
pine

JHU/APL

Apr 09, 2021

CONTENTS:

1	API Reference	1
1.1	pine	1
2	Indices and tables	83
	Python Module Index	85
	Index	87

API REFERENCE

This page contains auto-generated API reference documentation¹.

1.1 pine

1.1.1 Subpackages

`pine.backend`

Subpackages

`pine.backend.admin`

This module implements all methods required for user authentication when using PINE's db authentication rather than an external Auth service

Submodules

`pine.backend.admin.bp`

Module Contents

Functions

<code>get_users()</code>	Get the list of all users' details (id, email, and password hash)
<code>get_user(user_id)</code>	Given a <code>user_id</code> , return the user's details (id, email, and password hash)
<code>update_user_password(user_id)</code>	Change the password hash stored in the database for the given user to a newly calculated password hash derived from
<code>update_user(user_id)</code>	Change the details stored in the database for the given user to those provided in the json body of this request.

continues on next page

¹ Created with sphinx-autoapi

Table 1 – continued from previous page

<code>add_user()</code>	Add a new user to PINE, with the details provided in the json body of this request (id, email, and password hash).
<code>delete_user(user_id)</code>	Delete the user matching the given user_id
<code>system_export()</code>	Export the contents of the database as a zip file
<code>system_import()</code>	Import the contents of the data provided in the request body to the database
<code>init_app(app)</code>	

`pine.backend.admin.bp.bp`

`pine.backend.admin.bp.get_users()`

Get the list of all users' details (id, email, and password hash) :return: str

`pine.backend.admin.bp.get_user(user_id)`

Given a user_id, return the user's details (id, email, and password hash) :param user_id: str :return: str

`pine.backend.admin.bp.update_user_password(user_id)`

Change the password hash stored in the database for the given user to a newly calculated password hash derived from the password provided in the json body of this request. :param user_id: :return: Response

`pine.backend.admin.bp.update_user(user_id)`

Change the details stored in the database for the given user to those provided in the json body of this request. :param user_id: str :return: Response

`pine.backend.admin.bp.add_user()`

Add a new user to PINE, with the details provided in the json body of this request (id, email, and password hash). This method will calculate and store a password hash based upon the provided password :return: Response

`pine.backend.admin.bp.delete_user(user_id)`

Delete the user matching the given user_id :param user_id: str :return: Response

`pine.backend.admin.bp.system_export()`

Export the contents of the database as a zip file :return: Response

`pine.backend.admin.bp.system_import()`

Import the contents of the data provided in the request body to the database :return: Response

`pine.backend.admin.bp.init_app(app)`

`pine.backend.annotations`

This module contains the api methods required to perform and display annotations in the front-end and store the annotations in the backend

Submodules

`pine.backend.annotations.bp`

Module Contents

Functions

<code>check_document_view_by_id(doc_id: str)</code>	Verify that a document with the given doc_id exists and that the logged in user has permissions to access the
<code>check_document_view(doc: dict)</code>	
<code>check_document_annotate(doc: dict)</code>	
<code>get_my_annotations_for_document(doc_id)</code>	Get the list of annotations (key, start_index, end_index) produced by the logged in user for the document matching
<code>get_others_annotations_for_document(doc_id)</code>	Get the list of annotations (key, start_index, end_index) produced by all other users, not including the logged in
<code>get_annotations_for_document(doc_id)</code>	Get the list of annotations (key, start_index, end_index) produced by all users for the document matching the
<code>get_current_annotation(doc_id, user_id)</code>	Get all annotations of the provided document created by the given user.
<code>check_overlapping_annotations(collection, ner_annotations)</code>	
<code>set_document_to_annotated_by_user(doc_id, user_id)</code>	Modify the parameter in the database for the document signifying that the given user has annotated the given
<code>_make_annotations(body)</code>	
<code>_add_or_update_annotation(new_annotation)</code>	
<code>save_annotations(doc_id)</code>	Save new NER annotations and labels to the database as an entry for the logged in user, for the document. If there
<code>save_collection_annotations(collection_id: str)</code>	
<code>init_app(app)</code>	

pine.backend.annotations.bp.logger

pine.backend.annotations.bp.CONFIG_ALLOW_OVERLAPPING_NER_ANNOTATIONS = allow_overlapping_ner

pine.backend.annotations.bp.bp

pine.backend.annotations.bp.check_document_view_by_id(doc_id: str)

Verify that a document with the given doc_id exists and that the logged in user has permissions to access the document :param doc_id: str :return: dict

pine.backend.annotations.bp.check_document_view(doc: dict)

pine.backend.annotations.bp.check_document_annotate(doc: dict)

pine.backend.annotations.bp.get_my_annotations_for_document(doc_id)

Get the list of annotations (key, start_index, end_index) produced by the logged in user for the document matching the provided doc_id. :param doc_id: str :return: Response

pine.backend.annotations.bp.get_others_annotations_for_document(doc_id)

Get the list of annotations (key, start_index, end_index) produced by all other users, not including the logged in user for the document matching the provided doc_id. :param doc_id: str :return: str

pine.backend.annotations.bp.get_annotations_for_document(doc_id)

Get the list of annotations (key, start_index, end_index) produced by all users for the document matching the provided doc_id. :param doc_id: str :return: str

```
pine.backend.annotations.bp.get_current_annotation(doc_id, user_id)
    Get all annotations of the provided document created by the given user. :param doc_id: str :param user_id: str
    :return: List

pine.backend.annotations.bp.check_overlapping_annotations(collection,
                                                         ner_annotations)

pine.backend.annotations.bp.set_document_to_annotated_by_user(doc_id, user_id)
    Modify the parameter in the database for the document signifying that the given user has annotated the given
    document :param doc_id: str :param user_id: str :return: whether the update succeeded :rtype: bool

pine.backend.annotations.bp._make_annotations(body)

pine.backend.annotations.bp._add_or_update_annotation(new_annotation)

pine.backend.annotations.bp.save_annotations(doc_id)
    Save new NER annotations and labels to the database as an entry for the logged in user, for the document. If
    there are already annotations, use a patch request to update with the new annotations. If there are not, use a post
    request to create a new entry. :param doc_id: str :return: bool

pine.backend.annotations.bp.save_collection_annotations(collection_id: str)

pine.backend.annotations.bp.init_app(app)
```

pine.backend.auth

Submodules

pine.backend.auth.bp

Module Contents

Classes

AuthModule

Functions

is_flat()

get_logged_in_user()

login_required(view)

admin_required(view)

flask_get_module()

flask_get_flat() → flask.Response

continues on next page

Table 4 – continued from previous page

<code>flask_get_can_manage_users()</code>	→
<code>flask.Response</code>	
<code>flask_get_logged_in_user()</code>	→
<code>flask.Response</code>	
<code>flask_get_logged_in_user_details()</code>	→
<code>flask.Response</code>	
<code>flask_get_login_form()</code>	→ <code>flask.Response</code>
<code>flask_post_logout()</code>	→ <code>flask.Response</code>
<code>init_app(app)</code>	

```

pine.backend.auth.bp.CONFIG_AUTH_MODULE_KEY = AUTH_MODULE
pine.backend.auth.bp.bp
pine.backend.auth.bp.module
pine.backend.auth.bp.is_flat ()
pine.backend.auth.bp.get_logged_in_user ()
pine.backend.auth.bp.login_required (view)
pine.backend.auth.bp.admin_required (view)
pine.backend.auth.bp.flask_get_module ()
pine.backend.auth.bp.flask_get_flat () → flask.Response
pine.backend.auth.bp.flask_get_can_manage_users () → flask.Response
pine.backend.auth.bp.flask_get_logged_in_user () → flask.Response
pine.backend.auth.bp.flask_get_logged_in_user_details () → flask.Response
pine.backend.auth.bp.flask_get_login_form () → flask.Response
pine.backend.auth.bp.flask_post_logout () → flask.Response
class pine.backend.auth.bp.AuthModule (app, bp)
    Bases: object
    __metaclass__
    abstract is_flat (self) → bool
    abstract can_manage_users (self) → bool
    abstract get_login_form (self) → pine.backend.models.LoginForm
    get_logged_in_user (self)
    get_logged_in_user_details (self) → pine.backend.models.UserDetails
    logout (self)
pine.backend.auth.bp.init_app (app)

```

pine.backend.auth.eve

Module Contents

Classes

EveUser

EveModule

```
class pine.backend.auth.eve.EveUser (data)
    Bases: pine.backend.models.AuthUser

    property id (self)
    property username (self)
    property display_name (self)
    property is_admin (self)
    get_details (self) → pine.backend.models.UserDetails

class pine.backend.auth.eve.EveModule (app, bp)
    Bases: pine.backend.auth.bp.AuthModule

    is_flat (self) → bool
    can_manage_users (self) → bool
    get_logged_in_user_details (self)
    update_user_details (self)
    update_user_password (self)
    get_login_form (self) → pine.backend.models.LoginForm
    login (self) → flask.Response
    get_all_users (self)
```

pine.backend.auth.oauth

Module Contents

Classes

OAuthUser

OAuthModule

```
class pine.backend.auth.oauth.OAuthUser (data)
    Bases: pine.backend.models.AuthUser
```

```

    property id (self)
    property username (self)
    property display_name (self)
    property is_admin (self)
class pine.backend.auth.oauth.OAuthModule (app, bp)
    Bases: pine.backend.auth.bp.AuthModule
    abstract register_oauth (self, oauth, app)
    abstract get_login_form_button_text (self)
    is_flat (self) → bool
    can_manage_users (self) → bool
    get_login_form (self) → pine.backend.models.LoginForm
    login (self) → flask.Response
    authorize (self)

```

pine.backend.auth.password

Module Contents

Functions

hash_password(password: str) → str

check_password(password: str, hashed_password: str)

pine.backend.auth.password.**hash_password** (password: *str*) → str

pine.backend.auth.password.**check_password** (password: *str*, hashed_password: *str*)

pine.backend.auth.vegas

Module Contents

Classes

VegasAuthModule

```

class pine.backend.auth.vegas.VegasAuthModule (app, bp)
    Bases: pine.backend.auth.oauth.OAuthModule
    register_oauth (self, oauth, app)
    get_login_form_button_text (self)

```

Package Contents

Functions

login_required(view)

admin_required(view)

get_logged_in_user()

is_flat()

pine.backend.auth.**login_required**(view)
pine.backend.auth.**admin_required**(view)
pine.backend.auth.**module**
pine.backend.auth.**get_logged_in_user**()
pine.backend.auth.**is_flat**()

pine.backend.collections

This module contains the api methods required to interact with, organize, create, and display collections in the front-end and store the collections in the backend

Submodules

pine.backend.collections.bp

Module Contents

Functions

is_cached_last_collection(collection_id)

update_cached_last_collection(collection_id)

user_permissions_projection()

get_user_permissions(collection: dict) →
pine.backend.models.CollectionUserPermissions

get_user_permissions_by_id(collection_id:
str) → pine.backend.models.CollectionUserPermissions

get_user_permissions_by_ids(collection_ids:
Iterable[str]) → List[pine.backend.models.CollectionUserPermissions]

continues on next page

Table 10 – continued from previous page

<code>get_user_collections(archived, page)</code>	Return collections for the logged in user using pagination. Returns all collections if parameter “page” is “all”,
<code>get_unarchived_user_collections(page)</code>	Return unarchived user collections for the corresponding page value. Default value returns collections for all
<code>get_archived_user_collections(page)</code>	Return archived user collections for the corresponding page value. Default value returns collections for all
<code>archive_or_unarchive_collection(collection_id, archive)</code>	Set the “archived” boolean flag for the collection matching the provided collection_id.
<code>archive_collection(collection_id)</code>	Archive the collection matching the provided collection id
<code>unarchive_collection(collection_id)</code>	Unarchive the collection matching the provided collection id
<code>get_collection(collection_id)</code>	Return the collection object for the collection matching the provided collection id. This object has the fields:
<code>download_collection(collection_id)</code>	
<code>get_doc_and_overlap_ids(collection_id)</code>	Return lists of ids for overlapping and non-overlapping documents for the collection matching the provided
<code>add_annotator_to_collection(collection_id)</code>	
<code>add_viewer_to_collection(collection_id)</code>	
<code>add_label_to_collection(collection_id)</code>	
<code>get_overlap_ids(collection_id)</code>	Return the list of ids for overlapping documents for the collection matching the provided collection id.
<code>_upload_documents(collection, docs)</code>	
<code>create_collection()</code>	Create a new collection based upon the entries provided in the POST request’s associated form fields.
<code>_check_collection_and_get_image_dir(collection_id, path)</code>	
<code>get_static_collection_images(collection_id)</code>	
<code>get_collection_images(collection_id)</code>	
<code>get_collection_image(collection_id, path)</code>	
<code>get_collection_image_exists(collection_id, path)</code>	
<code>_path_split(path)</code>	
<code>_safe_path(path)</code>	
<code>endpoint_get_user_permissions(collection_id)</code>	
<code>_upload_collection_image_file(collection_id, path, image_file)</code>	
<code>post_collection_image(collection_id, path)</code>	

continues on next page

Table 10 – continued from previous page

<i>init_app</i> (app)	
<code>pine.backend.collections.bp.bp</code>	
<code>pine.backend.collections.bp.LOGGER</code>	
<code>pine.backend.collections.bp.DOCUMENTS_PER_TRANSACTION = 500</code>	
<code>pine.backend.collections.bp.LAST_COLLECTION_FOR_IMAGE</code>	
<code>pine.backend.collections.bp.is_cached_last_collection(collection_id)</code>	
<code>pine.backend.collections.bp.update_cached_last_collection(collection_id)</code>	
<code>pine.backend.collections.bp.user_permissions_projection()</code>	
<code>pine.backend.collections.bp.get_user_permissions(collection: dict)</code>	→ <i>pine.backend.models.CollectionUserPermissions</i>
<code>pine.backend.collections.bp.get_user_permissions_by_id(collection_id: str)</code>	→ <i>pine.backend.models.CollectionUserPermissions</i>
<code>pine.backend.collections.bp.get_user_permissions_by_ids(collection_ids: Iterable[str])</code>	→ List[models.CollectionUserPermissions]
<code>pine.backend.collections.bp.get_user_collections(archived, page)</code>	
Return collections for the logged in user using pagination. Returns all collections if parameter “page” is “all”, or the collections associated with the given page. Can return archived or un-archived collections based upon the “archived” flag. :param archived: Bool :param page: str :return: Response	
<code>pine.backend.collections.bp.get_unarchived_user_collections(page)</code>	
Return unarchived user collections for the corresponding page value. Default value returns collections for all pages. :param page: str :return: Response	
<code>pine.backend.collections.bp.get_archived_user_collections(page)</code>	
Return archived user collections for the corresponding page value. Default value returns collections for all pages. :param page: str :return: Response	
<code>pine.backend.collections.bp.archive_or_unarchive_collection(collection_id, archive)</code>	
Set the “archived” boolean flag for the collection matching the provided collection_id. :param collection_id: str :param archive: Bool :return: Response	
<code>pine.backend.collections.bp.archive_collection(collection_id)</code>	
Archive the collection matching the provided collection id :param collection_id: str :return: Response	
<code>pine.backend.collections.bp.unarchive_collection(collection_id)</code>	
Unarchive the collection matching the provided collection id :param collection_id: str :return: Response	
<code>pine.backend.collections.bp.get_collection(collection_id)</code>	
Return the collection object for the collection matching the provided collection id. This object has the fields: ‘creator_id’, ‘annotators’, ‘viewers’, ‘labels’, ‘metadata’, ‘archived’, and ‘configuration’. :param collection_id: str :return: Response	
<code>pine.backend.collections.bp.download_collection(collection_id)</code>	
<code>pine.backend.collections.bp.get_doc_and_overlap_ids(collection_id)</code>	
Return lists of ids for overlapping and non-overlapping documents for the collection matching the provided collection id. :param collection_id: str :return: tuple	
<code>pine.backend.collections.bp.add_annotator_to_collection(collection_id)</code>	

```

pine.backend.collections.bp.add_viewer_to_collection(collection_id)
pine.backend.collections.bp.add_label_to_collection(collection_id)
pine.backend.collections.bp.get_overlap_ids(collection_id)
    Return the list of ids for overlapping documents for the collection matching the provided collection id. :param
    collection_id: str :return: tuple
pine.backend.collections.bp._upload_documents(collection, docs)
pine.backend.collections.bp.create_collection()
    Create a new collection based upon the entries provided in the POST request's associated form fields. These
    fields include: collection - collection name overlap - ratio of overlapping documents. (0-1) with 0 being no
    overlap and 1 being every document has overlap, ex:

        .90 - 90% of documents overlap

    train_every - automatically train a new classifier after this many documents have been annotated pipelineId - the
    id value of the classifier pipeline associated with this collection (spacy, opennlp, corenlp) classifierParameters
    - optional parameters that adjust the configuration of the chosen classifier pipeline. archived - whether or not
    this collection should be archived. A collection can be created with documents listed in a csv file. Each new
    line in the csv represents a new document. The data of this csv can be passed to this method through the POST
    request's FILES field "file". used when creating a collection based on an uploaded csv file:

        csvTextCol - column of csv containing the text of the documents (default: 0) csvHasHeader - boolean
        for whether or not the csv file has a header row (default: False)

    A collection can also be created with a number of images through FILES fields "imageFileN" where N is an
    (ignored) index :return: information about the created collection
pine.backend.collections.bp._check_collection_and_get_image_dir(collection_id,
                                                                path)
pine.backend.collections.bp.get_static_collection_images(collection_id)
pine.backend.collections.bp.get_collection_images(collection_id)
pine.backend.collections.bp.get_collection_image(collection_id, path)
pine.backend.collections.bp.get_collection_image_exists(collection_id, path)
pine.backend.collections.bp._path_split(path)
pine.backend.collections.bp._safe_path(path)
pine.backend.collections.bp.endpoint_get_user_permissions(collection_id)
pine.backend.collections.bp._upload_collection_image_file(collection_id, path, im-
                                                         age_file)
pine.backend.collections.bp.post_collection_image(collection_id, path)
pine.backend.collections.bp.init_app(app)

```

Package Contents

Functions

user_permissions_projection()

get_user_permissions(collection: dict) →
pine.backend.models.CollectionUserPermissions

get_user_permissions_by_id(collection_id:
str) → pine.backend.models.CollectionUserPermissions

get_user_permissions_by_ids(collection_ids:
Iterable[str]) → List[pine.backend.models.CollectionUserPermissions]

pine.backend.collections.user_permissions_projection()

pine.backend.collections.get_user_permissions(collection: dict) →
pine.backend.models.CollectionUserPermissions

pine.backend.collections.get_user_permissions_by_id(collection_id: str) →
pine.backend.models.CollectionUserPermissions

pine.backend.collections.get_user_permissions_by_ids(collection_ids: Iterable[str]) →
List[pine.backend.models.CollectionUserPermissions]

pine.backend.data

Submodules

pine.backend.data.bp

Module Contents

Functions

init_app(app)

pine.backend.data.bp.init_app(app)

pine.backend.data.service

Module Contents

Classes

PerformanceHistory

Functions

<i>_standardize_path</i> (path, *additional_paths)	
<i>url</i> (path, *additional_paths)	Returns a complete URL for the given eve-relative path(s).
<i>where_params</i> (where)	Returns a “where” parameters object that can be passed to eve.
<i>params</i> (params)	Returns a parameters object that can be passed to eve.
<i>get</i> (path, **kwargs)	Wraps requests.get for the given eve-relative path.
<i>post</i> (path, **kwargs)	Wraps requests.post for the given eve-relative path.
<i>put</i> (path, **kwargs)	Wraps requests.put for the given eve-relative path.
<i>delete</i> (path, **kwargs)	Wraps requests.delete for the given eve-relative path.
<i>patch</i> (path, **kwargs)	Wraps requests.patch for the given eve-relative path.
<i>get_item_by_id</i> (path, item_id, params={})	Gets a single item by the given ID.
<i>get_all_versions_of_item_by_id</i> (path, item_id, params={})	
<i>get_all</i> (path, params={})	Returns ALL database items, using pagination if needed. This returns the “normal” eve
<i>get_all_items</i> (path, params={})	Returns ALL database items, using pagination if needed.
<i>convert_response</i> (requests_response)	
<i>remove_eve_fields</i> (obj, re-move_timestamps=True, remove_versions=True)	
<i>remove_nonupdatable_fields</i> (obj)	

```
pine.backend.data.service.logger
```

```
class pine.backend.data.service.PerformanceHistory
```

```
    Bases: object
```

```
    pformat (self, **kwargs)
```

```
    pprint (self)
```

```
    add (self, rest_type, path, response)
```

```
pine.backend.data.service.PERFORMANCE_HISTORY
```

```
pine.backend.data.service._standardize_path (path, *additional_paths)
```

```
pine.backend.data.service.url (path, *additional_paths)
```

```
    Returns a complete URL for the given eve-relative path(s).
```

Parameters

- **path** – str: eve-relative path (e.g. “collections” or [“collections”, id])
- **additional_paths** – str[]: any additional paths to append

Returns str url

`pine.backend.data.service.where_params` (*where*)

Returns a “where” parameters object that can be passed to eve.

Eve requires that dict parameters be serialized as JSON.

Parameters `where` – dict: dictionary of “where” params to pass to eve

Returns dict “where” params

`pine.backend.data.service.params` (*params*)

Returns a parameters object that can be passed to eve.

Eve requires that dict parameters be serialized as JSON.

Parameters `where` – dict: dictionary of “where” params to pass to eve

Returns dict “where” params

`pine.backend.data.service.get` (*path*, ***kwargs*)

Wraps requests.get for the given eve-relative path.

Parameters

- **path** – str: eve-relative path (e.g. “collections” or [“collections”, id])
- ****kwargs** – dict: any additional arguments to pass to requests.get

Returns requests.Response

`pine.backend.data.service.post` (*path*, ***kwargs*)

Wraps requests.post for the given eve-relative path.

Parameters

- **path** – str: eve-relative path (e.g. “collections” or [“collections”, id])
- ****kwargs** – dict: any additional arguments to pass to requests.post

Returns requests.Response

`pine.backend.data.service.put` (*path*, ***kwargs*)

Wraps requests.put for the given eve-relative path.

Parameters

- **path** – str: eve-relative path (e.g. “collections” or [“collections”, id])
- ****kwargs** – dict: any additional arguments to pass to requests.put

Returns requests.Response

`pine.backend.data.service.delete` (*path*, ***kwargs*)

Wraps requests.delete for the given eve-relative path.

Parameters

- **path** – str: eve-relative path (e.g. “collections” or [“collections”, id])
- ****kwargs** – dict: any additional arguments to pass to requests.delete

Returns requests.Response

`pine.backend.data.service.patch` (*path*, ***kwargs*)

Wraps requests.patch for the given eve-relative path.

Parameters

- **path** – str: eve-relative path (e.g. “collections” or [“collections”, id])

- ****kwargs** – dict: any additional arguments to pass to `requests.patch`

Returns `requests.Response`

```
pine.backend.data.service.get_item_by_id(path, item_id, params={})
```

Gets a single item by the given ID.

Parameters

- **path** – str: eve-relative path (e.g. “collections”)
- **item_id** – str: item ID
- **params** – dict: optional additional parameters to send in with GET

Returns the item as a dict

```
pine.backend.data.service.get_all_versions_of_item_by_id(path, item_id,
                                                         params={})
```

```
pine.backend.data.service.get_all(path, params={})
```

Returns ALL database items, using pagination if needed. This returns the “normal” eve JSON with “_items”, “_meta”, etc.

Parameters

- **path** – str: eve-relative path (e.g. “collections”)
- **params** – dict: optional additional parameters to send in with GET

Returns an eve collections dict with, e.g., `_items`

```
pine.backend.data.service.get_all_items(path, params={})
```

Returns ALL database items, using pagination if needed.

