用 JsonPath 帮助处理复杂 JSON

作者：ftonz
原文链接：https://ld246.com/article/1546443901289
来源网站：链滴

起因

  前些天PM下了一个很“简单的”需求：我们的页面需要弹出广告控制功能
控制某页面广告弹出与否，现有条件：

1、根据时间段进行限制；

2、根据业务进展情况进行控制；

方案

  听起来很简单，时间控制很好办，但所谓的根据业务进展情况进行控制可就
点不简单了~如果针对单一业务，那很容易，把业务相关参数填入系统后台，每次处理时系统根据业
数据进行判断就好。可如果面向的是不同的业务系统，每次都会进行不同的业务判断，那就不容易了
根据我们的架构特点，我们的系统存在一大特征：**整体而言，绝大多数业务是前后端分离架构** 那么绝
多数的业务数据，都可以调用HTTP接口获取业务JSON。所以，数据的获取问题容易解决，获取到的
据进行解析、判断成了另一个问题。常见的方法有：

1、引入规则引擎

2、引入动态语言执行引擎，如JS、Groovy等动态脚本嵌入执行程序

  鉴于项目规模较小，如果贸然引入规则引擎，很有大炮打蚊子的嫌疑。引入
态语言执行引擎，对于灵活性、完备性都具有较大优势，尤其处理JSON场景上，使用 JS 的话存在先
优势，特别是JAVA自带了 Nashron 引擎，几近0成本获取动态特性，并且前端还能较为便捷地进行
验、测试，不失为一个较优选择。但在调研过程中，我找到了 JsonPath 这一解决方案，发现其能
单、优雅地解决JSON处理这一问题。

   JsonPath其目标和闻名已久的XPath极其相似，是一种特定数据结构的查询
言，使用极其便利，以$为root，.操作符或[]索引的方式获取指定 JsonPath 数据，并且可以使用简单
条件过滤语句，示例如下：

```
{
   "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
```

原文链接：用 JsonPath 帮助处理复杂 JSON

```
        },
        {
            "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
    }
},
"expensive": 10
}
```

JsonPath	Result
$.store.book[*].author	所有book属性下的author
$..author	所有的 authors
$.store.*	store属性下的所有属性，包括 books 、bicycles
$.store..price	sotre属性下的所有price属性
$..book[2]	第三本书
$..book[-2]	倒数第二本书
$..book[0,1]	前两本书
$..book[:2]	从index 0开始（包含）到index 2 (包含)的书
$..book[1:2]	从 index 1 (包含) 到 index 2 (不包)的书
$..book[-2:]	最后两本书
$..book[2:]	从index2到最后一本书
$..book[?(@.isbn)]	所有含 ISBN 号的书
$.store.book[?(@.price < 10)]	所有单价低于 10的书
..book[?(@.price <= ['expensive'])]	所有单价于等于 "expensive" 属性的书
$..book[?(@.author =~ /.*REES/i)]	所有匹配正表达式 (ignore case)的书
$..*	获取所有属性
$..book.length()	获取book书性的长度

  作为一个表达式而言，JsonPath的表达能力已经很强大了对于一个简单的JSON结果判断，也应该轻而易举了。

  那么抛开这个场景，JsonPath还有哪些用途？如果各位有

原文链接：用 JsonPath 帮助处理复杂 JSON

大量对接过第三方API的经历的话，一定也饱受过JSON解析的痛苦，尤其是JAVA、C#之类的静态语，遇到深层级的JSON处理简直苦不堪言。

文艺CODER：

```
//HttpCode: 2XX(成功) 4XX（客户端错误） 5XX(服务端错误)
//2XX：直接Decode为业务类型
//4XX、5XX：约定好标准错误响应格式，进行标准数据返回
```

普通CODER:

```
public class Result<T>{
    boolean success;
    String msg;
    T data;
}
```

2X CODER:

```
public class Result{
    boolean success;
    String msg;
}

public class XXResult extends Result{
    Object 属性名就看我心情吧;
}
```

  通常，遇到前两种Coder风格Decode都会比较容易处理，但是遇到第三种格估计就得男默女泪了.....此时，你必须根据不同的业务属性名进行不同的业务处理。要么大量建类要么用JSONObject一类的方案进行动态处理。总之是免不了一堆繁杂的创建和处理...说好的高内聚耦合呢？都去见了马克思吧~

  此时，JSONPath大显身手的时候就来了，可以为每一个业结果类打上一个记录JSONPath的自定义注解，Decode时获取JSONPath,并反序列化其指定部分就，代码瞬间清爽了不知多少。

后记

  如果，你和我一样不幸，遇到了这样的返回值：

```
public class XXResult extends Result{
    Object list; //此时的list属性充分使用了OOP语言的多态特征：如果有多个元素，则为List<X>类，如果只有一个元素，则为X类型
}
```

  不要哭泣，不要悲伤，如果你有幸用到了com.jayway.jsonath:json-path这个库，那么恭喜你，它存在一个可配置项：ALWAYS_RETURN_LIST你在定义config，传入这一参数，所有的JSONPath返回值都会成为一个List这“多态”的list就能完美解析了。

  2019年开年为大家介绍了这一神器，希望各位在接下来寒冷的日子里对接口时，能多一点从容，少一点惊慌。码农们，天寒地冻，正是练功的好时节，与各位同学共勉。