Getting Unstuck Web

Aug 05, 2020

Contents

The Getting Unstuck web app.

The project repository is hosted here: https://github.com/GSE-CCL/getting-unstuck-web

CHAPTER 1

API documentation

CHAPTER 2

latest

2.1 app module

app.**about**()

app.**add_schema**()

app.**admin_index**()

app.**admin_page**(*page*)

app.**clear_cache**()

app.**edit_schema**(*id*)

app.**error**(*e*)
Handle errors.

app.**error_page**(*eid*)

app.**feedback_owner**(*pid*)

app.**generate_certificate**()

app.**generate_summary**()

app.**homepage**()

app.**ie**()

app.**index**()

app.**inject_vars**()

app.**login**()

app.**logout**()

app.**md**()

app.**project__id**(*pid*)

app.**project_download**()

app.**project_feedback**(*pid*)

app.**project_id**(*pid*)

app.**prompts**()

app.**redirect_to**()

app.**register**()

app.**reload_project**(*pid*)

app.**research**()

app.**schema_editor**(*id*)

app.**setup**()

app.**signup**()

app.**strategies**()

app.**studio**()

app.**studio_id**(*sid*)

app.**studio_list**(*sid*)

app.**summarize**()

app.**summary_image**()

app.**user_id**(*username*)

2.2 lib package

2.2.1 Submodules

2.2.2 lib.admin module

lib.admin.**get_info**(*page*)
Gets the relevant information that would be used on a given admin page.

Parameters **page** (*str*) – name of the admin page.

Returns A dictionary mapping keys of information to whatever that information is. Empty dictionary if not a valid admin page name. Purposely broad so as to be abstract-ish.

lib.admin.**set_info**(*page*, *form*)
Makes server-side changes as needed, and returns updated field(s) as needed.

Parameters

- **page** (*str*) – name of the admin page.
- **form** (*dict*) – the form data being passed in from the client.

Returns If successful, returns True. If not successful, returns False.

2.2.3 lib.authentication module

class lib.authentication.**User**(**args*, ***values*)
Bases: mongoengine.document.Document

exception DoesNotExist
Bases: mongoengine.errors.DoesNotExist

exception MultipleObjectsReturned
Bases: mongoengine.errors.MultipleObjectsReturned

deleted
Boolean field type.

New in version 0.1.2.

email
A field that validates input as an email address.

New in version 0.4.

first_name
A unicode string field.

id
A field wrapper around MongoDB's ObjectIds.

joined
Datetime field.

Uses the python-dateutil library if available alternatively use time.strptime to parse the dates. Note: python-dateutil's parser is fully featured and when installed you can utilise it to convert varying types of date formats into valid python datetime objects.

Note: To default the field to the current datetime, use: DateTimeField(default=datetime.utcnow)

Note: Microseconds are rounded to the nearest millisecond. Pre UTC microsecond support is effectively broken. Use ComplexDateTimeField if you need accurate microsecond support.

last_name
A unicode string field.

objects
The default QuerySet Manager.

Custom QuerySet Manager functions can extend this class and users can add extra queryset functionality. Any custom manager methods must accept a Document class as its first argument, and a QuerySet as its second argument.

The method function should return a QuerySet , probably the same one that was passed in, but modified in some way.

password
A unicode string field.

role
A unicode string field.

username
A unicode string field.

lib.authentication.**admin_required**(*f*)
Decorate routes to require login with admin role.

lib.authentication.**get_login_info**()
Returns session login info, if it exists. Else returns False.

lib.authentication.**login_required**(*f*)
Decorate routes to require login. http://flask.pocoo.org/docs/1.0/patterns/viewdecorators/

lib.authentication.**login_user**(*username*, *password*)
Logs in a user, setting a session cookie to that effect.

Parameters

- **username** (*str*) – either username or email of user.
- **password** (*str*) – plaintext password of this login attempt.

Returns True, if able to log in. Else False.

lib.authentication.**register_user**(*username*, *email*, *first_name*, *last_name*, *password*, *role='site_viewer'*)
Registers a user to our database.

Parameters

- **username** (*str*) – the username
- **email** (*str*) – the email
- **first_name** (*str*) – user's first name
- **last_name** (*str*) – user's last name
- **password** (*str*) – plaintext password
- **role** (*str*) – one of site_viewer or site_admin. Optional; defaults to site_viewer.

