# db\_utils.py — OERForge Database Utilities

## Overview

db\_utils.py provides utility functions for initializing, managing, and interacting with the SQLite database used in the OERForge project. It supports asset tracking, page-file relationships, site metadata, and general-purpose queries and inserts. This module is designed to be approachable for new programmers and extensible for contributors.

## Module Docstring

"""  
OERForge Database Utilities  
==========================  
  
This module provides utility functions for initializing, managing, and interacting with the SQLite database used in the OERForge project. It supports asset tracking, page-file relationships, site metadata, and general-purpose queries and inserts.  
  
Features:  
 - Database initialization and schema setup  
 - General-purpose record fetching and insertion  
 - Logging of database events  
 - Utility functions for linking files to pages  
 - Pretty-printing tables for debugging and inspection  
  
Intended Audience:  
 - Contributors to OERForge  
 - Anyone needing to interact with or extend the database layer  
  
Usage:  
 Import this module and use the provided functions to initialize the database, insert or fetch records, and link files to pages. All functions are documented with clear arguments and return values.  
"""

## Functions

### initialize\_database()

def initialize\_database():  
 """  
 Initializes the SQLite database for asset tracking in the OERForge project.  
  
 This function creates the following tables:  
 - files: Stores metadata about tracked files/assets.  
 - pages\_files: Maps files to pages where they are referenced.  
 - content: Tracks source and output paths for content.  
 - site\_info: Stores site-wide metadata and configuration.  
  
 Existing tables are dropped before creation to ensure a clean state.  
 The database file is located at <project\_root>/db/sqlite.db.  
 """

**Purpose:** Sets up the database schema, ensuring all necessary tables exist and are clean for a new build.

### get\_db\_connection(db\_path=None)

def get\_db\_connection(db\_path=None):  
 """  
 Returns a sqlite3 connection to the database.  
  
 Args:  
 db\_path (str, optional): Path to the SQLite database file.  
 If None, defaults to <project\_root>/db/sqlite.db.  
  
 Returns:  
 sqlite3.Connection: A connection object to the SQLite database.  
 """

**Purpose:** Provides a reusable way to obtain a database connection, using the default project path if none is specified.

### get\_records(table\_name, where\_clause=None, params=None, db\_path=None, conn=None, cursor=None)

def get\_records(table\_name, where\_clause=None, params=None, db\_path=None, conn=None, cursor=None):  
 """  
 Fetch records from a table with optional WHERE clause and parameters.  
 Args:  
 table\_name (str): Name of the table to query.  
 where\_clause (str, optional): SQL WHERE clause (without 'WHERE').  
 params (tuple or list, optional): Parameters for the WHERE clause.  
 db\_path (str, optional): Path to the SQLite database file.  
 conn, cursor: Optional existing connection/cursor.  
 Returns:  
 list of dict: List of records as dictionaries.  
 """

**Purpose:** Allows flexible querying of any table, returning results as a list of dictionaries for easy use in Python code.

### insert\_records(table\_name, records, db\_path=None, conn=None, cursor=None)

def insert\_records(table\_name, records, db\_path=None, conn=None, cursor=None):  
 """  
 General-purpose batch insert for any table.  
 Checks if table exists, inserts records, returns list of inserted row ids.  
 Args:  
 table\_name (str): Name of the table to insert into.  
 records (list of dict): Each dict contains column-value pairs.  
 db\_path (str, optional): Path to the SQLite database file.  
 conn, cursor: Optional existing connection/cursor.  
 Returns:  
 list of int: List of inserted row ids.  
 """

**Purpose:** Efficiently inserts multiple records into any table, handling connection management and error logging.

### pretty\_print\_table(table\_name, db\_path=None, conn=None, cursor=None)

def pretty\_print\_table(table\_name, db\_path=None, conn=None, cursor=None):  
 """  
 Pretty-print all rows of a table to the log for inspection/debugging.  
  
 Args:  
 table\_name (str): Name of the table to print.  
 db\_path (str, optional): Path to the SQLite database file.  
 conn, cursor: Optional existing connection/cursor.  
  
 Returns:  
 None  
  
 Output:  
 Logs a formatted table to the scan.log file and stdout.  
 """

**Purpose:** Helps developers inspect the contents of any table in a readable format, useful for debugging and development.

### log\_event(message, level="INFO")

def log\_event(message, level="INFO"):  
 """  
 Logs an event to both stdout and a log file in the project root.  
  
 Args:  
 message (str): The log message to record.  
 level (str): The severity level (e.g., "INFO", "ERROR", "WARNING").  
  
 The log file is named 'scan.log' and is located at <project\_root>/scan.log.  
 Each log entry is timestamped.  
 """

**Purpose:** Centralizes logging for all database operations, making it easier to track and debug issues.

### link\_files\_to\_pages(file\_page\_pairs, db\_path=None, conn=None, cursor=None)

def link\_files\_to\_pages(file\_page\_pairs, db\_path=None, conn=None, cursor=None):  
 """  
 Link files to pages in the pages\_files table.  
  
 Args:  
 file\_page\_pairs (list of tuple): Each tuple is (file\_id, page\_path).  
 db\_path (str, optional): Path to the SQLite database file.  
 conn, cursor: Optional existing connection/cursor.  
  
 Returns:  
 None  
  
 Example:  
 link\_files\_to\_pages([(1, 'index.html'), (2, 'about.html')])  
 """

**Purpose:** Specifically designed to create links between files and pages in the pages\_files table, supporting asset tracking and page relationships.

## Best Practices

* Always close database connections when done to avoid resource leaks.
* Use the provided logging functions to track database operations and errors.
* When adding new tables or relationships, update initialize\_database() and consider creating new utility functions for those tables.
* Read function docstrings for argument details and usage examples.

## Example Usage

from oerforge import db\_utils  
  
db\_utils.initialize\_database()  
conn = db\_utils.get\_db\_connection()  
files = db\_utils.get\_records('files', where\_clause='is\_image=1')  
db\_utils.pretty\_print\_table('files')  
db\_utils.link\_files\_to\_pages([(1, 'index.html'), (2, 'about.html')])

## For New Programmers

* Read the module and function docstrings for guidance.
* If you are unsure about database operations, start with get\_records and pretty\_print\_table to explore the data.
* Use logging to help debug and understand what your code is doing.
* Ask questions and refer to Python’s official documentation for sqlite3 if you want to learn more about database programming.

*This documentation is intended to be verbose and beginner-friendly. For further help, see the code comments and docstrings in db\_utils.py itself.*