Dokang documentation

Release 0.9.5

Polyconseil

Sep 27, 2019

Contents

CHAPTER 1

Dokang

Dokang is a lightweight documentation repository. It is a web application that:

1. Provides an endpoint for clients to upload their documentation.

   Sending documentation to Dokang is as simple as issuing a POST query such as:

   ```
   $ curl \
     -X POST \
     --form name=project_name \
     -F ":action=doc_upload" \
     -F content=@../documentation.zip \
     http://dokang:my-secret-token@dokang.example.com/upload
   ```

2. Serves a home page with a list of all documentations and a simple search form that lets users search in HTML, text and PDF files. Other formats can be handled through the use of extensions.
3. Serves all documentations.

Dokang also comes with a command line interface. It is lightweight in the sense that it is merely a wrapper around the Whoosh search engine with a very simple HTML text indexer. It can be extended to retrieve content from other types of files (such as PDF).

Dokang is similar to Readthedocs (although Readthedocs has a much broader set of features) but provides a global search across all hosted documentations. All of this is provided with a simple setup that does not need any relational database.

We (Polyconseil) use it to search through the Sphinx-generated documentation of all our projects. However, it may be used to host any kind of documentation.

1.1 Build and run your own docker image

To build your own image:

```
$ docker build -t dokang .
```

To run the image:

```
$ docker run --rm -e DOKANG_UPLOAD_TOKEN=my_little_secret \
  -e DOKANG_NAME='My docs' \
  -e DOKANG_DESCRIPTION='Documentations of all my projects' \
  -e DOKANG_SERVER_TRUSTED_PROXY=129.14.12.1  # Optional: the IP address of the proxy␣
↪to pass to waitress server's trusted_proxy
  -p 8080:6543
  dokang
```

Go to http://localhost:8080/ in your browser, you should see the list of documentations.

1.2 Topics

1.2.1 Basics

In this chapter, we will skim over the installation and configuration to search in the documentation of Dokang itself.

Concepts

A *document set* represents a single documentation, i.e. a set of related documents (files) that resides in a directory and its sub-directories, for example this documentation of Dokang. You may instruct Dokang to index one or more document sets, and then search in all sets or only one.

A *harvester* extracts content from a file: a list of words and a few metadata, like the title.

Installation

Dokang is compatible with Python 2 (>= 2.7) and Python 3 (>= 3.3).

In a brand new virtual environment, install with:

```
pip install Dokang
```

If you have cloned the Git repository, use this instead:

```
pip install -e .
```

Configuration

The entry point is an `INI` configuration file, an example of which is shipped with the source as dev.ini. It controls both the configuration of the web frontend and general settings. The latter are defined by `dokang.*` options:

dokang.hit_limit The maximum number of search results to fetch. It must be a positive number. If equals to 0 (or if the option is omitted from the file), no limit is set: all results are returned.

Default: no limit.

dokang.index_path The path of the index created by the Whoosh backend. It is a directory that will be created on-the-fly when *the index is initialized*.

dokang.uploaded_docs.dir The path where HTML documentation is uploaded.

To define this path, you may use `%(here)s` to denote the directory that holds the INI file.

dokang.uploaded_docs.token The identification token used to allow documentation upload.

dokang.uploaded_docs.harvester The harvester to use for all projects (fully qualified Python class name).

dokang.opensearch.name The name of your documentation repository, for OpenSearch (see *below*).

dokang.opensearch.name A description of your documentation repository, for OpenSearch (see *below*).

You may want to start from the example file and only customize these values. For further details about Pyramid-related settings, see the corresponding section as well as the Logging section in the Pyramid documentation.

Initializing the index

Once you have created the configuration file, you must initialize the search index. You may do so with the `init` command of the command-line client:

```
$ dokang --settings=dev.ini init
```

Note: If the index already exists and you would like to start from scratch, use the `--force` option to overwrite the index. The index will be **deleted and recreated empty**.

For further details about the arguments and options of the command line client, see *Command line reference*.

