Subject: Re: xsl transformation slow
Posted by braingeyser on Mon, 19 Mar 2007 14:28:53 GMT
View Forum Message <> Reply to Message

braingeyser wrote on Mon, 19 March 2007 03:25

If this logic is also applied to the table_tmp variable created in my original post

<xsl:variable name="table_tmp" select="/root/*[name()=$table]" />
instead of
<xsl:variable name="table_tmp" select="//*[name()=$table]" />
then the transformation time drops again to 8 seconds. But I suspect the data structure for this might not be so fixed, so this one is of dubious validity.

Ok scratch this one. As I suspected the data structure is not fixed sufficiently to make this work for all the places it's used. For example a link1, which has the link table data nested within the outer table data, making the above XPath incorrect for the nested data. The existing XPath is the better for keeping things generic.

If you wanted to make this work, you would need to pass in the context node from the external calling code. I tried this and it works fine, but it means adding an extra parameter in all of the places this template is called. Not a trivial amount of work I'd guess. Doing this makes the whole template a lot clearer though since you do away with most of the complicated xpaths.

If you add this...
<xsl:param name="context_node" />
then you don't need this...
<xsl:variable name="table_x" select="//*[name()=$table]" />

and you can replace this...
<xsl:variable name="field" select="$table_x[position()=$position]/*[name()=$fieldname]" />

with this...
<xsl:variable name="field" select="$context_node/*[name()=$fieldname]" />

The context node parameter (in the case of list1 for eg) comes from std.list1.xsl here...
<xsl:for-each select="//*[name()=$main][count(*)>0]">

```
    !-- display all the fields in the current row -->
    <xsl:call-template name="display_horizontal">
        <xsl:with-param name="zone" select="'main'"/>
        <xsl:with-param name="context_node" select="." />
    </xsl:call-template>
</xsl:for-each>
```

This is the code that would need to be changed in every template call in order to make it work. It speeds things up because your'e no longer getting the same table data node twice, once in the outer calling loop and once more in the display_horizontal template.

The extra time gains seem to be in the order of double, which is not a lot more than you would already get by implmenting the previous changes. In my case, I gained an extra 4 or 5 seconds on my originally 60+ sec page load, now down to between 7 & 8.