Parameters

- **path** – str: eve-relative path (e.g. “collections”)
- **params** – dict: optional additional parameters to send in with GET

Returns the items as a list of dicts

```
pine.backend.data.service.convert_response(requests_response)
```

```
pine.backend.data.service.remove_eve_fields(obj, remove_timestamps=True,
                                             remove_versions=True)
```

```
pine.backend.data.service.remove_nonupdatable_fields(obj)
```

pine.backend.data.users

Module Contents

Functions

```
get_all_users()
```

```
get_user(user_id)
```

continues on next page

Table 15 – continued from previous page

`get_user_by_email(email)`

`get_user_by_id_or_email(username)`

`get_user_details(user_id)`

`update_user(user_id: str, details:
pine.backend.models.UserDetails)`

`print_users_command()`

`add_admin_command(user_id, username, pass-
word)`

`set_user_password_by_id(user_id, password)`

`set_user_password(username, password)`

`reset_user_passwords()`

```
pine.backend.data.users.get_all_users()  
pine.backend.data.users.get_user(user_id)  
pine.backend.data.users.get_user_by_email(email)  
pine.backend.data.users.get_user_by_id_or_email(username)  
pine.backend.data.users.get_user_details(user_id)  
pine.backend.data.users.update_user(user_id: str, details: pine.backend.models.UserDetails)  
pine.backend.data.users.print_users_command()  
pine.backend.data.users.add_admin_command(user_id, username, password)  
pine.backend.data.users.set_user_password_by_id(user_id, password)  
pine.backend.data.users.set_user_password(username, password)  
pine.backend.data.users.reset_user_passwords()
```

pine.backend.documents**Submodules****pine.backend.documents.bp****Module Contents****Functions**

`_document_user_can_projection()`

continues on next page

Table 16 – continued from previous page

<code>get_collection_ids_for(document_ids) → set</code>
<code>get_user_permissions(document: dict) → pine.backend.models.CollectionUserPermissions</code>
<code>get_user_permissions_by_id(document_id: str) → pine.backend.models.CollectionUserPermissions</code>
<code>get_user_permissions_by_ids(document_ids: Iterable[str]) → List[models.CollectionUserPermissions]</code>
<code>get_document(doc_id)</code>
<code>count_documents_in_collection(col_id)</code>
<code>get_all_documents_in_collection(col_id)</code>
<code>get_paginated_documents_in_collection(collection_id)</code>
<code>_check_documents(documents) → dict</code>
<code>add_document()</code>
<code>endpoint_get_user_permissions(doc_id)</code>
<code>update_metadata(doc_id)</code>
<code>init_app(app)</code>
<code>pine.backend.documents.bp.bp</code>
<code>pine.backend.documents.bp._document_user_can_projection()</code>
<code>pine.backend.documents.bp.get_collection_ids_for(document_ids) → set</code>
<code>pine.backend.documents.bp.get_user_permissions(document: dict) → pine.backend.models.CollectionUserPermissions</code>
<code>pine.backend.documents.bp.get_user_permissions_by_id(document_id: str) → pine.backend.models.CollectionUserPermissions</code>
<code>pine.backend.documents.bp.get_user_permissions_by_ids(document_ids: It- erable[str]) → List[models.CollectionUserPermissions]</code>
<code>pine.backend.documents.bp.get_document(doc_id)</code>
<code>pine.backend.documents.bp.count_documents_in_collection(col_id)</code>
<code>pine.backend.documents.bp.get_all_documents_in_collection(col_id)</code>
<code>pine.backend.documents.bp.get_paginated_documents_in_collection(collection_id)</code>
<code>pine.backend.documents.bp._check_documents(documents) → dict</code>
<code>pine.backend.documents.bp.add_document()</code>
<code>pine.backend.documents.bp.endpoint_get_user_permissions(doc_id)</code>
<code>pine.backend.documents.bp.update_metadata(doc_id)</code>
<code>pine.backend.documents.bp.init_app(app)</code>

Package Contents

Functions

get_collection_ids_for(document_ids) → set

get_user_permissions(document: dict) →
pine.backend.models.CollectionUserPermissions

get_user_permissions_by_id(document_id:
str) → pine.backend.models.CollectionUserPermissions

get_user_permissions_by_ids(document_ids:
Iterable[str]) → List[models.CollectionUserPermissions]

pine.backend.documents.**get_collection_ids_for**(document_ids) → setpine.backend.documents.**get_user_permissions**(document: dict) →
pine.backend.models.CollectionUserPermissionspine.backend.documents.**get_user_permissions_by_id**(document_id: str) →
pine.backend.models.CollectionUserPermissionspine.backend.documents.**get_user_permissions_by_ids**(document_ids: Iterable[str]) →
List[models.CollectionUserPermissions]

pine.backend.job_manager

Submodules

pine.backend.job_manager.service

Module Contents

Classes

ServiceManager

type default_handler callable

pine.backend.job_manager.service.**config**pine.backend.job_manager.service.**logger****class** pine.backend.job_manager.service.**ServiceManager**(default_handler=None)
Bases: object**r_pool****r_conn****redis_key_prefix****redis_reg_key_prefix**

```

redis_channels_key
redis_channel_ttl_key_prefix
redis_work_queue_key_prefix
redis_work_mutex_key_prefix
redis_handler_mutex_key_prefix
redis_reg_key_ttl
redis_channels_key_ttl
redis_work_queue_key_ttl
redis_work_mutex_key_ttl
redis_handler_mutex_key_ttl
handler_timeout
registration_worker_name = registration_worker
processing_worker_name = processing_worker
channel_worker_name = channel_worker
shutdown_channel = shutdown
registration_channel = registration
reserved_channels

classmethod get_registered_channels (cls, include_ttl=False)
    Get list of registered channels, with registration time if requested. :type include_ttl: bool :rtype: list[str] |
    dict[str, datetime]

classmethod get_registered_service_details (cls, service_name=None)
    Get registration details of a service. :type service_name: str :rtype: None | dict

classmethod get_registered_services (cls, include_details=True)
    Get list of registered services and registration body if requested. :type include_details: bool :rtype: list[str]
    | list[dict]

classmethod send_service_request (cls, service_name, data, job_id=None, encoder=None)
    Queue's a job for the requested service. :type service_name: str :type data: dict :type job_id: str :type
    encoder: json.JSONEncoder :rtype: None | dict

start_listeners (self)
    Starts all the workers.

stop_listeners (self)
    Stops all the workers.

_start_registration_listener (self)
    Starts the registration worker. :rtype: bool

_start_processing_listeners (self)
    Starts the processing workers. :rtype: bool

_start_channel_watchdog (self)
    Starts the channel watchdog workers in an asyncio-only thread. It monitors the channel TTL's. :rtype:
    bool

_stop_channel_watchdog (self)
    Stops the channel watchdog workers in an asyncio-only thread. :rtype: bool

```

`_registration_listener` (*self*)

Registration Listener Implementation.

`async _channel_watchdog` (*self*)

Channel Watchdog Implementation. In asyncio, it monitors the channel-ttl keys and the channels SET to expire registered services as needed. The other functionality is to register new channels as they're added to the pubsub in the processing listener.

`static _thread_killer` (*thread_id*)

`_processing_listener_handler_wrapper` (*self, job_id, job_type, job_queue, job_data*)

`_processing_listener` (*self*)

Processing Listener Implementation. Runs a handler when a processing message gets send over an already registered channel.

`pine.backend.job_manager.service.sm`

`pine.backend.pineiaa`

Subpackages

`pine.backend.pineiaa.bratiaa`

Submodules

`pine.backend.pineiaa.bratiaa.agree`

Module Contents

Classes

Document

F1Agreement

Functions

input_generator(json_list)

compute_f1(tp, fp, fn)

compute_f1_agreement(annotators, documents,
labels, token_func=None, eval_func=None)

iaa_report(f1_agreement, precision=3)

`pine.backend.pineiaa.bratiaa.agree.Annotation`

`pine.backend.pineiaa.bratiaa.agree.AnnFile`


```

pine.backend.pineiaa.bratiaa.agree.LOGGER
class pine.backend.pineiaa.bratiaa.agree.Document (txt, doc_id)

    __slots__ = ['ann_files', 'txt', 'doc_id']
pine.backend.pineiaa.bratiaa.agree.input_generator (json_list)
pine.backend.pineiaa.bratiaa.agree.compute_f1 (tp, fp, fn)
class pine.backend.pineiaa.bratiaa.agree.F1Agreement (annotators, documents, labels,
                                                    eval_func=exact_match_instance_evaluation,
                                                    token_func=None)

    property annotators (self)
    property documents (self)
    property labels (self)
    _compute_tp_fp_fn (self, documents)
    _increment_counts (self, annotations, pair, doc, kind)
    mean_sd_per_label (self)
        Mean and standard deviation of all annotator combinations' F1 scores by label.
    mean_sd_per_document (self)
        Mean and standard deviation of all annotator combinations' F1 scores per document.
    mean_sd_total (self)
        Mean and standard deviation of all annotator combinations' F1 scores.
    mean_sd_per_label_one_vs_rest (self, annotator)
        Mean and standard deviation of all annotator combinations' F1 scores involving given annotator per label.
    mean_sd_total_one_vs_rest (self, annotator)
        Mean and standard deviation of all annotator combinations' F1 scores involving given annotator.
    _pairs_involving (self, annotator)
    static _mean_sd (f1_pairs)
        Mean and standard deviation along first axis.
    static print_table (row_label_header, row_labels, avg, stddev, precision=3)
    compute_total_f1_matrix (self)
        Returns (n x n) matrix, where n is the number of annotators, containing pair-wise total F1 scores between
        all annotators.

        By definition, the matrix is symmetric and F1 = 1 on the main diagonal.
    draw_heatmap (self, out_path)
        Draws heatmap based on square matrix of F1 scores.
pine.backend.pineiaa.bratiaa.agree.compute_f1_agreement (annotators, documents,
                                                         labels, token_func=None,
                                                         eval_func=None)
pine.backend.pineiaa.bratiaa.agree.iaa_report (f1_agreement, precision=3)

```

```
pine.backend.pineiaa.bratiaa.agree_cli
```

Module Contents

Functions

```
parse_args()
```

```
main()
```

```
pine.backend.pineiaa.bratiaa.agree_cli.parse_args()
```

```
pine.backend.pineiaa.bratiaa.agree_cli.main()
```

pine.backend.pineiaa.bratiaa.evaluation

Functions for computing the difference between two sets of annotations.

Module Contents

Functions

```
exact_match_instance_evaluation(ann_list_1,  
ann_list2, tokens=None)
```

<pre>exact_match_token_evaluation(ann_list_1, ann_list_2, tokens=None)</pre>	Annotations are split into token-sized bits before true positives, false positives and false negatives are computed.
--	--

```
counter2list(c)
```

<pre>_read_token_annotations(ann_list, tokens)</pre>	Yields a new annotation for each token overlapping with an annotation. If annotations are overlapping each other,
--	---

```
pine.backend.pineiaa.bratiaa.evaluation.Annotation
```

```
pine.backend.pineiaa.bratiaa.evaluation.exact_match_instance_evaluation(ann_list_1,  
ann_list2,  
to-  
kens=None)
```

```
pine.backend.pineiaa.bratiaa.evaluation.exact_match_token_evaluation(ann_list_1,  
ann_list_2,  
to-  
kens=None)
```

Annotations are split into token-sized bits before true positives, false positives and false negatives are computed.

Sub-token annotations are expanded to full tokens. Long annotations will influence the results more than short annotations. Boundary errors for adjacent annotations with the same label are ignored!

```
pine.backend.pineiaa.bratiaa.evaluation.counter2list(c)
```

```
pine.backend.pineiaa.bratiaa.evaluation._read_token_annotations(ann_list,  
                                                                tokens)
```

Yields a new annotation for each token overlapping with an annotation. If annotations are overlapping each other, there will be multiple annotations for a single token.

```
pine.backend.pineiaa.bratiaa.iaa_service
```

Module Contents

Functions

```
get_items(resource)
```

```
get_all_items(query)
```

```
get_doc_annotations(collection_id,          ex-  
                    clude=None)
```

```
fix_num_for_json(number)
```

```
getIAAReportForCollection(collection_id)
```

```
pine.backend.pineiaa.bratiaa.iaa_service.EVE_HEADERS
```

```
pine.backend.pineiaa.bratiaa.iaa_service.get_items(resource)
```

```
pine.backend.pineiaa.bratiaa.iaa_service.get_all_items(query)
```

```
pine.backend.pineiaa.bratiaa.iaa_service.get_doc_annotations(collection_id,  
                                                            exclude=None)
```

```
pine.backend.pineiaa.bratiaa.iaa_service.fix_num_for_json(number)
```

```
pine.backend.pineiaa.bratiaa.iaa_service.getIAAReportForCollection(collection_id)
```

```
pine.backend.pineiaa.bratiaa.utils
```

Module Contents

Classes

```
TokenOverlap
```

Data structure for quick lookup of tokens overlapping with given span.

Functions

tokenize(text)

read(path, encoding=ENCODING, newline='\r\n', mode='r')

`pine.backend.pineiaa.bratiaa.utils.ENCODING = utf-8``pine.backend.pineiaa.bratiaa.utils.TOKEN``pine.backend.pineiaa.bratiaa.utils.tokenize(text)``pine.backend.pineiaa.bratiaa.utils.read(path, encoding=ENCODING, newline='\r\n', mode='r')`**class** `pine.backend.pineiaa.bratiaa.utils.TokenOverlap(text, tokens)`

Data structure for quick lookup of tokens overlapping with given span. Assumes that the provided list of tokens is sorted by indices!

static `compute_mapping(text_length, tokens)`**overlapping_tokens** (*self*, *start*, *end*)

Package Contents

Classes

F1Agreement

Document

Functions

compute_f1_agreement(annotators, documents, labels, token_func=None, eval_func=None)

iaa_report(f1_agreement, precision=3)

input_generator(json_list)

exact_match_instance_evaluation(ann_list_1, ann_list_2, tokens=None)

exact_match_token_evaluation(ann_list_1, ann_list_2, tokens=None) Annotations are split into token-sized bits before true positives, false positives and false negatives are computed.

tokenize(text)

`pine.backend.pineiaa.bratiaa.compute_f1_agreement(annotators, documents, labels, token_func=None, eval_func=None)`

```

pine.backend.pineiaa.bratiaa.iaa_report (f1_agreement, precision=3)
pine.backend.pineiaa.bratiaa.AnnFile
class pine.backend.pineiaa.bratiaa.F1Agreement (annotators, documents, labels,
                                              eval_func=exact_match_instance_evaluation,
                                              token_func=None)

    property annotators (self)
    property documents (self)
    property labels (self)
    _compute_tp_fp_fn (self, documents)
    _increment_counts (self, annotations, pair, doc, kind)
    mean_sd_per_label (self)
        Mean and standard deviation of all annotator combinations' F1 scores by label.
    mean_sd_per_document (self)
        Mean and standard deviation of all annotator combinations' F1 scores per document.
    mean_sd_total (self)
        Mean and standard deviation of all annotator combinations' F1 scores.
    mean_sd_per_label_one_vs_rest (self, annotator)
        Mean and standard deviation of all annotator combinations' F1 scores involving given annotator per label.
    mean_sd_total_one_vs_rest (self, annotator)
        Mean and standard deviation of all annotator combinations' F1 scores involving given annotator.
    _pairs_involving (self, annotator)
    static _mean_sd (f1_pairs)
        Mean and standard deviation along first axis.
    static print_table (row_label_header, row_labels, avg, stddev, precision=3)
    compute_total_f1_matrix (self)
        Returns (n x n) matrix, where n is the number of annotators, containing pair-wise total F1 scores between
        all annotators.

        By definition, the matrix is symmetric and F1 = 1 on the main diagonal.
    draw_heatmap (self, out_path)
        Draws heatmap based on square matrix of F1 scores.
class pine.backend.pineiaa.bratiaa.Document (txt, doc_id)

    __slots__ = ['ann_files', 'txt', 'doc_id']
pine.backend.pineiaa.bratiaa.input_generator (json_list)
pine.backend.pineiaa.bratiaa.exact_match_instance_evaluation (ann_list_1,
                                                            ann_list2, to-
                                                            kens=None)
pine.backend.pineiaa.bratiaa.exact_match_token_evaluation (ann_list_1, ann_list_2,
                                                         tokens=None)
    Annotations are split into token-sized bits before true positives, false positives and false negatives are computed.
    Sub-token annotations are expanded to full tokens. Long annotations will influence the results more than short
    annotations. Boundary errors for adjacent annotations with the same label are ignored!

```

```
pine.backend.pineiaa.bratiaa.Annotation
pine.backend.pineiaa.bratiaa.tokenize(text)
```

Submodules

pine.backend.pineiaa.bp

Module Contents

Functions

```
get_current_report(collection_id)
get_iaa_report_by_collection_id(collection_id)
create_iaa_report_by_collection_id(collection_id)
init_app(app)
```

```
pine.backend.pineiaa.bp.logger
pine.backend.pineiaa.bp.bp
pine.backend.pineiaa.bp.get_current_report(collection_id)
pine.backend.pineiaa.bp.get_iaa_report_by_collection_id(collection_id)
pine.backend.pineiaa.bp.create_iaa_report_by_collection_id(collection_id)
pine.backend.pineiaa.bp.init_app(app)
```

pine.backend.pipelines

Submodules

pine.backend.pipelines.bp

Module Contents

Functions

```
get_pipelines()
get_pipeline_by_id(pipeline_id)
_get_collection_classifier(collection_id)
```

continues on next page

Table 29 – continued from previous page

<code>get_collection_classifier(collection_id)</code>
<code>_get_classifier_metrics(classifier_id)</code>
<code>get_metrics()</code>
<code>get_classifier_metrics(classifier_id)</code>
<code>_get_classifier(classifier_id)</code>
<code>_get_next_instance(classifier_id)</code>
<code>get_next_by_classifier(classifier_id)</code>
<code>advance_to_next_document_by_classifier(classifier_id, document_id)</code>
<code>predict()</code>
<code>test_redis()</code>
<code>init_app(app)</code>

```

pine.backend.pipelines.bp.logger
pine.backend.pipelines.bp.service_manager
pine.backend.pipelines.bp.bp
pine.backend.pipelines.bp.classifier_dict
pine.backend.pipelines.bp.classifier_pipelines
pine.backend.pipelines.bp.get_pipelines()
pine.backend.pipelines.bp.get_pipeline_by_id(pipeline_id)
pine.backend.pipelines.bp._get_collection_classifier(collection_id)
pine.backend.pipelines.bp.get_collection_classifier(collection_id)
pine.backend.pipelines.bp._get_classifier_metrics(classifier_id)
pine.backend.pipelines.bp.get_metrics()
pine.backend.pipelines.bp.get_classifier_metrics(classifier_id)
pine.backend.pipelines.bp._get_classifier(classifier_id)
pine.backend.pipelines.bp._get_next_instance(classifier_id)
pine.backend.pipelines.bp.get_next_by_classifier(classifier_id)
pine.backend.pipelines.bp.advance_to_next_document_by_classifier(classifier_id,
                                                                    docu-
                                                                    ment_id)

pine.backend.pipelines.bp.predict()
pine.backend.pipelines.bp.test_redis()
pine.backend.pipelines.bp.init_app(app)

```

pine.backend.shared

Submodules

pine.backend.shared.config

Module Contents

Classes

BaseConfig

TestConfig

ConfigBuilder

pine.backend.shared.config.LOGGER

class pine.backend.shared.config.**BaseConfig**(root_dir=None)

Bases: *object*

ROOT_DIR

BASE_DIR

BASE_CFG_FILE = config.yaml

BASE_ENV_PREFIX = AL_

DEBUG = True

TESTING = False

LOGGER_NAME

LOGGER_DIR = logs

LOGGER_FILE = debug.log

LOGGER_LEVEL

EVE_HOST = localhost

EVE_PORT = 5001

REDIS_HOST = localhost

REDIS_PORT = 6379

REDIS_USR

REDIS_PWD

REDIS_DBNUM = 0

REDIS_PREFIX = AL:

REDIS_EXPIRE = 3600

SCHEDULER_REGISTRATION_TIMEOUT


```

SCHEDULER_HANDLER_TIMEOUT
SCHEDULER_QUEUE_TIMEOUT
SERVICE_REGISTRATION_CHANNEL = registration
SERVICE_REGISTRATION_FREQUENCY = 60
SERVICE_LISTENING_FREQUENCY = 1
SERVICE_HANDLER_TIMEOUT = 3600
SERVICE_LIST
DATASETS_LOCAL_DIR = D:\Data\DEEPCATT2_DB_DATA\datasets
MODELS_LOCAL_DIR = D:\\Data\\DEEPCATT2_DB_DATA\\Models
classmethod _get_config_var_paths (cls, root_dict=None)
classmethod _process_paths (cls, alt_path=None)
classmethod _process_file_cfg (cls)
classmethod _process_env_vars (cls)
classmethod as_dict (cls)
    Return type dict
classmethod as_attr_dict (cls)
    Return type munch.Munch | dict
static _try_cast (value, _type, _default=None)
static _str2bool (_str, _default=None)
class pine.backend.shared.config.TestConfig (root_dir=None)
    Bases: pine.backend.shared.config.BaseConfig
class pine.backend.shared.config.ConfigBuilder
    Bases: object
    __env_cfg_variable = BUILDER_CFG_PROFILE
    __current_config_instance
    __current_config_instance_name
    __current_config_instance_print = False
    __arg_parser
    static __get_configs ()
        :rtype list[Callable[... , BaseConfig]]
    static get_config_names ()
        Return type list[str]
    classmethod get_arg_parser (cls)
        Return type ArgumentParser
    classmethod init_config (cls, config_name=None, config_base=None, enable_terminal=True,
        as_attr_dict=True)

```

```
classmethod get_config(cls, config_name=None, config_base=None, enable_terminal=True,
                        as_attr_dict=True)
```

Return type *BaseConfig*

```
classmethod set_config(cls, config_name=None, config_base=None, enable_terminal=True,
                        as_attr_dict=True)
```

```
classmethod __parse_terminal_config(cls)
```

Return type *argparse.Namespace*

`pine.backend.shared.transform`

Module Contents

Functions

<code>transform_module_by_config(module_ref, config_ref, config_prefix=None)</code>	Transforms a given module's properties based on ConfigBuilder Values.
---	---

`pine.backend.shared.transform.transform_module_by_config(module_ref, config_ref, config_prefix=None)`

Transforms a given module's properties based on ConfigBuilder Values. The prefix can be used to avoid blindly changing values and target a subset of matching values in config_ref. :type module_ref: ModuleType :type config_ref: dict :type config_prefix: str

Submodules

`pine.backend.app`

Module Contents

Functions

`handle_error(e)`

`handle_uncaught_exception(e)`

`create_app(test_config=None)`

`pine.backend.app.VERSION`

`pine.backend.app.LOGGER`

`pine.backend.app.handle_error(e)`

`pine.backend.app.handle_uncaught_exception(e)`

`pine.backend.app.create_app(test_config=None)`

pine.backend.config

Module Contents

```
pine.backend.config.SECRET_KEY = Cq13XII=%
pine.backend.config.DEBUG = True
pine.backend.config.EVE_SERVER
pine.backend.config.REDIS_SERVER
pine.backend.config.REDIS_PORT
pine.backend.config.AUTH_MODULE
pine.backend.config.AUTH_MODULE = vegas
pine.backend.config.VEGAS_CLIENT_SECRET
pine.backend.config.DOCUMENT_IMAGE_DIR
```

pine.backend.cors

Module Contents

Functions

not_found(e)

init_app(app)

```
pine.backend.cors.not_found(e)
pine.backend.cors.init_app(app)
```

pine.backend.log

Module Contents

Classes

Action

Generic enumeration.

Functions

`setup_logging()`

`get_flask_request_info()`

`get_flask_logged_in_user()`

`access_flask_login()`

`access_flask_logout(user: dict)`

`access_flask_add_collection(collection:
dict)`

`access_flask_view_document(document:
dict)`

`access_flask_add_document(document: dict)`

`access_flask_add_documents(documents:
List[dict])`

`access_flask_annotate_document(annotation)`

`access_flask_annotate_documents(annotations:
List[dict])`

`access(action, user, request_info, message, **ex-
tra_info)`

```
pine.backend.log.CONFIG_FILE_ENV = PINE_LOGGING_CONFIG_FILE
```

```
pine.backend.log.ACCESS_LOGGER_NAME = pine.access
```

```
pine.backend.log.ACCESS_LOGGER
```

```
class pine.backend.log.Action
```

```
    Bases: enum.Enum
```

```
    Generic enumeration.
```

```
    Derive from this class to define new enumerations.
```

```
    LOGIN
```

```
    LOGOUT
```

```
    CREATE_COLLECTION
```

```
    VIEW_DOCUMENT
```

```
    ADD_DOCUMENT
```

```
    ADD_DOCUMENTS
```

```
    ANNOTATE_DOCUMENT
```

```
    ANNOTATE_DOCUMENTS
```

```
pine.backend.log.setup_logging()
```

```
pine.backend.log.get_flask_request_info()
```

```

pine.backend.log.get_flask_logged_in_user()
pine.backend.log.access_flask_login()
pine.backend.log.access_flask_logout(user: dict)
pine.backend.log.access_flask_add_collection(collection: dict)
pine.backend.log.access_flask_view_document(document: dict)
pine.backend.log.access_flask_add_document(document: dict)
pine.backend.log.access_flask_add_documents(documents: List[dict])
pine.backend.log.access_flask_annotate_document(annotation)
pine.backend.log.access_flask_annotate_documents(annotations: List[dict])
pine.backend.log.access(action, user, request_info, message, **extra_info)

```

pine.backend.models

Module Contents

Classes

<i>LoginFormFieldType</i>	Generic enumeration.
<i>LoginFormField</i>	
<i>LoginForm</i>	
<i>AuthUser</i>	
<i>UserDetails</i>	
<i>CollectionUserPermissions</i>	Collection permissions for a user as a dictionary of boolean flags.

class pine.backend.models.LoginFormFieldType

Bases: `enum.Enum`

Generic enumeration.