Starting Dokang

The INI configuration file described above is a valid WSGI configuration file that you may use with your favorite WSGI server.

On a development machine, you may want to use something like Waitress. First, install Waitress:

```
$ pip install Waitress
```

Then run it:

```
$ pserve dev.ini
Starting server in PID 14135.
serving on http://0.0.0.0:6543
```

See the documentation of Waitress for further details.

Upload and index documentation

If you visit http://localhost:6543 in a web browser, the page will be quite empty. Let's upload the documentation of a project:

- zip the documentation (your ZIP file must have a top-level "index.html");
- post your documentation on http://localhost:6543/upload/ using `multipart/form-data` content type and the following fields:
  - `:action`, must be `doc_upload`,
  - `name`, the name of your project,

  - `content`, the ZIP file.

```
$ cd project_html_doc/
$ 7z a ../documentation.zip .
$ curl -X POST \
       --form name=project_name \
       -F ":action=doc_upload" \
       -F content=@../documentation.zip \
       http://dokang:my-secret-token@localhost:6543/upload
```

You should see a success message. If you refresh http://localhost:6543/ in your web browser, you should now be able to search and find terms that appear in the documentation you have uploaded.

OpenSearch

Dokang has basic support for OpenSearch. That means that you can set up an instance of Dokang as a custom search source (like Google and Wikipedia in Firefox).

1.2.2 Advanced configuration and usage

Configuring harvesters

In the configuration file described in the *Configuration* section of the previous chapter, you must tell Dokang how to analyze files of your document set. For that you need to provide a harvester configuration as a dictionary with the following keys:

exclude An optional list of regular expressions. If the relative path of a file matches one of these expressions, it will not be processed, unless it matches one of the expressions listed in `include`.

include An optional list of regular expressions. If the relative path of a file matches one of these expressions, it will be processed even if the path also matches one of the `exclude` expressions.

This makes it easier to write exclude and include regular expressions.

The configuration must also indicate which harvester to use for each supported file extension. The extensions must not include the leading dot. Here is an example of such a configuration:

```
{'html': SphinxHarvester,
 'include': ('_download', ),
 'exclude': ('^genindex.html$', '^search.html$', '/?_.*')
}
```

To make the configuration a bit easier, Dokang provides a few utilities that build sane configurations for you. For example, the code above is more or less equivalent to the following expression:

```
from dokang.harvesters import sphinx_html_config

sphinx_html_config()
```

You may customize those pre-defined configurations, like this:

```
sphinx_html_config(
    include=your_own_list_of_reg_exps,
    exclude=your_own_list_of_reg_exps,
    pdf=PdfHarvester)
```

For a list of all harvesting configurations and harvesters that ship with Dokang, see the *API* chapter.

External Python packages may also provide their own harvesters. Here is a list of the known ones:

- PDF harvester: dokang_pdf.

Command line reference

All commands of the `dokang` command line program accept a `--settings` argument that is the path to the configuration file:

```
$ dokang --settings=dev.ini init
```

Providing the configuration file in every command may be cumbersome. To work around that, you may define a `DOKANG_SETTINGS` environment variable and then omit the `--settings` option:

```
$ export DOKANG_SETTINGS=/path/to/your/ini.file
$ dokang init
```

Herebelow is the list of available commands of the `dokang` command line program:

`--help` Display a list of commands and general options. Use `dokang <command> --help` to get help and a list of options for a specific command.

`init [--force]` Initialize the index. If the index already exists, Dokang will refuse to overwrite it unless you provide the `--force` option.

`index [--docset DOC_SET_ID] [--force]` Index all configured document sets or only the given document set. If a document has already been indexed, the index is updated. If a document has not been modified since the last indexation, it is not reindexed again (unless the `force` option is provided).

`clear DOC_SET_ID` Remove the given document set from the index.

`search QUERY` Search the index.

1.2.3 Extending Dokang

Dokang currently supports a single backend: Whoosh. Whoosh is responsible for the indexation and the actual search. As of now, Dokang does not let you easily use another backend such as Elasticsearch. Contributions are welcome.

