AeselProjects Documentation

Release 0.0.1

AO Labs

Jan 26, 2019

Contents:

CHAPTER 1

Overview

AeselProjects is a peripheral service of Aesel, providing organization and project management functionality. It is not expected to provide meaningful benefits outside of this architecture.

AeselProjects is a part of the AO Aesel Project, along with CLyman, Crazy Ivan, Adrestia, and Kelona.

CHAPTER 2

Contact

Stuck and need help? Have general questions about the application? We encourage you to publish your question on Stack Overflow. We regularly monitor for the tag 'aesel' in questions.

We encourage the use of Stack Overflow for a few reasons:

- Once the question is answered, it is searchable and viewable by everyone else.
- The forum format offers an easy method to get a larger community involved with a tougher question.

2.1 Getting Started with AeselProjects

Go Home

2.1.1 Docker

An official Docker Image of AeselProjects is provided, and to get you up and running quickly, a Docker Compose file is provided as well. To start up a Mongo instance, a Consul instance, and a AeselProjects instance, simply run the following from the 'compose/min' folder:

```
docker-compose up
```

Once the services have started, test them by hitting AeselProjects' healthcheck endpoint:

```
curl http://localhost:5644/health
```

Keep in mind that this is not a secure deployment, but is suitable for exploring the *AeselProjects API*.

2.1.2 Building from Source

Once you've got the required backend services started, build and execute the tests for the repository.

```
./gradlew check
```

And, finally, start AeselProjects:

```
./gradlew bootRun
```

2.1.3 Using the Latest Release

AeselProjects can also be downloaded as a runnable JAR for the latest release from here.

When using a JAR, unzip the downloaded package, move to the main directory from a terminal, and run:

```
java -jar build/libs/aeselprojects-0.0.1.jar
```

2.2 Project API

A Project contains groups of scenes, as well as Asset Collections. It is primarily used for organization, and helps manage a full-scale animation production.

2.2.1 Project Creation

POST /v1/project

Create a new Project.

Request Headers

- Content-Type – application/json

Status Codes

- 200 OK – Success

http

```
POST /v1/project HTTP/1.1
Host: localhost:5635
Content-Type: application/json

{
    "name": "Test",
    "description": "This is a test",
    "category": "test",
    "tags": ["testTag"],
    "sceneGroups": [
            {
                    "name": "testGroup",
                    "description": "This is a test group",
                    "category": "test",
                    "scenes": ["1234"]
            }
    ],
    "assetCollectionIds": ["4321"]
}
```

curl

```
curl -i -X POST http://localhost:5635/v1/project -H 'Content-Type: application/json' -
↪-data-raw '{"assetCollectionIds": ["4321"], "category": "test", "description":
↪"This is a test", "name": "Test", "sceneGroups": [{"category": "test", "scenes": [
↪"1234"], "name": "testGroup", "description": "This is a test group"}], "tags": [
↪"testTag"]}'
```

wget

```
wget -S -O- http://localhost:5635/v1/project --header='Content-Type: application/json
↪' --post-data='{"assetCollectionIds": ["4321"], "category": "test", "description":
↪"This is a test", "name": "Test", "sceneGroups": [{"category": "test", "scenes": [
↪"1234"], "name": "testGroup", "description": "This is a test group"}], "tags": [
↪"testTag"]}'
```

httpie

```
echo '{
  "assetCollectionIds": [
    "4321"
  ],
  "category": "test",
  "description": "This is a test",
  "name": "Test",
  "sceneGroups": [
    {
      "category": "test",
      "description": "This is a test group",
      "name": "testGroup",
      "scenes": [
        "1234"
      ]
    }
  ],
  "tags": [
    "testTag"
  ]
}' | http POST http://localhost:5635/v1/project Content-Type:application/json
```

python-requests

```
requests.post('http://localhost:5635/v1/project', headers={'Content-Type':
↪'application/json'}, json={'assetCollectionIds': ['4321'], 'category': 'test',
↪'description': 'This is a test', 'name': 'Test', 'sceneGroups': [{'category': 'test
↪', 'scenes': ['1234'], 'name': 'testGroup', 'description': 'This is a test group'}],
↪ 'tags': ['testTag']})
```

response

```
HTTP/1.1 200 OK
Location: http://localhost:5635/v1/project

{
    "id": "5be8eeb4f5eee94951e553a9",
    "name": "Test",
    "description": "This is a test",
    "category": "test",
    "tags": [
```

(continues on next page)

(continued from previous page)

```
        "testTag"
    ],
    "sceneGroups": [
        {
            "name": "testGroup",
            "description": "This is a test group",
            "category": "test",
            "scenes": [
                "1234"
            ]
        }
    ],
    "assetCollectionIds": [
        "4321"
    ]
}
```

2.2.2 Project Retrieval

GET /v1/project/{key}

Get a project by ID.

Status Codes

- 200 OK – Success

http

```
GET /v1/project/{key} HTTP/1.1
Host: localhost:5635
```

curl

```
curl -i 'http://localhost:5635/v1/project/{key}'
```

wget

```
wget -S -O- 'http://localhost:5635/v1/project/{key}'
```

httpie

```
http 'http://localhost:5635/v1/project/{key}'
```

python-requests

```
requests.get('http://localhost:5635/v1/project/{key}')
```

2.2.3 Project Update

POST /v1/project/{key}

Create a new Project.

