# mbtest Documentation

***Release 2.5.0***

**Simon Brunning**

**Feb 16, 2021**

# CONTENTS:

Opinionated Python wrapper & utils for the Mountebank over the wire test double tool.

Includes pytest fixture and PyHamcrest matchers.

CHAPTER

# ONE

# GUIDE

(Work in progress)

## 1.1 Use with Docker

If you want to use your own mountebank service instance (Docker, for example) you have **no need to use npm** requirements.

```
docker run -p 2525:2525 -p IMPOSTER_PORT:IMPOSTER_PORT -d andyrbell/mountebank
```

You can do like this in your [`conftest.py`]:

```
import pytest
from mbtest.server import MountebankServer

@pytest.fixture(scope="session")
def mock_server():
    return MountebankServer(port=2525, host="localhost")
```

Don't forget to open docker ports for mountebank (default 2525) and for each it's imposters.

```
from mbtest.imposters import Imposter, Predicate, Response, Stub

imposter = Imposter(
    Stub(
        Predicate(path="/test") & Predicate(query={}) & Predicate(method="GET"),
        Response(body="sausages")
    ),
    record_requests=True,
    port=IMPOSTER_PORT)

with mock_server(imposter) as ms:
    response = requests.get(f"{imposter.url}/test")
    # Check your request
    print(ms.get_actual_requests())
```

If you don't specify port for Imposter it will be done randomly.

## 1.2 Extra

You can combine your Predicate with `&`(and), `|`(or).

## 1.3 TODO

- Basics
  - Server options
    - Executing
    - Existing server, e.g. docker
  - Running locally, against existing server (e.g. docker)
- Stubs, predicates, responses
  - And and or
  - Options
  - Injection
- Stubbing vs. Mocking
  - Assertions and matchers
- Proxies
  - Record/Playback
- SMTP

CHAPTER

TWO

# API REFERENCE

## 2.1 The *mbtest.server* module

mbtest.server.**mock_server**(*request*, *executable='node_modules/.bin/mb'*, *port=2525*, *timeout=5*, *debug=True*, *allow_injection=True*, *local_only=True*, *data_dir='.mbdb'*)

Pytest fixture, making available a mock server, running one or more imposters, one for each domain being mocked.

Use in a pytest conftest.py fixture as follows:

```
@pytest.fixture(scope="session")
def mock_server(request):
    return server.mock_server(request)
```

Test will look like:

```
def test_an_imposter(mock_server):
    imposter = Imposter(Stub(Predicate(path='/test'),
                             Response(body='sausages')),
                        record_requests=True)

    with mock_server(imposter) as s:
        r = requests.get(f"{imposter.url}/test")

        assert_that(r, is_response().with_status_code(200).and_body("sausages"))
        assert_that(s, had_request(path='/test', method="GET"))
```

**Parameters**

- **request** (FixtureRequest) – Request for a fixture from a test or fixture function.
- **executable** (Union[str, Path]) – Alternate location for the Mountebank executable.
- **port** (int) – Server port.
- **timeout** (int) – specifies how long to wait for the Mountebank server to start.
- **debug** (bool) – Start the server in debug mode, which records all requests. This needs to be *True* for the *mbtest.matchers.had_request()* matcher to work.
- **allow_injection** (bool) – Allow JavaScript injection. If *True*, *local_only* should also be *True*,as per Mountebank security.
- **local_only** (bool) – Accept request only from localhost.
- **data_dir** (Optional[str]) – Persist all operations to disk, in this directory.

**Return type** *ExecutingMountebankServer*

**Returns** Mock server.

**class** mbtest.server.**MountebankServer**(*port*, *scheme='http'*, *host='localhost'*, *imposters_path='imposters'*)

Allow addition of imposters to an already running Mountebank mock server.

Test will look like:

```
def test_an_imposter(mock_server):
    mb = MountebankServer(1234)
    imposter = Imposter(Stub(Predicate(path='/test'),
                             Response(body='sausages')),
                        record_requests=True)

    with mb(imposter):
        r = requests.get(f"{imposter.url}/test")

        assert_that(r, is_response().with_status_code(200).and_body("sausages"))
        assert_that(imposter, had_request(path='/test', method="GET"))
```

Imposters will be torn down when the *with* block is exited.

