seoaudit

Release 0.1dev4

Emanuel Guberović

Sep 16, 2022

CONTENTS:

CHAPTER

ONE

USING CLI SCRIPT

Using CLI

For analyzing a single site with default checks run with *seoaudit -u URL*, e.g.: *seoaudit -u https://green-light.agency*

To define extra urls just add another *u URL* argument: *seoaudit -u https://green-light.agency -u https://milenial.eu*

To use custom python checks config file (e.g. config.py) use option *-c PYTHON_MODULE*: *seoaudit -u https://green-light.agency -c config*

To parse sitemap.xml for extra urls to parse add *-p*: *seoaudit -u https://green-light.agency -p*

1.1 Custom config

TODO: add custom config definition

1.2 Extending predefined checks

TODO: Extending predefined checks

CHAPTER

TWO

API REFERENCE

2.1 SEO Auditer

2.2 Site Parser

This module contains SiteParser class which defines a parser at website level (list of urls).

Typical usage example:

site_parser = SiteParser(url, LXMLPageParser(url), urls=None, parse_sitemap_urls=True) while site_parser.parse_next_page():

print("Running checks for url: {}".format(site_parser.get_current_url())) # do something

class seoaudit.analyzer.site_parser.**SiteParser**(*base_url*, *page_parser: seoaudit.analyzer.page_parser.AbstractPageParser = None*, *urls=None*, *sitemap_link=None*, *parse_sitemap_urls=False*)

Website level parser, uses a page parser object as the core of it's parsing functionalities with the urls list being predefined or crawled from the sitemap file.

get_current_url()

Returns url of currently indexed page

Return type str

parse_next_page()

Parse next page using page parser object.

Returns True if next page was parse, False if end of list of urls was reached

Return type boolean

2.3 Pager Parser

This module contains page parses classes which define page parser objects at single web page level (single url).

Typical usage example:

page_parser = PageParser(url) sitemap_links = page_parser.get_elements("(/html/head/link[@rel='sitemap'])/@href") sitemap_link = sitemap_link[0] if len(sitemap_links) >= 1 else None

class seoaudit.analyzer.page_parser.**AbstractPageParser**(*url*)
Abstract web page parser. Used as a blueprint for page parser implementations.

abstract get_element_attribute(*element*, *attribute='textContent'*) → str
Given an HTML element and its attribute name, return attributes content.

Parameters

- **element** – HTML element
- **attribute** – attribute name, defaults to textContent

Returns HTML element's attribute text value

abstract get_element_code(*element*) → str
Given an HTML element return its HTML code.

Parameters **element** – HTML element

Returns string HTML code

abstract get_element_text(*element*) → str
Given an HTML element return its text content.

Parameters **element** – HTML element

Returns string text content

abstract get_elements(*xpath_query: str*)
Get a list of HTML elements using xpath query on page parsed web page.

Parameters **xpath_query** (*str*) – xpath elements query

Returns list of HTML elements that can be used in other parser methods

class seoaudit.analyzer.page_parser.**LXMLPageParser**(*url*)
Web page parser with lxml core.

get_element_attribute(*element: lxml.html.HtmlElement*, *attribute='textContent'*) → str
Given an HTML element and its attribute name, return attributes content.

Parameters

- **element** – HTML element of lxml HtmlElement type
- **attribute** – attribute name, defaults to textContent

Returns HTML element's attribute text value

get_element_code(*element*) → str
Given an HTML element return its HTML code.

Parameters **element** (*HtmlElement*) – HTML element of lxml HtmlElement type

Returns string HTML code

get_element_text(*element*) → str
Returns visible text of HTML element. If string HTML element is passed it returns it. This makes this function able to be iteratively called on page parser elements even if they are returned as a mix of HtmlElements and str.

Parameters

- **element** (*HtmlElement | str*) – HTML element of lxml HtmlElement type which has method text_content() or a string
- **of HTML element** (*representation*) –

Returns string text content

get_elements(*xpath_query: str*)
Get a list of HTML elements using xpath query on page parsed web page.

