
aur documentation

Release 0.11.0

Christopher Down

October 04, 2015

1	Core functions	3
2	The Package class	5
3	Exceptions	7
	Python Module Index	9

aur is a Python library that makes it easy to access and parse data from the [Arch User Repository API](#).

Core functions

The AUR API has four query types. For each of these query types, *aur* exposes a function that calls the API with this query type.

aur.info (*package_name_or_id*)
Return the *Package* with the exact name *package_name_or_id*.

```
>>> info('linux-bfs')
<Package: linux-bfs>
```

aur.msearch (*maintaining_user*)
Return *Package* objects where the maintainer is *maintaining_user*.

```
>>> msearch('cdown')
[<Package: mpdmenu>, <Package: tzupdate>, <Package: yturl>]
```

aur.multiinfo (*package_names_or_ids*)
Return *Package* objects matching the exact names or ids specified in the *iterable* *package_names_or_ids*.
Packages are returned in the form {package_name: package} for easy access.

```
>>> multiinfo(['yturl', 'tzupdate'])
{'tzupdate': <Package: tzupdate>, 'yturl': <Package: yturl>}
```

aur.search (*package_name_substring*)
Return *Package* objects where *package_name_substring* is a substring.

```
>>> search('poco')
[<Package: poco>, <Package: flopoco>, <Package: libpoco-basic>]
```

The Package class

Most functions in this library return *Package* objects in some form. They essentially act as storage objects for all metadata related to a package.

```
>>> yturl = info('yturl')
>>> yturl.description
'YouTube videos on the command line'
>>> yturl.last_modified
datetime.datetime(2015, 9, 8, 22, 26, 24)
>>> yturl.out_of_date
False
```

class `aur.Package`

All package information retrieved from the API is stored in a *Package*, which is a `namedtuple()` with some extensions.

All information about the package is available as attributes with the same name as those returned by the API for each package, except that each one is *snake case* instead of *Pascal case*.

Here are all of the attributes available:

- `category_id`
- `description`
- `first_submitted`
- `id`
- `last_modified`
- `license`
- `maintainer`
- `name`
- `num_votes`
- `out_of_date`
- `package_base`
- `package_base_id`
- `url`
- `url_path`
- `version`

Exceptions

aur uses `requests` internally, so general HTTP(S) exceptions will come from there.

There are also a number of more targeted exceptions defined in *aur* itself:

exception `aur.AURError`

The base class that all AUR exceptions inherit from.

exception `aur.APIError`

Raised when we get a generic API error that we don't have a more specific exception for.

exception `aur.QueryTooShortError`

Raised when the query entered was too short. Typically, most `search()` queries must be at least 3 characters long.

exception `aur.NoSuchPackageError`

Raised when we explicitly requested a particular package, but we don't have any reference to it in the returned data, which means that the requested package doesn't exist.

a

aur, [1](#)

A

APIError, 7
aur (module), 1
AURError, 7

C

category_id (aur.Package attribute), 5

D

description (aur.Package attribute), 5

F

first_submitted (aur.Package attribute), 5

I

id (aur.Package attribute), 5
info() (in module aur), 3

L

last_modified (aur.Package attribute), 5
license (aur.Package attribute), 5

M

maintainer (aur.Package attribute), 5
msearch() (in module aur), 3
multiinfo() (in module aur), 3

N

name (aur.Package attribute), 5
NoSuchPackageError, 7
num_votes (aur.Package attribute), 5

O

out_of_date (aur.Package attribute), 5

P

Package (class in aur), 5
package_base (aur.Package attribute), 5
package_base_id (aur.Package attribute), 5

Q

QueryTooShortError, 7

S

search() (in module aur), 3

U

url (aur.Package attribute), 5
url_path (aur.Package attribute), 5

V

version (aur.Package attribute), 5