XML VERSUS JSON

by Jeremy Fonte
11/11/2013

There are a seemingly infinite number of file formats, both binary and text-based - CSV files, XML, JSON, docx, PDF, JPEG, PNG, etc. When it comes to storing data for use within and between applications, the two dominant formats are JSON and XML. What follows is an outline of the pros and cons of each format.

XML is a powerful, verbose markup language with an expansive set of supporting technologies. There's XPath for navigating documents, and XSLT (eXtensible Stylesheet Language Transformation) to transform XML documents into other XML documents or completely different file formats. There are two technologies for defining the structure of an XML document - the older and simpler DTD (Document Type Definition), and the newer, more verbose and more capable XSD (XML Schema Definition).

XML's strength and weakness lies in its abundance of semantic markup. [Nearly] every piece of information in the document is part of a centralized tree - the main exceptions being the XML declaration, processing instructions, and the DOCTYPE declaration at the top of the document, before the root node. Every other piece of information provides a window into the meaning of the data.

This wealth of structured semantic data makes XML arguably a more human-readable format than JSON, which can be terse and slightly vague at times. Furthermore, it can sometimes be easier to read code that produces XML than code that writes JSON, assuming legible tag names are used. If you're using an established XML doctype, you might not have the luxury of a carefully thought-out naming convention; sadly, it's fairly often that XML tag names make little sense, even if you're familiar with the business logic behind the system.

JSON, on the other hand, will be a very comfortable format if you've spent much time programming in JavaScript. Experience with PHP, Java, C#, or pretty much any other C-style language will reduce the learning curve significantly. Basically, any given JSON file is a single JavaScript object literal. This parent object can contain other object literals, arrays, or variables. There are JSON libraries available for most popular languages, easing the workload to get JSON built, imported or exported.

AJAX is a prime example of the use of both XML and JSON. XML was the original data format for the XMLHttpRequest object, but nowadays libraries like jQuery offer simple convenience functions for performing JSON AJAX requests. One major benefit of using JSON for AJAX is the JSONP technique, which allow for AJAX requests to be made without the limiting effect of the same-origin principle. Generally, AJAX requests must be made to resources on the same domain - but by injecting a script tag filled with JSON data into the page, JSONP allows for AJAX requests from and to any compatible clients and servers.

My overall feeling is that JSON is a superior data format for web development, especially AJAX-based JavaScript filled web apps. XML, on the other hand, is likely a better format for heavy duty desktop or server-side processes, where it's important to be able to validate, transform, and transfer clearly defined documents and data.