XSLT Keys

Martin Holmes

XSLT Keys

- As your XSLT transformations get more complicated, they'll also start to take a little longer to run.
- This can become a problem when (for instance) you're generating content on-the-fly for a website.
- Time is often wasted because the same template performs the same expensive operation many times.
- `<xsl:key>` can help make your templates more efficient.

The nature of the problem

Imagine that we want to transform our Hamlet file (again). This time, we want to add a copy of the speaker's `<roleDesc>` immediately following the character's name at the beginning of every speech. This template would do it:

```
<xsl:template match="speaker">
  <xsl:apply-templates select="@* | node()"/>
  <xsl:text> (</xsl:text>
  <xsl:variable name="thisSpeaker"
        select="substring-after(parent::sp/@who, '#')"/>
  <xsl:value-of select="//castItem
        [role/@xml:id = $thisSpeaker]/roleDesc"/>
  <xsl:text>)</xsl:text>
</xsl:template>
```

Teaching opportunity: why are we using "substring-after"? What are some alternatives to this?

The nature of the problem

```
<xsl:template match="speaker">
<xsl:apply-templates select="@* | node()"/>
<xsl:text> (</xsl:text>
  <xsl:variable name="thisSpeaker"
  select="substring-after(parent::sp/@who, '#')"/>
  <xsl:value-of select="//castItem
  [role/@xml:id = $thisSpeaker]/roleDesc"/>
<xsl:text>)</xsl:text>
</xsl:template>
```

- Open parenthesis.
- Get the `@xml:id` of the role from the `@who` attribute of the parent `<sp>` element and stash it in a variable.
- Retrieve the matching `<castItem>` element in the cast list, and output its `<roleDesc>` content.
- Close parenthesis.

Inefficiencies in the previous template

Look again at this line:

```
<xsl:value-of select="//castItem[role/@xml:id = $thisSpeaker]/roleDesc"/>
```

Every time this is executed, the XSLT processor has to look through all the `<castItem>` elements to find the matching one. This happens for every `<speaker>` element matched by the template.

`<xsl:key>` to the rescue

- To avoid repeating the same lookup operation every time a template is invoked, we can use `<xsl:key>`.
- `<xsl:key>` builds an index to a specific node.

- It builds the index once, and keeps it available for quick lookups during the rest of the transformation.
- The index works like an **associative array**.

Building an `<xsl:key>` index

An `<xsl:key>` index is built using a single command. The command appears at the top level of your stylesheet, usually near the beginning of the file. It looks like this:

```
<xsl:key name="roles" match="castItem[@type='role']" use="role/@xml:id"/>
```

- This builds a key index named `"roles"`.
- The `@match` attribute specifies the nodes which will be indexed. In this case, it's indexing all `<castItem>` elements whose `@type` attribute is `"role"`.
- The last attribute, `@use`, specifies what we use to look up a particular item in the index.
- Here, we use the `@xml:id` attribute on the `<role>` child of the target `<castItem>` node.

Using an `<xsl:key>` index

Here's our index:

```
<xsl:key name="roles" match="castItem[@type='role']" use="role/@xml:id"/>
```

Here's how we can use it to accomplish the same job as before:

```
<xsl:template match="speaker">
  <xsl:apply-templates select="@* | node()"/>
  <xsl:text> (</xsl:text>
  <xsl:variable name="thisSpeaker"
                select="substring-after(parent::sp/@who, '#')"/>
  <xsl:value-of select="key('roles', $thisSpeaker)/roleDesc"/>
  <xsl:text>)</xsl:text>
</xsl:template>
```

`<xsl:key>` Task (and review of identity transforms)

- Open the *Hamlet* XML: http://web.uvic.ca/~mholmes/dhoxss2013/examples/hamlet.xml
- Create a new identity transform file in Oxygen. Don't forget to:
  - # set the correct `xmlns`
  - # set the correct `xpath-default-namespace`
- Carry out the steps in the previous slides:
  - # Create a template which matches `<speaker>`.
  - # In your template, add a copy of the speaker's roleDesc immediately following the character's name at the beginning of every speech.
  - # Now rewrite the process using the `<xsl:key>`.