Returns True if user added successfully, else either False or a specific error message to pass to the user.

lib.authentication.**session_active**()
True if there's a user login right now.

2.2.4 lib.cache module

class lib.cache.**CachedItem**(**args*, ***values*)
Bases: mongoengine.document.Document

exception DoesNotExist
Bases: mongoengine.errors.DoesNotExist

exception MultipleObjectsReturned
Bases: mongoengine.errors.MultipleObjectsReturned

data
A binary data field.

expires
Datetime field.

Uses the python-dateutil library if available alternatively use time.strptime to parse the dates. Note: python-dateutil's parser is fully featured and when installed you can utilise it to convert varying types of date formats into valid python datetime objects.

Note: To default the field to the current datetime, use: DateTimeField(default=datetime.utcnow)

Note: Microseconds are rounded to the nearest millisecond. Pre UTC microsecond support is effectively broken. Use `ComplexDateTimeField` if you need accurate microsecond support.

id
A field wrapper around MongoDB's ObjectIds.

meta = {'db_alias': 'cache'}

name
A unicode string field.

objects
The default QuerySet Manager.

Custom QuerySet Manager functions can extend this class and users can add extra queryset functionality. Any custom manager methods must accept a `Document` class as its first argument, and a `QuerySet` as its second argument.

The method function should return a `QuerySet` , probably the same one that was passed in, but modified in some way.

class lib.cache.**MongoCache**(*app*, *c*, *d*, *default_timeout=300*, *hash_method=<built-in function openssl_md5>*, *credentials='secure/db.json'*)
Bases: `flask_caching.backends.base.BaseCache`

A cache that stores the items in a MongoDB collection.

Parameters

- **default_timeout** (*int*) – the default timeout that is used (in seconds). A timeout of 0 indicates that the cache never expires.
- **hash_method** – Default hashlib.md5. The hash method used to generate the filename for cached results.
- **credentials** (*dict*) – the MongoDB credentials, passed to common.connect_db.

add(*key*, *value*, *timeout=None*)
Works like *set()* but does not overwrite the values of already existing keys.

Parameters

- **key** – the key to set
- **value** – the value for the key
- **timeout** – the cache timeout for the key in seconds (if not specified, it uses the default timeout). A timeout of 0 idicates that the cache never expires.

Returns Same as *set()*, but also `False` for already existing keys.

Return type boolean

clear()
Clears the cache. Keep in mind that not all caches support completely clearing the cache.

Returns Whether the cache has been cleared.

Return type boolean

delete(*key*)
Delete *key* from the cache.

Parameters **key** – the key to delete.

Returns Whether the key existed and has been deleted.

Return type boolean

get (*key*)
Look up key in the cache and return the value for it.

Parameters **key** – the key to be looked up.

Returns The value if it exists and is readable, else `None`.

has (*key*)
Checks if a key exists in the cache without returning it. This is a cheap operation that bypasses loading the actual data on the backend.

This method is optional and may not be implemented on all caches.

Parameters **key** – the key to check

set (*key*, *value*, *timeout=None*)
Add a new key/value to the cache (overwrites value, if key already exists in the cache).

Parameters

- **key** – the key to set
- **value** – the value for the key
- **timeout** – the cache timeout for the key in seconds (if not specified, it uses the default timeout). A timeout of 0 indicates that the cache never expires.

Returns `True` if key has been updated, `False` for backend errors. Pickling errors, however, will raise a subclass of `pickle.PickleError`.

Return type boolean

2.2.5 lib.certificate module

`lib.certificate.`**convert_cert** (*template*, *username*, *projectnum*, *cache_directory='cache'*)
Converts template to the certificate pdf with inputs.

Parameters

- **template** (*str*) – the certificate template provided as a string directed to a html page.
- **username** (*str*) – username to add to the certificate.
- **projectnum** (*int*) – the number of projects completed by the user, should be 10 or fewer.
- **cache_directory** (*str*) – will save this certificate into the cache directory specified. Defaults to settings.CACHE_DIRECTORY.

Returns True or False depending on the success of the certificate conversion.

2.2.6 lib.common module

`lib.common.`**connect_db** (*credentials_file='secure/db.json'*, *alias='default'*)
Connects to MongoDB using credentials.

Parameters

- **credentials_file** (*str*) – path to the credentials file, or the Python dictionary containing the contents thereof.
- **alias** (*str*) – alias for the connection.

