技术进展调查报告

——Node网页框架Express vs Koa vs Hapi

**1背景**

Node是基于Google的V8引擎打造了基于事件循环实现的异步I/O框架，而V8引擎提供的远超其它脚本语言的性能及异步I/O的特点正是Node成功的极其重要的因素。它是目前最火热的后端技术，在开源社区Github上，Node高居第二。我们小组创建的是teambuilder网站，本着对新技术的热爱和学习，我们选择了Node作为网页后端。

选择了Node，接下来就是要寻找一份合适的Web开发框架，有如php之Laravel、CodeIgniter，Python之Django、Tornado以及Ruby之Rails。借助各种社区推荐，我们发现了Express，Koa，Hapi这三款比较热门的基于Node的网页框架。

**2 框架的背景**

**2.1 Express**

Express无疑是当前Node中最流行的Web应用框架，它几乎成为了大多数Node Web应用程序的基本的依赖，甚至于在介绍Node的书籍上，Express的影子也是处处可见。Express 是一个小巧且灵活的 Node.js Web应用框架，它有一套健壮的特性，可用于开发单页、多页和混合Web应用。根据其github上的README.md，这个框架被描述成：疯一般快速(而简洁)的服务端Javascript Web开发框架，基于Node.js和V8 JavaScript引擎。

**2.2 Koa**

相对Express，Koa这个框架有着更新的历史。于2013年8月17日，Express的作者TJ Holowaychuk提交了Koa的第一次commit，他描述Koa为“表现力强劲的Node.js中间件，通过co使用generators使得编写web应用程序和REST API更加丝般顺滑”。其中generators是ECMAScript 6中的新技术，也即JavaScript的下一个版本，而co[http://github.com/tj/co]是基于Node的一个ECMAScript 6的实现。

**2.3 Hapi**

Hapi是三个中最与众不同的一个，它基于一个观念：配置优于编码，业务逻辑必须和传输层进行分离。起初Hapi是基于Express构建的，后来才发展成自己的框架。

**3 社区热门度**

以下是这几个流行框架的社区数据：

**Metric** Express.js Koa.js Hapi.js

Github Stars 16,158 4,846 3,283

Contributors 163 49 95

Packages that depend on 3,828 99 102

StackOverflow Questions 11,419 72 82

由上可见，正如刚刚所说，Express的热门度远远超过另外两个，而Koa与Hapi更像是新兴的。StackOverflow是国外一个十分著名的提问社区，Express有问题量大大超过另两个，正是代表着它的使用者的数目。Github Stars代表着其项目受关注的程度，相对于StackOverflow Questions，三者的Github Stars相差没那么大，除去Express出现的年代比另两个早的原因，其实他们受关注的程度已经接近，这也从某种程度说明了Koa及Hapi的发展潜力。

**4 HelloWorld**

HelloWorld是程序开发中最经典的入门例子，现在分别用三个框架来实现一个最基本的网页服务，对比它们三个的区别

**4.1 Express**

var express = require('express');

var app = express();

app.get('/', function(req, res) {

res.send(''welcome to Teambuilder);

});

var server = app.listen(3000, function() {

console.log('Express is listening to http://localhost:3000');

});

Express使用get()函数来捕获’GET /’请求然后调用一个回调函数，这个回调函数会被传入req和res两个对象。这个例子当中我们只利用了res的res.send()来返回整个页面的字符串。Express有很多内置的方法可以用来进行路由控制。get, post, put, head, delete等等这些方法都是Express支持的最常用的方法。

**4.2 Koa**

var koa = require('koa');

var app = koa();

app.use(function \*() {

this.body = 'Hello world';

});

var server = app.listen(3000, function() {

console.log('Koa is listening to http://localhost:3000');

});技术进展调查报告

——JSON VS XML

我们小组的项目标题为团队协作工作，呈现形式为网页WEB，而在WEB技术中，很重要的一点就是数据交换的格式问题，为了避免数据异构的问题，科学各界在过去的研究中提出了很多种数据交换的格式，下面将以JSON和XML这两种比较主流的数据交换语言进行对比和分析。

一、背景

随着WEB技术的兴起，通过网络来共享信息资源成为计算机技术发展的迫切需求。但是由于不一致的开发语言与应用系统环境、不同意的数据交换格式和传递机制导致了各系统间的数据异构问题，从而很难真正地实现数据共享。因此，要实现系统间的信息共享，本质上是要解决各系统间的数据异构问题，而解决数据异构问题的技术方案之一就是数据交换，任何数据交换都建立在某种特定的数据格式之上。因此，研究精巧的数据格式和高效的数据交换技术并将其应用于数据交换事件以实现信息共享具有现实的意义。

20世纪七十年代早期，国外就开始关注异构数据源的整合交换问题，数据交换也是在那时候开始进入专家与学者的研究领域。在此后的几十年的时间内，先后出现了多种数据交换格式，其中比较有名的就是由道格拉斯·克罗克福特提出的JSON 和万维网联盟发布的XML。

