MyGeneset.info Documentation

Release 1.0

The Su, Wu, and Greene Labs

Apr 27, 2021

CONTENTS

CHAPTER

ONE

INTRODUCTION

MyGeneset.info provides simple-to-use REST web services to query, retrieve, and store annotation data for collections of genes. It's designed with an emphasis on **simplicity** and **performance**.

1.1 Features

- Access curated genesets from public databases, including: GO, KEGG, MSigDB, Disease Ontology/OMIM, Wikipathways, and Reactome.
- Filter genesets to an organism of interest.
- Retrieve gene collections with an identifier of choice.

CHAPTER

TWO

QUICK START

MyGeneset.info provides two simple web services: one for querying stored geneset objects and the other for geneset annotation retrieval. Both return results in JSON format.

2.1 Geneset query service

2.1.1 URL

```
http://mygeneset.info/v1/query
```

2.1.2 Examples

```
# Search genesets with 'glucose' in a default query field (geneset name or␣
↪description)
http://mygeneset.info/v1/query?q=glucose

# Retreive NCBI gene ids for genes in genesets with 'glucose' in a default query field
http://mygeneset.info/v1/query?q=glucose&fields=genes.ncbigene

# Fetch all genesets containing the gene with symbol ABL1
http://mygeneset.info/v1/query?q=genes.symbol:ABL1

# Filter the previous query to human species
http://mygeneset.info/v1/query?q=genes.symbol:ABL1&species=human

# Fetch all genesets belonging to two mice or human organisms
http://mygeneset.info/v1/query?species=mouse,human

# Fetch all genesets from the source "wikipathways"
http://mygeneset.info/v1/query?q=source:wikipathways

# Wildcard queries
http://mygeneset.info/v1/query?q=genes.symbol=cdk?
http://mygeneset.info/v1/query?q=genes.symbol=IL*

# Search genesets containing all of the provided genes
http://localhost:8000/v1/query?q=genes.symbol:(ABL1 AND CEBPA AND FLT3)&fields=all

# Search genesets containing any of the provided genes
http://localhost:8000/v1/query?q=genes.symbol:(ABL1 OR CEBPA OR FLT3)&fields=all
```

Hint: View nicely formatted JSON results in your browser with this handy add-on: JSON formater for Chrome or JSONView for Firefox.

2.1.3 To learn more

- You can read the full description of our query syntax here.
- Try it live on interactive API page.
- Batch queries? Yes, you can. do it with a POST request.

2.2 Geneset annotation service

2.2.1 URL

```
http://mygeneset.info/v1/geneset/<genesetid>
```

2.2.2 Examples

```
http://mygeneset.info/v1/geneset/WP100
http://mygeneset.info/v1/geneset/GO_0000002_9606
http://mygeneset.info/v1/geneset/R-HSA-112043?fields=genes.ensemblgene,reactome,taxid
```

2.2.3 To learn more

- You can read the full description of our query syntax here.
- Try it live on interactive API page.
- Yes, batch queries via POST request as well.

CHAPTER

THREE

DOCUMENTATION

3.1 Geneset annotation data

3.1.1 Data sources

We currently obtain the geneset annotation data from several public data resources and keep them up-to-date, so that you don't have to do it:

Source	Update frequency
MSigDB	whenever a new release is available
Gene Ontology	whenever a new release is available
KEGG	whenever a new release is available
WikiPathways	whenever a new release is available
Reactome	whenever a new release is available
Disease Ontology	whenever a new release is available

The most updated data information can be accessed here.

3.1.2 Geneset object

Geneset annotation data are both stored and returned as a geneset object, which is essentially a collection of fields (attributes) and their values:

```
{
  "_id": "WP60",
  "_version": 1,
  "genes": [
    {
      "ensemblgene": "YOL165C",
      "mygene_id": "853999",
      "name": "putative aryl-alcohol dehydrogenase",
      "ncbigene": "853999",
      "symbol": "AAD15",
      "uniprot": "Q08361"
    },
    {
      "mygene_id": "850488",
      "name": "pseudo",
      "ncbigene": "850488",
      "symbol": "AAD6"
```

(continues on next page)