Derive from this class to define new enumerations.

TEXT = text

PASSWORD = password

class pine.backend.models.LoginFormField(name: *str*, display: *str*, field_type: `pine.backend.models.LoginFormFieldType`)

Bases: `object`

to_dict(self)

class pine.backend.models.LoginForm(fields: *list*, button_text: *str*)

Bases: `object`

to_dict(self)

```
class pine.backend.models.AuthUser
    Bases: object

    property id(self)
    property username(self)
    property display_name(self)
    property is_admin(self)
    to_dict(self)

class pine.backend.models.UserDetails(first_name, last_name, description)
    Bases: object

    to_dict(self)

class pine.backend.models.CollectionUserPermissions(view=False, annotate=False,
                                                    add_documents=False,
                                                    add_images=False,      mod-
                                                    ify_users=False,          mod-
                                                    ify_labels=False,          mod-
                                                    ify_document_metadata=False,
                                                    download_data=False,
                                                    archive=False)

    Bases: object

    Collection permissions for a user as a dictionary of boolean flags.

    view :bool
        Whether the user can view the collection and documents. :type: bool

    annotate :bool
        Whether the user can annotate collection documents. :type: bool

    add_documents :bool
        Whether the user can add documents to the collection. :type: bool

    add_images :bool
        Whether the user can add images to the collection. :type: bool

    modify_users :bool
        Whether the user can modify the list of viewers/annotators for the collection. :type: bool

    modify_labels :bool
        Whether the user can modify the list of labels for the collection. :type: bool

    modify_document_metadata :bool
        Whether the user can modify document metadata (such as changing the image). :type: bool

    download_data :bool
        Whether the user can download the collection data :type: bool

    archive :bool
        Whether the user can archive or unarchive the collection. :type: bool

    to_dict(self) → dict
        Returns a dict version of this object for conversion to JSON.

        Returns a dict version of this object for conversion to JSON

        Return type dict
```

Package Contents

Functions

`create_app(test_config=None)`

`pine.backend.create_app(test_config=None)`

`pine.backend.VERSION`

`pine.backend.__version__`

`pine.client`

PINE client module.

Submodules

`pine.client.client`

PINE client classes module.

Module Contents

Classes

<code>BaseClient</code>	Base class for a client using a REST interface.
<code>EveClient</code>	A client to access Eve and, optionally, its underlying MongoDB instance.
<code>PineClient</code>	A client to access PINE (more specifically: the back-end).
<code>LocalPineClient</code>	A client for a local PINE instance, including an <code>EveClient</code> .

Functions

`_standardize_path(path: str, *additional_paths: List[str]) → List[str]` Standardize path(s) into a list of path components.

`pine.client.client._standardize_path(path: str, *additional_paths: List[str]) → List[str]`
Standardize path(s) into a list of path components.

Parameters

- **path** (*str*) – relative path, e.g. "users"
- ***additional_paths** (*list(str)*, *optional*) – any additional path components

in a list

Returns the standardized path components in a list

Return type `list(str)`

class `pine.client.client.BaseClient` (*base_uri: str, name: str = None, verify_ssl: bool = True*)

Bases: `object`

Base class for a client using a REST interface.

Constructor.

Parameters

- **base_uri** (*str*) – the base URI for the server, e.g. "http://localhost:5000"
- **name** (*str, optional*) – optional human-readable name for the server, defaults to None
- **verify_ssl** (*bool, optional*) – whether to verify SSL/HTTPs calls; do not turn this off unless you are fully aware of the security consequences

__metaclass__

base_uri :`str`

The server's base URI.

Type `str`

session :`requests.Session`

The currently open session, or None.

Type `requests.Session`

verify_ssl :`bool`

Whether to verify SSL/HTTPS calls. If you turn this off you should be fully aware of the security consequences of such.

Type `bool`

abstract is_valid (*self*) → `bool`

Returns whether this client and its connection(s) are valid.

Returns whether this client and its connection(s) are valid

Return type `bool`

uri (*self, path: str, *additional_paths: List[str]*) → `str`

Makes a complete URI from the given path(s).

Parameters

- **path** (*str*) – relative path, e.g. "users"
- ***additional_paths** (*list(str), optional*) – any additional path components

Returns the complete, standardized URI including the base URI, e.g. "http://localhost:5000/users"

Return type `str`

_req (*self, method: str, path: str, *additional_paths: List[str], **kwargs*) → `requests.Response`

Makes a `requests` call, checks for errors, and returns the response.

Parameters

- **method** (*str*) – REST method ("get", "post", etc.)
- **path** (*str*) – relative path, e.g. "users"
- ***additional_paths** (*list(str)*, *optional*) – any additional path components
- ****kwargs** (*dict*) – any additional kwargs to send to `requests`

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the `requests` `requests.Response` object

Return type `requests.Response`

get (*self*, *path: str*, **additional_paths: List[str]*, ***kwargs*) → `requests.Response`

Makes a `requests` GET call, checks for errors, and returns the response.

Parameters

- **path** (*str*) – relative path, e.g. "users"
- ***additional_paths** (*list(str)*, *optional*) – any additional path components
- ****kwargs** (*dict*) – any additional kwargs to send to `requests`

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the `requests` `Response` object

Return type `requests.Response`

put (*self*, *path: str*, **additional_paths: List[str]*, ***kwargs*) → `requests.Response`

Makes a `requests` PUT call, checks for errors, and returns the response.

Parameters

- **path** (*str*) – relative path, e.g. "users"
- ***additional_paths** (*list(str)*, *optional*) – any additional path components
- ****kwargs** (*dict*) – any additional kwargs to send to `requests`

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the `requests` `Response` object

Return type `requests.Response`

patch (*self*, *path: str*, **additional_paths: List[str]*, ***kwargs*) → `requests.Response`

Makes a `requests` PATCH call, checks for errors, and returns the response.

Parameters

- **path** (*str*) – relative path, e.g. "users"
- ***additional_paths** (*list(str)*, *optional*) – any additional path components
- ****kwargs** (*dict*) – any additional kwargs to send to `requests`

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the `requests` `Response` object

Return type `requests.Response`

post (*self*, *path*: *str*, **additional_paths*: *List[str]*, ***kwargs*) → `requests.Response`

Makes a `requests` POST call, checks for errors, and returns the response.

Parameters

- **path** (*str*) – relative path, e.g. "users"
- ***additional_paths** (*list(str)*, *optional*) – any additional path components
- ****kwargs** (*dict*) – any additional kwargs to send to `requests`

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the `requests` `Response` object

Return type `requests.Response`

```
class pine.client.client.EveClient(eve_base_uri: str, mongo_base_uri: str = None,
                                   mongo_dbname: str = DEFAULT_DBNAME, verify_ssl:
                                   bool = True)
```

Bases: `pine.client.client.BaseClient`

A client to access Eve and, optionally, its underlying MongoDB instance.

Constructor.

Parameters

- **eve_base_uri** (*str*) – the base URI for the eve server, e.g. "http://localhost:5001"
- **mongo_base_uri** (*str*, *optional*) – the base URI for the mongodb server, e.g. "mongodb://localhost:27018", defaults to None
- **mongo_dbname** (*str*, *optional*) – the DB name that PINE uses, defaults to "pmap_nlp"
- **verify_ssl** (*bool*, *optional*) – whether to verify SSL/HTTPs calls; do not turn this off unless you are fully aware of the security consequences

DEFAULT_DBNAME :*str* = `pmap_nlp`

The default DB name used by PINE.

Type *str*

mongo_base_uri :*str*

The base URI for the MongoDB server.

Type *str*

mongo :`pymongo.MongoClient`

The `pymongo.mongo_client.MongoClient` instance.

Type `pymongo.mongo_client.MongoClient`

mongo_db :`pymongo.database.Database`

The `pymongo.database.Database` instance.

Type `pymongo.database.Database`

is_valid (*self*) → *bool*

Returns whether this client and its connection(s) are valid.

Returns whether this client and its connection(s) are valid

Return type *bool*

ping (*self*) → *Any*

Pings the eve server and returns the result.

Raises *exceptions.PineClientHttpException* – if the HTTP request returns an error

Returns the JSON response from the server (probably "pong")

about (*self*) → *dict*

Returns the ‘about’ dict from the server.

Raises *exceptions.PineClientHttpException* – if the HTTP request returns an error

Returns the JSON response from the server

Return type *dict*

get_resource (*self*, *resource*: *str*, *resource_id*: *str*) → *dict*

Gets a resource from eve by its ID.

Raises *exceptions.PineClientHttpException* – if the HTTP request returns an error

Returns the JSON object response from the server

Return type *dict*

_add_or_replace_resource (*self*, *resource*: *str*, *obj*: *dict*, *valid_fn*: *Callable[[dict, typing.Callable[[str], None]], bool] = None*) → *str*

Adds or replaces the given resource.

Parameters

- **resource** (*str*) – the resource type, e.g. "users"
- **obj** (*dict*) – the resource object
- **valid_fn** (*function*, *optional*) – a function to validate the resource object, defaults to None

Raises

- *exceptions.PineClientValueException* – if a valid_fn is passed in and the object fails
- *exceptions.PineClientHttpException* – if the HTTP request returns an error

Returns the ID of the added/replaced resource

Return type *str*

_add_resources (*self*, *resource*: *str*, *objs*: *List[dict]*, *valid_fn*: *Callable[[dict, typing.Callable[[str], None]], bool] = None*, *replace_if_exists*: *bool* = *False*)

Tries to add all the resource objects at once, optionally falling back to individual replacement if that fails.

Parameters

- **resource** (*str*) – the resource type, e.g. "users"
- **objs** (*list (dict)*) – the resource objects

- **valid_fn** (*function*, *optional*) – a function to validate the resource object, defaults to `None`
- **replace_if_exists** (*bool*, *optional*) – whether to replace the resource with the given value if it already exists on the server, defaults to `False`

Raises

- **`exceptions.PineClientValueException`** – if a `valid_fn` is passed in and any of the objects fails
- **`exceptions.PineClientHttpException`** – if the HTTP request returns an error

Returns the IDs of the added resources

Return type `list(str)`

add_users (*self*, *users*: `List[dict]`, *replace_if_exists*=`False`) → `List[str]`

Adds the given users.

Parameters

- **users** (`list(dict)`) – the user objects
- **replace_if_exists** (*bool*, *optional*) – whether to replace the resource with the given value if it already exists on the server, defaults to `False`

Raises

- **`exceptions.PineClientValueException`** – if any of the user objects are not valid, see `models.is_valid_eve_user()`
- **`exceptions.PineClientHttpException`** – if the HTTP request returns an error

Returns the IDs of the added users

Return type `list(str)`

get_users (*self*)

Gets all users.

Raises **`exceptions.PineClientHttpException`** – if the HTTP request returns an error

Returns all the users

Return type `list(dict)`

add_pipelines (*self*, *pipelines*: `List[dict]`, *replace_if_exists*=`False`) → `List[str]`

Adds the given pipelines.

Parameters

- **pipelines** (`list(dict)`) – the pipeline objects
- **replace_if_exists** (*bool*, *optional*) – whether to replace the resource with the given value if it already exists on the server, defaults to `False`

Raises

- **`exceptions.PineClientValueException`** – if any of the pipeline objects are not valid, see `models.is_valid_eve_pipeline()`
- **`exceptions.PineClientHttpException`** – if the HTTP request returns an error

Returns the IDs of the added pipelines

Return type `list(str)`

class `pine.client.client.PineClient` (*backend_base_uri: str, verify_ssl: bool = True*)

Bases: `pine.client.client.BaseClient`

A client to access PINE (more specifically: the backend).

Constructor.

Parameters

- **backend_base_uri** (*str*) – the base URI for the backend server, e.g. "http://localhost:5000"
- **verify_ssl** (*bool, optional*) – whether to verify SSL/HTTPs calls; do not turn this off unless you are fully aware of the security consequences

is_valid (*self*) → `bool`

Returns whether this client and its connection(s) are valid.

Returns whether this client and its connection(s) are valid

Return type `bool`

ping (*self*) → `Any`

Pings the backend server and returns the result.

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the JSON response from the server (probably "pong")

about (*self*) → `dict`

Returns the 'about' dict from the server.

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the JSON response from the server

Return type `dict`

get_logged_in_user (*self*) → `dict`

Returns the currently logged in user, or None if not logged in.

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns currently logged in user, or None if not logged in

Return type `dict`

get_my_user_id (*self*) → `str`

Returns the ID of the logged in user, or None if not logged in.

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the ID of the logged in user, or None if not logged in

Return type `str`

is_logged_in (*self*) → `bool`

Returns whether the user is currently logged in or not.

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns whether the user is currently logged in or not

Return type `bool`

`_check_login(self)`

Checks whether user is logged in and raises an `exceptions.PineClientAuthException` if not.

Raises

- `exceptions.PineClientAuthException` – if not logged in
- `exceptions.PineClientHttpException` – if the HTTP request returns an error

`get_auth_module(self) → str`

Returns the PINE authentication module, e.g. "eve".

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the PINE authentication module, e.g. "eve"

Return type `str`

`login_eve(self, username: str, password: str) → bool`

Logs in using eve credentials, and returns whether it was successful.

Parameters

- **`username`** (`str`) – username
- **`password`** (`str`) – password

Raises

- `exceptions.PineClientAuthException` – if auth module is not eve or login was not successful
- `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns whether the login was successful

Return type `bool`

`logout(self)`

Logs out the current user.

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

`get_pipelines(self) → List[dict]`

Returns all pipelines accessible to logged in user.

Raises

- `exceptions.PineClientAuthException` – if not logged in
- `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns all pipelines accessible to logged in user

Return type `list(dict)`

`collection_builder(self, **kwargs: dict) → pine.client.models.CollectionBuilder`

Makes and returns a new `models.CollectionBuilder` with the logged in user.

Parameters **`**kwargs`** (`dict`) – any additional args to pass in to the constructor

Returns a new `models.CollectionBuilder` with the logged in user

Return type `models.CollectionBuilder`

create_collection (*self*, *builder*: `pine.client.models.CollectionBuilder`) → `str`

Creates a collection using the current value of the given builder and returns its ID.

Parameters **builder** (`models.CollectionBuilder`) – collection builder

Raises

- `exceptions.PineClientValueException` – if the given collection is not valid, see `models.is_valid_collection()`
- `exceptions.PineClientAuthException` – if not logged in
- `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the created collection's ID

Return type `str`

get_collection_permissions (*self*, *collection_id*: `str`) → `pine.client.models.CollectionUserPermissions`

Returns collection permissions for the logged in user.

Parameters **collection_id** (`str`) – the ID of the collection

Returns the collection permissions

Return type `models.CollectionUserPermissions`

get_collection_documents (*self*, *collection_id*: `str`, *truncate*: `bool`, *truncate_length*: `int` = 30) → `List[dict]`

Returns all the documents in the given collection.

Parameters

- **collection_id** (`str`) – the ID of the collection
- **truncate** (`bool`) – whether to truncate the document text (a good idea unless you need it)
- **truncate_length** (`int`, *optional*) – how many characters of the text you want if truncated, defaults to 30

Returns all the documents in the given collection

Return type `list(dict)`

add_document (*self*, *document*: `dict` = {}, *creator_id*: `str` = None, *collection_id*: `str` = None, *overlap*: `int` = None, *text*: `str` = None, *metadata*: `dict` = None) → `str`

Adds a new document to a collection and returns its ID.

Will use the logged in user ID for the `creator_id` if none is given. Although all the parameters are optional, you must provide values either in the document or through the kwargs in order to make a valid document.

Parameters

- **document** (`dict`, *optional*) – optional document dict, will be overridden with any kwargs, defaults to {}
- **creator_id** (`str`, *optional*) – optional creator_id for the document, defaults to None (not set)
- **collection_id** (`str`, *optional*) – optional collection_id for the document, defaults to None (not set)
- **overlap** (`int`, *optional*) – optional overlap for the document, defaults to None (not set)

- **text** (*str*, *optional*) – optional text for the document, defaults to None (not set)
- **metadata** (*dict*, *optional*) – optional metadata for the document, defaults to None (not set)

Raises

- **`exceptions.PineClientValueException`** – if the given document parameters are not valid, see `models.is_valid_eve_document()`
- **`exceptions.PineClientAuthException`** – if not logged in
- **`exceptions.PineClientHttpException`** – if the HTTP request returns an error

Returns the created document's ID

Return type `str`

add_documents (*self*, *documents*: `List[dict]`, *creator_id*: `str = None`, *collection_id*: `str = None`) → `List[str]`

Adds multiple documents at once and returns their IDs.

Will use the logged in user ID for the `creator_id` if none is given.

Parameters

- **documents** (`list(dict)`) – the documents to add
- **creator_id** (*str*, *optional*) – optional `creator_id` to set in the documents, defaults to None (not set)
- **collection_id** (*str*, *optional*) – optional `collection_id` to set in the documents, defaults to None (not set)

Raises

- **`exceptions.PineClientValueException`** – if any of the given documents are not valid, see `models.is_valid_eve_document()`
- **`exceptions.PineClientAuthException`** – if not logged in
- **`exceptions.PineClientHttpException`** – if the HTTP request returns an error

Returns the created documents' IDs

Return type `list(str)`

annotate_document (*self*, *document_id*: `str`, *doc_annotations*: `List[str]`, *ner_annotations*: `List[Union[dict, list, tuple]]`) → `str`

Annotates the given document with the given values.

Parameters

- **document_id** (*str*) – the document ID to annotate
- **doc_annotations** (`list(str)`) – document annotations/labels
- **ner_annotations** (`list`) – NER annotations, where each annotation is either a list or a dict

Raises

- **`exceptions.PineClientValueException`** – if any of the given annotations are not valid, see `models.is_valid_annotation()`
- **`exceptions.PineClientAuthException`** – if not logged in
- **`exceptions.PineClientHttpException`** – if the HTTP request returns an error

Returns the annotation ID

Return type `str`

annotate_collection_documents (*self*, *collection_id*: `str`, *document_annotations*: `dict`, *skip_document_updates*=`False`) → `List[str]`

Annotates documents in a collection.

Parameters

- **collection_id** (`str`) – the ID of the collection containing the documents to annotate
- **document_annotations** (`dict`) – a dict containing “ner” list and “doc” list
- **skip_document_updates** (`bool`) – whether to skip updating the document “has_annotated” map, defaults to `False`. This should only be `True` if you properly set the “has_annotated” map when you created the document.

Raises

- **`exceptions.PineClientValueException`** – if any of the given annotations are not valid, see `models.is_valid_doc_annotations()`
- **`exceptions.PineClientAuthException`** – if not logged in
- **`exceptions.PineClientHttpException`** – if the HTTP request returns an error

Returns the annotation IDs

Return type `list(str)`

list_collections (*self*, *include_archived*: `bool` = `False`) → `List[dict]`

Returns a list of user’s collections.

Parameters **include_archived** (`bool`) – whether to include archived collections, defaults to `False`

Raises

- **`exceptions.PineClientAuthException`** – if not logged in
- **`exceptions.PineClientHttpException`** – if the HTTP request returns an error

Returns user’s collections

Return type `list(dict)`

download_collection_data (*self*, *collection_id*: `str`, *include_collection_metadata*: `bool` = `True`, *include_document_metadata*: `bool` = `True`, *include_document_text*: `bool` = `True`, *include_annotations*: `bool` = `True`, *include_annotation_latest_version_only*: `bool` = `True`)

Downloads collection data.

Parameters

- **collection_id** (`str`) – the ID of the collection for which to download data
- **include_collection_metadata** (`bool`) – whether to include collection metadata, defaults to `True`
- **include_document_metadata** (`bool`) – whether to include document metadata, defaults to `True`
- **include_document_text** (`bool`) – whether to include document text, defaults to `True`
- **include_annotations** (`bool`) – whether to include annotations, defaults to `True`

- **include_annotation_latest_version_only** (*bool*) – whether to include only the latest version of annotations (True) or all versions (False), defaults to True

Raises

- *exceptions.PineClientValueException* – if given empty collection ID
- *exceptions.PineClientAuthException* – if not logged in
- *exceptions.PineClientHttpException* – if the HTTP request returns an error, such as if the collection doesn't exist

Returns collection data

Return type *dict*

```
class pine.client.client.LocalPineClient(backend_base_uri: str, eve_base_uri: str,
                                         mongo_base_uri: str = None, mongo_dbname:
                                         str = EveClient.DEFAULT_DBNAME, verify_ssl:
                                         bool = True)
```

Bases: *pine.client.client.PineClient*

A client for a local PINE instance, including an *EveClient*.

Constructor.

Parameters

- **backend_base_uri** (*str*) – the base URI for the backend server, e.g. "http://localhost:5000"
- **eve_base_uri** (*str*) – the base URI for the eve server, e.g. "http://localhost:5001"
- **mongo_base_uri** (*str*, *optional*) – the base URI for the mongodb server, e.g. "mongodb://localhost:27018", defaults to None
- **mongo_dbname** (*str*, *optional*) – the DB name that PINE uses, defaults to "pmap_nlp"
- **verify_ssl** (*bool*, *optional*) – whether to verify SSL/HTTPs calls; do not turn this off unless you are fully aware of the security consequences

eve :*EveClient*

The local *EveClient* instance.

Type *EveClient*

is_valid (*self*) → *bool*

Returns whether this client and its connection(s) are valid.

Returns whether this client and its connection(s) are valid

Return type *bool*

pine.client.exceptions

PINE client exceptions module.

Module Contents

exception pine.client.exceptions.**PineClientException** (*message: str, cause: Exception = None*)

Bases: `Exception`

Base class for PINE client exceptions.

Constructor.

Parameters

- **message** (*str*) – the message
- **cause** (*Exception, optional*) – optional cause, defaults to None

message

The message.

Type `str`

exception pine.client.exceptions.**PineClientHttpException** (*method: str, path: str, resp: requests.Response*)

Bases: `pine.client.exceptions.PineClientException`

A PINE client exception caused by an underlying HTTP exception.

Constructor.

Parameters

- **method** (*str*) – the REST method ("get", "post", etc.)
- **path** (*str*) – the human-readable path that caused the exception
- **resp** (*requests.Response*) – the `Response` with the error info

resp : requests.Response

The `Response` with the error info

Type `requests.Response`

property `status_code` (*self*)

exception pine.client.exceptions.**PineClientValueException** (*obj: dict, obj_type: str*)

Bases: `pine.client.exceptions.PineClientException`

A PINE client exception caused by passing invalid data.

Constructor.

Parameters

- **obj** (*dict*) – the error data
- **obj_type** (*str*) – human-readable type of object

exception pine.client.exceptions.**PineClientAuthException** (*message: str, cause: Exception = None*)

Bases: `pine.client.exceptions.PineClientException`

Base class for PINE client exceptions.

Constructor.

Parameters

- **message** (*str*) – the message
- **cause** (*Exception*, *optional*) – optional cause, defaults to None

`pine.client.log`

Module Contents

Functions

<code>setup_logging()</code>	Sets up logging, if configured to do so.
------------------------------	--

`pine.client.log.CONFIG_FILE_ENV :str = PINE_LOGGING_CONFIG_FILE`
The environment variable that optionally contains the file to use for logging configuration.

Type `str`

`pine.client.log.setup_logging()`
Sets up logging, if configured to do so.

The environment variable named by `CONFIG_FILE_ENV` is checked and, if present, is passed to `logging.config.dictConfig()`.

`pine.client.models`

Module Contents

Classes

<code>CollectionBuilder</code>	A class that can build the form and files fields that are necessary to create a collection.
<code>CollectionUserPermissions</code>	Collection permissions for a user as a dictionary of boolean flags.