二、XML

XML（eXtensible Markup Language，可扩展置标语言），是一种和HTML类似的置标语言。置标语言是一种计算机文字编码，将于文件相关的信息（如文本的结构和表示信息等）与原本的文本结合在一起，使用标记（markup）进行标识。因为XML是W3C制定的，XML的标准化工作由W3C的XML工作组负责，该小组成员由来自各个地方和行业的专家组成，因此XML不是一个依附于特定浏览器的语言。XML的设计思想是将文件用标记内容来代替表述，并且人们可以通过这些标记内容还原成原始文件。XML 不是显示数据的工具，而是交换数据的工具，它将数据以某种格式的形式在各种计算机应用之间传输。XML的巨大优势在于用户可以自由地为特定的应用定义自身标识的标签，例如为内部数据交换定义一套自我识别的标签，为外部数据交换定义一套相关方都能识别的标签。总之，XML的出现为能够实现内部和外部数据交换提供了坚实的基础。

虽然XML被设计用来存储、传输数据，而不是用来显示数据，但是它常被用于数据交换，而不是数据存储。在各平台之间进行数据交换时，其核心问题之一就是需要一个各方面都能有效识别的统一的数据格式，由于XML的跨平台特性、良好的数据处理能力以及严谨的语法格式等特征，许多企业和组织在进行异构数据源之间数据交换时会选择XML，此时，XML统一了数据格式，使数据交换的实现成为可能。

XML 文档实际形成了一种树结构（结构可以参见下面实例的图），而且该文档必须包含根元素。该元素是所有其他元素的父元素，这棵树从根部开始，并扩展到树的最底端。所有元素均可拥有子元素：

<root>

<child>

<subchild>.....</subchild>

</child>

</root>

父、子以及同胞等术语用于描述元素之间的关系：父元素拥有子元素；相同层级上的子元素成为同胞（兄弟或姐妹）；所有元素均可拥有文本内容和属性（类似 HTML 中）。

下面会以Student这个实例的实现来看XML的具体使用方法。下面的代码描述了一个学号为2011150369，名字为Aron，并且是计算机与软件学院的软件工程专业的一名深大学生。从树形结构图中，可以看出，根元素为SZU，而且它必须有且只有一个根元素。

<SZU>

<student StuNO=”2011150369”>

<Name>Aron</Name>

<Gender>Male</Gender>

<College>College of computer science and software engineering</College>

<Major>Software engineering</Major>

</student>

</SZU>

XML作为一种比较严谨的数据交换语言，主要具有以下几点优点：

①跨平台特性：XML 文档本质上是一个文本文件。对于文本文件而言，它在所有的操作系统上都是同义的，因而XML 文档不需要就行特定的修改就可移植到任何操作系统上。

②扩展性强：因为XML 实现了数据和显示的分离，用户在扩展数据的时不用像以前那样必须修改HTML 而导致数据显示的变化。XML 允许用户创建特定应用的标签然后使用这些标签，而不去使用HTML 的有限标签。

③验证机制良好：XML 的标签是用户自己定义的，XML 的语法格式会自动检测标签的定义和使用是否符合语法。

尽管如此，XML还是不可避免会产生许多缺点：

①更新困难：XML 文档采用树状结构来存储数据，搜索树状结构的效率相当高，但是更新和新增显得特别困难，特别是在数据量很大的时候，很容易由于用户错误地插入或修改导致树状的结构被破坏。

②通信困难：由于XML 是置标语言，任何个人用户或企业用户都可以按照自己的应用需求来设计自己的标签，而这些标签没有统一的标准，利用这些没有统一标准的XML 文件进行数据通信的难度很大。

③效率低下：XML 是采用文本形式存储和传输数据的，当数据量很大时，XML文档由于太多标签导致其比数据本身要大得多，特别是相对于基于二进制形式的数据文件，其数据量大的不可接受，当数据需要通过网络进行传输时，XML 文档占用了很大的带宽，其效率极其低下。

三、JSON

JSON（JavaScript Object Notation，JavaScript对象标记）是一种基于文本的、易于人类阅读和编写、计算机生成和解析的轻量级数据交换格式。JSON易于人阅读和编写，同时也易于机器解析和生成。它基于JavaScript（Standard ECMA-262 3rd Edition – December 1999）的一个子集。JSON 采用完全独立于语言的文本格式，但是也使用了类似C语言家族的习惯（包括C，C++，C#，Java，JavaScript,Perl，Python等）。这些特性使JSON 成为理想的数据交换语言。

JSON简单说就是JavaScript中的对象和数组，所以这两种结构就是对象和数组两种结构，通过这两种结构可以表示各种复杂的结构：

1、对象：对象在JavaScript中表示为“{}”括起来的内容，数据结构为 {key：value,key：value,...}的键值对的结构，在面向对象的语言中，key为对象的属性，value为对应的属性值，所以很容易理解，取值方法为对象.key 获取属性值，这个属性值的类型可以是数字、字符串、数组、对象几种。

