SuperAnnotate Release July 3, 2022

In [1]:
```
from superannotate import SAClient
sa = SAClient(config_path = "~/.superannotate/dev_config.json")
sa.get_team_metadata()['name']
```

```
SA-PYTHON-SDK - INFO - Development version 4.3.5dev23 of SuperAnnotate SDK i
s being used.
SA-PYTHON-SDK - WARNING - There is a newer version of SuperAnnotate Python S
DK available on PyPI. Run 'pip install --upgrade superannotate' to upgrade f
rom your version 4.3.5.dev23 to 4.4.0
```

Out[1]:
```
'Team on The Dev Branch'
```

Subsets in SDK

`get_subsets()` to get existing Subsets

`SAClient.get_subsets(project)`

Get Subsets

Parameters:

- **project** *(str)* – project name (e.g., “project1”)

Returns: subsets’ metadata

Return type: list of dicts

In [2]:
```
# to get all existing Subsets
sa.get_subsets(
    project = "Image Project"
)
```

Out[2]:
```
[{'name': 'bad annotations'}, {'name': 'polylines'}]
```

`query()` to support querying in Subsets

`SAClient.query(project, query=None, subset=None)`

Return items that satisfy the given query. Query syntax should be in SuperAnnotate query language

Parameters:

- **project** *(str)* – project name or folder path (e.g., “project1/folder1”)
- **query** *(str)* - SAQuL query string.
- **subset** *(str)* - subset name. Allows you to query items in a specific subset. To return all the items in the specified subset, set the value of query param to None

Returns: subsets’ metadata

Return type: list of dicts

In [3]:
```
# to get all items in a Subset
sa.query(
    project="Image Project",
    query=None,
    subset="bad annotations"
)
```

Out[3]:
```
[{'createdAt': '2022-04-03T13:38:58.000Z',
  'updatedAt': '2022-04-28T15:14:01.000Z',
  'name': 'AliS copy 2.jpeg',
  'path': 'Image Project/Subfolder 1',
  'url': None,
  'annotator_email': 'issuereplicator@gmail.com',
  'qa_email': None,
  'annotation_status': 'InProgress',
  'entropy_value': 8140.11,
  'prediction_status': 'NotStarted',
  'segmentation_status': None,
  'approval_status': 'disapproved',
  'is_pinned': False},
 {'createdAt': '2022-04-03T13:38:59.000Z',
  'updatedAt': '2022-05-06T14:53:24.000Z',
  'name': 'AliS copy 12.jpeg',
  'path': 'Image Project/Subfolder 1',
  'url': None,
  'annotator_email': 'superannotator2@gmail.com',
  'qa_email': None,
  'annotation_status': 'Completed',
  'entropy_value': 8691.52,
  'prediction_status': 'NotStarted',
  'segmentation_status': None,
  'approval_status': None,
  'is_pinned': False}]
```

In [4]:
```
# to run a query in a specific Subset
sa.query(
    project="Image Project",
    query="metadata(annotatorEmail = superannotator2@gmail.com)",
    subset="bad annotations"
)
```

Out[4]:

```
[{'createdAt': '2022-04-03T13:38:59.000Z',
  'updatedAt': '2022-05-06T14:53:24.000Z',
  'name': 'AliS copy 12.jpeg',
  'path': 'Image Project/Subfolder 1',
  'url': None,
  'annotator_email': 'superannotator2@gmail.com',
  'qa_email': None,
  'annotation_status': 'Completed',
  'entropy_value': 8691.52,
  'prediction_status': 'NotStarted',
  'segmentation_status': None,
  'approval_status': None,
  'is_pinned': False}]
```

Generic Functions Vol.3

`download_annotations()` to download annotations without preparing export

`SAClient.download_annotations(project, path=None, items=None, recursive=False, callback=None)`

Downloads annotation JSON files of the selected items to the local directory.

