CERC persistence reference manual

Release 1.0.0.2

CERC Next-Generation Cities

Apr 08, 2024

CONTENTS:

1 CERC PERSISTENCE' reference manual **1**
- 1.1 Authors . . . 1
- 1.2 Contributors . . . 1
- 1.3 About the PERSISTENCE . . . 1
- 1.4 Folder structure . . . 2
- 1.5 Model Classes . . . 2
  - 1.5.1 Application . . . 2
  - 1.5.2 City . . . 2
  - 1.5.3 CityObject . . . 2
  - 1.5.4 SimulationResults . . . 3
  - 1.5.5 User . . . 3
- 1.6 Repository Classes . . . 3
  - 1.6.1 Application . . . 3
  - 1.6.2 City . . . 4
  - 1.6.3 CityObject . . . 5
  - 1.6.4 SimulationResults . . . 6
  - 1.6.5 User . . . 7
- 1.7 Classes . . . 8
  - 1.7.1 Configuration . . . 8
  - 1.7.2 DB Control . . . 8
  - 1.7.3 DB Setup . . . 11
  - 1.7.4 Repository . . . 11

2 Additional Files **12**
- 2.1 Readme . . . 12
- 2.2 License . . . 12
- 2.3 Code of conduct . . . 12
- 2.4 How to contribute . . . 12
- 2.5 Coding style . . . 12

Index **13**

CHAPTER

ONE

CERC PERSISTENCE' REFERENCE MANUAL

1.1 Authors

- Peter Yefi
- Guillermo Gutierrez Morote

1.2 Contributors

- Ruben Sanchez

1.3 About the PERSISTENCE

This document contains the essential documentation for the cerc persistence package, a set of classes, that allows the permanent storage of a cec_hub city. cerc persistence package is composed of two main components: **repositories** and **models**.

1.4 Folder structure

```
../cerc_persistence/cerc_persistence/
├── configuration.py
├── db_control.py
├── db_setup.py
├── __init__.py
├── models
│   ├── application.py
│   ├── city_object.py
│   ├── city.py
│   ├── __init__.py
│   ├── simulation_results.py
│   └── user.py
├── repositories
│   ├── application.py
│   ├── city_object.py
│   ├── city.py
│   ├── __init__.py
│   ├── simulation_results.py
│   └── user.py
├── repository.py
└── version.py

3 directories, 18 files
```

1.5 Model Classes

1.5.1 Application

class Application(**args: Any*, ***kwargs: Any*)

Inherit: declarative_base

Application(Models) class

1.5.2 City

class City(**args: Any*, ***kwargs: Any*)

Inherit: declarative_base

City(Models) class

1.5.3 CityObject

class CityObject(**args: Any*, ***kwargs: Any*)

Inherit: declarative_base

CityObject(Models) class

1.5.4 SimulationResults

class SimulationResults(**args: Any*, ***kwargs: Any*)

Inherit: declarative_base

SimulationResults(Models) class

1.5.5 User

class User(**args: Any*, ***kwargs: Any*)

Inherit: declarative_base

User(Models) class

1.6 Repository Classes

1.6.1 Application

class Application(*db_name*, *dotenv_path*, *app_env*)

Inherit: Repository

Application(Repository) class

delete(*application_uuid: str*)

Deletes an application with the application_uuid

Parameter application_uuid The application uuid

Returns None

get_by_uuid(*application_uuid: str*) → *Application*

Fetch Application based on the application uuid

Parameter application_uuid the application uuid

Returns Application with the provided application_uuid

insert(*name: str*, *description: str*, *application_uuid: str*)

Inserts a new application

Parameter name Application name

Parameter description Application description

Parameter application_uuid Unique identifier for the application

Returns Identity id

update(*application_uuid: str*, *name: str*, *description: str*)

Updates an application

Parameter application_uuid the application uuid of the application to be updated

Parameter name the application name

Parameter description the application description

Returns None

1.6.2 City

class City(*db_name*, *dotenv_path*, *app_env*)

Inherit: Repository

City(Repository) class

delete(*city_id: int*)

Deletes a City with the id

Parameter city_id the city id

Returns None

get_by_user_id_and_application_id(*user_id*, *application_id*) → [City'>]

Fetch city based on the user who created it

Parameter user_id the user id

Parameter application_id the application id

Returns ModelCity

get_by_user_id_application_id_and_scenario(*user_id*, *application_id*, *scenario*) → [City'>]

Fetch city based on the user who created it

Parameter user_id the user id

Parameter application_id the application id

Parameter scenario simulation scenario name

Returns [ModelCity]

insert(*city: hub.city_model_structure.city.City*, *pickle_path*, *scenario*, *application_id*, *user_id: int*)

