Scientific Journal of Impact Factor (SJIF): 4.72

e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 4, Issue 4, April -2017

Refining Data Exchange Methodologies using JavaScript Object Notation

Yogesh Kumar Das,Pulkit Jain

B. Tech Computer Science Engineering, SRM University
B. Tech Computer Science Engineering, SRM University

Abstract—*When designing an application that will communicate with a remote computer, a data format and exchange protocol must be selected. There are a variety of open, standardized options, and the ideal choice depends on the applications requirements and pre-existing functionality. For example, SOAP-based web services format the data in an XML payload wrapped within a SOAP envelope.While XML works well for many application scenarios, it has some drawbacks that make it less than ideal for others. One such space where XML is often less than ideal is with Ajax-style web applications. While most browsers can construct, send, and parse XML, JavaScript Object Notation (or JSON) provides a standardized data exchange format that is better-suited for Ajax-style web applications.When the client is created with JavaScript in a browser it's much easier to exchange data with something called JSON. JSON is an acronym for JavaScript Object Notation. It is a sting-like notation that makes it easy to declare complex structures like arrays and object.JSON is a data exchange format that was created from a subset of the literal object notation in JavaScript.The charm of JSON is in its simplicity. A message formatted according to the JSON standard is composed of a single top-level object or array. The array elements and object values can be objects, arrays, strings, numbers, Boolean values (true and false), or null.*

Keywords*-JSON;AJAX;Data feed; Remote procedure call; serialization; deserialization; Asynchronous calling*

I. INTRODUCTION

JSON or JavaScript Object Notation is a very popular data interchange format. It was developed by Douglas Crockford. JSON is text-based, lightweight, and a human readable format for data exchange between clients and servers. JSON can be used in web applications for data transfer. Prior to JSON, XML wasconsidered to be the chosen data interchange format. XML DOM implementation made life tedious, as querying for data had to be performed at two levels:

- First on the server side where the data was being queried from a database, and the
- Second time was on the client side using XPath. JSON does not need any specific implementations.XML is much more difficult to parse than JSON.JSON is parsed into a ready to use javascript object.

Languages such as PHP, Python, C#, C++, and Java provide a very good support for the JSON data interchange format. All the popular programming languages that support service-oriented architecture have understood the importance of JSON and its implementation for data transfer, thus, they have provided great support for JSON.A widely-used application of JSON is in mobile apps. Communication between the mobile device and the web server comes about by means of JSON calls. Data communication via JSON is such a popular method for mobile applications because the JSON data is so lightweight and can therefore be sent quickly and efficiently through a mobile connection.Another commonly used application of JSON is in websites. JSON allows just parts of the web page, and not the entire page, to be refreshed and updated. This means the web browser can refresh a fragment of the web page with very little effort, giving the visitor a smooth online user experience

II. DATA FEED

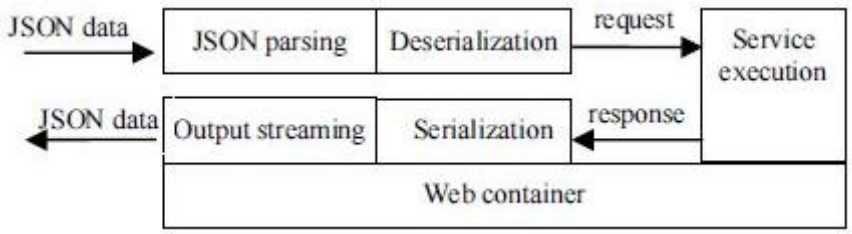
A JSON feed is a data format in JavaScript and is used as a communication method between web browser and web server. The JSON data can be fed in different formats, such as simple type (both string and numeric) as well as complex type (like arrays and data objects). A data feedserves the purpose of JSON application; a feed is a crude way of supplying data so that others can reuseit to display the data on their websites or to ingest the data and run their algorithmson it. Such data feeds are big in size and cannot directly be embedded into the script. It can be retrievedeither as a response from an asynchronous request or from a JSON feed.

III. ACCESSING OBJECT USING JSON

To traverse through any JSON feed, it is important to make a note of how the data is arranged. There are two ways to access data in a JSON object: dot syntax or array notation.