Request Headers

- Content-Type – application/json

Status Codes

- 200 OK – Success

http

```
POST /v1/project/{key} HTTP/1.1
Host: localhost:5635
Content-Type: application/json

{
    "name": "AnotherName",
    "description": "This is a second test",
    "category": "testing",
    "tags": [
        "testTag2"
    ],
    "sceneGroups": [
        {
            "name": "testGroup2",
            "description": "This is another test group",
            "category": "testing",
            "scenes": [
                "12345"
            ]
        }
    ],
    "assetCollectionIds": [
        "43212"
    ]
}
```

curl

```
curl -i -X POST 'http://localhost:5635/v1/project/{key}' -H 'Content-Type:␣
↪application/json' --data-raw '{"assetCollectionIds": ["43212"], "category": "testing
↪", "description": "This is a second test", "name": "AnotherName", "sceneGroups": [{
↪"category": "testing", "scenes": ["12345"], "name": "testGroup2", "description":
↪"This is another test group"}], "tags": ["testTag2"]}'
```

wget

```
wget -S -O- 'http://localhost:5635/v1/project/{key}' --header='Content-Type:␣
↪application/json' --post-data='{"assetCollectionIds": ["43212"], "category":
↪"testing", "description": "This is a second test", "name": "AnotherName",
↪"sceneGroups": [{"category": "testing", "scenes": ["12345"], "name": "testGroup2",
↪"description": "This is another test group"}], "tags": ["testTag2"]}'
```

httpie

```
echo '{
  "assetCollectionIds": [
    "43212"
  ],
  "category": "testing",
  "description": "This is a second test",
  "name": "AnotherName",
  "sceneGroups": [
    {
```

(continues on next page)

(continued from previous page)

```
      "category": "testing",
      "description": "This is another test group",
      "name": "testGroup2",
      "scenes": [
        "12345"
      ]
    }
  ],
  "tags": [
    "testTag2"
  ]
}' | http POST 'http://localhost:5635/v1/project/{key}' Content-Type:application/json
```

python-requests

```
requests.post('http://localhost:5635/v1/project/{key}', headers={'Content-Type':
↪'application/json'}, json={'assetCollectionIds': ['43212'], 'category': 'testing',
↪'description': 'This is a second test', 'name': 'AnotherName', 'sceneGroups': [{
↪'category': 'testing', 'scenes': ['12345'], 'name': 'testGroup2', 'description':
↪'This is another test group'}], 'tags': ['testTag2']})
```

response

```
HTTP/1.1 200 OK
Location: http://localhost:5635/v1/project

{
    "id": "5be8eeb4f5eee94951e553a9",
    "name": "Test",
    "description": "This is a test",
    "category": "test",
    "tags": [
        "testTag"
    ],
    "sceneGroups": [
        {
            "name": "testGroup",
            "description": "This is a test group",
            "category": "test",
            "scenes": [
                "1234"
            ]
        }
    ],
    "assetCollectionIds": [
        "4321"
    ]
}
```

2.2.4 Project Query

GET /v1/project

Query for projects by attribute.

Status Codes

- 200 OK – Success

http

```
GET /v1/project?name=test&num_records=10&page=0 HTTP/1.1
Host: localhost:5635
```

curl

```
curl -i 'http://localhost:5635/v1/project?name=test&num_records=10&page=0'
```

wget

```
wget -S -O- 'http://localhost:5635/v1/project?name=test&num_records=10&page=0'
```

httpie

```
http 'http://localhost:5635/v1/project?name=test&num_records=10&page=0'
```

python-requests

```
requests.get('http://localhost:5635/v1/project?name=test&num_records=10&page=0')
```

response

```
HTTP/1.1 200 OK
Location: http://localhost:5635/v1/project?name=AnotherName&num_records=10&page=0

[
    {
        "id": "5be8eeb4f5eee94951e553a9",
        "name": "AnotherName",
        "description": "This is a second test",
        "category": "testing",
        "tags": [
            "testTag2"
        ],
        "sceneGroups": [
            {
                "name": "testGroup2",
                "description": "This is another test group",
                "category": "testing",
                "scenes": [
                    "12345"
                ]
            }
        ],
        "assetCollectionIds": [
            "43212"
        ]
    }
]
```

2.2.5 Project Delete

DELETE /v1/project/{key}

Delete a project by ID.

Status Codes

- 200 OK – Success

http

```
DELETE /v1/project/{key} HTTP/1.1
Host: localhost:5635
```

curl

```
curl -i -X DELETE 'http://localhost:5635/v1/project/{key}'
```

wget

```
wget -S -O- --method=DELETE 'http://localhost:5635/v1/project/{key}'
```

httpie

```
http DELETE 'http://localhost:5635/v1/project/{key}'
```

python-requests

```
requests.delete('http://localhost:5635/v1/project/{key}')
```

2.3 Developer Notes

This page contains a series of notes intended to be beneficial for any contributors to AeselProjects.

2.3.1 Continuous Integration

Travis CI is used to run automated tests against AeselProjects each time a commit or pull request is submitted against the main repository. The configuration for this can be updated via the .travis.yml file in the main folder of the project repository.

Latest CI Runs

2.3.2 Documentation

Documentation is built using Sphinx and hosted on Read the Docs.

Updates to documentation can be made in the docs/ folder of the project repository, with files being in the .rst format.

Go Home

HTTP Routing Table

/v1