(continued from previous page)

```
    },
    {
      "ensemblgene": "YNL331C",
      "mygene_id": "855385",
      "name": "putative aryl-alcohol dehydrogenase",
      "ncbigene": "855385",
      "symbol": "AAD14",
      "uniprot": "P42884"
    },
    {
      "ensemblgene": "YCR107W",
      "mygene_id": "850471",
      "name": "putative aryl-alcohol dehydrogenase",
      "ncbigene": "850471",
      "symbol": "AAD3",
      "uniprot": "P25612"
    },
    {
      "ensemblgene": "YJR155W",
      "mygene_id": "853620",
      "name": "putative aryl-alcohol dehydrogenase",
      "ncbigene": "853620",
      "symbol": "AAD10",
      "uniprot": "P47182"
    },
    {
      "ensemblgene": "YDL243C",
      "mygene_id": "851354",
      "name": "putative aryl-alcohol dehydrogenase",
      "ncbigene": "851354",
      "symbol": "AAD4",
      "uniprot": "Q07747"
    }
  ],
  "is_public": true,
  "name": "Toluene degradation",
  "source": "wikipathways",
  "taxid": 559292,
}
```

The example above contains the most common available fields, but omits some fields that are specific to specific data sources. For a full example, you can check out a few examples: GO_0004568_9606 (a Gene Ontology geneset), WP60 (a Wikipathways geneset), or find a list of the available fields at: http://mygeneset.info/v1/metadata/fields

3.1.3 _id field

Each individual geneset object contains an "**_id**" field as the primary key. The value of the "**_id**" field is different for every built-in data source, but is typically the primary ID used in the source data. For example, for MSigDB, this is the original geneset id. For genesets coming from metabolic pathway databases (KEGG, GO, Wikipathways) which contain multiple species, **_id** is typically a combination of the pathway id, plus the organism taxid. User-submitted genesets have randomly generated **_id** fields. Here is an example. If searching for a particular GO term, or KEGG ID using the query endpoint, we recommend using "**kegg.id**"or "**go.id**", plus the species filter instead of "**_id**".

Note: Regardless how the value of the "**_id**" field looks like, it always works for our geneset annotation service

/v1/geneset/<geneid>.

3.1.4 _score field

You will often see a "**_score**" field in the returned geneset object, which is the internal score representing how well the query matches the returned geneset object. It probably does not mean much in the geneset annotation service when only one geneset object is returned. In the geneset query service, by default, the returned geneset hits are sorted by the scores in descending order.

3.1.5 Species filter

We support **ALL** species annotated by NCBI and Ensembl. All of our services allow you to pass a "**species**" parameter to limit the query results. "species" parameter accepts taxonomy ids as the input. You can look for the taxomony ids for your favorite species from NCBI Taxonomy.

For convenience, we allow you to pass these *common names* for commonly used species (e.g. "species=human,mouse,rat"):

Search term (common name)	Genus name	Taxonomy id
human	Homo sapiens	9606
mouse	Mus musculus	10090
rat	Rattus norvegicus	10116
mosquito	Anopheles gambiae	180454
fruitfly	Drosophila melanogaster	7227
nematode	Caenorhabditis elegans	6239
zebrafish	Danio rerio	7955
thale-cress	Arabidopsis thaliana	3702
rice	Oryza sativa	39947
dog	Canis lupus familiaris	9615
chicken	Gallus gallus	9031
horse	Equus caballus	9796
chimpanzee	Pan troglodytes	9598
frog	Xenopus tropicalis	8364
pig	Sus scrofa	9823
pseudomonas-aeruginosa	Pseudomonas aeruginosa	208964
brewers-yeast	Saccharomyces cerevisiae	559292

If needed, you can pass "species=all" to query against all available species, although, we recommend you to pass specific species you need for faster response.

3.2 Data release notes

This page contains metadata about each MyGeneset.info data release. Click a link to see more.

3.2.1 MyGeneset Releases

3.3 Geneset query service

This page describes the reference for MyGeneset.info geneset query web service. It's also recommended to try it live on our interactive API page.

