39 | XML、JSON、YAML比较

2019-12-09 四火

全栈工程师修炼指南 进入课程 >



讲述: 四火

时长 14:39 大小 10.07M



你好,我是四火。

XML 和 JSON,是程序员几乎每天都会打交道的数据、特别是配置数据的承载格式。我想你心里应该有一个大致的印象,它们二者各有什么优劣,但是也许没有系统地比较过。那今天我们就把它们放到一起,丁是丁卯是卯地分析分析,对比一下它们各自的特点。另外,这些年来,对于配置,特别是复杂 DSL 的配置,YAML 也逐渐流行开来,因此我们也把它拿过来剖析一番。

XML 和 JSON 的比较

XML 全称叫做 Extensible Markup Language,就像 HTML、CSS 一样,是一种标记语言(标记语言不属于传统意义上的编程语言),且是一种具备结构化特征的数据交换语言;类似地,JSON,也就是 JavaScript Object Notation,被称作 JavaScript 对象表示法,非结构化,更轻量,但归根结底也是一种数据交换语言。因此,二者具备相当程度的相似性,在实际应用中,往往也可以比较和替代。

1. 简洁还是严谨

在 ②[第 04 讲] 的时候,我介绍了 REST 和 SOAP 这样一个简洁、一个严谨的俩兄弟。而在本讲中,JSON 和 XML 也在一定程度上同样满足这样的比较关系,JSON 往往是更为简洁、快速的那一个,而 XML 则更为严谨、周全。

我们来看一个简单的例子, id 为 1 的城市北京:

如果用 JSON 表示:

```
1 {
2  "city": {
3     "name": "Beijing",
4     "id": 1
5     }
6 }
```

你可能会说,除了 XML tag 的名字,在 JSON 中只需要写一遍以外,看起来复杂复杂、严谨的程度似乎也差不太多啊。

别急,往下看。XML 的结构,强制要求每一个内容数据,都必须具备能够说明它的结构,而 JSON 则没有这样的要求。比方说,如果我们把城市组成数组,用 XML 来表示,请你把这个文件存成 cities.xml,因为我们会多次用到这个文件:

如果使用 JSON 的话,由于对于数组可以使用中括号直接支持,而不需要显式写出上例中的 city 这个 tag 的名称,请你同样建立 cities.json:

```
1 {
2  "cities": [
3      {"name": "Beijing", "id": 1},
4      {"name": "Shanghai", "id": 2}
5      ]
6 }
```

从这就可以看出,在这种情况下,JSON 似乎确实要更为简洁一些。上例中 JSON 能够使用中括号直接表示数组,能够直接支持数值、字符串和布尔型的表示。

等等,这样说的话,JSON 因为能够直接支持数值的表示,这个 id 没有双引号修饰,就是数值类型,可是从 XML 中并不能看出这一点啊。因此,从这个角度说,应该是 JSON 更严谨啊! 那为什么说 XML 更严谨,严谨在哪呢?

有些程序员朋友可能会马上想到,XML 是可以定义 tag 属性的,预定义一个 type 不就好了?

看起来也能达到"严谨"的目的,可这很可能就是一个不好的实践了,因为 XML 对于这些常见的数据类型,内置了直接的支持。我们可以通过定义 XML Schema Definition (XSD)来对 XML 的结构做出明确的要求,也就是说,我们不必自己去造轮子,来定义并实现这个 type 属性。针对上面的 cities.xml,我们可以定义这样的 XSD:

```
■ 复制代码
 1 <?xml version="1.0" encoding="UTF-8" ?>
 2 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
     <xs:element name="cities">
4
       <xs:complexType>
 5
         <xs:sequence>
           <xs:element name="city" max0ccurs="unbounded" min0ccurs="0">
 6
7
             <xs:complexType>
8
               <xs:sequence>
9
                  <xs:element type="xs:string" name="name"/>
                  <xs:element type="xs:byte" name="id"/>
10
11
               </xs:sequence>
12
             </xs:complexType>
13
           </xs:element>
14
         </xs:sequence>
15
       </xs:complexType>
16
     </xs:element>
17 </xs:schema>
```

这样一来,我们就对 cities 和 city 这两个 tag 做了严格的内容限定,包括包含的子节点有哪些,顺序如何,取值类型是什么等等。在实际的 XML 定义中,我们可以引用这个 XSD,这样 XML 的处理程序就会加载这个 XSD 并根据 schema 的规则对 XML 进行校验,从而发现 XML 不合要求的问题。

