url-matcher Documentation

Release 0.5.0

Zyte

Apr 15, 2024

GETTING STARTED

1 Introduction 3

2 API Reference 11

3 Contributing 13

4 Changelog 15

5 License 17

Python Module Index 19

Index 21

URL matching library that relates URLs with resources. Rules are defined using simple pattern definitions. It is simpler and faster than using regular expressions if the rules involves many domains.

License is BSD 3-clause.

CHAPTER

ONE

INTRODUCTION

Let's start with an example. Imagine that you have several proxy servers and you want to route requests to the right one. You could define the following rules:

- `site1.com` → `us_proxy`
- `site2.com/uk` → `uk_proxy`
- `site2.com/ie` → `ie_proxy`

All URLs from `site1.com` should use the US proxy. The situation for `site2.com` URLs are different: if the path starts with `/uk`, then the UK proxy should be used whereas if the path starts with `/ie` then the IE proxy should be used instead. This library allows to create a matcher that can be used to match URLs with the right proxy using these rules.

Let see how the library can handle this situation:

```
from url_matcher import URLMatcher, Patterns

matcher = URLMatcher()
matcher.add_or_update("us_proxy", Patterns(["site1.com"]))
matcher.add_or_update("uk_proxy", Patterns(["site2.com/uk"]))
matcher.add_or_update("ie_proxy", Patterns(["site2.com/ie"]))

proxy = matcher.match("http://site1.com/articles/article1")
# proxy is "us_proxy" here

proxy = matcher.match("http://site2.com/uk/a_page")
# proxy is "uk_proxy" here

proxy = matcher.match("https://www.site2.com/ie/a_page")
# proxy is "ie_proxy" here

proxy = matcher.match("http://example.com/a_differnt_page")
# proxy is None here
```

As can be seen the the class *url_matcher.URLMatcher* is handy to handle this use case.

Note: Relative URLs are not supported in the `match` method.

1.1 Patterns, include and exclude

A pattern is a URL that describes a set of URLs. For example, the pattern `example.com` describes any URL whose domain is `example.com` or any of its subdomains.

A single pattern is sometimes not enough to describe which URLs to match. This is why we can define instead a set of patterns that are matched against. There is then a list of positive patterns (`include`) and a list of negative ones (`exclude`).

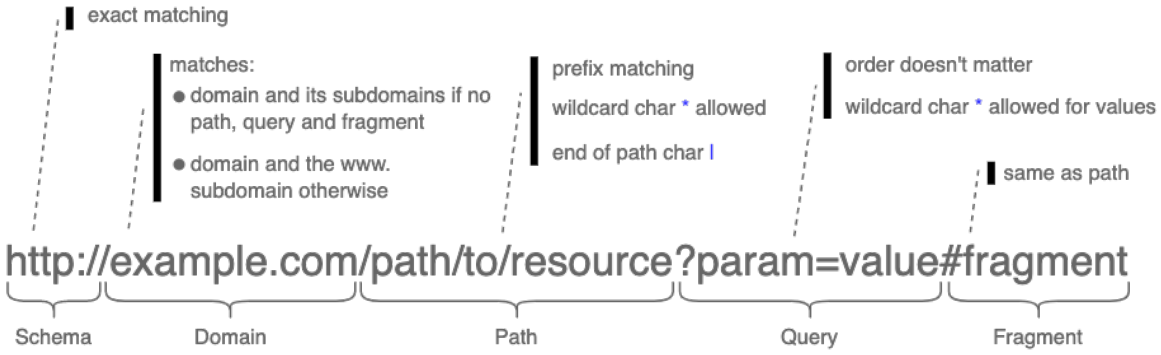
A URL is a match if it matches **at least one** of the patterns in `include` and **none** of the patterns in `exclude`.

This is an example of a rule using such a set of patterns:

```
patterns = Patterns(include=["example.com", "example.org"],
                    exclude=["*.jpg|", "*.jpeg|"])
matcher.add_or_update("proxy_1", patterns))
```

1.2 Patterns

A pattern is a URL that describes a set of URLs. It itself is just a URL. The following diagram summarizes its different parts and what do they mean.



Note: Matching is always **case-insensitive**.

