# Response Templating

Response headers and bodies, as well as proxy URLs, can optionally be rendered using [Handlebars templates](http://handlebarsjs.com/). This enables attributes of the request to be used in generating the response e.g. to pass the value of a request ID header as a response header or render an identifier from part of the URL in the response body.

## Enabling response templating

When starting WireMock programmatically, response templating can be enabled by adding ResponseTemplateTransformer as an extension e.g.

@Rule

public WireMockRule wm = new WireMockRule(options()

.extensions(new ResponseTemplateTransformer(false))

);

The boolean constructor parameter indicates whether the extension should be applied globally. If true, all stub mapping responses will be rendered as templates prior to being served.

Otherwise the transformer will need to be specified on each stub mapping by its name response-template:

### Java

wm.stubFor(get(urlPathEqualTo("/templated"))

.willReturn(aResponse()

.withBody("{{request.path.[0]}}")

.withTransformers("response-template")));

### JSON

{

"request": {

"urlPath": "/templated"

},

"response": {

"body": "{{request.path.[0]}}",

"transformers": ["response-template"]

}

}

Command line parameters can be used to enable templating when running WireMock [standalone](http://wiremock.org/docs/running-standalone/#command-line-options).

## Proxying

Templating also works when defining proxy URLs, e.g.

### Java

wm.stubFor(get(urlPathEqualTo("/templated"))

.willReturn(aResponse()

.proxiedFrom("{{request.headers.X-WM-Proxy-Url}}")

.withTransformers("response-template")));

### JSON

{

"request": {

"urlPath": "/templated"

},

"response": {

"proxyBaseUrl": "{{request.headers.X-WM-Proxy-Url}}",

"transformers": ["response-template"]

}

}

## Templated body file

The body file for a response can be selected dynamically by templating the file path:

### Java

wm.stubFor(get(urlPathMatching("/static/.\*"))

.willReturn(ok()

.withBodyFile("files/{{request.pathSegments.[1]}}")));

### JSON

{

"request" : {

"urlPathPattern" : "/static/.\*",

"method" : "GET"

},

"response" : {

"status" : 200,

"bodyFileName" : "files/{{request.pathSegments.[1]}}"

}

}

## The request model

The model of the request is supplied to the header and body templates. The following request attributes are available:

request.url - URL path and query

request.requestLine.path - URL path

request.requestLine.pathSegments.[<n>]- URL path segment (zero indexed) e.g. request.pathSegments.[2]

request.requestLine.query.<key>- First value of a query parameter e.g. request.query.search

request.requestLine.query.<key>.[<n>]- nth value of a query parameter (zero indexed) e.g. request.query.search.[5]

request.requestLine.method- request method e.g. POST

request.requestLine.host- hostname part of the URL e.g. my.example.com

request.requestLine.port- port number e.g. 8080

request.requestLine.scheme- protocol part of the URL e.g. https

request.requestLine.baseUrl- URL up to the start of the path e.g. https://my.example.com:8080

request.headers.<key>- First value of a request header e.g. request.headers.X-Request-Id

request.headers.[<key>]- Header with awkward characters e.g. request.headers.[$?blah]

request.headers.<key>.[<n>]- nth value of a header (zero indexed) e.g. request.headers.ManyThings.[1]

request.cookies.<key> - First value of a request cookie e.g. request.cookies.JSESSIONID

request.cookies.<key>.[<n>] - nth value of a request cookie e.g. request.cookies.JSESSIONID.[2]

request.body - Request body text (avoid for non-text bodies)

## Handlebars helpers

All of the standard helpers (template functions) provided by the [Java Handlebars implementation by jknack](https://github.com/jknack/handlebars.java) plus all of the [string helpers](https://github.com/jknack/handlebars.java/blob/master/handlebars/src/main/java/com/github/jknack/handlebars/helper/StringHelpers.java) are available e.g.

{{capitalize request.query.search}}

## Number and assignment helpers

Variable assignment and number helpers are available:

{{#assign 'myCapitalisedQuery'}}{{capitalize request.query.search}}{{/assign}}

{{isOdd 3}}

{{isOdd 3 'rightBox'}}

{{isEven 2}}

{{isEven 4 'leftBox'}}

{{stripes 3 'row-even' 'row-odd'}}

## XPath helpers

Addiionally some helpers are available for working with JSON and XML.

When the incoming request contains XML, the xPath helper can be used to extract values or sub documents via an XPath 1.0 expression. For instance, given the XML

<outer>

<inner>Stuff</inner>

</outer>

The following will render “Stuff” into the output:

{{xPath request.body '/outer/inner/text()'}}

And given the same XML the following will render <inner>Stuff</inner>:

{{xPath request.body '/outer/inner'}}

As a convenience the soapXPath helper also exists for extracting values from SOAP bodies e.g. for the SOAP document:

<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope/">

<soap:Body>

<m:a>

<m:test>success</m:test>

</m:a>

</soap:Body>

</soap:Envelope>

The following will render “success” in the output:

{{soapXPath request.body '/a/test/text()'}}

## JSONPath helper

It is similarly possible to extract JSON values or sub documents via JSONPath using the jsonPath helper. Given the JSON

{

"outer": {

"inner": "Stuff"

}

}

The following will render “Stuff” into the output:

{{jsonPath request.body '$.outer.inner'}}

And for the same JSON the following will render { "inner": "Stuff" }:

{{jsonPath request.body '$.outer'}}

## Date and time helpers

A helper is present to render the current date/time, with the ability to specify the format ([via Java’s SimpleDateFormat](https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html)) and offset.

{{now}}

{{now offset='3 days'}}

{{now offset='-24 seconds'}}

{{now offset='1 years'}}

{{now offset='10 years' format='yyyy-MM-dd'}}

Dates can be rendered in a specific timezone (the default is UTC):

{{now timezone='Australia/Sydney' format='yyyy-MM-dd HH:mm:ssZ'}}

Pass epoch as the format to render the date as unix epoch time.

{{now offset='2 years' format='epoch'}}

Dates can be parsed from other model elements:

{{date (parseDate request.headers.MyDate) offset='-1 days'}}

## Random value helper

Random strings of various kinds can be generated:

{{randomValue length=33 type='ALPHANUMERIC'}}

{{randomValue length=12 type='ALPHANUMERIC' uppercase=true}}

{{randomValue length=55 type='ALPHABETIC'}}

{{randomValue length=27 type='ALPHABETIC' uppercase=true}}

{{randomValue length=10 type='NUMERIC'}}

{{randomValue length=5 type='ALPHANUMERIC\_AND\_SYMBOLS'}}

{{randomValue type='UUID'}}

## Custom helpers

Custom Handlebars helpers can be registered with the transformer on construction:

Helper<String> stringLengthHelper = new Helper<String>() {

@Override

public Object apply(String context, Options options) throws IOException {

return context.length();

}

};

@Rule

public WireMockRule wm = new WireMockRule(options()

.extensions(new ResponseTemplateTransformer(false), "string-length", stringLengthHelper)

);