Returns A MongoEngine connection instance.

lib.common.**get_credentials**(*credentials_file*)
Gets credentials into a dict.

Parameters **credentials_file** (*str*) – path to the credentials file, or the Python dictionary containing the contents thereof.

Returns the credentials as a dictionary.

Return type credentials (dict)

lib.common.**get_selected**(*stat*)

lib.common.**human_block**(*opcode*)

lib.common.**indexOf**(*lst*, *value*)

lib.common.**md**(*text*)
Converts Markdown to HTML, including Scratchblocks.

Parameters **text** (*str*) – the Markdown to convert.

Returns A string of HTML from the converted Markdown.

lib.common.**pluralize**(*item*)

lib.common.**twodec**(*value*)

2.2.7 lib.display module

lib.display.**get_code_excerpt**(*project*, *sc*, *include_orphans=False*)
Gets a relevant code excerpt from a project based on the schema.

Parameters

- **project** (*dict*) – the project's Mongoengine representation.
- **schema** (*dict*) – the schema's Mongoengine representation.
- **include_orphans** (*bool*) – whether to include orphan blocks. Defaults to False.

Returns

A tuple – first, a string of Scratchblocks syntax, then info about the relevant sprite.

Both will be blank if couldn't find an example.

lib.display.**get_comparisons**(*project*, *sc*, *count*, *credentials_file='secure/db.json'*)
Gets comparison projects based on the schema.

Parameters

- **project** (*dict*) – the Mongoengine representation of the project.
- **sc** (*dict*) – the Mongoengine representation of the schema.
- **count** (*int*) – the number of projects to return.

Returns A list of projects, as stored in the database, to use as examples.

lib.display.**get_feels**(*feels='c:/users/jarch/code/getting-unstuck-web/lib/data/feels.json'*, *randomize=False*, *alphabetize=False*)
Get the emotions (feels, for the younger among us) that a user can choose to describe their experience.

Parameters

- **feels** (*str*) – the file path to feels.json, from which the feels are loaded.
- **randomize** (*bool*) – whether to randomize the order of feels. Default is False.
- **alphabetize** (*bool*) – whether to alphabetize by text. Default is False. Will override randomize.

Returns A list of feelings the end user may feel. False if problem loading file.

lib.display.**get_project_page**(*pid*, *cache_directory='cache'*)
Get a project page rendered in HTML given a project ID.

Parameters

- **pid** (*int*) – project ID.
- **cache_directory** (*str*) – the directory where cached projects are stored.

Returns A string containing the HTML for the page.

lib.display.**get_studio_stats**(*sc*, *studio*)
Gets the studio stats based on schema requirements.

Parameters

- **sc** (*dict*) – the schema.
- **studio** (*dict*) – the studio.

Returns A list of dicts, each with keys "name" and "value" to describe each statistic.

2.2.8 lib.errors module

class lib.errors.**Error**(**args*, ***values*)
Bases: mongoengine.document.Document

exception DoesNotExist
Bases: mongoengine.errors.DoesNotExist

exception MultipleObjectsReturned
Bases: mongoengine.errors.MultipleObjectsReturned

error_code
32-bit integer field.

id
A field wrapper around MongoDB's ObjectIds.

objects
The default QuerySet Manager.

Custom QuerySet Manager functions can extend this class and users can add extra queryset functionality. Any custom manager methods must accept a Document class as its first argument, and a QuerySet as its second argument.

The method function should return a QuerySet , probably the same one that was passed in, but modified in some way.

status
A unicode string field.

timestamp
Datetime field.

Uses the python-dateutil library if available alternatively use time.strptime to parse the dates. Note: python-dateutil's parser is fully featured and when installed you can utilise it to convert varying types of date formats into valid python datetime objects.

Note: To default the field to the current datetime, use: DateTimeField(default=datetime.utcnow)

Note: Microseconds are rounded to the nearest millisecond. Pre UTC microsecond support is effectively broken. Use `ComplexDateTimeField` if you need accurate microsecond support.

traceback
A unicode string field.

url
A field that validates input as an URL.

New in version 0.3.

`lib.errors.`**`add_error`**(*error_code*, *url*, *traceback*, *status='open'*, *credentials_file='secure/db.json'*)
Adds an error to database, returning the ID of the error as saved.

Parameters

- **`error_code`** (*int*) – the error code.
- **`url`** (*str*) – the URL where this error occurred.
- **`traceback`** (*str*) – the traceback.
- **`status`** (*str*) – the status of the error, from {open, closed}. Defaults to "open."
- **`credentials_file`** (*str*) – the path to the credentials file.