**Parameters**

- **port** (int) – Server port.
- **scheme** (str) – Server scheme, if not *http*.
- **host** (str) – Server host, if not *localhost*.
- **imposters_path** (str) – Imposters path, if not *imposters*.

**add_imposters**(*definition*)

Add imposters to Mountebank server.

**Parameters** **definition** (*Imposter or list(Imposter)*) – One or more Imposters.

**Return type** None

**delete_imposters**()

**Return type** None

**get_actual_requests**()

**Return type** Iterable[*Request*]

**property server_url**

**Return type** furl

**query_all_imposters**()

Yield all imposters running on the server, including those defined elsewhere.

**Return type** Iterator[*Imposter*]

**class** mbtest.server.**ExecutingMountebankServer**(*executable='node_modules/.bin/mb'*, *port=2525*, *timeout=5*, *debug=True*, *allow_injection=True*, *local_only=True*, *data_dir='.mbdb'*)

A Mountebank mock server, running one or more imposters, one for each domain being mocked.

Test will look like:

```
def test_an_imposter(mock_server):
    mb = ExecutingMountebankServer()
    imposter = Imposter(Stub(Predicate(path='/test'),
                             Response(body='sausages')),
                        record_requests=True)

    with mb(imposter) as s:
        r = requests.get(f"{imposter.url}/test")

        assert_that(r, is_response().with_status_code(200).and_body("sausages"))
        assert_that(s, had_request(path='/test', method="GET"))

    mb.close()
```

The mountebank server will be started when this class is instantiated, and needs to be closed if it's not to be left running. Consider using the *mock_server()* pytest fixture, which will take care of this for you.

**Parameters**

- **executable** (Union[str, Path]) – Optional, alternate location for the Mountebank executable.
- **port** (int) – Server port.
- **timeout** (int) – How long to wait for the Mountebank server to start.
- **debug** (bool) – Start the server in debug mode, which records all requests. This needs to be *True* for the *mbtest.matchers.had_request()* matcher to work.
- **allow_injection** (bool) – Allow JavaScript injection. If *True*, *local_only* should also be *True*,as per Mountebank security.
- **local_only** (bool) – Accept request only from localhost.
- **data_dir** (Optional[str]) – Persist all operations to disk, in this directory.

**running: Set[int] = {}**

**start_lock = <unlocked _thread.lock object>**

**close**()

**Return type** None

**exception** mbtest.server.**MountebankException**
Exception using Mountebank server.

**exception** mbtest.server.**MountebankPortInUseException**
Mountebank server failed to start - port already in use.

**exception** mbtest.server.**MountebankTimeoutError**
Mountebank server failed to start in time.

## 2.2 The *mbtest.imposters.imposters* module

**class** mbtest.imposters.imposters.**Imposter**(*stubs*, *port=None*, *protocol=<Protocol.HTTP: 'http'>*, *name=None*, *record_requests=True*, *mutual_auth=False*, *key=None*, *cert=None*)

Represents a Mountebank imposter. Think of an imposter as a mock website, running a protocol, on a specific port. Required behaviors are specified using stubs.

**Parameters**

- **stubs** (Union[*Stub*, Iterable[*Stub*]]) – One or more Stubs.
- **port** (Optional[int]) – Port.
- **protocol** (*Protocol*) – Protocol to run on.
- **name** (Optional[str]) – Imposter name - useful for interactive exploration of imposters on http://localhost:2525/imposters
- **record_requests** (bool) – Record requests made against this imposter, so they can be asserted against later.
- **mutual_auth** (bool) – Server will request a client certificate.
- **key** (Optional[str]) – SSL server certificate.
- **cert** (Optional[str]) – SSL server certificate.

**class Protocol**(*value*)

Imposter Protocol.

**HTTP = 'http'**

**HTTPS = 'https'**

**SMTP = 'smtp'**

**TCP = 'tcp'**

**property url**

**Return type** furl

**as_structure**()

Converted to a JSON serializable structure.

**Return type** Any

**Returns** Structure suitable for JSON serialisation.

**classmethod from_structure**(*structure*)

Converted from a JSON serializable structure.

