© Bob Brown, Transentia Pty. Ltd. 1999, 2000

The Ooga Bird Of Upper Wangala

Introduction

The Ooga bird of Upper Wangala is an extremely rare creature. Only two Ooga birds have ever been seen at any one time and they rarely meet. On those rare occasions when they do meet, however, they produce such beautiful songs that people from all over the world make the long and difficult journey to hear them.

The strange thing about these birds is that they always meet at precisely predictable times and always at one of three trees. Many generations of the nearby Oogaluu people have watched and listened to these birds and have recorded the meeting times and trees quite precisely.

The tourist office of Upper Wangala has decided to create a web page advertising these times in the hope of encouraging more tourists to come and hear the birds (and spending lots of money in the process, of course).

Your task is to produce the web page for the Upper Wangala tourist office using XML and XML-related technologies.

The Exercise

There are a number of tasks required in this exercise. Each task is detailed below.

Requirements

This exercise requires PERL and IE5 to be installed on your system.

Setting Up

Make a new directory for this exercise. Call this directory *OogaMeetings*:

```
C:\> mkdir OogaMeetings
C:\> cd OogaMeetings
```

All the files that you subsequently create as you do this exercise should be contained in this directory.

Create the Oogaluu-Formatted Data File

Create a file called *OogaMeetings.txt*. This will contain the records as the Upper Wangala tourist office gives them to you. This file is to be marked up according to the ages-old traditions of the Oogaluu tribe, as follows:

```
Jan 12,10:42~4!26,14:12~8,2
Feb 9,18:02~12,3
Apr 4,03:12~7!5,03:15~25
May 7,22:19~1,3
Jun 19,23:44~3,2!25,15:05~2!26,09:33~26,3
Jul 7,12:05~14
Aug 13,17:05~12!15,04:22~1,2
Sep 26,01:11~6
Oct 3,03:33~3!12,19:08~3,3
Dec 16,03:47~2
```

Specify an Appropriate DTD for the Data

It is helpful to specify a DTD *before* actually converting the data. This will allow you to check any data you create for validity at an early stage.

Examination of the data shows that the markup format used by the Oogaluu tribe is actually quite simple:

```
month day,time~duration[,tree][!day,time~duration[,tree]]*
```

Notes:

Transentia Pty. Ltd. DonationWare

- the tree on which the birds meet is only specified explicitly if it is not tree 1
- where there are multiple meetings in a month, each meeting is separated by a '!'
- short month names are used throughout, whereas you should use full month names

Create the following DTD in a file called *OogaMeetings.dtd*:

```
<?xml version="1.0"?>
<!ELEMENT OogaMeetings ( MONTH+ )>
<!ELEMENT MONTH ( EVENT+ )>
<!ATTLIST MONTH
  NAME ( January | February | April | May | June | July |
         August | September | October | December) #REQUIRED>
<!ELEMENT EVENT ( DAY, TIME, DURATION )>
<!ATTLIST EVENT
  TREE ( 1 | 2 | 3 ) "1">
<!ELEMENT DAY ( #PCDATA )>
<!ELEMENT TIME ( #PCDATA )>
<!ELEMENT DURATION ( #PCDATA )>
```

Convert the Records to XML

PERL (Practical Extraction and Report Language) is an ideal tool for converting this data to a version marked-up in XML. Create the following PERL script within a file called *OogaMeetings.pl*:

```
use strict;

sub i
  { my $s = "  "; $s x=$_[0]; };

sub XMLMonthStart
  {
  my $name = $_[0];
  my %months =
    (
    Jan => "January", Feb => "February", Apr => "April",
    May => "May", Jun => "June", Jul => "July", Aug => "August",
    Sep => "September", Oct => "October", Dec => "December",
    );
  print &i (1), "<MONTH NAME=\"", $months{$name}, "\">\n";
  };

sub XMLMonthEnd
  { print &i (1), "</MONTH>\n"; };

sub oogaMeetingXMLBegin ()
  {
  print "<?xml version=\"1.0\"?>\n";
  print "<!DOCTYPE OogaMeetings SYSTEM \"OogaMeetings.dtd\">\n";
  print "<OogaMeetings>\n";
  };

sub oogaMeetingXMLEnd ()
  { print "</OogaMeetings>\n"; };

sub XMLEvent
  {
  my ($dy, $t, $du, $tr) = @_;
  print &i (2), "<EVENT";
  print " TREE=\"", $tr, "\"" if @_ gt 3;
  print ">\n";
  print &i (3), "<DAY>\n", &i (4), $dy, "\n", &i (3), "</DAY>\n";
  print &i (3), "<TIME>\n", &i (4), $t, "\n", &i (3), "</TIME>\n";
  print &i (3), "<DURATION>\n", &i (4), $du, "\n", &i (3), "</DURATION>\n";
  print &i (2), "</EVENT>\n";
  };

&oogaMeetingXMLBegin ();
my $in;
while ($in = <STDIN>)
  {
  chomp $in;
  my ($month, $events) = split (/ /, $in);
  &XMLMonthStart ($month);
  foreach my $event (split (/\!/, $events))
    {
    my @info = split (/[!,~]/, $event);
    &XMLEvent (@info);
```

