WsgiService Documentation

Release 0.4.0

Patrice Neff

February 28, 2014

Contents

WsgiService is a project to create a lean Python WSGI framework for very easy creation of REST services.

A REST service in this context is a HTTP service to be used by machines. So a service should output something like XML, JSON or any other machine-readable format.

CHAPTER 1

Contents

1.1 Philosophy

WsgiService allows very easy creation of REST web services. The basic goal is to provide a great HTTP service implementation to free the service author from some tedious work in relation to HTTP. It takes advantage of HTTP by making it easy to provide cachable responses, handle conditional requests and do content negotiation to service a wide spectrum of clients.

The primary guiding principle is that the services should be as easy and small to write as possible.

WsgiService is not intended to be frontend framework. Creating full-featured frontend applications with WsgiService is quite cumbersome.

1.1.1 Goals

- Abstract away error and status code handling
- Make it easy to create machine readable output
- Easily validate input
- Easy deployment using good configuration file handling
- Make testing easy
- Create usable REST API documentation from source
- Content negotiation to automatically use the correct output format

1.2 Tutorial

Creating a WsgiService application requires very little code. First you'll need to import wsgiservice of course:

```
from wsgiservice import *
```

Next you create a subclass of `wsgiservice.Resource` which will handle the different representations of a resource. For example a document resource which stores documents in an in-memory dictionary:

```
data = {}

@mount('/{id}')
class Document(Resource):
```

```
    NOT_FOUND = (KeyError,)

    def GET(self, id):
        """Return the document indicated by the ID."""
        return data[id]

    def PUT(self, id):
        """Overwrite or create the document indicated by the ID."""
        is_new = id not in data
        data.setdefault(id, {'id': id})
        for key in self.request.POST:
            data[id][key] = self.request.POST[key]
        retval = {'id': id, 'saved': True}
        if is_new:
            self.response.body_raw = retval
            raise_201(self, id)
        else:
            return retval

    def DELETE(self, id):
        """Delete the document indicated by the ID."""
        del data[id]

    def get_etag(self, id):
        return id
```

Each resource defines the different HTTP methods it accepts. Additionally there are a special methods such as `get_etag` as described in `wsgiservice.Resource` in more detail. All of these methods can specify any number of parameters to accept. Those will be filled automatically from these locations (in that order):

1. The special `request` parameter which will be filled with an instance of `webob.Request` for the current request.
2. Parameters extracted from the path. *id* in the example above.
3. Parameters from the query string.
4. Parameters from the POST data.

Let's also create a `Documents` resource which can be used to create a new document:

```
import uuid

@mount('/')
class Documents(Resource):
    def POST(self):
        """Create a new document, assigning a unique ID."""
        id = str(uuid.uuid4())
        res = self.get_resource(Document)
        return res.PUT(id)
```

You see how easy it is to use an existing request to do the work of saving.

Finally you'll need to create the actual WSGI application and serve it:

```
app = get_app(globals())

if __name__ == '__main__':
    from wsgiref.simple_server import make_server
    print "Running on port 8001"
    make_server('', 8001, app).serve_forever()
```

This uses the wsgiref simple server which comes with Python. For production you'll probably want to use a different server. But for now you can run the resulting file and on port 8001 you'll have a simple document server available.

1.3 wsgiservice – Reference

1.3.1 resource

Every resource to be served with WsgiService should inherit from the `wsgiservice.resource.Resource` class.

Resource class

class `wsgiservice.resource.`**Resource**(*request*, *response*, *path_params*, *application=None*)

Base class for all WsgiService resources. A resource is a unique REST endpoint which accepts different methods for different actions.

For each HTTP call the corresponding method (equal to the HTTP method) will be called.

OPTIONS()

Default implementation of the OPTIONS verb. Outputs a list of allowed methods on this resource in the `Allow` response header.

assert_condition_etag()

If the resource has an ETag (see `get_etag()`) the request headers `If-Match` and `If-None-Match` are verified. May abort the request with 304 or 412 response codes.