**Parameters** **structure** (Any) – JSON structure to be converted.

**Return type** *Imposter*

**Returns** Converted object.

**get_actual_requests**()

**Return type** Sequence[*Request*]

**attach**(*host*, *port*, *server_url*)

Attach imposter to a running MB server.

**Return type** None

**property attached**
Imposter is attached to a running MB server.

**Return type** bool

**property configuration_url**

**Return type** furl

**query_all_stubs**()
Return all stubs running on the impostor, including those defined elsewhere.

**playback**()

**Return type** Sequence[*Stub*]

**add_stubs**(*definition*, *index=None*)

**class** mbtest.imposters.imposters.**Request**

**static from_json**(*json*)

**Return type** *Request*

**class** mbtest.imposters.imposters.**HttpRequest**(*method*, *path*, *query*, *headers*, *body*, ***kwargs*)

**static from_json**(*json*)

**Return type** *HttpRequest*

**class** mbtest.imposters.imposters.**Address**(*address*, *name*)

**property address**
Alias for field number 0

**property name**
Alias for field number 1

**class** mbtest.imposters.imposters.**SentEmail**(*from_*, *to*, *cc*, *bcc*, *subject*, *text*, ***kwargs*)

**static from_json**(*json*)

**Return type** *SentEmail*

mbtest.imposters.imposters.**smtp_imposter**(*name='smtp'*, *record_requests=True*)
Canned SMTP server imposter.

**Return type** *Imposter*

## 2.3 The *mbtest.imposters.stubs* module

**class** mbtest.imposters.stubs.**Stub**(*predicates=None*, *responses=None*)
Represents a Mountebank stub. Think of a stub as a behavior, triggered by a matching predicate.

**Parameters**

- **predicates** (Union[*BasePredicate*, Iterable[*BasePredicate*], None]) – Trigger this stub if one of these predicates matches the request

- **responses** (Union[*BaseResponse*, Iterable[*BaseResponse*], None]) – Use these response behaviors (in order)

**as_structure**()
Converted to a JSON serializable structure.

**Return type** Any

**Returns** Structure suitable for JSON serialisation.

**classmethod from_structure**(*structure*)
Converted from a JSON serializable structure.

**Parameters** **structure** (Any) – JSON structure to be converted.

**Return type** *Stub*

**Returns** Converted object.

**class** mbtest.imposters.stubs.**AddStub**(*stub=None*, *index=None*)
Represents a *Mountebank add stub request <http://www.mbtest.org/docs/api/overview#add-stub>*. To add new stab to an existing imposter.

**Parameters**

- **index** (Optional[int]) – The index in imposter stubs array. If you leave off the index field, the stub will be added to the end of the existing stubs array.
- **stub** (Optional[*Stub*]) – The stub that will be added to the existing stubs array

**as_structure**()
Converted to a JSON serializable structure.

**Return type** Any

**Returns** Structure suitable for JSON serialisation.

**static from_structure**(*structure*)
Converted from a JSON serializable structure.

**Parameters** **structure** (Any) – JSON structure to be converted.

**Return type** *AddStub*

**Returns** Converted object.

## 2.4 The *mbtest.imposters.predicates* module

**class** mbtest.imposters.predicates.**BasePredicate**

**classmethod from_structure**(*structure*)
Converted from a JSON serializable structure.

**Parameters** **structure** (Any) – JSON structure to be converted.

**Return type** *BasePredicate*

**Returns** Converted object.

**class** mbtest.imposters.predicates.**LogicallyCombinablePredicate**

**class** mbtest.imposters.predicates.**Predicate**(*path=None*, *method=None*, *query=None*, *body=None*, *headers=None*, *xpath=None*, *operator=<Operator.EQUALS: 'equals'>*, *case_sensitive=True*)

Represents a Mountebank predicate. A predicate can be thought of as a trigger, which may or may not match a request.

