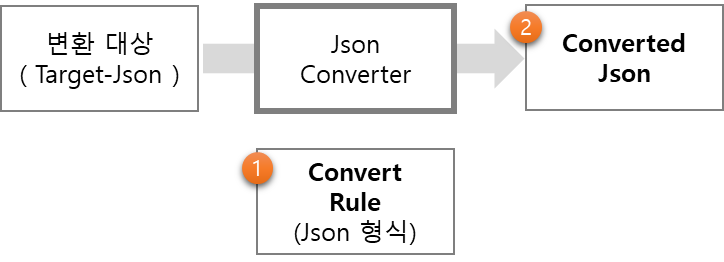
# **Converting (변환 기능)**

**Target-Json을 변환규칙에 따라 변환하는 기능.**

****

# **변환 규칙**

## **[ 변환규칙 훑어보기]**

****

위는 변환 규칙의 사례이다.( Sample내용 전체는 별도 파일참고)

변환 규칙을 살펴보면, 아래와 같은 특징이 있음을 알 수 있다.

1. 변환 규칙은 Josn 형식으로 작성  
   ( 규칙 작성시에 형식오류를 쉽게 발견할 수 있다.)
2. 반복 또는 Recursive 구조를 표현하는 **예약어(필드키)**들이 있다.  
   ( **Target-Json을 변환해서 생성해야 하는 Json의 구조를 정의** )  
   ( 복잡한 구조의 Json을 생성할 수 있도록 하기 위해서 )
3. Target-Json에서 Array, Object, 값을 **추출하는 문법**이 있다.  
   ( **생성해야 하는 Json의 구조의 각 요소에 원본데이터를 어떻게 추출할 것인가를 정의** )  
   ( 데이터 추출방법을 지시하는 문법이 있다. $, 또는 @로 시작하는 문자열 값들 )

## **Convert 룰 해석해 보기.**

****

이하에서는 변환기능이 위의 Convert룰을 어떻게 해석해서 동작하는 지 간단히 따라가 보도록 하자.

### **구조 생성에 해당하는 부분**

**샘플의 내용에서 구조를 정의하는 부분을 순서에 따라 해석하면 다음과 같다.**

1. Target-Json (배열 또는 객체) 에서   
   **[$$]** 조건에 해당하는 것을 **추출**하여 배열을 만들 건데,  
   **[$>]** 에 정의한 방식으로 반복되는 객체를 만들겠다.  
     
   보충) 2번라인의 뜻  
   Target-Json에서 TRT\_INFO, ISR\_INFO, RCPT\_HEADER, FEE\_DETAIL, PRS\_INFO가 포함된 child-node를 반복추출.
2. Target-Json**반복추출한 Child**에서   
   **{::}** 로 지정된 객체를 추출하여  
   **{=>}** 로 지정된 방식으로 대응시켜서 Field들을 만들겠다. (4~27번 라인)
3. Target-Json에서 반복추출한 Child에서  
   **{::}** 로 지정된 객체를 추출하여  
   **{=>}** 로 지정된 방식으로 대응시켜서 객체를 만들고,  
   (반복 생성할 객체의) Receipt 필드의 값으로 하겠다.
4. Target-Json에서 반복추출한 Child에서  
   **[::]** 로 지정된 객체를 추출하면 배열이 될 텐데, 그 배열의 각 요소를  
   **[=>]** 로 지정된 방식으로 변환시켜서 배열을 만들고,  
   (반복 생성할 객체의) ReceiptItem 필드의 값으로 하겠다.
5. 4와 같은 방식으로, receiptDetail, prescriptionList 필드도 만들겠다.

**Developer Note1)**

1. 생성할 Json의 반복구조는 위에서 살펴본 “구조 추출-변환 문법”에 따라 임의의 깊이로 recursive하게 정의할 수 있다. (특별한 제한이 없다는 뜻)
2. 눈치 챘겠지만, Target-Json을 변환해서 객체로 변환하려는 경우에는   
   **{$$}**, **{$>}** 예약어를 쓴다.

**Developer Note2)** 편의문법 기능 **{::}, {=>}**

조금 곰곰히 생각을 해보면, **{::}, {=>}** 예약어는 없어도 될 기능이다.

생성할 필드와 추출할 값을 해당 위치에 나열하면 되므로, {::}, {=>} 키워드는 사용하지 않아도 원하는 규칙을 정의할 수 있다.

편의 문법으로 **{::}, {=>}** 를 정의한 것은 일련의 정보를 같은 노드에서 추출해서 필드를 생성할 경우에 “같은 노드” 를 반복적으로 써야하는 수고를 없애기 위해서 이다. 구조적으로 관리하기에 유리하도록 하기 위한 기능이라는 뜻.

