wehrlab-datajoint

Release 0.1.0

jonny

Jun 09, 2022

PYTHON:

tryin to put the data in a place

CHAPTER

ONE

CONNECT

wehrdj.connect.**connect**(*host=None*, *user=None*, *password=None*, *file=PosixPath('/home/docs/.djcredentials.json')*, ***kwargs*) → Connection

Connect to datajoint database.

Wrapper around `datajoint.conn()`, but loads credentials from an (unencrypted) file.

A bit less implicit than using environment variables, and less manual than typing them in every time.

Please for the love of god do not commit any passwords to a git repo ever.

Parameters

- **host** (*str*) – IP Address
- **user** (*str*) – Username (typically "root" by default)
- **password** (*str*) – uh Password
- **file** (`pathlib.Path`) – file to load/save credentials to
- ****kwargs** – passed to dj.conn

Returns `datajoint.connection.Connection`

CHAPTER

TWO

ELEMENTS

Module that contains imports and activations for datajoint elements schemas.

Other schema should be written elsewhere, presumably in a schema module, and then given a central `activate` function..

Don't be fooled by the "module imported but not used" errors your linter will give you, for some reason you do have to import *Subject* et al even if they aren't used directly. Don't ask me why.

`wehrdj.elements.`**`activate`**`()`

Call the activation functions from each of the imported elements. Must have already called *`wehrdj.connect()`*

Currently:

- element_lab.lab
- element_animal.subject
- element_animal.genotyping
- element_session.session

It uses `wehrdj.elements` as the linking module, which I believe is necessary because it looks for a particular context when instantiating the schema? Not really sure on that one.

CHAPTER

THREE

INGEST

Ingestion routines for existing data!

3.1 Colony

Ingest the colony database into datajoint

wehrdj.ingest.colony.**MOUSE_DB_MAP = {'DOB': 'subject_birth_date', "Date Sac'd": 'death_date', 'Protocol': 'protocol', 'Sex': 'sex', 'Unnamed: 0': 'subject'}**

Mapping between values in our database and names in the datajoint model

class wehrdj.ingest.colony.**MouseDB**(*data=None*, *index: Axes | None = None*, *columns: Axes | None = None*, *dtype: Dtype | None = None*, *copy: bool | None = None*)

Trivial subtype of dataframe to indicate this is a mouse db dataframe

wehrdj.ingest.colony.**insert_subjects**(*mousedb:* wehrdj.ingest.colony.MouseDB)

Insert the loaded subject database into the datajoint database.

wehrdj.connect() must have already been called.

Parameters **mousedb** (*MouseDB*) – the loaded mouse database

wehrdj.ingest.colony.**load_mouse_db**(*path: pathlib.Path*) → *wehrdj.ingest.colony.MouseDB*

Load the mouse database from a .csv export of the "Mice" sheet from the colony database

Parameters **path** (pathlib.Path) – The path of the csv exported from google sheets

Returns *MouseDB*

3.2 Session

3.3 Utils

Utility functions for ingestion! what else!

wehrdj.ingest.utils.**col_to_datetime**(*column: pandas.core.series.Series*) → pandas.core.series.Series

Fix date column with improperly padded m/d/y formatting

wehrdj.ingest.utils.**filter_nans**(*df: pandas.core.frame.DataFrame*) → pandas.core.frame.DataFrame

filter any rows with NANs with warning

CHAPTER

FOUR

INTERFACE

Tools for making abstract interfaces from data structures to datajoint models

4.1 Interface Classes

Tools for ingesting data into a datajoint database.

Model a datajoint schema, tagging properties of the model as needed

class wehrdj.interface.interface.**SchemaInterface**

Metaclass for making interfaces from existing data structures to datajoint schema.

To use:

- Assign the relevant schema as a `schema` class attribute
- Write code to either assign values from your particular data structure as attributes or properties with the same names as the fields in the database
- Connect and hopefully it will be able to insert your row!

For an example, see `Session`

property field_names: List[str]

List of fields that must be defined for this schema

Returns list[str] = list of field names

property field_values: Dict[str, Any]

Values that have been given, either as attrs or properties, for all the items in the schema

Returns dict[str, Any]

insert(***kwargs*)

Insert this schema entry into the table. You must have already called `wehrcj.connect()` or else datajoint will prompt you.

Parameters ****kwargs** – passed on to `insert()`

Raises *wehrdj.exceptions.ValidationError* –

property name: str

Name of this schema (gotten from the schema's __name__ attr)

Returns str

abstract property schema: UserTable

The schema that this class models.

(Should be overridden as a class attribute rather than a property, this is just how the abc interface works)

property table: datajoint_babel.model.table.Table

The abstract representation of the datajoint model in the `definition`

eg. for session:

```
Table(
  name='Session',
  tier='Manual',
  comment=None,
  keys=[
    Dependency(dependency='Subject'),
    Attribute(
      name='session_datetime',
      datatype=DJ_Type(datatype='datetime', args=3, unsigned=False),
      comment='',
      default=None
    )
  ],
  attributes=[]
)
```

Returns `Table`

validate() → bool

Validate that all the required fields of this schema have been provided

Returns `True` if they have all been declared

Return type bool

CHAPTER

FIVE

EXCEPTIONS

exception wehrdj.exceptions.**ValidationError**

We did not have all the requirements for insertion into the table!

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

W

INDEX