2、数组：数组在JavaScript中是中括号“[]”括起来的内容，数据结构为 ["java","JavaScript","VB",...]，取值方式和所有语言中一样，使用索引获取，字段值的类型可以是数字、字符串、数组、对象几种。

相比于XML，JSON的数据格式是轻量级的、人类可读的并且是计算机可解析与生成的，以便进行数据交换。具体的使用方法可以见下面的实例，从这个实例可以看出只要明白了JSON的结构，代码写起来就相当的简单。

|  |
| --- |
| {  "StuNO":"2011150369",  "Name":"Aron"  "Gender":"Male",  "College":"College of computer science and software engineering",  "Major":"Software engineering"  } |

JSON也具有相当多的优点，主要有以下几点：

①简单易读：从JSON 数据格式的语法格式可以看出，JSON 格式的数据时简单的，并且可读性很强，用户可以通过JSON 格式的数据理解数据的内容。

②支持广泛：从JSON 的官方网站发布的信息可以出来，Java、C/C++、C#、PHP、Python 等常用的程序设计语言都已经提供了JSON 数据格式的解析器和生成器。

③高压缩性：相对于XML 格式的数据，JSON 格式的数据明显是压缩了的。XML 数据格式的语法中规定开始和结尾标记是必需的；而JSON 数据格式语法中，不需要纷繁复杂的标记、只需大括号或中括号就能代替。在含有数以千计数据单元的数据交换中，解析XML 标签要耗费大量的时间，这会使数据交换的时间有所延长。

相对于XML，JSON 的不足如下：

①应用不广泛：因为JSON 后于XML 出现，而且推广力度没有XML 那样大，JSON 数据格式远没有XML 数据格式应用得广泛。

②通用性不强：JSON 没有XML 那样普遍应用于各行各业的数据交换中。因为应用不广泛的缘故导致了JSON 在数据交换的某些方面通用性没有XML 那么强。

四、XML和JSON对比

用相同的数据创建了一个XML和JSON来表示它。下面的XML和JSON文件是使用 Programming Pearls中的信息创建的。

**XML版本**

|  |
| --- |
| <student>       <StuNO>2011150369</StuNO>       <Name>Aron</Name>       <Gender>Male</Gender>       <Contact>12345678910</Contact>       <ID>445281111111111111</ID>       <Hometown>Shenzhen Guangdong Province</Hometown>       <College>College of computer science and software engineering</College>       <Major>Software engineering</Major>       <Dorm>South area dormitory of Shenzhen University</Dorm>       <Class>Software Engineering 3</Class>  </student> |

**JSON版本**

|  |  |
| --- | --- |
|  | {   "student": {       "StuNO": "2011150369",       "Name": "Aron",       "Gender": "Male",       "Contact": "12345678910",       "ID": "445281111111111111",       "Hometown": "Shenzhen Guangdong Province",       "College": "College of computer science and software engineering",       "Major": "Software engineering",       "Dorm": "South area dormitory of Shenzhen University",       "Class": "Software Engineering"       }   } |

使用上面的XML和JSON文件分别运行解析测试10,000,000次。结果并不令人惊讶，解析和转换JSON成一个Java对象的速度比XML快30％左右。 JSON平均运行时间为3.7052423564201369E-5，而XML平均运行时间：5.536980356900239E-5.因此相对于XML，JSON的解析速度提高了30％,占用空间少30％。这些结果似乎和多数开发社区对两种格式的看法一样。换用JSON处理数据在性能上可以有不小的提升，而且还会减少空间的占用。

五、总结

两者的共同优点是都是文本表示的数据格式，可以跨平台、跨系统交换数据。

就可读性和可扩展性而言，JSON和XML不相上下。就编码难度而言，JSON略比XML简单，要写好XML最好还是借助XML编辑器，标签的嵌套和配对手动输入来说容易出错。因此，从代码的可维护性角度而言，JSON优于XML。就内容解析而言，JSON在客户端只要使用eval()便可完成解析，在服务器端有包括ActionScript, C, C#, ColdFusion, Java, JavaScript, Perl, PHP, Python, Ruby等语言的解析支持，非常简单；而XML则相对复杂，不管在服务器端还是在客户端都需要花费大量的代码来实现解析。就数据表示和传输性能而言，JSON明显比XML简洁，格式简单，占用带宽少。就安全性而言，由于大部分JavaScript库都是使用eval()来解析数据，存在执行恶意JSON数据的安全漏洞；当然可以使用专门的JSON解析器来避免这个问题。相比，XML更安全一些。

通过上面的比较和实验的结果可以看出，JSON在性能上拥有较大的优势，因此如果一个网页是以性能优先的情况下，可以优先采用JSON 作为数据交换语言。而如果性能对于项目而言并不是那么重要，但是相反可读性和安全性却是比较重要的特性，则可以优先采用XML格式来进行数据交换。