However, you may want to add your own harvester. The harvester is responsible for retrieving data (title and content) from a document. Dokang provides a few harvesters but you may implement your own.

A harvester should be a subclass of `dokang.harvesters.Harvester` and implement a `harvest_file(path)` method that should return a dictionary with the following keys. All values should be text-like: a string (in Python 3) or a unicode object (in Python 2).

title The title of the document.

content The concatenated content of the document.

kind The kind of document: HTML, PDF, etc.

Here is an example of a simple harvester for text files.

```
import codecs
import os

from dokang.harvesters import Harvester
```

(continues on next page)

(continued from previous page)

```
class TextHarvester(Harvester):

    def harvest_file(path):
        with codecs.open(path, encoding='utf-8') as fp:
            return {
                'title': os.path.basename(path),  # Use the filename as the title
                'content: 'fp.read()',
                'kind': 'TXT',
            }
```

1.2.4 Contributing to Dokang

Dokang is hosted on GitHub at https://github.com/polyconseil/dokang/. Suggestions and patches are welcome.

Continuous tests are run on Travis CI. Current status:

Dokang is good enough for us for now, but here are some vague plans:

- adding autocomplete on the search field in the web frontend;
- providing better search results by tweaking Whoosh configuration;
- making the configuration easier.

Dokang is written by Polyconseil and is licensed under the 3-clause BSD license, a copy of which is included in the source and reproduced below:

1.2.5 API

Backends

Index and search backends for Whoosh.

class `dokang.backends.whoosh.`**`WhooshIndexer`**(*index_path*)

Encapsulate indexation through Whoosh.

`initialize`()

Initialize the index.

If an index already exists, it is deleted and recreated from scratch.

`clear_set`(*doc_set*)

Remove all documents of this set from the index.

`index_documents`(*documents*)

Add or update documents in the index.

`delete_documents`(*doc_set*, *paths*)

Delete documents from the index.

class `dokang.backends.whoosh.`**`WhooshSearcher`**(*index_path*)

Encapsulate search through Whoosh.

`get_hashes`()

Return the hash of each indexed document.

`search`(*query_string*, *limit=None*)

Search the query string in the index.

Predefined harvesting configuration

`dokang.harvesters.html.`**`html_config`**(*harvester=<class 'dokang.harvesters.html.HtmlHarvester'>*, *include=None*, *exclude=None*, ***extensions*)

Return a configuration that is suitable for an HTML document set.

`dokang.harvesters.sphinx.`**`sphinx_config`**(*harvester=<class 'dokang.harvesters.sphinx.SphinxHarvester'>*, *include=None*, *exclude=None*, ***extensions*)

Return a configuration that is suitable for a Sphinx-based documentation.

If the documentation uses "Read The Docs" theme, you should rather use `sphinx_rtd_config`.

`dokang.harvesters.sphinx.`**`sphinx_rtd_config`**(*harvester=<class 'dokang.harvesters.sphinx.ReadTheDocsSphinxHarvester'>*, *include=None*, *exclude=None*, ***extensions*)

Return a configuration that is suitable for a Sphinx-based documentation that uses the ReadTheDocs theme.

Harvesters

class `dokang.harvesters.base.`**`Harvester`**

An abstract class for all harvesters.

class `dokang.harvesters.html.`**`HtmlHarvester`**

Harvest content from HTML files.

class `dokang.harvesters.sphinx.`**`SphinxHarvester`**

Harvest content from the HTML rendered version of a Sphinx-based set of documents.

We look at the rendered HTML and not the source files to avoid wrongly indexing files included with the `include` directive.

class `dokang.harvesters.sphinx.`**`ReadTheDocsSphinxHarvester`**

Harvest content from the HTML rendered version of a Sphinx-based set of documents that uses the "Read The Docs" theme.

The "Read The Docs" theme does not generate the `<div>` that we look for in the super class. We have to look for a different one.