**Parameters**

- **path** (Union[str, furl, None]) – URL path.
- **method** (Optional[*Method*]) – HTTP method.
- **query** (Optional[Mapping[str, Union[str, int, bool]]]) – Query arguments, keys and values.
- **body** (Optional[str]) – Body text. Can be a string, or a JSON serialisable data structure.
- **headers** (Optional[Mapping[str, str]]) – Headers, keys and values.
- **xpath** (Optional[str]) – xpath query
- **operator** (*Operator*) –
- **case_sensitive** (bool) –

**exception InvalidPredicateOperator**

**class Method**(*value*)

Predicate HTTP method.

**DELETE = 'DELETE'**

**GET = 'GET'**

**HEAD = 'HEAD'**

**POST = 'POST'**

**PUT = 'PUT'**

**PATCH = 'PATCH'**

**class Operator**(*value*)

Predicate operator.

**EQUALS = 'equals'**

**DEEP_EQUALS = 'deepEquals'**

**CONTAINS = 'contains'**

**STARTS_WITH = 'startsWith'**

**ENDS_WITH = 'endsWith'**

**MATCHES = 'matches'**

**EXISTS = 'exists'**

**classmethod has_value**(*name*)

**Return type** bool

**as_structure**()

Converted to a JSON serializable structure.

**Return type** Any

**Returns** Structure suitable for JSON serialisation.

**classmethod from_structure**(*structure*)
Converted from a JSON serializable structure.

**Parameters** **structure** (Any) – JSON structure to be converted.

**Return type** *Predicate*

**Returns** Converted object.

**fields_from_structure**(*inner*)

**fields_as_structure**()

**class** mbtest.imposters.predicates.**AndPredicate**(*left*, *right*)

**as_structure**()
Converted to a JSON serializable structure.

**Return type** Any

**Returns** Structure suitable for JSON serialisation.

**classmethod from_structure**(*structure*)
Converted from a JSON serializable structure.

**Parameters** **structure** (Any) – JSON structure to be converted.

**Return type** *AndPredicate*

**Returns** Converted object.

**class** mbtest.imposters.predicates.**OrPredicate**(*left*, *right*)

**as_structure**()
Converted to a JSON serializable structure.

**Return type** Any

**Returns** Structure suitable for JSON serialisation.

**classmethod from_structure**(*structure*)
Converted from a JSON serializable structure.

**Parameters** **structure** (Any) – JSON structure to be converted.

**Return type** *OrPredicate*

**Returns** Converted object.

**class** mbtest.imposters.predicates.**NotPredicate**(*inverted*)

**as_structure**()
Converted to a JSON serializable structure.

**Return type** Any

**Returns** Structure suitable for JSON serialisation.

**classmethod from_structure**(*structure*)
Converted from a JSON serializable structure.

**Parameters** **structure** (Any) – JSON structure to be converted.

**Return type** *NotPredicate*

**Returns** Converted object.

**class** mbtest.imposters.predicates.**TcpPredicate**(*data*)

Represents a Mountebank TCP predicate. A predicate can be thought of as a trigger, which may or may not match a request.

**Parameters** **data** (str) – Data to match the request.

**as_structure**()

Converted to a JSON serializable structure.

**Return type** Any

**Returns** Structure suitable for JSON serialisation.

**classmethod from_structure**(*structure*)

Converted from a JSON serializable structure.

**Parameters** **structure** (Any) – JSON structure to be converted.

**Return type** *TcpPredicate*

**Returns** Converted object.

**class** mbtest.imposters.predicates.**InjectionPredicate**(*inject*)

Represents a Mountebank injection predicate. A predicate can be thought of as a trigger, which may or may not match a request.

Injection requires Mountebank version 2.0 or higher.

**Parameters** **inject** (str) – JavaScript function to inject.

**classmethod from_structure**(*structure*)

Converted from a JSON serializable structure.

**Parameters** **structure** (Any) – JSON structure to be converted.

**Return type** *InjectionPredicate*

**Returns** Converted object.

## 2.5 The *mbtest.imposters.responses* module

**class** mbtest.imposters.responses.**BaseResponse**

**classmethod from_structure**(*structure*)

Converted from a JSON serializable structure.