### **값 추출에 해당하는 부분**

Convert룰의 값부분에 정의된 **$** 또는 **@**로 시작하는 문자열은 Target-Json에서 정보를 추출하라는 지시어에 해당한다.

**[$$] {$$} [::] {::}** 의 값은 반드시 추출지시문을 써야 한다. (당연하게도.)

|  |  |
| --- | --- |
| **[$$]** | Target-Json에서 추출지시문에 따라 추출하여, 배열을 만들어라. (배열의 요소를 만드는 규칙은 **[$>]** 이하에 정의.) |
| **{$$}** | Target-Json에서 추출지시문에 따라 추출하여, Object를 만들어라. (배열의 요소를 만드는 규칙은 **{$>}** 이하에 정의.) |
| **[::]** | 현재 Node에서 추출지시문에 따라 추출하여, 배열을 만들어라. (요소를 만드는 규칙은 **[=>]** 이하에 정의.) |
| **{::}** | 현재 Node에서 추출지시문에 따라 추출하여, Object을 만들어라. (요소를 만드는 규칙은 **{=>}** 이하에 정의.) |

값을 할당하는 부분에서는 추출지시문 또는 임의의 값을 쓸 수 있다.

“my key” : “ABC”, <<- 임의의 값을 쓴 경우.

“name” : “**@.Name**” <<- 현재의 Target-Json Node에서 Name 값을 읽어서 (추출지시문)

\* 만약, 추출지시문 대신 $ 또는 @로 시작하는 임의의 문자열을 사용하려면,   
\ (Back-slash)로 escape하여 작성한다.

**[ 기타 ]** 추출 문법은 ***Json-Path 내용을 참고***하도록 하자.

**Developer Note)**

1. **[ 어디에서 추출할 것인가? ]**

추출해서 변환하는 구조가 임의의 깊이에서 수행될 때,   
필요한 정보가 원하는 추출한 현재의 Node에 존재하지 않는 경우가 있을 수 있다.  
그래서, 추출정보를 어디에서 뽑을 것인가를 정하는 예약어가 2개 있다.

|  |  |
| --- | --- |
| 추출 위치 예약어 | |
| $ | Root에서부터 지정한 경로로 추출 |
| @ | 현재의 Node에서 추출 |

**[ 하나 더]**

$ (Root에서 추출)가 가리키는 위치는 사용되는 곳에 따라 다른 곳을 가리킨다.  
**[$$]** 또는 **{$$}** 예약어에서는 Target-Json의 Root를 가리키고,  
그 외의 예약어에서는 **[$$]** 또는 **{$$}** 에 따라 추출된 반복되는 객체의 Root를 가리킨다.  
( Target-Json이 배열이라고 생각해보면,   
 첫번째 배열에 있는 정보는 그 묶음내에서 정보를 찾는 것이 맞을 것이므로. )

# **변환 규칙 사례**

긴 Sample은 URL참고 ( To-be-updated )

**\* Rule을 잘못 만들었을 때의 SyntaxError 정보**

{

"[$>]" : {

"{=>}" : {

"insuranceCd" : "syntax error :: @....INSURE\_TYPE\_CD"

},

"receipt" : {

"{::}" : "syntax error :: $...RCPT\_HEADER",

"{=>}" : {

"patientId" : "syntax error :: @...PAT\_ID"

}

}

}

}

작성한 Rule의 구조를 따라서, 잘못 작성된 부분을 표시함.

**\* 변환 실패시의 메시지 읽어보기**

{

"[$$]" : "$[?(.TRT\_INFO && .ISR\_INFO && .RCPT\_HEADER && .FEE\_DETAIL && .PRS\_INFO)] : 1 -> empty",

"[$>]" : [ {

"{=>}" : [ {

"addItem4" : "@.CUSTOM4\_ : empty"

} ],

"receiptDetail" : {

"[::]" : "$.FEE\_DETAIL[0:] : 2 -> empty",

"[=>]" : [ {

"accumulatedLclsCd" : "@.TRT\_CD\_ : empty"

}, {

"accumulatedLclsCd" : "@.TRT\_CD\_ : empty"

} ]

}

} ]

}

변환실패가 발생한 이유는 @.TRT\_CD\_ 의 값을 Target-Json에서 찾을 수 없었기 때문이다.  
바로 위의 지점을 보면, "$.FEE\_DETAIL 의 추출한 배열이 2개인데, 두 개 모두에서 @.TRT\_CD\_ 값을 못 찾았으므로, “2 -> empty”가 표시되었다.  
더 거슬러 가면, Target-Json에서 1개의 객체를 추출했는데, 변환조건 충족이 되지 않았으므로  
 “1-> empty” 메시지가 표시되었음을 알 수 있다.