
Parkour Documentation

Release

Evgeny Anatskiy

Jan 30, 2017

Contents

1	Contents	3
1.1	Installation	3
1.2	Parkour API Documentation	3
1.2.1	Request API	3
1.2.2	Library API	4
1.2.3	Sample API	5
1.2.4	Quality Check API	5
1.2.5	Index Generator API	5
1.2.6	Library Preparation API	6
1.2.7	Pooling API	6
1.2.8	Flowcell API	6
2	Indices and tables	9
	Python Module Index	11

Parkour is ... short description.

1.1 Installation

test

1.2 Parkour API Documentation

test

1.2.1 Request API

API operations on requests.

`request.views.delete_request(request)`

POST /request/delete_request/ Delete request with a given id and all its libraries and samples.

Parameters `request_id` – request id

Returns {'success': not error, 'error': error}

Return type JSON response

`request.views.generate_deep_sequencing_request(request)`

Generate Deep Sequencing Request form in PDF.

`request.views.get_all(request)`

GET /request/get_all/ Get the list of all requests.

Returns list with requests

Return type JSON response

`request.views.get_libraries_and_samples(request)`

GET /request/libraries_and_samples/?request_id={request_id} Get the list of all libraries and samples for a given request.

Parameters `request_id` – request id

Returns libraries and samples in a given request

Return type JSON reponse

`request.views.save_request(request)`

POST /request/save_request/ Add new or edit an existing request.

Parameters

- **mode** – add/edit - controles the mode: either add or edit a request
- **request_id** – request id if mode='edit'; otherwise, ''
- **libraries** – list of library ids
- **samples** – list of samples ids
- **description** – request description

Returns {'success': not error, 'error': error}

Return type JSON response

`request.views.upload_deep_sequencing_request(request)`

Upload Deep Sequencing request with PI's signature and change request status to 1.

1.2.2 Library API

API operations on libraries.

`library.views.delete_library(request)`

POST /library/delete_library/ Delete library with a given id.

Parameters `record_id` – library id

Returns {'success': not error, 'error': error}

Return type JSON response

`library.views.get_all(request)`

GET /library/get_all/ Get the list of all libraries and samples.

Returns list with libraries and samples

Return type JSON response

`library.views.get_files(request)`

`library.views.get_library_protocols(request)`

GET /library/get_library_protocols/ Get the list of all library protocols.

Returns list with library protocols

Return type JSON response

`library.views.get_library_type(request)`

GET /library/get_library_type/?library_protocol_id={library_protocol_id}/ Get the list of all library types for a given library protocol.

Parameters `library_protocol_id` – library protocol id

Returns list with library types

Return type JSON response

`library.views.save_library(request)`

Add a new library or update an existing one.

`library.views.upload_files(request)`

1.2.3 Sample API

API operations on samples.

`sample.views.delete_sample(request)`

Delete sample with a given id.

`sample.views.get_files(request)`

`sample.views.get_nucleic_acid_types(request)`

Get the list of all nucleic acid types.

`sample.views.get_sample_protocols(request)`

Get the list of all sample protocols.

`sample.views.save_sample(request)`

Add a new sample or update an existing one.

`sample.views.upload_files(request)`

1.2.4 Quality Check API

API operations on Quality Check.

`quality_check.views.update(request)`

Update library or sample field(s). If a library/sample passes the quality check, then set its status to 2. Otherwise, set the status to -1 and notify the user about the failure via email.

1.2.5 Index Generator API

API operations on pools.

`index_generator.views.generate_indices(request)`

Generate indices for given libraries and samples.

`index_generator.views.pooling_tree(request)`

Get libraries ready for pooling.

`index_generator.views.save_pool(request)`

Create a pool after generating indices, add libraries and “converted” samples to it, update the pool size, and create a Library Preparation object and a Pooling object for each added library/sample.

`index_generator.views.update_index_type(request)`

Update Index Type for a given sample.

`index_generator.views.update_read_length(request)`

Update Read Length for a given library or sample.

1.2.6 Library Preparation API

API operations on Library Preparation.

`library_preparation.views.download_benchtopy_protocol(request)`

Generate Benchtopy Protocol as XLS file for selected samples.

`library_preparation.views.edit(request)`

Edit Library Preparation object.

`library_preparation.views.get_all(request)`

Get the list of all samples for Library Preparation.

`library_preparation.views.upload_benchtopy_protocol(request)`

Upload a file and add it to all samples with a given Library Protocol.

1.2.7 Pooling API

API operations on Pooling.

`pooling.views.download_pooling_template(request)`

`pooling.views.edit(request)`

Edit Pooling object.

`pooling.views.get_all(request)`

Get the list of all libraries.

`pooling.views.get_request(record, record_type)`

Get request for a current library/sample.

`pooling.views.upload_pooling_template(request)`

Upload a file and attach it to a given Pool.

1.2.8 Flowcell API

API operations on flowcells.

`flowcell.views.get_all(request)`

Get the list of all Flowcells.

`flowcell.views.pool_info(request)`

Get additional information for a given pool.

`flowcell.views.pool_list(request)`

Get the list of pools for loading flowcells.

`flowcell.views.save(request)`

Save a new flowcell.

`flowcell.views.sequencer_list` (*request*)
Get the list of all sequencers.

CHAPTER 2

Indices and tables

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

f

flowcell.views, 6

i

index_generator.views, 5

l

library.views, 4

library_preparation.views, 6

p

pooling.views, 6

q

quality_check.views, 5

r

request.views, 3

s

sample.views, 5

D

delete_library() (in module library.views), 4
delete_request() (in module request.views), 3
delete_sample() (in module sample.views), 5
download_benchmark_protocol() (in module library_preparation.views), 6
download_pooling_template() (in module pooling.views), 6

E

edit() (in module library_preparation.views), 6
edit() (in module pooling.views), 6

F

flowcell.views (module), 6

G

generate_deep_sequencing_request() (in module request.views), 3
generate_indices() (in module index_generator.views), 5
get_all() (in module flowcell.views), 6
get_all() (in module library.views), 4
get_all() (in module library_preparation.views), 6
get_all() (in module pooling.views), 6
get_all() (in module request.views), 3
get_files() (in module library.views), 4
get_files() (in module sample.views), 5
get_libraries_and_samples() (in module request.views), 3
get_library_protocols() (in module library.views), 4
get_library_type() (in module library.views), 5
get_nucleic_acid_types() (in module sample.views), 5
get_request() (in module pooling.views), 6
get_sample_protocols() (in module sample.views), 5

I

index_generator.views (module), 5

L

library.views (module), 4

library_preparation.views (module), 6

P

pool_info() (in module flowcell.views), 6
pool_list() (in module flowcell.views), 6
pooling.views (module), 6
pooling_tree() (in module index_generator.views), 5

Q

quality_check.views (module), 5

R

request.views (module), 3

S

sample.views (module), 5
save() (in module flowcell.views), 6
save_library() (in module library.views), 5
save_pool() (in module index_generator.views), 5
save_request() (in module request.views), 4
save_sample() (in module sample.views), 5
sequencer_list() (in module flowcell.views), 6

U

update() (in module quality_check.views), 5
update_index_type() (in module index_generator.views), 6
update_read_length() (in module index_generator.views), 6
upload_benchmark_protocol() (in module library_preparation.views), 6
upload_deep_sequencing_request() (in module request.views), 4
upload_files() (in module library.views), 5
upload_files() (in module sample.views), 5
upload_pooling_template() (in module pooling.views), 6