Wednesday, November 18, 2009

Simpler than JSON?

With XML falling out of fashion, JSON is all the rage these days.

The last few years we've been using a simple online store application to demo application composition with SCA. That store app allows you to pick fruits and vegetables out of a catalog and put them in your shopping cart.

There's versions of the store app in different languages, Java, Python, and an HTML+JavaScript client in the Apache Tuscany project, and more languages to come over time. The shopping cart is currently represented in XML in an ATOM feed.

I've been wondering... Should we switch from XML to JSON? Is JSON simpler? Is the JavaScript syntax so great? Any alternatives? Let's try and compare different representations of our shopping cart:

XML:
Here's the XML currently produced from Java bean representations of the cart, fruits and vegetables using some fancy JAXB code:

```
<ns2:root xmlns:ns2="http://tuscany.apache.org/xmlns/sca/databinding/jaxb/1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="cart">
  <items xsi:type="fruit">
    <name xmlns="">Apple</name>
    <price xmlns="">2.99</price>
  </items>
  <items xsi:type="fruit">
    <name xmlns="">Orange</name>
    <price xmlns="">3.55</price>
  </items>
  <items xsi:type="vegetable">
    <name xmlns="">Broccoli</name>
    <price xmlns="">1.99</price>
  </items>
</ns2:root>
```

Yuk...

Here's without the namespace mess, as it's not needed by the store app to disambiguate what's flowing (as the app knows what it's doing), and would probably over-complicate any non-XML representation as well...

```
<cart>
  <fruit>
    <name>Apple</name>
    <price>2.99</price>
  </fruit>
  <fruit>
    <name>Orange</name>
    <price>3.55</price>
  </fruit>
  <vegetable>
    <name>Broccoli</name>
    <price>1.99</price>
  </vegetable>
</cart>
```

That looks a little better.

Here's with attributes instead of elements. It's shorter and should work too, I guess?

```
<cart>
  <fruit name="Apple" price="2.99"/>
  <fruit name="Orange" price="3.55"/>
  <vegetable name="Broccoli" price="1.99"/>
</cart>
```

The problem is that over the years I've started to develop an allergy to XML (mostly because of the usual namespace mess), so let's go with the flow, be cool, and try JSON...

JSON:
Here's a possible JSON representation. The cart contains an array of objects, actually two levels of objects to distinguish fruits and vegetables.

```
{"cart":[
  {"fruit":{"name":"Apple","price":2.99}},
  {"fruit":{"name":"Orange","price":3.55}},
  {"vegetable":{"name":"Broccoli","price":1.99}}]}
```

Another try, with the funky javaClass attribute expected by the Java JSON code to figure the type of each item in the cart... Not great.

```
{"cart":[
  {javaClass:"sample.Fruit","name":"Apple","price":2.99},
  {javaClass:"sample.Fruit","name":"Orange","price":3.55},
  {javaClass:"sample.Vegetable","name":"Broccoli","price":1.99}]}
```

Here's another one following the Jettison or Badgerfish XML / JSON mapping. Multiple items of the same name are grouped in an array field. So in my example fruit is an array, vegetable is not. Now, add another vegetable to your cart, and vegetable morphs into an array... Confusing for lack of a better word :)

```
{"cart":{  "fruit":[
    {"name":"Apple","price":2.99},
    {"name":"Orange","price":3.55}],
  "vegetable":{"name":"Broccoli","price",1.99}
}}
```

Now I'm a little disappointed. JSON doesn't look significantly simpler than XML here, and Javascript concepts like arrays, (soft) objects and confusing mappings of multiple occurences of items are starting to bleed in my cart.

Let's try a little harder and find alternative representations:

YAML:

```
cart:  - type: fruit
    name: Apple
    price: 2.99
  - type: fruit
    name: Orange
    price: Pear
  - type: fruit
    name: Orange
    price: 3.55
  - type: vegetable
    name: Broccoli
  price: 1.99
```

I still need to add an artificial 'type' attribute to each item to represent different fruit and vegetable elements, but YAML is not bad for the cart example which has only one level of nesting. With more levels of nesting, YAML gets a little akward in my opinion (see section 2.1, lists of lists and mappings of mappings, in the YAML spec).

YAML also lacks the nice programming language feel of JSON, which I can just evaluate in the JavaScript interpreter, allowing me to blur the line between my shopping cart data and the code that uses it.

Scheme:

```
'(cart
  (fruit (name "Apple")(price 2.99))
  (fruit (name "Orange")(price 3.55))
  (vegetable (name "Broccoli")(price 1.99)))
```

That Scheme language representation is not bad at all!
- it's really concise;
- the Scheme syntax is simple and easy to parse, () for lists, space as a separator;
- as the above expression is just constructing lists, I don't need to make my shopping cart data fit in Objects, Arrays or Maps;
- it's easy to distinguish symbols like fruit or name and user data like "Apple".
- a list can simply be tagged with a fruit or vegetable symbol to indicate its type;
- like JSON, the above expression is a piece of code easy to evaluate and test in a Scheme interpreter.

Python:
Another variation on the same theme with Python tuples, as Python is more widely known than Scheme or Lisp.

```
('cart',
  ('fruit',('name','Apple'),('price',2.99)),
  ('fruit',('name','Orange'),('price',3.55)),
  ('vegetable',('name','Broccoli'),('price',1.99)))
```

A little more verbose than scheme, but not bad.

So, I must admit... I think I much prefer the Scheme representation, or the Python tuple syntax. I realize that Scheme was created in the 70's, but hey, that doesn't mean it's bad :) -- see the Wikipedia entry on Scheme for its history.

I also realize that JSON is natively supported by the JavaScript interpreters used in Web browsers... but it looks like most people are using handcrafted JSON parser implementations (written in JavaScript) instead of the native JavaScript eval() or JSON.parse() so that argument is not very solid... and it shouldn't be hard to write similar parsers for the Scheme or Python syntaxes as well.

Oh, by the way... check-out that JSON parser page. Here's what it says:
JavaScript is a general purpose programming language that was introduced as the page scripting language for Netscape Navigator. It is still widely believed to be a subset of Java, but it is not. It is a Scheme-like language with C-like syntax and soft objects.

Why not just stick with Scheme's minimalistic syntax then? Why would I need JSON's C-like syntax or these fancy soft objects to represent my shopping cart? :)

Let me know what you think...