

---

# **Pando Documentation**

*Release 0.46*

**Chad Whitacre et al.**

**Jun 03, 2019**



---

## Contents

---

|          |                            |           |
|----------|----------------------------|-----------|
| <b>1</b> | <b>Installation</b>        | <b>3</b>  |
| <b>2</b> | <b>Contents</b>            | <b>5</b>  |
| 2.1      | Tutorial . . . . .         | 5         |
| 2.2      | Reference . . . . .        | 6         |
|          | <b>Python Module Index</b> | <b>21</b> |
|          | <b>Index</b>               | <b>23</b> |



This is Pando, a Python web framework.

Pando's source code is on [GitHub](#), and is [MIT-licensed](#).



# CHAPTER 1

---

## Installation

---

*pando* is available on PyPI:

```
$ pip install pando
```





## 2.1 Tutorial

### 2.1.1 Quick Start

Given: [POSIX](#) and [virtualenv](#)

Step 1: Make a sandbox:

```
$ virtualenv foo
$ cd foo
$ . bin/activate
```

Step 2: Install [pando](#) from PyPI:

```
(foo)$ pip install pando
blah
blah
blah
```

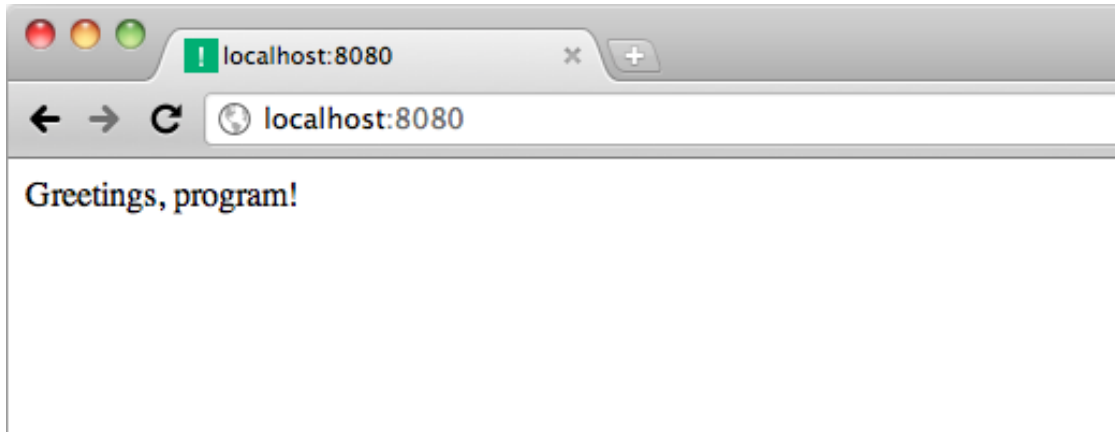
Step 3: Create a website root:

```
(foo)$ mkdir www
(foo)$ cd www
```

Step 4: Create a web page, and start pando inside it:

```
(foo)$ echo Greetings, program! > index.html.spt
(foo)$ python -m pando
Greetings, program! Welcome to port 8080.
```

Step 5: Check [localhost](#) for your new page!



## 2.2 Reference

This is the API reference for the Pando library.

### 2.2.1 `body_parsers`

This module contains Pando's built-in body parsers.

Body parsers are optional ways to enable Pando to uniformly parse POST body content according to its supplied `Content-Type`.

A body parser has the signature:

```
def name(raw, headers):
```

where `raw` is the raw bytestring to be parsed, and `headers` is the `Headers` mapping of the supplied headers.

`pando.body_parsers.formdata` (`raw, headers`)  
Parse raw as form data.

Supports `application/x-www-form-urlencoded` and `multipart/form-data`.

`pando.body_parsers.jsondata` (`raw, headers`)  
Parse raw as JSON data.

### 2.2.2 `exceptions`

Custom exceptions raised by Pando

**exception** `pando.exceptions.CRLFInjection`

A 400 `Response` (per #249) raised if there's a suspected CRLF Injection attack in the headers.

**exception** `pando.exceptions.MalformedHeader` (`header`)

A 400 `Response` (per RFC7230 section 3.2.4) raised if there's no `:` in a header field, or if there's leading or trailing whitespace in the key part of a header field.

**exception** `pando.exceptions.MalformedBody` (`msg`)

A 400 `Response` raised if parsing the body of a POST request fails.

**exception** `pando.exceptions.UnknownBodyType` (*ctype*)  
 A 415 *Response* raised if the Content-Type of the body of a POST request doesn't have a `body_parser` registered for it.

**exception** `pando.exceptions.BadLocation` (*msg*)  
 A 500 *Response* raised if an invalid redirect is attempted.

## 2.2.3 http

### baseheaders

**class** `pando.http.baseheaders.BaseHeaders` (*d*)  
 Bases: `pando.http.mapping.CaseInsensitiveMapping`

Represent the headers in an HTTP Request or Response message.

[How to send non-English unicode string using HTTP header?](#) and [What character encoding should I use for a HTTP header?](#) have good notes on why we do everything as pure bytes here.

