Vacuity

May 24, 2022

Contents:

1 Available clients **1**
- 1.1 vacuity.client.cvtc module . . . 1
- 1.2 vacuity.client.mock module . . . 4

2 Client abstraction **7**
- 2.1 vacuity.client.abstract module . . . 7

3 vacuity package **11**
- 3.1 Submodules . . . 11
- 3.2 vacuity.controllers module . . . 11
- 3.3 Module contents . . . 11

4 Indices and tables **13**

Python Module Index **15**

Index **17**

CHAPTER 1

Available clients

This page lists built-in Vacuity clients. These can be used to connect to a room schedule and lab availability database.

Clients are instantiated in your `run` file. For example, this file instantiates and runs using the Mock client:

```
from flask import Flask

import vacuity.controllers
from vacuity.client import mock

vacuity.controllers.COMPLEX = mock.MOCKED_COMPLEX

# Static folder is disabled as long as we only have one blueprint
# https://stackoverflow.com/a/25804585
app = Flask(__name__, static_folder=None)

# Clean up Jinja output
app.jinja_env.trim_blocks = True
app.jinja_env.lstrip_blocks = True

app.register_blueprint(vacuity.controllers.main, url_prefix="/")
```

When saved as `run.py` and run with ``gunicorn run:app`, this app loads the mocked complex defined in the mock module. The useful line is `vacuity.controllers.COMPLEX = ...`, which sets the complex used in the thread.

1.1 vacuity.client.cvtc module

Client for CVTC room schedule data API and LabStats lab usage API

This client retrieves its list of rooms from LabStats. Each top-level group in LabStats is considered a building in the Complex. Every group underneath a root group is considered a candidate for display in Vacuity. Candidates are selected for pre-filtering if they match these criteria:

- Group contains stations, not groups

- **Group has a selection criteria, EITHER:**
  - Group has "lab features enabled"
  - Group's description contains the string "vacuity_enable"

Once a group meets these criteria, its lab and room schedule information will be displayed in Vacuity. To prevent lab usage information from being displayed, add "vacuity_nolab" to the group's description.

class vacuity.client.cvtc.**CVTCBuilding**(*identifier*, *labstats_group: labstats_api.models.Group = None*, *session: requests.sessions.Session = None*)

Bases: *vacuity.client.abstract.Building*

Parameters

- **identifier** – The Banner ID of this building
- **labstats_group** – An optional LabStats group representing this building. Searches for room groups will occur within this group.

abbreviation
A shorter name for the building, one to five characters long ideally.

Used to denote the building alongside room codes or anywhere else that it is not possible to display the building description.

description
A long, human-recognizable name for the building.

Used whenever possible when the user needs to select a building. For example, a list of building descriptions is displayed when asking the user to filter by building.

identifier
The meaningful identifier of this building.

Used to represent this building when requesting information about it from the backend.

room_for_identifier(*identifier*)
Returns the Room with the specified identifier.

Raises ***RoomNotFoundError*** – The requested identifier does not correspond to a room we know about.

rooms
Returns a list of Room that this building contains.

class vacuity.client.cvtc.**CVTCComplex**(*labstats_api_url*, *labstats_api_key*, *session: requests.sessions.Session*)

Bases: *vacuity.client.abstract.Complex*

Uses hard-coded data to return CVTC buildings

Parameters

- **labstats_api_url** – URL of hosted LabStats instance's API.
- **labstats_api_key** – API key used to get data from LabStats.
- **session** – A Requests Session object. It is recommended to use CacheControl or a similar library to add caching, else you will hit API rate limiting.

building_for_identifier(*identifier*)
Returns the Building with the specified identifier.

Raises ***BuildingNotFoundError*** – The requested identifier does not correspond to a building we know about.

buildings
Returns all of the Buildings we know about.

classmethod formal_id_for_banner_id(*banner_id*)
Given the Banner ID of a CVTC building, get the ID used elsewhere.

For example, the Business Education Center is "EC BUS" in Banner but "BEC" elsewhere.

vacuity_feedback_url
Returns an HTML-safe feedback URL.

This URL will be placed in a "Give Feedback" link near the bottom of every page.

If you do not provide a feedback URL, the "Give Feedback" link will not be shown.

class vacuity.client.cvtc.**CVTCLab**(*labstats_group: labstats_api.models.Group*)
Bases: *vacuity.client.abstract.ComputerLab*

A Lab which pulls data from LabStats.

Parameters **labstats_group** – Group corresponding to this Lab.

computer_availability_now
A LabComputerAvailability representing the lab's current state.

installed_software_names
Human-readable names of interesting software installed in this lab.