进一步地,你可以自己动动手,看一下<mark>⊘这个工具</mark>,它可以帮助你通过 XML 快速生成样例 XSD; 而<mark>⊘这个工具</mark>则可以帮你快速验证 XML 是不是满足某 XSD 的要求,它们都很有用。

补充一下,你可能也听说过,或使用过类似的叫做 DTD,也就是 Document Type Definition 的方式,能想到这个很好,但是 XSD 相对来说有着更大的优势,并成为了 W3C 的标准。因此我在这里不提 DTD,但是我在扩展阅读中放了关于 XSD 和 DTD 的比较材料,供感兴趣的朋友拓展。

我想,从 XSD 你应该可以体会到 XML 的严谨性了。那喜爱使用 JSON 的程序员,就不能创造一个类似的东西,来填补这个坑——即定义和保证 JSON 的严谨性吗?

有,它就是 ⊘JSON Schema,也许你已经在项目中使用了,但是还没有统一成标准,也没有被足够广泛地接纳,因此我就不展开来说了。你可以自己实践一下,把上面提到的 JSON 填写到这个 ⊘JSON Schema 推断工具上面,去看看生成的 JSON Schema 样例。

2. JavaScript 一家亲

对于全栈工程师来说,和 XML 比较起来,JSON 对于前端开发来说,可以说有着不可比拟的亲和力。本来,JSON 就是 JavaScript 对象表示法,就是从 JavaScript 这个源头发展而来的,当然,JSON 如今是不依赖于任何编程语言的。这个"一家亲",首先表现在,JSON 和 JavaScript 对象之间的互相转化,可以说是轻而易举的。

我们来动动手实践一下,打开 Chrome 的开发者工具,切换到 Console 页,打开前面建立的 cities.json,拷贝其中的内容到一对反引号(backtick,就是键盘上 esc 下面的那个按键)中,并赋给变量 text:

```
□ 复制代码

□ var text = `JSON 字符串`;
```

我们很轻松地就可以把它转化为 JavaScript 对象(反序列化),不需要任何第三方的 parser:

```
□ 复制代码

1 var obj = JSON.parse(text);
```

在早些时候的 ES 版本中,这个方法不支持,那么还可以使用 eval 大法,效果是一样的:

```
1 var obj = eval('(' + text + ')');
```

不过,在现代浏览器中,如果 text 不是绝对安全的,就不要使用这样的方法,因为 eval 可以执行任何恶意代码。

当然,我们也可以把 JavaScript 对象转换回 (序列化) JSON 文本:

```
□ 复制代码
1 var serializedText = JSON.stringify(obj);
```

完成以后, 先不要关闭控制台, 下面还会用到。

3. 路径表达式

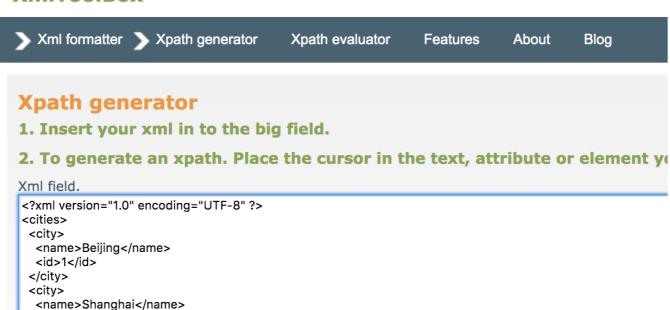
对于一段巨大的 XML 或 JSON 文本,我们经常需要找出其中特定的一个或多个节点 (tag)、内容 (content) 或者属性,二者都有相似的方法。

对于 XML 来说,它就是 XPath,也是 ⊘W3C 的标准。现在我们来动手操作一下吧:

我们可以利用 Xpath Generator 来直观地生成相应的 XPath。让我们打开 <u>②这个工具</u>,找到我们刚才保存下来的 cities.xml 文件,拷贝里面的内容,粘贴到页面上。

XmlToolBox

<id>2</id>
</city>
</cities>



接着,点击第一个 id 标签,你将看到这样的 XPath:

```
□ 复制代码

□ /cities/city[1]/id
```

这就唯一确定了 XML 中, cities 这个节点下, city 节点中的第一个, 它下面的 id 节点。

我们再来试一试,点击 Shanghai 这个 content,你将看到:

```
□ 复制代码
1 /cities/city[2]/name/text()
```