**\_\_init\_\_** (*d*)  
 Takes headers as a dict, list, or bytestring.

**\_\_setitem\_\_** (*name, value*)  
 Checks for CRLF in *value*, then calls the superclass method:  
`CaseInsensitiveMapping.__setitem__(name, value)`

**add** (*name, value*)  
 Checks for CRLF in *value*, then calls the superclass method:  
`CaseInsensitiveMapping.add(name, value)`

**raw**  
 Return the headers as a bytestring, formatted for an HTTP message.

**\_\_contains\_\_** (*k*) → True if D has a key *k*, else False

**\_\_getitem\_\_** (*name*)

**all** (*name*)

**get** (*name, default=None*)

**keyerror** (*name*)  
 Raises a 400 *Response*.

**ones** (*\*names*)  
 Given one or more names of keys, return a list of their values.

**pop** (*name*)

**popall** (*name*)  
`D.pop(k[,d]) -> v`, remove specified key and return the corresponding value. If key is not found, *d* is returned if given, otherwise `KeyError` is raised

### mapping

**class** `pando.http.mapping.Mapping`  
 Bases: `aspen.http.mapping.Mapping`

**keyerror** (*name*)

Raises a 400 *Response*.

**\_\_getitem\_\_** (*name*)

Given a name, return the last value or call self.keyerror.

**\_\_setitem\_\_** (*name, value*)

Given a name and value, clobber any existing values.

**add** (*name, value*)

Given a name and value, clobber any existing values with the new one.

**all** (*name*)

Given a name, return a list of values, possibly empty.

**get** (*name, default=None*)

Override to only return the last value.

**ones** (*\*names*)

Given one or more names of keys, return a list of their values.

**pop** (*name, default=<object object>*)

Given a name, return a value.

This removes the last value from the list for name and returns it. If there was only one value in the list then the key is removed from the mapping. If name is not present and default is given, that is returned instead. Otherwise, self.keyerror is called.

**popall** ()

D.pop(k[,d]) -> v, remove specified key and return the corresponding value. If key is not found, d is returned if given, otherwise KeyError is raised

**class** pando.http.mapping.**CaseInsensitiveMapping** (*\*a, \*\*kw*)

Bases: *pando.http.mapping.Mapping*

**\_\_init\_\_** (*\*a, \*\*kw*)

Initializes the mapping.

Loops through positional arguments first, then through keyword args.

Positional arguments can be dicts or lists of items.

**\_\_contains\_\_** (*k*) → True if D has a key k, else False

**\_\_getitem\_\_** (*name*)

**\_\_setitem\_\_** (*name, value*)

**add** (*name, value*)

**get** (*name, default=None*)

**all** (*name*)

**pop** (*name*)

**popall** (*name*)

D.pop(k[,d]) -> v, remove specified key and return the corresponding value. If key is not found, d is returned if given, otherwise KeyError is raised

**keyerror** (*name*)

Raises a 400 *Response*.

**ones** (*\*names*)

Given one or more names of keys, return a list of their values.

## request

Define a Request class and child classes.

Here is how we analyze the structure of an HTTP message, along with the objects we use to model each:

|               |                  |               |
|---------------|------------------|---------------|
| - request     | Request          |               |
| - line        | Line             |               |
| - method      | Method           | ASCII         |
| - uri         | URI              |               |
| - path        | Path             |               |
| - parts       | list of PathPart |               |
| - querystring | Querystring      |               |
| - version     | Version          | ASCII         |
| - headers     | Headers          | str           |
| - cookie      | Cookie           | str           |
| - host        | unicode          | str           |
| - scheme      | unicode          | str           |
| - body        | Body             | Content-Type? |

`pando.http.request.make_franken_uri` (*path*, *qs*)

Given two bytestrings, return a bytestring.

We want to pass ASCII to Request. However, our friendly neighborhood WSGI servers do friendly neighborhood things with the Request-URI to compute PATH\_INFO and QUERY\_STRING. In addition, our friendly neighborhood browser sends “raw, unescaped UTF-8 bytes in the query during an HTTP request” (<http://web.lookout.net/2012/03/unicode-normalization-in-urls.html>).

Our strategy is to try decoding to ASCII, and if that fails (we don’t have ASCII) then we’ll quote the value before passing to Request. What encoding are those bytes? Good question. The above blog post claims that experiment reveals all browsers to send UTF-8, so let’s go with that? BUT WHAT ABOUT MAXTHON?!?!?!.

`pando.http.request.make_franken_headers` (*environ*)

Takes a WSGI environ, returns a dict of HTTP headers.

<https://www.python.org/dev/peps/pep-3333/#environ-variables>

`pando.http.request.kick_against_goad` (*environ*)

Kick against the goad. Try to squeeze blood from a stone. Do our best.

**class** `pando.http.request.Request` (*website*, *method*='GET', *uri*='/', *server\_software*="", *version*='HTTP/1.1', *headers*="", *body*=None)

Bases: `object`

Represent an HTTP Request message.

### line

See *Line*.

### headers

A mapping of HTTP headers. See *Headers*.

**\_\_init\_\_** (*website*, *method*='GET', *uri*='/', *server\_software*="", *version*='HTTP/1.1', *headers*="", *body*=None)

*body* is expected to be a file-like object.

**classmethod** `from_wsgi` (*website*, *environ*)

Given a WSGI environ, return a new instance of the class.

The conversion from HTTP to WSGI is lossy. This method does its best to go the other direction, but we can’t guarantee that we’ve reconstructed the bytes as they were on the wire.

