XSLTDisableOutputEscaping

What's bad with `disable-output-escaping` in XSLT?

```
<xsl:text disable-output-escaping="yes">&amp;amp;nbsp;</xsl:text>
```

Quoting a message from Michael Kay from March 2001:

"...Cocoon, I think, does further processing of the XSLT output. This is exactly the scenario that makes "`disable-output-escaping`" bad programming practice, and the reason that the XSLT spec says processors aren't obliged to support it. Basically, if the XSLT output is serialized to text and sent straight to the browser, your're probably OK, but if the output is a tree that is then input to further stages of processing, then you aren't..."

J. Pietschmann provided a collection of several issues:

"A JAXP conformant XSLT processor may generate PIs embedded in the output to signal a `disable-output-escaping="yes"` to downstream processing stages. If the serializer finally gets these PIs and understands them, d-o-e will work fine even in Cocoon.

However, there's plenty of stuff that can go wrong:

1. The d-o-e'd stuff is still text to downstream XML processing stages, it can't be processed as elements.
2. Generating the PIs is not required. Xalan does, I'm not sure whether Saxon does.
3. The PIs are not standardized. So you'd better use an identity transformation with the same XSLT processor which generates them for serialization. Fortunately, Cocoon does exactly this. Nevertheless, if you mix processors, you're likely to get trouble.
4. If some processing stage decides to eat or mangle the PIs or to change their position relative to the stuff intended to be d-o-e'd, you're hosed."

Other stuff about `disable-output-escaping` can be found in the XSLT FAQ.