Raises

- `webob.exceptions.ResponseException` of status 304 if the ETag matches the `If-None-Match` request header (GET/HEAD requests only).
- `webob.exceptions.ResponseException` of status 412 if the ETag matches the `If-None-Match` request header (for requests other than GET/HEAD) or the ETag does not match the `If-Match` header.

assert_condition_last_modified()

If the resource has a last modified date (see `get_last_modified()`) the request headers `If-Modified-Since` and `If-Unmodified-Since` are verified. May abort the request with 304 or 412 response codes.

Raises

- `webob.exceptions.ResponseException` of status 304 if the `If-Modified-Since` is later than the last modified date.
- `webob.exceptions.ResponseException` of status 412 if the last modified date is later than the `If-Unmodified-Since` header.

assert_condition_md5()

If the `Content-MD5` request header is present in the request it's verified against the MD5 hash of the request body. If they don't match, a 400 HTTP response is returned.

Raises `webob.exceptions.ResponseException` of status 400 if the MD5 hash does not match the body.

assert_conditions()

Handles various HTTP conditions and raises HTTP exceptions to abort the request.

- Content-MD5 request header must match the MD5 hash of the full input (`assert_condition_md5()`).
- If-Match and If-None-Match etags are checked against the ETag of this resource (`assert_condition_etag()`).
- If-Modified-Since and If-Unmodified-Since are checked against the modification date of this resource (`assert_condition_last_modified()`).

Todo

Return a 501 exception when any Content-* headers have been set in the request. (See **RFC 2616**, section 9.6)

---

call_method(*method_name*)

Call an instance method filling in all the method parameters based on their names. The parameters are filled in from the following locations (in that order of precedence):

1. Path parameters from routing
2. GET parameters
3. POST parameters

All values are validated using the method `validate_param()`. The return value of the method is returned unaltered.

Parameters **method_name** (*str*) – Name of the method on the current instance to call.

clean_etag(*etag*)

Cleans the ETag as returned by `get_etag()`. Will wrap it in quotes and append the extension for the current MIME type.

convert_param(*method*, *param*, *value*)

Converts the parameter using the function 'convert' function of the validation rules. Same parameters as the *validate_param* method, so it might have just been added there. But lumping together the two functionalities would make overwriting harder.

Parameters

- **method** (*Python function*) – A function to get the validation information from (done using `_get_validation()`).
- **param** (*str*) – Name of the parameter to validate the value for.
- **value** (*Any valid Python value*) – Value passed in for the given parameter.

Raises `wsgiservice.exceptions.ValidationException` if the value is invalid for the given method and parameter.

convert_response()

Finish filling the instance's response object so it's ready to be served to the client. This includes converting the body_raw property to the content type requested by the user if necessary.

data

Returns the request data as a dictionary.

Merges the path parameters, GET parameters and POST parameters (form-encoded or JSON dictionary). If a key is present in multiple of these, the first one defined is used.

get_allowed_methods()

Returns a coma-separated list of method names that are allowed on this instance. Useful to set the `Allowed` response header.

get_content_type()
Returns the Content Type to serve from either the extension or the Accept headers. Uses the EXTENSION_MAP list for all the configured MIME types.

get_etag()
Returns a string to be used as the ETag for this resource. Used to set the ETag response headers and for conditional requests using the If-Match and If-None-Match request headers.

get_last_modified()
Return a datetime.datetime object of the when the resource was last modified. Used to set the Last-Modified response header and for conditional requests using the If-Modified-Since and If-Unmodified-Since request headers.

Return type datetime.datetime

get_method(*method=None*)
Returns the method to call on this instance as a string. Raises a HTTP exception if no method can be found. Aborts with a 405 status code for known methods (based on the KNOWN_METHODS list) and a 501 status code for all other methods.

Parameters **method** (*str*) – Name of the method to return. Must be all-uppercase.