"Interesting" depends on your institution. All of the software installed on the machine may be interesting to you, or only a few things.

class vacuity.client.cvtc.**CVTCLabComputerAvailability**(*available*, *total*)
Bases: *vacuity.client.abstract.LabComputerAvailability*

Point-in-time state of computers available in a lab at CVTC.

available
The number of computers that are available for use.

summary
A single-word summary of the availability of computers in the room.

See also:

LabComputerAvailabilitySummary

total
The number of computers that the room contains.

class vacuity.client.cvtc.**CVTCRoom**(*identifier*, *banner_building_id*, *formal_building_id*, *labstats_group: labstats_api.models.Group = None*, *session: requests.sessions.Session = None*)
Bases: *vacuity.client.abstract.Room*

A Room capable of combining data from LabStats and Banner.

Parameters

- **identifier** – This room's meaningful ID.
- **banner_building_id** – The name of the building this room belongs to in banner.
- **formal_building_id** – The name of the building that this room belongs to, according to the "rest of the world".

- **labstats_group** – The LabStats group corresponding to this room. If provided, this Room is capable of returning computer lab usage data. If not provided, any requests for computer lab usage data will return None.

availability_now
A `RoomAvailability` for the room's status at access time.

Of course, like any value, you may choose to cache results so "now" may be "15 minutes ago", just as long as the data is up-to-date enough for your users.

code
The human-readable name for the room.

May be a number (e.g. 104), a name (e.g. Bailey), or even the same as the identifier.

identifier
The meaningful identifier of this room.

Used to represent this room when requesting information about it from the backend.

lab
A `LabInformation` representing the computer lab this room contains, or None if the room does not contain a computer lab.

class vacuity.client.cvtc.**CVTCRoomAvailability**(*moment*, *summary*, *humanized*)
Bases: `tuple`

humanized
Alias for field number 2

moment
Alias for field number 0

summary
Alias for field number 1

vacuity.client.cvtc.**event_json_to_naive_events**(*event_json*)
Converts a very specific JSON-formatted list of objects to Chronology NaiveEvents.

vacuity.client.cvtc.**get_group_in_list_of_groups**(*group_name*, *group_list*)
Returns the group with name matching group_name in group_list.

If no match is found, ValueError is raised.

1.2 vacuity.client.mock module

Client implementation with pre-made test and demonstration data

class vacuity.client.mock.**MockBuilding**(*identifier: str*, *abbreviation: str*, *description: str*, *rooms: dict*)
Bases: *vacuity.client.abstract.Building*

abbreviation
A shorter name for the building, one to five characters long ideally.

Used to denote the building alongside room codes or anywhere else that it is not possible to display the building description.

description
A long, human-recognizable name for the building.

Used whenever possible when the user needs to select a building. For example, a list of building descriptions is displayed when asking the user to filter by building.

identifier
The meaningful identifier of this building.

Used to represent this building when requesting information about it from the backend.

room_for_identifier(*identifier*)
Returns the Room with the specified identifier.

Raises ***RoomNotFoundError*** – The requested identifier does not correspond to a room we know about.

rooms
Returns a list of Room that this building contains.

class vacuity.client.mock.**MockComplex**(*buildings: dict*)
Bases: *vacuity.client.abstract.Complex*

building_for_identifier(*identifier*)
Returns the Building with the specified identifier.

Raises ***BuildingNotFoundError*** – The requested identifier does not correspond to a building we know about.

buildings
Returns all of the Buildings we know about.

class vacuity.client.mock.**MockComputerLab**(*total_computers: int, available_computers: int, software: List[str]*)
Bases: *vacuity.client.abstract.ComputerLab*

computer_availability_now
A `LabComputerAvailability` representing the lab's current state.

installed_software_names
Human-readable names of interesting software installed in this lab.

"Interesting" depends on your institution. All of the software installed on the machine may be interesting to you, or only a few things.

class vacuity.client.mock.**MockLabComputerAvailability**(*available*, *total*)
Bases: *vacuity.client.abstract.LabComputerAvailability*

available
The number of computers that are available for use.

summary
A single-word summary of the availability of computers in the room.

See also:

`LabComputerAvailabilitySummary`

total
The number of computers that the room contains.

class vacuity.client.mock.**MockRoom**(*identifier: str, code: str, unavailability: List[vacuity.client.mock.MockUnavailableBlock], lab: vacuity.client.abstract.ComputerLab = None*)
Bases: *vacuity.client.abstract.Room*

availability_now
A `RoomAvailability` for the room's status at access time.

Of course, like any value, you may choose to cache results so "now" may be "15 minutes ago", just as long as the data is up-to-date enough for your users.

code
The human-readable name for the room.

