oerforge.scan

Static Site Asset and Content Scanner for OERForge

Overview

`oerforge.scan` provides functions for scanning site content, extracting assets, and populating the SQLite database with page, section, and file records. It supports Markdown, Jupyter Notebooks, and DOCX files, and maintains hierarchical relationships from the Table of Contents (TOC) YAML. Logging is standardized and all major operations are traced for debugging.

Functions

batch_read_files

```
def batch_read_files(file_paths)
```

Read multiple files and return their contents as a dictionary `{path: content}`. Supports `.md`, `.ipynb`, `.docx`, and other file types.

Parameters - `file_paths` (list[str]): List of file paths.

Returns - `dict`: Mapping of file paths to contents.

---

read_markdown_file

```
def read_markdown_file(path)
```

Read a Markdown file and return its content as a string.

Parameters - `path` (str): Path to the Markdown file.

Returns - `str` or `None`: File content or `None` on error.

---

read_notebook_file

```
def read_notebook_file(path)
```

Read a Jupyter notebook file and return its content as a dictionary.

Parameters - `path` (str): Path to the notebook file.

Returns - `dict` or `None`: Notebook content or `None` on error.

---

read_docx_file

```
def read_docx_file(path)
```

Read a DOCX file and return its text content as a string.

Parameters - `path` (str): Path to the DOCX file.

Returns - `str` or `None`: Text content or `None` on error.

---

batch_extract_assets

```
def batch_extract_assets(contents_dict, content_type, **kwargs)
```

Extract assets from multiple file contents in one pass. Returns a dictionary `{path: [asset_records]}`.

Parameters - `contents_dict` (dict): Mapping of file paths to contents. - `content_type` (str): Type of content ('markdown', 'notebook', 'docx', etc.). - `**kwargs`: Additional arguments for DB connection/cursor.

Returns - `dict`: Mapping of file paths to lists of asset records.

---

extract_linked_files_from_markdown_content

```
def extract_linked_files_from_markdown_content(md_text, page_id=None)
```

Extract asset links from Markdown text.

Parameters - `md_text` (str): Markdown content. - `page_id` (optional): Page identifier for DB linking.

Returns - `list[dict]`: File record dicts for each asset found.

---

extract_linked_files_from_notebook_cell_content

```
def extract_linked_files_from_notebook_cell_content(cell, nb_path=None)
```

Extract asset links from a notebook cell.

Parameters - `cell` (dict): Notebook cell. - `nb_path` (str, optional): Notebook file path.

Returns - `list[dict]`: File record dicts for each asset found.

---

extract_linked_files_from_docx_content

```
def extract_linked_files_from_docx_content(docx_path, page_id=None)
```

Extract asset links from a DOCX file.

Parameters - `docx_path` (str): Path to DOCX file. - `page_id` (optional): Page identifier for DB linking.

Returns - `list[dict]`: File record dicts for each asset found.

---

populate_site_info_from_config

```
def populate_site_info_from_config(config_filename='_config.yml')
```

Populate the `site_info` table from the given config file.

Parameters - `config_filename` (str): Name of the config file (default: '_config.yml').

---

get_conversion_flags

```
def get_conversion_flags(extension)
```

Get conversion capability flags for a given file extension using the database.

Parameters - `extension` (str): File extension (e.g., '.md', '.ipynb').

Returns - `dict`: Conversion capability flags.

---

scan_toc_and_populate_db

```
def scan_toc_and_populate_db(config_path)
```

Walk the TOC from the config YAML, read each file, extract assets/images, and populate the DB with content and asset records. Maintains TOC hierarchy and section relationships.

Parameters - `config_path` (str): Path to the config YAML file.

---

get_descendants_for_parent

```
def get_descendants_for_parent(parent_output_path, db_path)
```

Query all children, grandchildren, and deeper descendants for a given parent section using a recursive CTE.

Parameters - `parent_output_path` (str): Output path of the parent section. - `db_path` (str): Path to the SQLite database.

Returns - `list[dict]`: Dicts for each descendant (id, title, output_path, parent_output_path, slug, level).

Usage Example

```
from oerforge import scan
scan.scan_toc_and_populate_db('_content.yml')
```

Requirements

- Python 3.7+
- SQLite3
- PyYAML
- python-docx

See Also

- Python-Markdown
- Jupyter Notebook Format
- python-docx Documentation

License

See `LICENSE` in the project root.