design package

Submodules

design.admin module

design.file module

generate design result image

@author: Bowen

`design.file.`**`createFolder`**(*contentType*) [source]

create the folders with time steamp

@param contentType: file type in the folder @type contentType: str @return: created path @rtype: str

`design.file.`**`drawCurve`**(*drawer*, *cbox*, *isForward*) [source]

`design.file.`**`drawOnePart`**(*name*, *position*, *drawer*, *isSmall*, *icon_im*) [source]

`design.file.`**`drawSequence`**(*sequenceInfo*, *width*, *height*, *file_path*) [source]

`design.file.`**`geneFileName`**(*name*, *surfix*) [source]

generate file name with time and given name

@param name: name for the file @type name: str @param surfix: file extension @type surfix:str @return : filename @rtype: str

`design.file.`**`getSequenceResultImage`**(*sequence*, *width*, *height*, *name*) [source]

get a part sequence image

@param sequence: part sequence @type sequence: str @param width: image width @type width: int @param height: image height @param name: name for that image @type name:str @return : image file path @rtype: str

design.models module

class `design.models.`**`chain`**(*id*, *sequence*, *project_id*, *name*, *isModified*, *image_file_path*) [source]

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

exception **`DoesNotExist`**

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

exception `chain.`**`MultipleObjectsReturned`**

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

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

`chain.`**`project`**

class `design.models.`**`features`**(*feature_id*, *title*, *feature_type*, *direction*, *startpos*, *endpos*) [source]

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

exception **`DoesNotExist`**

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

exception `features.`**`MultipleObjectsReturned`**

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

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

features.**part_features_set**

class design.models.**functions**(*id*, *function*) [source]

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

exception **DoesNotExist**

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

exception functions.**MultipleObjectsReturned**

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

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

functions.**project_set**

functions.**teams_set**

functions.**track_functions_set**

class design.models.**paper**(*paper_id*, *paper_name*, *paper_file_location*, *paper_url*) [source]

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

exception **DoesNotExist**

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

exception paper.**MultipleObjectsReturned**

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

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

paper.**part_papers_set**

class design.models.**part_features**(*id*, *part_id*, *feature_id*) [source]

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

exception **DoesNotExist**

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

exception part_features.**MultipleObjectsReturned**

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

part_features.**feature**

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

part_features.**part**

class design.models.**part_papers**(*id*, *part_id*, *paper_id*) [source]

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

exception **DoesNotExist**

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

exception part_papers.**MultipleObjectsReturned**

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

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

part_papers.**paper**

part_papers.**part**

class design.models.**part_parameters**(*id*, *part_id*, *name*, *value*) [source]

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

exception **DoesNotExist**

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

exception part_parameters.**MultipleObjectsReturned**

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

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

part_parameters.**part**

class design.models.**part_twins**(*id*, *part_1_id*, *part_2_id*) [source]

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

exception **DoesNotExist**

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

exception part_twins.**MultipleObjectsReturned**

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

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

part_twins.**part_1**

part_twins.**part_2**

class design.models.**parts**(*part_id*, *ok*, *part_name*, *short_desc*, *description*, *part_type*, *author*, *status*, *dominant*, *discontinued*, *part_status*, *sample_status*, *p_status_cache*, *s_status_cache*, *in_stock*, *results*, *favorite*, *specified_u_list*, *deep_u_list*, *deep_count*, *ps_string*, *scars*, *barcode*, *notes*, *source*, *nickname*, *premium*, *categories*, *sequence*, *sequence_length*, *part_url*, *score*) [source]

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

exception **DoesNotExist**

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

parts.**FK_PART_TWIN2**

exception parts.**MultipleObjectsReturned**

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

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

parts.**part_features_set**

parts.**part_papers_set**

parts.**part_parameters_set**

parts.**part_twins_set**

parts.**team_parts_set**

class design.models.**project**(*id*, *project_name*, *creator_id*, *create_time*, *function_id*, *track_id*, *is_deleted*) [source]