3.3.1 Service endpoint

```
http://mygeneset.info/v1/query
```

3.3.2 GET request

Query parameters

q

Required, passing user query. The detailed query syntax for parameter "**q**" we explained *below*.

fields

Optional, can be a comma-separated list of fields to limit the fields returned from the matching geneset hits. The supported field names can be found from any geneset object (e.g. geneset WP60). Note that it supports dot notation as well, e.g., you can pass "genes.symbol". If "fields=all", all available fields will be returned. Default: "_id,genes,name,description,source,author,date,is_public,taxid".

species

Optional, can be used to limit the geneset hits from given species. You can use "common names" for 17 common species. For all other species, you can provide their taxonomy ids. Multiple species can be passed using comma as a separator. Passing "all" will query against all available species. Default: all.

size

Optional, the maximum number of matching geneset hits to return (with a cap of 1000 at the moment). Default: 10.

from

Optional, the number of matching geneset hits to skip, starting from 0. Default: 0

Hint: The combination of "**size**" and "**from**" parameters can be used to get paging for a large query:

```
q=cdk*&size=50                    first 50 hits
q=cdk*&size=50&from=50            the next 50 hits
```

fetch_all

Optional, a boolean, which when TRUE, allows fast retrieval of all unsorted query hits. The return object contains a **_scroll_id** field, which when passed as a parameter to the query endpoint, returns the next 1000 query results. Setting **fetch_all** = TRUE causes the results to be inherently unsorted, therefore the **sort** parameter is ignored. For more information see *examples using fetch_all here*. Default: FALSE.

scroll_id

Optional, a string containing the **_scroll_id** returned from a query request with **fetch_all** = TRUE. Supplying a valid **scroll_id** will return the next 1000 unordered results. If the next results are not obtained within 1 minute of the previous set of results, the **scroll_id** becomes stale, and a new one must be obtained with another query request with **fetch_all** = TRUE. All other parameters are ignored when the **scroll_id** parameter is supplied. For more information see *examples using scroll_id here*.

sort

Optional, the comma-separated fields to sort on. Prefix with "-" for descending order, otherwise in ascending order. Default: sort by matching scores in descending order.

facets

Optional, a single field or comma-separated fields to return facets, for example, "facets=taxid", "facets=taxid,source". See *examples of faceted queries here*.

facet_size

Optional, an integer (1 <= **facet_size** <= 1000) that specifies how many buckets to return in a faceted query.

species_facet_filter

Optional, relevant when faceting on species (i.e., "facets=taxid" are passed). It's used to pass species filter without changing the scope of faceting, so that the returned facet counts won't change. Either species name or taxonomy id can be used, just like "*species*" parameter above. See *examples of faceted queries here*.

always_list

Optional, forces a field, or set of fields to always return an array. This is useful for fields such as **genes** and **genes.ensemblgene**, which may return an array or string. Can take a comma sepparated list of fields such as: "always_list=genes,genes.ensemblgene".

allow_null

Optional, forces a field, or set of fields to return "**Null**" if the field does not exist. Can take a comma sepparated list of fields such as: "allow_null=genes.ncbigene,genes.ensemblgene".

callback

Optional, you can pass a "**callback**" parameter to make a JSONP call.

dotfield

Optional, can be used to control the format of the returned geneset object. If "dotfield" is true, the returned data object is returned flattened (no nested objects) using dotfield notation for key names. Default: false.

filter

Alias for "fields" parameter.

limit

Alias for "size" parameter.

skip

Alias for "from" parameter.

email

Optional, if you are regular users of our services, we encourage you to provide us an email, so that we can better track the usage or follow up with you.

Query syntax

Examples of query parameter "**q**":

Simple queries

search for everything:

```
q=glucose                             search for any fields
q=tumor suppressor                    default as "AND" for all query terms
q="cyclin-dependent kinase"           search for the phrase
```

Fielded queries

```
q=genes.entrezgene:1017
q=genes.symbol:cdk2
q=source:reactome
q=kegg.id:"path:hsa04723"
```

Available fields

This table lists some commonly used fields can be used for "fielded queries". Check out http://mygeneset.info/v1/metadata/fields for the complete list of available fields.

