IndieWeb Utils

Release 0.5.0

capjamesg

Oct 13, 2022

CONTENTS

1 Contents **3**

1.1 Discovery . . . 3
1.2 URLs . . . 10
1.3 Generate Reply Context . . . 12
1.4 Generate a URL Summary . . . 13
1.5 IndieAuth . . . 14
1.6 Webmention . . . 24
1.7 Changelog . . . 26

2 Feature Set **31**

Index **33**

indieweb-utils is a Python library that provides building blocks for people implementing IndieWeb applications. This library also contains implementations for some specifications that may be useful in IndieWeb applications.

To install IndieWeb Utils, use this command:

```
pip3 install indieweb-utils
```

You can import the package using the following line of code:

```
import indieweb_utils
```

Below, you will find links to all of the methods available in IndieWeb Utils.

CHAPTER

ONE

CONTENTS

1.1 Discovery

Indieweb utils provides a number of functions to help you determine properties from webpages.

1.1.1 Discover IndieWeb endpoints

The *discover_endpoints()* function parses HTTP Link headers and HTML <*link*> tags to find the specified endpoints.

indieweb_utils.**discover_endpoints**(*url: str*, *headers_to_find: List[str]*, *request: Optional[Response] = None*)

Return a dictionary of specified endpoint locations for the given URL, if available.

Parameters

- **url** (*str*) – The URL to discover endpoints for.
- **headers_to_find** (*dict[str, str]*) – The headers to find. Values you may want to use include: microsub, micropub, token_endpoint, authorization_endpoint, subscribe.

Returns

The discovered endpoints.

Return type

dict[str, str]

```
import indieweb_utils
import requests

url = "https://jamesg.blog/"

# find the microsub rel link on a web page
headers_to_find = ["microsub"]

endpoints = indieweb_utils.discover_endpoints(
    url
)

print(endpoints) # {'aperture': 'https://aperture.p3k.io/'}
```

Raises

requests.exceptions.RequestException – Error raised while making the network request to discover endpoints.

This function only returns the specified endpoints if they can be found. It does not perform any validation to check that the discovered endpoints are valid URLs.

We recommend using the discover_webmention_endpoint function to discover webmention endpoints as this performs additional validation useful in webmention endpoint discovery.

1.1.2 Find an article author

You can discover the original author of an article as per the Authorship Specification.

To do so, use this function:

indieweb_utils.**discover_author**(*url: str*, *html: str = ''*, *parsed_mf2: Optional[Parser] = None*) → dict

Discover the author of a post per the IndieWeb Authorship specification.

Refs
https://indieweb.org/authorship-spec

Parameters

- **url** (*str*) – The URL of the post.
- **page_contents** (*str*) – The optional page contents to use. Specifying this value prevents a HTTP request being made to the URL.

Returns
A h-card of the post.

Return type
dict

```
import indieweb_utils
import mf2py

url = "https://jamesg.blog/2022/01/28/integrated-indieweb-services/"

parsed_mf2 = mf2py.parse(url=url)

post_author = indieweb_utils.discover_author(
    h_entry
)

print(post_author) # A h-card object representing the post author.
```

Here are the arguments you can use:

- *url*: The URL of the web page whose author you want to discover.
- *page_contents*: The unmodified HTML of a web page whose author you want to discover.

The page_contents argument is optional.

If no page_contents argument is specified, the URL you stated is retrieved. Then, authorship inference begins.

If you specify a page_contents value, the HTML you parsed is used for authorship discovery. This saves on a HTML request if you have already retrieved the HTML for another reason (for example, if you need to retreive other values in the page HTML). You still need to specify a URL even if you specify a page_contents value.

The discover_author function can return one of two values:

- An author name (i.e. “James”).

- The h-card of an author.

These are the two outputs defined in the authorship inference algorithm. Your program should be able to handle both of these outputs.

This code returns the following h-card:

```
{
    'type': ['h-card'],
    'properties': {
        'url': ['https://aaronparecki.com/'],
        'name': ['Aaron Parecki'],
        'photo': ['https://aaronparecki.com/images/profile.jpg']
    },
    'value': 'https://aaronparecki.com/'
}
```

1.1.3 Find a post type

To find the post type associated with a web page, you can use the *get_post_type* function.

The *get_post_type* function function uses the following syntax:

indieweb_utils.**get_post_type**(*h_entry: dict = {}*, *custom_properties: List[Tuple[str, str]] = []*) → str

Return the type of a h-entry per the Post Type Discovery algorithm.

Parameters

- **h_entry** (*dict*) – The h-entry whose type to retrieve.
- **custom_properties** (*list[tuple[str, str]]*) – The optional custom properties to use for the Post Type Discovery algorithm.

Returns

The type of the h-entry.

Return type

str

Here is an example of the function in action:

```
import indieweb_utils
import mf2py

url = "https://jamesg.blog/2022/01/28/integrated-indieweb-services/"

parsed_mf2 = mf2py.parse(url=url)

h_entry = [e for e in parsed_mf2["items"] if e["type"] == ["h-entry"]][0]

post_type = indieweb_utils.get_post_type(
    h_entry
)

print(post_type) # article
```

Raises
PostTypeFormattingError – Raised when you specify a custom_properties tuple in the wrong format.

This function returns a single string with the post type of the specified web page.

See the Post Type Discovery specification for a full list of post types.

1.1.4 Custom Properties

The structure of the custom properties tuple is:

```
(attribute_to_look_for, value_to_return)
```

An example custom property value is:

```
custom_properties = [
    ("poke-of", "poke")
]
```

This function would look for a poke-of attribute on a web page and return the "poke" value.

By default, this function contains all of the attributes in the Post Type Discovery mechanism.

Custom properties are added to the end of the post type discovery list, just before the "article" property. All specification property types are checked before your custom attribute.