Returns The MongoDB document ID of this error, as a string.

`lib.errors.`**`get_error`**(*eid*, *credentials_file='secure/db.json'*)
Gets an error from the database.

Parameters

- **`eid`** (*str*) – the error ID.
- **`credentials_file`** (*str*) – the path to the credentials file.

Returns errors.Error if found, else False.

2.2.9 lib.reports module

`lib.reports.`**`get_projects`**()
Gets projects as a CSV.

`lib.reports.`**`get_studios`**()
Gets studios as a CSV.

2.2.10 lib.schema module

`class` `lib.schema.`**`Blockify`**(**args*, ***kwargs*)
Bases: `mongoengine.document.EmbeddedDocument`

comments
32-bit integer field.

costumes
32-bit integer field.

sounds
32-bit integer field.

sprites
32-bit integer field.

variables
32-bit integer field.

class lib.schema.**Challenge**(**args*, ***values*)
Bases: mongoengine.document.Document

exception DoesNotExist
Bases: mongoengine.errors.DoesNotExist

exception MultipleObjectsReturned
Bases: mongoengine.errors.MultipleObjectsReturned

comparison_basis
A dictionary field that wraps a standard Python dictionary. This is similar to an embedded document, but the structure is not defined.

Note: Required means it cannot be empty - as the default for DictFields is {}

New in version 0.3.

Changed in version 0.5: - Can now handle complex / varying types of data

description
A unicode string field.

id
A field wrapper around MongoDB's ObjectIds.

min_blockify
An embedded document field - with a declared document_type. Only valid values are subclasses of EmbeddedDocument.

min_comments_made
32-bit integer field.

min_description_length
32-bit integer field.

min_instructions_length
32-bit integer field.

modified
Datetime field.

Uses the python-dateutil library if available alternatively use time.strptime to parse the dates. Note: python-dateutil's parser is fully featured and when installed you can utilise it to convert varying types of date formats into valid python datetime objects.

Note: To default the field to the current datetime, use: DateTimeField(default=datetime.utcnow)

Note: Microseconds are rounded to the nearest millisecond. Pre UTC microsecond support is effectively broken. Use ComplexDateTimeField if you need accurate microsecond support.

objects
The default QuerySet Manager.

Custom QuerySet Manager functions can extend this class and users can add extra queryset functionality. Any custom manager methods must accept a `Document` class as its first argument, and a `QuerySet` as its second argument.

The method function should return a `QuerySet` , probably the same one that was passed in, but modified in some way.

required_block_categories
A dictionary field that wraps a standard Python dictionary. This is similar to an embedded document, but the structure is not defined.

Note: Required means it cannot be empty - as the default for DictFields is {}

New in version 0.3.

Changed in version 0.5: - Can now handle complex / varying types of data

required_blocks
A list field that wraps a standard field, allowing multiple instances of the field to be used as a list in the database.

If using with ReferenceFields see: one-to-many-with-listfields

---

Note: Required means it cannot be empty - as the default for ListFields is []

---

required_blocks_failure
A unicode string field.

required_text
A list field that wraps a standard field, allowing multiple instances of the field to be used as a list in the database.

If using with ReferenceFields see: one-to-many-with-listfields

---

Note: Required means it cannot be empty - as the default for ListFields is []

---

required_text_failure
A unicode string field.

short_label
A unicode string field.

stats
A list field that wraps a standard field, allowing multiple instances of the field to be used as a list in the database.

If using with ReferenceFields see: one-to-many-with-listfields

---

Note: Required means it cannot be empty - as the default for ListFields is []

---

text
An embedded document field - with a declared document_type. Only valid values are subclasses of `EmbeddedDocument`.

title
A unicode string field.

url
An embedded document field - with a declared document_type. Only valid values are subclasses of `EmbeddedDocument`.

class lib.schema.**Link**(**args*, ***kwargs*)
Bases: mongoengine.document.EmbeddedDocument

text
A unicode string field.

url
A field that validates input as an URL.