Raises webob.exceptions.ResponseException of status 405 or 501 if the method is not implemented on this resource.

get_request_data()
Read the input values.

Returns a list of dictionaries. These will be used to automatically pass them into the method.

Additionally a combined dictionary is written to *self.data*.

In the case of JSON input, that element in this list will be the parsed JSON value. That may not be a dictionary.

get_resource(*resource*, ***kwargs*)
Returns a new instance of the resource class passed in as resource. This is a helper to make future-compatibility easier when new arguments get added to the constructor.

Parameters

- **resource** (Resource) – Resource class to instantiate. Gets called with the named arguments as required for the constructor.
- **kwargs** (*dict*) – Additional named arguments to pass to the constructor function.

handle_exception(*e*, *status=500*)
Handle the given exception. Log, sets the response code and output the exception message as an error message.

Parameters

- **e** (Exception) – Exception which is being handled.
- **status** (*int*) – Status code to set.

handle_exception_404(*e*)
Handle the given exception. Log, sets the response code to 404 and output the exception message as an error message.

Parameters **e** (Exception) – Exception which is being handled.

handle_ignored_resources()
Ignore robots.txt and favicon.ico GET requests based on a list of absolute paths in IGNORED_PATHS. Aborts the request with a 404 status code.

This is mostly a usability issue to avoid extra log entries for resources we are not interested in.

Raises webob.exceptions.ResponseException of status 404 if the resource is ignored.

set_response_content_md5()
Set the Content-MD5 response header. Calculated from the the response body by creating the MD5 hash from it.

set_response_content_type()
Set the Content-Type in the response. Uses the type instance attribute which was set by get_content_type(). Also declares a UTF-8 charset.

set_response_headers()
Sets all the calculated response headers.

to_application_json(*raw*)
Returns the JSON version of the given raw Python object.

Parameters **raw** (*Any valid Python value*) – The return value of the resource method.

Return type string

to_text_xml(*raw*)
Returns the XML string version of the given raw Python object. Uses _get_xml_value() which applies some heuristics for converting data to XML.

The default root tag is 'response', but that can be overwritten by changing the XML_ROOT_TAG instance variable.

Uses wsgiservice.xmlserializer.dumps() for the actual work.

Parameters **raw** (*Any valid Python value*) – The return value of the resource method.

Return type string

validate_param(*method*, *param*, *value*)
Validates the parameter according to the configurations in the _validations dictionary of either the method or the instance. This dictionaries are written by the decorator wsgiservice.decorators.validate().

Todo

Allow validation by type (e.g. header, post, query, etc.)

Parameters

- **method** (*Python function*) – A function to get the validation information from (done using _get_validation()).
- **param** (*str*) – Name of the parameter to validate the value for.
- **value** (*Any valid Python value*) – Value passed in for the given parameter.

Raises wsgiservice.exceptions.ValidationException if the value is invalid for the given method and parameter.

`Help` class

class `wsgiservice.resource.`**`Help`**(*request*, *response*, *path_params*, *application=None*)
Bases: `wsgiservice.resource.Resource`

Provides documentation for all resources of the current application.

Todo

Allow documentation of output.

Todo

Use first sentence of docstring for summary, add bigger version at the bottom.

`GET`()
Returns documentation for the application.

`to_text_html`(*raw*)
Returns the HTML string version of the given raw Python object. Hard-coded to return a nicely-presented service information document.

Parameters **raw** (*Any valid Python object*) – The return value of the resource method.

Return type string

Todo

Treat paragraphs and/or newlines better in output.

`to_text_html_methods`(*retval*, *resource*)
Add the methods of this resource to the HTML output.

Parameters

- **retval** (*list*) – The list of strings which is used to collect the HTML response.
- **resource** (*Dictionary*) – The documentation of one resource.

`to_text_html_overview`(*retval*, *raw*)
Add the overview table to the HTML output.