May be a number (e.g. 104), a name (e.g. Bailey), or even the same as the identifier.

identifier
The meaningful identifier of this room.

Used to represent this room when requesting information about it from the backend.

lab
A `LabInformation` representing the computer lab this room contains, or None if the room does not contain a computer lab.

class vacuity.client.mock.**MockRoomAvailability**(*moment*, *summary*, *humanized*)
Bases: `tuple`

humanized
Alias for field number 2

moment
Alias for field number 0

summary
Alias for field number 1

class vacuity.client.mock.**MockUnavailableBlock**(*start*, *end*)
Bases: `tuple`

end
Alias for field number 1

start
Alias for field number 0

vacuity.client.mock.**midnight_shifted_by**(*hours*, *minutes*)
Utility to shift today (at module import time) by hours and minutes

CHAPTER 2

Client abstraction

Implement these abstractions to allow your data source to be displayed in Vacuity.

See *Available clients* for information on clients that are available from the current Vacuity codebase, and how to use them.

Client modules should **never** carry state unless it is saved in an external place. A separate module state could be created for each thread or worker in a WSGI server.

2.1 vacuity.client.abstract module

class vacuity.client.abstract.**Complex**
Bases: abc.ABC

Manages initial access to *Building*, allowing filtering.

building_for_identifier(*identifier*) → vacuity.client.abstract.Building
Returns the Building with the specified identifier.

Raises ***BuildingNotFoundError*** – The requested identifier does not correspond to a building we know about.

buildings
Returns all of the Buildings we know about.

vacuity_feedback_url
Returns an HTML-safe feedback URL.

This URL will be placed in a "Give Feedback" link near the bottom of every page.

If you do not provide a feedback URL, the "Give Feedback" link will not be shown.

class vacuity.client.abstract.**Building**
Bases: abc.ABC

Describes a building.

A building contains *Room*

abbreviation
A shorter name for the building, one to five characters long ideally.

Used to denote the building alongside room codes or anywhere else that it is not possible to display the building description.

description
A long, human-recognizable name for the building.

Used whenever possible when the user needs to select a building. For example, a list of building descriptions is displayed when asking the user to filter by building.

identifier
The meaningful identifier of this building.

Used to represent this building when requesting information about it from the backend.

room_for_identifier(*identifier*) → vacuity.client.abstract.Room
Returns the Room with the specified identifier.

Raises ***RoomNotFoundError*** – The requested identifier does not correspond to a room we know about.

rooms
Returns a list of Room that this building contains.

class vacuity.client.abstract.**ComputerLab**
Bases: abc.ABC

A group of computers in a room with a common set of installed software.

computer_availability_now
A *LabComputerAvailability* representing the lab's current state.

installed_software_names
Human-readable names of interesting software installed in this lab.

"Interesting" depends on your institution. All of the software installed on the machine may be interesting to you, or only a few things.

class vacuity.client.abstract.**LabComputerAvailability**
Bases: abc.ABC

A point-in-time state of the available or unavailable computers in a lab.

available
The number of computers that are available for use.

summary
A single-word summary of the availability of computers in the room.

See also:

LabComputerAvailabilitySummary

total
The number of computers that the room contains.

class vacuity.client.abstract.**AvailabilitySummary**
Bases: enum.Enum

Represents momentary one-word of availability of a Vacuity resource.

GOOD means the resource is plentiful.

POOR means the resource is not very available.

NONE means the resource is unavailable (or very close to being unavailable, depending on your needs).

ERROR means Vacuity could not determine the state of the resource.

ERROR = 'error'

GOOD = 'good'

NONE = 'none'

POOR = 'poor'

class vacuity.client.abstract.**Room**

Bases: abc.ABC

Describes a room.

A room may be scheduled, and it may contain computers, of which some are available and others are in use.

availability_now

A *RoomAvailability* for the room's status at access time.

Of course, like any value, you may choose to cache results so "now" may be "15 minutes ago", just as long as the data is up-to-date enough for your users.

code

The human-readable name for the room.

May be a number (e.g. 104), a name (e.g. Bailey), or even the same as the identifier.

identifier

The meaningful identifier of this room.

Used to represent this room when requesting information about it from the backend.

lab

A LabInformation representing the computer lab this room contains, or None if the room does not contain a computer lab.

class vacuity.client.abstract.**RoomAvailability**

Bases: abc.ABC

Describes a room's availability status at a point in time

humanized

A human-readable description of when the room will become available or unavailable.