Parameters **xpath_query** (*str*) – xpath elements query

Returns: a list of lxml HtmlElement elements

class seoaudit.analyzer.page_parser.**SeleniumPageParser**(*url*)
Web page parser with Selenium Webdriver core.

get_element_attribute(*element: selenium.webdriver.remote.webelement.WebElement, attribute='textContent'*) → str
Given an HTML element and its attribute name, return attributes content.

Parameters

- **element** – HTML element of Selenium WebElement type
- **attribute** – attribute name, defaults to textContent

Returns HTML element's attribute text value

get_element_code(*element*) → str
Given an HTML element return its HTML code.

Parameters **element** (*WebElement*) – HTML element of Selenium WebElement type

Returns string HTML code

get_element_text(*element*) → str
Returns visible text of HTML element. If string HTML element is passed it returns it. This makes this function able to be iteratively called on page parser elements even if they are returned as a mix of WebElements and str.

Parameters

- **element** (*WebElement | str*) – HTML element of Selenium WebElement type which has attribute text or a string
- **of HTML element** (*representation*) –

Returns string text content

get_elements(*xpath_query: str*)
Get a list of HTML elements using xpath query on page parsed web page.

Parameters **xpath_query** (*str*) – xpath elements query

Returns: a list of selenium Webdriver elements

2.4 Element Checks

This modules contains all of the predefined element checks. Element check works at single DOM element level.

Predefined element checks are enumerated in ElementCheck enum with each enum value containing the name of the class that implements the defined check by extending AbstractElementCheck class.

When functionality of predefined element checks is not enough, custom ElementCheck can be created by extending AbstractElementCheck class.

Typical usage example:

content = “abc” check = check_content(ElementCheck.MIN_LENGTH, “abc”, 2) # check = True check = check_content(ElementCheck.MIN_LENGTH, “abc”, 4) # check = False

class seoaudit.checks.element.**AbstractElementCheck**
Abstract class that serves as a blueprint for element check classes.

abstract check_content(*content: str*, ***kwargs*)
Returns check validity of the given element.

Parameters

- **content** (*str*) – element content value on which check is performed
- ***kwargs** – keyword check arguments (e.g. a number representing a minimal length value)

Returns a boolean value representing checks validity preceded by any extra check result information

class seoaudit.checks.element.**AttributeFoundCheck**
Checks if content attribute is found and not empty.

check_content(*content: str*, ***unused*)

Parameters

- **content** – element content value on which check is performed
- **unused** – unused parameter defined to extend AbstractElementCheck

Returns boolean check result

class seoaudit.checks.element.**ElementCheck**
Enum representing all of the predefined element check types.

class seoaudit.checks.element.**MaxLengthCheck**
Check if content length is smaller or equal to maximal length..

check_content(*content: str*, ***kwargs*)

Parameters

- **content** – element content value on which check is performed
- **kwargs** – keyword arguments (map) that includes ‘max_length’ parameter which defaults to 0 if not defined

Returns tuple including boolean check result and content length

Return type Tuple(boolean, int)

class seoaudit.checks.element.**MinLengthCheck**
Check if content length is bigger or equal to minimal length.

check_content(*content: str*, ***kwargs*)

Parameters

- **content** – element content value on which check is performed
- **kwargs** – keyword arguments (map) that includes ‘min_length’ parameter which defaults to 0 if not defined

Returns tuple including boolean check result and content length

Return type Tuple(boolean, int)

class seoaudit.checks.element.**RegexMatchCheck**
Implements content regex match check.

check_content(*content: str*, ***kwargs*)

Parameters

- **content** – element content value on which check is performed
- **kwargs** – keyword argument (map) that includes 'regex' parameter which defaults to '.*' if not defined

Returns tuple including boolean check result and content length

Return type Tuple(boolean, int)

seoaudit.checks.element.**check_content**(*check: seoaudit.checks.element.ElementCheck*, *content: str*, ***kwargs*)
Wrapper function to perform a check defined by the given ElementCheck.

Parameters

- **check** (ElementCheck) – Enum identifying type of the check
- **content** (*str*) – element content value on which check is performed
- ***kwargs** – various content check arguments (e.g. a number representing a minimal length value)

Returns a boolean value representing checks validity preceded by any extra check result information

2.5 Page Checks

2.6 Site Checks

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX

R

S