New in version 0.3.

class lib.schema.**ResultText**(**args*, ***kwargs*)
Bases: mongoengine.document.EmbeddedDocument

comparison_framing_text
A unicode string field.

comparison_reflection_text
A unicode string field.

concluding_text
A unicode string field.

explanation
A unicode string field.

prompt_framing_text
A unicode string field.

stats_framing_text
A unicode string field.

lib.schema.**add_schema**(*mins={}*, *min_blockify={}*, *required_text=[]*, *required_block_categories={}*, *required_blocks=[]*, *required_blocks_failure=None*, *required_text_failure=None*, *comparison_basis={'basis': '__none__', 'priority': []}*, *stats=[]*, *short_label=None*, *title=None*, *description=None*, *url=None*, *text={}*, *credentials_file='secure/db.json'*)
Adds a new challenge schema to the database. No arguments are required; but passing in no arguments is pretty useless.

Parameters

- **mins** (*dict*) – a dictionary mapping meta names from the set {"instructions_length", "description_length", "comments_made"} to minimum values.
- **min_blockify** (*dict*) – a dictionary mapping blockify names from the set {"comments", "costumes", "sounds", "sprites", "variables"} to minimum counts.
- **required_text** (*list*) – a list of lists. If thought about as a list of shape (i, j), then required_text[i][j] is one option, along with required_text[i][j + 1] etc., to satisfy required_text[i]. All required_text[i] must be satisfied to pass overall.
- **required_block_categories** (*dict*) – a dict, mapping category name to minimum count for respective category.

- **required_blocks** (*list*) – a list of dicts. Each dict maps opcode to minimum count. To satisfy overall requirement, at least one requirement in each dict must be satisfied. (That is, the list functions as "AND"; the dict functions as "OR".)
- **require_blocks_failure** (*str*) – the failure message to show if block requirement isn't met.
- **require_text_failure** (*str*) – the failure message to show if text requirement isn't met.
- **comparison_basis** (*str*) – determines what to base code excerpts on. Must be from set of {"__none__", "required_text", "required_blocks", "required_block_categories"}
- **stats** (*list*) – a list of studio-wide stats to feature on the project results page. Separate category and stat with a slash, e.g. max/comments.
- **short_label** (*str*) – the short label descriptor of the prompt.
- **title** (*str*) – the title of the prompt.
- **description** (*str*) – the description of the prompt.
- **url** (*dict*) – the URL the prompts page will feature. Is a dictionary with keys "url" and "text".
- **text** (*dict*) – a dictionary mapping results page text items from the set {"explanation", "concluding_text", "comparison_reflection_text", "comparison_framing_text", "prompt_framing_text", "stats_framing_text"}
- **credentials_file** (*str*) – path to the database credentials file.

Returns The object ID of the new challenge schema in the database. False if the arguments violate the validation schema.

lib.schema.**get_schema**(*schema_id*, *credentials_file='secure/db.json'*)
Gets a schema from the database.

Parameters

- **schema_id** (*str*) – the ID of the schema.
- **credentials_file** (*str*) – path to the database credentials file.

Returns A dictionary representation of the schema.

lib.schema.**valid_comparison_basis**(*param*)
Raises a ValidationError if doesn't meet format for comparison_basis.

lib.schema.**valid_required_blocks**(*param*)
Raises a ValidationError if doesn't meet format for required_blocks.

lib.schema.**valid_required_text**(*param*)
Raises a ValidationError if doesn't meet format for required_text.

lib.schema.**validate_project**(*schema*, *project*, *studio_id*, *credentials_file='secure/db.json'*)
Determines if the project meets the standards of a given schema.

Parameters

- **schema** – The validation schema. Can be given by its ID in the database or with a dictionary representing its values.
- **project** – The project. Given as either the Mongo object or the project_id.
- **studio_id** (*int*) – The studio ID.

- **credentials_file** (*str*) – path to the database credentials file.

Returns A modified version of the validation schema, revealing whether each requirement was met. False if couldn't successfully validate the project, as when there's no valid schema.

2.2.11 lib.scrape module

class lib.scrape.**Comment**(**args*, ***values*)

Bases: mongoengine.document.Document

exception DoesNotExist

Bases: mongoengine.errors.DoesNotExist

exception MultipleObjectsReturned

Bases: mongoengine.errors.MultipleObjectsReturned

author

A unicode string field.

comment_id

32-bit integer field.

content

A unicode string field.

date

Datetime field.

Uses the python-dateutil library if available alternatively use time.strptime to parse the dates. Note: python-dateutil's parser is fully featured and when installed you can utilise it to convert varying types of date formats into valid python datetime objects.

