JSON PATH DESIN

2018 年 6 月 27 日

目录

1 设计思路

1.1 表达式构成

- "/[选择器? 过滤器]" "/{选择器? 过滤器}"
  1. {} 表示 dict, [] 表示 list
  2. 选择器、过滤器、比较器参考下述描述
- 操作过程为
  1. 过滤 (filter): 过滤 list 或 dict 里面的项，结果仍然是一个 dict 或一个 list
  2. 选择 (choose): 选择 list 或 dict 中的元素，或多个元素返回取值内容作为结果

1.2 选择器

选择器用于选择节点

1.2.1 单元素选择器

- 单一选择器执行选择操作后，返回对应的元素 (相当于当前节点切换到下一层)
- 举例
  - /[1] 返回列表的第 2 个元素
  - /[-1] 返回列表的倒数第 1 个元素
  - /{key1} 返回字典 key1 对应的数据

1.2.2 多元素选择器

- 批量选择器执行选择操作后，返回元素列表或字典 (相当于 root 节点仍保留在当前层)
- 举例
  - /[1:-1] 返回列表的第 2 个到最后一个元素, 构成的新列表
  - /{~key1} 返回字典中除 key1 以外的其他所有数据, 结果仍然是字典
  - /{k1,k2,k3} 返回字典中 key1,key2,key3 组成的数据, 结果仍然是字典
  - /{~k1,k2,k3} 返回字典中除 k1,k2,k3 以外的其他数据, 结果仍然是字典

1.2.3 返回下一级内容的选择器

- 获取下一级内容 (下一级内容只支持获取单属性)
- 举例
  - data = [Student1, Student2, Student3] Student 结构为 {'name': xxx, 'sex': xxx, 'score': 80}
  - 返回分数超过 80 分的所有学生 (先过虑, 再选择 name 属性), 则对应的选择器书写为/[?@score>=80]|[@name]
  - /[i:j@k1] /{@k1@k13}

1.3 过滤器

相当于为 filter 函数设计的简写版 lambda 语法, 适用于 list 的元素以及 dict 的元素 filter(list or dict, $\text{lambda}_{\text{expression}}$, sequence))

- 过滤器有如下形式

  - **?op value** 表示每个元素的取值满足 op 的比较条件 (按值筛选)
  - **?$ op value** 表示 dict 的 key 是否取值满足 op 的比较条件, 也就是按 key 筛选
  - **?@k1 op value** 表示每个元素的 k1 属性是否满足 op 的比较条件 (值是字典, 按字典里的 k1 对应的值筛选) 将遍历当前 list 或 dict, 对于每个 item, 找出满足 item.k1 op value 的 item, 组成新的 list 或 dict 返回
  - **?@k1@k13 op value** 表示每个元素的 k1 属性下的 k13 属性值是否满足 op 的比较条件

- 过滤器支持 pipline 操作, 使用符号 | 表示, pipeline 之间是 and 关系实际上 | 等同于/, 可视为语法糖, 仅仅是为了使数据层次性看起来清楚一些

1.4 比较操作 (目的是用于测试元素或子元素是否满足某些条件)

比较操作一般是针对所选择的节点对应的 value 进行比较如 list[0] 的数值与指定值比较 dict1[key1] 与指定值比较仅 $ op value 形式为与键值比较 (当数据为 dict, 且无 @ 相关 subkey 字段时方可生效)

1.4.1 当前支持以下比较类型

- not rin 形式 selector not rin $\text{want}_{\text{value}}$, 如果 $\text{want}_{\text{value}}$ not in selector 所选择的节点取值中, 返回 true, 否则返回 false
- not in 形式 selector not in $\text{want}_{\text{value}}$, 如果 selector 所选择的节点的值 not in $\text{want}_{\text{value}}$, 返回 true, 否则返回 false
- rin 类似 not rin, 只是逻辑变为 in
- in 类似 not in, 只是逻辑变为 in
- >= 大于或等于
- <= 小于或等于
- != 不等于
- = 等于
- > 大于
- < 小于

2 样例

2.1 每次选择以/分割, 形式如下

- /{key1} single return 过滤出当前字典中 key1 对应的数据 (进入下一层)
- /{~key1} batch return 过滤当前字典中除 key1 以外的所有数据 (保留在当前层)
- /[1] single return 过滤出列表第二个数据（进入下一层）
- /[-1] single return 过滤出列表倒数第一个数据（进入下一层）
- /[:-1] batch return 过滤出当前列表从 0 到倒数第二个数据 (保留在当前层)

2.2 实例分析

- '\/{pageInfo}/{list}/[?@guideComment]|[?@brandName= 农夫山泉]'
  1. 选取当前节点下的 pageInfo 节点内容 (当前层为 dict)
  2. 接着选择下一层的 list 节点内容 (当前层为 dict)
  3. 再接着选择下一层的所有 list, 过滤出其子节点 guideComment 不为空的元素 (停留在 list 层)
  4. 接着继续在此 list 层操作, 过滤出其子节点 brandName 等于农夫山泉的所有数据 (返回结果是个 list)
- '\/{brandName}' 返回当前层中 brandName 节点的内容
- '\/{imageUrls}/[?$ rin http)]'
  1. 返回当前层下 imageUrls 节点内容
  2. 接着选择下一层所有节点, 过滤出节点内容包含 http 的元素列表
- '\/{ProductData}/{~id?@Id rin 478}'
  1. 返回当前层 ProductData 节点下的内容
  2. 过滤出除名字为 ~id 的其他所有节点内容, 然后进一步判断这些节点下的 Id 子节点内容, 过滤出内容包含 478 的所有节点