The best way to understand how the patterns work is to look at some examples:

1.2.1 Basic patterns

Pattern	Behaviour
The empty string	Universal pattern. Match any URL
`example.com`	Match any URL whose domain is `example.com` or any of its subdomains. **Match:** • `http://example.com/anything?id=24` • `https://www.example.com/page#with_fragment` **Don't match:** • `http://myexample.com`
`example.com/articles/`	Match any URL whose domain is `example.com` or `www.example.com` and path starts by `/articles/`. **Match:** • `http://www.example.com/articles/article1` • `https://example.com/articles/another_article?id=23` **Don't match:** • `http://example.com/articles` • `http://shop.example.com/articles/article1`

1.2.2 Domain patterns

Pattern	Behaviour
`shop.example.com`	Match any URL whose domain is `shop.example.com` or any of its subdomains. **Match:** • `https://shop.example.com/foo?id=34#fragment` • `http://uk.shop.example.com/foo?id=34` **Don't match:** • `http://myshop.example.com`
`shop.example.com/`	Match any URL whose domain is `shop.example.com` or `www.shop.example.com`. **Match:** • `https://shop.example.com/foo?id=34#fragment` • `http://www.shop.example.com/foo?id=34` **Don't match:** • `http://myshop.example.com` • `http://uk.shop.example.com/foo?id=34`

Note: Rules above only differ by the `/` character and this is enough to change the matching behaviour. The general rule is that the pattern matches the domain or any of the subdomains only if the pattern does not contain a path, a query or a fragment. Otherwise, only URLs with the exact same domain after removing `www.` will match the pattern.

1.2.3 Path patterns

A URL matches if the pattern path is a prefix of it.

Besides, the following modifier characters can be used:

- The `*` character matches any number of characters.
- Use the `|` character at the end of the pattern path if a exact path matching is required.

Pattern	Behaviour
`/articles/`	Match any URL whose path starts by `/articles/`. **Match:** • `http://example.com/articles/an_article?id=23#main` • `https://foo.com/articles/` **Don't match:** • `https://foo.com/articles`
`example.com/index.html|`	Match any URL whose domain is `example.com` or `www.example.com` and path is exactly `/index.html` **Match:** • `http://example.com/index.html?id=24` • `https://www.example.com/index.html#main` **Don't match:** • `http://shop.example.com/index.html` • `http://shop.example.com/index.html_2`
`/images/*.jpg|`	Match any URL whose path starts by `/images/` and whose path ends by `.jpg` **Match:** • `http://example.com/images/foo.jpg` • `https://example.org/images/other/subpath/FOO.JPG?id=23` **Don't match:** • `http://example.com/images/foo.jpeg` • `http://example.com/images/foo.jpg_2`

1.2.4 Query patterns

It serves to match URLs that have some specific parameters in the URL. The order of parameters in the query string is irrelevant. The wildcard char * can be used for values.

If a parameter is repeated in the pattern it will match if any of the values provided is matched

Pattern	Behaviour
`/product|?id=34`	Match any URL whose path is `/product` and contains the query parameter `id` with the value 34 **Match:** • `http://example.com/product?cat=shoes&id=34` **Don't match:** • `http://example.com/product?id=12` • `http://example.com/product/other?id=34`
`/product|?id=*`	Match any URL whose path is `/product` and contains the query parameter `id` with any value **Match:** • `http://example.com/product?cat=shoes&id=34` • `https://example.com/product?id=12&cat=clothes` • `https://example.com/product?id=` **Don't match:** • `http://example.com/product?cat=shoes` • `http://example.com/product?cat=shoes&ids=34`
`?cat=shoes&cat=pants`	Match any URL containing the query parameters `cat` with the values `shoes` or `pants` **Match:** • `http://example.com/product?cat=shoes&id=34` • `http://example.org/p?cat=pants` **Don't match:** • `http://example.org/p?cat=pant`

1.2.5 Fragment patterns

It works exactly like the path.

1.3 Rules conflict resolution

Sometimes several rules can match the same URL. We have then a conflict. By default the library will prioritize the most specific rule. For example, if a URL is matching both a rule with a pattern `example.com` and another with the pattern `example.com/articles` then the later one will be final match because it is more specific.

Alternatively, it is possible to control manually the order of rules by using the `priority` parameter of the *url_matcher.Patterns*. In case of conflict, the rule with the highest priority will be chosen.

The full criteria applied to resolve a conflict between rules are:

1. universality (rules with non universal include patterns are prioritized over rules with universal ones)
2. priority (the highest wins)
3. specificity (the most specific include patterns for the concerning domain wins)
4. the rule id (the rule with the highest id wins)

1.4 Efficiency

Internally, the library clusters the rules by the top level domain of their include patterns. This is done to speed up the matching because it reduces the space of possible rules that can match a URL.

The drawback is that the rules with `include` patterns that do not belong to any top level domain are not supported. In fact, an error is raised.

An exception were done for the universal matching pattern. It is the only cross-top-level-domain `include` pattern that is allowed. The rationale is that is can be convenient to define defaults (e.g. to define the default proxy to use if no other rule matches).

