brownant Documentation

Release 0.1.0

Jiangge Zhang

September 29, 2013

CONTENTS

CHAPTER

ONE

USER'S GUIDE

1.1 Quick Start

There is a simple crawling application written with BrownAnt. It could get the download link from the PyPI home page of given project:

```
from brownant.app import BrownAnt
from brownant.site import Site
from lxml import html
from requests import Session

site = Site(name="pypi")
http = Session()


@site.route("pypi.python.org", "/pypi/<name>", defaults={"version": None})
@site.route("pypi.python.org", "/pypi/<name>/<version>")
def pypi_info(request, name, version):
    url = request.url.geturl()
    etree = html.fromstring(http.get(url).content)
    download_url = etree.xpath(".//div[@id='download-button']/a/@href")[0]

    return {
        "name": name,
        "version": version,
        "download_url": download_url,
    }

app = BrownAnt()
app.mount_site(site)

if __name__ == "__main__":
    from pprint import pprint
    pprint(app.dispatch_url("https://pypi.python.org/pypi/Werkzeug/0.9.4"))
```

And run it, we will get the output:

```
$ python example.py
{'download_url': 'https://.../source/W/Werkzeug/Werkzeug-0.9.4.tar.gz',
 'name': u'Werkzeug',
 'version': u'0.9.4'}
```

CHAPTER

TWO

API REFERENCE

2.1 Basic API

The basic API included the application framework and routing system (provided by `werkzeug.routing`) of BrownAnt.

2.1.1 brownant.app

class `brownant.app.`**BrownAnt**

The app which could manage whole crawler system.

add_url_rule(*host*, *rule_string*, *endpoint*, ***options*)

Add a url rule to the app instance.

The url rule is the same with Flask apps and other Werkzeug apps.

Parameters

- **host** – the matched hostname. e.g. "www.python.org"
- **rule_string** – the matched path pattern. e.g. "/news/<int:id>"
- **endpoint** – the endpoint name as a dispatching key such as the qualified name of the object.

dispatch_url(*url_string*)

Dispatch the URL string to the target endpoint function.

Parameters **url_string** – the origin URL string.

Returns the return value of calling dispatched function.

mount_site(*site*)

Mount a supported site to this app instance.

Parameters **site** – the site instance be mounted.

parse_url(*url_string*)

Parse the URL string with the url map of this app instance.

Parameters **url_string** – the origin URL string.

Returns the tuple as *(url, url_adapter, query_args)*, the url is parsed by the standard library *urlparse*, the url_adapter is from the werkzeug bound URL map, the query_args is a multidict from the werkzeug.

2.1.2 brownant.request

class `brownant.request.`**`Request`**(*url*, *args*)
The crawling request object.

Parameters

- **url** (`urllib.parse.ParseResult`) – the raw URL inputted from the dispatching app.
- **args** (`werkzeug.datastructures.MultiDict`) – the query arguments decoded from query string of the URL.

2.1.3 brownant.site

class `brownant.site.`**`Site`**(*name*)
The site supported object which could be mounted to app instance.

Parameters **name** – the name of the supported site.

`play_actions`(*target*)
Play record actions on the target object.

Parameters **target** (`brownant.site.Site`) – the target which recive all record actions, is a brown ant app instance normally.

`record_action`(*method_name*, **args*, ***kwargs*)
Record the method-calling action.

The actions expect to be played on an target object.

Parameters

- **method_name** – the name of called method.
- **args** – the general arguments for calling method.
- **kwargs** – the keyword arguments for calling method.

`route`(*host*, *rule*, ***options*)
The decorator to register wrapped function to the brown ant app.

The parameters of this method is compatible with the `BrownAnt.add_url_rule()` method.

Parameters

- **host** – the limited host name.
- **rule** – the URL path rule as string.
- **options** – the options to be forwarded to the `Rule` object.

2.2 Declarative API

The declarative API is around the “dinergate” and “pipeline property”.

2.2.1 brownant.dinergate

class brownant.dinergate.**Dinergate**(*request*, *http_client=None*, ***kwargs*)

The simple classify crawler.

