**Json-path 문법**

참고)  Stefan Goessner의 원본 Post (2007) :

<https://goessner.net/articles/JsonPath/>

1. **목적**Json 형식을 가진 데이터에서 원하는 데이터를 추출하기 위해서, 보편적으로 알려진 문법을 차용하여 사용키로 함.  
     
   << 차용의 이유 >>  
   1) Json convert를 위해서는 원하는 데이터를 추출하는 규칙의 표기방법이 필요.  
   2) 매우 다양한 추출 case가 있어서, 단기간에 새로 정의하는 것은 비경제적.  
   3) json-path는 비교적 잘 알려져 있는 문법이므로, 습득 및 활용이 용이할 것으로 판단.
2. 참고) **구현체와 참조 정보**  
   \* Web-front를 위한 java-script 버전과 Java개발자를 위한 Java 버전의 Library가 존재하며, Web브라우져에서 테스트할 수 있는 도구도 존재함. (<https://jsonpath.com/>)

|  |  |
| --- | --- |
| Language | Source, Refence URL |
| JavaScript | <https://www.npmjs.com/package/jsonpath> |
| Java | https://github.com/json-path/JsonPath |
| Scala | <https://github.com/gatling/gatling/tree/main/gatling-jsonpath> |

1. **자체 개발의 이유**
2. 구현체간의 Spec차이  
   세부 스펙에서의 동작이 구현체마다 각기 미묘하게 차이가 나타나며, 또한 허용되지 않는 문법을 사용할 경우 기대치 못한 동작을 하는 것으로 판단됨.
3. 기능확장을 위한 제한  
   특수목적의 기능제공을 위해 문법변형을 가하기 위해서는 자체 Code base를 가질 필요가 있다고 판단.
4. **문법 설명  
   [[ 주요 Token 설명 ]]**1) 문서에서 위치를 나타내는 Token

|  |  |  |
| --- | --- | --- |
| 의미 | Token | 설명 |
| **Position** |  | 탐색중인 노드의 위치를 지정하는 방법 |
| **$** | Json의 Root Node를 가리킴 |
| **@** | 현재 탐색(?) 중인 Node를 가리킴 |
|  | Position을 생략할 수 있음. ( 이 경우, @를 지정한 것으로 간주) |

**2) Json의 Object노드에서 값을 추출하는 문법**

|  |  |  |
| --- | --- | --- |
| **Object Selector** | **.문자열** | Object노드의 Child값을 추출하는 방법 |
|  |  |
| **.key** | key라는 이름을 가진 child의 값을 추출 |
| **.'key name'** | key name'이라는 이름을 가진 child의 값을 추출 |
| **."key name"** | 상동 |
| **['key1','key2']** | key1과 key2의 값들들 (복수 필드값 추출하는 방법) |
| **.\*** | 모든 필드의 값들 |
|  |  |
| **..key** | 노드를 탐색하여 key이름을 가진 **필드의 값**을 추출 |
| **..\*** | 하위의 모든 값들 (값으로 표현가능한 모든 것: Object, Array, String…) |

**3) Json의 Array노드에서 값을 추출하는 문법**

|  |  |  |
| --- | --- | --- |
| **Array Selector** | **[정수:정수:정수]** | Array노드의 Child값을 추출하는 방법 |
| [start:end:step] | (start, end, step 모두 정수) : start-Index에서 end-index전까지 step칸씩 |
|  | 세 값 모두 생략 가능 ( 모두 생략하면 [\*]와 같은 의미) |
| [0:] | 0-index부터 끝까지 (start index는 0부터 시작) |
| [1::2] | 1-index부터 끝까지 2칸씩 (ex) 1, 3, 5.. 번째 값들 ) |
| [-2:] | 끝에서 2개만 추출 ( start index에 음수를 쓸 경우, 끝에서부터의 의미; 끝은 -1) |
| [3, 7, 2] | 3, 7, 2번 index의 값들 |

**4) Json의 Array 또는 Object노드에서 조건에 부합하는 child추출하는 문법**

|  |  |  |
| --- | --- | --- |
| **Predicate Selector** | [**?**(조건)] | Array노드 또는 Obejct노드의 특정조건 기준으로 Child값을 추출하는 방법 |
| [?(@.price)] | price라는 필드를 가진 Child들을 추출 |
| [?(@.price > 10)] | 값이 10보다 큰 price필드를 가진 Child들을 추출 |
| [?(@.price > 10 $$ @.isbn)] | 위의 조건 + isbn필드를 가진 child들을 추출 |
| [?(@.price > $.ref)] | Root.ref의 값보다 큰 price 필드를 가진 child를 추출 |

**5) Predicate 조건절에서 쓸 수 있는 numeric operator들.**

|  |  |  |
| --- | --- | --- |
|  | 연산자 |  |
| 숫자 (lhs) | > | 숫자 (rhs) |
| < |
| >= |
| <= |
| == |
| != |

1. **형변환**lhs의 type으로 rhs의 type을 변환함.  
   정수형 > 실수형 🡺 정수형 > 정수형
2. rhs와 rhs에 **상수를 쓸 경우**소수점이 없으면 정수형으로 소수점이 있으면 실수형으로 취급함.
3. lhs와 rhs가 **json에서 선택한 값**일 경우

만약, 둘 중 하나가 여러 개의 값일 경우, 그 값들 모두가 조건에 맞아야 함.

|  |  |
| --- | --- |
| [1,2,3] > 2 | false |
| 5 > [1,2,3] | true |
| [1,2] > [3,4,5] | False  ( **여러 값들 간의 비교는 계산을 하지 않고, false반환함** ) |

1. lhs와 rhs가 **json에서 선택한 값**일 경우의 타입은?