Parameters:

- **project** *(str)* – project name or folder path (e.g., “project1/folder1”)
- **path** *(Path-like (str or Path))* - local directory path where the annotations will be downloaded. If `None`, the current directory is used.
- **items** *(list of str)* - list of item names whose annotations will be downloaded (e.g., `[“Image_1.jpeg”, “Image_2.jpeg”]`). If the value is `None`, then all the annotations of the given directory will be downloaded.
- **recursive** *(bool)* - download annotations from the project’s root and all of its folders with the preserved structure. If the value is False, download only from the project’s root or given directory.
- **callback** *(callable)* - a function that allows you to modify each annotation’s dict before downloading. The function receives each annotation as an argument and the returned value will be converted to JSON and applied to the download.

Returns: local path of the downloaded annotations folder.

Return type: string

In [5]:
```
# to download annotations from project root
sa.download_annotations(
    project="Image Project",
    path="/Users/arturmanukyan/Desktop/Annotations"
)
```

```
SA-PYTHON-SDK - INFO - Downloading the annotations of the requested items to
/Users/arturmanukyan/Desktop/Annotations/Image Project July 07 2022 13_05
This might take a while…
SA-PYTHON-SDK - INFO - Downloaded annotations for 6 items.
```

Out[5]:
```
'/Users/arturmanukyan/Desktop/Annotations/Image Project July 07 2022 13_05'
```

In [6]:
```
# to download all annotations
sa.download_annotations(
    project="Image Project",
    path="/Users/arturmanukyan/Desktop/Annotations",
    recursive=True
)
```

```
SA-PYTHON-SDK - INFO - Downloading the annotations of the requested items to
/Users/arturmanukyan/Desktop/Annotations/Image Project July 07 2022 13_05
This might take a while…
SA-PYTHON-SDK - INFO - Downloaded annotations for 11 items.
```

Out[6]:
```
'/Users/arturmanukyan/Desktop/Annotations/Image Project July 07 2022 13_05'
```

In [7]:
```
# to download annotations for specific items

# step 1: query completed annotations and store item names
completed_item_names = [
    item['name'] for item in sa.query(
        project="Image Project",
        query="metadata(status = Completed)"
    )
]

print(f'there are {len(completed_item_names)} completed items names are: {co

# step 2: download annotations by providing the list of item names
sa.download_annotations(
    project="Image Project",
    path="/Users/arturmanukyan/Desktop/Annotations",
    items = completed_item_names,
    recursive=True
)
```

```
there are 5 completed items names are: ['AliS copy 3.jpeg', 'AliS copy 10.jp
eg', 'AliS copy 12.jpeg', 'AliS.jpeg', 'AliS copy.jpeg']
SA-PYTHON-SDK - INFO - Downloading the annotations of the requested items to
/Users/arturmanukyan/Desktop/Annotations/Image Project July 07 2022 13_05
This might take a while…
SA-PYTHON-SDK - INFO - Downloaded annotations for 5 items.
```

Out[7]:
```
'/Users/arturmanukyan/Desktop/Annotations/Image Project July 07 2022 13_05'
```

In [8]:
```
# to download annotations using preferred JSON structure

# step 1: define converter function, that accepts SA annotation dict as an a
def convert(sa_annotation):
    filtered_boxes = [
        instance for instance in sa_annotation['instances'] if instance['typ
    ]

    boxes = list(map(lambda box:
            {
                'type': box['type'],
                'class': box['className'],
                'coordinates': box['points']
            }, filtered_boxes)
    )
    return {'objects': boxes}

#step 2: pass converter function to callback parameter
sa.download_annotations(
```

```
    project="Image Project",
    path="/Users/arturmanukyan/Desktop/Annotations",
    items=["AliS.jpeg"],
    recursive=True,
    callback=convert
)
```

```
SA-PYTHON-SDK - INFO - Downloading the annotations of the requested items to
/Users/arturmanukyan/Desktop/Annotations/Image Project July 07 2022 13_05
This might take a while…
SA-PYTHON-SDK - INFO - Downloaded annotations for 1 items.
```

Out[8]: `'/Users/arturmanukyan/Desktop/Annotations/Image Project July 07 2022 13_05'`

`assign_items()` | `unassign_items()`

`SAClient.assign_items(project, path=None, items=None, recursive=False, callback=None)`

Assigns items to a user. The assignment role, QA or Annotator, will be deduced from the user's role in the project. The type of the objects` image, video or text will be deduced from the project type. With SDK, the user can be assigned to a role in the project with the share_project function.