**Parameters** **structure** (Any) – JSON structure to be converted.

**Return type** *BaseResponse*

**Returns** Converted object.

**class** mbtest.imposters.responses.**Response**(*body=''*, *status_code=200*, *wait=None*, *repeat=None*, *headers=None*, *mode=None*, *copy=None*, *decorate=None*, *lookup=None*, *shell_transform=None*)

Represents a Mountebank 'is' response behavior.

**Parameters**

- **body** (str) – Body text for response. Can be a string, or a JSON serialisable data structure.
- **status_code** (Union[int, str]) – HTTP status code
- **wait** (Union[int, str, None]) – Add latency, in ms.
- **repeat** (Optional[int]) – Repeat this many times before moving on to next response.
- **headers** (Optional[Mapping[str, str]]) – Response HTTP headers
- **mode** (Optional[*Mode*]) – Mode - text or binary
- **copy** (Optional[*Copy*]) – Copy behavior
- **decorate** (Optional[str]) – Decorate behavior.
- **lookup** (Optional[*Lookup*]) – Lookup behavior
- **shell_transform** (Union[str, Iterable[str], None]) – shellTransform behavior

**class Mode**(*value*)

An enumeration.

**TEXT = 'text'**

**BINARY = 'binary'**

**property body**

**Return type** str

**as_structure**()

Converted to a JSON serializable structure.

**Return type** Any

**Returns** Structure suitable for JSON serialisation.

**classmethod from_structure**(*structure*)

Converted from a JSON serializable structure.

**Parameters** **structure** (Any) – JSON structure to be converted.

**Return type** *Response*

**Returns** Converted object.

**fields_from_structure**(*structure*)

**Return type** None

**class** mbtest.imposters.responses.**TcpResponse**(*data*)

**as_structure**()

Converted to a JSON serializable structure.

**Return type** Any

**Returns** Structure suitable for JSON serialisation.

**classmethod from_structure**(*structure*)

Converted from a JSON serializable structure.

**Parameters** **structure** (Any) – JSON structure to be converted.

**Return type** *TcpResponse*

**Returns** Converted object.

**class** mbtest.imposters.responses.**Proxy**(*to*, *wait=None*, *inject_headers=None*, *mode=<Mode.ONCE: 'proxyOnce'>*, *predicate_generators=None*)

Represents a Mountebank proxy.

**Parameters** **to** (Union[furl, str]) – The origin server, to which the request should proxy.

**class Mode**(*value*)

Defines the replay behavior of the proxy.

**ONCE = 'proxyOnce'**

**ALWAYS = 'proxyAlways'**

**TRANSPARENT = 'proxyTransparent'**

**as_structure**()

Converted to a JSON serializable structure.

**Return type** Any

**Returns** Structure suitable for JSON serialisation.

**classmethod from_structure**(*structure*)

Converted from a JSON serializable structure.

**Parameters** **structure** (Any) – JSON structure to be converted.

**Return type** *Proxy*

**Returns** Converted object.

**class** mbtest.imposters.responses.**PredicateGenerator**(*path=False*, *query=False*, *operator=<Operator.EQUALS: 'equals'>*, *case_sensitive=True*)

**as_structure**()

Converted to a JSON serializable structure.

**Return type** Any

**Returns** Structure suitable for JSON serialisation.

**classmethod from_structure**(*structure*)

Converted from a JSON serializable structure.

**Parameters** **structure** (Any) – JSON structure to be converted.

**Return type** *PredicateGenerator*

**Returns** Converted object.

**class** mbtest.imposters.responses.**InjectionResponse**(*inject*)

Represents a Mountebank injection response.

Injection requires Mountebank version 2.0 or higher.

**Parameters** **inject** (str) – JavaScript function to inject .

**classmethod from_structure**(*structure*)

Converted from a JSON serializable structure.

**Parameters** **structure** (Any) – JSON structure to be converted.

**Return type** *InjectionResponse*

**Returns** Converted object.

## 2.6 The *mbtest.imposters.behaviors.copy* module

**class** mbtest.imposters.behaviors.copy.**Copy**(*from_*, *into*, *using*)
Represents a copy behavior.

**Parameters**

- **from** – The name of the request field to copy from, or, if the request field is an object, then an object specifying the path to the request field.
- **into** (str) – The token to replace in the response with the selected request value.
- **using** (*Using*) – The configuration needed to select values from the response.