1.과 같은 방식의 형변환을 하며,  
json값의 숫자형식은 정수(Long, BigInt), 실수(Double, BigDecimal)형으로 처리함.

**6) Predicate 조건절에서 쓸 수 있는 string operator들.**

|  |  |  |
| --- | --- | --- |
|  | 연산자 |  |
| 문자열 (lhs) | > | 문자열 (rhs) |
|
| < |
| >= |
| <= |
| == |
| != |
| =~ |

* 비교연산자 ( >, <, >=, <=, ==, != )은 일반 프로그래밍언어에서의 문자열 비교와 같은 방식으로 처리됨.
* **=~** 는 정규표현식 matching여부를 검사하는 operator **rhs 에 정규표현식을 사용. Quote(또는 double-quote 문자열로 작성)   
  Ignore-case로 matching할 경우에는 (?i)를 정규표현식 앞에 쓰면 됨**

**Ex) $.book[?(.author =~ ‘(?i)\*ree’ )**

book은 object(또는 array)인데, 그 값들 object이고   
ree(ignore-case)로 끝나는 author라는 필드를 가진 것을 추출.

**7)** Predicate 조건절은 **괄호**와 **&&**, **||**를 이용해서 중첩 정의할 수 있음.

1. **간단한 연습 문제  
   ( from :** <https://goessner.net/articles/JsonPath/> )

{ *"store"*: {

*"book"*: [

{ *"category"*: *"reference"*,

*"author"*: *"Nigel Rees"*,

*"title"*: *"Sayings of the Century"*,

*"price"*: 8.95

},

{ *"category"*: *"fiction"*,

*"author"*: *"Evelyn Waugh"*,

*"title"*: *"Sword of Honour"*,

*"price"*: 12.99

},

{ *"category"*: *"fiction"*,

*"author"*: *"Herman Melville"*,

*"title"*: *"Moby Dick"*,

*"isbn"*: *"0-553-21311-3"*,

*"price"*: 8.99

},

{ *"category"*: *"fiction"*,

*"author"*: *"J. R. R. Tolkien"*,

*"title"*: *"The Lord of the Rings"*,

*"isbn"*: *"0-395-19395-8"*,

*"price"*: 22.99

}

],

*"bicycle"*: {

*"color"*: *"red"*,

*"price"*: 19.95

}

}

}

|  |  |  |
| --- | --- | --- |
| **JSONPath** | **Result** | **참고** |
| $.store.book[\*].author | the authors of all books in the store | Root -> store -> book(객체 또는 배열) -> 모든 값(객체) -> author |
| $..author | all authors | Json내의 author 키에 해당하는 값들 |
| $.store.\* | all things in store, which are some books and a red bicycle. | Root -> store(객체 또는 배열)  ->모든 값 |
| $.store..price | the price of everything in the store. | Root -> Store -> (Store이하의 모든) price키에 해당하는 값들 |
| $..book[2] | the third book |  |
| $..book[(@.length-1)] | the last book in order. | 지원하지 않음.  (숫자표기가 더 편리하고 일관성 있으므로) |
| $..book[-1:] |  |
| $..book[0,1] | the first two books |  |
| $..book[:2] |  |
| $..book[?(@.isbn)] | filter all books with isbn number | 모든 book 키에 해당하는 값들 중  (그 값이 객체이고) isbn키가 있는 것. |
| $..book[?(@.price<10)] | filter all books cheapier than 10 |  |
| $..\* | All members of JSON structure. |  |

\*\* Note) 조건절에서 Current-Position을 나타내는 @는 생략해도 됨.  
 (번거로워서 빼도 되도록 했음.)

1. **생각해 보아야 할 사항.**

**[‘key1’, ‘key2’][-1:]**

객체에서 여러 필드를 추출하고(Key1, Key2의 값들), ArraySelect를 할 경우.

어떻게 동작하도록 해야 할 것인지 애매한 상황이 생김.

예를 들어,

key1의 값이 배열이고, (예를 들어 [{“n”: 1}, {“n”: 2}, {“n”: 3}] 이라고 하자.)

key2의 값이 문자열이라면 어떻게 해야 할까? (예를 들어 “abc”)라고 하자.

이 경우의 처리가 구현체마다 미묘하게 헷갈리는 것 같다.

우리는 이 경우,

Key1의 값 => 배열 => 마지막 값 선택 [-1:]

Key2의 값 => 배열 아님 => 값 없음으로 처리하기로 했다.

(즉, 개별 값들이 Array이냐를 보고 Array인 경우에 slice를 적용한다는 것.)

그 이유는,

Key1이 이름의 목록이고

Key2가 체중의 목록인 경우, 각각 array-select를 하는 게 맞기 때문이고

아마도 이 경우가 더 자주 있을 법한 case로 보았기 때문.

( 컬럼지향 데이터 처리가 생각나기도 하는 지점. ㅡㅡ; )

1. **조금 어려운(?) 연습문제.**

\*\* 조건 절을 조금 복잡하게 꼬아보자.