For example, "available for four hours", "available until 4PM", or "unavailable until 3:55 PM". The format is up to you, but try to keep it between three to five words.

moment

The moment in time that this unavailable status represents

summary

Whether the room is available or not.

See also:

RoomAvailabilitySummary

class vacuity.client.abstract.**VacuityException**

Bases: Exception

Base class for errors arising from Vacuity

class vacuity.client.abstract.**BuildingNotFoundError**
Bases: *vacuity.client.abstract.VacuityException*

Thrown when a Building could not be found in a Complex

class vacuity.client.abstract.**RoomNotFoundError**
Bases: *vacuity.client.abstract.VacuityException*

Thrown when a Room could not be found in a Building

CHAPTER 3

vacuity package

3.1 Submodules

3.2 vacuity.controllers module

Uses client abstractions to serve Vacuity webpages.

Set the COMPLEX module variable prior to calling any functions in this module.

vacuity.controllers.**browse**(**args*, ***kwargs*)

vacuity.controllers.**browse_redirect**()

vacuity.controllers.**get_proper_os**(*possible: str*) → str
Returns “Windows” or “macOS” if a similar string is passed in, “any” otherwise.

Effectively escapes any values and ensures they are correct for passing to output or other Vacuity functions.

Similarity to “Windows” or “macOS” is determined by casefolding both the input and the desired strings. Any possible inputs that don’t match these are discarded and “any” is returned.

vacuity.controllers.**index**(**args*, ***kwargs*)

vacuity.controllers.**lab**(**args*, ***kwargs*)

vacuity.controllers.**route_with_complex**(*f*)

3.3 Module contents

A web application to help people find space on campus.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

v

vacuity, 11
vacuity.client.cvtc, 1
vacuity.client.mock, 4
vacuity.controllers, 11

Index

A

abbreviation (*vacuity.client.abstract.Building attribute*), 7
abbreviation (*vacuity.client.cvtc.CVTCBuilding attribute*), 2
abbreviation (*vacuity.client.mock.MockBuilding attribute*), 4
availability_now (*vacuity.client.abstract.Room attribute*), 9
availability_now (*vacuity.client.cvtc.CVTCRoom attribute*), 4
availability_now (*vacuity.client.mock.MockRoom attribute*), 5
AvailabilitySummary (*class in vacuity.client.abstract*), 8
available (*vacuity.client.abstract.LabComputerAvailability attribute*), 8
available (*vacuity.client.cvtc.CVTCLabComputerAvailability attribute*), 3
available (*vacuity.client.mock.MockLabComputerAvailability attribute*), 5

B

browse() (*in module vacuity.controllers*), 11
browse_redirect() (*in module vacuity.controllers*), 11
Building (*class in vacuity.client.abstract*), 7
building_for_identifier() (*vacuity.client.abstract.Complex method*), 7
building_for_identifier() (*vacuity.client.cvtc.CVTCComplex method*), 2
building_for_identifier() (*vacuity.client.mock.MockComplex method*), 5
BuildingNotFoundError (*class in vacuity.client.abstract*), 9
buildings (*vacuity.client.abstract.Complex attribute*), 7
buildings (*vacuity.client.cvtc.CVTCComplex attribute*), 3
buildings (*vacuity.client.mock.MockComplex attribute*), 5

C

code (*vacuity.client.abstract.Room attribute*), 9
code (*vacuity.client.cvtc.CVTCRoom attribute*), 4
code (*vacuity.client.mock.MockRoom attribute*), 6
Complex (*class in vacuity.client.abstract*), 7
computer_availability_now (*vacuity.client.abstract.ComputerLab attribute*), 8
computer_availability_now (*vacuity.client.cvtc.CVTCLab attribute*), 3
computer_availability_now (*vacuity.client.mock.MockComputerLab attribute*), 5
ComputerLab (*class in vacuity.client.abstract*), 8
CVTCBuilding (*class in vacuity.client.cvtc*), 2
CVTCComplex (*class in vacuity.client.cvtc*), 2
CVTCLab (*class in vacuity.client.cvtc*), 3
CVTCLabComputerAvailability (*class in vacuity.client.cvtc*), 3
CVTCRoom (*class in vacuity.client.cvtc*), 3
CVTCRoomAvailability (*class in vacuity.client.cvtc*), 4

D

description (*vacuity.client.abstract.Building attribute*), 8
description (*vacuity.client.cvtc.CVTCBuilding attribute*), 2
description (*vacuity.client.mock.MockBuilding attribute*), 4

