Notes.net

Iris Today

by
Susanna
Doyle

Level: Intermediate
Works with: Designer 5.0
Updated: 07/07/2000

There is continued interest in using XML (Extensible Markup Language) when developing applications, and with good reason. XML has become the *de facto* standard for data interchange, freeing developers from the more mundane issues related to data interchange so they can focus on the data itself and how it will be represented.

As you may know from reading the *Iris Today* article **Exercising XML with Domino Designer**, you can mark up Notes forms and views using XML and display the XML to an XML-compatible browser the same way you would include HTML markup in a Notes database. You can use XSL to filter and style the XML data, or you can use a cascading style sheet (CSS) if you only want to use styles without any filtering.

But suppose you want to keep most of the Domino database design you already have, which Domino automatically serves to browsers as HTML, and include only a small amount of XML data within the HTML in your documents. To do this, you can use an XML data island in a Notes document. A data island is a piece of XML within an HTML page. You can use a little JavaScript to fetch the island, apply XSL, and display the island when the document is loaded.

This article describes the JavaScript you currently need to display an XML data island within a document using XSL in Microsoft Internet Explorer 5, while hiding the code from non-XML compatible browsers. (As browser handling of XML improves, more ways to do this using combinations of JavaScript and XML/XSL may evolve, and you might then be able to change your display formula to hide the XML from fewer browsers.)
For general information about XML and XSL (Extensible Stylesheet Language), see the **World Wide Web Consortium (W3C) XML page**.

You can also download the **Domino 5 Designer Help (with XML)** example database from the Notes.net Doc Library to see the design elements discussed in this article.

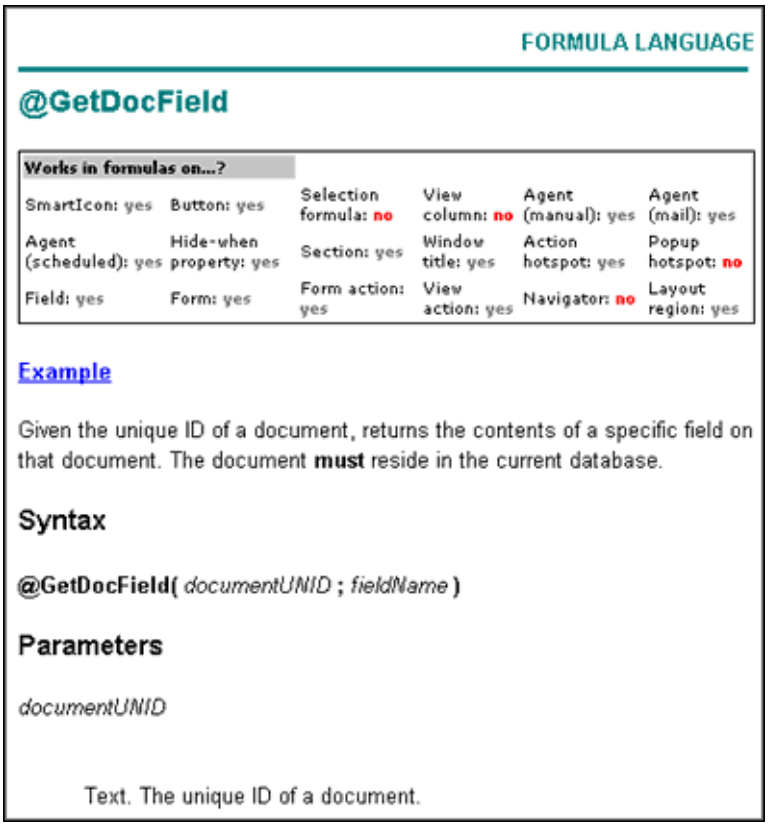
Where to place the island

You can place a data island either on the background of a Notes form or within a rich text field in any document. If the XML is on the background of a form, you can use field data as part of the XML and show the information in many documents, but the XML must appear either before or after any rich text fields (such as the Body field) on the form. If you put an XML data island within a rich text field, you must repeat the XML data in all documents where you want it to appear. Additionally, you can't use field data within a rich text field.

Using an island on a form

Here is an example that uses XML to add a table of context information to @function topics in Domino 5 Designer Help. We've placed the XML on the topic form using a computed subform, but you could also place the XML directly on the form and use a hide-when formula to hide the XML from

non-XML compatible browsers.