Almost all the keys and values in a WSGI environ dict are (supposed to be) of type *str*, meaning bytestrings in python 2 and unicode strings in python 3. In this function we normalize them to bytestrings. Ref: <https://www.python.org/dev/peps/pep-3333/#a-note-on-string-types>

**method**

**path**

**qs**

**cookie**

**content\_length**

This property attempts to parse the `Content-Length` header.

Returns zero if the header is missing or empty.

Raises a 400 *Response* if the header is not a valid integer.

**body\_bytes**

Lazily read the whole request body.

Returns `b''` if the request doesn't have a body.

**body**

This property calls `parse_body()` and caches the result.

**parse\_body()**

Parses `body_bytes` using `headers` to determine which of the `body_parsers` should be used.

Raises *UnknownBodyType* if the HTTP `Content-Type` isn't recognized, and *MalformedBody* if the parsing fails.

**\_\_str\_\_()**

Lazily load the body and return the whole message.

When working with a Request object interactively or in a debugging situation we want it to behave transparently string-like. We don't want to read bytes off the wire if we can avoid it, though, because for mega file uploads and such this could have a big impact.

**\_\_repr\_\_()**  $\leq \Rightarrow repr(x)$

**\_\_cmp\_\_()** (*other*)

**allow** (*\*methods*)

Given method strings, raise 405 if ours is not among them.

The method names are case insensitive (they are uppercased). If 405 is raised then the Allow header is set to the methods given.

**is\_xhr()**

Check the value of X-Requested-With.

**class** `pando.http.request.Line`

Bases: `str`

Represent the first line of an HTTP Request message.

**static** `__new__(cls, method, uri, version)`

Takes three bytestrings.

`pando.http.request.STANDARD_METHODS = set(['CONNECT', 'DELETE', 'GET', 'HEAD', 'OPTION`

A set containing the 8 basic HTTP methods.

If your application uses other standard methods (see the [HTTP Method Registry](#)), or custom methods, you can add them to this set to improve performance.

**class** pando.http.request.**Method**

Bases: `str`

Represent the HTTP method in the first line of an HTTP Request message.

**static** `__new__` (*cls, raw*)

Creates a new Method object.

Raises a 400 *Response* if the given bytestring is not a valid HTTP method, per RFC7230 section 3.1.1:

Recipients of an invalid request-line SHOULD respond with either a 400 (Bad Request) error or a 301 (Moved Permanently) redirect with the request-target properly encoded.

RFC7230 defines valid methods as:

```
method      = token

token       = 1*tchar

tchar       = "!" / "#" / "$" / "%" / "&" / "'" / "*"
              / "+" / "-" / "." / "^" / "_" / "`" / "|" / "~"
              / DIGIT / ALPHA
              ; any VCHAR, except delimiters
```

**class** pando.http.request.**URI**

Bases: `str`

Represent the Request-URI in the first line of an HTTP Request message.

**static** `__new__` (*cls, raw*)

Creates a URI object from a raw bytestring.

We require that `raw` be decodable with ASCII, if it isn't a 400 *Response* is raised.

**class** pando.http.request.**Path**

Bases: `str`

**decoded**

The path decoded to text.

**mapping**

*Mapping* of path variables.

**parts**

List of `PathPart` instances.

**static** `__new__` (*cls, raw*)

Creates a Path object from a raw bytestring.

**class** pando.http.request.**Querystring**

Bases: `str`

**decoded**

The querystring decoded to text.

**mapping**

*Mapping* of querystring variables.

**static** `__new__` (*cls, raw*)

Creates a Querystring object from a raw bytestring.

**class** pando.http.request.**Version**

Bases: `str`

Holds the version from the HTTP status line, e.g. HTTP/1.1.

Accessing the *info*, *major*, or *minor* attribute will raise a 400 *Response* if the version is invalid.

RFC7230 section 2.6:

```
HTTP-version = HTTP-name "/" DIGIT "." DIGIT
HTTP-name    = %x48.54.54.50 ; "HTTP", case-sensitive
```

```
__slots__ = []
```

**info**

**major**

**minor**

**safe\_decode()**

**class** pando.http.request.**Headers** (*raw*)

Bases: *pando.http.baseheaders.BaseHeaders*

Model headers in an HTTP Request message.

```
__init__(raw)
```

Extend BaseHeaders to add extra attributes.

```
__contains__(k) → True if D has a key k, else False
```

```
__getitem__(name)
```

```
__setitem__(name, value)
```

Checks for CRLF in value, then calls the superclass method:

CaseInsensitiveMapping.**\_\_setitem\_\_(name, value)**

```
add(name, value)
```

Checks for CRLF in value, then calls the superclass method:

CaseInsensitiveMapping.**add(name, value)**

```
all(name)
```

```
get(name, default=None)
```

```
keyerror(name)
```

Raises a 400 *Response*.

```
ones(*names)
```

Given one or more names of keys, return a list of their values.

```
pop(name)
```

```
popall(name)
```

D.pop(k[,d]) -> v, remove specified key and return the corresponding value. If key is not found, d is returned if given, otherwise KeyError is raised

```
raw
```

Return the headers as a bytestring, formatted for an HTTP message.