1.1.5 Find the original version of a post

To find the original version of a post per the Original Post Discovery algorithm, use this code:

indieweb_utils.**discover_original_post**(*posse_permalink: str*, *soup: Optional[BeautifulSoup] = None*, *html: str = ''*) → str

Find the original version of a post per the Original Post Discovery algorithm.

refs: https://indieweb.org/original-post-discovery#Algorithm

Parameters
posse_permalink (*str*) – The permalink of the post.

Returns
The original post permalink.

Return type
str

Example:

```
import indieweb_utils

original_post_url = indieweb_utils.discover_original_post("https://example.com")

print(original_post_url)
```

Raises
PostDiscoveryError – A candidate URL cannot be retrieved or when a specified post is not marked up with h-entry.

This function returns the URL of the original version of a post, if one is found. Otherwise, None is returned.

1.1.6 Find all feeds on a page

To discover the feeds on a page, use this function:

indieweb_utils.**discover_web_page_feeds**(*url: str*, *user_mime_types: Optional[List[str]] = None*, *html: str = ''*) → List[*FeedUrl*]

Get all feeds on a web page.

Parameters

- **url** (*str*) – The URL of the page whose associated feeds you want to retrieve.
- **user_mime_types** (*Optional[List[str]]*) – A list of mime types whose associated feeds you want to retrieve.
- **html** (*str*) – A string with the HTML on a page.

Returns
A list of FeedUrl objects.

Return type
List[*FeedUrl*]

Example:

```
import indieweb_utils

url = "https://jamesg.blog/"

feeds = indieweb_utils.discover_web_page_feeds(url)

# print the url of all feeds to the console
for f in feeds:
    print(f.url)
```

This function returns a list with all feeds on a page.

Each feed is structured as a FeedUrl object. FeedUrl objects contain the following attributes:

class indieweb_utils.**FeedUrl**(*url: str*, *mime_type: str*, *title: str*)

1.1.7 Get a Representative h-card

To find the h-card that is considered representative of a web resource per the Representative h-card Parsing Algorithm, use the following function:

indieweb_utils.**get_representative_h_card**(*url: str*, *html: str = ''*, *parsed_mf2: Optional[Parser] = None*) → Dict[str, Any]

Get the representative h-card on a page per the Representative h-card Parsing algorithm.

refs: https://microformats.org/wiki/representative-h-card-parsing

Url
The url to parse.

Returns
The representative h-card.

Return type
dict

Example:

```
import indieweb_utils

url = "https://jamesg.blog/"

h_card = indieweb_utils.get_representative_h_card(url)

print(h_card) # {'type': ['h-card'], 'properties': {...}}
```

Raises
RepresentativeHCardParsingError – Representative h-card could not be parsed.

This function returns a dictionary with the h-card found on a web page.

1.1.8 Get a Page h-feed

The *discover_h_feed()* function implements the proposed microformats2 h-feed discovery algorithm.

This function looks for a h-feed on a given page. If one is not found, the function looks for a rel tag to a h-feed. If one is found, that document is parsed.

If a h-feed is found on the related document, the h-feed is returned.

This function returns a dictionary with the h-card found on a web page.

indieweb_utils.**discover_h_feed**(*url: str*, *html: str = ''*) → Dict

Find the main h-feed that represents a web page as per the h-feed Discovery algorithm.

refs: https://microformats.org/wiki/h-feed#Discovery

Parameters

- **url** (*str*) – The URL of the page whose associated feeds you want to retrieve.
- **html** (*str*) – The HTML of a page whose feeds you want to retrieve

Returns
The h-feed data.

Return type
dict

Example:

```
import indieweb_utils

url = "https://jamesg.blog/"

hfeed = indieweb_utils.discover_h_feed(url)

print(hfeed)
```

1.1.9 Infer the Name of a Page

To find the name of a page per the Page Name Discovery Algorithm, use this function:

indieweb_utils.**get_page_name**(*url: str*, *html: Optional[str] = None*, *soup: Optional[BeautifulSoup] = None*) → str

Retrieve the name of a page using the Page Name Discovery algorithm.

Refs
https://indieweb.org/page-name-discovery

Parameters

- **url** (*str*) – The url of the page whose title you want to retrieve.
- **html** (*str*) – The HTML of the page whose title you want to retrieve.

Returns
A representative "name" for the page.

Return type
str

Example:

```
import indieweb_utils

page_name = indieweb_utils.get_page_name("https://jamesg.blog")

print(page_name) # "Home | James' Coffee Blog"
```

This function searches:

1. For a h-entry title. If one is found, it is returned;
2. For a h-entry summary. If one is found, it is returned;
3. For a HTML page <title> tag. If one is found, it is returned;

Otherwise, this function returns "Untitled".

1.1.10 Get all URLs a Post Replies To

To find all of the URLs to which a reply post is replying, use this function:

indieweb_utils.**get_reply_urls**(*url: str*, *html: Optional[str] = None*) → List[str]

Retrieve a list of all of the URLs to which a given post is responding using a u-in-reply-to microformat.

Refs
https://indieweb.org/in-reply-to#How_to_consume_in-reply-to

Parameters

- **url** (*str*) – The URL to get replies to.
- **html** (*str*) – The HTML of the page whose replies you want to retrieve.

Returns
A list of all of the URLs to which the given post responds.

Return type
list

Example:

```
import indieweb_utils

reply_urls = indieweb_utils.get_reply_urls("https://aaronparecki.com/2022/10/10/17/
↪")

print(reply_urls) # ["https://twitter.com/amandaljudkins/status/1579680989135384576?
↪s=12"]
```

1.2 URLs

1.2.1 Canonicalize a URL

Canonicalization turns a relative URL into a complete URL.