Functions

<code>_check_field_required_bool(obj: dict, field: str) → bool</code>	Checks that the given field is in the object and is a bool.
<code>_check_field_int(obj: dict, field: str) → bool</code>	Checks that if the given field is in the object, that it is an int.
<code>_check_field_required_int(obj: dict, field: str) → bool</code>	Checks that the given field is in the object and is an int.
<code>_check_field_float(obj: dict, field: str) → bool</code>	Checks that if the given field is in the object, that it is a float.

continues on next page

Table 42 – continued from previous page

<code>_check_field_required_float(obj: dict, field: str) → bool</code>	dict,	Checks that the given field is in the object and is a float.
<code>_check_field_string(obj: dict, field: str) → bool</code>	dict, field: str	Checks that if the given field is in the object, that it is a string.
<code>_check_field_required_string(obj: dict, field: str) → bool</code>	dict,	Checks that the given field is in the object and is a string.
<code>_check_field_string_list(obj: dict, field: str, min_length: int = 0) → bool</code>	dict, field: str,	Checks that if the given field is in the object, that it is a string list.
<code>_check_field_required_string_list(obj: dict, field: str, min_length: int = 0) → bool</code>	dict, field: str,	Checks that the given field is in the object and is a string list.
<code>_check_field_dict(obj: dict, field: str) → bool</code>	dict, field: str	Checks that if the given field is in the object, that it is a dict.
<code>_check_field_required_dict(obj: dict, field: str) → bool</code>	dict, field: str	Checks that the given field is in the object and is a dict.
<code>_check_field_bool(obj: dict, field: str) → bool</code>	dict, field: str	Checks that if the given field is in the object, that it is a bool.
<code>is_valid_eve_user(user: dict, error_callback: Callable[[str], None] = None) → bool</code>	dict, error_callback: Callable[[str], None]	Checks whether the given user object is valid.
<code>is_valid_eve_pipeline(pipeline: dict, error_callback: Callable[[str], None] = None) → bool</code>	dict, error_callback: Callable[[str], None]	Checks whether the given pipeline object is valid.
<code>is_valid_eve_collection(collection: dict, error_callback: Callable[[str], None] = None) → bool</code>	dict, error_callback: Callable[[str], None]	Checks whether the given collection object is valid.
<code>is_valid_collection(form: dict, files: dict, error_callback: Callable[[str], None] = None) → bool</code>	dict, files: dict, error_callback: Callable[[str], None]	Checks whether the given form and files parameters are valid for creating a collection.
<code>is_valid_eve_document(document: dict, error_callback: Callable[[str], None] = None) → bool</code>	dict, error_callback: Callable[[str], None]	Checks whether the given document object is valid.
<code>is_valid_doc_annotation(ann: Any, error_callback: Callable[[str], None] = None) → bool</code>	Any, error_callback: Callable[[str], None]	Checks whether the given annotation is a valid document label/annotation.
<code>is_valid_ner_annotation(ann: Any, error_callback: Callable[[str], None] = None) → bool</code>	Any, error_callback: Callable[[str], None]	Checks whether the given annotation is a valid document NER annotation.
<code>is_valid_annotation(body: dict, error_callback: Callable[[str], None] = None) → bool</code>	dict, error_callback: Callable[[str], None]	Checks whether the given body is valid to create an annotation.
<code>is_valid_doc_annotations(doc_annotations: dict, error_callback: Callable[[str], None] = None) → bool</code>	dict, error_callback: Callable[[str], None]	Checks whether the given document annotations are valid.
<code>remove_eve_fields(obj: dict, remove_timestamps: bool = True, remove_versions: bool = True)</code>	dict, remove_timestamps: bool = True, remove_versions: bool = True	Removes fields inserted by eve from the given object.
<code>remove_nonupdatable_fields(obj: dict)</code>	dict	Removes all non-updatable fields from the given object.

`pine.client.models.ID_FIELD :str = _id`

The field used to store database ID.

Type str

`pine.client.models.ITEMS_FIELD :str = _items`

The field used to access the items in a multi-item database response.

Type str

`pine.client.models._check_field_required_bool (obj: dict, field: str) → bool`
Checks that the given field is in the object and is a bool.

Parameters

- **obj** (*dict*) – the object to check
- **field** (*str*) – the field to check

Returns whether the given field is in the object and is a bool

Return type `bool`

`pine.client.models._check_field_int (obj: dict, field: str) → bool`
Checks that if the given field is in the object, that it is an int.

Parameters

- **obj** (*dict*) – the object to check
- **field** (*str*) – the field to check

Returns if the given field is in the object, that it is an int

Return type `bool`

`pine.client.models._check_field_required_int (obj: dict, field: str) → bool`
Checks that the given field is in the object and is an int.

Parameters

- **obj** (*dict*) – the object to check
- **field** (*str*) – the field to check

Returns whether the given field is in the object and is an int

Return type `bool`

`pine.client.models._check_field_float (obj: dict, field: str) → bool`
Checks that if the given field is in the object, that it is a float.

Parameters

- **obj** (*dict*) – the object to check
- **field** (*str*) – the field to check

Returns if the given field is in the object, that it is a float

Return type `bool`

`pine.client.models._check_field_required_float (obj: dict, field: str) → bool`
Checks that the given field is in the object and is a float.

Parameters

- **obj** (*dict*) – the object to check
- **field** (*str*) – the field to check

Returns whether the given field is in the object and is a float

Return type `bool`

`pine.client.models._check_field_string (obj: dict, field: str) → bool`
Checks that if the given field is in the object, that it is a string.

Parameters

- **obj** (*dict*) – the object to check
- **field** (*str*) – the field to check

Returns if the given field is in the object, that it is a string

Return type `bool`

`pine.client.models._check_field_required_string(obj: dict, field: str) → bool`

Checks that the given field is in the object and is a string.

Parameters

- **obj** (*dict*) – the object to check
- **field** (*str*) – the field to check

Returns whether the given field is in the object and is a string

Return type `bool`

`pine.client.models._check_field_string_list(obj: dict, field: str, min_length: int = 0) →`

Checks that if the given field is in the object, that it is a string ^{`bool`} list.

Parameters

- **obj** (*dict*) – the object to check
- **field** (*str*) – the field to check
- **min_length** (*int*, *optional*) – the minimum length of the list (if > 0), defaults to 0

Returns if the given field is in the object, that it is a string list

Return type `bool`

`pine.client.models._check_field_required_string_list(obj: dict, field: str, min_length: int = 0) → bool`

Checks that the given field is in the object and is a string list.

Parameters

- **obj** (*dict*) – the object to check
- **field** (*str*) – the field to check
- **min_length** (*int*, *optional*) – the minimum length of the list (if > 0), defaults to 0

Returns if the given field is in the object, that it is a string list

Return type `bool`

`pine.client.models._check_field_dict(obj: dict, field: str) → bool`

Checks that if the given field is in the object, that it is a dict.

Parameters

- **obj** (*dict*) – the object to check
- **field** (*str*) – the field to check

Returns if the given field is in the object, that it is a dict

Return type `bool`

`pine.client.models._check_field_required_dict(obj: dict, field: str) → bool`

Checks that the given field is in the object and is a dict.

Parameters

- **obj** (*dict*) – the object to check
- **field** (*str*) – the field to check

Returns whether the given field is in the object and is a dict

Return type `bool`

```
pine.client.models._check_field_bool(obj: dict, field: str) → bool
```

Checks that if the given field is in the object, that it is a bool.

Parameters

- **obj** (*dict*) – the object to check
- **field** (*str*) – the field to check

Returns if the given field is in the object, that it is a bool

Return type `bool`

```
pine.client.models.is_valid_eve_user(user: dict, error_callback: Callable[[str], None] =  
                                     None) → bool
```

Checks whether the given user object is valid.

A valid user object has an `_id`, `firstname`, and `lastname` that are non-empty string fields. If `email`, `description`, or `passwdhash` are present, they are string fields. If `role` is present, it is a list of strings that are either `administrator` or `user`.

Parameters

- **user** (*dict*) – user object
- **error_callback** (*function*, *optional*) – optional callback that is called with any error messages, defaults to `None`

Returns whether the given user object is valid

Return type `bool`

```
pine.client.models.is_valid_eve_pipeline(pipeline: dict, error_callback: Callable[[str],  
                                         None] = None) → bool
```

Checks whether the given pipeline object is valid.

A valid pipeline has an `_id`, `title`, and `name` that are non-empty string fields. If `description` is provided, it is a string field. If `parameters` are provided, it is a dict field.

Parameters

- **pipeline** (*dict*) – pipeline object
- **error_callback** (*function*, *optional*) – optional callback that is called with any error messages, defaults to `None`

Returns whether the given pipeline object is valid

Return type `bool`

```
pine.client.models.is_valid_eve_collection(collection: dict, error_callback: Callable[[str], None] = None) → bool
```

Checks whether the given collection object is valid.

A valid collection has a `creator_id` that is a non-empty string field. It has a `labels` that is a non-empty list of strings. If `annotators` or `viewers` are provided, they are lists of strings. If `metadata` or `configuration` are provided, they are dicts. If `archived` is provided, it is a bool.

Parameters

- **collection** (*dict*) – collection object
- **error_callback** (*function*, *optional*) – optional callback that is called with any error messages, defaults to None

Returns whether the given collection object is valid

Return type `bool`

```
pine.client.models.is_valid_collection (form: dict, files: dict, error_callback:
Callable[[str], None] = None) → bool
```

Checks whether the given form and files parameters are valid for creating a collection.

A valid form has a `collection` that is a dict field and is valid via `is_valid_eve_collection()`. Additionally, the collection has string `title` and `description` fields in its metadata. It also has at least one element for `labels`, `viewers`, and `annotators`, and the `creator_id` must be in both `viewers` and `annotators`.

The form also has `overlap` as a float field between 0 and 1 (inclusive), `train_every` as an int field that is at least 5, and `pipelineId` as a string field.

If files are provided, file `file` and any files starting with `imageFile` are checked. If a file `file` is provided, the form must also have a boolean `csvHasHeader` field and an int `csvTextCol` field.

Parameters

- **form** (*dict*) – form data to send to backend
- **files** (*dict*) – file data to send to backend
- **error_callback** (*function*, *optional*) – optional callback that is called with any error messages, defaults to None

Returns whether the given form and files parameters are valid for creating a collection

Return type `bool`

```
pine.client.models.is_valid_eve_document (document: dict, error_callback: Callable[[str],
None] = None) → bool
```

Checks whether the given document object is valid.

A valid document has a `creator_id` and `collection_id` that are non-empty string fields. Optionally, it may have an int `overlap` field, string `text` field, and dict `metadata` and `has_annotated` fields.

Parameters

- **document** (*dict*) – document object
- **error_callback** (*function*, *optional*) – optional callback that is called with any error messages, defaults to None

Returns whether the given document object is valid

Return type `bool`

```
pine.client.models.is_valid_doc_annotation (ann: Any, error_callback: Callable[[str],
None] = None) → bool
```

Checks whether the given annotation is a valid document label/annotation.

This means that it is a non-empty string.

Parameters

- **ann** – annotation

- **error_callback** (*function*, *optional*) – optional callback that is called with any error messages, defaults to None

Returns whether the given annotation is a valid document label/annotation

Return type `bool`

```
pine.client.models.is_valid_ner_annotation(ann: Any, error_callback: Callable[[str],  
None] = None) → bool
```

Checks whether the given annotation is a valid document NER annotation.

Valid NER annotations take one of two forms: a `dict` or a `list/tuple` of size 3.

A valid NER `dict` has the following fields:

- `start`: an `int` that is ≥ 0
- `end`: an `int` that is ≥ 0
- `label`: a non-empty `str`

A valid NER `list/tuple` has the following elements:

- `element 0`: an `int` that is ≥ 0
- `element 1`: an `int` that is ≥ 0
- `element 2`: a non-empty `str`

Parameters

- **ann** – annotation
- **error_callback** (*function*, *optional*) – optional callback that is called with any error messages, defaults to None

Returns whether the given annotation is a valid document label/annotation

Return type `bool`

```
pine.client.models.is_valid_annotation(body: dict, error_callback: Callable[[str], None] =  
None) → bool
```

Checks whether the given body is valid to create an annotation.

A valid body is a `dict` with two fields:

- `doc`: a list of valid doc annotations (see `is_valid_doc_annotation()`)
- `ner`: a list of valid NER annotations (see `is_valid_ner_annotation()`)

Parameters

- **body** (*dict*) – annotation body
- **error_callback** (*function*, *optional*) – optional callback that is called with any error messages, defaults to None

Returns whether the given body is valid to create an annotation

Return type `bool`

```
pine.client.models.is_valid_doc_annotations(doc_annotations: dict, error_callback:  
Callable[[str], None] = None) → bool
```

Checks whether the given document annotations are valid.

A valid document annotations object is a `dict`, where the keys are `str` document IDs, and the values are valid annotation bodies (see `is_valid_annotation()`).

Parameters

- **doc_annotations** – document annotations
- **error_callback** (*function*, *optional*) – optional callback that is called with any error messages, defaults to None

Returns whether the given body is valid to create an annotation

Return type `bool`

```
class pine.client.models.CollectionBuilder (collection: dict = None, creator_id: str =
None, viewers: List[str] = None, annotators:
List[str] = None, labels: List[str] = None,
title: str = None, description: str = None,
allow_overlapping_ner_annotations: bool =
None, pipelineId: str = None, overlap: float
= None, train_every: int = None, classifierPa-
rameters: dict = None, document_csv_file: str
= None, document_csv_file_has_header: bool
= None, document_csv_file_text_column: int
= None, image_files: List[str] = None)
```

Bases: `object`

A class that can build the form and files fields that are necessary to create a collection.

Constructor.

Parameters

- **collection** (*dict*, *optional*) – starting parameters for the collection, defaults to None (not set)
- **creator_id** (*str*, *optional*) – user ID for the creator, see `creator_id()`, defaults to None (not set)
- **viewers** (*list(str)*, *optional*) – viewer IDs for the collection, see `viewer()`, defaults to None (not set)
- **annotators** (*list(str)*, *optional*) – annotator IDs for the collection, see `annotator()`, defaults to None (not set)
- **labels** (*list(str)*, *optional*) – labels for the collection, see `label()`, defaults to None (not set)
- **title** (*str*, *optional*) – metadata title, see `title()`, defaults to None (not set)
- **description** (*str*, *optional*) – metadata description, see `description()`, defaults to None (not set)
- **allow_overlapping_ner_annotations** (*bool*) – optional configuration for allowing overlapping NER annotations, see `allow_overlapping_ner_annotations()`, defaults to None (not set)
- **pipelineId** (*str*, *optional*) – the ID of the pipeline from which to create the classifier, see `classifier()`, defaults to None (not set)
- **overlap** (*float*, *optional*) – the classifier overlap, see `classifier()`, defaults to None (not set)
- **train_every** (*int*, *optional*) – train the model after this many documents are annotated, see `classifier()`, defaults to None (not set)

- **classifierParameters** (*dict*, *optional*) – any parameters to pass to the classifier, see *classifier()*, defaults to None (not set)
- **document_csv_file** (*str*, *optional*) – the filename of the local document CSV file, see *document_csv_File()*, defaults to None (not set)
- **document_csv_file_has_header** (*bool*, *optional*) – whether the document CSV file has a header, see *document_csv_File()*, defaults to None (not set)
- **document_csv_file_text_column** (*int*, *optional*) – if the document CSV file has headers, the document text can be found in this column index (the others are used for document metadata), see *document_csv_File()*, defaults to None (not set)
- **image_files** (*list(str)*) – any image files to add to the collection, see *image_file()*, defaults to None (not set)

form

The form data.

Type *dict*

files

The files data.

Type *dict*

property collection (*self*) → *dict*

Returns the collection information from the form.

Returns collection information from the form

Return type *dict*

property form_json (*self*) → *dict*

Returns the form where the values have been JSON-encoded.

Returns the form where the values have been JSON-encoded

Return type *dict*

creator_id (*self*, *user_id: str*) → *pine.client.models.CollectionBuilder*

Sets the *creator_id* to the given, and adds to viewers and annotators.

Parameters *user_id (str)* – the user ID to use for the *creator_id*

Returns *self*

Return type *models.CollectionBuilder*

viewer (*self*, *user_id: str*) → *pine.client.models.CollectionBuilder*

Adds the given user to the list of viewers.

Parameters *user_id (str)* – the user ID to add as a viewer

Returns *self*

Return type *models.CollectionBuilder*

annotator (*self*, *user_id: str*) → *pine.client.models.CollectionBuilder*

Adds the given user to the list of annotators.

Parameters *user_id (str)* – the user ID to add as an annotator

Returns *self*

Return type *models.CollectionBuilder*

label (*self*, *label*: *str*) → *pine.client.models.CollectionBuilder*

Adds the given label to the collection.

Parameters **label** (*str*) – label to add

Returns *self*

Return type *models.CollectionBuilder*

metadata (*self*, *key*: *str*, *value*: *Any*) → *pine.client.models.CollectionBuilder*

Adds the given metadata key/value to the collection.

Parameters

- **key** (*str*) – metadata key
- **value** – metadata value

Returns *self*

Return type *models.CollectionBuilder*

title (*self*, *title*: *str*) → *pine.client.models.CollectionBuilder*

Sets the metadata title to the given.

Parameters **title** (*str*) – collection title

Returns *self*

Return type *models.CollectionBuilder*

description (*self*, *description*: *str*) → *pine.client.models.CollectionBuilder*

Sets the metadata description to the given.

Parameters **description** (*str*) – collection description

Returns *self*

Return type *models.CollectionBuilder*

configuration (*self*, *key*: *str*, *value*: *Any*) → *pine.client.models.CollectionBuilder*

Adds the given configuration key/value to the collection.

Parameters

- **key** (*str*) – configuration key
- **value** – configuration value

Returns *self*

Return type *models.CollectionBuilder*

allow_overlapping_ner_annotations (*self*, *allow_overlapping_ner_annotations*: *bool*)

Sets the configuration value for `allow_overlapping_ner_annotations` to the given.

Parameters **allow_overlapping_ner_annotations** (*bool*) – whether to allow overlapping NER annotations

Returns *self*

Return type *models.CollectionBuilder*

classifier (*self*, *pipelineId*: *str*, *overlap*: *float* = 0, *train_every*: *int* = 100, *classifierParameters*: *dict* = {}) → *pine.client.models.CollectionBuilder*

Sets classifier information for the created collection.

Parameters

- **pipelineId** (*str*) – the ID of the pipeline from which to create the classifier
- **overlap** (*float*, *optional*) – the classifier overlap, defaults to 0
- **train_every** (*int*, *optional*) – train the model after this many documents are annotated, defaults to 100
- **classifierParameters** (*dict*, *optional*) – any parameters to pass to the classifier, defaults to { }

Returns self

Return type *models.CollectionBuilder*

document_csv_file (*self*, *csv_filename*: *str*, *has_header*: *bool*, *text_column*: *int*) → *pine.client.models.CollectionBuilder*

Sets the CSV file used to create documents to the given.

May raise an Exception if there is a problem opening the indicated file.

Parameters

- **csv_filename** (*str*) – the filename of the local CSV file
- **has_header** (*bool*) – whether the CSV file has a header
- **text_column** (*int*) – if the CSV file has headers, the document text can be found in this column index (the others are used for document metadata)

Returns self

Return type *models.CollectionBuilder*

image_file (*self*, *image_filename*: *str*) → *pine.client.models.CollectionBuilder*

Adds the given image file to the collection.

May raise an Exception if there is a problem opening the indicated file.

Parameters **image_filename** (*str*) – the filename of the local image file

Returns self

Return type *models.CollectionBuilder*

is_valid (*self*, *error_callback*: *Callable[[str], None]* = *None*)

Checks whether the currently set values are valid or not.

See *is_valid_collection()*.

Parameters **error_callback** (*function*, *optional*) – optional callback that is called with any error messages, defaults to None

Returns whether the currently set values are valid or not

Return type *bool*

```
class pine.client.models.CollectionUserPermissions (view=False, annotate=False,
                                                    add_documents=False,
                                                    add_images=False, mod-
                                                    ify_users=False, mod-
                                                    ify_labels=False, mod-
                                                    ify_document_metadata=False,
                                                    download_data=False,
                                                    archive=False)
```

Bases: *object*

Collection permissions for a user as a dictionary of boolean flags.

view :bool

Whether the user can view the collection and documents. :type: bool

annotate :bool

Whether the user can annotate collection documents. :type: bool

add_documents :bool

Whether the user can add documents to the collection. :type: bool

add_images :bool

Whether the user can add images to the collection. :type: bool

modify_users :bool

Whether the user can modify the list of viewers/annotators for the collection. :type: bool

modify_labels :bool

Whether the user can modify the list of labels for the collection. :type: bool

modify_document_metadata :bool

Whether the user can modify document metadata (such as changing the image). :type: bool

download_data :bool

Whether the user can download the collection data :type: bool

archive :bool

Whether the user can archive or unarchive the collection. :type: bool

to_dict (*self*) → dict

Returns a dict version of this object for conversion to JSON.

Returns a dict version of this object for conversion to JSON

Return type dict

pine.client.models.remove_eve_fields(obj: dict, remove_timestamps: bool = True, remove_versions: bool = True)

Removes fields inserted by eve from the given object.

Parameters

- **obj** (dict) – the object
- **remove_timestamps** (bool) – whether to remove the timestamp fields, defaults to True
- **remove_versions** (bool) – whether to remove the version fields, defaults to True

pine.client.models.remove_nonupdatable_fields(obj: dict)

Removes all non-updatable fields from the given object.

These fields would cause a PUT/PATCH to be rejected because they are not user-modifiable.

Parameters **obj** (dict) – the object

`pine.client.password`

Module Contents

Functions

<code>hash_password(password: str) → str</code>	Hashes the given password for use in user object.
<code>check_password(password: str, hashed_password: str) → str</code>	Checks the given password against the given hash.

`pine.client.password.hash_password(password: str) → str`

Hashes the given password for use in user object.

Parameters `password(str)` – password

Returns hashed password

Return type `str`

`pine.client.password.check_password(password: str, hashed_password: str) → str`

Checks the given password against the given hash.

Parameters

- `password(str)` – password to check
- `hashed_password(str)` – hashed password to check against

Returns whether the password matches the hash

Return type `bool`

Package Contents

Classes

<code>PineClient</code>	A client to access PINE (more specifically: the back-end).
<code>LocalPineClient</code>	A client for a local PINE instance, including an EveClient.
<code>CollectionBuilder</code>	A class that can build the form and files fields that are necessary to create a collection.

Functions

<code>setup_logging()</code>	Sets up logging, if configured to do so.
------------------------------	--

class `pine.client.PineClient(backend_base_uri: str, verify_ssl: bool = True)`

Bases: `pine.client.client.BaseClient`

A client to access PINE (more specifically: the backend).

Constructor.

Parameters

- **backend_base_uri** (*str*) – the base URI for the backend server, e.g. "http://localhost:5000"
- **verify_ssl** (*bool*, *optional*) – whether to verify SSL/HTTPs calls; do not turn this off unless you are fully aware of the security consequences

is_valid (*self*) → *bool*

Returns whether this client and its connection(s) are valid.

Returns whether this client and its connection(s) are valid

Return type *bool*

ping (*self*) → *Any*

Pings the backend server and returns the result.

Raises *exceptions.PineClientHttpException* – if the HTTP request returns an error

Returns the JSON response from the server (probably "pong")

about (*self*) → *dict*

Returns the 'about' dict from the server.

Raises *exceptions.PineClientHttpException* – if the HTTP request returns an error

Returns the JSON response from the server

Return type *dict*

get_logged_in_user (*self*) → *dict*

Returns the currently logged in user, or None if not logged in.

Raises *exceptions.PineClientHttpException* – if the HTTP request returns an error

Returns currently logged in user, or None if not logged in

Return type *dict*

get_my_user_id (*self*) → *str*

Returns the ID of the logged in user, or None if not logged in.

Raises *exceptions.PineClientHttpException* – if the HTTP request returns an error

Returns the ID of the logged in user, or None if not logged in

Return type *str*

is_logged_in (*self*) → *bool*

Returns whether the user is currently logged in or not.

Raises *exceptions.PineClientHttpException* – if the HTTP request returns an error

Returns whether the user is currently logged in or not

Return type *bool*

_check_login (*self*)

Checks whether user is logged in and raises an *exceptions.PineClientAuthException* if not.

Raises

- `exceptions.PineClientAuthException` – if not logged in
- `exceptions.PineClientHttpException` – if the HTTP request returns an error

get_auth_module (*self*) → *str*

Returns the PINE authentication module, e.g. "eve".

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the PINE authentication module, e.g. "eve"

Return type *str*

login_eve (*self*, *username: str*, *password: str*) → *bool*

Logs in using eve credentials, and returns whether it was successful.

Parameters

- **username** (*str*) – username
- **password** (*str*) – password

Raises

- `exceptions.PineClientAuthException` – if auth module is not eve or login was not successful
- `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns whether the login was successful

Return type *bool*

logout (*self*)

Logs out the current user.

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

get_pipelines (*self*) → *List[dict]*

Returns all pipelines accessible to logged in user.

Raises

- `exceptions.PineClientAuthException` – if not logged in
- `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns all pipelines accessible to logged in user

Return type *list(dict)*

collection_builder (*self*, ***kwargs: dict*) → *pine.client.models.CollectionBuilder*

Makes and returns a new `models.CollectionBuilder` with the logged in user.

Parameters ****kwargs** (*dict*) – any additional args to pass in to the constructor

Returns a new `models.CollectionBuilder` with the logged in user

Return type *models.CollectionBuilder*

create_collection (*self*, *builder: pine.client.models.CollectionBuilder*) → *str*

Creates a collection using the current value of the given builder and returns its ID.