Note: To default the field to the current datetime, use: DateTimeField(default=datetime.utcnow)

Note: Microseconds are rounded to the nearest millisecond. Pre UTC microsecond support is effectively broken. Use ComplexDateTimeField if you need accurate microsecond support.

id

A field wrapper around MongoDB's ObjectIds.

objects

The default QuerySet Manager.

Custom QuerySet Manager functions can extend this class and users can add extra queryset functionality. Any custom manager methods must accept a Document class as its first argument, and a QuerySet as its second argument.

The method function should return a QuerySet , probably the same one that was passed in, but modified in some way.

project_id

32-bit integer field.

recipient

A unicode string field.

class lib.scrape.**Project**(**args*, ***values*)

Bases: mongoengine.document.Document

exception DoesNotExist

Bases: mongoengine.errors.DoesNotExist

exception MultipleObjectsReturned
Bases: mongoengine.errors.MultipleObjectsReturned

author
A unicode string field.

cache_expires
Datetime field.

Uses the python-dateutil library if available alternatively use time.strptime to parse the dates. Note: python-dateutil's parser is fully featured and when installed you can utilise it to convert varying types of date formats into valid python datetime objects.

Note: To default the field to the current datetime, use: DateTimeField(default=datetime.utcnow)

Note: Microseconds are rounded to the nearest millisecond. Pre UTC microsecond support is effectively broken. Use ComplexDateTimeField if you need accurate microsecond support.

description
A unicode string field.

engagement
A dictionary field that wraps a standard Python dictionary. This is similar to an embedded document, but the structure is not defined.

Note: Required means it cannot be empty - as the default for DictFields is {}

New in version 0.3.

Changed in version 0.5: - Can now handle complex / varying types of data

history
A dictionary field that wraps a standard Python dictionary. This is similar to an embedded document, but the structure is not defined.

Note: Required means it cannot be empty - as the default for DictFields is {}

New in version 0.3.

Changed in version 0.5: - Can now handle complex / varying types of data

id
A field wrapper around MongoDB's ObjectIds.

image
A unicode string field.

instructions
A unicode string field.

objects
The default QuerySet Manager.

Custom QuerySet Manager functions can extend this class and users can add extra queryset functionality. Any custom manager methods must accept a Document class as its first argument, and a QuerySet as its second argument.

The method function should return a QuerySet , probably the same one that was passed in, but modified in some way.

project_id
32-bit integer field.

reload_page
Boolean field type.

New in version 0.1.2.

remix
A dictionary field that wraps a standard Python dictionary. This is similar to an embedded document, but the structure is not defined.

Note: Required means it cannot be empty - as the default for DictFields is {}

New in version 0.3.

Changed in version 0.5: - Can now handle complex / varying types of data

stats
A dictionary field that wraps a standard Python dictionary. This is similar to an embedded document, but the structure is not defined.

Note: Required means it cannot be empty - as the default for DictFields is {}

New in version 0.3.

Changed in version 0.5: - Can now handle complex / varying types of data

studio_id
32-bit integer field.

title
A unicode string field.

validation
A dictionary field that wraps a standard Python dictionary. This is similar to an embedded document, but the structure is not defined.

Note: Required means it cannot be empty - as the default for DictFields is {}

New in version 0.3.

Changed in version 0.5: - Can now handle complex / varying types of data

class lib.scrape.**ProjectReflection**(**args*, ***values*)
Bases: mongoengine.document.Document

exception DoesNotExist
Bases: mongoengine.errors.DoesNotExist

exception MultipleObjectsReturned
Bases: mongoengine.errors.MultipleObjectsReturned

feelings
A list field that wraps a standard field, allowing multiple instances of the field to be used as a list in the database.

If using with ReferenceFields see: one-to-many-with-listfields

Note: Required means it cannot be empty - as the default for ListFields is []

gu_uid
A unicode string field.

id
A field wrapper around MongoDB's ObjectIds.

minutes
32-bit integer field.

objects
The default QuerySet Manager.

Custom QuerySet Manager functions can extend this class and users can add extra queryset functionality. Any custom manager methods must accept a `Document` class as its first argument, and a `QuerySet` as its second argument.

The method function should return a `QuerySet` , probably the same one that was passed in, but modified in some way.

project_id
32-bit integer field.

timestamp
Datetime field.

Uses the python-dateutil library if available alternatively use time.strptime to parse the dates. Note: python-dateutil's parser is fully featured and when installed you can utilise it to convert varying types of date formats into valid python datetime objects.

