# JSONPath

## A.基本概念介绍

- JSONPath 是 xpath 在 json 的应用。

xml 最大的优点就有大量的工具可以分析，转换和选择性的提取文档中的数据。XPath 是这些最强大的工具之一。

如果使用 xpath 来解析 json，可以解决以下的问题：

- 可以在客户端上以 JSON 结构交互式地找到和提取数据，而无需特殊脚本。
- 客户端请求的 JSON 数据可以减少到服务器上的相关部分，例如最小化服务器响应的带宽使用。

事实上，json 是由 c 系统编程语言表示自然数据，由特定语言的特定语法来访问 json 数据。

例如 xpath 的表达式：/store/book[1]/title

我们可以看作：x.store.book[0].title 或 x['store']['book'][0]['title']

JSONPath 参照 xpath 表达式来解析 xml 文档，json 数据结构通常是匿名的并且不一定需要有根元素。JSONPath 用一个抽象的名字$来表示最外层对象。

## B.表达式

1.JSONPath 表达式可以使用.符号表示：$.store.book[0].title

或者使用[]符号表示：$['store']['book'][0]['title']

从输入路径来看。内部或者输出的路径都会转化成-符号。

2.JSONPath 允许使用通配符 * 表示所有的子元素名和数组索引。还允许使用 '..' 模糊匹配和数组切片语法[start:end:step]。

3.可以使用显示的名称或者索引来表示：

$.store.book[(@.length-1)].title

4.使用'@'符号表示当前的对象，?(<判断表达式>) 使用逻辑表达式来过滤。 $.store.book[?(@.price < 10)].title

5.支持多选操作。

$.store.book[?(@.price < 10)].[price,title]

注释：当前版本的 JSONPath 支持单双引号。

JSONPath 语法元素和对应 XPath 元素的对比

| XPath | JSONPath | Description |
|---|---|---|
| / | $ | 表示根元素 |
| . | @ | 当前元素 |
| / | . or [] | 子元素 |
| .. | n/a | 父元素 |

| | | |
|---|---|---|
| // | .. | 递归下降，JSONPath 是从 ECMAScript for XML （E4X）是一扩展了 ECMAScript （JavaScript）的程式语言借鉴的。 |
| * | * | 通配符，表示所有的元素 |
| @ | n/a | 属性访问字符 |
| [] | [] | 子元素操作符 |
| \| | [,] | Union 操作符在 XPath 可以合并其它结点集合。JSONPath 允许 name 或者数组索引。 |
| n/a | [start:end:step] | 数组分割操作 |
| [] | ?() | 应用过滤表示式 |
| n/a | () | 脚本表达式，使用底层脚本引擎。 |
| () | n/a | Xpath 分组 |

- []在 xpath 表达式总是从前面的路径来操作数组，索引是从 1 开始。
- 使用 JOSNPath 的[]操作符操作一个对象或者数组，索引是从 0 开始。

XPath 还有很多的语法（本地路径，操作符，和函数）没有列在这里。

## C.用法示例

接下我们看 jsonpath 表示的例子。下面是一个简单的 json 数据结构代表一个书店（原始 xml 文件）

```
import jsonpath
d = { "store": {
    "book": [
      { "category": "纪录片",
        "author": "赵",
        "title": "A",
        "price": 8.95
      },
      { "category": "喜剧片",
        "author": "钱",
        "title": "B",
        "price": 12.99
      },
      { "category": "喜剧片",
        "author": "孙",
```

```
            "title": "C",
            "isbn": "0-553-21311-3",
            "price": 8.99
          },
          { "category": "喜剧片",
            "author": "李",
            "title": "D",
            "isbn": "0-395-19395-8",
            "price": 22.99
          }
        ],
        "bicycle": {
          "color": "red",
          "price": 19.95
        }
      }
    }

    tmp1 = jsonpath.jsonpath(d,'$.store.book[*].author') # *通配符，表示所有的元素; 在此表示所有书的作者
    print("tmp1:%s"%tmp1)
    tmp1_1 = jsonpath.jsonpath(d,'$.store') #表示 store 内所有的元素
    print("tmp1_1:%s"%tmp1_1)

    tmp2 = jsonpath.jsonpath(d,'$..author')#所有书的作者， ..表示模糊匹配
    print("tmp2:%s"%tmp2)

    tmp3 = jsonpath.jsonpath(d,'$.store.*') #表示 store 下 book 和 bicycle 的所有元素
    print("tmp3:%s"%tmp3)

    tmp4 = jsonpath.jsonpath(d,'$.store..price') #所有物品的价格
    print("tmp4:%s"%tmp4)

    tmp5 = jsonpath.jsonpath(d,'$..book[2]') #第三本书的所有信息
    print("tmp5:%s"%tmp5)

    tmp6 = jsonpath.jsonpath(d,'$..book[(@.length-1)].title') #最后一本书
    print("tmp6:%s"%tmp6)
    tmp6_1 = jsonpath.jsonpath(d,'$..book[(@.length)]') #没有则输出 False
    print("tmp6_1:%s"%tmp6_1)
```

```
    tmp6_2 = jsonpath.jsonpath(d,'$..book[-1:]') #最后一本书
    print("tmp6_2:%s"%tmp6_2)

    tmp7 = jsonpath.jsonpath(d,'$..book[0,1]') #前两本书  多选操作
    print("tmp7:%s"%tmp7)

    tmp8 = jsonpath.jsonpath(d,'$..book[:2]') #前两本书  切片操作
    print("tmp8:%s"%tmp8)

    tmp9 = jsonpath.jsonpath(d,'$..book[?(@.isbn)].title') #筛选操作 筛选 book 中包含 isbn 的 book
    print("tmp9:%s"%tmp9)

    tmp10 = jsonpath.jsonpath(d,'$..book[?(@.price!=10)].[title,author]') # @表示当前对象 book，逻辑表达式用来筛选
    print("tmp10:%s"%tmp10)

    tmp11 = jsonpath.jsonpath(d,'$..*')
    print("tmp11:%s"%tmp11)
```

| XPath | JSONPath | 结果 |
| --- | --- | --- |
| /store/book/author | $.store.book[*].author | 书店所有书的作者 |
| //author | $..author | 所有的作者 |
| /store/* | $.store.* | store 的所有元素。所有的 book 和 bicycle |
| /store//price | $.store..price | store 里面所有东西的 price |
| //book[3] | $..book[2] | 第三个书 |
| //book[last()] | $..book[(@.length-1)] | 最后一本书 |
| //book[position()<3] | $..book[0,1]<br>$..book[:2] | 前面的两本书。 |
| //book[isbn] | $..book[?(@.isbn)] | 过滤出所有的包含 isbn 的书。 |

| | | |
|---|---|---|
| //book[price<10] | $..book[?(@.price<10)] | 过滤出价格低于 10 的书。 |
| //* | $..* | 所有元素。 |

**注：**上述用例可通过 http://jsonpath.com/网站进行校验，当表达式不能匹配不到相应数据时，页面不会有错误提示，而在 pycharm 中运行打印则会输出 False。