Field	Description	Examples
name	Geneset name	q=name:kinase
description	Geneset description	q=description:cytosine deamination
source	Name of data source (if built-in)	q=source:kegg
author	Geneset creator (if user-created)	q=author:biothings
taxid	Taxonomic id	q=taxid:9606
genes.symbol	HGNC gene symbol	q=genes.symbol:cdk1
genes.ncbigene	NCBI gene id	q=genes.ncbigene:1017
genes.ensemblgene	Ensembl gene id	q=genes.ensemblgene:ENSG00000123374
genes.name	Gene name	q=genes.name:cyclin-dependent
genes.uniprot	UniProt SwissProt ID	q=genes.uniprot:P24941
genes.mygene_id	MyGene.info primary gene _id	q=genes.mygene_id:30
go.id	Gene Ontology id	q=go.id:GO_0042753
kegg.id	KEGG id	q=kegg.id:"ds:H01348"
wikipathways.id	Wikipathways id	q=wikipathways.id:WP60
msigdb.id	MSigDB id	q=msigdb.id:CCATCCA_MIR432
do.id	Disease Ontology id	q=do.id:DOID_0111329
ctd.id	CTD chemical id	q=ctd.id:C016705_9606
ctd.cas	chemical CAS id	q=ctd.cas:62-55-5
ctd.mesh	chemical mesh id	q=ctd.mesh:C016705
ctd.chemical_name	chemical name	q=ctd.chemical_name:acetaminophen

Wildcard queries

Wildcard character "*" or "?" is supported in either simple queries or fielded queries:

```
q=CDK?                               single character wildcard
q=genes.symbol:CDK?                        single character wildcard within "symbol"␣
↪field
q=IL*R                               multiple character wildcard
```

Note: Wildcard character can not be the first character. It will be ignored.

Boolean operators and grouping

You can use **AND/OR/NOT** boolean operators and grouping to form complicated queries:

```
q=tumor AND suppressor                        AND operator
q=genes.symbol:CDK2 OR BTK                    OR operator
q="tumor suppressor" NOT receptor             NOT operator
q=(interleukin OR insulin) AND receptor       the use of parentheses
```

Returned object

A GET request like this:

```
http://mygeneset.info/v1/query?q=genes.symbol:cdk2
```

should return hits as:

```
{
  "hits": [
    {
      "_id": "C016805_9606",
      "_score": 7.2883334,
      "genes": ["..."],
      "is_public": true,
      "taxid": "9606"
    },
    {
      "_id": "C017690_9606",
      "_score": 7.2883334,
      "genes": ["..."],
      "is_public": true,
      "taxid": "9606"
    },
    {
      "_id": "C017875_9606",
      "_score": 7.2883334,
      "genes": ["..."],
      "is_public": true,
      "taxid": "9606"
    }
  ]
}
```

Faceted queries

If you need to perform a faceted query, you can pass an optional "*facets*" parameter. For example, if you want to get the facets on species, you can pass "facets=taxid":

A GET request like this:

```
http://mygeneset.info/v1/query?q=glugose&size=1&facets=taxid
```

should return hits as:

```
{
  "hits":[
    {"..."}
  ],
  "total":26,
  "max_score":400.43347,
  "took":7,
  "facets":{
    "taxid":{
      "_type":"terms",
      "total":26,
```

(continues on next page)

(continued from previous page)

```
      "terms":[
        {
          "count":14,
          "term":9606
        },
        {
          "count":7,
          "term":10116
        },
        {
          "count":5,
          "term":10090
        }
      ],
      "other":0,
      "missing":0
    }
  }
}
```

Another useful field to get facets on is "type_of_gene":

```
http://mygeneset.info/v1/query?q=cdk2&size=1&facets=type_of_gene
```

It should return hits as:

```
{
  "hits":[
    {
      "entrezgene":1017,
      "name":"cyclin-dependent kinase 2",
      "_score":400.43347,
      "symbol":"CDK2",
      "_id":"1017",
      "taxid":9606
    }
  ],
  "total":26,
  "max_score":400.43347,
  "took":97,
  "facets":{
    "type_of_gene":{
      "_type":"terms",
      "total":26,
      "terms":[
        {
          "count":20,
          "term":"protein-coding"
        },
        {
          "count":6,
          "term":"pseudo"
        }
      ],
      "other":0,
      "missing":0
    }
```

(continues on next page)

(continued from previous page)

```
  }
}
```

If you need to, you can also pass multiple fields as comma-separated list:

```
http://mygeneset.info/v1/query?q=cdk2&size=1&facets=taxid,type_of_gene
```

Particularly relevant to species facets (i.e., "facets=taxid"), you can pass a "*species_facet_filter*" parameter to filter the returned hits on a given species, without changing the scope of the facets (i.e. facet counts will not change). This is useful when you need to get the subset of the hits for a given species after the initial faceted query on species.