**as_structure**()
Converted to a JSON serializable structure.

**Return type** Any

**Returns** Structure suitable for JSON serialisation.

**classmethod from_structure**(*structure*)
Converted from a JSON serializable structure.

**Parameters** **structure** (Any) – JSON structure to be converted.

**Return type** *Copy*

**Returns** Converted object.

## 2.7 The *mbtest.imposters.behaviors.lookup* module

**class** mbtest.imposters.behaviors.lookup.**Lookup**(*key*, *datasource_path*, *datasource_key_column*, *into*)
Represents a lookup behavior.

**Parameters**

- **key** (*Key*) – How to select the key from the request.
- **datasource_path** (Union[str, Path]) – The path to the data source.
- **datasource_key_column** (str) – The header of the column to match against the key.
- **into** (str) – The token to replace in the response with the selected request value.

**as_structure**()
Converted to a JSON serializable structure.

**Return type** Any

**Returns** Structure suitable for JSON serialisation.

**classmethod from_structure**(*structure*)
Converted from a JSON serializable structure.

**Parameters** **structure** (Any) – JSON structure to be converted.

**Return type** *Lookup*

**Returns** Converted object.

**class** mbtest.imposters.behaviors.lookup.**Key**(*from_*, *using*, *index=0*)
The information on how to select the key from the request.

**Parameters**

- **from** – The name of the request field to copy from, or, if the request field is an object, then an object specifying the path to the request field.
- **using** (*Using*) – The configuration needed to select values from the response
- **index** (int) – Index of the iten from the result array to be selected.

**as_structure**()
Converted to a JSON serializable structure.

**Return type** Any

**Returns** Structure suitable for JSON serialisation.

**classmethod from_structure**(*structure*)
Converted from a JSON serializable structure.

**Parameters** **structure** (Any) – JSON structure to be converted.

**Return type** *Key*

**Returns** Converted object.

## 2.8 The *mbtest.imposters.behaviors.using* module

**class** mbtest.imposters.behaviors.using.**Using**(*method*, *selector*)
How to select values from the response.

**Parameters**

- **method** (*Method*) – The method used to select the value(s) from the request.
- **selector** (str) – The selector used to select the value(s) from the request.

**class Method**(*value*)
An enumeration.

**REGEX = 'regex'**

**XPATH = 'xpath'**

**JSONPATH = 'jsonpath'**

**as_structure**()
Converted to a JSON serializable structure.

**Return type** Any

**Returns** Structure suitable for JSON serialisation.

**classmethod from_structure**(*structure*)
Converted from a JSON serializable structure.

**Parameters** **structure** (Any) – JSON structure to be converted.

**Return type** *Using*

**Returns** Converted object.

**class** mbtest.imposters.behaviors.using.**UsingRegex**(*selector*, *ignore_case=False*, *multiline=False*)

Select values from the response using a regular expression.

**Parameters**

- **selector** (str) – The selector used to select the value(s) from the request.
- **ignore_case** (bool) – Uses a case-insensitive regular expression
- **multiline** (bool) – Uses a multiline regular expression

**as_structure**()

Converted to a JSON serializable structure.

**Return type** Any

**Returns** Structure suitable for JSON serialisation.

**classmethod from_structure**(*structure*)

Converted from a JSON serializable structure.

**Parameters** **structure** (Any) – JSON structure to be converted.

**Return type** *UsingRegex*

**Returns** Converted object.

**class** mbtest.imposters.behaviors.using.**UsingXpath**(*selector*, *ns=None*)

Select values from the response using an xpath expression.

**Parameters**

- **selector** (str) – The selector used to select the value(s) from the request.
- **ns** (Optional[Mapping[str, str]]) – The ns object maps namespace aliases to URLs

**as_structure**()

Converted to a JSON serializable structure.

**Return type** Any

**Returns** Structure suitable for JSON serialisation.

**classmethod from_structure**(*structure*)

Converted from a JSON serializable structure.

**Parameters** **structure** (Any) – JSON structure to be converted.

**Return type** *UsingXpath*

**Returns** Converted object.

**class** mbtest.imposters.behaviors.using.**UsingJsonpath**(*selector*)

Select values from the response using a jsonpath expression.