- Dot-syntax: It is a way of indicating an object's property—specifically, by adding a period between the name of the object and the property you wish to access. You've seen this in use with properties of different JavaScript objects like strings and arrays. Dot notation is designed for easy, general use and common use cases. Queries of JSON data that use dot-notation syntax return JSON values whenever possible. If a *single* JSON value is targeted, then that value is the string content, whether it is a JSON scalar, object, or array. If *multiple* JSON values are targeted, then the string content is a JSON array whose elements are those values. The dot-notation syntax is a table alias (mandatory) followed by a dot, that is, a period (.), the name of a JSON column, and one or more pairs of the formjson_field orjson_fieldfollowed by array_step, where json_field is a JSON field name and array_step is an array step expression as described in Basic SQL/JSON Path Expression Syntax.
- Array Notification: To tackle an array of objects, we have to handle them in an iterative method. To resolve this we use an iterative solution in which we target one object at a time; once the object is accessed, we would not target that object another time. This allows us to maintain data integrity as we can avoid accessing the same object multiple times, thereby avoiding any redundancies.

Image No 1



Steps for Invoking JSON-based Web Service

IV. DATA TRANSFER INBETWEEN SERVER AND CLIENT

4.1. Server-Side Operations:

4.1.1. Data Transmission:

- Create JSONObject Java object
- Add name/value pairs using put method
- Convert it to String type using toString method and send it to the client with content-type as "text/xml" or "text/plain"

Example:

myString = new JSONObject (). put ("JSON","Hello, World!").toString(); // myString is {"JSON": "Hello, World"}

4.1.2. Data Reception:

- Read the JSON data as a String type
- Create JSONObject Java object from the string

Example:

String json = readJSONStringFromRequestBody(request); //Use the JSON-Java binding library to create a JSON object in Java JSONObjectjsonObject = null;

```
try
{
jsonObject = new JSONObject(json);}
catch(ParseExceptionpe) {}
```

4.2. Client-Side Operation:

4.2.1. Data Transmission:

- Create JSON JavaScript object
- Use "POST" HTTP method in the open method of the XMLHttpRequest object
- Pass JSON JavaScript object in the send method of XMLHttpRequest object


```
Example:
varcarAsJSON = JSON.stringify(car);
varurl = "JSONExample?timeStamp=" + new Date().getTime();
createXMLHttpRequest();
xmlHttp.open("POST" , url, true);
xmlHttp.onreadystatechange = handleStateChange;
xmlHttp.setRequestHeader("Content-Type" , "application/x-www-form-urlencoded");
xmlHttp.send(carAsJSON);
```

4.2.2. Data Reception:

- JSON data is received as a string
- Calling eval() will generate JSON object in JavaScript code
  varJSONdata = eval(req.responseText);
- Once you have JSON object, you can use . notation to access its properties
  var name = JSONdata.name;
  var address = JSONdata.addresses[3];
  varstreetname = JSONdata.addresses[3].street;

V. MODIFYING JSON

JSON retrieved from a JSON feed is always going to be read-only; as such data feedsdo not provide functionality to modify their data from unverified sources. There are many cases where we would want to ingest the data from an external data feed, and then modify that content as per our requirements.JSON provides us with features which help modify json objects that represent json arrays and object. Json declare the following set() methods for modifying json array elements and json object properties:

- Json set(int index ,Object value)
- Json set(String propName,json value)
- Json set(String property ,Object value)

```
string json = @"{
  'channel': {
     'title': 'Star Wars',
    'link': 'http://www.starwars.com',
    'description': 'Star Wars blog.',
    'obsolete': 'Obsolete value',
     'item': []
   }
 }";
JObjectrss = JObject.Parse(json);
JObject channel = (JObject)rss["channel"];
channel["title"] = ((string)channel["title"]).ToUpper();
channel["description"] = ((string)channel["description"]).ToUpper();
channel.Property("obsolete").Remove();
channel.Property("description").AddAfterSelf(new JProperty("new", "New value"));
JArray item = (JArray)channel["item"];
item.Add("Item 1");
item.Add("Item 2");
Console.WriteLine(rss.ToString());
```

VI. AJAX CALLS WITH JSON DATA

In AJAX (AsynchronousJavaScript and XML), web requests are made via JavaScript and the data interchangeoriginally happened in XML. The "X" in AJAX was originally considered to be XML,but today it can be any data interchange format, such as XML, JSON, text file, oreven HTML. The data format being used for the data transfer has to be mentionedin the MIME type headers.

Dynamic web development is all about data transfer between two parties, the client and the server. We use programs such as a web server, a database, and a server-side programming language to fetch and store dynamic data. In AJAX (Asynchronous JavaScript and XML), web requests are made via JavaScript and the data interchangeoriginally happened in XML. Asynchronous requests allow developers to divide web pages into multiple components independent of each other, thereby saving a lot of memory by sending data on demand.