Note: To default the field to the current datetime, use: DateTimeField(default=datetime.utcnow)

Note: Microseconds are rounded to the nearest millisecond. Pre UTC microsecond support is effectively broken. Use `ComplexDateTimeField` if you need accurate microsecond support.

class `lib.scrape.`**Studio**(**args*, ***values*)
Bases: `mongoengine.document.Document`

exception DoesNotExist
Bases: `mongoengine.errors.DoesNotExist`

exception MultipleObjectsReturned
Bases: `mongoengine.errors.MultipleObjectsReturned`

challenge_id
A field wrapper around MongoDB's ObjectIds.

description
A unicode string field.

id
A field wrapper around MongoDB's ObjectIds.

objects
The default QuerySet Manager.

Custom QuerySet Manager functions can extend this class and users can add extra queryset functionality. Any custom manager methods must accept a `Document` class as its first argument, and a `QuerySet` as its second argument.

The method function should return a `QuerySet` , probably the same one that was passed in, but modified in some way.

public_show
Boolean field type.

New in version 0.1.2.

stats
A dictionary field that wraps a standard Python dictionary. This is similar to an embedded document, but the structure is not defined.

Note: Required means it cannot be empty - as the default for DictFields is {}

New in version 0.3.

Changed in version 0.5: - Can now handle complex / varying types of data

status
A unicode string field.

studio_id
32-bit integer field.

title
A unicode string field.

lib.scrape.**add_comments**(*project_id*, *username*, *credentials_file='secure/db.json'*)
Inserts a project's comments into the database. These are public comments on the project itself, not code comments.

Parameters

- **project_id** (*int*) – the ID of the project whose comments we're scraping.
- **username** (*str*) – the username of the user who created the project.
- **credentials_file** (*str*) – path to the database credentials file.

Returns None.

lib.scrape.**add_project**(*project_id*, *studio_id=0*, *cache_directory=None*, *credentials_file='secure/db.json'*)
Inserts a project into the database after scraping it. Updates existing database entries.

Parameters

- **project_id** (*int*) – the ID of the project to scrape.
- **studio_id** (*int*) – the studio ID with which this project should be associated.
- **cache_directory** (*str*) – if set, will save this project JSON into the cache directory specified.
- **credentials_file** (*str*) – path to the database credentials file.

Returns True, if a new insertion or if updated a record. False if Scratch 2.

Raises `IOError` – if couldn't write the JSON file to the given cache_directory.

lib.scrape.**get_default_studio_stats**()
Returns the default studio stats dictionary.

lib.scrape.**get_project**(*project_id*, *cache_directory=None*, *credentials_file='secure/db.json'*)
Retrieves a project from database and cache (if available).

Parameters

- **project_id** (*int*) – the ID of the project to retrieve.
- **cache_directory** (*str*) – if set, will get the project in JSON form.
- **credentials_file** (*str*) – path to the database credentials file.

Returns A tuple, the first element being the database object as a dictionary, the second the Scratch project as a dictionary.

Raises ArgumentError, if project_id can't be cast to an integer.

lib.scrape.**get_project_from_cache**(*project_id*, *cache_directory='cache'*)
Retrieves a project from the cache.

Parameters

- **project_id** (*int*) – the ID of the project to retrieve.
- **cache_directory** (*str*) – if set, will get the project in JSON form.

Returns The project as a dictionary. Empty if unavailable.

lib.scrape.**get_projects_with_block**(*opcode*, *project_id=0*, *studio_id=0*, *credentials_file='secure/db.json'*)
Finds projects with given opcode.

Parameters

- **opcode** (*str*) – Scratch opcodes for the block type.
- **project_id** (*int*) – exclude this project from the search.
- **studio_id** (*int*) – limit to projects in this studio.
- **credentials_file** (*str*) – path to the database credentials file.

Returns A list of projects, as stored in the database, with that opcode.

lib.scrape.**get_projects_with_category**(*category*, *count=1*, *project_id=0*, *studio_id=0*, *credentials_file='secure/db.json'*)
Finds projects with given category.

Parameters

- **category** (*str*) – block category name.
- **count** (*int*) – minimum block count for that category.
- **project_id** (*int*) – exclude this project from the search.
- **studio_id** (*int*) – limit to projects in this studio.
- **credentials_file** (*str*) – path to the database credentials file.