To canonicalize a URL, use this function:

indieweb_utils.**canonicalize_url**(*url: str*, *domain: str = ''*, *full_url: str = ''*, *protocol: str = 'https'*) → str

Return a canonical URL for the given URL.

Parameters

- **url** (*str*) – The URL to canonicalize.
- **domain** (*str*) – The domain to use for the canonical URL.
- **full_url** (*str or None*) – Optional full URL to use for the canonical URL.
- **protocol** (*str or None*) – Optional protocol to use for the canonical URL.

Returns

The canonical URL.

Return type

str

```
import indieweb_utils

url = "/contact"
domain = "jamesg.blog"
protocol = "https"

endpoints = indieweb_utils.canonicalize_url(
    url, domain, protocol=protocol
)

print(webmention_endpoint) # https://jamesg.blog/contact/
```

This function returns a URL with a protocol, host, and path.

The domain of the resource is needed so that it can be added to the URL during canonicalization if the URL is relative.

A complete URL returned by this function looks like this:

```
https://indieweb.org/POSSE
```

1.2.2 Add hashtags and person tags to a string

The *autolink_tags()* function replaces hashtags (#) with links to tag pages on a specified site. It also replaces person tags (ex. @james) with provided names and links to the person's profile.

This function is useful for enriching posts.

To use this function, pass in the following arguments:

`indieweb_utils.`**`autolink_tags`**(*text: str*, *tag_prefix: str*, *people: dict*, *tags: Optional[List[str]] = None*) → str

Replace hashtags (#) and person tags (@) with links to the respective tag page and profile URL.

Parameters

- **text** (*str*) – The text to process.
- **tag_prefix** (*str*) – The prefic to append to identified tags.
- **people** (*dict*) – A dictionary of people to link to.
- **tags** (*List[str]*) – A list of tags to link to (optional).

Returns

The processed text.

Return type

str

Example:

```
import indieweb_utils

note = "I am working on a new #muffin #recipe with @jane"

# tag to use, name of person, domain of person
people = { "jane": ("Jane Doe", "https://jane.example.com") }

note_with_tags = indieweb_utils.autolink_tags(note, "/tag/", people, tags=["muffin",
↪ "recipe"])
```

This function will only substitute tags in the list of tags passed through to this function if a *tags* value is provided. This ensures that the function does not create links that your application cannot resolve.

If you do not provide a *tags* value, the function will create links for all hashtags, as it is assumed that your application can resolve all hashtags.

Tagging people is enabled by providing a dictionary with information on all of the people to whom you can tag.

If a person in an @ link is not in the tag dictionary, this function will not substitute that given @ link.

Here is an example value for a person tag database:

```
{
    "james": (
        "James' Coffee Blog",
        "https://jamesg.blog/""
    )
}
```

This function maps the *@james* tag with the name "James' Coffee Blog" and the URL "https://jamesg.blog/". More people can be added as keys to the dictionary.

1.3 Generate Reply Context

To generate reply context for a given page, use the following function:

indieweb_utils.**get_reply_context**(*url: str*, *twitter_bearer_token: str = ''*, *summary_word_limit: int = 75*) → *ReplyContext*

Generate reply context for use on your website based on a URL.

param url
The URL of the post to generate reply context for.

type url
str

param twitter_bearer_token
The optional Twitter bearer token to use. This token is used to retrieve a Tweet from Twitter's API if you want to generate context using a Twitter

URL.

type twitter_bearer_token
str

param summary_word_limit
The maximum number of words to include in the summary (default 75).

type summary_word_limit
int

return
A ReplyContext object with information about the specified web page.

rtype
ReplyContext

Example:

```
import indieweb_utils

context = indieweb_utils.get_reply_context(
    url="https://jamesg.blog",
    summary_word_limit=50
)

# print the name of the specified page to the console
print(context.name) # "Home | James' Coffee Blog"
```

raises ReplyContextRetrievalError
Reply context cannot be retrieved.

raises UnsupportedScheme
The specified URL does not use http:// or https://.

This function returns a ReplyContext object that looks like this:

class indieweb_utils.**ReplyContext**(*webmention_endpoint: str*, *photo: str*, *name: str*, *video: str*, *post_html: str*, *post_text: str*, *authors: List[PostAuthor]*, *description: str*)

Context about a web page and its contents.

1.4 Generate a URL Summary

You can generate a summary of a URL without retrieving the page using the *get_url_summary* function.

By default, this function can generate a summary for the following URLs:

- github.com
- twitter.com
- eventbrite.com / eventbrite.co.uk
- upcoming.com
- calagator.com
- events.indieweb.org
- indieweb.org

indieweb_utils.**get_url_summary**(*url: str*, *custom_templates: Optional[list] = None*)

Return a text summary for given *url*.

Parameters

- **url** – The URL to summarize.
- **custom_templates** – A list of tuples with patterns against which to check when generating a summary associated with results to return.

Returns

A summary of the URL.

Return type

str

```
import indieweb_utils

# a dictionary of custom patterns against which to match during the lookup
custom_properties = {
    "jamesg.blog": [
        (r"coffee/maps/(?P<location>.+)", "A map of {location} coffee shops on␣
↪jamesg.blog")
    ]
}

summary = indieweb_utils.get_summary("https://github.com/capjamesg/indieweb-utils/
↪pulls/1")

print(summary) # "A comment on a pull request in the indieweb-utils GitHub␣
↪repository"

summary = indieweb_utils.get_summary("https://jamesg.blog/coffee/maps/london")

print(summary) # "A map of london coffee shops on jamesg.blog"
```

You can specify custom mappings for other domains using the *custom_mappings* parameter.