CHAPTER

TWO

API REFERENCE

2.1 Module `url_matcher`

class Patterns(*include: List[str]*, *exclude: List[str] | None = None*, *priority: int = 500*)

__init__(*include: List[str]*, *exclude: List[str] | None = None*, *priority: int = 500*)

all_includes_have_domain() → bool

Return true if all the include patterns have a domain

exclude: `Tuple[str, ...]`

get_domains() → List[str]

get_includes_for(*domain: str*) → List[str]

get_includes_without_domain() → List[str]

include: `Tuple[str, ...]`

is_universal_pattern() → bool

Return true if there are no include patterns or they are empty. A universal pattern matches any domain

priority: `int`

class URLMatcher(*data: Mapping[Any, Patterns] | Iterable[Tuple[Any, Patterns]] | None = None*)

__init__(*data: Mapping[Any, Patterns] | Iterable[Tuple[Any, Patterns]] | None = None*)

A class that matches URLs against a list of patterns, returning the identifier of the rule that matched the URL.

Example usage:

```
matcher = URLMatcher()
matcher.add_or_update(1, Patterns(include=["example.com/product"]))
matcher.add_or_update(2, Patterns(include=["other.com"]))

assert matcher.match("http://example.com/product/a_product.html") == 1
assert matcher.match("http://other.com/a_different_page") == 2
```

Parameters

data – A map or a list of tuples with identifier, patterns pairs to initialize the object from

add_or_update(*identifier: Any*, *patterns:* Patterns)

get(*identifier: Any*) → *Patterns* | None

match(*url: str*, *, *include_universal=True*) → Any | None

match_all(*url: str*, *, *include_universal=True*) → Iterator[Any]

match_universal() → Iterator[Any]

remove(*identifier: Any*)

CHAPTER

THREE

CONTRIBUTING

`url-matcher` is an open-source project. Your contribution is very welcome!

3.1 Issue Tracker

If you have a bug report, a new feature proposal or simply would like to make a question, please check our issue tracker on Github: https://github.com/zytedata/url-matcher/issues

3.2 Source code

Our source code is hosted on Github: https://github.com/zytedata/url-matcher

Before opening a pull request, it might be worth checking current and previous issues. Some code changes might also require some discussion before being accepted so it might be worth opening a new issue before implementing huge or breaking changes.

3.3 Testing

We use tox to run tests with different Python versions:

```
tox
```

The command above also runs type checks; we use mypy.

CHAPTER

FOUR

CHANGELOG

4.1 0.5.0 (2024-04-15)

- Added the `include_universal` argument to *URLMatcher.match()* and *URLMatcher.match_all()*. It can be set to `False` to skip universal matchers.
- Added the *URLMatcher.match_universal()* method that returns only identifiers of universal matchers.
- Added `.readthedocs.yml`.

4.2 0.4.0 (2024-04-03)

- Added official support for Python 3.12.
- Added the *URLMatcher.match_all()* method that returns all matching identifiers.
- Adding a *Patterns* instance with several patterns for the same domain to a *URLMatcher* no longer creates multiple identical `PatternsMatcher` instances.
- CI improvements.

4.3 0.3.0 (2023-09-21)

- Drop Python 3.7 support, make Python 3.11 support official.
- Support tldextract >= 3.6, make the requirement of tldextract >= 1.2 explicit.

4.4 0.2.0 (2022-02-01)

- Update *Patterns* to be **frozen** so instances can easily be deduped based on its hash uniqueness.
- Remove Python 3.6 support

4.5 0.1.0 (2021-11-19)

- Initial release

CHAPTER

FIVE

LICENSE

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of Zyte nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

PYTHON MODULE INDEX

u

url_matcher, 11

INDEX

Symbols

__init__() (*Patterns method*), 11
__init__() (*URLMatcher method*), 11

A

add_or_update() (*URLMatcher method*), 11
all_includes_have_domain() (*Patterns method*), 11

E

exclude (*Patterns attribute*), 11

G

get() (*URLMatcher method*), 12
get_domains() (*Patterns method*), 11
get_includes_for() (*Patterns method*), 11
get_includes_without_domain() (*Patterns method*), 11

I

include (*Patterns attribute*), 11
is_universal_pattern() (*Patterns method*), 11

M

match() (*URLMatcher method*), 12
match_all() (*URLMatcher method*), 12
match_universal() (*URLMatcher method*), 12
module
    url_matcher, 11

P

Patterns (*class in url_matcher*), 11
priority (*Patterns attribute*), 11

R

remove() (*URLMatcher method*), 12

U

url_matcher
    module, 11
URLMatcher (*class in url_matcher*), 11