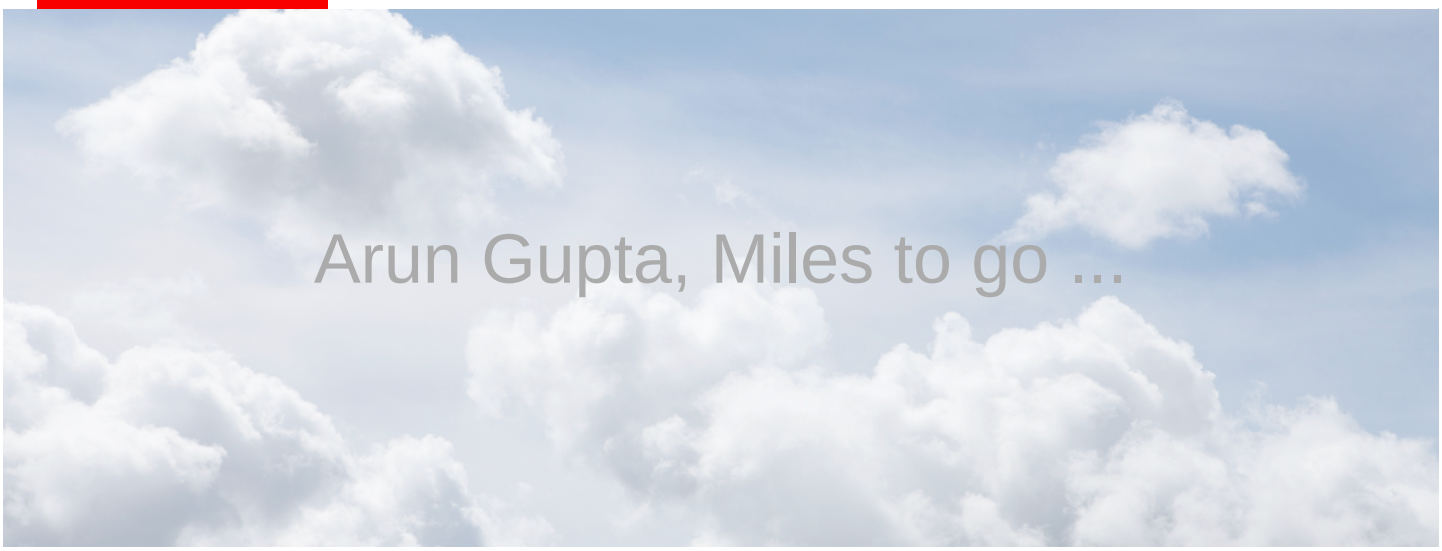

WEB2.0 | Thursday, March 1, 2007

Language-neutral data format: XML and JSON

By: Guest Author

XML and JSON are the two prevalent choices for language-neutral data format. That means a format used to exchange data between client and server, independent of the language used on each end. We are familiar with XML pointy bracket syntax which has served us well so far. With Rich Internet Applications becoming more common, there is a need to have a light-weight data interchange format. And so JSON is catching up (11% for data transfer in 2006).

Basically, JSON is built on two structures:

- A collection of name/value pairs with unique names (*associative array*)
- An ordered list of values (*array*)

See message samples formatted in JSON and equivalent XML. Tim Bray summarizes when to use which format.

Here is a collection of interesting articles in case you want to dig deeper:

- JSON: The Fat-Free alternative to XML
- What XML, SOAP and XML-RPC does not have, that JSON does ?
- JSON vs XML: The Debate
- Comparing processing time of JSON and XML
- Bi-directional conversion between JSON and XML
- JSON for Ajax Web services
- XML, HTML or JSON ?

The key advantages of JSON I derived from my reading of the above articles are:

- Much simpler than XML because it is not a markup language and a natural representation of data.

- JSON is better data exchange format, XML is a better document exchange format.

- JSON is easier to read for machines with no/thin client-side library.

- JSON is a natural fit for data consumption by browser clients, for example Ajax components.

- Ability to represent general data structures: records, lists and trees.

- Wikipedia entry for JSON reports parsing and generating JSON support in 21 languages.

There are some disadvantages as well:

- JSON format is hard to read for humans; for example complicated-looking syntax, like the

  ```
  }}]}
  ```

  at the end of data snippet is frightening and debugging pain.

- JSON is a newer format so not enough tools to help with authoring & parsing. Some available are:
  - JSON Tools - Java Tools for the JSON Format (parser, renderer, serializer, mapper, validator)

    - JSON-lib - Java library for transforming beans, maps, collections, java arrays and XML to JSON and back again to beans.

    - JSON in Java - Java APIs from json.org (see more below)

    - JSON-taglib - JSON-taglib is a JSP 2.0 tag library used to render JSON data from within JSP code.

    - Could not find an editor that would allow me to edit JSON objects.

- JSON does not have a `<[CDATA[]]>` feature, so it is not well suited to act as a carrier of sounds or images or other large binary payloads.

- Unlike XML, JSON does not provide any display capabilities because it is not a document markup language. JSON was not even intended for that purpose.

- JSON is not extensible - it does not need to be because it's not a document markup language.

In jMaki, we use JSON in Java.
Here is a sample code to create a JSON object using these APIs:

```
import org.json.\*;
import java.io.\*;

public class JSONSample {
  public static void main(String[] args) throws Exception {
```

```
        // basic object creation
        JSONObject person = new JSONObject();
        person.put("name", "duke");
        person.put("age", "10");
        System.out.println(person.toString());

        // how to create array and write to a "writer"
        JSONObject address = new JSONObject();
        JSONArray array = new JSONArray();
        array.put("4140, Network Circle");
        array.put("Santa Clara");
        array.put("CA - 95054");
        address.append("address", array);
        OutputStreamWriter osw = new OutputStreamWriter(System.out);
        address.write(osw);
        osw.flush();

        // XML->JSON conversion
        JSONObject likes = XML.toJSONObject("<likes><running/><skiing/>
</likes>");
        System.out.println(likes.toString());
      }
    }
```

And here is the corresponding output:

```
{"age":"10","name":"duke"}
{"address":[["4140, network circle","Santa Clara","CA - 95054"]]}
{"likes":{"skiing":{},"running":{}}}
```

This API also allows conversion from comma-delimited text, HTTP, Cookie, and CookieList to JSON conversions. The source code for JSON

in Java is freely available
but here are two suggestions for ease-of-use:

1. Provide a jar file that is ready to use

2. Publish the link to framed
   version of javadocs on the main page since that is
   more useful.

In summary, XML is document-oriented and JSON is data-oriented. So if you
want to deal with highly structured documents that requires a complex structure,
binary data, exact ordering of elements and be able to render itself then use
XML. OTOH, if you are focused on light-weight data exchange then JSON is the way
to go.

Follow the JSON blog and enjoy!

Technorati: XML JSON

DataFormat JavaScript

Web2.0 Ajax

jMaki

Join the discussion