## response

**class** pando.http.response.**CloseWrapper** (*request, body*)

Conform to WSGI's facility for running code *after* a response is sent.



```
__iter__()
```

```
close()
```

**exception** `pando.http.response.Response` (*code=200, body=u”, headers=None*)

Represent an HTTP Response message.

```
request = None
```

```
whence_raised = (None, None)
```

```
__init__(code=200, body=u”, headers=None)
```

Takes an int, a string, a dict.

- `code` an HTTP response code, e.g., 404
- `body` the message body as a string
- `headers` a dict, list, or bytestring of HTTP headers

Code is first because when you’re raising your own Responses, they’re usually error conditions. Body is second because one more often wants to specify a body without headers, than a header without a body.

```
to_wsgi( environ, start_response, charset)
```

```
__repr__() <==> repr(x)
```

```
__str__() <==> str(x)
```

```
set_whence_raised()
```

Sets `self.whence_raised`

It’s a tuple, (filename, linenum) where we were raised from.

This function needs to be called from inside the *except* block.

## 2.2.4 logging

Pando logging convenience wrappers

```
pando.logging.log(*messages, **kw)
```

Make logging more convenient - use magic to get the `__name__` of the calling module/function and log as it.

‘level’ if present as a kwarg, is the level to log at. ‘upframes’ if present as a kwarg, is how many frames up to look for the name.

other kwargs are passed through to `Logger.log()`

```
pando.logging.log_dammit(*messages, **kw)
```

like `log()`, but critical instead of warning

## 2.2.5 state\_chain

These functions comprise the request processing functionality of Pando.

The order of functions in this module defines Pando’s state chain for request processing. The actual parsing is done by `StateChain.from_dotted_name()`.

Dependencies are injected as specified in each function definition. Each function should return `None`, or a dictionary that will be used to update the state in the calling routine.

It's important that function names remain relatively stable over time, as downstream applications are expected to insert their own functions into this chain based on the names of our functions here. A change in function names or ordering here would constitute a backwards-incompatible change.

`pando.state_chain.parse_enviro`**n\_into\_request** (*environ, website*)

`pando.state_chain.insert_variables_for_aspen` (*request, website*)

`pando.state_chain.request_available` ()

No-op placeholder for easy hookage

`pando.state_chain.raise_200_for_OPTIONS` (*request*)

A hook to return 200 to an 'OPTIONS \*' request

`pando.state_chain.redirect_to_base_url` (*website, request*)

`pando.state_chain.dispatch_path_to_filesystem` ()

`aspen.request_processor.algorithm.dispatch_path_to_filesystem` (*request\_processor,*  
*path,*  
*querystring*)

`pando.state_chain.handle_dispatch_exception` (*website, exception*)

`pando.state_chain.apply_typecasters_to_path` ()

`aspen.request_processor.algorithm.apply_typecasters_to_path` (*request\_processor,*  
*path, state*)

`pando.state_chain.load_resource_from_filesystem` ()

`aspen.request_processor.algorithm.load_resource_from_filesystem` (*request\_processor,*  
*dispatch\_result*)

`pando.state_chain.resource_available` ()

No-op placeholder for easy hookage

`pando.state_chain.create_response_object` (*state*)

`pando.state_chain.render_response` (*state, resource, response, request\_processor*)

`pando.state_chain.get_response_for_exception` (*website, exception*)

`pando.state_chain.response_available` ()

No-op placeholder for easy hookage

`pando.state_chain.log_traceback_for_5xx` (*response, traceback=None*)

`pando.state_chain.delegate_error_to_simplite` (*website, state, response, request=None, resource=None*)

`pando.state_chain.log_traceback_for_exception` (*website, exception*)

`pando.state_chain.log_result_of_request` (*website, request=None, dispatch\_result=None, response=None*)

Log access. With our own format (not Apache's).

## 2.2.6 testing

### client

**exception** `pando.testing.client.DidntRaiseResponse`

**class** `pando.testing.client.FileUpload` (*data, filename, content\_type=None*)  
 Model a file upload for testing. Takes data and a filename.

`pando.testing.client.encode_multipart` (*boundary, data*)  
 Encodes multipart POST data from a dictionary of form values.

The key will be used as the form data name; the value will be transmitted as content. Use the FileUpload class to simulate file uploads (note that they still come out as FieldStorage instances inside of simplates).

**class** `pando.testing.client.Client` (*www\_root=None, project\_root=None*)  
 This is the Pando test client. It is probably useful to you.

**hydrate\_website** (*\*\*kwargs*)

**website**

**load\_resource** (*path*)  
 Given an URL path, return a Resource instance.

**get\_session** ()

**GET** (*\*a, \*\*kw*)

**POST** (*\*a, \*\*kw*)

**OPTIONS** (*\*a, \*\*kw*)

**HEAD** (*\*a, \*\*kw*)

**PUT** (*\*a, \*\*kw*)

**DELETE** (*\*a, \*\*kw*)

**TRACE** (*\*a, \*\*kw*)

**CONNECT** (*\*a, \*\*kw*)

**GxT** (*\*a, \*\*kw*)

**PxST** (*\*a, \*\*kw*)

**xPTIONS** (*\*a, \*\*kw*)

**HxAD** (*\*a, \*\*kw*)

**PxT** (*\*a, \*\*kw*)

**DxLETE** (*\*a, \*\*kw*)

**TRxCE** (*\*a, \*\*kw*)

**CxNNECT** (*\*a, \*\*kw*)

**hxt** (*\*a, \*\*kw*)

**hit** (*method, path=u'/', data=None, body="", content\_type='multipart/form-data; boundary=BoUnDaRyStRiNg', raise\_immediately=True, return\_after=None, want=u'response', \*\*headers*)

**static resolve\_want** (*state, want*)

**build\_wsgi\_environ** (*method, url, body, content\_type, cookies=None, \*\*kw*)

**class** pando.testing.client.**StatefulClient** (\*a, \*\*kw)

This is a Client subclass that keeps cookies between calls.