Parameters **builder** (*models.CollectionBuilder*) – collection builder

Raises

- **`exceptions.PineClientValueException`** – if the given collection is not valid, see `models.is_valid_collection()`
- **`exceptions.PineClientAuthException`** – if not logged in
- **`exceptions.PineClientHttpException`** – if the HTTP request returns an error

Returns the created collection's ID

Return type `str`

`get_collection_permissions` (*self*, *collection_id*: `str`) → `pine.client.models.CollectionUserPermissions`

Returns collection permissions for the logged in user.

Parameters **`collection_id`** (`str`) – the ID of the collection

Returns the collection permissions

Return type `models.CollectionUserPermissions`

`get_collection_documents` (*self*, *collection_id*: `str`, *truncate*: `bool`, *truncate_length*: `int` = 30) → `List[dict]`

Returns all the documents in the given collection.

Parameters

- **`collection_id`** (`str`) – the ID of the collection
- **`truncate`** (`bool`) – whether to truncate the document text (a good idea unless you need it)
- **`truncate_length`** (`int`, *optional*) – how many characters of the text you want if truncated, defaults to 30

Returns all the documents in the given collection

Return type `list(dict)`

`add_document` (*self*, *document*: `dict` = {}, *creator_id*: `str` = None, *collection_id*: `str` = None, *overlap*: `int` = None, *text*: `str` = None, *metadata*: `dict` = None) → `str`

Adds a new document to a collection and returns its ID.

Will use the logged in user ID for the `creator_id` if none is given. Although all the parameters are optional, you must provide values either in the document or through the kwargs in order to make a valid document.

Parameters

- **`document`** (`dict`, *optional*) – optional document dict, will be overridden with any kwargs, defaults to {}
- **`creator_id`** (`str`, *optional*) – optional `creator_id` for the document, defaults to None (not set)
- **`collection_id`** (`str`, *optional*) – optional `collection_id` for the document, defaults to None (not set)
- **`overlap`** (`int`, *optional*) – optional overlap for the document, defaults to None (not set)
- **`text`** (`str`, *optional*) – optional text for the document, defaults to None (not set)
- **`metadata`** (`dict`, *optional*) – optional metadata for the document, defaults to None (not set)

Raises

- **`exceptions.PineClientValueException`** – if the given document parameters are not valid, see `models.is_valid_eve_document()`
- **`exceptions.PineClientAuthException`** – if not logged in
- **`exceptions.PineClientHttpException`** – if the HTTP request returns an error

Returns the created document's ID

Return type `str`

add_documents (*self*, *documents*: `List[dict]`, *creator_id*: `str = None`, *collection_id*: `str = None`) → `List[str]`

Adds multiple documents at once and returns their IDs.

Will use the logged in user ID for the `creator_id` if none is given.

Parameters

- **`documents`** (`list(dict)`) – the documents to add
- **`creator_id`** (`str`, *optional*) – optional `creator_id` to set in the documents, defaults to `None` (not set)
- **`collection_id`** (`str`, *optional*) – optional `collection_id` to set in the documents, defaults to `None` (not set)

Raises

- **`exceptions.PineClientValueException`** – if any of the given documents are not valid, see `models.is_valid_eve_document()`
- **`exceptions.PineClientAuthException`** – if not logged in
- **`exceptions.PineClientHttpException`** – if the HTTP request returns an error

Returns the created documents' IDs

Return type `list(str)`

annotate_document (*self*, *document_id*: `str`, *doc_annotations*: `List[str]`, *ner_annotations*: `List[Union[dict, list, tuple]]`) → `str`

Annotates the given document with the given values.

Parameters

- **`document_id`** (`str`) – the document ID to annotate
- **`doc_annotations`** (`list(str)`) – document annotations/labels
- **`ner_annotations`** (`list`) – NER annotations, where each annotation is either a list or a dict

Raises

- **`exceptions.PineClientValueException`** – if any of the given annotations are not valid, see `models.is_valid_annotation()`
- **`exceptions.PineClientAuthException`** – if not logged in
- **`exceptions.PineClientHttpException`** – if the HTTP request returns an error

Returns the annotation ID

Return type `str`

annotate_collection_documents (*self*, *collection_id*: *str*, *document_annotations*: *dict*, *skip_document_updates*=*False*) → List[*str*]

Annotates documents in a collection.

Parameters

- **collection_id** (*str*) – the ID of the collection containing the documents to annotate
- **document_annotations** (*dict*) – a dict containing “ner” list and “doc” list
- **skip_document_updates** (*bool*) – whether to skip updating the document “has_annotated” map, defaults to *False*. This should only be *True* if you properly set the “has_annotated” map when you created the document.

Raises

- **exceptions.PineClientValueException** – if any of the given annotations are not valid, see *models.is_valid_doc_annotations()*
- **exceptions.PineClientAuthException** – if not logged in
- **exceptions.PineClientHttpException** – if the HTTP request returns an error

Returns the annotation IDs

Return type list(*str*)

list_collections (*self*, *include_archived*: *bool* = *False*) → List[*dict*]

Returns a list of user’s collections.

Parameters **include_archived** (*bool*) – whether to include archived collections, defaults to *False*

Raises

- **exceptions.PineClientAuthException** – if not logged in
- **exceptions.PineClientHttpException** – if the HTTP request returns an error

Returns user’s collections

Return type list(*dict*)

download_collection_data (*self*, *collection_id*: *str*, *include_collection_metadata*: *bool* = *True*, *include_document_metadata*: *bool* = *True*, *include_document_text*: *bool* = *True*, *include_annotations*: *bool* = *True*, *include_annotation_latest_version_only*: *bool* = *True*)

Downloads collection data.

Parameters

- **collection_id** (*str*) – the ID of the collection for which to download data
- **include_collection_metadata** (*bool*) – whether to include collection metadata, defaults to *True*
- **include_document_metadata** (*bool*) – whether to include document metadata, defaults to *True*
- **include_document_text** (*bool*) – whether to include document text, defaults to *True*
- **include_annotations** (*bool*) – whether to include annotations, defaults to *True*
- **include_annotation_latest_version_only** (*bool*) – whether to include only the latest version of annotations (*True*) or all versions (*False*), defaults to *True*

Raises

- `exceptions.PineClientValueException` – if given empty collection ID
- `exceptions.PineClientAuthException` – if not logged in
- `exceptions.PineClientHttpException` – if the HTTP request returns an error, such as if the collection doesn't exist

Returns collection data

Return type `dict`

```
class pine.client.LocalPineClient(backend_base_uri: str, eve_base_uri: str,
                                  mongo_base_uri: str = None, mongo_dbname: str =
                                  EveClient.DEFAULT_DBNAME, verify_ssl: bool = True)
```

Bases: `pine.client.client.PineClient`

A client for a local PINE instance, including an EveClient.

Constructor.

Parameters

- **backend_base_uri** (`str`) – the base URI for the backend server, e.g. "http://localhost:5000"
- **eve_base_uri** (`str`) – the base URI for the eve server, e.g. "http://localhost:5001"
- **mongo_base_uri** (`str`, *optional*) – the base URI for the mongodb server, e.g. "mongodb://localhost:27018", defaults to None
- **mongo_dbname** (`str`, *optional*) – the DB name that PINE uses, defaults to "pmap_nlp"
- **verify_ssl** (`bool`, *optional*) – whether to verify SSL/HTTPs calls; do not turn this off unless you are fully aware of the security consequences

eve :`EveClient`

The local EveClient instance.

Type `EveClient`

is_valid (`self`) → `bool`

Returns whether this client and its connection(s) are valid.

Returns whether this client and its connection(s) are valid

Return type `bool`

```
pine.client.setup_logging()
```

Sets up logging, if configured to do so.

The environment variable named by `CONFIG_FILE_ENV` is checked and, if present, is passed to `logging.config.dictConfig()`.

```
class pine.client.CollectionBuilder (collection: dict = None, creator_id: str = None, view-
                                ers: List[str] = None, annotators: List[str] = None,
                                labels: List[str] = None, title: str = None, descrip-
                                tion: str = None, allow_overlapping_ner_annotations:
                                bool = None, pipelineId: str = None, overlap: float
                                = None, train_every: int = None, classifierParamete-
                                rs: dict = None, document_csv_file: str = None,
                                document_csv_file_has_header: bool = None, docu-
                                ment_csv_file_text_column: int = None, image_files:
                                List[str] = None)
```

Bases: `object`

A class that can build the form and files fields that are necessary to create a collection.

Constructor.

Parameters

- **collection** (*dict*, *optional*) – starting parameters for the collection, defaults to None (not set)
- **creator_id** (*str*, *optional*) – user ID for the creator, see `creator_id()`, defaults to None (not set)
- **viewers** (*list(str)*, *optional*) – viewer IDs for the collection, see `viewer()`, defaults to None (not set)
- **annotators** (*list(str)*, *optional*) – annotator IDs for the collection, see `annotator()`, defaults to None (not set)
- **labels** (*list(str)*, *optional*) – labels for the collection, see `label()`, defaults to None (not set)
- **title** (*str*, *optional*) – metadata title, see `title()`, defaults to None (not set)
- **description** (*str*, *optional*) – metadata description, see `description()`, defaults to None (not set)
- **allow_overlapping_ner_annotations** (*bool*) – optional configuration for allowing overlapping NER annotations, see `allow_overlapping_ner_annotations()`, defaults to None (not set)
- **pipelineId** (*str*, *optional*) – the ID of the pipeline from which to create the classifier, see `classifier()`, defaults to None (not set)
- **overlap** (*float*, *optional*) – the classifier overlap, see `classifier()`, defaults to None (not set)
- **train_every** (*int*, *optional*) – train the model after this many documents are annotated, see `classifier()`, defaults to None (not set)
- **classifierParameters** (*dict*, *optional*) – any parameters to pass to the classifier, see `classifier()`, defaults to None (not set)
- **document_csv_file** (*str*, *optional*) – the filename of the local document CSV file, see `document_csv_File()`, defaults to None (not set)
- **document_csv_file_has_header** (*bool*, *optional*) – whether the document CSV file has a header, see `document_csv_File()`, defaults to None (not set)
- **document_csv_file_text_column** (*int*, *optional*) – if the document CSV file has headers, the document text can be found in this column index (the others are used for document metadata), see `document_csv_File()`, defaults to None (not set)

- **image_files** (*list(str)*) – any image files to add to the collection, see *image_file()*, defaults to None (not set)

form

The form data.

Type *dict*

files

The files data.

Type *dict*

property collection (*self*) → *dict*

Returns the collection information from the form.

Returns collection information from the form

Return type *dict*

property form_json (*self*) → *dict*

Returns the form where the values have been JSON-encoded.

Returns the form where the values have been JSON-encoded

Return type *dict*

creator_id (*self, user_id: str*) → *pine.client.models.CollectionBuilder*

Sets the creator_id to the given, and adds to viewers and annotators.

Parameters **user_id** (*str*) – the user ID to use for the creator_id

Returns *self*

Return type *models.CollectionBuilder*

viewer (*self, user_id: str*) → *pine.client.models.CollectionBuilder*

Adds the given user to the list of viewers.

Parameters **user_id** (*str*) – the user ID to add as a viewer

Returns *self*

Return type *models.CollectionBuilder*

annotator (*self, user_id: str*) → *pine.client.models.CollectionBuilder*

Adds the given user to the list of annotators.

Parameters **user_id** (*str*) – the user ID to add as an annotator

Returns *self*

Return type *models.CollectionBuilder*

label (*self, label: str*) → *pine.client.models.CollectionBuilder*

Adds the given label to the collection.

Parameters **label** (*str*) – label to add

Returns *self*

Return type *models.CollectionBuilder*

metadata (*self, key: str, value: Any*) → *pine.client.models.CollectionBuilder*

Adds the given metadata key/value to the collection.

Parameters

- **key** (*str*) – metadata key
- **value** – metadata value

Returns self

Return type *models.CollectionBuilder*

title (*self*, *title*: *str*) → *pine.client.models.CollectionBuilder*

Sets the metadata title to the given.

Parameters **title** (*str*) – collection title

Returns self

Return type *models.CollectionBuilder*

description (*self*, *description*: *str*) → *pine.client.models.CollectionBuilder*

Sets the metadata description to the given.

Parameters **description** (*str*) – collection description

Returns self

Return type *models.CollectionBuilder*

configuration (*self*, *key*: *str*, *value*: *Any*) → *pine.client.models.CollectionBuilder*

Adds the given configuration key/value to the collection.

Parameters

- **key** (*str*) – configuration key
- **value** – configuration value

Returns self

Return type *models.CollectionBuilder*

allow_overlapping_ner_annotations (*self*, *allow_overlapping_ner_annotations*: *bool*)

Sets the configuration value for `allow_overlapping_ner_annotations` to the given.

Parameters **allow_overlapping_ner_annotations** (*bool*) – whether to allow overlapping NER annotations

Returns self

Return type *models.CollectionBuilder*

classifier (*self*, *pipelineId*: *str*, *overlap*: *float* = 0, *train_every*: *int* = 100, *classifierParameters*: *dict* = {}) → *pine.client.models.CollectionBuilder*

Sets classifier information for the created collection.

Parameters

- **pipelineId** (*str*) – the ID of the pipeline from which to create the classifier
- **overlap** (*float*, *optional*) – the classifier overlap, defaults to 0
- **train_every** (*int*, *optional*) – train the model after this many documents are annotated, defaults to 100
- **classifierParameters** (*dict*, *optional*) – any parameters to pass to the classifier, defaults to { }

Returns self

Return type *models.CollectionBuilder*

document_csv_file (*self*, *csv_filename*: *str*, *has_header*: *bool*, *text_column*: *int*) → *pine.client.models.CollectionBuilder*

Sets the CSV file used to create documents to the given.

May raise an Exception if there is a problem opening the indicated file.

Parameters

- **csv_filename** (*str*) – the filename of the local CSV file
- **has_header** (*bool*) – whether the CSV file has a header
- **text_column** (*int*) – if the CSV file has headers, the document text can be found in this column index (the others are used for document metadata)

Returns *self*

Return type *models.CollectionBuilder*

image_file (*self*, *image_filename*: *str*) → *pine.client.models.CollectionBuilder*

Adds the given image file to the collection.

May raise an Exception if there is a problem opening the indicated file.

Parameters **image_filename** (*str*) – the filename of the local image file

Returns *self*

Return type *models.CollectionBuilder*

is_valid (*self*, *error_callback*: *Callable[[str], None]* = *None*)

Checks whether the currently set values are valid or not.

See *is_valid_collection()*.

Parameters **error_callback** (*function*, *optional*) – optional callback that is called with any error messages, defaults to *None*

Returns whether the currently set values are valid or not

Return type *bool*

pine.pipelines

Subpackages

pine.pipelines.app

Subpackages

pine.pipelines.app.listener

Submodules

pine.pipelines.app.listener.main

Module Contents

Functions

`backoff_hdlr(details)`

`setup_logging()`

`run()`

`pine.pipelines.app.listener.main.backoff_hdlr(details)`
`pine.pipelines.app.listener.main.setup_logging()`
`pine.pipelines.app.listener.main.run()`
`pine.pipelines.app.listener.service_listener`

Module Contents

Classes

`ServiceRegistration`

`ServiceListener`

`type services list[ServiceRegistration]`

`pine.pipelines.app.listener.service_listener.config`
`pine.pipelines.app.listener.service_listener.logger`

```
class pine.pipelines.app.listener.service_listener.ServiceRegistration (name,
                                                                    ver-
                                                                    sion,
                                                                    chan-
                                                                    nel,
                                                                    frame-
                                                                    work,
                                                                    frame-
                                                                    work_types)
```

Bases: `object`

classmethod `from_registration_format(cls, **kwargs)`

to_registration_format(self)

```
class pine.pipelines.app.listener.service_listener.ServiceListener (services=None)
```

Bases: `object`

r_pool

r_conn

```
registration_poll
listener_poll
processor_poll
processing_limit
processing_queue_key
processing_queue_key_timeout
processing_lock_key
processing_lock_key_timeout
preprocessing_lock_key
preprocessing_lock_key_timeout
preprocessing_worker_lock_key
preprocessing_worker_lock_key_timeout
registration_channel
classmethod publish_response(cls, channel, data)
    Return type bool
start_workers(self)
stop_workers(self)
pre_process_message(self, message_channel, message_data)
    Return type bool | dict
static process_message(job_details)
_start_registration_task(self)
_start_channel_task(self)
_start_listener_task(self)
_start_queue_processor_task(self)
```

`pine.pipelines.shared`

Submodules

`pine.pipelines.shared.config`

Module Contents

Classes

BaseConfig

continues on next page

Table 48 – continued from previous page

*TestConfig**ConfigBuilder*

```

pine.pipelines.shared.config.LOGGER

class pine.pipelines.shared.config.BaseConfig(root_dir=None)
    Bases: object

    ROOT_DIR
    BASE_DIR
    BASE_CFG_FILE = config.yaml
    BASE_ENV_PREFIX = AL_
    DEBUG = True
    TESTING = False
    LOGGER_NAME
    LOGGER_DIR = logs
    LOGGER_FILE = debug.log
    LOGGER_LEVEL
    PIPELINE = opennlp
    EVE_HOST = localhost
    EVE_PORT = 5001
    REDIS_HOST = localhost
    REDIS_PORT = 6379
    REDIS_USR
    REDIS_PWD
    REDIS_DBNUM = 0
    REDIS_PREFIX = AL:
    REDIS_EXPIRE = 3600
    SCHEDULER_REGISTRATION_TIMEOUT
    SCHEDULER_HANDLER_TIMEOUT
    SCHEDULER_QUEUE_TIMEOUT
    SERVICE_REGISTRATION_CHANNEL = registration
    SERVICE_REGISTRATION_FREQUENCY = 60
    SERVICE_LISTENING_FREQUENCY = 1
    SERVICE_HANDLER_TIMEOUT = 60
    SERVICE_LIST
    MODELS_DIR

```

```
classmethod _get_config_var_paths (cls, root_dict=None)
classmethod _process_paths (cls, alt_path=None)
classmethod _process_file_cfg (cls)
classmethod _process_env_vars (cls)
classmethod as_dict (cls)
    Return type dict
classmethod as_attr_dict (cls)
    Return type munch.Munch | dict
static _try_cast (value, _type, _default=None)
static _str2bool (_str, _default=None)
class pine.pipelines.shared.config.TestConfig (root_dir=None)
    Bases: pine.pipelines.shared.config.BaseConfig
class pine.pipelines.shared.config.ConfigBuilder
    Bases: object
    __env_cfg_variable = BUILDER_CFG_PROFILE
    __current_config_instance
    __current_config_instance_name
    __current_config_instance_print = False
    __arg_parser
    static __get_configs ()
        :rtype list[Callable[... BaseConfig]]
    static get_config_names ()
        Return type list[str]
    classmethod get_arg_parser (cls)
        Return type ArgumentParser
    classmethod init_config (cls, config_name=None, config_base=None, enable_terminal=True,
        as_attr_dict=True)
    classmethod get_config (cls, config_name=None, config_base=None, enable_terminal=True,
        as_attr_dict=True)
        Return type BaseConfig
    classmethod set_config (cls, config_name=None, config_base=None, enable_terminal=True,
        as_attr_dict=True)
    classmethod __parse_terminal_config (cls)
        Return type argparse.Namespace
```

`pine.pipelines.shared.transform`

Module Contents

Functions

<code>transform_module_by_config(module_ref, config_ref, config_prefix=None)</code>	Transforms a given module's properties based on ConfigBuilder Values.
---	---

`pine.pipelines.shared.transform.transform_module_by_config(module_ref, config_ref, config_prefix=None)`

Transforms a given module's properties based on ConfigBuilder Values. The prefix can be used to avoid blindly changing values and target a subset of matching values in config_ref. :type module_ref: ModuleType :type config_ref: dict :type config_prefix: str

Submodules

`pine.pipelines.EveClient`

Module Contents

Classes

<code>EveClient</code>

`pine.pipelines.EveClient.logger`

`pine.pipelines.EveClient.config`

class `pine.pipelines.EveClient.EveClient` (`entry_point='{}:{'}.format(config.EVE_HOST, config.EVE_PORT)`)

Bases: `object`

eve_headers

add (`self, resource, add_object`)

get_obj (`self, resource, id`)

get_all_items (`self, resource`)

get_all_ids (`self, resource`)

get_items (`self, resource`)

get_documents (`self, collection_id`)

get_docs_with_annotations (`self, collection_id, doc_map`)

update (`self, resource, id, etag, update_obj`)

`pine.pipelines.NERWrapper`

Module Contents

Classes

NERWrapper

```
class pine.pipelines.NERWrapper.NERWrapper (classifier_type, model_dir='models', entry_point='localhost:5000')
```

```
    eve_headers
```

```
    load_classifier (self, classifier_id)
```

```
    predict (self, classifier_id, documents, document_ids)
```

```
    update_model (self, classifier_id)
```

```
    update (self, resource, id, etag, update_obj)
```

```
    get_obj (self, resource, id)
```

```
    get_all_items (self, resource)
```

```
    get_all_ids (self, resource)
```

```
    get_items (self, resource)
```

```
pine.pipelines.NERWrapper.parser
```

`pine.pipelines.NER_API`

Module Contents

Classes

ner_api

```
pine.pipelines.NER_API.logger
```

```
pine.pipelines.NER_API.config
```

```
class pine.pipelines.NER_API.ner_api
```

```
    Bases: object
```

```
    perform_fold (self, model, train_data, test_data, pipeline_parameters)
```

```
    perform_five_fold (self, model, documents, annotations, doc_ids, pipeline_parameters)
```

```
    get_document_ranking (self, model, doc_map, doc_ids)
```

```
    get_classifier_pipeline_metrics_objs (self, classifier_id)
```

```
    train_model (self, custom_filename, classifier_id, pipeline_name)
```

```
predict (self, classifier_id, pipeline_name, documents, document_ids)
```

```
pine.pipelines.RankingFunctions
```

Module Contents

Functions

<i>rank</i> (results, metric)	if metric == 'lc': return least_confidence(results)
<i>least_confidence</i> (results)	
<i>least_confidence_squared</i> (results)	
<i>least_confidence_squared_by_entity</i> (results)	
<i>largest_margin</i> (results)	
<i>entropy_rank</i> (results, N=None)	
<i>random_rank</i> (results)	
<i>most_of_least_popular</i> (results)	

```
pine.pipelines.RankingFunctions.rank (results, metric)
    if metric == 'lc': return least_confidence(results) if metric == 'ma': return largest_margin(results) if metric ==
    'en': return entropy_rank(results) if metric == 'lcs': return least_confidence_squared(results) if metric == 'lce':
    return least_confidence_squared_by_entity(results) if metric == 'ra': return random_rank(results) if metric ==
    'mlp': return most_of_least_popular(results) return -1
```

```
    #Dictionary method is inefficient as it runs every method before returning one
```

```
pine.pipelines.RankingFunctions.least_confidence (results)
```

```
pine.pipelines.RankingFunctions.least_confidence_squared (results)
```

```
pine.pipelines.RankingFunctions.least_confidence_squared_by_entity (results)
```

```
pine.pipelines.RankingFunctions.largest_margin (results)
```

```
pine.pipelines.RankingFunctions.entropy_rank (results, N=None)
```

```
pine.pipelines.RankingFunctions.random_rank (results)
```

```
pine.pipelines.RankingFunctions.most_of_least_popular (results)
```

`pine.pipelines.corenlp_NER_pipeline`

Module Contents

Classes

`corenlp_NER`

`pine.pipelines.corenlp_NER_pipeline.config`

`pine.pipelines.corenlp_NER_pipeline.logger`

class `pine.pipelines.corenlp_NER_pipeline.corenlp_NER` (*java_dir=None,*
ner_path=None,
load_model=None,
tmp_dir=None)

Bases: `pine.pipelines.pipeline.Pipeline`

`__jar =`

`__jdk_dir =`

`__train_file =`

`__test_file =`

`__model =`

`__temp_dir`

`__crf`

`__props`

`__id`

`fit` (*self, X, y, params=None*)

`predict` (*self, X, Xid*)

`predict_proba` (*self, X, Xid, get_all_labels=False, include_other=False*)

`next_example` (*self, X, Xid*)

`get_id` (*self*)

`format_data` (*self, X, y*)

`save_model` (*self, model_name*)

`load_model` (*self, model_name*)

`tokenize` (*self, input_text*)

`evaluate` (*self, X, y, Xid, verbose=False*)

`evaluate_orig` (*self, X, y, Xid*)

`pine.pipelines.opennlp_NER_pipeline`

Module Contents

Classes

opennlp_NER

`pine.pipelines.opennlp_NER_pipeline.config`

`pine.pipelines.opennlp_NER_pipeline.logger`

class `pine.pipelines.opennlp_NER_pipeline.opennlp_NER` (*java_dir=None, ner_path=None, tmp_dir=None*)