Inserts a city

Parameter city The complete city instance

Parameter pickle_path Path to the pickle

param scenario: Simulation scenario name

Parameter application_id Application id owning the instance

Parameter user_id User id owning the instance

Returns Identity id

update(*city_id: int*, *city: hub.city_model_structure.city.City*)

Updates a city name (other updates makes no sense)

Parameter city_id the id of the city to be updated

Parameter city the city object

Returns None

1.6.3 CityObject

class CityObject(*db_name*, *dotenv_path*, *app_env*)

Inherit: Repository

CityObject(Repository) class

building_in_cities_info(*name*, *cities*)

Fetch a city object based on name and city id

Parameter name city object name

Parameter cities city identifiers

Returns [CityObject] with the provided name or alias belonging to the city with id city_id

delete(*city_id: int*, *name: str*)

Deletes an application with the application_uuid

Parameter city_id The id for the city owning the city object

Parameter name The city object name

Returns None

get_by_name_or_alias_and_city(*name*, *city_id*) → OptionalCityObject]

Fetch a city object based on name and city id

Parameter name city object name

Parameter city_id a city identifier

Returns [CityObject] with the provided name or alias belonging to the city with id city_id

get_by_name_or_alias_in_cities(*name*, *city_ids*) → *CityObject*

Fetch a city object based on name and city ids

Parameter name city object name

Parameter city_ids a list of city identifiers

Returns [CityObject] with the provided name or alias belonging to the city with id city_id

insert(*city_id: int*, *building: hub.city_model_structure.building.Building*)

Inserts a new city object

Parameter city_id city id for the city owning this city object

Parameter building the city object (only building for now) to be inserted

return Identity id

update(*city_id: int*, *building: hub.city_model_structure.building.Building*)

Updates an application

Parameter city_id the city id of the city owning the city object

Parameter building the city object

Returns None

1.6.4 SimulationResults

class SimulationResults(*db_name*, *dotenv_path*, *app_env*)

Inherit: Repository

SimulationResults(Repository) class

delete(*name: str*, *city_id=None*, *city_object_id=None*)

Deletes an application with the application_uuid

Parameter name The simulation results tool and workflow name

Parameter city_id The id for the city owning the simulation results

Parameter city_object_id the id for the city_object owning these simulation results

Returns None

get_simulation_results_by_city_id_city_object_id_and_names(*city_id*, *city_object_id*, *result_names=None*) → [SimulationResults'>]

Fetch the simulation results based in the city_id or city_object_id with the given names or all

Parameter city_id the city id

Parameter city_object_id the city object id

Parameter result_names if given filter the results

Returns [SimulationResult]

get_simulation_results_by_city_object_id_and_names(*city_object_id*, *result_names=None*) → [SimulationResults'>]

Fetch the simulation results based in the city_object_id with the given names or all

Parameter city_object_id the city object id

Parameter result_names if given filter the results

Returns [SimulationResult]

insert(*name: str*, *values: str*, *city_id=None*, *city_object_id=None*)

Inserts simulations results linked either with a city as a whole or with a city object

Parameter name results name

Parameter values the simulation results in json format

Parameter city_id optional city id

Parameter city_object_id optional city object id

Returns Identity id

update(*name: str*, *values: str*, *city_id=None*, *city_object_id=None*)

Updates simulation results for a city or a city object

Parameter name The simulation results tool and workflow name

Parameter values the simulation results in json format

Parameter city_id optional city id

Parameter city_object_id optional city object id

Returns None

1.6.5 User

class User(*db_name*, *dotenv_path*, *app_env*)

Inherit: Repository

User(Repository) class

delete(*user_id: int*)

Deletes a user with the id

Parameter user_id the user id

Returns None

get_by_name_and_application(*name: str*, *application_id: int*) → *User*

Fetch user based on the email address

Parameter name Username

Parameter application_id User application name

Returns User matching the search criteria or None

get_by_name_application_id_and_password(*name: str*, *password: str*, *application_id: int*) → *User*

Fetch user based on the name, password and application id

Parameter name Username

Parameter password User password

Parameter application_id Application id

Returns User

get_by_name_application_uuid_and_password(*name: str*, *password: str*, *application_uuid: str*) → *User*

Fetch user based on the email and password

Parameter name Username

Parameter password User password

Parameter application_uuid Application uuid

Returns User

insert(*name: str*, *password: str*, *role: UserRoles*, *application_id: int*)

Inserts a new user

Parameter name username

Parameter password user password

Parameter role user rol [Admin or Hub_Reader]

Parameter application_id user application id

Returns Identity id

update(*user_id: int*, *name: str*, *password: str*, *role: UserRoles*)