**\_\_enter\_\_** ()

**\_\_exit\_\_** (\*a)

**hit** (\*a, \*\*kw)

## **harness**

pando.testing.harness.**teardown** ()

Standard teardown function.

- reset the current working directory
- remove FSFIX = % {tempdir}/fsfix
- reset Pando's global state
- clear out sys.path\_importer\_cache

**class** pando.testing.harness.**Harness**

A harness to be used in the Pando test suite itself. Probably not useful to you.

**teardown** ()

**simple** (contents=u'Greetings, program!', filepath=u'index.html.spt', uripath=None, website\_configuration=None, \*\*kw)

A helper to create a file and hit it through our machinery.

**make\_request** (\*a, \*\*kw)

**make\_dispatch\_result** (\*a, \*\*kw)

## **2.2.7 utils**

pando.utils.**maybe\_encode** (s, codec=u'ascii')

pando.utils.**total\_seconds** (td)

Python 2.7 adds a total\_seconds method to timedelta objects.

See [http://docs.python.org/library/datetime.html#datetime.timedelta.total\\_seconds](http://docs.python.org/library/datetime.html#datetime.timedelta.total_seconds)

This function is taken from [https://bitbucket.org/jaraco/jaraco.compat/src/e5806e6c1bcb/py26compat/\\_\\_init\\_\\_.py#cl-26](https://bitbucket.org/jaraco/jaraco.compat/src/e5806e6c1bcb/py26compat/__init__.py#cl-26)

**class** pando.utils.**UTC**

UTC - <http://docs.python.org/library/datetime.html#tzinfo-objects>

**utcoffset** (dt)

datetime -> minutes east of UTC (negative for west of UTC).

**tzname** (dt)

datetime -> string name of time zone.

**dst** (dt)

datetime -> DST offset in minutes east of UTC.

pando.utils.**utcnow** ()

Return a tz-aware datetime.datetime.

`pando.utils.to_rfc822(dt)`

Given a `datetime.datetime`, return an RFC 822-formatted unicode.

Sun, 06 Nov 1994 08:49:37 GMT

According to RFC 1123, day and month names must always be in English. If not for that, this code could use `strftime()`. It can't because `strftime()` honors the locale and could generated non-English names.

`pando.utils.typecheck(*checks)`

Assert that arguments are of a certain type.

Checks is a flattened sequence of objects and target types, like this:

```
( {'foo': 2}, dict
, [1,2,3], list
, 4, int
, True, bool
, 'foo', (basestring, None)
)
```

The target type can be a single type or a tuple of types. None is special-cased (you can specify None and it will be interpreted as `type(None)`).

```
>>> typecheck()
>>> typecheck('foo')
Traceback (most recent call last):
...
AssertionError: typecheck takes an even number of arguments.
>>> typecheck({'foo': 2}, dict)
>>> typecheck([1,2,3], list)
>>> typecheck(4, int)
>>> typecheck(True, bool)
>>> typecheck('foo', (str, None))
>>> typecheck(None, None)
>>> typecheck(None, type(None))
>>> typecheck('foo', unicode)
Traceback (most recent call last):
...
TypeError: Check #1: 'foo' is of type str, but unicode was expected.
>>> typecheck('foo', (basestring, None))
Traceback (most recent call last):
...
TypeError: Check #1: 'foo' is of type str, not one of: basestring, NoneType.
>>> class Foo(object):
...     def __repr__(self):
...         return "<Foo>"
...
>>> typecheck(Foo(), dict)
Traceback (most recent call last):
...
TypeError: Check #1: <Foo> is of type __main__.Foo, but dict was expected.
>>> class Bar:
...     def __repr__(self):
...         return "<Bar>"
...
>>> typecheck(Bar(), dict)
Traceback (most recent call last):
...
TypeError: Check #1: <Bar> is of type instance, but dict was expected.
```

(continues on next page)

(continued from previous page)

```
>>> typecheck('foo', str, 'bar', unicode)
Traceback (most recent call last):
...
TypeError: Check #2: 'bar' is of type str, but unicode was expected.
```

## 2.2.8 website

**class** pando.website.**Website** (\*\*kwargs)

Represent a website.

This object holds configuration information, and how to handle HTTP requests (per WSGI). It is available to user-developers inside of their simplates and state chain functions.

**\_\_init\_\_** (\*\*kwargs)

Takes configuration in kwargs.

**request\_processor** = None

An Aspen `RequestProcessor` instance.

**state\_chain** = None

The chain of functions used to process an HTTP request, imported from `pando.state_chain`.

**body\_parsers** = None

Mapping of content types to parsing functions.

**\_\_call\_\_** (environ, start\_response)

Alias of `wsgi_app()`.

**wsgi\_app** (environ, start\_response)

WSGI interface.