Bases: *pine.pipelines.pipeline.Pipeline*

`__jar =`

`__jdk_dir =`

`__temp_dir`

`__train_file =`

`__test_file =`

`__model`

`__nameFinder`

`__id`

`fit` (*self, X, y, params*)

`predict` (*self, X, Xid*)

`predict_proba` (*self, X, Xid, get_all_labels=False, include_other=False*)

`next_example` (*self, X, Xid*)

`save_model` (*self, model_name*)

`load_model` (*self, model_name*)

`find_all_init` (*self*)

`find_all` (*self, tokens*)

`get_id` (*self*)

`format_data` (*self, X, y*)

`convert_ann_collection_to_per_token` (*self, annotations, tokens*)

`evaluate` (*self, X, y, Xid*)

`evaluate_orig` (*self, X, y, Xid*)

pine.pipelines.pipeline

Module Contents

Classes

Pipeline

```
class pine.pipelines.pipeline.Pipeline
    Bases: object

    abstract fit (self, X, y)
    abstract predict (self, X, Xid)
    abstract predict_proba (self, X, Xid)
    abstract next_example (self, X, Xid)
    abstract save_model (self, model_name)
    abstract load_model (self, model_name)
```

pine.pipelines.pmap_ner

Module Contents

Classes

NER

pine.pipelines.pmap_ner.logger

```
class pine.pipelines.pmap_ner.NER (lib=None, **kwargs)
    Bases: pine.pipelines.pipeline.Pipeline

    __lib =
    pipeline
    __SUPPORTED_PIPELINES = ['spacy', 'corenlp', 'opennlp']
    pipe_init (self, x, **kwargs)
    fit (self, X, y, kwargs)
    predict (self, X, Xid)
    predict_proba (self, X, Xid, **kwargs)
    evaluate (self, X, y, Xid, **kwargs)
    next_example (self, X, Xid)
    save_model (self, model_path)
```

```
load_model (self, model_name)
```

```
pine.pipelines.run_service
```

```
pine.pipelines.spacy_NER_pipeline
```

For more details, see the documentation: * Training: <https://spacy.io/usage/training> * NER: <https://spacy.io/usage/linguistic-features#named-entities>

Compatible with: spaCy v2.0.0+

Module Contents

Classes

spacy_NER

```
class pine.pipelines.spacy_NER_pipeline.spacy_NER (model_path=None)
```

```
    Bases: pine.pipelines.pipeline.Pipeline
```

```
    __nlp = []
```

```
    __ner = []
```

```
    __optimizer = []
```

```
    fit (self, X, y, params=None)
```

```
    evaluate (self, X, y, Xid)
```

```
    predict (self, X, Xid)
```

```
    predict_proba (self, X, Xid)
```

```
    next_example (self, X, Xid)
```

```
    format_data (self, X, y)
```

```
    add_label (self, entity)
```

```
    save_model (self, model_path)
```

```
    load_model (self, model_path)
```


INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

p

- pine, 1
- pine.backend, 1
 - pine.backend.admin, 1
 - pine.backend.admin.bp, 1
 - pine.backend.annotations, 2
 - pine.backend.annotations.bp, 2
 - pine.backend.app, 30
 - pine.backend.auth, 4
 - pine.backend.auth.bp, 4
 - pine.backend.auth.eve, 6
 - pine.backend.auth.oauth, 6
 - pine.backend.auth.password, 7
 - pine.backend.auth.vegas, 7
 - pine.backend.collections, 8
 - pine.backend.collections.bp, 8
 - pine.backend.config, 31
 - pine.backend.cors, 31
 - pine.backend.data, 12
 - pine.backend.data.bp, 12
 - pine.backend.data.service, 12
 - pine.backend.data.users, 15
 - pine.backend.documents, 16
 - pine.backend.documents.bp, 16
 - pine.backend.job_manager, 18
 - pine.backend.job_manager.service, 18
 - pine.backend.log, 31
 - pine.backend.models, 33
 - pine.backend.pineiaa, 20
 - pine.backend.pineiaa.bp, 26
 - pine.backend.pineiaa.bratiaa, 20
 - pine.backend.pineiaa.bratiaa.agree, 20
 - pine.backend.pineiaa.bratiaa.agree_cli, 22
 - pine.backend.pineiaa.bratiaa.evaluation, 22
 - pine.backend.pineiaa.bratiaa.iaa_service, 23
 - pine.backend.pineiaa.bratiaa.utils, 23
 - pine.backend.pineiaa.bratiaa.iaa_service, 23
 - pine.backend.pipelines, 26
 - pine.backend.pipelines.bp, 26
 - pine.backend.shared, 28
 - pine.backend.shared.config, 28
 - pine.backend.shared.transform, 30
 - pine.client, 35
 - pine.client.client, 35
 - pine.client.exceptions, 47
 - pine.client.log, 48
 - pine.client.models, 48
 - pine.client.password, 60
 - pine.pipelines, 70
 - pine.pipelines.app, 70
 - pine.pipelines.app.listener, 70
 - pine.pipelines.app.listener.main, 70
 - pine.pipelines.app.listener.service_listener, 71
 - pine.pipelines.corenlp_NER_pipeline, 78
 - pine.pipelines.EveClient, 75
 - pine.pipelines.NER_API, 76
 - pine.pipelines.NERWrapper, 76
 - pine.pipelines.opennlp_NER_pipeline, 79
 - pine.pipelines.pipeline, 80
 - pine.pipelines.pmap_ner, 80
 - pine.pipelines.RankingFunctions, 77
 - pine.pipelines.run_service, 81
 - pine.pipelines.shared, 72
 - pine.pipelines.shared.config, 72
 - pine.pipelines.shared.transform, 75
 - pine.pipelines.spacy_NER_pipeline, 81

INDEX

Symbols

`__SUPPORTED_PIPELINES`
(*pine.pipelines.pmap_ner.NER* attribute), 80

`__arg_parser` (*pine.backend.shared.config.ConfigBuilder* attribute), 29

`__arg_parser` (*pine.pipelines.shared.config.ConfigBuilder* attribute), 74

`__crf` (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER* attribute), 78

`__current_config_instance`
(*pine.backend.shared.config.ConfigBuilder* attribute), 29

`__current_config_instance`
(*pine.pipelines.shared.config.ConfigBuilder* attribute), 74

`__current_config_instance_name`
(*pine.backend.shared.config.ConfigBuilder* attribute), 29

`__current_config_instance_name`
(*pine.pipelines.shared.config.ConfigBuilder* attribute), 74

`__current_config_instance_print`
(*pine.backend.shared.config.ConfigBuilder* attribute), 29

`__current_config_instance_print`
(*pine.pipelines.shared.config.ConfigBuilder* attribute), 74

`__env_cfg_variable`
(*pine.backend.shared.config.ConfigBuilder* attribute), 29

`__env_cfg_variable`
(*pine.pipelines.shared.config.ConfigBuilder* attribute), 74

`__get_configs()` (*pine.backend.shared.config.ConfigBuilder* static method), 29

`__get_configs()` (*pine.pipelines.shared.config.ConfigBuilder* static method), 74

`__id` (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER* attribute), 78

`__id` (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER* attribute), 79

`__jar` (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER* attribute), 78

`__jar` (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER* attribute), 79

`__jdk_dir` (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER* attribute), 78

`__jdk_dir` (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER* attribute), 79

`__lib` (*pine.pipelines.pmap_ner.NER* attribute), 80

`__metaclass__` (*pine.backend.auth.bp.AuthModule* attribute), 5

`__metaclass__` (*pine.client.client.BaseClient* attribute), 36

`__model` (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER* attribute), 78

`__model` (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER* attribute), 79

`__nameFinder` (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER* attribute), 79

`__ner` (*pine.pipelines.spacy_NER_pipeline.spacy_NER* attribute), 81

`__nlp` (*pine.pipelines.spacy_NER_pipeline.spacy_NER* attribute), 81

`__optimizer` (*pine.pipelines.spacy_NER_pipeline.spacy_NER* attribute), 81

`__parse_terminal_config()`
(*pine.backend.shared.config.ConfigBuilder* class method), 30

`__parse_terminal_config()`
(*pine.pipelines.shared.config.ConfigBuilder* class method), 74

`__props` (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER* attribute), 78

`__slots__` (*pine.backend.pineiaa.bratiaa.Document* attribute), 25

`__slots__` (*pine.backend.pineiaa.bratiaa.agree.Document* attribute), 21

`__temp_dir` (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER* attribute), 78

`__temp_dir` (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER* attribute), 79

`__test_file` (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER*

attribute), 78
 __test_file(*pine.pipelines.opennlp_NER_pipeline.opennlp_NERRule pine.backend.documents.bp*), 17
attribute), 79
 __train_file(*pine.pipelines.corenlp_NER_pipeline.corenlp_NERRule pine.backend.pipelines.bp*), 27
attribute), 78
 __train_file(*pine.pipelines.opennlp_NER_pipeline.opennlp_NERRule pine.backend.pipelines.bp*), 27
attribute), 79
 __version__ (in module *pine.backend*), 35
 _add_or_replace_resource()
 (*pine.client.client.EveClient* method), 39
 _add_or_update_annotation() (in module
 pine.backend.annotations.bp), 4
 _add_resources() (*pine.client.client.EveClient*
 method), 39
 _channel_watchdog()
 (*pine.backend.job_manager.service.ServiceManager*
 method), 20
 _check_collection_and_get_image_dir()
 (in module *pine.backend.collections.bp*), 11
 _check_documents() (in module
 pine.backend.documents.bp), 17
 _check_field_bool() (in module
 pine.client.models), 52
 _check_field_dict() (in module
 pine.client.models), 51
 _check_field_float() (in module
 pine.client.models), 50
 _check_field_int() (in module
 pine.client.models), 50
 _check_field_required_bool() (in module
 pine.client.models), 49
 _check_field_required_dict() (in module
 pine.client.models), 51
 _check_field_required_float() (in module
 pine.client.models), 50
 _check_field_required_int() (in module
 pine.client.models), 50
 _check_field_required_string() (in module
 pine.client.models), 51
 _check_field_required_string_list() (in
 module *pine.client.models*), 51
 _check_field_string() (in module
 pine.client.models), 50
 _check_field_string_list() (in module
 pine.client.models), 51
 _check_login() (*pine.client.PineClient* method), 61
 _check_login() (*pine.client.client.PineClient*
 method), 42
 _compute_tp_fp_fn()
 (*pine.backend.pineiaa.bratiaa.F1Agreement*
 method), 25
 _compute_tp_fp_fn()
 (*pine.backend.pineiaa.bratiaa.agree.F1Agreement*
 method), 21
 _document_user_can_projection() (in mod-
 ule
 _get_classifier() (in module
 pine.backend.pipelines.bp), 27
 _get_classifier_metrics() (in module
 pine.backend.pipelines.bp), 27
 _get_collection_classifier() (in module
 pine.backend.pipelines.bp), 27
 _get_config_var_paths()
 (*pine.backend.shared.config.BaseConfig* class
 method), 29
 _get_config_var_paths()
 (*pine.pipelines.shared.config.BaseConfig*
 class method), 73
 _get_next_instance() (in module
 pine.backend.pipelines.bp), 27
 _increment_counts()
 (*pine.backend.pineiaa.bratiaa.F1Agreement*
 method), 25
 _increment_counts()
 (*pine.backend.pineiaa.bratiaa.agree.F1Agreement*
 method), 21
 _make_annotations() (in module
 pine.backend.annotations.bp), 4
 _mean_sd() (*pine.backend.pineiaa.bratiaa.F1Agreement*
 static method), 25
 _mean_sd() (*pine.backend.pineiaa.bratiaa.agree.F1Agreement*
 static method), 21
 _pairs_involving()
 (*pine.backend.pineiaa.bratiaa.F1Agreement*
 method), 25
 _pairs_involving()
 (*pine.backend.pineiaa.bratiaa.agree.F1Agreement*
 method), 21
 _path_split() (in module
 pine.backend.collections.bp), 11
 _process_env_vars()
 (*pine.backend.shared.config.BaseConfig* class
 method), 29
 _process_env_vars()
 (*pine.pipelines.shared.config.BaseConfig*
 class method), 74
 _process_file_cfg()
 (*pine.backend.shared.config.BaseConfig* class
 method), 29
 _process_file_cfg()
 (*pine.pipelines.shared.config.BaseConfig*
 class method), 74
 _process_paths() (*pine.backend.shared.config.BaseConfig*
 class method), 29
 _process_paths() (*pine.pipelines.shared.config.BaseConfig*
 class method), 74
 _processing_listener()
 (*pine.backend.job_manager.service.ServiceManager*

method), 20

`_processing_listener_handler_wrapper()`
(*pine.backend.job_manager.service.ServiceManager* method), 20

`_read_token_annotations()` (in module *pine.backend.pineiaa.bratiaa.evaluation*), 22

`_registration_listener()`
(*pine.backend.job_manager.service.ServiceManager* method), 19

`_req()` (*pine.client.client.BaseClient* method), 36

`_safe_path()` (in module *pine.backend.collections.bp*), 11

`_standardize_path()` (in module *pine.backend.data.service*), 13

`_standardize_path()` (in module *pine.client.client*), 35

`_start_channel_task()`
(*pine.pipelines.app.listener.service_listener.ServiceListener* method), 72

`_start_channel_watchdog()`
(*pine.backend.job_manager.service.ServiceManager* method), 19

`_start_listener_task()`
(*pine.pipelines.app.listener.service_listener.ServiceListener* method), 72

`_start_processing_listeners()`
(*pine.backend.job_manager.service.ServiceManager* method), 19

`_start_queue_processor_task()`
(*pine.pipelines.app.listener.service_listener.ServiceListener* method), 72

`_start_registration_listener()`
(*pine.backend.job_manager.service.ServiceManager* method), 19

`_start_registration_task()`
(*pine.pipelines.app.listener.service_listener.ServiceListener* method), 72

`_stop_channel_watchdog()`
(*pine.backend.job_manager.service.ServiceManager* method), 19

`_str2bool()` (*pine.backend.shared.config.BaseConfig* static method), 29

`_str2bool()` (*pine.pipelines.shared.config.BaseConfig* static method), 74

`_thread_killer()` (*pine.backend.job_manager.service.ServiceManager* static method), 20

`_try_cast()` (*pine.backend.shared.config.BaseConfig* static method), 29

`_try_cast()` (*pine.pipelines.shared.config.BaseConfig* static method), 74

`_upload_collection_image_file()` (in module *pine.backend.collections.bp*), 11

`_upload_documents()` (in module *pine.backend.collections.bp*), 11

`about()` (*pine.client.client.EveClient* method), 39

`about()` (*pine.client.client.PineClient* method), 41

`about()` (*pine.client.PineClient* method), 61

`access()` (in module *pine.backend.log*), 33

`access_flask_add_collection()` (in module *pine.backend.log*), 33

`access_flask_add_document()` (in module *pine.backend.log*), 33

`access_flask_add_documents()` (in module *pine.backend.log*), 33

`access_flask_annotate_document()` (in module *pine.backend.log*), 33

`access_flask_annotate_documents()` (in module *pine.backend.log*), 33

`access_flask_login()` (in module *pine.backend.log*), 33

`access_flask_logout()` (in module *pine.backend.log*), 33

`access_flask_view_document()` (in module *pine.backend.log*), 33

`ACCESS_LOGGER` (in module *pine.backend.log*), 32

`ACCESS_LOGGER_NAME` (in module *pine.backend.log*), 32

`Action` (class in *pine.backend.log*), 32

`add()` (*pine.backend.data.service.PerformanceHistory* method), 13

`add()` (*pine.pipelines.EveClient.EveClient* method), 75

`add_admin_command()` (in module *pine.backend.data.users*), 16

`add_annotator_to_collection()` (in module *pine.backend.collections.bp*), 10

`ADD_DOCUMENT` (*pine.backend.log.Action* attribute), 32

`add_document()` (in module *pine.backend.documents.bp*), 17

`add_document()` (*pine.client.client.PineClient* method), 43

`add_document()` (*pine.client.PineClient* method), 63

`ADD_DOCUMENTS` (*pine.backend.log.Action* attribute), 32

`add_documents` (*pine.backend.models.CollectionUserPermissions* attribute), 34

`add_documents` (*pine.client.models.CollectionUserPermissions* attribute), 59

`add_documents()` (*pine.client.client.PineClient* method), 44

`add_documents()` (*pine.client.PineClient* method), 64

`add_images` (*pine.backend.models.CollectionUserPermissions* attribute), 34

`add_images` (*pine.client.models.CollectionUserPermissions* attribute), 59

[add_label\(\)](#) (*pine.pipelines.spacy_NER_pipeline.spacy_NER_pipeline* attribute), 81
[add_label_to_collection\(\)](#) (*in module pine.backend.collections.bp*), 11
[add_pipelines\(\)](#) (*pine.client.client.EveClient* method), 40
[add_user\(\)](#) (*in module pine.backend.admin.bp*), 2
[add_users\(\)](#) (*pine.client.client.EveClient* method), 40
[add_viewer_to_collection\(\)](#) (*in module pine.backend.collections.bp*), 10
[admin_required\(\)](#) (*in module pine.backend.auth*), 8
[admin_required\(\)](#) (*in module pine.backend.auth.bp*), 5
[advance_to_next_document_by_classifier\(\)](#) (*in module pine.backend.pipelines.bp*), 27
[allow_overlapping_ner_annotations\(\)](#) (*pine.client.CollectionBuilder* method), 69
[allow_overlapping_ner_annotations\(\)](#) (*pine.client.models.CollectionBuilder* method), 57
[AnnFile](#) (*in module pine.backend.pineiaa.bratiaa*), 25
[AnnFile](#) (*in module pine.backend.pineiaa.bratiaa.agree*), 20
[annotate](#) (*pine.backend.models.CollectionUserPermissions* attribute), 34
[annotate](#) (*pine.client.models.CollectionUserPermissions* attribute), 59
[annotate_collection_documents\(\)](#) (*pine.client.client.PineClient* method), 45
[annotate_collection_documents\(\)](#) (*pine.client.PineClient* method), 64
[ANNOTATE_DOCUMENT](#) (*pine.backend.log.Action* attribute), 32
[annotate_document\(\)](#) (*pine.client.client.PineClient* method), 44
[annotate_document\(\)](#) (*pine.client.PineClient* method), 64
[ANNOTATE_DOCUMENTS](#) (*pine.backend.log.Action* attribute), 32
[Annotation](#) (*in module pine.backend.pineiaa.bratiaa*), 25
[Annotation](#) (*in module pine.backend.pineiaa.bratiaa.agree*), 20
[Annotation](#) (*in module pine.backend.pineiaa.bratiaa.evaluation*), 22
[annotator\(\)](#) (*pine.client.CollectionBuilder* method), 68
[annotator\(\)](#) (*pine.client.models.CollectionBuilder* method), 56
[annotators\(\)](#) (*pine.backend.pineiaa.bratiaa.agree.FIAgreement* property), 21
[annotators\(\)](#) (*pine.backend.pineiaa.bratiaa.FIAgreement* property), 25
[archive](#) (*pine.backend.models.CollectionUserPermissions* attribute), 34
[archive](#) (*pine.client.models.CollectionUserPermissions* attribute), 59
[archive_collection\(\)](#) (*in module pine.backend.collections.bp*), 10
[archive_or_unarchive_collection\(\)](#) (*in module pine.backend.collections.bp*), 10
[as_attr_dict\(\)](#) (*pine.backend.shared.config.BaseConfig* class method), 29
[as_attr_dict\(\)](#) (*pine.pipelines.shared.config.BaseConfig* class method), 74
[as_dict\(\)](#) (*pine.backend.shared.config.BaseConfig* class method), 29
[as_dict\(\)](#) (*pine.pipelines.shared.config.BaseConfig* class method), 74
[AUTH_MODULE](#) (*in module pine.backend.config*), 31
[AuthModule](#) (*class in pine.backend.auth.bp*), 5
[authorize\(\)](#) (*pine.backend.auth.oauth.OAuthModule* method), 7
[AuthUser](#) (*class in pine.backend.models*), 33

B

[Backoff_hdlr\(\)](#) (*in module pine.pipelines.app.listener.main*), 71
[BASE_CFG_FILE](#) (*pine.backend.shared.config.BaseConfig* attribute), 28
[BASE_CFG_FILE](#) (*pine.pipelines.shared.config.BaseConfig* attribute), 73
[BASE_DIR](#) (*pine.backend.shared.config.BaseConfig* attribute), 28
[BASE_DIR](#) (*pine.pipelines.shared.config.BaseConfig* attribute), 73
[BASE_ENV_PREFIX](#) (*pine.backend.shared.config.BaseConfig* attribute), 28
[BASE_ENV_PREFIX](#) (*pine.pipelines.shared.config.BaseConfig* attribute), 73
[base_uri](#) (*pine.client.client.BaseClient* attribute), 36
[BaseClient](#) (*class in pine.client.client*), 36
[BaseConfig](#) (*class in pine.backend.shared.config*), 28
[BaseConfig](#) (*class in pine.pipelines.shared.config*), 73
[bp](#) (*in module pine.backend.admin.bp*), 2
[bp](#) (*in module pine.backend.annotations.bp*), 3
[bp](#) (*in module pine.backend.auth.bp*), 5
[bp](#) (*in module pine.backend.collections.bp*), 10
[bp](#) (*in module pine.backend.documents.bp*), 17
[bp](#) (*in module pine.backend.pineiaa.bp*), 26
[bp](#) (*in module pine.backend.pipelines.bp*), 27

C

[can_manage_users\(\)](#) (*pine.backend.auth.bp.AuthModule* method), 5
[can_manage_users\(\)](#) (*pine.backend.auth.eve.EveModule* method), 6

`can_manage_users()` (*pine.backend.auth.oauth.OAuthModule* method), 7
`channel_worker_name` (*pine.backend.job_manager.service.ServiceManager* attribute), 19
`check_document_annotate()` (in module *pine.backend.annotations.bp*), 3
`check_document_view()` (in module *pine.backend.annotations.bp*), 3
`check_document_view_by_id()` (in module *pine.backend.annotations.bp*), 3
`check_overlapping_annotations()` (in module *pine.backend.annotations.bp*), 4
`check_password()` (in module *pine.backend.auth.password*), 7
`check_password()` (in module *pine.client.password*), 60
`classifier()` (*pine.client.CollectionBuilder* method), 69
`classifier()` (*pine.client.models.CollectionBuilder* method), 57
`classifier_dict` (in module *pine.backend.pipelines.bp*), 27
`classifier_pipelines` (in module *pine.backend.pipelines.bp*), 27
`collection()` (*pine.client.CollectionBuilder* property), 68
`collection()` (*pine.client.models.CollectionBuilder* property), 56
`collection_builder()` (*pine.client.client.PineClient* method), 42
`collection_builder()` (*pine.client.PineClient* method), 62
`CollectionBuilder` (class in *pine.client*), 66
`CollectionBuilder` (class in *pine.client.models*), 55
`CollectionUserPermissions` (class in *pine.backend.models*), 34
`CollectionUserPermissions` (class in *pine.client.models*), 58
`compute_fl()` (in module *pine.backend.pineiaa.bratiaa.agree*), 21
`compute_fl_agreement()` (in module *pine.backend.pineiaa.bratiaa*), 24
`compute_fl_agreement()` (in module *pine.backend.pineiaa.bratiaa.agree*), 21
`compute_mapping()` (*pine.backend.pineiaa.bratiaa.utils.TokenOverlap* static method), 24
`compute_total_fl_matrix()` (*pine.backend.pineiaa.bratiaa.agree.FIAgreement* method), 21
`compute_total_fl_matrix()` (*pine.backend.pineiaa.bratiaa.FIAgreement* method), 25
`config` (in module *pine.backend.job_manager.service*), 18
`config` (in module *pine.pipelines.app.listener.service_listener*), 71
`config` (in module *pine.pipelines.corenlp_NER_pipeline*), 78
`config` (in module *pine.pipelines.EveClient*), 75
`config` (in module *pine.pipelines.NER_API*), 76
`config` (in module *pine.pipelines.opennlp_NER_pipeline*), 79
`CONFIG_ALLOW_OVERLAPPING_NER_ANNOTATIONS` (in module *pine.backend.annotations.bp*), 3
`CONFIG_AUTH_MODULE_KEY` (in module *pine.backend.auth.bp*), 5
`CONFIG_FILE_ENV` (in module *pine.backend.log*), 32
`CONFIG_FILE_ENV` (in module *pine.client.log*), 48
`ConfigBuilder` (class in *pine.backend.shared.config*), 29
`ConfigBuilder` (class in *pine.pipelines.shared.config*), 74
`configuration()` (*pine.client.CollectionBuilder* method), 69
`configuration()` (*pine.client.models.CollectionBuilder* method), 57
`convert_ann_collection_to_per_token()` (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER* method), 79
`convert_response()` (in module *pine.backend.data.service*), 15
`corenlp_NER` (class in *pine.pipelines.corenlp_NER_pipeline*), 78
`count_documents_in_collection()` (in module *pine.backend.documents.bp*), 17
`counter2list()` (in module *pine.backend.pineiaa.bratiaa.evaluation*), 22
`create_app()` (in module *pine.backend*), 35
`create_app()` (in module *pine.backend.app*), 30
`CREATE_COLLECTION` (*pine.backend.log.Action* attribute), 32
`create_collection()` (in module *pine.backend.collections.bp*), 11
`create_collection()` (*pine.client.client.PineClient* method), 42
`create_collection()` (*pine.client.PineClient* method), 62
`create_iaa_report_by_collection_id()` (in module *pine.backend.pineiaa.bp*), 26
`creator_id()` (*pine.client.CollectionBuilder* method), 68
`creator_id()` (*pine.client.models.CollectionBuilder* method), 56