Parameters

- **retval** (*list*) – The list of strings which is used to collect the HTML response.
- **raw** (*Dictionary*) – The original return value of this resources `GET()` method.

`to_text_html_resources`(*retval*, *raw*)
Add the resources details to the HTML output.

Parameters

- **retval** (*list*) – The list of strings which is used to collect the HTML response.
- **raw** (*Dictionary*) – The original return value of this resources `GET()` method.

1.3.2 `decorators`

`wsgiservice.decorators.`**`expires`**(*duration*, *vary=None*, *currtime=<built-in function time>*)
Decorator. Apply on a `wsgiservice.Resource` method to set the max-age cache control parameter to the

given duration. Also calculates the correct `Expires` response header.

Parameters

- **duration** (`datetime.timedelta`) – Age which this resource may have before becoming stale.
- **vary** (*list of strings*) – List of headers that should be added to the Vary response header.
- **currtime** (Function returning a `time.struct_time`) – Function used to find out the current UTC time. This is used for testing and not required in production code.

`wsgiservice.decorators.`**`mount`**(*path*)

Decorator. Apply on a `wsgiservice.Resource` to mount it at the given path. The same can be achieved by setting the `_path` attribute on the class directly.

Parameters **path** – A path to mount this resource on. See `wsgiservice.routing.Router` for a description of how this path has to be formatted.

`wsgiservice.decorators.`**`validate`**(*name*, *re=None*, *convert=None*, *doc=None*)

Decorator. Apply on a `wsgiservice.Resource` or any of it's methods to validates a parameter on input. When a parameter does not validate, a `wsgiservice.exceptions.ValidationException` exception will be thrown.

Parameters

- **name** (*string*) – Name of the input parameter to validate.
- **re** (*regular expression*) – Regular expression to search for in the input parameter. If this is not set, just validates if the parameter has been set.
- **convert** (*callable*) – Callable to convert the validated parameter value to the final data type. Ideal candidates for this are the built-ins int or float functions. If the function raises a ValueError, this is reported to the client as a 400 error.
- **doc** (*string*) – Parameter description for the API documentation.

1.3.3 `status`

Helper methods to raise responses for the various HTTP status codes.

The motivation for these methods is to be able to easily document the HTTP response headers which are highly recommended or required. For example the `Location` header which should be set for 201 responses.

The following status codes don't have a method here:

`wsgiservice.status.`**`raise_200`**(*instance*)

Abort the current request with a 200 (OK) response code.

Parameters **instance** (`webob.resource.Resource`) – Resource instance (used to access the response)

Raises `webob.exceptions.ResponseException` of status 200

`wsgiservice.status.`**`raise_201`**(*instance*, *location*)

Abort the current request with a 201 (Created) response code. Sets the Location header correctly. If the location does not start with a slash, the path of the current request is prepended.

Parameters **instance** (`webob.resource.Resource`) – Resource instance (used to access the response)

Raises `webob.exceptions.ResponseException` of status 201

wsgiservice.status.**raise_202**(*instance*)
Abort the current request with a 202 (Accepted) response code.

Parameters **instance** (webob.resource.Resource) – Resource instance (used to access the response)

Raises webob.exceptions.ResponseException of status 202

wsgiservice.status.**raise_204**(*instance*)
Abort the current request with a 204 (No Content) response code. Clears out the body of the response.

Parameters **instance** (webob.resource.Resource) – Resource instance (used to access the response)

Raises webob.exceptions.ResponseException of status 204

wsgiservice.status.**raise_205**(*instance*)
Abort the current request with a 205 (Reset Content) response code. Clears out the body of the response.

Parameters **instance** (webob.resource.Resource) – Resource instance (used to access the response)

Raises webob.exceptions.ResponseException of status 205

wsgiservice.status.**raise_300**(*instance*)
Abort the current request with a 300 (Multiple Choices) response code.