Wrap this method (instead of the website object itself) when you want to use WSGI middleware:

```
website = Website()
website.wsgi_app = WSGIMiddleware(website.wsgi_app)
```

**respond** (environ, raise\_immediately=None, return\_after=None)

Given a WSGI environ, return a state dict.

**redirect** (location, code=None, permanent=False, base\_url=None, response=None)

Raise a redirect Response.

If code is None then it will be set to 301 (Moved Permanently) if permanent is True and 302 (Found) if it is False. If url doesn't start with base\_url (defaulting to self.base\_url), then we prefix it with base\_url before redirecting. This is a protection against open redirects. If you wish to use a relative path or full URL as location, then base\_url must be the empty string; if it's not, we raise `BadLocation`. If you provide your own response we will set `.code` and `.headers['Location']` on it.

**canonicalize\_base\_url** (request)

Enforces a base\_url such as `http://localhost:8080` (no path part).

**find\_ours** (filename)

Given a filename, return the filepath to pando's internal version of that filename.

No existence checking is done, this just abstracts away the `__file__` reference nastiness.

**ours\_or\_theirs** (filename)

Given a filename, return a filepath or None.

It looks for the file in `self.project_root`, then in Pando's default files directory. None is returned if the file is not found in either location.

## 2.2.9 wsgi

Provide a WSGI callable.

(It could be nice if this was at `pando:wsgi` instead of `pando.wsgi:website`, but then `Website` would be instantiated every time you import the `pando` module. Here, it's only instantiated when you pass this to a WSGI server like `gunicorn`, `spawning`, etc.)

```
pando.wsgi.website = <pando.website.Website object>
```

This is the WSGI callable, an instance of *Website*.

```
pando.wsgi.application = <pando.website.Website object>
```

Alias of `website`. A number of WSGI servers look for this name by default, for example running `gunicorn pando.wsgi` works.





### p

- pando, 6
- pando.body\_parsers, 6
- pando.exceptions, 6
- pando.http, 7
- pando.http.baseheaders, 7
- pando.http.mapping, 7
- pando.http.request, 8
- pando.http.response, 12
- pando.logging, 13
- pando.state\_chain, 13
- pando.testing, 14
- pando.testing.client, 15
- pando.testing.harness, 16
- pando.utils, 16
- pando.website, 18
- pando.wsgi, 19



## Symbols

- \_\_call\_\_() (*pando.website.Website* method), 18  
 \_\_cmp\_\_() (*pando.http.request.Request* method), 10  
 \_\_contains\_\_() (*pando.http.baseheaders.BaseHeaders* method), 7  
 \_\_contains\_\_() (*pando.http.mapping.CaseInsensitiveMapping* method), 8  
 \_\_contains\_\_() (*pando.http.request.Headers* method), 12  
 \_\_enter\_\_() (*pando.testing.client.StatefulClient* method), 16  
 \_\_exit\_\_() (*pando.testing.client.StatefulClient* method), 16  
 \_\_getitem\_\_() (*pando.http.baseheaders.BaseHeaders* method), 7  
 \_\_getitem\_\_() (*pando.http.mapping.CaseInsensitiveMapping* method), 8  
 \_\_getitem\_\_() (*pando.http.mapping.Mapping* method), 8  
 \_\_getitem\_\_() (*pando.http.request.Headers* method), 12  
 \_\_init\_\_() (*pando.http.baseheaders.BaseHeaders* method), 7  
 \_\_init\_\_() (*pando.http.mapping.CaseInsensitiveMapping* method), 8  
 \_\_init\_\_() (*pando.http.request.Headers* method), 12  
 \_\_init\_\_() (*pando.http.request.Request* method), 9  
 \_\_init\_\_() (*pando.http.response.Response* method), 13  
 \_\_init\_\_() (*pando.website.Website* method), 18  
 \_\_iter\_\_() (*pando.http.response.CloseWrapper* method), 12  
 \_\_new\_\_() (*pando.http.request.Line* static method), 10  
 \_\_new\_\_() (*pando.http.request.Method* static method), 11  
 \_\_new\_\_() (*pando.http.request.Path* static method), 11  
 \_\_new\_\_() (*pando.http.request.Querystring* static method), 11  
 \_\_new\_\_() (*pando.http.request.URI* static method), 11  
 \_\_repr\_\_() (*pando.http.request.Request* method), 10  
 \_\_repr\_\_() (*pando.http.response.Response* method), 13  
 \_\_setitem\_\_() (*pando.http.baseheaders.BaseHeaders* method), 7  
 \_\_setitem\_\_() (*pando.http.mapping.BaseHeaders.CaseInsensitiveMapping* method), 7  
 \_\_setitem\_\_() (*pando.http.mapping.CaseInsensitiveMapping* method), 8  
 \_\_setitem\_\_() (*pando.http.mapping.Headers.CaseInsensitiveMapping* method), 12  
 \_\_setitem\_\_() (*pando.http.mapping.Mapping* method), 8  
 \_\_setitem\_\_() (*pando.http.request.Headers* method), 12  
 slots\_\_ (*pando.http.request.Version* attribute), 12  
 \_\_str\_\_() (*pando.http.request.Request* method), 10  
 \_\_str\_\_() (*pando.http.response.Response* method), 13
- ## A
- add() (*pando.http.baseheaders.BaseHeaders* method), 7  
 add() (*pando.http.mapping.BaseHeaders.CaseInsensitiveMapping* method), 7  
 add() (*pando.http.mapping.CaseInsensitiveMapping* method), 8  
 add() (*pando.http.mapping.Headers.CaseInsensitiveMapping* method), 12  
 add() (*pando.http.mapping.Mapping* method), 8  
 add() (*pando.http.request.Headers* method), 12  
 all() (*pando.http.baseheaders.BaseHeaders* method), 7  
 all() (*pando.http.mapping.CaseInsensitiveMapping* method), 8  
 all() (*pando.http.mapping.Mapping* method), 8  
 all() (*pando.http.request.Headers* method), 12  
 allow() (*pando.http.request.Request* method), 10  
 application (in module *pando.wsgi*), 19  
 apply\_typecasters\_to\_path() (in module *aspen.request\_processor.algorithm*), 14

