CIT360 – W03 – JSON Notes

May 6, 2019

FROM: CIT-360 Coding Topics Introduction.pptx

* Summary
  + This technology is a simple text-based format that allows for the communication of objects across platforms. I can send a simple or complex object from one server to another using this format.
  + There are a couple of different parsers available for JSON. One is the simple parser that comes from Json.org. Another is one create by Google called Jackson.
* References
  + <https://examples.javacodegeeks.com/core-java/json/java-json-parser-example>
  + <http://www.mkyong.com/java/jackson-2-convert-java-object-to-from-json/>
  + <https://www.tutorialspoint.com/json/json_java_example.htm>
* Tips to get to Basic
  + I want to see more than a web page that uses JSON with JQuery. I would like to see a Java program that sends and receives JSON and see object -> JSON and JSON -> object.

FROM: “Doing More With JAVA”

*<https://www.github.com/yenrab/qcJSON>*

QCJSON and JSON:

JavaScript Object Notation – JSON Serialization –

1. Why do serialization when every Java object has a toString method?

2. Why are there different types of serialization?

3. Why use streams? Why not write directly to a file?

“Choosing JSON yields human and machine readable code that minimizes the size of the data to be stored or sent. This can become significant when, instead of writing to a file, you and millions of your applications’ users are sending the data to one or more machines across a network.”

JSON VS. XML – XML is wordy

“type of serialization”

“To *serialize* an object means to convert its state to a byte stream so that the byte stream can be reverted back into a copy of the object.”

JSON serialization

Binary serialization (not human readable)

XML serialization

JSON was originally was used with JavaScript however implementations of JSON libraries exist in many different languages.

The JSON specification allows for several different types of elements. The two most obvious are arrays and objects. Arrays in JSON are contained within square brackets and the individual elements of the array are separated by commas.

Arrays

Java – {1, 2, 75, 13}

JSON – “[1, 2, 75, 13]”

JSON Representation of an instance of the Person class

{“age”:24, “firstName”:”Sue”, “lastName”:”Oskarson”, “phoneNumber”:”(208)234-8888”}

There are many JSON libraries and several are written in JAVA. QCJSON is one of them

QCJSON library available at

[*https://www.github.com/yenrab/qcJSON*](https://www.github.com/yenrab/qcJSON)

**Streams and JSON**

JSON data is written as name/value pairs, just like JavaScript object properties.

A name/value pair consists of a field name (in double quotes), followed by a colon, followed by a value:

"firstName":"John"

JSON names require double quotes. JavaScript names do not.

**JSON Arrays**

JSON arrays are written inside square brackets.

Just like in JavaScript, an array can contain objects:

"employees":[  
    {"firstName":"John", "lastName":"Doe"},   
    {"firstName":"Anna", "lastName":"Smith"},   
    {"firstName":"Peter", "lastName":"Jones"}  
]

In the example above, the object "employees" is an array. It contains three objects.

Each object is a record of a person (with a first name and a last name).

<https://www.w3schools.com/whatis/whatis_json.asp>

<https://www.geeksforgeeks.org/parse-json-java/>

Java has two basic types of streams that are used to move information, ***InputStreams*** and ***OutputStreams***. ***InputStreams*** are used to pull data in to your application. You push data out using ***Output-Streams***.

There are several classes that inherit from ***InputStream*** and ***Output-Stream***. Two of these, used to read and write data to files, are ***FileInputStream*** and ***FileOutputStream***. For a basic description of these streams take a look in the Doing Stuff with Java book. Java also allows you to ‘wrap’ these streams with additional functionality as you need or want. The QCJSON library takes advantageof this by creating two ‘wrappers’, ***JSONOutputStream*** and ***JSONInputStream***. As seen in Code Snippet 4, if you want to write JSON to a file, instead of using the JSONUtilities parse and stringify methods you use these two JSON streams. You do this by sending a ***FileOutputStream*** object to the constructor of the ***JSONOutputStream***.

<https://www.oracle.com/technetwork/articles/java/json-1973242.html>

**The Java API for JSON Processing provides portable APIs to parse, generate, transform, and query JSON.**

Published July 2013

[JSON](http://json.org/) (JavaScript Object Notation) is a lightweight, text-based, language-independent data exchange format that is easy for humans and machines to read and write. JSON can represent two structured types: *objects* and *arrays*. An object is an unordered collection of zero or more name/value pairs. An array is an ordered sequence of zero or more values. The values can be strings, numbers, booleans, null, and these two structured types.

Listing 1 is an example from [Wikipedia](http://en.wikipedia.org/wiki/JSON) that shows the JSON representation of an object that describes a person. The object has string values for first name and last name, a number value for age, an object value representing the person's address, and an array value of phone number objects.

{

"firstName": "John",

"lastName": "Smith",

"age": 25,

"address": {

"streetAddress": "21 2nd Street",

"city": "New York",

"state": "NY",

"postalCode": 10021

},

"phoneNumbers": [

{

"type": "home",

"number": "212 555-1234"

},

{

"type": "fax",

"number": "646 555-4567"

}

]

}

**Listing 1. Example of JSON representation of an object**

JSON is often used in Ajax applications, configurations, databases, and RESTful web services. All popular websites offer JSON as the data exchange format with their RESTful web services.

**JSON Processing**

The Java API for JSON Processing ([JSR 353](http://jcp.org/en/jsr/detail?id=353)) provides portable APIs to parse, generate, transform, and query JSON using object model and streaming APIs.

The object model API creates a random-access, tree-like structure that represents the JSON data in memory. The tree can then be navigated and queried. This programming model is the most flexible and enables processing that requires random access to the complete contents of the tree. However, it is often not as efficient as the streaming model and requires more memory.

The streaming API provides a way to parse and generate JSON in a streaming fashion. It hands over parsing and generation control to the programmer. The streaming API provides an event-based parser and allows an application developer to ask for the next event rather than handling the event in a callback. This gives a developer more procedural control over the JSON processing. Application code can process or discard the parser event and ask for the next event (pull the event). The streaming model is adequate for local processing where random access of other parts of the data is not required. Similarly, the streaming API provides a way to generate well-formed JSON to a stream by writing one event at a time.