1.2.6 Change log

0.9.5 (2016-12-14)

- Improve data repository initialization.

0.9.4 (2016-07-05)

- fix dockerfile.

0.9.3 (2016-07-04)

- Add dockerfile.

0.9.2 (2016-04-26)

- Keep title when updating documentation.

0.9.1 (2016-04-01)

- Fix packaging

0.9.0 (2016-04-01)

- Allow running simultaneous threads of Dokang web application.

  Until now, Dokang updated its list of document sets at startup and when a new document set was uploaded. Running multiple threads of the web application was obviously not working great, as one thread would not see any new document set if it was added by another thread.

  This limitation has now been lifted and Dokang web application can run with multiple threads (for example with multiple uWSGI workers).

Dokang 0.8.2 (2016-02-24)

- Update doc set title after uploading a new version of the documentation.

Dokang 0.8.1 (2016-02-24)

- Fix packaging.

Dokang 0.8.0 (2016-02-24)

- Use the title of the index page as the title of each doc set.
- Group doc sets by the first letter of their title.

Dokang 0.7.0 (2016-02-01)

- Add support of Python 3.5.
- When initializing the index, `dokang init` now creates all needed intermediate-level directories.
- Add purge option to `dokang clear` to delete uploaded files.
- Fix change detection: we used to store and use the modification time of the files. We now compute and store an MD5 hash for each file. It is slower than getting the modification time, but it handles more use cases.

  This is a backward-incompatible change. You must reindex all documents, like this:

  ```
  dokang init --force
  dokang index
  ```

- Remove bogus indexation optimization. The indexation should be a lot faster now, especially on large document base.
- Fix encoding error when parsing non ASCII, non UTF-8 HTML files. UTF-8 files were correctly processed, though.
- Add basic support for OpenSearch.
- Exclude more Sphinx-generated files like `objects.inv` and `searchindex.js`.
- Display path of files in the search results of the command line client.
- Fix bug in document deletion. When a document was detected as deleted from a document set (i.e. when a file was not present anymore in the "upload" directory), the indexation process deleted from the index *all* documents with the same path (for example `index.html`) in *all* document sets. The files themselves were not deleted so the next indexation would add them back to the index.
- Use an asynchronous index writer that allows multiple indexation to be done concurrently. Without this, a `whoosh.index.LockError` exception is raised.

Dokang 0.6.1 (2015-03-03)

- Fix redirection error when uploading documentation.

Dokang 0.6.0 (2015-03-03)

Brown bag release.

- Drop Python 2.6 support.
- Make documentation available from the root of Dokang ("/"). This change is backward-incompatible.

  Before this commit, if the upload dir was named "uploaded", the documentation would be available at `/uploaded/<doc_set_id>`. This was a bit too verbose.

  With this (backward-incompatible) change, the documentation is now available at `/<doc_set_id>`.

Dokang 0.5.0 (2015-02-18)

- Add "highlight" in the query string of the URLs of search results. This parameter is understood by Sphinx-generated HTML files.
- Add documentation uploading end point (to use Dokang web frontend to serve the documentation)

Dokang 0.4.2 (2014-09-01)

- Fix bad-looking (but working) URLs generated in the web front-end. They used to contain two consecutive slashes (for example http://example.com/project//doc.html) when the configuration of the project had a slash at the end of its URL.

Dokang 0.4.1 (2014-08-27)

- Fixed MANIFEST.in so that the Python package contains all templates and stylesheets required by the web front-end.

Dokang 0.4.0 (2014-07-04)

- A new `dokang.hit_limit` option has been added to the INI configuration file. It limits the number of results shown on the web front-end (or lifts this limit if the option is absent).

Dokang 0.3.0 (2014-07-04)

- Fix bug in the HTML harvester. Trying to use it would fail with an exception because Whoosh would complain about something that unexpectedly is a byte string.
- Fix bug in the handling of deleted documents. They were not deleted from the index.

Dokang 0.2.0 (2014-06-24)

Initial version.

Python Module Index

Index