Table No 1

```
<html>
<head>
<link rel="stylesheet" href="example1.css" type="text/css"/>
<script type="text/javascript">

 var ajax;

 function ajaxCreate(){
  if(window.XMLHttpRequest){
   //For IE7+, Firefox, Chrome, Opera, Safari
   return new XMLHttpRequest();
  } else {
   //For IE6, IE5
   return new ActiveXObject("Microsoft.XMLHTTP");
  }
 }

 function receiveData(){
  if(ajax.readyState == 4){
   if(ajax.status == 200){
    var content = document.getElementById('content');
    content.innerHTML = ajax.responseText;
   }else{
    alert("Server process error");
   }
  }
 }

 function sendRequest(url){
  ajax = ajaxCreate();

  if(!ajax){
   alert("Browser is not compatable with XMLHttpRequest");
   return 0;
  }

  ajax.onreadystatechange = receiveData;
  ajax.open("GET", url, true);
  ajax.send(null);
 }

</script>
</head>

<body>
 <div class="letters">
  <div class="letter" id="letter-a">
   <h3>
    <a href="javascript:sendRequest('example1.php')">A</a>
   </h3>
  </div>
 </div>
 <div id="content"></div>
</body>
</html>
```

Basic AJAX call sample code for server and client-side application

Table No 2

Properties	Description [3]
onreadystatechange	A JavaScript function object that is called whenever the readyState attribute changes. The callback is called from the user interface thread.
readyState	Returns values that indicate the current state of the object.
responseText	The response to the request as text, or null if the request was unsucessful or has not yet been sent.
responseXML	The response to the request as a DOM Document object, or null if the request was unsuccessful, has not yet been sent, or cannot be parsed as XML. The response is parsed as if it were a text/xml stream.
status	The status of the response to the request. This is the HTTP result code (for example, status is 200 for a successful request).
statusText	The response string returned by the HTTP server. Unlike status, this includes the entire text of the response message ("200 OK", for example).

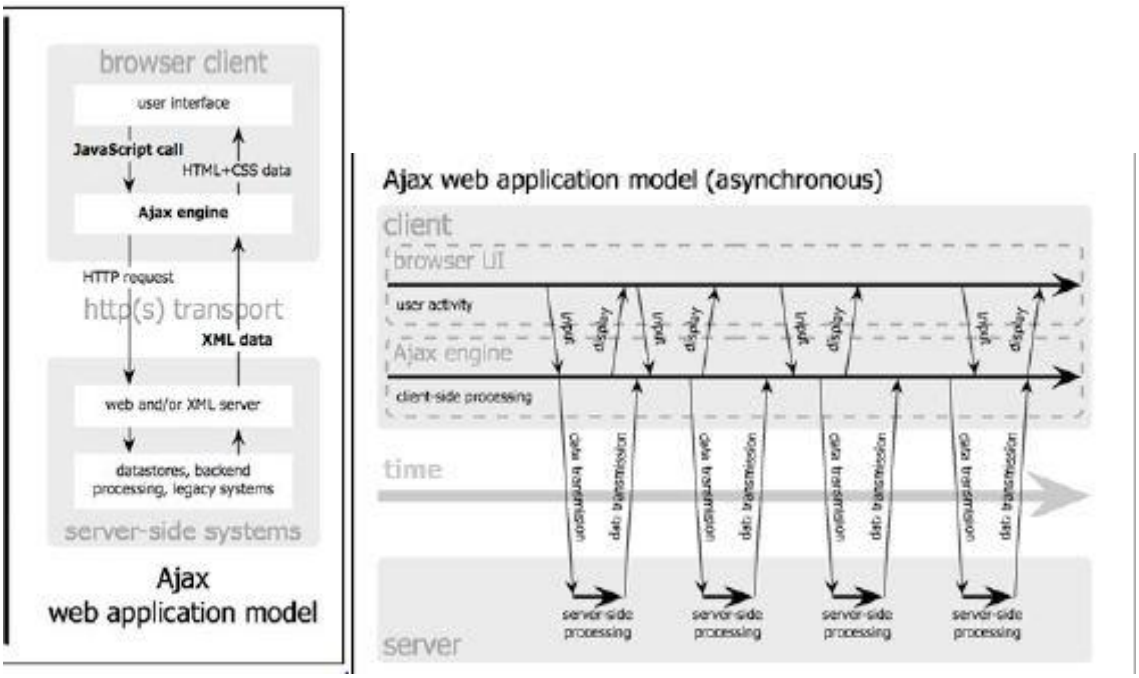
Value	State	Description
0	UNINITIALIZED	open() has not been called yet.
1	LOADING	send() has not been called yet.
2	LOADED	send() has been called, and headers and status are available.
3	INTERACTIVE	Downloading; responseText holds partial data.
4	COMPLETED	The operation is complete.