Parameters **instance** (webob.resource.Resource) – Resource instance (used to access the response)

Raises webob.exceptions.ResponseException of status 300

wsgiservice.status.**raise_301**(*instance*, *location*)
Abort the current request with a 301 (Moved Permanently) response code. Sets the Location header correctly. If the location does not start with a slash, the path of the current request is prepended.

Parameters **instance** (webob.resource.Resource) – Resource instance (used to access the response)

Raises webob.exceptions.ResponseException of status 301

wsgiservice.status.**raise_302**(*instance*, *location*)
Abort the current request with a 302 (Found) response code. Sets the Location header correctly. If the location does not start with a slash, the path of the current request is prepended.

Parameters **instance** (webob.resource.Resource) – Resource instance (used to access the response)

Raises webob.exceptions.ResponseException of status 302

wsgiservice.status.**raise_303**(*instance*, *location*)
Abort the current request with a 303 (See Other) response code. Sets the Location header correctly. If the location does not start with a slash, the path of the current request is prepended.

Parameters **instance** (webob.resource.Resource) – Resource instance (used to access the response)

Raises webob.exceptions.ResponseException of status 303

wsgiservice.status.**raise_304**(*instance*)
Abort the current request with a 304 (Not Modified) response code. Clears out the body of the response.

Parameters **instance** (webob.resource.Resource) – Resource instance (used to access the response)

Raises webob.exceptions.ResponseException of status 304

wsgiservice.status.**raise_305**(*instance*, *location*)
Abort the current request with a 305 (Use Proxy) response code. Sets the Location header correctly. If the location does not start with a slash, the path of the current request is prepended.

Parameters **instance** (webob.resource.Resource) – Resource instance (used to access the response)

Raises webob.exceptions.ResponseException of status 305

wsgiservice.status.**raise_307**(*instance*, *location*)
Abort the current request with a 307 (Temporary Redirect) response code. Sets the Location header correctly. If the location does not start with a slash, the path of the current request is prepended.

Parameters **instance** (webob.resource.Resource) – Resource instance (used to access the response)

Raises webob.exceptions.ResponseException of status 307

wsgiservice.status.**raise_400**(*instance*, *msg=None*)
Abort the current request with a 400 (Bad Request) response code. If the message is given it's output as an error message in the response body (correctly converted to the requested MIME type).

Parameters **instance** (webob.resource.Resource) – Resource instance (used to access the response)

Raises webob.exceptions.ResponseException of status 400

wsgiservice.status.**raise_401**(*instance*, *authenticate*, *msg=None*)
Abort the current request with a 401 (Unauthorized) response code. If the message is given it's output as an error message in the response body (correctly converted to the requested MIME type). Outputs the WWW-Authenticate header as given by the authenticate parameter.

Parameters **instance** (webob.resource.Resource) – Resource instance (used to access the response)

Raises webob.exceptions.ResponseException of status 401

wsgiservice.status.**raise_402**(*instance*, *msg=None*)
Abort the current request with a 402 (Payment Required) response code. If the message is given it's output as an error message in the response body (correctly converted to the requested MIME type).

Parameters **instance** (webob.resource.Resource) – Resource instance (used to access the response)

Raises webob.exceptions.ResponseException of status 402

wsgiservice.status.**raise_403**(*instance*, *msg=None*)
Abort the current request with a 403 (Forbidden) response code. If the message is given it's output as an error message in the response body (correctly converted to the requested MIME type).

Parameters **instance** (webob.resource.Resource) – Resource instance (used to access the response)

Raises webob.exceptions.ResponseException of status 403

wsgiservice.status.**raise_404**(*instance*)
Abort the current request with a 404 (Not Found) response code.

Parameters **instance** (webob.resource.Resource) – Resource instance (used to access the response)

Raises webob.exceptions.ResponseException of status 404

wsgiservice.status.**raise_405**(*instance*)
Abort the current request with a 405 (Method Not Allowed) response code. Sets the `Allow` response header to the return value of the `Resource.get_allowed_methods()` function.