This parameter accepts a dictionary of with domain names mapped to lists of tuples with patterns to match and strings to return, like this:

```
{
    "example.com": [
        (r"example.com/(\d+)", "Example #{}"),
    ]
}
```

If a summary cannot be generated, this function returns "A post by [domain_name].", where domain name is the domain of the URL you passed into the function.

1.5 IndieAuth

The indieweb-utils library provides a number of helper functions that will enable you to implement your own IndieAuth authentication and token endpoints in Python.

These functions may be useful if you want to bundle an authentication and token provider with a service you are building, such as a personal blog or a social reader.

This page outlines how to use the IndieAuth features provided in this library.

1.5.1 Get application scope reference

The library comes with a constant variable that lists various scopes that are commonly used in IndieWeb community applications. These scopes cover values you might see in Micropub, Microsub, and other applications.

You may want to use this reference to give a user additional information about the scopes to which they are granting access by authenticating with a service.

To access the scope reference, import the following variable:

```
from indieweb_utils import SCOPE_DEFINITIONS
```

1.5.2 Get a h-app object

You might want to retrieve a h-app object to show context to the user about the application that is requesting a user to authenticate.

You can do this using the following function:

indieweb_utils.**get_h_app_item**(*web_page: str*) → ApplicationInfo

Get the h-app item from the web page.

Parameters

- **web_page** (*str*) – The web page to parse.
- **client_id** (*str*) – The client id of your application.

Returns

The h-app item.

Return type

ApplicationInfo

Example:

```
import indieweb_utils

app_url = "https://quill.p3k.io/"
client_id = "https://quill.p3k.io/"

h_app_item = indieweb_utils.get_h_app_item(
    app_url, client_id
)

print(h_app_item.name) # Quill
```

Raises
HAppNotFound – Raised if no mf2 h-app data was found on the specified page.

This function returns an object with the name, logo, url, and summary found in a h-app item.

Note: The h-app item is not widely supported. As a result, you might want to add a fallback in the case that the h-app function does not find any values to return.

1.5.3 Get a profile response object

If you want your IndieAuth object to return profile information when the "profile" scope is requested, the get_profile() function may come in handy.

This function takes a URL and retrieves the name, photo, url, and email properties found on the h-card of the specified page. If a h-card is not provided, empty strings are returned.

Usage information for this function is below.

indieweb_utils.**get_profile**(*me: str*, *html: str = ''*, *soup: Optional[BeautifulSoup] = None*) → Profile

Return the profile information for the given me URL.

Parameters
me (*str*) – The me URL to get the profile information for.

Returns
The profile information.

Return type
Profile

Get a profile from a url.

Parameters
me (*str*) – The url to get the profile from.

Returns
The profile.

Return type
Profile

Example:

```
import indieweb_utils

me = "https://jamesg.blog"
```

(continues on next page)

(continued from previous page)

```
profile = indieweb_utils.get_profile(me)

assert profile.email == "james@jamesg.blog"
assert profile.name == "James"
assert profile.photo == "https://jamesg.blog/me.jpg"
assert profile.url == "https://jamesg.blog
```

Raises

ProfileError – Profile could not be retrieved.

1.5.4 Retrieve Valid Links for Use in RelMeAuth

To authenticate a user with RelMeAuth, you need to validate that there is a two-way link between two resources.

IndieWeb Utils implements a helper function that checks whether the URLs linked with rel=me on a web page contain a link back to the source.

To check whether there is a two-way rel=me link between two resources, you can use this function:

indieweb_utils.**get_valid_relmeauth_links**(*url: str*, *require_rel_me_link_back: bool = True*, *html: Optional[str] = None*, *parsed_mf2: Optional[Parser] = None*) → List[str]

Get the valid links on a page that point back to a rel=me URL per RelMeAuth.

refs: https://indieweb.org/RelMeAuth refs: http://microformats.org/wiki/RelMeAuth

Url

The url to parse.

Require_rel_me_link_back

Whether to require a rel=me link back to the specified URL. If this property is set to False, this function will return all sites that link back to the specified URL, even if they do not have a rel=me attribute. If this property is set to True (the default), this function will only return sites that have a rel=me link pointing back to your URL.

Returns

The valid relmeauth links.

Return type

dict

Example:

```
import indieweb_utils

url = "https://jamesg.blog"

valid_relmeauth_links = indieweb_utils.get_valid_relmeauth_links(url)

for link in valid_relmeauth_links:
    print(link)
```

This function does not check whether a URL has an OAuth provider. Your application should check the list of valid rel me links and only use those that integrate with the OAuth providers your RelMeAuth service supports. For example, if

your service does not support Twitter, you should not present Twitter as a valid authentication option to a user, even if the *get_valid_relmeauth_links()* function found a valid two-way rel=me link.

1.5.5 IndieAuth Endpoint Scaffolding

indieweb-utils includes a *indieauth.server* module with scaffolding to help you build your own IndieAuth endpoints.

Generate an authentication token

The *generate_auth_token()* function validates that an authentication request contains all required values. Then, this function generates a JWT-encoded token with the following pieces of information:

- me
- code_verifier
- expires
- client_id
- redirect_uri
- scope
- code_challenge
- code_challenge_method

You can later refer to these values during the stage where you decode a token.

Here is the syntax for this function:

`indieweb_utils.indieauth.server.`**`generate_auth_token`**(*me: str*, *client_id: str*, *redirect_uri: str*, *response_type: str*, *state: str*, *code_challenge_method: str*, *final_scope: str*, *secret_key: str*, ***kwargs*) → AuthTokenResponse

Generates an IndieAuth authorization token.

Parameters

- **me** (*str*) – The URL of the user's profile.
- **client_id** (*str*) – The client ID of the authorization request.
- **redirect_uri** (*str*) – The redirect URI of the authorization request.
- **response_type** (*str*) – The response type of the authorization request.
- **state** (*str*) – The state of the authorization request.
- **code_challenge_method** (*str*) – The code challenge method, used for PKCE.
- **final_scope** (*str*) – The scopes approved by the user.
- **secret_key** (*str*) – The secret key used to sign the token.
- **kwargs** (*dict*) – Additional parameters to include in the token.

