Elemental XML · glenn mcdonald

Elemental XML is a recommended voluntary discipline to use a syntactically reduced subset of XML for data representation:

- elements, but no attributes, empty tags, entities, namespaces or processing instructions
- element names can have only letters, numbers and _, and are case-insensitive
- an element can contain sub-elements or text, but not both
- an element can be repeated; sibling order is significant in this case and no other
- an element cannot contain sub-elements with the same name
- only declared elements are parsed, so only declared tags need to be escaped in text

In structural terms, Elemental XML provides ordered arrays, named hashes, non-recursive nesting and lexical opaqueness.

An **XML Model** is a simplified declaration of the structure of an Elemental XML document, with sufficient rigor to allow useful computed validity, but sufficient clarity for human authors to easily assess a simple document's validity according to its model, and to easily produce a model from an example document and vice versa.

An XML Model is largely an Elemental XML document with everything except the tags removed, and repetition collapsed (but decorative whitespace is allowed). The first character in each Model element is one of four symbols indicating that there must be exactly one (.), zero or one,(?), one or more (+), or zero or more (*) of this element. Rules on sub-elements apply to their sub-appearance within the parent element, while the rule on the parent element applies to the appearance of the whole set.

In the other direction, an XML Model is inherently a document template: copy it and replace the .?+* symbols with actual data.

An Elemental XML document does not explicitly specify an XML Model, and since Elemental XML ignores undeclared structure, an Elemental XML document's validity might be tested against multiple XML Models of varying specificity or scope. (Reuse of container elements, for which no explicit mechanism is provided, can thus be done by defining a reused container opaquely in a top-level XML Model, and also validating the document against a sub-Model of the reused container's substructure.)

Here are some notable advantages of this reduction:

- eliminating the element/attribute distinction simplifies data design, writing and reading
- lexical opaqueness facilitates passing one level's markup through others, notably allowing for unescaped HTML within non-colliding Elemental XML wrapper tags
- eliminating recursion allows arbitrary element extraction by simple regular-expression matching, instead of requiring the whole document to be parsed into a tree before use (which isn't possible for the otherwise structurally similar JSON or S-expressions)
- full XML validity is not required, but allowed

A simple Elemental XML document and its Model are shown to the right. The Model declares that this document type must contain one Blog element. The Blog element contains one or more BlogEntry elements. Each BlogEntry must contain an ID and a Date, and may have an optional Title, Body, and References section. The References section may have any number of Tag elements and any number of SeeAlso elements. Each SeeAlso element, if any, must have an saRef element, and may have an optional saTitle element. ID, Date, Title, Tag, saTitle and saRef are text, or more precisely, are undeclared here and thus opaque to this Model, but must contain something other than whitespace.

Note that the order of BlogEntry elements is structurally significant, as is the order of Tag elements within any one References section. The relative order of ID, Date, Title, Body and References within a containing BlogEntry element, however, is not significant, and they can thus appear in any sequence. Substructure is named or ordered, not both.

Note also that because the Model does not define the <a> element, it does not need to be escaped inside of the Body elements. In fact, the Body element *must* be opaque for Elemental XML purposes, since it intermingles text and tags.

A more-detailed explanation of Elemental XML, and a simple Perl implementation of a validator, are available at the URL below.

an Elemental XML document

```
<?xml version="1.0"?>

<Blog>

<BlogEntry>
<ID>157</ID>
<Date>Wed, 31 May 2006 22:00:00
EDT</Date>
<Title>BarCampBoston</Title>
<Body>See you <a href="http://
barcamp.org/BarCampBoston">there</
a>.</Body>
<References>
<Tag>community</Tag>
<Tag>hacking</Tag>
<Tag>anarchy</Tag>
<SeeAlso>
<saTitle>Elemental XML</saTitle>
<saRef>156</saRef>
</SeeAlso>
</References>
</BlogEntry>

<BlogEntry>
<Date>Tue, 30 Aug 2005 11:15:00
EDT</Date>
<Title>Cool idea for using a simplified
subset of XML...</Title>
<Body>Check out <a
href="http://www.furia.com?type=code&
id=ElementalXML">this</a>.</Body>
<ID>156</ID>
<References>
<Tag>hacking</Tag>
<Tag>simplify</Tag>
</References>
</BlogEntry>

</Blog>
```

an Elemental XML Model

```
<?xml version="1.0"?>

<Blog>.
  <BlogEntry>+
    <ID>.</ID>
    <Date>.</Date>
    <Title>?</Title>
    <Body>?</Body>
    <References>?
      <Tag>*</Tag>
      <SeeAlso>*
        <saTitle>?</saTitle>
        <saRef>.</saRef>
      </SeeAlso>
    </References>
  </BlogEntry>
</Blog>
```