Parameters **instance** (`webob.resource.Resource`) – Resource instance (used to access the response)

Raises `webob.exceptions.ResponseException` of status 405

wsgiservice.status.**raise_406**(*instance*)
Abort the current request with a 406 (Not Acceptable) response code.

Parameters **instance** (`webob.resource.Resource`) – Resource instance (used to access the response)

Raises `webob.exceptions.ResponseException` of status 406

wsgiservice.status.**raise_409**(*instance*)
Abort the current request with a 409 (Conflict) response code.

Parameters **instance** (`webob.resource.Resource`) – Resource instance (used to access the response)

Raises `webob.exceptions.ResponseException` of status 409

wsgiservice.status.**raise_410**(*instance*)
Abort the current request with a 410 (Gone) response code.

Parameters **instance** (`webob.resource.Resource`) – Resource instance (used to access the response)

Raises `webob.exceptions.ResponseException` of status 410

wsgiservice.status.**raise_412**(*instance*, *msg=None*)
Abort the current request with a 412 (Precondition Failed) response code. If the message is given it's output as an error message in the response body (correctly converted to the requested MIME type).

Parameters **instance** (`webob.resource.Resource`) – Resource instance (used to access the response)

Raises `webob.exceptions.ResponseException` of status 412

wsgiservice.status.**raise_415**(*instance*, *msg=None*)
Abort the current request with a 415 (Unsupported Media Type) response code. If the message is given it's output as an error message in the response body (correctly converted to the requested MIME type).

Parameters **instance** (`webob.resource.Resource`) – Resource instance (used to access the response)

Raises `webob.exceptions.ResponseException` of status 415

wsgiservice.status.**raise_500**(*instance*, *msg=None*)
Abort the current request with a 500 (Internal Server Error) response code. If the message is given it's output as an error message in the response body (correctly converted to the requested MIME type).

Parameters **instance** (`webob.resource.Resource`) – Resource instance (used to access the response)

Raises `webob.exceptions.ResponseException` of status 500

wsgiservice.status.**raise_501**(*instance*)
Abort the current request with a 501 (Not Implemented) response code. Sets the `Allow` response header to the return value of the `Resource.get_allowed_methods()` function.

Parameters **instance** (`webob.resource.Resource`) – Resource instance (used to access the response)

Raises `webob.exceptions.ResponseException` of status 501

`wsgiservice.status.`**`raise_503`**(*instance*)
Abort the current request with a 503 (Service Unavailable) response code.

Parameters **instance** (`webob.resource.Resource`) – Resource instance (used to access the response)

Raises `webob.exceptions.ResponseException` of status 503

1.3.4 application

Components responsible for building the WSGI application.

class `wsgiservice.application.`**`Application`**(*resources*)
WSGI application wrapping a set of WsgiService resources. This class can be used as a WSGI application according to **PEP 333**.

Parameters **resources** – A list of `wsgiservice.Resource` classes to be served by this application.

Todo

Easy deployment using good configuration file handling

`wsgiservice.application.`**`get_app`**(*defs*, *add_help=True*)
Small wrapper function to returns an instance of `Application` which serves the objects in the defs. Usually this is called with return value globals() from the module where the resources are defined. The returned WSGI application will serve all subclasses of `wsgiservice.Resource` found in the dictionary.

Parameters

- **defs** (*dict*) – Each `wsgiservice.Resource` object found in the values of this dictionary is used as application resource. The other values are discarded.
- **add_help** (*boolean*) – Whether to add the Help resource which will expose the documentation of this service at /_internal/help

Return type `Application`

1.3.5 exceptions

Declares different exceptions as used throughout WsgiService.

exception `wsgiservice.exceptions.`**`ResponseException`**(*response*)
Wraps a `webob.Response` object to be thrown as an exception.

exception `wsgiservice.exceptions.`**`ValidationException`**(**args*, ***kwargs*)
Exception thrown when a validation fails. See `wsgiservice.decorators.validate()` for it's use.