Returns

The authorization token.

Return type

str

Example:

```
import indieweb_utils
import random
import string
token = indieweb_utils.indieauth.server.generate_auth_token(
    me="https://test.example.com/user",
    client_id="https://example.com",
    redirect_uri="https://example.com/callback",
    response_type="code",
    state="".join(random.choice(string.ascii_letters) for _ in range(32)),
    code_challenge_method="S256",
    final_scope="read write",
    secret_key="secret"
)
```

Raises

AuthenticationError – Authentication request is invalid.

This function returns both the code you should send to the client in the authentication redirect response as well as the code_verifier used in the token. This code_verifier should be saved, perhaps in session storage, for later use in checking the validity of a token redemption request.

Validate an authorization response

The *_validate_indieauth_response* function contains five checks. These five checks validate an authorization response according to the IndieAuth specification.

1. Ensures the grant_type provided is authorization_code.
2. Validates that a code, client_id, and redirect_uri are provided.
3. Checks that the code challenge method provided is allowed.
4. Verifies the length of the code challenge is within the range of 43 and 128 characters.

You should use this function to ensure a POST request to an authorization endpoint to redeem an authorization code (per 5.3 Redeeming the Authorization Code in the IndieAuth spec) is valid.

Here is the syntax for this function:

indieweb_utils.**validate_authorization_response**(*grant_type: str*, *code: str*, *client_id: str*, *redirect_uri: str*, *code_challenge: str*, *code_challenge_method: str*, *allowed_methods: list = ['S256']*) → None

Conducts checks to validate the response from an IndieAuth authorization endpoint.

Parameters

- **grant_type** (*str*) – The grant type of the authorization request.
- **code** (*str*) – The code returned from the authorization request.
- **client_id** (*str*) – The client ID of the authorization request.
- **redirect_uri** (*str*) – The redirect URI of the authorization request.
- **code_challenge** (*str*) – The code challenge, used for PKCE.
- **code_challenge_method** (*str*) – The code challenge method, used for PKCE.

- **allowed_methods** (*list*) – The list of allowed code challenge methods (default: ["S256"]).

Returns
A boolean indicating whether the response is valid.

Return type
bool

Example:

```
import indieweb_utils

indieweb_utils.validate_authorization_response(
    grant_type="authorization_code",
    code="12345",
    client_id="https://example.com",
    redirect_uri="https://example.com/callback",
    code_challenge="12345",
    code_challenge_method="S256",
    allowed_methods=["S256"]
)
```

Raises
TokenValidationError – If the response is invalid.

This function does not return a value if an authorization response is valid. If a response is invalid, an exception will be raised with a relevant error message.

Redeem an IndieAuth code at a token endpoint

You can redeem an IndieAuth authorization code for an access token if needed. This is a common need for Micropub and Microsub clients.

The *redeem_code()* function validates all the required parameters are provided in a request. Then, this function decodes the provided code using the code, secret key, and algorithm specified. If the code is invalid or the code challenge in the decoded code is invalid, AuthenticationErrors will be raised. The function will also verify that:

1. An authorization code has not expired.
2. The specified redirect URI matches the decoded redirect URI.
3. The specified client ID matches the decoded client ID.

Finally, this function encodes an access token that you can return in response to a request to create a token for a token endpoint.

Here is the syntax for this function:

indieweb_utils.**redeem_code**(*grant_type: str*, *code: str*, *client_id: str*, *redirect_uri: str*, *code_verifier: str*, *secret_key: str*, *algorithms: list = ['HS256']*, ***kwargs*) → TokenEndpointResponse

Redeems an IndieAuth code for an access token.

Parameters

- **grant_type** (*str*) – The grant type of the authorization request.
- **code** (*str*) – The code returned from the authorization request.
- **client_id** (*str*) – The client ID of the authorization request.

- **redirect_uri** (*str*) – The redirect URI of the authorization request.
- **code_verifier** (*str*) – The code verifier, used for PKCE.
- **secret_key** (*str*) – The secret key used to sign the token.
- **algorithms** (*list*) – The list of algorithms to use for signing the token (default: ["HS256"]).
- **kwargs** (*dict*) – Additional parameters to include in the token.

Returns
A token endpoint response object.

Return type
TokenEndpointResponse

Example:

```
import indieweb_utils

token_response = indieweb_utils.indieauth.server.redeem_code(
    grant_type="authorization_code",
    code="code",
    client_id="https://example.com",
    redirect_uri="https://example.com/callback",
    code_verifier="code_verifier",
    secret_key="secret"
)

print(token_response.access_token)
print(token_response.token_type)
print(token_response.scope)
print(token_response.me)
```

Raises

- **AuthorizationCodeExpiredError** – If the authorization code has expired.
- **TokenValidationError** – If the decoded code is invalid.
- **AuthenticationError** – If the token request is invalid.

Validate an access token created by a token endpoint

A server may ask your token endpoint to validate that a provided token is in fact valid. This may be done by a Micropub client to ensure a token is still active, for example.

You can validate an access token using the *validate_access_token()* function.

This function decodes an authorization code using the specified secret key and algorithm(s). Then, the function will check that the authorization code has not yet expired.

If the code can be decoded, the me, client_id, and scope values will be returned.

Here is the syntax for the function:

indieweb_utils.**validate_access_token**(*authorization_code: str*, *secret_key: str*, *algorithms: list = ['HS256']*) → DecodedAuthToken

Validates an access token provided by a token endpoint.