`apply_typecasters_to_path()` (in module `pando.state_chain`), 14

## B

`BadLocation`, 7

`BaseHeaders` (class in `pando.http.baseheaders`), 7

`body` (`pando.http.request.Request` attribute), 10

`body_bytes` (`pando.http.request.Request` attribute), 10

`body_parsers` (`pando.website.Website` attribute), 18

`build_wsgi_environ()` (`pando.testing.client.Client` method), 15

## C

`canonicalize_base_url()` (`pando.website.Website` method), 18

`CaseInsensitiveMapping` (class in `pando.http.mapping`), 8

`Client` (class in `pando.testing.client`), 15

`close()` (`pando.http.response.CloseWrapper` method), 13

`CloseWrapper` (class in `pando.http.response`), 12

`CONNECT()` (`pando.testing.client.Client` method), 15

`content_length` (`pando.http.request.Request` attribute), 10

`cookie` (`pando.http.request.Request` attribute), 10

`create_response_object()` (in module `pando.state_chain`), 14

`CRLFInjection`, 6

`CxNNECT()` (`pando.testing.client.Client` method), 15

## D

`decoded` (`pando.http.request.Path` attribute), 11

`decoded` (`pando.http.request.Querystring` attribute), 11

`delegate_error_to_simplite()` (in module `pando.state_chain`), 14

`DELETE()` (`pando.testing.client.Client` method), 15

`DidntRaiseResponse`, 15

`dispatch_path_to_filesystem()` (in module `aspen.request_processor.algorithm`), 14