You can see the different "hits" are returned in the following queries, while "facets" keeps the same:

```
http://mygeneset.info/v1/query?q=cdk?&size=1&facets=taxid&species_facet_filter=human
```

v.s.

```
http://mygeneset.info/v1/query?q=cdk?&size=1&facets=taxid&species_facet_filter=mouse
```

Scrolling queries

If you want to return ALL results of a very large query (>10,000 results), sometimes the paging method described *above* can take too long. In these cases, you can use a scrolling query. This is a two-step process that turns off database sorting to allow very fast retrieval of all query results. To begin a scrolling query, you first call the query endpoint as you normally would, but with an extra parameter **fetch_all** = TRUE. For example, a GET request to:

```
http://mygeneset.info/v1/query?q=brain&fetch_all=TRUE
```

Returns the following object:

```
{
  "_scroll_id":
↪"cXVlcnlUaGVuRmV0Y2g7MTA7MjA1NjY1MzMwOl9HM29rRkg2VFZ5S1c3cTJtYkI4RHc7MjA1NjY1MjY3OlM0V1VCa194UWdLY
↪",
  "max_score": 13.958638,
  "took": 270,
  "total": 14571,
  "hits": [
    {
      "_id": "390259",
      "_score": 13.958638,
      "entrezgene": 390259,
      "name": "brain specific homeobox",
      "symbol": "BSX",
      "taxid": 9606
    },
    .
    .
    .
  ]
}
```

At this point, the first 1000 hits have been returned (of ~14,000 total), and a scroll has been set up for your query. To get the next batch of 1000 unordered results, simply execute a GET request to the following address, supplying the _scroll_id from the first step into the **scroll_id** parameter in the second step:

```
http://mygeneset.info/v1/query?scroll_
↪id=cXVlcnlUaGVuRmV0Y2g7MTA7MjA1NjY1MzMwOl9HM29rRkg2VFZ5S1c3cTJtYkI4RHc7MjA1NjY1MjY3OlM0V1VCa194UWd
```

Hint: Your scroll will remain active for 1 minute from the last time you requested results from it. If your scroll expires before you get the last batch of results, you must re-request the scroll_id by setting **fetch_all** = TRUE as in step 1.

3.3.3 Batch queries via POST

Although making simple GET requests above to our geneset query service is sufficient in most of use cases, there are some cases you might find it's more efficient to make queries in a batch (e.g., retrieving gene annotation for multiple genes). Fortunately, you can also make batch queries via POST requests when you need:

```
URL: http://mygeneset.info/v1/query
HTTP method:  POST
```

Query parameters

q

Required, multiple query terms seperated by comma (also support "+" or white space), but no wildcard, e.g., 'q=1017,1018' or 'q=CDK2+BTK'

scopes

Optional, specify one or more fields (separated by comma) as the search "scopes", e.g., "scopes=entrezgene", "scopes=entrezgene,ensemblgene". The available "fields" can be passed to "**scopes**" parameter are *listed above*. Default: "scopes=entrezgene,ensemblgene,retired" (either Entrez or Ensembl gene ids).

species

Optional, can be used to limit the gene hits from given species. You can use "common names" for nine common species (human, mouse, rat, fruitfly, nematode, zebrafish, thale-cress, frog and pig). All other species, you can provide their taxonomy ids. See more details here. Multiple species can be passed using comma as a separator. Default: all.

fields

Optional, can be a comma-separated fields to limit the fields returned from the matching gene hits. The supported field names can be found from any gene object (e.g. gene 1017). Note that it supports dot notation as well, e.g., you can pass "refseq.rna". If "fields=all", all available fields will be returned. Default: "symbol,name,taxid,entrezgene".

dotfield

Optional, can be used to control the format of the returned fields when passed "fields" parameter contains dot notation, e.g. "fields=refseq.rna". If "dofield" is true, the returned data object contains a single "refseq.rna" field, otherwise, a single "refseq" field with a sub-field of "rna". Default: false.

email

Optional, if you are regular users of our services, we encourage you to provide us an email, so that we can better track the usage or follow up with you.