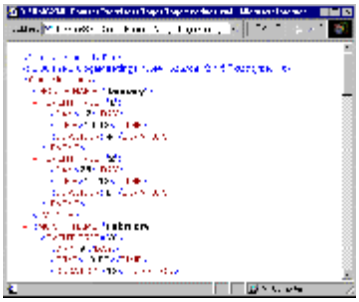
Transentia Pty. Ltd. DonationWare

```
        }
      &XMLMonthEnd ();
      }
    &oogaMeetingXMLEnd ();
```

Once you have created this PERL script, use it to create the file *OogaMeetings.xml*. Execute the following command:

```
C:\OogaMeetings> perl OogaMeetings.pl < OogaMeetings.txt > OogaMeetings.xml
```

You can double-click on the file that you just created to view it in IE5. You should see something like this:



The file is displayed using IE5's default stylesheet for XML.

Some things to note:

- you can collapse a node that is prefixed with a hyphen by double-clicking on it
- a TREE attribute is always shown, regardless of whether or not the original XML explicitly specified a value for the attribute

Develop an Appropriate XSL Stylesheet

IE5's default stylesheet is not very user-friendly (or indeed, pretty!). XSL allows one to do better.

Create the following file (call it *OogaMeetings.xsl*):

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
    <HTML>
      <HEAD>
        <TITLE>Ooga Bird Meeting Schedule</TITLE>
      </HEAD>
      <BODY>
        <H1>Ooga Bird Meeting Schedule</H1>
        <TABLE BORDER="2" WIDTH="100%">
          <xsl:for-each select="OogaMeetings/MONTH">
            <TR>
              <xsl:element name="TD STYLE='font-size:18; font-weight:bold'">
                <xsl:attribute name="BGCOLOR">green</xsl:attribute>
                <xsl:value-of select="@NAME"/>
              </xsl:element>
            </TR>
            <TR>
              <TD>
              <TABLE BORDER="1" WIDTH="100%">
                <THEAD ALIGN="left">
                  <TH BGCOLOR="silver">Day</TH>
                  <TH BGCOLOR="silver">Time</TH>
                  <TH BGCOLOR="silver">Tree</TH>
                  <TH BGCOLOR="silver">Duration</TH>
                </THEAD>
                <TBODY>
                  <xsl:for-each select="EVENT">
                    <TR>
                      <TD>
                        <xsl:value-of select="DAY"/>
                      </TD>
                      <TD>
                        <xsl:value-of select="TIME"/>
                      </TD>
                      <TD>
```

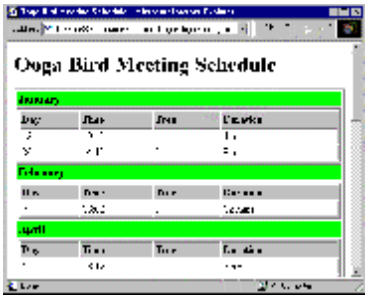
Transentia Pty. Ltd. DonationWare

```
                        <xsl:value-of select="@TREE"/>
                      </TD>
                      <TD>
                        <xsl:value-of select="DURATION"/> mins.
                      </TD>
                    </TR>
                  </xsl:for-each>
                </TBODY>
              </TABLE>
              </TD>
            </TR>
            </xsl:for-each>
          </TABLE>
        </BODY>
      </HTML>
    </xsl:template>
  </xsl:stylesheet>
```

You must also add the following as the **second** line of the *OogaMeetings.xml* file:

```
<?xml:stylesheet type="text/xsl" href="OogaMeetings.xsl"?>
```

If you now double-click on the *OogaMeetings.xml* file, you will see a very different representation of the file, as shown in this screen shot:



You now have a valid, clearly-formatted and displayable representation of the original Oogaluu tribe's data.

Transentia Pty. Ltd. DonationWare