$population->{description}\t$population->{pipeline}\n";
}

## example output

Population ID   Population Description  Pipeline
gos-phase-I-sanger      All GOS Phase I Sanger Libraries        prok
..
```

Listing 8: List Population Libraries and Library Meta-Data

```
use METAREP::MetarepDao;
use strict;
use warnings;

## create METAREP data access object
my $metarep = new METAREP::MetarepDao();

my $libraries = $metarep->getPopulationLibraries('gos-phase-I-sanger');

## print results
foreach my $library(@$libraries) {
  print "-----------------------------------------------------\n";
  print("Library ID:\t                    $library->{id}\n");
  print("Library Label:\t                 $library->{label}\n");
  print("Library Description:\t\t         $library->{description}\n");
  print("Sample Date:\t                   $library->{date}\n");
  print("Sample Longitude:\t              $library->{longitude}\n");
  print("Sample Latitude:\t               $library->{latitude}\n");
  print("Sample Habitat:\t\t              $library->{habitat}\n");
  print("Sample Altitude:\t               $library->{altitude}\n");
  print("Sample Depth:\t                  $library->{depth}\n");
  print("Sample Filter:\t                 $library->{filter}\n");
  print("Annotation Pipeline:\t\t         $library->{pipeline}\n");
}

## example output

-----------------------------------------------------
Library ID:                             GS-25-01-01-1P8-2P2KB
Library Label:                          NA
Library Description:                    Dirty Rock, Cocos Island, Costa Rica
Sample Date:                            2004-01-28
Sample Longitude:                       5.5
Sample Latitude:                         -87.066667
Sample Habitat:                         fringing reef
Sample Altitude:                        NA
Sample Depth:                           0.98
Sample Filter:                          0.8
```

```
Annotation Pipeline:                    prok
-------------------------------------------------------
Library ID:                             GS-11-01-01-1P3-1P8KB
Library Label:                          NA
Library Description:                    Next to Nuclear Power Plant, Delaware Bay, NJ, USA
Sample Date:                            2003-11-18
Sample Longitude:                       39.426667
Sample Latitude:                         -75.504167
Sample Habitat:                         estuary
Sample Altitude:                        NA
Sample Depth:                           1
Sample Filter:                          0.1
Annotation Pipeline:                    prok
...
```