E

end (*vacuity.client.mock.MockUnavailableBlock attribute*), 6
ERROR (*vacuity.client.abstract.AvailabilitySummary attribute*), 9

event_json_to_naive_events() (*in module vacuity.client.cvtc*), 4

F

formal_id_for_banner_id() (*vacuity.client.cvtc.CVTCComplex class method*), 3

G

get_group_in_list_of_groups() (*in module vacuity.client.cvtc*), 4
get_proper_os() (*in module vacuity.controllers*), 11
GOOD (*vacuity.client.abstract.AvailabilitySummary attribute*), 9

H

humanized (*vacuity.client.abstract.RoomAvailability attribute*), 9
humanized (*vacuity.client.cvtc.CVTCRoomAvailability attribute*), 4
humanized (*vacuity.client.mock.MockRoomAvailability attribute*), 6

I

identifier (*vacuity.client.abstract.Building attribute*), 8
identifier (*vacuity.client.abstract.Room attribute*), 9
identifier (*vacuity.client.cvtc.CVTCBuilding attribute*), 2
identifier (*vacuity.client.cvtc.CVTCRoom attribute*), 4
identifier (*vacuity.client.mock.MockBuilding attribute*), 5
identifier (*vacuity.client.mock.MockRoom attribute*), 6
index() (*in module vacuity.controllers*), 11
installed_software_names (*vacuity.client.abstract.ComputerLab attribute*), 8
installed_software_names (*vacuity.client.cvtc.CVTCLab attribute*), 3
installed_software_names (*vacuity.client.mock.MockComputerLab attribute*), 5

L

lab (*vacuity.client.abstract.Room attribute*), 9
lab (*vacuity.client.cvtc.CVTCRoom attribute*), 4
lab (*vacuity.client.mock.MockRoom attribute*), 6
lab() (*in module vacuity.controllers*), 11
LabComputerAvailability (*class in vacuity.client.abstract*), 8

M

midnight_shifted_by() (*in module vacuity.client.mock*), 6
MockBuilding (*class in vacuity.client.mock*), 4
MockComplex (*class in vacuity.client.mock*), 5
MockComputerLab (*class in vacuity.client.mock*), 5
MockLabComputerAvailability (*class in vacuity.client.mock*), 5
MockRoom (*class in vacuity.client.mock*), 5
MockRoomAvailability (*class in vacuity.client.mock*), 6
MockUnavailableBlock (*class in vacuity.client.mock*), 6
moment (*vacuity.client.abstract.RoomAvailability attribute*), 9
moment (*vacuity.client.cvtc.CVTCRoomAvailability attribute*), 4
moment (*vacuity.client.mock.MockRoomAvailability attribute*), 6

N

NONE (*vacuity.client.abstract.AvailabilitySummary attribute*), 9

P

POOR (*vacuity.client.abstract.AvailabilitySummary attribute*), 9

R

Room (*class in vacuity.client.abstract*), 9
room_for_identifier() (*vacuity.client.abstract.Building method*), 8
room_for_identifier() (*vacuity.client.cvtc.CVTCBuilding method*), 2
room_for_identifier() (*vacuity.client.mock.MockBuilding method*), 5
RoomAvailability (*class in vacuity.client.abstract*), 9
RoomNotFoundError (*class in vacuity.client.abstract*), 10
rooms (*vacuity.client.abstract.Building attribute*), 8
rooms (*vacuity.client.cvtc.CVTCBuilding attribute*), 2
rooms (*vacuity.client.mock.MockBuilding attribute*), 5
route_with_complex() (*in module vacuity.controllers*), 11

S

start (*vacuity.client.mock.MockUnavailableBlock attribute*), 6
summary (*vacuity.client.abstract.LabComputerAvailability attribute*), 8
summary (*vacuity.client.abstract.RoomAvailability attribute*), 9

summary (*vacuity.client.cvtc.CVTCLabComputerAvailability attribute*), 3
summary (*vacuity.client.cvtc.CVTCRoomAvailability attribute*), 4
summary (*vacuity.client.mock.MockLabComputerAvailability attribute*), 5
summary (*vacuity.client.mock.MockRoomAvailability attribute*), 6

T

total (*vacuity.client.abstract.LabComputerAvailability attribute*), 8
total (*vacuity.client.cvtc.CVTCLabComputerAvailability attribute*), 3
total (*vacuity.client.mock.MockLabComputerAvailability attribute*), 5

V

vacuity (*module*), 11
vacuity.client.cvtc (*module*), 1
vacuity.client.mock (*module*), 4
vacuity.controllers (*module*), 11
vacuity_feedback_url (*vacuity.client.abstract.Complex attribute*), 7
vacuity_feedback_url (*vacuity.client.cvtc.CVTCComplex attribute*), 3
VacuityException (*class in vacuity.client.abstract*), 9