Returns A QuerySet of projects, as stored in the database, with at least the given count of blocks of the given category.

lib.scrape.**get_reload_project**(*pid*)
Returns whether to reload the project result page regardless of the cache state.

Parameters **pid** (*int*) – the project ID.

Returns True, if should be reloaded. Else, False.

lib.scrape.**get_studio**(*studio_id*, *credentials_file='secure/db.json'*)
Retrieves a studio from database.

Parameters

- **studio_id** (*int*) – the ID of the studio to retrieve.
- **credentials_file** (*str*) – path to the database credentials file.

Returns The studio as a dictionary.

Raises ArgumentError, if studio_id can't be cast to an integer.

lib.scrape.**get_studio_stats**(*studio_id*, *credentials_file='secure/db.json'*)
Returns a dictionary of statistics about a studio.

Parameters

- **studio_id** (*int*) – the ID of the studio to scrape.
- **credentials_file** (*str*) – path to the database credentials file.

Returns A dictionary of statistics, including mean, min, and max.

lib.scrape.**proper_feelings**(*param*)
Raises a ValidationError if doesn't meet format for feelings.

lib.scrape.**set_reload_page**(*pid*)
Sets a project to reload result page regardless of cache state.

Parameters **pid** (*int*) – project ID.

Returns True, if successful. Else, False.

2.2.12 lib.settings module

2.2.13 lib.summary module

lib.summary.**get_author_origins**(*authors*)
Gets the origin locations of project authors.

Parameters **authors** (*array-like*) – a set of authors for whom origin locations are to be counted.

Returns A dictionary mapping countries to number of authors from there.

lib.summary.**get_image_urls**(*studio_ids=None*, *credentials_file='secure/db.json'*)
Gets image URLs from database.

Parameters

- **studio_ids** (*array-like*) – a set of studios from which to gather project images. Defaults to None, in which case will get all studios' project images.
- **credentials_file** (*str*) – path to the database credentials file.

Returns A set of image URLs.

lib.summary.**get_ordered_studios**(*credentials_file='secure/db.json'*)
Gets the studios ordered by short label and title.

Parameters **credentials_file** (*str*) – path to the database credentials file.

Returns An ordered list of studios. Studios without schemas will be excluded.

lib.summary.**get_stitched**(*urls*, *x*, *y=0*, *w=24*, *h=18*, *solids=False*)
Stitches together images from a set of URLs.

Parameters

- **urls** (*array-like*) – a set of URLs to images to stitch together.
- **x** (*int*) – number of columns of the composite image.
- **y** (*int*) – number of rows of the composite image. Defaults to however many needed based on x.
- **w** (*int*) – width of individual images in composite. Default 24.
- **h** (*int*) – height of individual images in composite. Default 18.
- **solids** (*bool*) – whether to include solid color images (e.g., all white or all black). Default False.

Returns PIL.Image of composite image.

lib.summary.**get_total_categories**(*studios*)
Gets total category counts.

Parameters **studios** (*array-like*) – a list of studios, either as MongoDB representations or as dictionaries.

Returns A dictionary mapping each category to a total block count.

lib.summary.**get_total_engagement**(*studio_ids*, *credentials_file='secure/db.json'*)
Gets engagement for projects within a studio.

Parameters

- **studio_ids** (*array-like*) – a list of studio IDs for which to retrieve project engagement.
- **credentials_file** (*str*) – path to the database credentials file.

Returns A dictionary mapping {views, loves, favorites} to integers representing the total counts in the studios chosen.

lib.summary.**get_unique_authors**(*studio_ids*, *credentials_file='secure/db.json'*)
Gets the unique authors of projects across studios.

Parameters

- **studio_ids** (*array-like*) – the list of studio IDs for studios for which a set of unique authors is desired.
- **credentials_file** (*str*) – path to the database credentials file.

Returns A set of unique authors of projects.

2.2.14 lib.tasks module

lib.tasks.**make_celery**(*name*, *backend*, *broker*, *app*)
Make the Celery app. https://flask.palletsprojects.com/en/1.1.x/patterns/celery/

Parameters

- **name** (*str*) – the name of the app.
- **backend** (*str*) – the URI of the backend.
- **broker** (*str*) – the URI of the broker.
- **app** (*Flask.app*) – the Flask app we're adding the Celery instance to.

Returns An instance of celery.Celery.

2.2.15 Module contents

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

l

Index

H

I

J

L

M

N

O

P

R

S

T

U

V