system package

Submodules

system.admin module

system.blastn module

blastn.py is the ntsequence search tool

@author Bowen

system.fasta_reader module

`system.fasta_reader.`**`parse_fasta_file`**(*fasta_filename*) [source]

`system.fasta_reader.`**`parse_fasta_str`**(*fasta_str*) [source]

`system.fasta_reader.`**`read_fastas`**(*fp*) [source]

system.gene module

gene.py realize the methods that are related to system recommendation.

@author: Bowen

`system.gene.`**`format_fuzzy_result`**(*es_result*) [source]

format the es search result to front end processable format

@param es_result: the es search result @type es_result: dict @return: the front end processable format, while will be like this:

```
[{'compound_id': id, 'name': name},...]
```

@rtype: list

`system.gene.`**`fuzzy_search_compound`**(*es, keyword*) [source]

fuzzy search compound based on the keyword with elasticsearch

@param es: the elasticsearch object @param keyword: the search keyword @type es: Elasticsearch @type keyword: str @return a dict generated by the elasticsearch, which contains the search result @rtype: dict

class `system.gene.`**`gene_graph`**(*cid_list, ogm*) [source]

gene graph, including calculation and generate of gene & protein relation graph

`cal_graph`() [source]

calculate the relation graph

`create_n_link`(*center_node, compound_obj*) [source]

create nodes and link them @param center_node: source node @type center_node:compound @param compound_obj: compound object @type compound_obj: compound

`create_node`(*name, id*) [source]

create a node (gene or compound) in the graph

@param name: name for the node @param id: id for the node @type name : str @type id : str

get_compound_object(*cid*) [source]

get compound object by compound id

@param cid: compound id @type cid: str @return: compound object or none if not found @rtype: compound

get_graph() [source]

get the graph

@return: th graph @rtype: dict

get_or_create_gene(*gid*) [source]

find gene in database, if found, return gene, or search in ncbi

@param gid: gene id @type gid: str @return gene object @rtype: gene

related_compound(*cid*) [source]

find a compound's related compound

@param cid: compound id @type cid: str @return: list of related compound @rtype: list

retrive_gene_detain(*gid*) [source]

get gene data from ncib

@param gid: gene id @type gid: str @return: gene information in dict or none @rtype: dict

save_relation_to_db(*geneIdList*, *compound_obj*) [source]

save relation between compound_obj and gene to database

@param geneIdList: gene id in a list @type geneIdList: list @param compound_obj: compound object @type compound_obj: compound

search_gene(*compound_obj*) [source]

find gene realted to a compound

@param compound_obj: the compound object @type compound_obj: compound @return related genes @rtype: list

`system.gene.`**get_compound_info**(*cid*) [source]

get a specific compound's information

@param cid: compound id @type cid: str @return: a tunple that contains is compound can be retrived and the information @rtype: dict

`system.gene.`**get_gene_info**(*gid*) [source]

get gene information from the database

@param gid: the gene id @ytpe gid: str @return: gene information dict @rtype: dict

`system.gene.`**get_or_create_gene**(*gid*) [source]

get gene object from database, or get from ncbi and create one

@param gid: gene id @type gid: str @return: gene object @rtype: gene

`system.gene.`**retrive_gene_detain**(*gid*) [source]

get gene data from ncbi

@param gid: gene id @type gid: str @return: a dictionary that contains gene information @rtype: dict

`system.gene.`**search_compound**(*keyword*) [source]

search compound based on the keyword

@param keyword: the keyword that the user typed. Which would be used in search @type keyword: str @return: return a list that contains searched compounds @rtype: list

system.models module

class `system.models.`**`compound`**(*compound_id, name, nicknames, formula, exact_mass, mol_mass*) [source]

Bases: **`django.db.models.base.Model`**

exception **`DoesNotExist`**

Bases: **`django.core.exceptions.ObjectDoesNotExist`**

exception `compound.`**`MultipleObjectsReturned`**

Bases: **`django.core.exceptions.MultipleObjectsReturned`**

`compound.`**`compound_gene_set`**

`compound.`**`objects`** = *<django.db.models.manager.Manager object>*

`compound.`**`pathway_compound_set`**

`compound.`**`reaction_compound_set`**

class `system.models.`**`compound_gene`**(*id, compound_id, gene_id*) [source]