Parameters:

- **project** *(str)* – project name or folder path (e.g., "project1/folder1")
- **items** *(list of str)* - list of items to assign
- **user** *(str)* - user email

Returns:

Return type:

`SAClient.unassign_items(project, items)`

Removes assignment of given items for all assignees. With SDK, the user can be assigned to a role in the project with the share_project function.

Parameters:

- **project** *(str)* – project name or folder path (e.g., "project1/folder1")
- **items** *(list of str)* - list of items to unassign

Returns:

Return type:

In [ ]:
```
assignments = [item['name'] for item in sa.search_items("Image Project")]
print(assignments)

# to assign items to user
sa.assign_items(
    project = "Image Project",
    items = assignments,
```

```
    user = 'superannotator2@gmail.com'
)
```

In [ ]:
```
# to clear assignments from items
sa.unassign_items(
    project = "Image Project",
    items = assignments,
)
```

`delete_items()`

`SAClient.delete_items(project, items)`

Delete items.

Parameters:

- **project** *(str)* – project name or folder path (e.g., “project1/folder1”)
- **items** *(list of str)* - list of item names to delete. If the value is None, all the items in the given directory will be deleted.

Returns:

Return type:

In [ ]:
```
sa.delete_items(
    project="Image Project",
    items=['img_1.png', 'img_2.png']
)
```

SuperAnnotate Client

`class SAClient`

class `superannotate.SAClient(token=None, config_path=None)`

Create `SAClient` instance to authorize SDK in a team scope. If no argument is provided, `SA_TOKEN` environmental variable will be checked or `$HOME/.superannotate/config.json` will be used.

Parameters:

- **token** *(str)* – team token
- **config_path** *(path-like (str or Path))* - path to the config file

Returns:

Return type:

In [ ]:
```
# using team token directly
team_1 = SAClient(
```

```
    token = "48c827b33e1a2a92fcdfccd54ecfebde1a1b787c741dd34935fe6c00430d80e
)

# using path to the config file
team_1 = SAClient(config_path = "~/.superannotate/dev_config.json")
```

If no argument is provided, the algorithm will look up the `SA_TOKEN` environmental variable first. If `SA_TOKEN` is not there, then it will automatically check for the **$HOME/.superannotate/config.json** file

In [ ]:
```
# using without any arguments
team_1 = SAClient()
```

In [ ]:
```
# to create SuperAnnotate client for a team via config.json file
team_1 = SAClient(config_path = "~/.superannotate/dev_config.json")
# to get the name of the team you've instantiated SA Client for
team_1.get_team_metadata()['name']
```

In [ ]:
```
# to create a SuperAnnotate client for another team in parallel (via another
team_2 = SAClient(config_path = "~/.superannotate/dev_2_config.json")
# to get the name of the team you've instantiated SA Client for
team_2.get_team_metadata()['name']
```

In [ ]:
```
# Example script for duplicating projects across different teams
from superannotate import SAClient

# Get project metadata, annotation classes and settings
team_1 = SAClient(config_path = "~/.superannotate/dev_config.json")
project_metadata = team_1.get_project_metadata(
    project="Image Project",
    include_annotation_classes=True,
    include_settings=True,
    include_workflow=False,
    include_contributors=False,
    include_complete_image_count=False)

# Create a project for Team 2 using collected data
team_2 = SAClient(config_path = "~/.superannotate/dev_2_config.json")
team_2.create_project_from_metadata(project_metadata)

# List projects for both teams
team_1.search_projects()
team_2.search_projects()
```