Updates a user

Parameter user_id the id of the user to be updated

Parameter name the name of the user

Parameter password the password of the user

Parameter role the role of the user

Returns None

1.7 Classes

1.7.1 Configuration

class Configuration(*db_name: str*, *dotenv_path: str*, *app_env='TEST'*)

Inherit: :py:class:object

Configuration class to hold common persistence configuration

property connection_string

Returns a connection string postgresql

Returns connection string

property db_name

retrieve the configured database name

property db_user

retrieve the configured username

1.7.2 DB Control

class DBControl(*db_name*, *app_env*, *dotenv_path*)

Inherit: :py:class:object

DBFactory class

add_simulation_results(*name*, *values*, *city_id=None*, *city_object_id=None*)

Add simulation results to the city or to the city_object to the database

Parameter name simulation and simulation engine name

Parameter values simulation values in json format

Parameter city_id city id or None

Parameter city_object_id city object id or None

application_info(*application_uuid*) → *Application*

Retrieve the application info for the given uuid from the database

Parameter application_uuid the uuid for the application

Returns Application

building(*name*, *user_id*, *application_id*, *scenario*) → *CityObject*

Retrieve the building from the database

Parameter name Building name

Parameter user_id User id

Parameter application_id Application id

Parameter scenario Scenario

:

`building_info`(*name*, *city_id*) → *CityObject*

Retrieve the building info from the database

Parameter name Building name

Parameter city_id City ID

Returns CityObject

`building_info_in_cities`(*name*, *cities*) → *CityObject*

Retrieve the building info from the database

Parameter name Building name

Parameter cities [City ID]

Returns CityObject

`cities_by_user_and_application`(*user_id*, *application_id*) → [City'>]

Retrieve the cities belonging to the user and the application from the database

Parameter user_id User id

Parameter application_id Application id

Returns [City]

`create_user`(*name: str*, *application_id: int*, *password: str*, *role: UserRoles*)

Creates a new user in the database

Parameter name the name of the user

Parameter application_id the application id of the user

Parameter password the password of the user

Parameter role the role of the user

`delete_application`(*application_uuid*)

Deletes a single application from the database

Parameter application_uuid the id of the application to get

`delete_city`(*city_id*)

Deletes a single city from the database

Parameter city_id the id of the city to get

`delete_results_by_name`(*name*, *city_id=None*, *city_object_id=None*)

Deletes city object simulation results from the database

Parameter name simulation name

Parameter city_id if given, delete delete the results for the city with id city_id

Parameter city_object_id if given, delete delete the results for the city object with id city_object_id

`delete_user`(*user_id*)

Delete a single user from the database

Parameter user_id the id of the user to delete

get_by_name_and_application(*name: str*, *application: int*)

Retrieve a single user from the database

Parameter name username

Parameter application application accessing hub

persist_application(*name: str*, *description: str*, *application_uuid: str*)

Creates information for an application in the database

Parameter name name of application

Parameter description the description of the application

Parameter application_uuid the uuid of the application to be created

persist_city(*city: City*, *pickle_path*, *scenario*, *application_id: int*, *user_id: int*)

Creates a city into the database

Parameter city City to be stored

Parameter pickle_path Path to save the pickle file

Parameter scenario Simulation scenario name

Parameter application_id Application id owning this city

Parameter user_id User who create the city

return identity_id

results(*user_id*, *application_id*, *request_values*, *result_names=None*) → Dict

Retrieve the simulation results for the given cities from the database

Parameter user_id the user id owning the results

Parameter application_id the application id owning the results

Parameter request_values dictionary containing the scenario and building names to grab the results

Parameter result_names if given, filter the results to the selected names

update_application(*name: str*, *description: str*, *application_uuid: str*)

Update the application information stored in the database

Parameter name name of application

Parameter description the description of the application

Parameter application_uuid the uuid of the application to be created

update_city(*city_id*, *city*)

Update an existing city in the database

Parameter city_id the id of the city to update

Parameter city the updated city object

update_user(*user_id: int*, *name: str*, *password: str*, *role: UserRoles*)

Updates a user in the database

Parameter user_id the id of the user

Parameter name the name of the user

Parameter password the password of the user

Parameter role the role of the user

user_info(*name*, *password*, *application_id*) → *User*

Retrieve the user info for the given name and password and application_id from the database

Parameter name the username

Parameter password the user password

Parameter application_id the application id

Returns User

user_login(*name*, *password*, *application_uuid*) → *User*

Retrieve the user info from the database

Parameter name the username

Parameter password the user password

Parameter application_uuid the application uuid

Returns User

1.7.3 DB Setup

class DBSetup(*db_name*, *app_env*, *dotenv_path*, *admin_password*, *application_uuid*)

Inherit: :py:class:object

Creates a Persistence database structure

1.7.4 Repository

class Repository(*db_name*, *dotenv_path: str*, *app_env='TEST'*)