Parameters

- **authorization_code** (*str*) – The authorization code returned from the authorization request.
- **secret_key** (*str*) – The secret key used to sign the token.
- **algorithms** (*list*) – The algorithms used to sign the token (default: ["HS256"]).

Returns
An object with the me, client_id, and scope values from the access token.

Return type
DecodedAuthToken

Example:

```
import indieweb_utils

try:
    decoded_token = indieweb_utils.indieauth.server.validate_access_token(
        authorization_code="code",
        secret_key="secret"
    )

    print(decoded_token.me)
    print(decoded_token.client_id)
    print(decoded_token.scope)
    print(decoded_token.decoded_authorization_code)
except indieweb_utils.AuthenticationError as e:
    print(e)
except indieweb_utils.AuthorizationCodeExpiredError as e:
    print(e)
```

Raises

- **AuthorizationCodeExpiredError** – Authorization code has expired.
- **AuthenticationError** – Authorization code provided is invalid.

Determine if a user is authenticated

To check if a user is authenticated in a Flask application, use the following function:

indieweb_utils.**is_authenticated**(*token_endpoint: str*, *headers: dict*, *session: dict*, *approved_user: Optional[bool] = None*) → bool

Check if a user has provided a valid Authorization header or access token in session. Designed for use with Flask.

Parameters

- **token_endpoint** – The token endpoint of the user's IndieAuth server.
- **headers** – The headers sent by a request.
- **session** – The session object from a Flask application.
- **approved_user** – The optional URL of the that is approved to use the API.

Returns
True if the user is authenticated, False otherwise.

Return type
bool

Example:

```
import indieweb_utils
from Flask import flask, request

app = Flask(__name__)

@app.route("/")
def index():
    user_is_authenticated = indieweb_utils.is_authenticated(
        "https://tokens.indieauth.com/token",
        request.headers,
        session,
        "https://example.com/",
    )

    if user_is_authenticated is False:
        return "Not authenticated"

    return "Authenticated"
```

Raises
AuthenticationError – The token endpoint could not be accessed.

This function checks if an authorization token is provided in a header or user storage. If a token is provided, that token is verified with the specified token endpoint.

A True value is returned if a user has provided a token and that token is valid. A False value is returned if a user has not provided a token or if the token is invalid.

Handle an IndieAuth callback request

The last stage of the IndieAuth authentication flow for a client is to verify a callback response and exchange the provided code with a token.

This function implements a callback handler to verify the response frmo an authorization server and redeem a token.

To use this function, you need to pass in the following arguments:

indieweb_utils.**indieauth_callback_handler**(*, *code: str*, *state: str*, *token_endpoint: str*, *code_verifier: str*, *session_state: str*, *me: str*, *callback_url: str*, *client_id: str*, *required_scopes: List[str]*) → IndieAuthCallbackResponse

Exchange a callback 'code' for an authentication token.

Parameters

- **code** (*str*) – The callback 'code' to exchange for an authentication token.
- **state** (*str*) – The state provided by the authentication server in the callback response.
- **token_endpoint** (*str*) – The token endpoint to use for exchanging the callback 'code' for an authentication token.

- **code_verifier** (*str*) – The code verifier to use for exchanging the callback 'code' for an authentication token.
- **session_state** (*str*) – The state stored in session used to verify the callback state is valid.
- **me** (*str*) – The URL of the user's profile.
- **callback_url** (*str*) – The callback URL used in the original authentication request.
- **client_id** (*str*) – The client ID used in the original authentication request.
- **required_scopes** (*list[str]*) – The scopes required for the application to work. This list should not include optional scopes.

Returns
A message indicating the result of the callback (success or failure) and the token endpoint response. The endpoint response will be equal to None if the callback failed.

Return type
tuple[str, dict]

Example:

```
import indieweb_utils
from Flask import flask

app = Flask(__name__)

@app.route("/indieauth/callback")
def callback():
    response = indieweb_utils.indieauth_callback_handler(
        code=request.args.get("code"),
        state=request.args.get("state"),
        token_endpoint="https://tokens.indieauth.com/token",
        code_verifier=session["code_verifier"],
        session_state=session["state"],
        me=session["me"],
        callback_url=session["callback_url"],
        client_id=session["client_id"],
        required_scopes=["create", "update", "delete"],
    )

    return response.message
```

Raises
AuthenticationError – The token endpoint could not be accessed or authentication failed.

This function verifies that an authorization server has returned a valid response and redeems a token.

You can leave the "me" value equal to None if any URL should be able to access your service.

Otherwise, set "me" to the URL of the profile that should be able to access your service.

Setting a me value other than None may be useful if you are building personal services that nobody else should be able to access.

If successful, this function returns an IndieAuthCallbackResponse object that contains:

indieweb_utils.**IndieAuthCallbackResponse**(*message: str*, *response: dict*)

This class contains a *response* value. This value is equal to the JSON response sent by the IndieAuth web server.

An example endpoint response looks like this:

```
{
    "me": "https://jamesg.blog/",
    "access_token": "ACCESS_TOKEN",
    "scope": "SCOPE_LIST"
}
```

This function does not check whether a URL has an OAuth provider. Your application should check the list of valid rel me links and only use those that integrate with the OAuth providers your RelMeAuth service supports. For example, if your service does not support Twitter, you should not present Twitter as a valid authentication option to a user, even if the *get_valid_relmeauth_links()* function found a valid two-way rel=me link.

1.6 Webmention

1.6.1 Discover a Webmention Endpoint

Webmention endpoint discovery is useful if you want to know if you can send webmentions to a site or if you want to send a webmention to a site.

You can discover if a URL has an associated webmention endpoint using the *discover_webmention_endpoint* function:

indieweb_utils.**discover_webmention_endpoint**(*target: str*) → WebmentionDiscoveryResponse

Return the webmention endpoint for the given target.