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

exception **DoesNotExist**

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

exception project.**MultipleObjectsReturned**

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

project.**chain_set**

project.**creator**

project.**function**

project.**get_next_by_create_time**(**moreargs*, ***morekwargs*)

project.**get_previous_by_create_time**(**moreargs*, ***morekwargs*)

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

project.**track**

project.**user_project_set**

class design.models.**team_parts**(*id*, *team_id*, *part_id*) [source]

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

exception **DoesNotExist**

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

exception team_parts.**MultipleObjectsReturned**

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

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

team_parts.**part**

team_parts.**team**

class design.models.**teams**(*team_id*, *name*, *track_id*, *function_id*, *year*) [source]

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

exception **DoesNotExist**

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

exception teams.**MultipleObjectsReturned**

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

teams.**function**

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

teams.**team_parts_set**

teams.**track**

class design.models.**track_functions**(*id*, *track_id*, *function_id*) [source]

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

exception **DoesNotExist**

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

exception track_functions.**MultipleObjectsReturned**

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

track_functions.**function**

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

track_functions.**track**

class design.models.**tracks**(*id*, *track*) [source]

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

exception **DoesNotExist**

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

exception tracks.**MultipleObjectsReturned**

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

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

tracks.**project_set**

tracks.**teams_set**

tracks.**track_functions_set**

class design.models.**user_project**(*id*, *user_id*, *project_id*) [source]

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

exception **DoesNotExist**

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

exception user_project.**MultipleObjectsReturned**

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

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

user_project.**project**

user_project.**user**

design.project module

design.project.**formatProjectList**(*projectList*) [source]

design.project.**getChain**(*chainId*) [source]

design.project.**getChainList**(*projectId*) [source]

design.project.**getUserProject**(*userObj*) [source]

design.project.**searchProject**(*keyword*, *userObj*) [source]

design.recommend module

implement recommend for parts

@author: Bowen, Ray, Yu

design.recommend.**analyseData**(*dataList*, *dataLength=2*) [source]

design.recommend.**getApriorRecommend**(*chainStr*, *funcStr=None*) [source]

get recommendations with aprior algorithm

@param chainStr: part chain @type chainStr: str @return : recommendations @rytpe: dict

design.recommend.**getMarkovRecommend**(*part_id*) [source]

get recommendations with Markov algorithm

@param part_id: part id @type part_id: str @return : recommendations @rytpe: dict

design.recommend.**getPartNameAndType**(*part_id*) [source]

design.recommend.**getResult**(*currentList*, *dataList*) [source]

design.recommend.**get_chain**(*elem*, *num*, *process*) [source]

get chain which had predicted

according to information in process, get the chain from first element to elem variable and save the chain in a list

args:

elem: the last element in chain num: the line number in process process: a variable record the predict process

return:

a chain from first to elem variable

`design.recommend.`**`loadA`**() [source]

`design.recommend.`**`predict`**(*m*, *count*, *s*, *A*) [source]

predict the chain after s

calculate the probability of a m-length chain, then return chains. CAUTION the number of chains maybe less then count

args:

m: the length of predict chain count: the number of predict chain s: the last element of the current chain A: transition matrix

return:

some chains save in list

`design.recommend.`**`toBeOne`**(*data*) [source]

`design.recommend.`**`toFrozenset`**(*data*) [source]

design.search_part module

search_part.py realize the part search

@author: Bowen

`design.search_part.`**`ambiguousSearch`**(*keyword*, *funcs*) [source]

ambiguous search parts with the keyword, and adjust result with the functions

@param keyword: search keyword @type keyword: str @param funcs: functions @type: str @return: search result @rtype: list

`design.search_part.`**`format_fuzzy_result`**(*hits*) [source]

format search result

@param hits: searched parts @type hists: list @return part informaions @rtype: list

`design.search_part.`**`fuzzy_search_parts`**(*es*, *keyword*) [source]

fuzzy search part with elasticsearch

@param es: elasticsearch object @type es: Elasticsearch @param keyword: search keyword @type keyword: str @return: elasticsearch search result @rtype: dict

