APS Groovy Tools Lib

User Guide

1.0.0

Tommy Svensson

MapJsonDocValidator

This takes a schema (made up of a `Map<String, Object>`, see below) and another `Map<String, Object>` representing the JSON. So the catch here is that you need a JSON parser that allows you to get the content as a Map. The Vertx JSON parser does. This uses `Map` since it is generic, does not need to hardcode dependency on a specific parser, and maps are very easy to work with in Groovy.

Useage

```
 private Map<String, Object> schema = [
        "meta/header": "meta",
        header_1: [
                type_1      : "service",
                "meta/type" : "metadata",
                address_1   : "?aps\\.admin\\..*",
                classifier_1: "?public|private"
        ],
        body_1  : [
                action_1: "get-webs"
        ],
        reply_0: [
                webs_1: [
                        [
                                name_1: "?.*",
                                url_1: "?^https?://.*",
                                no1_0: "#1-100",
                                no2_0: "#<=10",
                                no3_0: "#>100",
                                no4_0: "#1.2-3.4"
                        ]
                ]
        ]
] as Map<String, Object>

private MapJsonDocValidator verifier = new MapJsonDocValidator( validStructure: schema )

...

verifier.validate(myJsonMap)
```

This will throw a runtime exception on validation failure.

Schema

Keys

<key>_0 - The key is optional.

<key>_1 - The key is required.

Values

"?regexp"

The '?' indicates that the rest of the value is a regular expression. This regular expression will be applied to each value.

"<hash><range>"

This indicates that this is a number and defines the number range allowed. The following variants are available:

"#from-to" : This specifies a range of allowed values, from lowest to highest.

"#<=num" : This specifies that the numeric value must be less than or equal to the specified number.

"#>=num" : This specifies that the numeric value must be larger than or equal to the specified number.

"#<num" : This specifies that the numeric value must be less than the specified number.

"#>num" : This specifies that the numeric value must be larger than the specified number.

Note: Both floating point numbers and integers are allowed.

"bla"

This requires values to be exactly "bla".

Example

```
Map<String, Object> struct = [
   header_1: [
      type_1      : "service",
      address_1   : "aps.admin.web",
      classifier_0: "?public|private"
   ],
   body_1  : [
      action_1: "get-webs"
   ],
   reply_0: [
      webs_1: [
         [
            name_1: "?.*",
            url_0: "?^https?://.*",
            someNumber_0: "#0-100" // Also valid: ( ">0" "<100" ) ( ">=0" "<=100" )
         ]
      ]
   ]
]
```