NCSS Tornado Cheatsheet

Tim Dawborn

NCSS 2018

Our Tornado wrappers live in the `tornado.ncss` module. There are only two symbols exported in this module: `Server` and `ncssbook_log`. `Server` contains all of the logic for proxying user defined functions to request and setting up the Tornado Application instance. `ncssbook_log` is a standard Python logger instance setup with pretty console logging for use in peoples apps.

```
class Server:
  def __init__(self, *, hostname:str='', port:int=8888, static_path:str='static') -> None:
    """ Constructor arguments are passed to the Tornado Application instance. """
  def register(self, url_pattern:str, handler, *,
               delete=None, get=None, patch=None, post=None, put=None,
               url_name:str=None, **kwargs) -> None:
    """ Declares a mapping between a URL pattern and a set of callables.
    Requests to all HTTP methods map to `handler` by default, unless their corresponding kwarg is
    specified. `url_name` can be provided if you would like URL reversal support (see
    `RequestHander.reverse_url`).
    `handler` can also be a normal class-based Tornado handler as well. Useful if people want to
    write REST APIs or use WebSockets. The `kwargs` are used as the arguments to `initialise`
    in this case. """
  def run(self) -> None:
    """ This method should be the last thing called in your main.
    Starts the Tornado IOLoop instance and does not return. """
```

Each of the callables registered to `Server.register` must take at least one argument: the Tornado `RequestHandler`[1] instance. The `RequestHandler` instance provided as this first argument is a subclass of the standard Tornado `RequestHandler` with some NCSS wrapping added to make the API more beginnner friendly. Each handler should have an additional n arguments where n is the number of captures in the URL pattern(s) that map to the handler. For example:

```
from tornado.ncss import Server, ncssbook_log

def index_handler(request):
  ...

def book_handler(request, book_id):
  book_id = int(book_id)
  if book_id not in books_database:
    ncssbook_log.error('Book not found: %d', book_id)
    ...
  else:
    ...

server = Server()
server.register(r'/', index_handler)
server.register(r'/book/(\d+)/', book_handler)
server.run()
```

Captured URL values are passed as `str` instances to their handler functions.

[1] http://www.tornadoweb.org/en/stable/web.html#request-handlers

The methods and properties of interest on the provided `RequestHandler` instance are:

```
class Handler(tornado.web.RequestHandler):
  # GET and POST parameters.
  def get_field(name:str, default=None) -> str or None:
    """ Returns the corresponding value for a GET parameter named `name`. """
  def get_fields() -> {str: str}
    """ Returns a dictionary of all GET parameters. """

  # File uploads (multipart/form-data).
  def get_file(name:str, default=None) -> (str, str, bytes):
    """ Returns a 3-tuple of (filename, content_type, content) for the uploaded file given by
    `name`. `filename` and `content_type` are both strings and `content` is a bytes. If the
    file was not in the POST payload, all three values will be None. """

  # Cookies.
  def get_secure_cookie(name:str, default=None) -> bytes or None:
    """ Returns the corresponding cookie value, or `default` if not set or if cookie fails to
    validate. Note that this returns a bytes, not a str. """
  def set_secure_cookie(name:str, value:str or bytes or None) -> None:
    """ Sets the corresponding cookie. `value`s of type str are UTF-8 encoded. """
  def clear_cookie(name:str) -> None:
    """ Clears the corresponding cookie. """

  # HTTP headers.
  def set_header(name:str, value:str) -> None:
    """ Set a HTTP header. """
  def clear_header(name:str) -> None:
    """ Clear a HTTP header. """

  # HTTP request.
  request -> tornado.httpserver.HTTPRequest
  """ Contains useful properties such as `method`. """

  # HTTP response.
  def write(data:str or bytes or dict) -> None:
    """ Writes a chunk of `data` to the output stream. If `data` is a dict instance, the
    Content-Type header is set to application/json and `data` is JSON encoded. If `data` is
    an instance of str, it is UTF-8 encoded before being written. """
  def redirect(url:str) -> None:
    """ Set the HTTP status to 302 and redirect to `url`. The `url` argument can be
    constructed by `reverse_url` if named URL patterns are defined. """

  # Reversing paths.
  def reverse_url(url_name:str, *args:[str]) -> str or KeyError:
    """ Reverses a URL name to a URL, with `*args` used to populate the URL captures
    element-wise. If the named URL does not exist, a KeyError is raised. If the wrong
    number of arguments are provided, Tornado fails an assertion (wat).
    This functionality is rather limited in Tornado. For example, you cannot reverse a URL
    pattern with a non-capturing group (e.g. r'/book/(?:(\d+)/)?'). """
  def static_url(path:str, include_host:bool=False) -> str:
    """ Used to construct a path to the static asset given the relative path inside the
    static asset directory. Tornado also does a file existance check when this method
    is used. """
```