Bases: **`django.db.models.base.Model`**

exception **`DoesNotExist`**

Bases: **`django.core.exceptions.ObjectDoesNotExist`**

exception `compound_gene.`**`MultipleObjectsReturned`**

Bases: **`django.core.exceptions.MultipleObjectsReturned`**

`compound_gene.`**`compound`**

`compound_gene.`**`gene`**

`compound_gene.`**`objects`** = *<django.db.models.manager.Manager object>*

class `system.models.`**`gene`**(*gene_id, name, nicknames, definition, organism_short, organism, position, ntseq_length, ntseq*) [source]

Bases: **`django.db.models.base.Model`**

exception **`DoesNotExist`**

Bases: **`django.core.exceptions.ObjectDoesNotExist`**

exception `gene.`**`MultipleObjectsReturned`**

Bases: **`django.core.exceptions.MultipleObjectsReturned`**

`gene.`**`compound_gene_set`**

`gene.`**`objects`** = *<django.db.models.manager.Manager object>*

`gene.`**`part_gene_set`**

class `system.models.`**`organism`**(*organism_id, organism_short, organism_name*) [source]

Bases: **`django.db.models.base.Model`**

exception **`DoesNotExist`**

Bases: **`django.core.exceptions.ObjectDoesNotExist`**

exception `organism.`**`MultipleObjectsReturned`**

Bases: **`django.core.exceptions.MultipleObjectsReturned`** `organism.`**`objects`** = *<django.db.models.manager.Manager object>*

`organism.`**`pathway_set`**

class `system.models.`**`part_gene`**(*id*, *part_id*, *gene_id*, *score*) [source]

Bases: **`django.db.models.base.Model`**

exception **`DoesNotExist`**

Bases: **`django.core.exceptions.ObjectDoesNotExist`**

exception `part_gene.`**`MultipleObjectsReturned`**

Bases: **`django.core.exceptions.MultipleObjectsReturned`**

`part_gene.`**`gene`**

`part_gene.`**`objects`** = *<django.db.models.manager.Manager object>*

`part_gene.`**`part`**

class `system.models.`**`pathway`**(*pathway_id*, *pathway_name*, *organism_id*) [source]

Bases: **`django.db.models.base.Model`**

exception **`DoesNotExist`**

Bases: **`django.core.exceptions.ObjectDoesNotExist`**

exception `pathway.`**`MultipleObjectsReturned`**

Bases: **`django.core.exceptions.MultipleObjectsReturned`**

`pathway.`**`objects`** = *<django.db.models.manager.Manager object>*

`pathway.`**`organism`**

`pathway.`**`pathway_compound_set`**

class `system.models.`**`pathway_compound`**(*id*, *pathway_id*, *compound_id*, *score*) [source]

Bases: **`django.db.models.base.Model`**

exception **`DoesNotExist`**

Bases: **`django.core.exceptions.ObjectDoesNotExist`**

exception `pathway_compound.`**`MultipleObjectsReturned`**

Bases: **`django.core.exceptions.MultipleObjectsReturned`**

`pathway_compound.`**`compound`**

`pathway_compound.`**`objects`** = *<django.db.models.manager.Manager object>*

`pathway_compound.`**`pathway`**

class `system.models.`**`reaction`**(*reaction_id*, *name*, *definition*, *equation*) [source]

Bases: **`django.db.models.base.Model`**

exception **`DoesNotExist`**

Bases: **`django.core.exceptions.ObjectDoesNotExist`**

exception `reaction.`**`MultipleObjectsReturned`**

Bases: **`django.core.exceptions.MultipleObjectsReturned`**

`reaction.`**`objects`** = *<django.db.models.manager.Manager object>*

`reaction.`**`reaction_compound_set`**

class `system.models.`**`reaction_compound`**(*id*, *reaction_id*, *compound_id*, *isReactant*, *isResultant*, *amount*) [source]

Bases: `django.db.models.base.Model`

exception **`DoesNotExist`**

Bases: `django.core.exceptions.ObjectDoesNotExist`

exception `reaction_compound.`**`MultipleObjectsReturned`**

Bases: `django.core.exceptions.MultipleObjectsReturned`

`reaction_compound.`**`compound`**

`reaction_compound.`**`objects`** = *<django.db.models.manager.Manager object>*

`reaction_compound.`**`reaction`**

system.tests module

system.urls module

system.views module

`system.views.`**`getCompound`**(**args*, ***kwargs*) [source]

`system.views.`**`getGene`**(**args*, ***kwargs*) [source]

`system.views.`**`getRelatedCompound`**(**args*, ***kwargs*) [source]

`system.views.`**`searchCompound`**(**args*, ***kwargs*) [source]

`system.views.`**`systemView`**(**args*, ***kwargs*) [source]

Module contents