Inherit: :py:class:object

Base repository class to establish db connection

CHAPTER

TWO

ADDITIONAL FILES

2.1 Readme

README.md

2.2 License

LICENSE.md

2.3 Code of conduct

CODE_OF_CONDUCT.md

2.4 How to contribute

CONTRIBUTING.md

2.5 Coding style

PYGUIDE.md

INDEX

A

add_simulation_results() (*cerc_persistence.db_control.DBControl method*), 8
Application (*built-in class*), 2, 3
application_info() (*cerc_persistence.db_control.DBControl method*), 8

B

building() (*cerc_persistence.db_control.DBControl method*), 8
building_in_cities_info() (*cerc_persistence.repositories.city_object.CityObject method*), 5
building_info() (*cerc_persistence.db_control.DBControl method*), 9
building_info_in_cities() (*cerc_persistence.db_control.DBControl method*), 9

C

cities_by_user_and_application() (*cerc_persistence.db_control.DBControl method*), 9
City (*built-in class*), 2, 4
CityObject (*built-in class*), 2, 5
Configuration (*built-in class*), 8
connection_string (*cerc_persistence.configuration :property:.Configuration property*), 8
create_user() (*cerc_persistence.db_control.DBControl method*), 9

D

db_name (*cerc_persistence.configuration :property:.Configuration property*), 8
db_user (*cerc_persistence.configuration :property:.Configuration property*), 8
DBControl (*built-in class*), 8
DBSetup (*built-in class*), 11
delete() (*cerc_persistence.repositories.application.Application method*), 3
delete() (*cerc_persistence.repositories.city.City method*), 4
delete() (*cerc_persistence.repositories.city_object.CityObject method*), 5
delete() (*cerc_persistence.repositories.simulation_results.SimulationResults method*), 6
delete() (*cerc_persistence.repositories.user.User method*), 7
delete_application() (*cerc_persistence.db_control.DBControl method*), 9
delete_city() (*cerc_persistence.db_control.DBControl method*), 9
delete_results_by_name() (*cerc_persistence.db_control.DBControl method*), 9
delete_user() (*cerc_persistence.db_control.DBControl method*), 9

G

get_by_name_and_application() (*cerc_persistence.db_control.DBControl method*), 9
get_by_name_and_application() (*cerc_persistence.repositories.user.User method*), 7
get_by_name_application_id_and_password() (*cerc_persistence.repositories.user.User method*), 7
get_by_name_application_uuid_and_password() (*cerc_persistence.repositories.user.User method*), 7
get_by_name_or_alias_and_city() (*cerc_persistence.repositories.city_object.CityObject method*), 5
get_by_name_or_alias_in_cities() (*cerc_persistence.repositories.city_object.CityObject method*), 5
get_by_user_id_and_application_id() (*cerc_persistence.repositories.city.City method*), 4
get_by_user_id_application_id_and_scenario() (*cerc_persistence.repositories.city.City method*), 4
get_by_uuid() (*cerc_persistence.repositories.application.Application method*), 3
get_simulation_results_by_city_id_city_object_id_and_names() (*cerc_persistence.repositories.simulation_results.SimulationResults method*), 6
get_simulation_results_by_city_object_id_and_names() (*cerc_persistence.repositories.simulation_results.SimulationResults method*), 6

I

insert() (*cerc_persistence.repositories.application.Application method*), 3
insert() (*cerc_persistence.repositories.city.City method*), 4
insert() (*cerc_persistence.repositories.city_object.CityObject method*), 5
insert() (*cerc_persistence.repositories.simulation_results.SimulationResults method*), 6
insert() (*cerc_persistence.repositories.user.User method*), 7

P

persist_application() (*cerc_persistence.db_control.DBControl method*), 10
persist_city() (*cerc_persistence.db_control.DBControl method*), 10

R

Repository (*built-in class*), 11
results() (*cerc_persistence.db_control.DBControl method*), 10

S

SimulationResults (*built-in class*), 3, 6

U

update() (*cerc_persistence.repositories.application.Application method*), 3
update() (*cerc_persistence.repositories.city.City method*), 4
update() (*cerc_persistence.repositories.city_object.CityObject method*), 5
update() (*cerc_persistence.repositories.simulation_results.SimulationResults method*), 6
update() (*cerc_persistence.repositories.user.User method*), 7
update_application() (*cerc_persistence.db_control.DBControl method*), 10
update_city() (*cerc_persistence.db_control.DBControl method*), 10
update_user() (*cerc_persistence.db_control.DBControl method*), 10
User (*built-in class*), 3, 7
user_info() (*cerc_persistence.db_control.DBControl method*), 10
user_login() (*cerc_persistence.db_control.DBControl method*), 11