XML and TPF - *more < > please*

by Morry Veer

OK, so imagine we have a nice character stream of EBCDIC characters representing an XML document. What in heaven's name does an XML document look like? Well, it looks like HTML. And it has lots of <'s and >'s. An example is always good.

```
<?xml version="1.0"?>
<Booking>
  <Name>Morry Veer</Name>
  <Itinerary>
    <Flight airline="UA" number="889" date="12AUG" class="F" />
    <Flight airline="UA" number="887" date="20AUG" class="F" />
  </Itinerary>
  <ContactInformation>
    <HomePhone>3035551212</HomePhone>
    <Email>morryspam1@yahoo.com</Email>
  </ContactInformation>
</Booking>
```

As you can see, there are lots of < and > symbols. The above is formatted nicely with tabs and line breaks, but that is just for your benefit. The below is just as valid as the above:

```
<?xml version="1.0"?><Booking><Name>Morry Veer</Name><Itinerary><Flight
airline="UA" number="889" date="12AUG" class="F"/><Flight airline="UA" num-
ber="887" …
```

Whitespace is generally considered filler. An XML document starts with a line declaring itself as an XML document, and the XML version. Note the ? marks – these indicate processing directives that are not part of the "content" of the document.

The rest is deceptively simple. Tags are Booking, Name, Itinerary, Flight and so on. A start tag is simply <tag> and the end tag is the same, but beginning with a /, as in </tag>. Tags are always paired, unless they are empty tags (which enclose no text). Flight is an example of an empty tag. Empty tags, at their simplest, are formed like this: <tag/>.

Information in an XML document is communicated in two ways, through enclosed text (the text between a start tag and an end tag), and attributes. Attributes are contained within the boundaries of a start tag, and look like simple assignments. airline="UA" is an example of an attribute. 3035551212 is an example of enclosed text.

So where do the tag names and attribute names come from? Essentially, someone just makes them up – in the beginning. But the purpose of XML is to facilitate communication between systems, so there must be a way of communicating the structure of a document, which is of course the document type definition or DTD.

DTDs and more fun acronyms

XML allows the building of a "language". The document type definition is the grammar of that language. It completes the contract between two communicators, which began with the adoption of XML. Using a DTD is **not** a requirement in order to use XML. An XML document can simply exist without one, can imbed the DTD, or can use an xml:Schema, which is a more powerful version of the DTD. In any case, having a DTD is a good idea, as it essentially guarantees the recipient will receive a valid document.

A couple of definitions are needed here. XML documents can be well-formed, and well-formed documents can be valid. A well formed document:

1. Has no unclosed tags. An empty tag is considered closed. All start tags have an end tag.
2. Has no overlapping tags. All tags are nested in a hierarchical order.
3. Attribute values are always enclosed in quotation marks (“).
4. If you need to use the special characters < > and “ as part of a non-markup XML, it must be represented by the entities `<` `>` and `&quot;` respectively. The leading `&` and the trailing `;` are required on entities.

A valid document simply matches the specified DTD. A sample DTD for the above xml document could be:

```
<!— This is an example DTD for the example XML —>
<!ELEMENT Booking (Name+, Itinerary, ContactInformation?) >
<!ELEMENT Itinerary (Flight | Hotel | Car)+ >
<!ELEMENT ContactInformation (HomePhone+, Email*) >

<!ELEMENT Flight EMPTY>
<!ATTLIST Flight
            airline CDATA “AC”
            number  CDATA #REQUIRED
            date    CDATA #IMPLIED
            class   (F, J, Y) “Y”  >

<!ELEMENT Name (#PCDATA)>
<!ELEMENT HomePhone (#PCDATA)>
<!ELEMENT Email (#PCDATA)>
<!ELEMENT Hotel (#PCDATA)>
<!ELEMENT Car (#PCDATA)>
```

Reading from top to bottom, a booking contains Name, Itinerary, and ContactInformation. Name must appear at least once, but may appear more times (the +). ContactInformation is optional – it can appear either once, or not at all, but may not appear multiple times (the ?). Within the Itinerary, either a Flight, a Hotel or a Car tag must appear (the | indicates a selection), and one of these three must appear at least once, and may occur multiple times (the + on the end). This is just like any normal itinerary – you have to have something in your itinerary, but it can be a mix of hotels, cars and flights. Within the contact information, HomePhone may appear 1 or more times, and Email can occur multiple times, or not at all.

In the Flight element, no data can occur as specified by EMPTY. The attribute list for Flight shows that the airline attribute is character data, and defaults to Air Canada if not specified. The number attribute is simply mandatory or required, the date attribute is optional (known as implied), and if not specified does not default to any value, although the end-application will probably assume today’s date. For the class attribute there are only 3 possible enumerated values, F, J and Y, and Y class is the default value.

The remainder of the elements (Name, HomePhone…) contain simply parsed character data, which is simply normal text. They do not, and cannot, contain any other elements / tags.

Links and Resources

UTF-EBCDIC	http://www.unicode.org/unicode/reports/tr16/
XML.org	http://www.xml.org
Worldwide web consortium	http://www.w3.org
Unicode	http://www.unicode.org
TPF TODAY	http://www.tpftoday.com/
This article online	http://members.bigfoot.com/~morryveer/index.html/tpfxml
IBM TPF	http://www-4.ibm.com/software/ts/tpf/
XML for TPF online users guide	http://www-4.ibm.com/software/ts/tpf/pubs/xml/xhome.htm
Apache project	http://xml.apache.org/