oerforge.db_utils

Database utilities for OERForge: initialization, asset tracking, page-file relationships, site metadata, and general-purpose queries for SQLite.

Overview

`oerforge.db_utils` provides a set of functions to manage and interact with the SQLite database used in the OERForge project. It supports schema setup, record insertion and fetching, asset linking, conversion tracking, and pretty-printing for debugging. Designed for contributors and users who need to extend or inspect the database layer.

Functions

initialize_database

```
def initialize_database()
```

Initialize the database schema, create all required tables, and insert default conversion capabilities.

---

set_relative_link

```
def set_relative_link(content_id, relative_link, db_path=None)
```

Update the `relative_link` for a content item.

Parameters - `content_id` (int): Content item ID. - `relative_link` (str): Relative link to set. - `db_path` (str, optional): Path to the database file.

---

set_menu_context

```
def set_menu_context(content_id, menu_context, db_path=None)
```

Update the `menu_context` for a content item.

Parameters - `content_id` (int): Content item ID. - `menu_context` (str): Menu context value. - `db_path` (str, optional): Path to the database file.

---

get_menu_items

```
def get_menu_items(db_path=None)
```

Fetch all menu items with their links and context.

Parameters - `db_path` (str, optional): Path to the database file.

Returns - `list[dict]`: List of menu item dictionaries.

---

get_db_connection

```
def get_db_connection(db_path=None)
```

Get a SQLite3 connection to the database.

Parameters - `db_path` (str, optional): Path to the database file.

Returns - `sqlite3.Connection`: Connection object.

---

get_records

```
def get_records(table_name, where_clause=None, params=None, db_path=None, conn=None, cursor=
```

Fetch records from a table with optional WHERE clause and parameters.

Parameters - `table_name` (str): Table name. - `where_clause` (str, optional): SQL WHERE clause (without 'WHERE'). - `params` (tuple/list, optional): Parameters for WHERE clause. - `db_path` (str, optional): Path to the database file. - `conn`, `cursor`: Optional existing connection/cursor.

Returns - `list[dict]`: List of records as dictionaries.

---

insert_records

```
def insert_records(table_name, records, db_path=None, conn=None, cursor=None)
```

Batch insert records into any table. Returns list of inserted row IDs.

Parameters - `table_name` (str): Table name. - `records` (list[dict]): List of column-value dicts. - `db_path` (str, optional): Path to the database file. - `conn`, `cursor`: Optional existing connection/cursor.

Returns - `list[int]`: List of inserted row IDs.

---

get__enabled__conversions

```
def get_enabled_conversions(source_format, db_path=None)
```

Get enabled target formats for a given source format.

Parameters - `source_format` (str): Source file extension (e.g., '.md'). - `db_path` (str, optional): Path to the database file.

Returns - `list[str]`: List of enabled target formats.

---

pretty__print__table

```
def pretty_print_table(table_name, db_path=None, conn=None, cursor=None)
```

Pretty-print all rows of a table to the log and terminal for inspection/debugging.

Parameters - `table_name` (str): Table name. - `db_path` (str, optional): Path to the database file. - `conn`, `cursor`: Optional existing connection/cursor.

---

log__event

```
def log_event(message, level="INFO")
```

Log an event to both stdout and a log file in the project root.

Parameters - `message` (str): Log message. - `level` (str): Severity level ("INFO", "ERROR", etc.).

---

link__files__to__pages

```
def link_files_to_pages(file_page_pairs, db_path=None, conn=None, cursor=None)
```

Link files to pages in the `pages_files` table.

Parameters - `file_page_pairs` (list[tuple]): Each tuple is (file__id, page__path). - `db_path` (str, optional): Path to the database file. - `conn`, `cursor`: Optional existing connection/cursor.

---

get__available__conversions__for__page

```
def get_available_conversions_for_page(output_path, db_path=None)
```

Return all successful conversions for a page output path.

Parameters - `output_path` (str): Output path of the page. - `db_path` (str, optional): Path to the database file.

Returns - `list[dict]`: List of dicts with `target_format`, `output_path`, and `status`.

Usage Example

```python
from oerforge import db_utils
conn = db_utils.get_db_connection()
db_utils.pretty_print_table('content', conn=conn)
```

Requirements

- Python 3.7+
- SQLite3

See Also

- SQLite Documentation

License

See `LICENSE` in the project root.