Sqlite-Integrated Documentation

Contents

Module `sqlite_integrated`

Classes

Class `Database`

```
class Database(
    path: str,
    new=False,
    default_id_field='id',
    silent=False
)
```

Main database class for manipulating sqlite3 databases

```
Parameters:
    path:               Path to the database file

Optional
    new:                A new blank database will be created where the ´self.path´ is pointing
    default_id_field:   The default name for the id field in tables
    silent:             Disables all feedback in the form of prints
```

Instance variables

Variable `conn` The sqlite3 connection

Variable `cursor` The sqlite3 cursor

Variable `default_id_field` The default name for the id_field in returned DatabaseEntry

Variable `path` Path to the database file

Variable `silent` Disables all feedback in the form of prints

Methods

Method `add_table_entry`

```
def add_table_entry(
    self,
    table,
    entry: dict,
    fill_null=False,
    silent=False
)
```

Add an entry to the database. The entry must have values for all fields in the table. You can pass ´fill_null=True´ to fill remaining fields with None/null. Use ´silent=True´ to suppress warnings and messages.

Method `close`

```
def close(
    self
)
```

saves and closes the database. If you want to explicitly close without saving use: ´self.conn.close()´

Method `get_entry_by_id`

```
def get_entry_by_id(
    self,
    table,
    ID,
    id_field=None
)
```

Get table entry by id

Method `get_table`

```
def get_table(
    self,
    name: str,
    get_only=None,
    id_field=None
) -> list
```

Returns all entries in a table as python dictionaries. This function loops over all entries in the table, so it is not the best in big databases

Method `get_table_collums`

```
def get_table_collums(
    self,
    name: str
)
```

Returns the collum names for a given table

Method `get_table_info`

```
def get_table_info(
    self,
    name: str
)
```

Returns sql information about a table (runs PRAGMA TABLE_INFO(name))

Method `get_table_names`

```
def get_table_names(
    self
) -> list
```

Returns the names of all tables in the database

Method `get_table_raw`

```
def get_table_raw(
    self,
    name: str,
    get_only=None
) -> list
```

Returns all entries in a table as tuples

Method `is_table`

```
def is_table(
    self,
    table_name: str
) -> bool
```

Check if database has a table with a certain name

Method `overview`

```
def overview(
    self
)
```

Returns an overview of all the tables in the database with their fields. Intended to to be run in a python shell or print with ´print´

Method `raw_entry_to_entry`

```
def raw_entry_to_entry(
    self,
    raw_entry: tuple,
    table: str,
    id_field,
    fields=None
) -> sqlite_integrated.DatabaseEntry
```

Convert a raw entry (tuple) to a DatabaseEntry

Method `save`

```
def save(
    self
)
```

Writes any changes to the database file

Method `table_overview`

```
def table_overview(
    self,
    name: str,
    max_len: int = 40,
    get_only=None
)
```

Returns a pretty table (with a name). Intended to to be run in a python shell or print with ´print´

Method `update_table_entry`

```
def update_table_entry(
    self,
    entry: sqlite_integrated.DatabaseEntry,
    id_field: str = None,
    fill_null=False,
    silent=False
)
```

Update entry in database with a DatabaseEntry

Class `DatabaseEntry`

```
class DatabaseEntry(
    entry_dict: dict,
    table: str,
    id_field
)
```

A python dictionary that keeps track of the table where it came from, and the name and value of its id field. This class is not supposed to be created manually

" Constructs the entry by saving the table and id_field as attributes. The ´entry_dict´ is used to populate this object with data.

```
Parameters:
    id_field: The collum name for the entry's id
    table:      The name of the table the entry is a part of
    entry_dict: A dictionary containing all the information. This information can be accesed just li
```

Ancestors (in MRO)

- builtins.dict

Class `DatabaseException`

```
class DatabaseException(
    *args,
    **kwargs
)
```

Raised when the database fails to execute command

Ancestors (in MRO)

- builtins.Exception
- builtins.BaseException

Generated by *pdoc* 0.10.0 (https://pdoc3.github.io).