Parameters
target (*str*) – The target to discover the webmention endpoint for.

Returns
The discovered webmention endpoint.

Return type
str

```
import indieweb_utils

target = "https://jamesg.blog/"

webmention_endpoint = indieweb_utils.discover_webmention_endpoint(
    target
)

print(webmention_endpoint) # https://webmention.jamesg.blog/webmention
```

Raises

- **TargetNotProvided** – Target is not provided.
- **WebmentionEndpointNotFound** – Webmention endpoint is not found.
- **UnacceptableIPAddress** – Endpoint does not connect to an accepted IP.

- **LocalhostEndpointFound** – Discovered endpoint is equal to localhost.

If successful, this function returns the URL of the webmention endpoint associated with a resource. In this case, the message value is a blank string.

If a webmention endpoint could not be found, URL is equal to None. In this case, a string message value is provided that you can use for debugging or present to a user.

1.6.2 Send a Webmention

To send a webmention to a target, use this function:

`indieweb_utils.send_webmention(source: str, target: str, me: str = '') → SendWebmentionResponse`

Send a webmention to a target URL.

Parameters

- **source** (*str*) – The source URL of the webmention.
- **target** (*str*) – The target URL to which you want to send the webmention.
- **me** (*str*) – The URL of the user.

Returns
The response from the webmention endpoint.

Return type
SendWebmentionResponse

Example:

```
import indieweb_utils

response = indieweb_utils.send_webmention(
    "https://example.com",
    "https://example.example.com/post/1",
    "https://test.example"
)
```

Raises

- **TargetIsNotApprovedDomain** – Target is not in list of approved domains.
- **GenericWebmentionError** – Generic webmention error.
- **CouldNotConnectToWebmentionEndpoint** – Could not connect to the receiver's webmention endpoint.

This function returns a SendWebmentionResponse object with this structure:

`class indieweb_utils.SendWebmentionResponse(title: str, description: str, url: str, status_code: Optional[int], headers: List[indieweb_utils.webmentions.send.Header])`

1.7 Changelog

All notable changes to this project will be documented in this file.

The format is based on Keep a Changelog, and this project adheres to Semantic Versioning.

1.7.1 [0.5.0] - 2022-10-13

Added

- `get_reply_context` now performs discovery on `property` and `name` values for og:image, twitter:image:src, description, og:description, and twitter:description tags.

Tests

- Added new tests for the `get_reply_context` function.
- Added `responses.activate` decorators to remaining tests that did not already have this decorator present. This ensures all tests run on the contents of local files rather than making network requests to get data from a page.

Fixed

- `get_reply_context` would use a h-entry even if the h-entry only provided a URL and no other content.
- `indieweb_utils.SCOPE_DEFINITIONS` can now be imported into a project. This previously returned an `ImportError` exception.

1.7.2 [0.4.0] - 2022-10-11

Added

Development

- Documentation for the `discover_endpoints` function.
- The `indieauth_callback_handler` function returns the JSON response from an IndieAuth endpoint represented as a dictionary instead of a blank dictionary.
- The `discover_endpoints` docstring contains an example about Microsub and an updated common values list. This is because we recommend use of the `discover_webmention_endpoint` function for Webmention endpoint discovery.

Functions

- `get_reply_urls` to retrieve all of the URLs to which a specified page is replying.
- `get_page_name` to find the name of a page per the IndieWeb Page Name Discovery algorithm.
- `get_syndicated_copies` to retrieve all of the URLs to which a specified page has been syndicated.

Tests

- Added test cases for:
  - `get_reply_urls`
  - `get_page_name`
  - `get_syndicated_copies`
- Updated test cases for `get_reply_context` were to look for `description` values where appropriate.

Fixed

- The `indieauth_callback_handler` function no longer raises a JSON error during the `_validate_indieauth_response` function call.
- The `get_reply_context` function now returns a description based on the first two sentences of the e-content of a specified page if a summary cannot be found when analysing a h-entry.
- The `get_reply_context` function returns a string `summary` value instead of a dictionary or a list.
- `get_reply_context` now looks at `og:description` and `twitter:description` meta tags for a description if a `description` value cannot be found. This happens when analysing a page that does not contain a h-entry.

1.7.3 [0.3.1] - 2022-10-10

Fixed import issue in `setup.cfg` so PyPi can discover the README for indieweb-utils.

1.7.4 [0.3.0] - 2022-10-10

Added

Development

- Provide docstrings for all functions in the library that did not have a docstring.
- Fix docstring rendering issues with library documentation so that all docstrings show up on Read the Docs.
- Add `:raises:` statements to docstrings to document existing
- Add code examples to docstrings and remove redundant examples from RS documentation.
- Add a SECURITY.md policy.
- Split up documentation into more sections to enhance one's ability to navigate the documentation.

Functions

- `discover_h_feed()` function to discover the representative h-feed on a page.
- `get_valid_relmeauth_links()` function to find both one-way and bi-directional rel=me links on a web page.
- `get_representative_h_card()` function to get the representative h-card associated with a web page.
- `get_url_summary()` function to generate a summary from a URL, based on the experimental CASSIS auto_url_summary PHP function.
  - This function provides examples for GitHub, Twitter, Upcoming, Eventbrite (.com and .co.uk), Calagator, IndieWeb Events, and the IndieWeb wiki.
- `autolink_tags()` function to replace hashtags (#) with relevant tag pages and person tags (@) with the names and domains of people tagged.
- Create internal helper functions:
  - `get_parsed_mf2_data()` to retrieve microformats2 data from a page given a parsed mf2py.Parse object, a HTML string, and a URL.
  - `get_soup()` to retrieve a BeautifulSoup object from a provided HTML string and URL.

