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

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
"""
```

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=
    """
    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)

```python
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)

```python
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")`

```python
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)`

```python
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

```python
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.