**Parameters** **selector** (str) – The selector used to select the value(s) from the request.

**classmethod from_structure**(*structure*)

Converted from a JSON serializable structure.

**Parameters** **structure** – JSON structure to be converted.

**Return type** *UsingJsonpath*

**Returns** Converted object.

## 2.9 The *mbtest.matchers* module

mbtest.matchers.**had_request**(*method=<hamcrest.core.core.isanything.IsAnything object>*, *path=<hamcrest.core.core.isanything.IsAnything object>*, *query=<hamcrest.core.core.isanything.IsAnything object>*, *headers=<hamcrest.core.core.isanything.IsAnything object>*, *body=<hamcrest.core.core.isanything.IsAnything object>*, *times=<hamcrest.core.core.isanything.IsAnything object>*)

Mountebank server has recorded call matching.

Build criteria with *with_* and *and_* methods:

assert_that(server, had_request().with_path("/test").and_method("GET"))

Available attributes as per parameters.

**Parameters**

- **method** (Union[str, Matcher[str]]) – Request's method matched...
- **path** (Union[furl, str, Matcher[Union[furl, str]]]) – Request's path matched...
- **query** (Union[Mapping[str, str], Matcher[Mapping[str, str]]]) – Request's query matched...
- **headers** (Union[Mapping[str, str], Matcher[Mapping[str, str]]]) – Request's headers matched...
- **body** (Union[str, Matcher[str]]) – Request's body matched...
- **times** (Union[int, Matcher[int]]) – Request's number of times called matched matched...

**Return type** Matcher[Union[*Imposter*, *MountebankServer*]]

**class** mbtest.matchers.**HadRequest**(*method=<hamcrest.core.core.isanything.IsAnything object>*, *path=<hamcrest.core.core.isanything.IsAnything object>*, *query=<hamcrest.core.core.isanything.IsAnything object>*, *headers=<hamcrest.core.core.isanything.IsAnything object>*, *body=<hamcrest.core.core.isanything.IsAnything object>*, *times=<hamcrest.core.core.isanything.IsAnything object>*)

Mountebank server has recorded call matching

**Parameters**

- **method** (Union[str, Matcher[str]]) – Request's method matched...
- **path** (Union[furl, str, Matcher[Union[furl, str]]]) – Request's path matched...
- **query** (Union[Mapping[str, str], Matcher[Mapping[str, str]]]) – Request's query matched...
- **headers** (Union[Mapping[str, str], Matcher[Mapping[str, str]]]) – Request's headers matched...
- **body** (Union[str, Matcher[str]]) – Request's body matched...
- **times** (Union[int, Matcher[int]]) – Request's number of times called matched matched...

**describe_to**(*description*)
Generates a description of the object.

The description may be part of a description of a larger object of which this is just a component, so it should be worded appropriately.

**Parameters** **description** (Description) – The description to be built or appended to.

**Return type** None

**static append_matcher_description**(*field_matcher*, *field_name*, *description*)

**Return type** None

**describe_mismatch**(*actual*, *description*)
Generates a description of why the matcher has not accepted the item.

The description will be part of a larger description of why a matching failed, so it should be concise.

This method assumes that `matches(item)` is `False`, but will not check this.

**Parameters**

- **item** – The item that the Matcher has rejected.
- **mismatch_description** – The description to be built or appended to.

**Return type** None

**with_method**(*method*)

**and_method**(*method*)

**with_path**(*path*)

**and_path**(*path*)

**with_query**(*query*)

**and_query**(*query*)

**with_headers**(*headers*)

**and_headers**(*headers*)

**with_body**(*body*)

**and_body**(*body*)

**with_times**(*times*)

**and_times**(*times*)

mbtest.matchers.**email_sent**(*to=<hamcrest.core.core.isanything.IsAnything object>*, *subject=<hamcrest.core.core.isanything.IsAnything object>*, *body_text=<hamcrest.core.core.isanything.IsAnything object>*)
Mountebank SMTP server was asked to sent email matching:

**Parameters**

- **to** (Union[str, Matcher[str]]) – Email's to field matched...
- **subject** (Union[str, Matcher[str]]) – Email's subject field matched...
- **body_text** (Union[str, Matcher[str]]) – Email's body matched...