`design.search_part.`**`getPart`**(*partName*) [source]

find the part with part name

@param partName: name of a part @type partName: str @return : part information @rtype: dict

`design.search_part.`**`get_func_parts`**(*func_list*) [source]

get parts related to functions

@param func_list: functions @type func_list: list @return : parts related to functions @rtype: list

`design.search_part.`**`sort_result`**(*es_result*, *funcs*) [source]

sort result according to the functions

@param funcs: functions @type funcs : list @return : sorted result @rtype: list

design.simulation module

simulation.py implement the simulation function

@author: Bowen

class design.simulation.**reaction_simulator**(*reaction_list*, *martial_list*, *reaction_time=100*) [source]

reaction simulator

calA(*reaction*) [source]

cal a in SSA

@param reaction: reaction to cal with @type reaction: dict @return : a0 @rtype: float

calA0() [source]

calItoJ(*i*, *j*) [source]

calTT(*a0*, *r1*) [source]

cal when reaction happens

@param a0 : a0 in SSA @type a0: int @param r1: random number @type r1: float @return : reaction happend time point @rtype: float

doReaction(*index*) [source]

make reaction happens

@param index: reaction index @type index: int

doSimulation() [source]

simulate the reaction

form_result() [source]

getProcess() [source]

get the simulate result

@return: simulate result @rtype: list

get_compound_amount(*t*, *compound*) [source]

compound amount at time t

@param t: time point @type t: int @param compound: compound name @type compound: str @return: amount of compound @rtype:int

isJOccurs(*j*, *a0*, *r2*) [source]

isReactantReady(*j*) [source]

design.tests module

class design.tests.**MarkovTestCase**(*methodName='runTest'*) [source]

Bases: **django.test.testcases.TestCase**

test_get_chain() [source]

test_predict() [source]

class design.tests.**RecommendTestCase**(*methodName='runTest'*) [source]

Bases: **django.test.testcases.TestCase**

setUp() [source]

test_MarkovRecommend() [source]

design.urls module

design.views module

@author: Bowen

design.views.**changeProjectName**(**args*, ***kwargs*) [source]

design.views.**changeProjectTrack**(**args*, ***kwargs*) [source]

design.views.**createNewDevice**(**args*, ***kwargs*) [source]

design.views.**createProject**(**args*, ***kwargs*) [source]

design.views.**dashboardView**(**args*, ***kwargs*) [source]

design.views.**deleteProject**(**args*, ***kwargs*) [source]

design.views.**getARecommend**(**args*, ***kwargs*) [source]

design.views.**getChainLength**(**args*, ***kwargs*) [source]

design.views.**getCurrentUserObj**(*request*) [source]

design.views.**getMRecommend**(**args*, ***kwargs*) [source]

design.views.**getParts**(**args*, ***kwargs*) [source]

design.views.**getProject**(**args*, ***kwargs*) [source]

design.views.**getProjectChain**(**args*, ***kwargs*) [source]

design.views.**getProjectChains**(**args*, ***kwargs*) [source]

design.views.**getResultImage**(**args*, ***kwargs*) [source]

design.views.**getTrackFunctions**(**args*, ***kwargs*) [source]

design.views.**getTracks**(**args*, ***kwargs*) [source]

design.views.**getUserProjects**(**args*, ***kwargs*) [source]

design.views.**isAccountActive**(*request*) [source]

design.views.**isLoggedIn**(*request*) [source]

design.views.**newProject**(*name*, *user*, *track*) [source]

design.views.**projectView**(**args*, ***kwargs*) [source]

design.views.**saveChain**(**args*, ***kwargs*) [source]

design.views.**searchParts**(**args*, ***kwargs*) [source]

design.views.**simulate**(**args*, ***kwargs*) [source]

design.views.**simulationView**(**args*, ***kwargs*) [source]

Module contents