

oerforge.verify — Accessibility Verification and Reporting

Overview

`oerforge.verify` provides tools for automated accessibility checking, badge generation, and reporting for static HTML sites. It integrates with Pa11y for WCAG compliance checks, manages results in a SQLite database, and generates visual badges and detailed reports for each page. This module is designed for new users, educators, and developers seeking robust accessibility workflows in open educational resources.

- **Automated accessibility checks (WCAG AA/AAA) using Pa11y**
 - **Badge and report generation for each HTML page**
 - **Database integration for tracking results**
 - **Jinja2-based report rendering**
 - **Robust logging and error handling**
-

Functions

`load_pally_config(yml_path='_content.yml')`

Load Pa11y configuration and logo info from YAML and JSON files.

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

Returns - dict: Parsed configuration dictionary.

`run_pally_on_file(html_path, config_path=None, wcag_level='AA')`

Run Pa11y accessibility checks on a single HTML file.

Parameters - `html_path` (str): Path to the HTML file. - `config_path` (str, optional): Path to Pa11y config file. - `wcag_level` (str): WCAG level ('AA' or 'AAA').

Returns - list[dict] or None: Parsed Pa11y JSON results, or None on error.

Notes - Handles errors robustly, logs issues, and attempts to parse output even on failure.

`get_content_id_for_file(html_path, conn)`

Get the content ID for a given HTML file from the database.

Parameters - `html_path` (str): Path to the HTML file. - `conn` (sqlite3.Connection): Database connection.

Returns - `int` or `None`: Content ID if found, else `None`.

store_accessibility_result(`content_id`, `pally_json`, `badge_html`, `wcag_level`, `error_count`, `warning_count`, `notice_count`, `conn=None`)

Store the latest accessibility result for a page in the database.

Parameters - `content_id` (int): Content ID for the page. - `pally_json` (list[dict]): Pally results. - `badge_html` (str): Badge HTML markup. - `wcag_level` (str): WCAG level. - `error_count`, `warning_count`, `notice_count` (int): Issue counts. - `conn` (sqlite3.Connection): Database connection.

generate_badge_html(`wcag_level`, `error_count`, `logo_info`, `report_link`)

Generate badge HTML for a given WCAG level and error count.

Parameters - `wcag_level` (str): WCAG level. - `error_count` (int): Number of errors. - `logo_info` (dict): Mapping of WCAG levels to badge/logo URLs. - `report_link` (str): Link to the accessibility report.

Returns - `str`: Badge HTML markup.

inject_badge_into_html(`html_path`, `badge_html`, `report_link`, `logo_info`)

Inject the badge/button into the HTML file after `<main>`.

Parameters - `html_path` (str): Path to the HTML file. - `badge_html` (str): Badge HTML markup. - `report_link` (str): Link to the accessibility report. - `logo_info` (dict): Badge/logo info.

generate_nav_menu(`context`)

Generate top-level navigation menu items from the content database.

Parameters - `context` (dict): Context dict, should include `'rel_path'`.

Returns - `list[dict]`: List of menu item dicts: `{'title': str, 'link': str}`

generate_wcag_report(html_path, issues, badge_html, config)

Generate a detailed HTML accessibility report for a file using Jinja2 templates.

Parameters - **html_path** (str): Path to the HTML file. - **issues** (list[dict]): Pa11y issues. - **badge_html** (str): Badge HTML markup. - **config** (dict): Page and site config.

Returns - **str**: Path to the generated report file.

process_all_html_files(build_dir='build', config_file=None, db_path='db/sqlite.db')

Process all HTML files in the build directory: - Run Pa11y checks - Store results in DB - Generate badges and reports - Inject badges into HTML - Copy changed files to docs/

Parameters - **build_dir** (str): Build directory. - **config_file** (str, optional): Pa11y config file. - **db_path** (str): Path to SQLite database.

copy_to_docs()

Copy all changed files from build/ to docs/.

main()

CLI entry point. Parses arguments, runs checks, stores results, and generates reports as needed.

Logging

All major operations and errors are logged to `log/pa11y.log` for debugging and auditing.

Error Handling

Robust error handling is implemented for subprocess calls, file I/O, database operations, and template rendering. All failures are logged with context.

Example Usage

```
from oerforge import verify
verify.process_all_html_files()
```

See Also

- [Pa11y Documentation](#)
 - [WCAG Guidelines](#)
 - [Jinja2 Templates](#)
-

License

See LICENSE in the project root.