Example code

Unlike GET requests, you can easily test them from browser, make a POST request is often done via a piece of code. Here is a sample python snippet:

```python
import requests
headers = {'content-type': 'application/x-www-form-urlencoded'}
params = 'q=1017,1018&scopes=entrezgene&fields=name,symbol,taxid,entrezgene'
res = requests.post('http://mygeneset.info/v1/query', data=params, headers=headers)
```

Returned object

Returned result (the value of "res.text" variable above) from above example code should look like this:

```
[
  {
    '_id': '1017',
    '_score': 22.757837,
    'entrezgene': 1017,
    'name': 'cyclin dependent kinase 2',
    'query': '1017',
    'symbol': 'CDK2',
    'taxid': 9606
  },
  {
    '_id': '1018',
    '_score': 22.757782,
    'entrezgene': 1018,
    'name': 'cyclin dependent kinase 3',
    'query': '1018',
    'symbol': 'CDK3',
    'taxid': 9606
  }
]
```

Tip: "query" field in returned object indicates the matching query term.

Note: if no "fields" parameter is specified, all available fields will be returned

If a query term has no match, it will return with "**notfound**" field as "**true**":

```
params = 'q=1017,dummy&scopes=entrezgene&fields=name,symbol,taxid,entrezgene'
res = requests.post('http://mygeneset.info/v1/query', data=params, headers=headers)
```

```
[
  {
    "name": "cyclin-dependent kinase 2",
    "symbol": "CDK2",
    "taxid": 9606,
    "entrezgene": 1017,
    "query": "1017",
    "_id": "1017"
  },
  {
    "query": "dummy",
    "notfound": true
  }
]
```

If a query term has multiple matches, they will be included with the same "query" field:

```
params = 'q=tp53,1017&scopes=symbol,entrezgene&fields=name,symbol,taxid,entrezgene'
res = requests.post('http://mygeneset.info/v1/query', data=params, headers=headers)
```

```
[
  {
    "name": "tumor protein p53",
    "symbol": "TP53",
    "taxid": 9606,
    "entrezgene": 7157,
    "query": "tp53",
    "_id": "7157"
  },
  {
    "name": "tumor protein p53",
    "symbol": "Tp53",
    "taxid": 10116,
    "entrezgene": 24842,
    "query": "tp53",
    "_id": "24842"
  },
  {
    "name": "cyclin-dependent kinase 2",
    "symbol": "CDK2",
    "taxid": 9606,
    "entrezgene": 1017,
    "query": "1017",
    "_id": "1017"
  }
]
```

3.4 Geneset annotation service

This page describes the reference for MyGeneset.info geneset annotation web service. It's also recommended to try it live on our interactive API page.

3.4.1 Service endpoint

```
http://mygeneset.info/v1/geneset
```

3.4.2 GET request

To obtain the geneset annotation via our web service is as simple as calling this URL:

```
http://mygeneset.info/v1/geneset/<genesetid>
```

By default, this will return the complete geneset annotation object in JSON format. See *here* for an example and *here* for more details. If the input **genesetid** is not valid, 404 (NOT FOUND) will be returned.

Optionally, you can pass a "**fields**" parameter to return only the annotations you want (by filtering returned object fields):

```
http://mygeneset.info/v1/geneset/WP100?fields=genes.ncbigene,wikipathways.pathway_name
```

"**fields**" accepts any attributes (a.k.a fields) available from the geneset object. Multiple attributes should be seperated by commas. If an attribute is not available for a specific geneset object, it will be ignored. Note that the attribute names are case-sensitive.

Just like geneset query service, you can also pass a "**callback**" parameter to make a JSONP call.

Query parameters

fields

Optional, can be a comma-separated fields to limit the fields returned from the geneset object. If "fields=all", all available fields will be returned. Note that it supports dot notation as well, e.g., you can pass "refseq.rna". Default: "fields=all".

callback

Optional, you can pass a "**callback**" parameter to make a *JSONP <http://ajaxian.com/archives/jsonp-json-with-padding>* call.

filter

Alias for “fields” parameter.

dotfield

Optional, can be used to control the format of the returned fields when passed “fields” parameter contains dot notation, e.g. “fields=refseq.rna”. If “dofield” is true, the returned data object contains a single “refseq.rna” field, otherwise, a single “refseq” field with a sub-field of “rna”. Default: false.

email

Optional, if you are regular users of our services, we encourage you to provide us an email, so that we can better track the usage or follow up with you.