1.3.6 routing

Implements a simple routing class.

class `wsgiservice.routing.`**`Router`**(*resources*)

Simple routing. Path parameters can be extracted with the syntax `{keyword}` where keyword is the path parameter. That parameter will then be passed on to the called request method.

Parameters **resources** – A list of `wsgiservice.Resource` classes to be routed to.

1.4 WsgiService's implementation of HTTP 1.1

Wherever possible WsgiService follows the HTTP 1.1 standard as defined in **RFC 2616**. There are some features which WsgiService does not implement as outlined in this document. Apart from that all deviations from the standard are considered bugs and should be reported to the Author.

1.4.1 Missing features

The following features are not currently implemented. Usually this is because the need was not currently here or because the web server usually implements them.

Protocol-level details

For big parts WsgiService relies on the web server which serves the application.

This affects the following section of the RFC:

- 10.1: Informational 1xx: Both status codes 100 and 101 are related to protocol level details.
- 10.4.12: 411 Length Required
- 10.4.14: 413 Request Entity Too Large
- 10.4.15: 414 Request-URI Too Long
- 10.4.18: 417 Expectation Failed
- 10.5.6: 505 HTTP Version Not Supported

Range requests

Content ranges are not currently implemented at all. This affects the following section of the RFC:

- 3.12: Range Units
- 10.2.7: 206 Partial Content
- 10.4.17: 416 Requested Range Not Satisfiable

Proxy server

Big parts of the HTTP 1.1 standard are relevant only for proxy servers or gateways.

This affects the following section of the RFC:

- 9.8: TRACE method not implemented as it's mostly relevant because of the `Max-Forwards` request header and it's not clear how useful that's for non-gateways. Implementation would be easy to do however, similar to the existing OPTIONS method.
- 9.9: CONNECT

- 10.2.4: 203 Non-Authoritative Information
- 10.4.8: 407 Proxy Authentication Required
- 10.4.9: 408 Request Timeout
- 10.5.3: 502 Bad Gateway
- 10.5.5: 504 Gateway Timeout
- 13: Caching in HTTP. Most of the chapter is for proxies, though some parts are also relevant for applications and implemented in WsgiService

1.4.2 Open questions

The following details are still untested and undecided:

- 100 (Continue) status: Section 8.2.3 of the RFC. Not sure if this should be implemented, if it is done by the server already, etc.

1.4.3 Possible future directions

Some details can't really be considered missing features but are outlined here. They may be implemented if anybody sees a value in them.

- 503 Service Unavailable: This status code would be ideal to communicate a planned downtime. Especially because it can include a `Retry-After` response header containing the estimated time of the downtime. This could be output automatically for example after touching some file.
- `Accept-Charset` request header: When the client sends this header the service output could be converted automatically. Currently WsgiService always returns UTF-8 and that's probably workable enough.
- Authentication is not very well supported, yet. This might be made easier by some access checks on a resource level.

1.5 Todo

Todo

Return a 501 exception when any Content-* headers have been set in the request. (See **RFC 2616**, section 9.6)

(The *original entry* is located in docstring of wsgiservice.resource.Resource.assert_conditions, line 12.)

Todo

Allow validation by type (e.g. header, post, query, etc.)

(The *original entry* is located in docstring of wsgiservice.resource.Resource.validate_param, line 6.)

Todo

Allow documentation of output.

(The *original entry* is located in docstring of wsgiservice.resource.Help, line 3.)

Todo

Use first sentence of docstring for summary, add bigger version at the bottom.

(The *original entry* is located in docstring of wsgiservice.resource.Help, line 4.)

Todo

Treat paragraphs and/or newlines better in output.

(The *original entry* is located in docstring of wsgiservice.resource.Help.to_text_html, line 8.)

Todo

Easy deployment using good configuration file handling

(The *original entry* is located in docstring of wsgiservice.application.Application, line 7.)

CHAPTER 2

Indices and tables

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

W