`dispatch_path_to_filesystem()` (in module `pando.state_chain`), 14

`dst()` (`pando.utils.UTC` method), 16

`DxLETE()` (`pando.testing.client.Client` method), 15

## E

`encode_multipart()` (in module `pando.testing.client`), 15

## F

`FileUpload` (class in `pando.testing.client`), 15

`find_ours()` (`pando.website.Website` method), 18

`formdata()` (in module `pando.body_parsers`), 6

`from_wsgi()` (`pando.http.request.Request` class method), 9

## G

`get()` (`pando.http.baseheaders.BaseHeaders` method), 7

`get()` (`pando.http.mapping.CaseInsensitiveMapping` method), 8

`get()` (`pando.http.mapping.Mapping` method), 8

`get()` (`pando.http.request.Headers` method), 12

`GET()` (`pando.testing.client.Client` method), 15

`get_response_for_exception()` (in module `pando.state_chain`), 14

`get_session()` (`pando.testing.client.Client` method), 15

`GxT()` (`pando.testing.client.Client` method), 15

## H

`handle_dispatch_exception()` (in module `pando.state_chain`), 14

`Harness` (class in `pando.testing.harness`), 16

`HEAD()` (`pando.testing.client.Client` method), 15

`Headers` (class in `pando.http.request`), 12

`headers` (`pando.http.request.Request` attribute), 9

`hit()` (`pando.testing.client.Client` method), 15

`hit()` (`pando.testing.client.StatefulClient` method), 16

`HxAD()` (`pando.testing.client.Client` method), 15

`hxt()` (`pando.testing.client.Client` method), 15

`hydrate_website()` (`pando.testing.client.Client` method), 15

## I

`info` (`pando.http.request.Version` attribute), 12

`insert_variables_for_aspen()` (in module `pando.state_chain`), 14

`is_xhr()` (`pando.http.request.Request` method), 10

## J

`jsondata()` (in module `pando.body_parsers`), 6

## K

`keyerror()` (`pando.http.baseheaders.BaseHeaders` method), 7

`keyerror()` (`pando.http.mapping.CaseInsensitiveMapping` method), 8

`keyerror()` (`pando.http.mapping.Mapping` method), 7

`keyerror()` (`pando.http.request.Headers` method), 12

`kick_against_goad()` (in module `pando.http.request`), 9

## L

`Line` (class in `pando.http.request`), 10

`line` (`pando.http.request.Request` attribute), 9

`load_resource()` (`pando.testing.client.Client` method), 15

- load\_resource\_from\_filesystem() (in module *aspen.request\_processor.algorithm*), 14
- load\_resource\_from\_filesystem() (in module *pando.state\_chain*), 14
- log() (in module *pando.logging*), 13
- log\_dammit() (in module *pando.logging*), 13
- log\_result\_of\_request() (in module *pando.state\_chain*), 14
- log\_traceback\_for\_5xx() (in module *pando.state\_chain*), 14
- log\_traceback\_for\_exception() (in module *pando.state\_chain*), 14
- ## M
- major (*pando.http.request.Version* attribute), 12
- make\_dispatch\_result() (*pando.testing.harness.Harness* method), 16
- make\_franken\_headers() (in module *pando.http.request*), 9
- make\_franken\_uri() (in module *pando.http.request*), 9
- make\_request() (*pando.testing.harness.Harness* method), 16
- MalformedBody, 6
- MalformedHeader, 6
- Mapping (class in *pando.http.mapping*), 7
- mapping (*pando.http.request.Path* attribute), 11
- mapping (*pando.http.request.Querystring* attribute), 11
- maybe\_encode() (in module *pando.utils*), 16
- Method (class in *pando.http.request*), 10
- method (*pando.http.request.Request* attribute), 10
- minor (*pando.http.request.Version* attribute), 12
- ## O
- ones() (*pando.http.baseheaders.BaseHeaders* method), 7
- ones() (*pando.http.mapping.CaseInsensitiveMapping* method), 8
- ones() (*pando.http.mapping.Mapping* method), 8
- ones() (*pando.http.request.Headers* method), 12
- OPTIONS() (*pando.testing.client.Client* method), 15
- ours\_or\_theirs() (*pando.website.Website* method), 18
- ## P
- pando (module), 6
- pando.body\_parsers (module), 6
- pando.exceptions (module), 6
- pando.http (module), 7
- pando.http.baseheaders (module), 7
- pando.http.mapping (module), 7
- pando.http.request (module), 8
- pando.http.response (module), 12
- pando.logging (module), 13
- pando.state\_chain (module), 13
- pando.testing (module), 14
- pando.testing.client (module), 15
- pando.testing.harness (module), 16
- pando.utils (module), 16
- pando.website (module), 18
- pando.wsgi (module), 19
- parse\_body() (*pando.http.request.Request* method), 10
- parse\_enviro\_into\_request() (in module *pando.state\_chain*), 14
- parts (*pando.http.request.Path* attribute), 11
- Path (class in *pando.http.request*), 11
- path (*pando.http.request.Request* attribute), 10
- pop() (*pando.http.baseheaders.BaseHeaders* method), 7
- pop() (*pando.http.mapping.CaseInsensitiveMapping* method), 8
- pop() (*pando.http.mapping.Mapping* method), 8
- pop() (*pando.http.request.Headers* method), 12
- popall() (*pando.http.baseheaders.BaseHeaders* method), 7
- popall() (*pando.http.mapping.CaseInsensitiveMapping* method), 8
- popall() (*pando.http.mapping.Mapping* method), 8
- popall() (*pando.http.request.Headers* method), 12
- POST() (*pando.testing.client.Client* method), 15
- PUT() (*pando.testing.client.Client* method), 15
- PxST() (*pando.testing.client.Client* method), 15
- PxT() (*pando.testing.client.Client* method), 15
- ## Q
- qs (*pando.http.request.Request* attribute), 10
- Querystring (class in *pando.http.request*), 11
- ## R
- raise\_200\_for\_OPTIONS() (in module *pando.state\_chain*), 14
- raw (*pando.http.baseheaders.BaseHeaders* attribute), 7
- raw (*pando.http.request.Headers* attribute), 12
- redirect() (*pando.website.Website* method), 18
- redirect\_to\_base\_url() (in module *pando.state\_chain*), 14
- render\_response() (in module *pando.state\_chain*), 14
- Request (class in *pando.http.request*), 9
- request (*pando.http.response.Response* attribute), 13
- request\_available() (in module *pando.state\_chain*), 14
- request\_processor (*pando.website.Website* attribute), 18
- resolve\_want() (*pando.testing.client.Client* static method), 15

resource\_available() (in module *pando.state\_chain*), 14  
respond() (*pando.website.Website* method), 18  
Response, 13  
response\_available() (in module *pando.state\_chain*), 14

## S

safe\_decode() (*pando.http.request.Version* method), 12  
set\_whence\_raised() (*pando.http.response.Response* method), 13  
simple() (*pando.testing.harness.Harness* method), 16  
STANDARD\_METHODS (in module *pando.http.request*), 10  
state\_chain (*pando.website.Website* attribute), 18  
StatefulClient (class in *pando.testing.client*), 15

## T

teardown() (in module *pando.testing.harness*), 16  
teardown() (*pando.testing.harness.Harness* method), 16  
to\_rfc822() (in module *pando.utils*), 16  
to\_wsgi() (*pando.http.response.Response* method), 13  
total\_seconds() (in module *pando.utils*), 16  
TRACE() (*pando.testing.client.Client* method), 15  
TRxCE() (*pando.testing.client.Client* method), 15  
typecheck() (in module *pando.utils*), 17  
tzname() (*pando.utils.UTC* method), 16

## U

UnknownBodyType, 6  
URI (class in *pando.http.request*), 11  
UTC (class in *pando.utils*), 16  
utcnow() (in module *pando.utils*), 16  
utcoffset() (*pando.utils.UTC* method), 16

## V

Version (class in *pando.http.request*), 11

## W

Website (class in *pando.website*), 18  
website (in module *pando.wsgi*), 19  
website (*pando.testing.client.Client* attribute), 15  
whence\_raised (*pando.http.response.Response* attribute), 13  
wsgi\_app() (*pando.website.Website* method), 18

## X

xPTIONS() (*pando.testing.client.Client* method), 15