Returned object

A GET request like this:

```
http://mygeneset.info/v1/gene/1017
```

should return a geneset object below:

```
```

{ “_id”: “WP60”, “_version”: 1, “genes”: [

{ “ensemblgene”: “YOL165C”, “mygene_id”: “853999”, “name”: “putative aryl-alcohol dehydrogenase”, “ncbigene”: “853999”, “symbol”: “AAD15”, “uniprot”: “Q08361”

}, {

“mygene_id”: “850488”, “name”: “pseudo”, “ncbigene”: “850488”, “symbol”: “AAD6”

}, {

“ensemblgene”: “YNL331C”, “mygene_id”: “855385”, “name”: “putative aryl-alcohol dehydrogenase”, “ncbigene”: “855385”, “symbol”: “AAD14”, “uniprot”: “P42884”

}, {

“ensemblgene”: “YCR107W”, “mygene_id”: “850471”, “name”: “putative aryl-alcohol dehydrogenase”, “ncbigene”: “850471”, “symbol”: “AAD3”, “uniprot”: “P25612”

}, {

“ensemblgene”: “YJR155W”, “mygene_id”: “853620”, “name”: “putative aryl-alcohol dehydrogenase”, “ncbigene”: “853620”, “symbol”: “AAD10”, “uniprot”: “P47182”

}, {

“ensemblgene”: “YDL243C”, “mygene_id”: “851354”, “name”: “putative aryl-alcohol dehydrogenase”, “ncbigene”: “851354”, “symbol”: “AAD4”, “uniprot”: “Q07747”

}

], “is_public”: true, “name”: “Toluene degradation”, “source”: “wikipathways”, “taxid”: 559292, “wikipathways”: {

"_license": "https://www.wikipathways.org/index.php/WikiPathways:License_Terms", "id": "WP60", "pathway_name": "Toluene degradation", "url": "http://www.wikipathways.org/instance/WP60_r89654"

}

}

3.4.3 Batch queries via POST

Although making simple GET requests above to our geneset query service is sufficient in most of use cases, there are some cases you might find it's more efficient to make queries in a batch (e.g., retrieving gene annotation for multiple genes). Fortunately, you can also make batch queries via POST requests when you need:

```
URL: http://mygeneset.info/v1/gene
HTTP method:  POST
```

Query parameters

ids

Required. Accept multiple geneset ids seperated by comma, e.g., 'ids=1017,1018' or 'ids=695,ENSG00000123374'. Note that currently we only take the input ids up to **1000** maximum, the rest will be omitted.

fields

Optional, can be a comma-separated fields to limit the fields returned from the matching hits. If "fields=all", all available fields will be returned. Note that it supports dot notation as well, e.g., you can pass "refseq.rna". Default: "symbol,name,taxid,entrezgene".

species

Optional, can be used to limit the gene hits to a given species. You can use "common names" for 17 common species. All other species, you can provide their taxonomy ids. See more details here. Multiple species can be passed using comma as a separator. Passing "all" will query against all available species. Default: all.

dotfield

Optional, can be used to control the format of the returned fields when passed "fields" parameter contains dot notation, e.g. "fields=refseq.rna". If "dofield" is true, the returned data object contains a single "refseq.rna" field, otherwise, a single "refseq" field with a sub-field of "rna". Default: false.

email

Optional, if you are regular users of our services, we encourage you to provide us an email, so that we can better track the usage or follow up with you.

