Hurl HTTP requests as tests

Request METHOD URL, headers, body → `HTTP` → **Response** HTTP 200 → `[Captures]` → **Variables** {{name}} → `next entry` → **[Asserts]** pass or fail, exit code

CONCEPTS

A .hurl file is a list of entries: a request, then the response you expect.

Entries run in order, so a capture in one is usable as {{name}} in the next.

RUN

`hurl` *`<file.hurl>`*
Run a file

`hurl` *`<a.hurl> <b.hurl>`*
Run several files in order

`hurl --test` *`<files>`*
Test mode: progress and summary

`hurl --glob "`*`<pattern>`***`"`**
Run files matching a glob

`hurl --continue-on-error` *`<file>`*
Do not stop at the first failed assert

`hurl --retry` *`<n>`* **`--retry-interval`** *`<ms>`* *`<file>`*
Retry failed entries

REQUEST

`GET` *`<url>`*
First line: method and URL

REQUEST (cont.)

*`<Name>`***`:`** *`<value>`*
Header lines straight after the URL

`[Query]` *`<key>`***`:`** *`<value>`*
Query string params, unencoded

`[Form]` *`<key>`***`:`** *`<value>`*
URL-encoded form fields

`[BasicAuth]` *`<user>`***`:`** *`<password>`*
Basic authentication

`[Options]` *`<option>`***`:`** *`<value>`*
Per-request options

`{ "key": "value" }`
JSON body, Content-Type set for you

```` ```json ... ``` ````
Multiline body, with templates

RESPONSE & ASSERTS

`HTTP 200`
Expect this status

`[Asserts]`
Section of checks, one per line

`status < 300`
Status check

RESPONSE & ASSERTS (cont.)

`header "`*`<Name>`***`" contains "`***`<text>`***`"`**
Header check

`jsonpath "$.a.b" == "`*`<value>`***`"`**
Check a JSON value

`jsonpath "$.items" count ==` *`<n>`*
Check list length

`xpath "//h1" exists`
Check HTML or XML

`body contains "`*`<text>`***`"`**
Substring of the body

`jsonpath "$.x" not exists`
Negate any predicate with not

PREDICATES

`== ≠ > ≥ < ≤`
Compare values

`startsWith endsWith contains`
String and list checks

`matches /`*`<re>`***`/`**
Regex match

`exists isEmpty`
Presence checks

PREDICATES (cont.)

`isString isInteger isBoolean`
Type checks (also isFloat, isList...)

Prefix any predicate with not, e.g. jsonpath "$.x" not exists.

CAPTURES & VARIABLES

`[Captures]`
Section that stores response values

`id: jsonpath "$.id"`
Capture a JSON value

`tok: header "X-Token"`
Capture a header

`{{name}}`
Use a variable in URL, header, body

`hurl --variable` *`<name>=<value>`*
Set a variable

`hurl --variables-file` *`<file>`*
Variables from name=value lines

`hurl --secret` *`<name>=<value>`*
Variable redacted from logs

HTTP OPTIONS

`-L, --location`
Follow redirects

`-H, --header` *`<Name:value>`*
Add a header to every request

`-u, --user` *`<user:password>`*
Basic authentication

`-k, --insecure`
Skip TLS verification

OUTPUT & REPORTS

`-v, --verbose`
Request and response details on stderr

`--very-verbose`
Debug output with bodies

`-i, --include`
Print response headers too

`-o, --output` *`<file>`*
Write the body to a file

`-h, --help / -V, --version`
Help and version

`--report-html` *`<dir>`*
HTML report

`--report-junit` *`<file>`*
JUnit XML for CI