Tests

- Added test cases for:
  - `discover_h_feed()`
  - `get_representative_h_card()`
  - `get_valid_relmeauth_links()`
  - `get_url_summary()`
  - `autolink_tags()`

Changed

- Support importing IndieAuth functions directly from `indieweb_utils` without having to use `indieweb_utils.indieauth.`.
- Simplify `get_h_app_item()` logic.
- Raise `HAppNotFound` exception when `get_h_app_item()` cannot identify a h-app microformat.
- Renamed `_discover_endpoints` to `discover_endpoints`.
- `discover_endpoints` can raise a `requests.exceptions.RequestException` if there was an error making a request to retrieve an endpoint.
- `discover_webmention_endpoint()` can now raise LocalhostEndpointFound, TargetNotProvided, UnacceptableIPAddress, and WebmentionEndpointNotFound exceptions when there is an issue validating a webmention.
- `send_webmention()` can now raise MissingSourceError, MissingTargetError, UnsupportedProtocolError, TargetIsNotApprovedDomain, GenericWebmentionError, and CouldNotConnectToWebmentionEndpoint if there was an issue sending a webmention.
- `send_webmention()` now returns the HTTP status code and headers of a successful webmention.

- `get_post_type()` raises an `PostTypeFormattingError` exception if an invalid `custom_properties` tuple is provided.
- `get_reply_context()` raises an `ReplyContextRetrievalError` if there was an error retrieving context for a URL. This function also raises an `UnsupportedScheme` error if a URL does not use either HTTP or HTTPS.
- `validate_webmention()` can raise WebmentionIsGone or WebmentionValidationError exceptions if there was an error validating a webmention.
- `canonicalize_url()` returns the exact URL passed in if the URL contains a protocol that is not HTTP or HTTPS.

1.7.5 [0.2.0] - 2022-02-15

Added

- Constants that document different scopes one may want to use in an IndieAuth server.
- Test cases for all main library functions.
- Web page feed discovery function now looks for more MIME types by default.
- New exceptions to throw various errors.
- Add X-Pingback support to feed parsing.
- Use urllib to retrieve domain names, protocols, and paths throughout the library.

Development

- Use tox, black, isort, flake8, and mypy to control quality of code.
- Type hints are used for all functions.
- New documentation has been added for all functions in the library.
- New code snippet examples to function docstrings.

Functions

- `get_h_app_item` function to retrieve a h-app object from a web page.
- `validate_authorization_response` function to validate an IndieAuth authorization response.
- `_verify_decoded_code` function that verifies a decoded code in an IndieAuth request.
- `generate_auth_token` function to generate an authentication token as part of an IndieAuth server.
- `redeem_code` function to handle token redemption in an IndieAuth server.
- `send_webmention` function to send a webmention.
- `validate_webmention` to validate a webmention according to the Webmention specification. Vouch support is implemented as an optional feature to use during the validation process.
- `get_profile` function to retrieve profile information from a h-card on a URL from a URL.

Changed

- Functions now return documented objects instead of arbitrary dictionaries.
- Exceptions are now thrown instead of returning None values or empty dictionaries.
- Fixed various bugs in the reply context function.
- Refactored test cases.
- Code has been formatted using black and isort for readability and adherence to PEP 8.

CHAPTER

TWO

FEATURE SET

This package provides functions that cater to the following needs:

- Generating reply context for a given page.
- Finding the original version of a post per the Original Post Discovery specification.
- Finding the post type per the Post Type Discovery W3C note.
- Finding the webmention endpoint on a page, if one is provided.
- Canonicalizing a URL.
- Discovering the author of a post per the Authorship Specification.
- Handling the response from an IndieAuth callback request.

If any of the above use cases resonate with you, this library may be helpful. Please note this library does not fully implement all IndieWeb specifications.

Rather, this library provides a set of building blocks that you can use to speed up your development of IndieWeb applications.

Here are a few of the many applications that may benefit from the functions provided in this library:

- Micropub server.
- Microsub server.
- Webmention sender.
- Any application that needs to canonicalize a URL.
- An application implementing IndieAuth authentication.

INDEX

A

autolink_tags() (*in module indieweb_utils*), 11

C

canonicalize_url() (*in module indieweb_utils*), 10

D

discover_author() (*in module indieweb_utils*), 4
discover_endpoints() (*in module indieweb_utils*), 3
discover_h_feed() (*in module indieweb_utils*), 8
discover_original_post() (*in module indieweb_utils*), 6
discover_web_page_feeds() (*in module indieweb_utils*), 7
discover_webmention_endpoint() (*in module indieweb_utils*), 24

F

FeedUrl (*class in indieweb_utils*), 7

G

generate_auth_token() (*in module indieweb_utils.indieauth.server*), 17
get_h_app_item() (*in module indieweb_utils*), 14
get_page_name() (*in module indieweb_utils*), 9
get_post_type() (*in module indieweb_utils*), 5
get_profile() (*in module indieweb_utils*), 15
get_reply_context() (*in module indieweb_utils*), 12
get_reply_urls() (*in module indieweb_utils*), 9
get_representative_h_card() (*in module indieweb_utils*), 7
get_url_summary() (*in module indieweb_utils*), 13
get_valid_relmeauth_links() (*in module indieweb_utils*), 16

I

indieauth_callback_handler() (*in module indieweb_utils*), 22
IndieAuthCallbackResponse() (*in module indieweb_utils*), 23
is_authenticated() (*in module indieweb_utils*), 21

R

redeem_code() (*in module indieweb_utils*), 19
ReplyContext (*class in indieweb_utils*), 12

S

send_webmention() (*in module indieweb_utils*), 25
SendWebmentionResponse (*class in indieweb_utils*), 25

V

validate_access_token() (*in module indieweb_utils*), 20
validate_authorization_response() (*in module indieweb_utils*), 18