Example code

Unlike GET requests, you can easily test them from browser, make a POST request is often done via a piece of code, still trivial of course. Here is a sample python snippet:

```
import requests
headers = {'content-type': 'application/x-www-form-urlencoded'}
params = 'ids=WP60,WP100&fields=name,taxid,genes.symbol'
res = requests.post('http://mygeneset.info/v1/geneset', data=params, headers=headers)
```

Returned object

The returned result (the value of "res.text") from the example code above should look like this:

```
[
  {
    "query": "WP60",
    "_id": "WP60",
    "_version": 1,
    "genes": [
      {
        "symbol": "AAD15"
      },
      {
        "symbol": "AAD6"
      },
      {
        "symbol": "AAD14"
      },
      {
        "symbol": "AAD3"
      },
      {
        "symbol": "AAD10"
      },
      {
        "symbol": "AAD4"
      }
    ],
    "name": "Toluene degradation",
    "taxid": 559292
  },
  {
    "query": "WP100",
    "_id": "WP100",
    "_version": 1,
    "genes": [
      {
        "symbol": "GGT2"
      },
```

(continues on next page)

(continued from previous page)

```
      {
        "symbol": "GGTLC2"
      },
      {
        "symbol": "ANPEP"
      },
      {
        "symbol": "OPLAH"
      },
      {
        "symbol": "GSTA5"
      },
      {
        "symbol": "GGTLC1"
      },
      {
        "symbol": "IDH1"
      },
      {
        "symbol": "GPX3"
      },
      {
        "symbol": "GSTM1"
      },
      {
        "symbol": "GPX2"
      },
      {
        "symbol": "GGT1"
      },
      {
        "symbol": "GPX1"
      },
      {
        "symbol": "GSTT2"
      },
      {
        "symbol": "GGT5"
      },
      {
        "symbol": "LOC102724197"
      },
      {
        "symbol": "GCLM"
      },
      {
        "symbol": "GSTA1"
      },
      {
        "symbol": "GCLC"
      },
      {
        "symbol": "GSS"
      },
      {
        "symbol": "GSR"
      },
```

(continues on next page)

(continued from previous page)

```
      {
        "symbol": "GSTM2"
      },
      {
        "symbol": "G6PD"
      },
      {
        "symbol": "GPX4"
      }
    ],
    "name": "Glutathione metabolism",
    "taxid": 9606
  }
]
```

3.5 Server response

The MyGeneset.info server returns a variety of query responses, and response status codes. They are listed here.

Note: These examples show query responses using the python requests package.

3.5.1 Status code *200*

A **200** status code indicates a successful query, and is accompanied by the query response payload.

```
In [1]: import requests

In [2]: r = requests.get('http://mygeneset.info/v1/query?q=_exists_:go')

In [3]: r.status_code
Out[3]: 200

In [4]: data = r.json()

In [5]: data.keys()
Out[5]: dict_keys(['total', 'max_score', 'took', 'hits'])
```

3.5.2 Status code *400*

A **400** status code indicates an improperly formed query, and is accompanied by a response payload describing the source of the error.

```
In [6]: r = requests.get('http://mygeneset.info/v1/query?q=_exists_:go&size=u')

In [7]: r.status_code
Out[7]: 400

In [8]: data = r.json()
```

(continues on next page)

(continued from previous page)

```
In [9]: data
Out[9]:
{'error': "Expect type int.",
 'success': False}
```

3.5.3 Status code *404*

A **404** status code indicates either an unrecognized URL, as in (*/query* is misspelled */quer* resulting in an unrecognized URL):

```
In [10]: r = requests.get('http://mygeneset.info/v1/quer?q=_exists_:go')

In [11]: r.status_code
Out[11]: 404
```

or, for the **/geneset** endpoint, a **404** status code could be from querying for a nonexistent geneset ID, as in:

```
In [12]: r = requests.get('http://mygeneset.info/v1/geneset/0')

In [13]: r.status_code
Out[13]: 404

In [14]: data = r.json()

In [15]: data
Out[15]:
{'error': "ID '0' not found",
 'success': False}
```

3.5.4 Status code *5xx*

Any **5xx** status codes are the result of uncaught query errors. Ideally, these should never occur. We routinely check our logs for these types of errors and add code to catch them, but if you see any status **5xx** responses, please submit a bug report to help@mygeneset.info.

CHAPTER

FOUR

HOW TO CITE

See citation page here: http://mygeneset.info/about#cite

CHAPTER FIVE

FAQ

See FAQ page here: http://mygeneset.info/about#faqs

CHAPTER SIX

RELATED LINKS

- mygeneset.info on Github

CHAPTER

SEVEN

CONTACT US

- help@mygeneset.info
- @mygeneinfo