D

DATASETS_LOCAL_DIR (pine.backend.shared.config.BaseConfig attribute), 29

DEBUG (in module pine.backend.config), 31

DEBUG (pine.backend.shared.config.BaseConfig attribute), 28

DEBUG (pine.pipelines.shared.config.BaseConfig attribute), 73

DEFAULT_DBNAME (pine.client.client.EveClient attribute), 38

delete() (in module pine.backend.data.service), 14

delete_user() (in module pine.backend.admin.bp), 2

description() (pine.client.CollectionBuilder method), 69

description() (pine.client.models.CollectionBuilder method), 57

display_name() (pine.backend.auth.eve.EveUser property), 6

display_name() (pine.backend.auth.oauth.OAuthUser property), 7

display_name() (pine.backend.models.AuthUser property), 34

Document (class in pine.backend.pineiaa.brataia), 25

Document (class in pine.backend.pineiaa.brataia.agree), 21

document_csv_file() (pine.client.CollectionBuilder method), 69

document_csv_file() (pine.client.models.CollectionBuilder method), 58

DOCUMENT_IMAGE_DIR (in module pine.backend.config), 31

documents() (pine.backend.pineiaa.brataia.agree.FIAgreement property), 21

documents() (pine.backend.pineiaa.brataia.FIAgreement property), 25

DOCUMENTS_PER_TRANSACTION (in module pine.backend.collections.bp), 10

download_collection() (in module pine.backend.collections.bp), 10

download_collection_data() (pine.client.client.PineClient method), 45

download_collection_data() (pine.client.PineClient method), 65

download_data (pine.backend.models.CollectionUserPermissions attribute), 34

download_data (pine.client.models.CollectionUserPermissions attribute), 59

draw_heatmap() (pine.backend.pineiaa.brataia.agree.FIAgreement method), 21

draw_heatmap() (pine.backend.pineiaa.brataia.FIAgreement method), 25

E

ENCODING (in module pine.backend.pineiaa.brataia.utils), 24

endpoint_get_user_permissions() (in module pine.backend.collections.bp), 11

endpoint_get_user_permissions() (in module pine.backend.documents.bp), 17

entropy_rank() (in module pine.pipelines.RankingFunctions), 77

evaluate() (pine.pipelines.corenlp_NER_pipeline.corenlp_NER method), 78

evaluate() (pine.pipelines.opennlp_NER_pipeline.opennlp_NER method), 79

evaluate() (pine.pipelines.pmap_ner.NER method), 80

evaluate() (pine.pipelines.spacy_NER_pipeline.spacy_NER method), 81

evaluate_orig() (pine.pipelines.corenlp_NER_pipeline.corenlp_NER method), 78

evaluate_orig() (pine.pipelines.opennlp_NER_pipeline.opennlp_NER method), 79

eve (pine.client.client.LocalPineClient attribute), 46

eve (pine.client.LocalPineClient attribute), 66

EVE_HEADERS (in module pine.backend.pineiaa.brataia.iaa_service), 23

eve_headers (pine.pipelines.EveClient.EveClient attribute), 75

eve_headers (pine.pipelines.NERWrapper.NERWrapper attribute), 76

EVE_HOST (pine.backend.shared.config.BaseConfig attribute), 28

EVE_HOST (pine.pipelines.shared.config.BaseConfig attribute), 73

EVE_PORT (pine.backend.shared.config.BaseConfig attribute), 28

EVE_PORT (pine.pipelines.shared.config.BaseConfig attribute), 73

EVE_SERVER (in module pine.backend.config), 31

EveClient (class in pine.client.client), 38

EveClient (class in pine.pipelines.EveClient), 75

EveModule (class in pine.backend.auth.eve), 6

EveUser (class in pine.backend.auth.eve), 6

exact_match_instance_evaluation() (in module pine.backend.pineiaa.brataia), 25

exact_match_instance_evaluation() (in module pine.backend.pineiaa.brataia.evaluation), 22

exact_match_token_evaluation() (in module pine.backend.pineiaa.brataia), 25

exact_match_token_evaluation() (in module pine.backend.pineiaa.brataia.evaluation), 22

F

FlAgreement (class in *pine.backend.pineiaa.brataiaa*), 25

FlAgreement (class in *pine.backend.pineiaa.agree*), 21

files (*pine.client.CollectionBuilder* attribute), 68

files (*pine.client.models.CollectionBuilder* attribute), 56

find_all() (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER* method), 79

find_all_init() (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER* method), 79

fit() (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER* method), 78

fit() (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER* method), 79

fit() (*pine.pipelines.pipeline.Pipeline* method), 80

fit() (*pine.pipelines.pmap_ner.NER* method), 80

fit() (*pine.pipelines.spacy_NER_pipeline.spacy_NER* method), 81

fix_num_for_json() (in module *pine.backend.pineiaa.brataiaa.iaa_service*), 23

flask_get_can_manage_users() (in module *pine.backend.auth.bp*), 5

flask_get_flat() (in module *pine.backend.auth.bp*), 5

flask_get_logged_in_user() (in module *pine.backend.auth.bp*), 5

flask_get_logged_in_user_details() (in module *pine.backend.auth.bp*), 5

flask_get_login_form() (in module *pine.backend.auth.bp*), 5

flask_get_module() (in module *pine.backend.auth.bp*), 5

flask_post_logout() (in module *pine.backend.auth.bp*), 5

form (*pine.client.CollectionBuilder* attribute), 68

form (*pine.client.models.CollectionBuilder* attribute), 56

form_json() (*pine.client.CollectionBuilder* property), 68

form_json() (*pine.client.models.CollectionBuilder* property), 56

format_data() (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER* method), 78

format_data() (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER* method), 79

format_data() (*pine.pipelines.spacy_NER_pipeline.spacy_NER* method), 81

from_registration_format() (*pine.pipelines.app.listener.service_listener.ServiceRegistration* class method), 71

G

get() (in module *pine.backend.data.service*), 14

get() (*pine.client.client.BaseClient* method), 37

get_all() (in module *pine.backend.data.service*), 15

get_all_documents_in_collection() (in module *pine.backend.documents.bp*), 17

get_all_ids() (*pine.pipelines.EveClient.EveClient* method), 75

get_all_ids() (*pine.pipelines.NERWrapper.NERWrapper* method), 76

get_all_items() (in module *pine.backend.data.service*), 15

get_all_items() (in module *pine.backend.pineiaa.brataiaa.iaa_service*), 23

get_all_items() (*pine.pipelines.EveClient.EveClient* method), 75

get_all_items() (*pine.pipelines.NERWrapper.NERWrapper* method), 76

get_all_users() (in module *pine.backend.data.users*), 16

get_all_users() (*pine.backend.auth.eve.EveModule* method), 6

get_all_versions_of_item_by_id() (in module *pine.backend.data.service*), 15

get_annotations_for_document() (in module *pine.backend.annotations.bp*), 3

get_archived_user_collections() (in module *pine.backend.collections.bp*), 10

get_arg_parser() (*pine.backend.shared.config.ConfigBuilder* class method), 29

get_arg_parser() (*pine.pipelines.shared.config.ConfigBuilder* class method), 74

get_auth_module() (*pine.client.client.PineClient* method), 42

get_auth_module() (*pine.client.PineClient* method), 62

get_classifier_metrics() (in module *pine.backend.pipelines.bp*), 27

get_classifier_pipeline_metrics_objs() (*pine.pipelines.NER_API.ner_api* method), 76

get_collection() (in module *pine.backend.collections.bp*), 10

get_collection_classifier() (in module *pine.backend.pipelines.bp*), 27

get_collection_documents() (*pine.client.client.PineClient* method), 43

get_collection_documents() (*pine.client.PineClient* method), 63

get_collection_ids_for() (in module *pine.backend.documents*), 18

get_collection_ids_for() (in module *pine.backend.documents.bp*), 17

get_collection_image() (in module

pine.backend.collections.bp), 11
 get_collection_image_exists() (in module *pine.backend.collections.bp*), 11
 get_collection_images() (in module *pine.backend.collections.bp*), 11
 get_collection_permissions() (*pine.client.client.PineClient* method), 43
 get_collection_permissions() (*pine.client.PineClient* method), 63
 get_config() (*pine.backend.shared.config.ConfigBuilder* class method), 29
 get_config() (*pine.pipelines.shared.config.ConfigBuilder* class method), 74
 get_config_names() (*pine.backend.shared.config.ConfigBuilder* static method), 29
 get_config_names() (*pine.pipelines.shared.config.ConfigBuilder* static method), 74
 get_current_annotation() (in module *pine.backend.annotations.bp*), 3
 get_current_report() (in module *pine.backend.pineiaa.bp*), 26
 get_details() (*pine.backend.auth.eve.EveUser* method), 6
 get_doc_and_overlap_ids() (in module *pine.backend.collections.bp*), 10
 get_doc_annotations() (in module *pine.backend.pineiaa.bratiaa.iaa_service*), 23
 get_docs_with_annotations() (*pine.pipelines.EveClient.EveClient* method), 75
 get_document() (in module *pine.backend.documents.bp*), 17
 get_document_ranking() (*pine.pipelines.NER_API.ner_api* method), 76
 get_documents() (*pine.pipelines.EveClient.EveClient* method), 75
 get_flask_logged_in_user() (in module *pine.backend.log*), 32
 get_flask_request_info() (in module *pine.backend.log*), 32
 get_id() (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER* method), 78
 get_id() (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER* method), 79
 get_iaa_report_by_collection_id() (in module *pine.backend.pineiaa.bp*), 26
 get_item_by_id() (in module *pine.backend.data.service*), 15
 get_items() (in module *pine.backend.pineiaa.bratiaa.iaa_service*), 23
 get_items() (*pine.pipelines.EveClient.EveClient* method), 75
 get_items() (*pine.pipelines.NERWrapper.NERWrapper* method), 76
 get_logged_in_user() (in module *pine.backend.auth*), 8
 get_logged_in_user() (in module *pine.backend.auth.bp*), 5
 get_logged_in_user() (*pine.backend.auth.bp.AuthModule* method), 5
 get_logged_in_user() (*pine.client.client.PineClient* method), 41
 get_logged_in_user() (*pine.client.PineClient* method), 61
 get_logged_in_user_details() (*pine.backend.auth.bp.AuthModule* method), 5
 get_logged_in_user_details() (*pine.backend.auth.eve.EveModule* method), 6
 get_login_form() (*pine.backend.auth.bp.AuthModule* method), 5
 get_login_form() (*pine.backend.auth.eve.EveModule* method), 6
 get_login_form() (*pine.backend.auth.oauth.OAuthModule* method), 7
 get_login_form_button_text() (*pine.backend.auth.oauth.OAuthModule* method), 7
 get_login_form_button_text() (*pine.backend.auth.vegas.VegasAuthModule* method), 7
 get_metrics() (in module *pine.backend.pipelines.bp*), 27
 get_my_annotations_for_document() (in module *pine.backend.annotations.bp*), 3
 get_my_user_id() (*pine.client.client.PineClient* method), 41
 get_my_user_id() (*pine.client.PineClient* method), 61
 get_next_by_classifier() (in module *pine.backend.pipelines.bp*), 27
 get_obj() (*pine.pipelines.EveClient.EveClient* method), 75
 get_obj() (*pine.pipelines.NERWrapper.NERWrapper* method), 76
 get_others_annotations_for_document() (in module *pine.backend.annotations.bp*), 3
 get_overlap_ids() (in module *pine.backend.collections.bp*), 11
 get_paginated_documents_in_collection() (in module *pine.backend.documents.bp*), 17
 get_pipeline_by_id() (in module *pine.backend.pipelines.bp*), 27
 get_pipelines() (in module

[pine.backend.pipelines.bp](#)), 27
[get_pipelines\(\)](#) ([pine.client.client.PineClient](#)
[method](#)), 42
[get_pipelines\(\)](#) ([pine.client.PineClient](#) [method](#)),
62
[get_registered_channels\(\)](#)
([pine.backend.job_manager.service.ServiceManager](#)
[class method](#)), 19
[get_registered_service_details\(\)](#)
([pine.backend.job_manager.service.ServiceManager](#)
[class method](#)), 19
[get_registered_services\(\)](#)
([pine.backend.job_manager.service.ServiceManager](#)
[class method](#)), 19
[get_resource\(\)](#) ([pine.client.client.EveClient](#)
[method](#)), 39
[get_static_collection_images\(\)](#) ([in module](#)
[pine.backend.collections.bp](#)), 11
[get_unarchived_user_collections\(\)](#) ([in](#)
[module pine.backend.collections.bp](#)), 10
[get_user\(\)](#) ([in module pine.backend.admin.bp](#)), 2
[get_user\(\)](#) ([in module pine.backend.data.users](#)), 16
[get_user_by_email\(\)](#) ([in module](#)
[pine.backend.data.users](#)), 16
[get_user_by_id_or_email\(\)](#) ([in module](#)
[pine.backend.data.users](#)), 16
[get_user_collections\(\)](#) ([in module](#)
[pine.backend.collections.bp](#)), 10
[get_user_details\(\)](#) ([in module](#)
[pine.backend.data.users](#)), 16
[get_user_permissions\(\)](#) ([in module](#)
[pine.backend.collections](#)), 12
[get_user_permissions\(\)](#) ([in module](#)
[pine.backend.collections.bp](#)), 10
[get_user_permissions\(\)](#) ([in module](#)
[pine.backend.documents](#)), 18
[get_user_permissions\(\)](#) ([in module](#)
[pine.backend.documents.bp](#)), 17
[get_user_permissions_by_id\(\)](#) ([in module](#)
[pine.backend.collections](#)), 12
[get_user_permissions_by_id\(\)](#) ([in module](#)
[pine.backend.collections.bp](#)), 10
[get_user_permissions_by_id\(\)](#) ([in module](#)
[pine.backend.documents](#)), 18
[get_user_permissions_by_id\(\)](#) ([in module](#)
[pine.backend.documents.bp](#)), 17
[get_user_permissions_by_ids\(\)](#) ([in module](#)
[pine.backend.collections](#)), 12
[get_user_permissions_by_ids\(\)](#) ([in module](#)
[pine.backend.collections.bp](#)), 10
[get_user_permissions_by_ids\(\)](#) ([in module](#)
[pine.backend.documents](#)), 18
[get_user_permissions_by_ids\(\)](#) ([in module](#)
[pine.backend.documents.bp](#)), 17

[get_users\(\)](#) ([in module pine.backend.admin.bp](#)), 2
[get_users\(\)](#) ([pine.client.client.EveClient](#) [method](#)), 40
[getIAAReportForCollection\(\)](#) ([in module](#)
[pine.backend.pineiaa.brataia.iaa_service](#)), 23

H

[handle_error\(\)](#) ([in module pine.backend.app](#)), 30
[handle_uncaught_exception\(\)](#) ([in module](#)
[pine.backend.app](#)), 30
[handler_timeout\(\)](#) ([pine.backend.job_manager.service.ServiceManager](#)
[attribute](#)), 19
[hash_password\(\)](#) ([in module](#)
[pine.backend.auth.password](#)), 7
[hash_password\(\)](#) ([in module pine.client.password](#)),
60

I

[iaa_report\(\)](#) ([in module](#)
[pine.backend.pineiaa.brataia](#)), 24
[iaa_report\(\)](#) ([in module](#)
[pine.backend.pineiaa.brataia.agree](#)), 21
[id\(\)](#) ([pine.backend.auth.eve.EveUser](#) [property](#)), 6
[id\(\)](#) ([pine.backend.auth.oauth.OAuthUser](#) [property](#)), 6
[id\(\)](#) ([pine.backend.models.AuthUser](#) [property](#)), 34
[ID_FIELD](#) ([in module pine.client.models](#)), 49
[image_file\(\)](#) ([pine.client.CollectionBuilder](#)
[method](#)), 70
[image_file\(\)](#) ([pine.client.models.CollectionBuilder](#)
[method](#)), 58
[init_app\(\)](#) ([in module pine.backend.admin.bp](#)), 2
[init_app\(\)](#) ([in module pine.backend.annotations.bp](#)),
4
[init_app\(\)](#) ([in module pine.backend.auth.bp](#)), 5
[init_app\(\)](#) ([in module pine.backend.collections.bp](#)),
11
[init_app\(\)](#) ([in module pine.backend.cors](#)), 31
[init_app\(\)](#) ([in module pine.backend.data.bp](#)), 12
[init_app\(\)](#) ([in module pine.backend.documents.bp](#)),
17
[init_app\(\)](#) ([in module pine.backend.pineiaa.bp](#)), 26
[init_app\(\)](#) ([in module pine.backend.pipelines.bp](#)), 27
[init_config\(\)](#) ([pine.backend.shared.config.ConfigBuilder](#)
[class method](#)), 29
[init_config\(\)](#) ([pine.pipelines.shared.config.ConfigBuilder](#)
[class method](#)), 74
[input_generator\(\)](#) ([in module](#)
[pine.backend.pineiaa.brataia](#)), 25
[input_generator\(\)](#) ([in module](#)
[pine.backend.pineiaa.brataia.agree](#)), 21
[is_admin\(\)](#) ([pine.backend.auth.eve.EveUser](#) [prop-](#)
[erty](#)), 6
[is_admin\(\)](#) ([pine.backend.auth.oauth.OAuthUser](#)
[property](#)), 7

`is_admin()` (*pine.backend.models.AuthUser* property), 34
`is_cached_last_collection()` (in module *pine.backend.collections.bp*), 10
`is_flat()` (in module *pine.backend.auth*), 8
`is_flat()` (in module *pine.backend.auth.bp*), 5
`is_flat()` (*pine.backend.auth.bp.AuthModule* method), 5
`is_flat()` (*pine.backend.auth.eve.EveModule* method), 6
`is_flat()` (*pine.backend.auth.oauth.OAuthModule* method), 7
`is_logged_in()` (*pine.client.client.PineClient* method), 41
`is_logged_in()` (*pine.client.PineClient* method), 61
`is_valid()` (*pine.client.client.BaseClient* method), 36
`is_valid()` (*pine.client.client.EveClient* method), 38
`is_valid()` (*pine.client.client.LocalPineClient* method), 46
`is_valid()` (*pine.client.client.PineClient* method), 41
`is_valid()` (*pine.client.CollectionBuilder* method), 70
`is_valid()` (*pine.client.LocalPineClient* method), 66
`is_valid()` (*pine.client.models.CollectionBuilder* method), 58
`is_valid()` (*pine.client.PineClient* method), 61
`is_valid_annotation()` (in module *pine.client.models*), 54
`is_valid_collection()` (in module *pine.client.models*), 53
`is_valid_doc_annotation()` (in module *pine.client.models*), 53
`is_valid_doc_annotations()` (in module *pine.client.models*), 54
`is_valid_eve_collection()` (in module *pine.client.models*), 52
`is_valid_eve_document()` (in module *pine.client.models*), 53
`is_valid_eve_pipeline()` (in module *pine.client.models*), 52
`is_valid_eve_user()` (in module *pine.client.models*), 52
`is_valid_ner_annotation()` (in module *pine.client.models*), 54
`ITEMS_FIELD` (in module *pine.client.models*), 49

L

`label()` (*pine.client.CollectionBuilder* method), 68
`label()` (*pine.client.models.CollectionBuilder* method), 56
`labels()` (*pine.backend.pineiaa.brataiaa.agree.FIAgreement* property), 21
`labels()` (*pine.backend.pineiaa.brataiaa.FIAgreement* property), 25
`largest_margin()` (in module *pine.pipelines.RankingFunctions*), 77
`LAST_COLLECTION_FOR_IMAGE` (in module *pine.backend.collections.bp*), 10
`least_confidence()` (in module *pine.pipelines.RankingFunctions*), 77
`least_confidence_squared()` (in module *pine.pipelines.RankingFunctions*), 77
`least_confidence_squared_by_entity()` (in module *pine.pipelines.RankingFunctions*), 77
`list_collections()` (*pine.client.client.PineClient* method), 45
`list_collections()` (*pine.client.PineClient* method), 65
`listener_poll` (*pine.pipelines.app.listener.service_listener.ServiceListener* attribute), 72
`load_classifier()` (*pine.pipelines.NERWrapper.NERWrapper* method), 76
`load_model()` (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER* method), 78
`load_model()` (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER* method), 79
`load_model()` (*pine.pipelines.pipeline.Pipeline* method), 80
`load_model()` (*pine.pipelines.pmap_ner.NER* method), 80
`load_model()` (*pine.pipelines.spacy_NER_pipeline.spacy_NER* method), 81
`LocalPineClient` (class in *pine.client*), 66
`LocalPineClient` (class in *pine.client.client*), 46
`logger` (in module *pine.backend.annotations.bp*), 3
`LOGGER` (in module *pine.backend.app*), 30
`LOGGER` (in module *pine.backend.collections.bp*), 10
`logger` (in module *pine.backend.data.service*), 13
`logger` (in module *pine.backend.job_manager.service*), 18
`logger` (in module *pine.backend.pineiaa.bp*), 26
`LOGGER` (in module *pine.backend.pineiaa.brataiaa.agree*), 20
`logger` (in module *pine.backend.pipelines.bp*), 27
`LOGGER` (in module *pine.backend.shared.config*), 28
`logger` (in module *pine.pipelines.app.listener.service_listener*), 71
`logger` (in module *pine.pipelines.corenlp_NER_pipeline*), 78
`logger` (in module *pine.pipelines.EveClient*), 75
`logger` (in module *pine.pipelines.NER_API*), 76
`logger` (in module *pine.pipelines.opennlp_NER_pipeline*), 79
`logger` (in module *pine.pipelines.pmap_ner*), 80
`LOGGER` (in module *pine.pipelines.shared.config*), 73
`LOGGER_DIR` (*pine.backend.shared.config.BaseConfig* attribute), 28