对于 JSON 来说,也有 JSONPath 这样的东西,但是,我们却很少提及它,因为正如上文我提到的,它和 JavaScript 极强的亲和力。我们在前端代码中已经不自觉地通过对象的点操作符,或是数组的下标操作符使用了,于是,JSONPath 在多数场景中就显得不那么重要了。

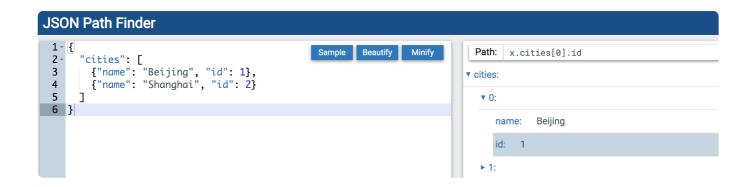
举例来说,上面我给出的两个 XPath 例子,我们在将 cities.json 反序列化成 JavaScript 对象以后,我们可以直接访问(你可以在前面实践的控制台上,继续键入):

```
国 复制代码
1 obj.cities[0].id
```

以及:

```
□ 复制代码
□ obj['cities'][1].name
```

但是,还有很多场景,特别是对于 JSON 支持不像 JavaScript 那么足够的场景, JSONPath 就有其用武之地了。和前面介绍的 XPath 查找的例子一样,你可以打开 Ø JSON Path Finder 页面,把之前 cities.json 的文本粘贴到左侧的输入框中,在右侧选择 对应的节点或值,上方就会显示出 JSONPath 了:



所以,Beijing 的 id 和 Shanghai 的 name 分别显示为:

```
且 复制代码
1 x.cities[0].id
2 x.cities[1].name
```

这和 JavaScript 对象访问的表达式是一致的。

4. 特殊字符

任何格式都要使用特定的字符来表示结构和关系,那么 XML 和 JSON 也不例外,这些特定字符,如果被用来表示实际内容,就会出现"冲突",于是我们就需要转义。

对于 XML 来说,且看下面的表格,第三列是"预定义实体",也就是字符转义后相应的形式:

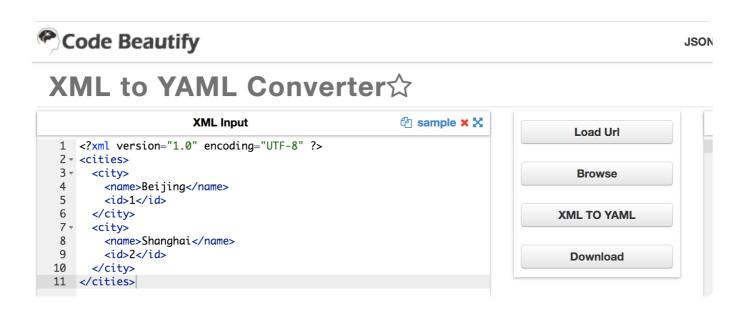
字符	名字	实体格式
П	quot	"
1	apos	'
&	amp	&
>	gt	>
<	It	<

值得一提的是,在 XML 中,我们还能够以 CDATA 来表示内容,这种方式可以尽可能地避免对于转义的使用,并可以直接使用换行等控制字符,增加 XML 的可读性。比方说下面这个例子,实际的 function 使用 CDATA 的方式给嵌在了 function 这个 tag 内:

```
1 <function><![CDATA[
2 function compare(a, b) {
3     ...
4 }
5 ]]></function>
```

对于 JSON 来说,没有这样的预定义实体,但是我们也经常需要转义,比如说,如果双引号出现在了一对双引号内的字符串中,这时候我们可以用常规的反斜杠 ⊘ 转义序列来表示,比如引号"转义为 \"等。

审视一番 YAML



然后点击 XML TO YAML 按钮,你就能看到前面的例子,如果使用 YAML 来表示,它会是这么个样子:

```
1 cities:
2 city:
3 -
4 name: Beijing
5 id: 1
6 -
7 name: Shanghai
8 id: 2
```

你看,这种方式对于层次结构的表达,可以说比 XML 或 JSON 更为清晰。对于 XML 和 JSON 的表达,对于每一层节点,你都要去找结尾标记,并和节点开头标记对应起来看;但 是 YAML 则完全不用,它利用了人阅读的时候,一行一行往下顺序进行的特性,利用直观 的缩进块来表达一个特定深度的节点。对于某些强调层次结构的特定信息表达的场景,比如 说电子邮件消息,或者是商品信息、候选人的简历等等,使用 YAML 比其它的数据交换格 式都要直接和清晰。