In order to work with unnamed properties such as the instances of brownant.pipeline.base.PipelineProperty, the meta class brownant.dinergate.DinergateType will scan subclasses of this class and name all unnamed members which are instances of werkzeug.utils.cached_property.

Parameters

- **request** – the standard parameter passed by app.
- **http_client** – the instance of requests.Session.
- **kwargs** – other arguments from the URL pattern.

URL_TEMPLATE = None

the URL template string for generating crawled target. the *self* could be referenced in the template. .e.g. *"http://www.example.com/items/{self.item_id}?page={self.page}"*

url

The fetching target URL.

The default behavior of this property is build URL string with the URL_TEMPLATE.

The subclasses could override URL_TEMPLATE or give a different implementation of this property.

class brownant.dinergate.**DinergateType**

Bases: type

The metaclass of Dinergate and its subclasses.

This metaclass will give all members are instance of cached_property default names. It is because many pipeline properties are subclasses of cached_property, but them would not be created by decorating functions. They will has not built-in __name__, which may cause them could not cache values as expected.

2.2.2 brownant.pipeline.base

class brownant.pipeline.base.**PipelineProperty**(***kwargs*)

Bases: werkzeug.utils.cached_property

The base class of pipeline properties.

There are three kinds of initial parameters.

- The required attribute. If a keyword argument's name was defined in required_attrs, it will be assigned as a instance attribute.
- The *attr_name*. It is the member of attr_names, whose name always end with *_attr*, such as *text_attr*.
- The option. It will be placed at an instance owned attribute named options. The subclasses could set default option value in the prepare().

A workable subclass of PipelineProperty should implement provide_value(self, obj)(), which accept an argument, the instance of Dinergate.

The implementation of prepare(self)() is optional in subclasses.

Parameters **kwargs** – the parameters with the three kinds.

attr_names = None
the definition of attr_names

get_attr(*obj*, *name*)
Get attribute of the target object with the configured attribute name in the `attr_names` of this instance.

Parameters

- **obj** – the target object.
- **name** – the internal name used in the `attr_names`. .e.g. *"text_attr"*

options = None
the definition of options

prepare()
This method will be called after instance ininialized. The subclasses could override the implementation.

In general purpose, the implementation of this method should give default value to options and the members of `attr_names`.

Example:

```
def prepare(self):
    self.attr_names.setdefault("text_attr", "text")
    self.options.setdefault("use_proxy", False)
```

required_attrs = set([])
the names of required attributes.

2.2.3 brownant.pipeline.network

class brownant.pipeline.network.**URLQueryProperty**(***kwargs*)
The query argument property.

Parameters

- **name** – the query argument name.
- **request_attr** – optional. default: *"request"*.
- **type** – optionl. default: *None*. this value will be passed to `get()`.
- **required** – optionl. default: *True*. while this value be true, the `NotSupported` will be raised for meeting empty value.

class brownant.pipeline.network.**TextResponseProperty**(***kwargs*)
The text response which returned by fetching network resource.

Parameters

- **url_attr** – optional. default: *"url"*. it point to the property which could provide the fetched url.
- **http_client_attr** – optional. default: *"http_client"*. it point to an http client property which is instance of `Session`

2.2.4 brownant.pipeline.html

class brownant.pipeline.html.**ElementTreeProperty**(***kwargs*)
The element tree built from a raw html property.

Parameters **text_response_attr** – optional. default: *"text_response"*.

class brownant.pipeline.html.**XPathTextProperty**(***kwargs*)

The text extracted from a element tree property by XPath.

Parameters

- **xpath** – the xpath expression for extracting text.
- **etree_attr** – optional. default: *"etree"*.
- **strip_spaces** – optional. default: *False*. if it be *True*, the spaces in the beginning and the end of texts will be striped.
- **pick_mode** – optional. default: *"join"*, and could be "join" or "first". while *"join"* be detected, the texts will be joined to one. otherwise the *"first"* be detected, only the first text would be picked.
- **joiner** – optional. default: *" "*. be useable while the *pick_mode* is *"join"*. the texts will be joined with this string.

CHAPTER

THREE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*