Description of various Ajax properties

VII. JSON-REMOTE PROCEDURE CONTROL

- JSON-RPC is a simple remote procedure call protocol similar to XML-RPC although it uses thelightweight JSON format instead of XML.
- JSON-RPC-Java is a Java implementation of the JSON-RPC protocol.
- It allows you to transparently call server-side Java code from JavaScript with an included lightweight JSON-RPC JavaScript client
- It is designed to run in a Servlet container such as Tomcat and can be used with J2EE Application servers to allow calling of plain Java or EJB methods from within a JavaScript DHTML web application

Image No 2

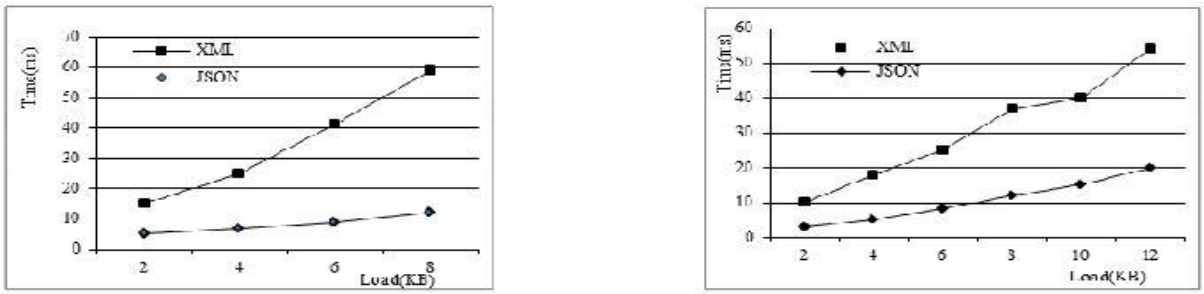


Asynchronous Interaction Using AJAX

VIII. EXPERIMENTAL EVALUATION

In this section, experiments were employed to verify the performance of the DMT driven JSON processingapproach. It was compare with that of the XML-based web services. Experiments were conducted on a PCwith 1.73 Intel Core Duo CPU, 2GB Memory installed with Windows XP Profession SP3. The JVM used is JDK1.6.0 and Apache Tomcat 5.5 was exploited as the Web container. XML and JSON data wasparsed with Simple 2.5.2 and Jackson1.7.6, respectively. Experiments were conducted to investigate the scalability of serializing/deserializing java data objects to/from JSON and XML data objects. The time was measured for serializing/deserializing as the size of data objects changed from 2kb to 12kb.

Graph No 1



(a) Comparison of Serialization (b) Comparison of Deserialization

Graph No1: Comparison of Scalability between JSON and XML

Graph No1 presents the result of the experiments. Graph No 1(a) describes the measurement of time needed for the serialization of Java data objects to JSON/XML. From the figure, we can see that, the time needed for serializing java data objects both increases to JSON and XML data objects. However, for JSON data, it increases much slower than that for XML data, which means the scalability for using JSON as the data format for exchanging has better performance than using XML. Graph No1(b) illustrates the deserializingJSON/XML data objects to java data objects. Similar to serialization, the performance of deserializingJSON data objects to java data objects has surpassed that of deserializing XML data objects to java data objects.

REFERENCES

[1].Drew Adams, "Oracle®Database JSON Developer's Guide" , 12c Release 2(12.2) E85247-01,Feb 2017

[2]. Sai Srinivas Sriparasa, "Javascript and JSON Essentials", PACKT Publishing,ISBN-10: 1783286032, January 2013, pp.45-60.

[3]. Jeff Friesen "Java , XML and JSON",Apress Publication, ISBN 978-1-4842-1916-4, June 2016,pg-165-177.

[4]. Chen-Hsiang (Jones) Yu "AJAX and JSON with jQuery", CSAIL(MIT Press Bookstore), Jan 2010.

[5]. D. PENG Boas, L. CAO and W. XU.Oliveira, Using JSON forData Exchanging in Web Service Applications, Journal of ComputationalInformation Systems 7:16, 2011, 5883-5890.

[6].Sang Shin, "Introduction to JSON", Sun Microsystem Inc.