值得注意的是,对于缩进,YAML要求不可以使用TAB,只能使用空格,这和Python是不同的;而对于每层缩进多少个空格,这不重要,只要保证不同层次的缩进空格数量不同即可,这一点和Python是相同的。

YAML 由于极强的可读性,它在数据类型的明确性上做了一定程度的牺牲。从上面的例子你可以发现,本来我们希望 name 是字符串,id 是数值,可是 YAML 它根本不关心这一点,如你所见,字符串也没有双引号修饰,它只是诚实地把具体的文本数值罗列展示出来罢了。这一点,在我们权衡不同的数据交换格式的时候(比如设计哪一种格式作为我们的模块配置项文件),需要注意。

总结思考

今天我们一边动手比较、一边学习了 XML 和 JSON 的前前后后,包括它们的风格、schema 和路径表达式等等,并在之后了解了可读性至上的另一种数据交换语言 YAML。希望这些内容能够帮助你对于这些数据交换语言有更为全面的认识,并能够在工作中选择合适的技术来解决实际问题。

今天的提问环节,我想换个形式。这一讲我们已经比较了许多 XML 和 JSON 的特性了,其中一些涉及到了它们之间的优劣。那么,你能不能归纳出 XML 和 JSON 各自的一些优劣来呢?比如有这样几个方面供你参考,当然,你完全可以谈其它方面:

```
报文大小;
数据类型的表达能力;
Schema 的支持;
可读性;
数据校验的支持性;
序列化和反序列化(编程访问)的难易程度;
程序处理的性能;
Web 中的普适性;
可扩展性(自定义 DSL 的能力);
```

扩展阅读

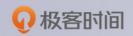
关于 DTD, 你可以在 ② 这里找到许多例子; 而 XSD, 它的例子则在 ② 这里。而且, 在 ② XML Schema 与 XML DTD 的技术比较与分析这篇文章里, 你可以得知为什么 W3C 最后选择了 XSD, 而不是 DTD。另外, 对于 XSD 的批评, 你可以参看 ② 这里。

对于 XPath,如果想全面地了解它的语法,请参阅 ② XPath 词条;如果你想检验所学,校验自己书写的 XPath 的正确性,那么你可以使用 ② 这个工具,这个工具和文中介绍的 Xpath Generator 配合起来使用,可以非常有效地帮助你理解 XPath。

相应的,对于 JSONPath,你可以阅读 ② 这篇文章来进一步了解它的语法,你也可以使用 ② 这个工具来校验其正确性。

对于 YAML, 你可以阅读 ② 这个词条来获得较为全面的了解; 另外, 你可能听说过 YAML 和 JSON 之间的超集与子集这样的关系, 那我推荐你阅读 YAML 官方文档的 ② 这一小段关于它与 JSON 的区别和联系。

文中介绍了数据交换语言这个大家族中的三个,其实还有其它成员,你可以阅读一**⊘下** 这个列表。



全栈工程师修炼指南

从全栈入门到技能实战

熊燚

Oracle 首席软件工程师



新版升级:点击「探请朋友读」,20位好友免费读,邀请订阅更有现金奖励。

⑥ 版权归极客邦科技所有,未经许可不得传播售卖。 页面已增加防盗追踪,如有侵权极客邦将依法追究其法律责任。

上一篇 38 | 分页的那些事儿

精选留言(3)





靠人品去赢

2019-12-09

JSON前段后台通讯用比较不错,看了一下文中关于JSON和YAML的比较,总体感觉就是YAML要求更高,不管是目的,YAML在保持可读性的同时还要支持连续的基本数据结构,像什么key的唯一性YAML是必须的,JSON就可以乱搞。

但是工作中,感觉YAML对手更应该是XML,还有一个"free style"选手,properties配置文件。

展开~







許敲敲

2019-12-09

JSON的http报文大小更小,并且文件格式对于JS,Python来讲更容易操作,所以在web中更流行吧。







从操作系统方面而言其实代表了两类操作系统数据库: windows和linux;sql server2005开始就有不少数据用了xml格式, JSON的典型代表是mongodb。

xml的强结构化,可读性非常好;曾经主攻sql server时大量的开发直接生成xml,其文档的可读性和强结构化让代码做优化和修改时非常方便,当今主流的mybatis框架用的同样是xml格式。老师在说javascripe时其实漏了一个东西BSON,JSON同时代的产物,只是… 展开 >

作者回复: 凸

BSON 是众多数据交换格式中的一种,现在使用并不多,而且是二进制的。