**Return type** Matcher[Union[*Imposter*, *MountebankServer*]]

**class** mbtest.matchers.**EmailSent**(*to=<hamcrest.core.core.isanything.IsAnything object>*, *subject=<hamcrest.core.core.isanything.IsAnything object>*, *body_text=<hamcrest.core.core.isanything.IsAnything object>*)

Mountebank SMTP server was asked to sent email matching:

**Parameters**

- **to** (Union[str, Matcher[str]]) – Email's to field matched...
- **subject** (Union[str, Matcher[str]]) – Email's subject field matched...
- **body_text** (Union[str, Matcher[str]]) – Email's body matched...

**describe_to**(*description*)

Generates a description of the object.

The description may be part of a description of a larger object of which this is just a component, so it should be worded appropriately.

**Parameters** **description** (Description) – The description to be built or appended to.

**Return type** None

**describe_mismatch**(*actual*, *description*)

Generates a description of why the matcher has not accepted the item.

The description will be part of a larger description of why a matching failed, so it should be concise.

This method assumes that `matches(item)` is `False`, but will not check this.

**Parameters**

- **item** – The item that the Matcher has rejected.
- **mismatch_description** – The description to be built or appended to.

**Return type** None

**static get_sent_email**(*actual*)

**Return type** Sequence[*SentEmail*]

**get_matching_emails**(*sent_email*)

**Return type** Sequence[*SentEmail*]

## 2.10 The *mbtest.imposters.base* module

**class** mbtest.imposters.base.**JsonSerializable**

Object capable of being converted to a JSON serializable structure (using *as_structure()*) or from such a structure ((using *from_structure()*).

**abstract as_structure**()

Converted to a JSON serializable structure.

**Return type** Any

**Returns** Structure suitable for JSON serialisation.

**abstract classmethod from_structure**(*structure*)

Converted from a JSON serializable structure.

**Parameters** **structure** (Any) – JSON structure to be converted.

**Return type** *JsonSerializable*

**Returns** Converted object.

**static add_if_true**(*dictionary*, *key*, *value*)

**Return type** None

**set_if_in_dict**(*dictionary*, *key*, *name*)

**Return type** None

**class** mbtest.imposters.base.**Injecting**(*inject*)

**as_structure**()

Converted to a JSON serializable structure.

**Return type** Any

**Returns** Structure suitable for JSON serialisation.

CHAPTER

THREE

# INDICES AND TABLES

- genindex
- modindex
- search

CHAPTER

FOUR

# INSTALLATION

Install from Pypi as usual, using pip , tox, or `setup.py`.

Also requires Mountebank to have been installed:

```
$ npm install mountebank@2.4 --production
```

CHAPTER

FIVE

# USAGE

A basic example:

```
import requests
from hamcrest import assert_that
from brunns.matchers.response import is_response
from mbtest.matchers import had_request
from mbtest.imposters import Imposter, Predicate, Response, Stub

def test_request_to_mock_server(mock_server):
    # Set up mock server with required behavior
    imposter = Imposter(Stub(Predicate(path="/test"),
                             Response(body="sausages")))

    with mock_server(imposter):
        # Make request to mock server - exercise code under test here
        response = requests.get(f"{imposter.url}/test")

        assert_that("We got the expected response",
                    response, is_response().with_status_code(200).and_body("sausages
↪"))
        assert_that("The mock server recorded the request",
                    imposter, had_request().with_path("/test").and_method("GET"))
```

Needs a pytest fixture, most easily defined in conftest.py:

```
import pytest
from mbtest import server

@pytest.fixture(scope="session")
def mock_server(request):
    return server.mock_server(request)
```

CHAPTER

SIX

# INDICES AND TABLES

- genindex
- modindex
- search

# PYTHON MODULE INDEX

## m

# INDEX

## A



## B

## C



## D



## E



## F



## G

## H



## I



## J



## K



## L



## M



## N



## O



## P

## Q



## R



## S



## T



## U



## W



## X