LOGGER_DIR (*pine.pipelines.shared.config.BaseConfig attribute*), 73
 LOGGER_FILE (*pine.backend.shared.config.BaseConfig attribute*), 28
 LOGGER_FILE (*pine.pipelines.shared.config.BaseConfig attribute*), 73
 LOGGER_LEVEL (*pine.backend.shared.config.BaseConfig attribute*), 28
 LOGGER_LEVEL (*pine.pipelines.shared.config.BaseConfig attribute*), 73
 LOGGER_NAME (*pine.backend.shared.config.BaseConfig attribute*), 28
 LOGGER_NAME (*pine.pipelines.shared.config.BaseConfig attribute*), 73
 LOGIN (*pine.backend.log.Action attribute*), 32
 login() (*pine.backend.auth.eve.EveModule method*), 6
 login() (*pine.backend.auth.oauth.OAuthModule method*), 7
 login_eve() (*pine.client.client.PineClient method*), 42
 login_eve() (*pine.client.PineClient method*), 62
 login_required() (*in module pine.backend.auth*), 8
 login_required() (*in module pine.backend.auth.bp*), 5
 LoginForm (*class in pine.backend.models*), 33
 LoginFormField (*class in pine.backend.models*), 33
 LoginFormFieldType (*class in pine.backend.models*), 33
 LOGOUT (*pine.backend.log.Action attribute*), 32
 logout() (*pine.backend.auth.bp.AuthModule method*), 5
 logout() (*pine.client.client.PineClient method*), 42
 logout() (*pine.client.PineClient method*), 62
M
 main() (*in module pine.backend.pineiaa.brataia.agree_cli*), 22
 mean_sd_per_document() (*pine.backend.pineiaa.brataia.agree.FIAgreement method*), 21
 mean_sd_per_document() (*pine.backend.pineiaa.brataia.FIAgreement method*), 25
 mean_sd_per_label() (*pine.backend.pineiaa.brataia.agree.FIAgreement method*), 21
 mean_sd_per_label() (*pine.backend.pineiaa.brataia.FIAgreement method*), 25
 mean_sd_per_label_one_vs_rest() (*pine.backend.pineiaa.brataia.agree.FIAgreement method*), 21
 mean_sd_per_label_one_vs_rest() (*pine.backend.pineiaa.brataia.FIAgreement method*), 25
 mean_sd_total() (*pine.backend.pineiaa.brataia.agree.FIAgreement method*), 21
 mean_sd_total() (*pine.backend.pineiaa.brataia.FIAgreement method*), 25
 mean_sd_total_one_vs_rest() (*pine.backend.pineiaa.brataia.agree.FIAgreement method*), 21
 mean_sd_total_one_vs_rest() (*pine.backend.pineiaa.brataia.FIAgreement method*), 25
 message (*pine.client.exceptions.PineClientException attribute*), 47
 metadata() (*pine.client.CollectionBuilder method*), 68
 metadata() (*pine.client.models.CollectionBuilder method*), 57
 MODELS_DIR (*pine.pipelines.shared.config.BaseConfig attribute*), 73
 MODELS_LOCAL_DIR (*pine.backend.shared.config.BaseConfig attribute*), 29
 modify_document_metadata (*pine.backend.models.CollectionUserPermissions attribute*), 34
 modify_document_metadata (*pine.client.models.CollectionUserPermissions attribute*), 59
 modify_labels (*pine.backend.models.CollectionUserPermissions attribute*), 34
 modify_labels (*pine.client.models.CollectionUserPermissions attribute*), 59
 modify_users (*pine.backend.models.CollectionUserPermissions attribute*), 34
 modify_users (*pine.client.models.CollectionUserPermissions attribute*), 59
 module
 pine, 1
 pine.backend, 1
 pine.backend.admin, 1
 pine.backend.admin.bp, 1
 pine.backend.annotations, 2
 pine.backend.annotations.bp, 2
 pine.backend.app, 30
 pine.backend.auth, 4
 pine.backend.auth.bp, 4
 pine.backend.auth.eve, 6
 pine.backend.auth.oauth, 6
 pine.backend.auth.password, 7
 pine.backend.auth.vegas, 7
 pine.backend.collections, 8
 pine.backend.collections.bp, 8
 pine.backend.config, 31
 pine.backend.cors, 31
 pine.backend.data, 12

pine.backend.data.bp, 12
 pine.backend.data.service, 12
 pine.backend.data.users, 15
 pine.backend.documents, 16
 pine.backend.documents.bp, 16
 pine.backend.job_manager, 18
 pine.backend.job_manager.service, 18
 pine.backend.log, 31
 pine.backend.models, 33
 pine.backend.pineiaa, 20
 pine.backend.pineiaa.bp, 26
 pine.backend.pineiaa.bratiaa, 20
 pine.backend.pineiaa.bratiaa.agree, 20
 pine.backend.pineiaa.bratiaa.agree_cli, 22
 pine.backend.pineiaa.bratiaa.evaluation, 22
 pine.backend.pineiaa.bratiaa.iaa_service, 23
 pine.backend.pineiaa.bratiaa.utils, 23
 pine.backend.pipelines, 26
 pine.backend.pipelines.bp, 26
 pine.backend.shared, 28
 pine.backend.shared.config, 28
 pine.backend.shared.transform, 30
 pine.client, 35
 pine.client.client, 35
 pine.client.exceptions, 47
 pine.client.log, 48
 pine.client.models, 48
 pine.client.password, 60
 pine.pipelines, 70
 pine.pipelines.app, 70
 pine.pipelines.app.listener, 70
 pine.pipelines.app.listener.main, 70
 pine.pipelines.app.listener.service_listener, 71
 pine.pipelines.corenlp_NER_pipeline, 78
 pine.pipelines.EveClient, 75
 pine.pipelines.NER_API, 76
 pine.pipelines.NERWrapper, 76
 pine.pipelines.opennlp_NER_pipeline, 79
 pine.pipelines.pipeline, 80
 pine.pipelines.pmap_ner, 80
 pine.pipelines.RankingFunctions, 77
 pine.pipelines.run_service, 81
 pine.pipelines.shared, 72
 pine.pipelines.shared.config, 72
 pine.pipelines.shared.transform, 75

pine.pipelines.spacy_NER_pipeline, 81
 module (in module pine.backend.auth), 8
 module (in module pine.backend.auth.bp), 5
 mongo (pine.client.client.EveClient attribute), 38
 mongo_base_uri (pine.client.client.EveClient attribute), 38
 mongo_db (pine.client.client.EveClient attribute), 38
 most_of_least_popular() (in module pine.pipelines.RankingFunctions), 77

N

NER (class in pine.pipelines.pmap_ner), 80
 ner_api (class in pine.pipelines.NER_API), 76
 NERWrapper (class in pine.pipelines.NERWrapper), 76
 next_example() (pine.pipelines.corenlp_NER_pipeline.corenlp_NER method), 78
 next_example() (pine.pipelines.opennlp_NER_pipeline.opennlp_NER method), 79
 next_example() (pine.pipelines.pipeline.Pipeline method), 80
 next_example() (pine.pipelines.pmap_ner.NER method), 80
 next_example() (pine.pipelines.spacy_NER_pipeline.spacy_NER method), 81
 not_found() (in module pine.backend.cors), 31

O

OAuthModule (class in pine.backend.auth.oauth), 7
 OAuthUser (class in pine.backend.auth.oauth), 6
 opennlp_NER (class in pine.pipelines.opennlp_NER_pipeline), 79
 overlapping_tokens() (pine.backend.pineiaa.bratiaa.utils.TokenOverlap method), 24

P

params() (in module pine.backend.data.service), 14
 parse_args() (in module pine.backend.pineiaa.bratiaa.agree_cli), 22
 parser (in module pine.pipelines.NERWrapper), 76
 PASSWORD (pine.backend.models.LoginFormFieldType attribute), 33
 patch() (in module pine.backend.data.service), 14
 patch() (pine.client.client.BaseClient method), 37
 perform_five_fold() (pine.pipelines.NER_API.ner_api method), 76
 perform_fold() (pine.pipelines.NER_API.ner_api method), 76
 PERFORMANCE_HISTORY (in module pine.backend.data.service), 13

PerformanceHistory	(class	in	pine.backend.log
<i>pine.backend.data.service</i>), 13			module, 31
pformat()	(<i>pine.backend.data.service.PerformanceHistory</i>		pine.backend.models
<i>method</i>), 13			module, 33
pine			pine.backend.pineiaa
module, 1			module, 20
pine.backend			pine.backend.pineiaa.bp
module, 1			module, 26
pine.backend.admin			pine.backend.pineiaa.bratiaa
module, 1			module, 20
pine.backend.admin.bp			pine.backend.pineiaa.bratiaa.agree
module, 1			module, 20
pine.backend.annotations			pine.backend.pineiaa.bratiaa.agree_cli
module, 2			module, 22
pine.backend.annotations.bp			pine.backend.pineiaa.bratiaa.evaluation
module, 2			module, 22
pine.backend.app			pine.backend.pineiaa.bratiaa.iaa_service
module, 30			module, 23
pine.backend.auth			pine.backend.pineiaa.bratiaa.utils
module, 4			module, 23
pine.backend.auth.bp			pine.backend.pipelines
module, 4			module, 26
pine.backend.auth.eve			pine.backend.pipelines.bp
module, 6			module, 26
pine.backend.auth.oauth			pine.backend.shared
module, 6			module, 28
pine.backend.auth.password			pine.backend.shared.config
module, 7			module, 28
pine.backend.auth.vegas			pine.backend.shared.transform
module, 7			module, 30
pine.backend.collections			pine.client
module, 8			module, 35
pine.backend.collections.bp			pine.client.client
module, 8			module, 35
pine.backend.config			pine.client.exceptions
module, 31			module, 47
pine.backend.cors			pine.client.log
module, 31			module, 48
pine.backend.data			pine.client.models
module, 12			module, 48
pine.backend.data.bp			pine.client.password
module, 12			module, 60
pine.backend.data.service			pine.pipelines
module, 12			module, 70
pine.backend.data.users			pine.pipelines.app
module, 15			module, 70
pine.backend.documents			pine.pipelines.app.listener
module, 16			module, 70
pine.backend.documents.bp			pine.pipelines.app.listener.main
module, 16			module, 70
pine.backend.job_manager			pine.pipelines.app.listener.service_listener
module, 18			module, 71
pine.backend.job_manager.service			pine.pipelines.corenlp_NER_pipeline
module, 18			module, 78

pine.pipelines.EveClient
 module, 75
 pine.pipelines.NER_API
 module, 76
 pine.pipelines.NERWrapper
 module, 76
 pine.pipelines.opennlp_NER_pipeline
 module, 79
 pine.pipelines.pipeline
 module, 80
 pine.pipelines.pmap_ner
 module, 80
 pine.pipelines.RankingFunctions
 module, 77
 pine.pipelines.run_service
 module, 81
 pine.pipelines.shared
 module, 72
 pine.pipelines.shared.config
 module, 72
 pine.pipelines.shared.transform
 module, 75
 pine.pipelines.spacy_NER_pipeline
 module, 81
 PineClient (class in pine.client), 60
 PineClient (class in pine.client.client), 40
 PineClientAuthException, 47
 PineClientException, 47
 PineClientHttpException, 47
 PineClientValueException, 47
 ping () (pine.client.client.EveClient method), 39
 ping () (pine.client.client.PineClient method), 41
 ping () (pine.client.PineClient method), 61
 pipe_init () (pine.pipelines.pmap_ner.NER method), 80
 Pipeline (class in pine.pipelines.pipeline), 80
 pipeline (pine.pipelines.pmap_ner.NER attribute), 80
 PIPELINE (pine.pipelines.shared.config.BaseConfig attribute), 73
 post () (in module pine.backend.data.service), 14
 post () (pine.client.client.BaseClient method), 38
 post_collection_image () (in module pine.backend.collections.bp), 11
 pprint () (pine.backend.data.service.PerformanceHistory method), 13
 pre_process_message ()
 (pine.pipelines.app.listener.service_listener.ServiceListener method), 72
 predict () (in module pine.backend.pipelines.bp), 27
 predict () (pine.pipelines.corenlp_NER_pipeline.corenlp_NER method), 78
 predict () (pine.pipelines.NER_API.ner_api method), 76
 predict () (pine.pipelines.NERWrapper.NERWrapper method), 76
 predict () (pine.pipelines.opennlp_NER_pipeline.opennlp_NER method), 79
 predict () (pine.pipelines.pipeline.Pipeline method), 80
 predict () (pine.pipelines.pmap_ner.NER method), 80
 predict () (pine.pipelines.spacy_NER_pipeline.spacy_NER method), 81
 predict_proba () (pine.pipelines.corenlp_NER_pipeline.corenlp_NER method), 78
 predict_proba () (pine.pipelines.opennlp_NER_pipeline.opennlp_NER method), 79
 predict_proba () (pine.pipelines.pipeline.Pipeline method), 80
 predict_proba () (pine.pipelines.pmap_ner.NER method), 80
 predict_proba () (pine.pipelines.spacy_NER_pipeline.spacy_NER method), 81
 preprocessing_lock_key
 (pine.pipelines.app.listener.service_listener.ServiceListener attribute), 72
 preprocessing_lock_key_timeout
 (pine.pipelines.app.listener.service_listener.ServiceListener attribute), 72
 preprocessing_worker_lock_key
 (pine.pipelines.app.listener.service_listener.ServiceListener attribute), 72
 preprocessing_worker_lock_key_timeout
 (pine.pipelines.app.listener.service_listener.ServiceListener attribute), 72
 print_table () (pine.backend.pineiaa.brataiaa.agree.FIAgreement static method), 21
 print_table () (pine.backend.pineiaa.brataiaa.FIAgreement static method), 25
 print_users_command () (in module pine.backend.data.users), 16
 process_message ()
 (pine.pipelines.app.listener.service_listener.ServiceListener static method), 72
 processing_limit (pine.pipelines.app.listener.service_listener.ServiceListener attribute), 72
 processing_lock_key
 (pine.pipelines.app.listener.service_listener.ServiceListener attribute), 72
 processing_lock_key_timeout
 (pine.pipelines.app.listener.service_listener.ServiceListener attribute), 72
 processing_queue_key
 (pine.pipelines.app.listener.service_listener.ServiceListener attribute), 72
 processing_queue_key_timeout
 (pine.pipelines.app.listener.service_listener.ServiceListener attribute), 72

processing_worker_name (pine.backend.job_manager.service.ServiceManager attribute), 19
 processor_poll (pine.pipelines.app.listener.service_listener.ServiceListener attribute), 72
 publish_response() (pine.pipelines.app.listener.service_listener.ServiceListener class method), 72
 put() (in module pine.backend.data.service), 14
 put() (pine.client.client.BaseClient method), 37
R
 r_conn (pine.backend.job_manager.service.ServiceManager attribute), 18
 r_conn (pine.pipelines.app.listener.service_listener.ServiceListener attribute), 71
 r_pool (pine.backend.job_manager.service.ServiceManager attribute), 18
 r_pool (pine.pipelines.app.listener.service_listener.ServiceListener attribute), 71
 random_rank() (in module pine.pipelines.RankingFunctions), 77
 rank() (in module pine.pipelines.RankingFunctions), 77
 read() (in module pine.backend.pineiaa.bratiaa.utils), 24
 redis_channel_ttl_key_prefix (pine.backend.job_manager.service.ServiceManager attribute), 19
 redis_channels_key (pine.backend.job_manager.service.ServiceManager attribute), 18
 redis_channels_key_ttl (pine.backend.job_manager.service.ServiceManager attribute), 19
 REDIS_DBNUM (pine.backend.shared.config.BaseConfig attribute), 28
 REDIS_DBNUM (pine.pipelines.shared.config.BaseConfig attribute), 73
 REDIS_EXPIRE (pine.backend.shared.config.BaseConfig attribute), 28
 REDIS_EXPIRE (pine.pipelines.shared.config.BaseConfig attribute), 73
 redis_handler_mutex_key_prefix (pine.backend.job_manager.service.ServiceManager attribute), 19
 redis_handler_mutex_key_ttl (pine.backend.job_manager.service.ServiceManager attribute), 19
 REDIS_HOST (pine.backend.shared.config.BaseConfig attribute), 28
 REDIS_HOST (pine.pipelines.shared.config.BaseConfig attribute), 73
 redis_key_prefix (pine.backend.job_manager.service.ServiceManager attribute), 18
 REDIS_PORT (in module pine.backend.config), 31
 REDIS_PORT (pine.backend.shared.config.BaseConfig attribute), 28
 REDIS_PORT (pine.pipelines.shared.config.BaseConfig attribute), 73
 REDIS_PREFIX (pine.backend.shared.config.BaseConfig attribute), 28
 REDIS_PREFIX (pine.pipelines.shared.config.BaseConfig attribute), 73
 REDIS_PWD (pine.backend.shared.config.BaseConfig attribute), 28
 REDIS_PWD (pine.pipelines.shared.config.BaseConfig attribute), 73
 redis_reg_key_prefix (pine.backend.job_manager.service.ServiceManager attribute), 18
 redis_reg_key_ttl (pine.backend.job_manager.service.ServiceManager attribute), 19
 REDIS_SERVER (in module pine.backend.config), 31
 REDIS_USR (pine.backend.shared.config.BaseConfig attribute), 28
 REDIS_USR (pine.pipelines.shared.config.BaseConfig attribute), 73
 redis_work_mutex_key_prefix (pine.backend.job_manager.service.ServiceManager attribute), 19
 redis_work_mutex_key_ttl (pine.backend.job_manager.service.ServiceManager attribute), 19
 redis_work_queue_key_prefix (pine.backend.job_manager.service.ServiceManager attribute), 19
 redis_work_queue_key_ttl (pine.backend.job_manager.service.ServiceManager attribute), 19
 register_oauth() (pine.backend.auth.oauth.OAuthModule method), 7
 register_oauth() (pine.backend.auth.vegas.VegasAuthModule method), 7
 registration_channel (pine.backend.job_manager.service.ServiceManager attribute), 19
 registration_channel (pine.pipelines.app.listener.service_listener.ServiceListener attribute), 72
 registration_poll (pine.pipelines.app.listener.service_listener.ServiceListener attribute), 71
 registration_worker_name (pine.backend.job_manager.service.ServiceManager attribute), 19

`remove_eve_fields()` (in module `pine.backend.data.service`), 15
`remove_eve_fields()` (in module `pine.client.models`), 59
`remove_nonupdatable_fields()` (in module `pine.backend.data.service`), 15
`remove_nonupdatable_fields()` (in module `pine.client.models`), 59
`reserved_channels` (`pine.backend.job_manager.service.ServiceManager` attribute), 19
`reset_user_passwords()` (in module `pine.backend.data.users`), 16
`resp` (`pine.client.exceptions.PineClientHttpException` attribute), 47
`ROOT_DIR` (`pine.backend.shared.config.BaseConfig` attribute), 28
`ROOT_DIR` (`pine.pipelines.shared.config.BaseConfig` attribute), 73
`run()` (in module `pine.pipelines.app.listener.main`), 71

S

`save_annotations()` (in module `pine.backend.annotations.bp`), 4
`save_collection_annotations()` (in module `pine.backend.annotations.bp`), 4
`save_model()` (`pine.pipelines.corenlp_NER_pipeline.corenlp_NER` method), 78
`save_model()` (`pine.pipelines.opennlp_NER_pipeline.opennlp_NER` method), 79
`save_model()` (`pine.pipelines.pipeline.Pipeline` method), 80
`save_model()` (`pine.pipelines.pmap_ner.NER` method), 80
`save_model()` (`pine.pipelines.spacy_NER_pipeline.spacy_NER` method), 81
`SCHEDULER_HANDLER_TIMEOUT` (`pine.backend.shared.config.BaseConfig` attribute), 28
`SCHEDULER_HANDLER_TIMEOUT` (`pine.pipelines.shared.config.BaseConfig` attribute), 73
`SCHEDULER_QUEUE_TIMEOUT` (`pine.backend.shared.config.BaseConfig` attribute), 29
`SCHEDULER_QUEUE_TIMEOUT` (`pine.pipelines.shared.config.BaseConfig` attribute), 73
`SCHEDULER_REGISTRATION_TIMEOUT` (`pine.backend.shared.config.BaseConfig` attribute), 28
`SCHEDULER_REGISTRATION_TIMEOUT` (`pine.pipelines.shared.config.BaseConfig` attribute), 73
`SECRET_KEY` (in module `pine.backend.config`), 31
`send_service_request()` (`pine.backend.job_manager.service.ServiceManager` class method), 19
`SERVICE_HANDLER_TIMEOUT` (`pine.backend.shared.config.BaseConfig` attribute), 29
`SERVICE_HANDLER_TIMEOUT` (`pine.pipelines.shared.config.BaseConfig` attribute), 73
`SERVICE_LIST` (`pine.backend.shared.config.BaseConfig` attribute), 29
`SERVICE_LIST` (`pine.pipelines.shared.config.BaseConfig` attribute), 73
`SERVICE_LISTENING_FREQUENCY` (`pine.backend.shared.config.BaseConfig` attribute), 29
`SERVICE_LISTENING_FREQUENCY` (`pine.pipelines.shared.config.BaseConfig` attribute), 73
`service_manager` (in module `pine.backend.pipelines.bp`), 27
`SERVICE_REGISTRATION_CHANNEL` (`pine.backend.shared.config.BaseConfig` attribute), 29
`SERVICE_REGISTRATION_CHANNEL` (`pine.pipelines.shared.config.BaseConfig` attribute), 73
`SERVICE_REGISTRATION_FREQUENCY` (`pine.backend.shared.config.BaseConfig` attribute), 29
`SERVICE_REGISTRATION_FREQUENCY` (`pine.pipelines.shared.config.BaseConfig` attribute), 73
`ServiceListener` (class in `pine.pipelines.app.listener.service_listener`), 71
`ServiceManager` (class in `pine.backend.job_manager.service`), 18
`ServiceRegistration` (class in `pine.pipelines.app.listener.service_listener`), 71
`session` (`pine.client.client.BaseClient` attribute), 36
`set_config()` (`pine.backend.shared.config.ConfigBuilder` class method), 30
`set_config()` (`pine.pipelines.shared.config.ConfigBuilder` class method), 74
`set_document_to_annotated_by_user()` (in module `pine.backend.annotations.bp`), 4
`set_user_password()` (in module `pine.backend.data.users`), 16
`set_user_password_by_id()` (in module `pine.backend.data.users`), 16
`setup_logging()` (in module `pine.backend.log`), 32
`setup_logging()` (in module `pine.client`), 66
`setup_logging()` (in module `pine.client.log`), 48

setup_logging() (in module *pine.pipelines.app.listener.main*), 71
 shutdown_channel (pine.backend.job_manager.service.ServiceManager (in module *pine.backend.pineiaa.bratiaa.utils*), 24
 attribute), 19
 sm (in module *pine.backend.job_manager.service*), 20
 tokenize() (in module *pine.backend.pineiaa.bratiaa*), 26
 spacy_NER (class in *pine.pipelines.spacy_NER_pipeline*), 81
 tokenize() (pine.pipelines.corenlp_NER_pipeline.corenlp_NER method), 78
 TokenOverlap (class in *pine.backend.pineiaa.bratiaa.utils*), 24
 start_listeners() (pine.backend.job_manager.service.ServiceManager.main_model() (pine.pipelines.NER_API.ner_api method), 19
 method), 19
 start_workers() (pine.pipelines.app.listener.service_listener.ServiceListener.configure_by_config() (in module *pine.backend.shared.transform*), 30
 method), 72
 status_code() (pine.client.exceptions.PineClientHttpException.form_module_by_config() (in module *pine.pipelines.shared.transform*), 75
 property), 47
 stop_listeners() (pine.backend.job_manager.service.ServiceManager method), 19
 stop_workers() (pine.pipelines.app.listener.service_listener.ServiceListener.handle_service_exception() (in module *pine.backend.collections.bp*), 10
 method), 72
 system_export() (in module *pine.backend.admin.bp*), 2
 update() (pine.pipelines.EveClient.EveClient method), 75
 system_import() (in module *pine.backend.admin.bp*), 2
 update() (pine.pipelines.NERWrapper.NERWrapper method), 76
 update_cached_last_collection() (in module *pine.backend.collections.bp*), 10
 update_metadata() (in module *pine.backend.documents.bp*), 17
 update_model() (pine.pipelines.NERWrapper.NERWrapper method), 76
 update_user() (in module *pine.backend.admin.bp*), 2
 update_user() (in module *pine.backend.data.users*), 16
 update_user_details() (pine.backend.auth.eve.EveModule method), 6
 update_user_password() (in module *pine.backend.admin.bp*), 2
 update_user_password() (pine.backend.auth.eve.EveModule method), 6
 uri() (pine.client.client.BaseClient method), 36
 url() (in module *pine.backend.data.service*), 13
 user_permissions_projection() (in module *pine.backend.collections*), 12
 user_permissions_projection() (in module *pine.backend.collections.bp*), 10
 UserDetails (class in *pine.backend.models*), 34
 username() (pine.backend.auth.eve.EveUser property), 6
 username() (pine.backend.auth.oauth.OAuthUser property), 7
 username() (pine.backend.models.AuthUser property), 34
 VEGAS_CLIENT_SECRET (in module *pine.backend.config*), 31
 test_redis() (in module *pine.backend.pipelines.bp*), 27
 TestConfig (class in *pine.backend.shared.config*), 29
 TestConfig (class in *pine.pipelines.shared.config*), 74
 TESTING (pine.backend.shared.config.BaseConfig attribute), 28
 TESTING (pine.pipelines.shared.config.BaseConfig attribute), 73
 TEXT (pine.backend.models.LoginFormFieldType attribute), 33
 title() (pine.client.CollectionBuilder method), 69
 title() (pine.client.models.CollectionBuilder method), 57
 to_dict() (pine.backend.models.AuthUser method), 34
 to_dict() (pine.backend.models.CollectionUserPermissions method), 34
 to_dict() (pine.backend.models.LoginForm method), 33
 to_dict() (pine.backend.models.LoginFormField method), 33
 to_dict() (pine.backend.models.UserDetails method), 34
 to_dict() (pine.client.models.CollectionUserPermissions method), 59
 to_registration_format() (pine.pipelines.app.listener.service_listener.ServiceRegistration method), 71
 TOKEN (in module *pine.backend.pineiaa.bratiaa.utils*), 24

VegasAuthModule (class in
 pine.backend.auth.vegas), 7
verify_ssl (*pine.client.client.BaseClient* attribute),
 36
VERSION (in module *pine.backend*), 35
VERSION (in module *pine.backend.app*), 30
view (*pine.backend.models.CollectionUserPermissions*
 attribute), 34
view (*pine.client.models.CollectionUserPermissions* at-
 tribute), 58
VIEW_DOCUMENT (*pine.backend.log.Action* attribute),
 32
viewer() (*pine.client.CollectionBuilder* method), 68
viewer() (*pine.client.models.CollectionBuilder*
 method), 56

W

where_params() (in module
 pine.backend.data.service), 13