FORMULA LANGUAGE

@GetDocField

Works in formulas on...?					
SmartIcon: yes	Button: yes	Selection formula: no	View column: no	Agent (manual): yes	Agent (mail): yes
Agent (scheduled): yes	Hide-when property: yes	Section: yes	Window title: yes	Action hotspot: yes	Popup hotspot: no
Field: yes	Form: yes	Form action: yes	View action: yes	Navigator: no	Layout region: yes

Example

Given the unique ID of a document, returns the contents of a specific field on that document. The document **must** reside in the current database.

Syntax

@GetDocField(*documentUNID* **;** *fieldName* **)**

Parameters

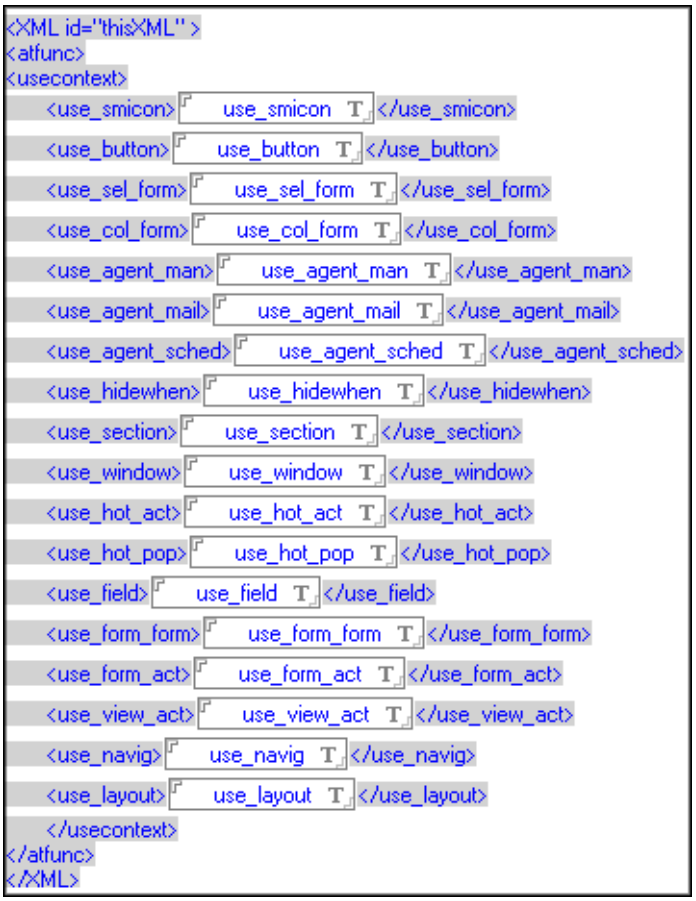
documentUNID

Text. The unique ID of a document.

The XML data for the table actually consists only of yes/no answers for various contexts in which an @function may be used. XSL creates the table, styles the fonts, adds the label text (for example, SmartIcon:), and even performs conditional filtering to show any "no" answer in red. You could, instead, use a cascading style sheet to assign the HTML tags and style the table, but only XSL can filter the data conditionally.

The XML data island

The data island is a set of XML tags wrapped in an overall <XML> element, with an ID to be used by JavaScript (our ID is "thisXML"). In this example, the island is on a subform and all its text is set as PassThru HTML. The fields to hold the XML data are wrapped in the XML start and end tags.



Your XML island can be as long or short as you want. It takes the basic form:

```
<XML ID="yourXML">
<yourdata>some data</yourdata>
</XML>
```

The JavaScript

The JavaScript follows immediately after the data island and is also set as PassThru HTML. It displays the XML on the onLoad event for the document, using XSL, which is stored as a page element in the same database. The reference to the XSL page is relative, that is, "../yourstylesheet.xsl." This is the JavaScript that references the "functiontable.xsl" page and the "thisXML" island. We've placed the resulting HTML inside a <DIV> tag, but you could put it into any similar tag.

```
<XML id="style" src="../functiontable.xsl"></XML>
<SCRIPT FOR="window" EVENT="onLoad">
    xslTarget.innerHTML = thisXML.transformNode(style.XMLDocument);
</SCRIPT>

<DIV id="xslTarget"></DIV>
```

The XSL page

The XSL page can have any name but must end in .xsl. It must also be set in Page Properties to "Treat page contents as HTML."

This XSL is much like the XSL for the table in the article **Exercising XML**

with Domino Designer. However, it also uses xsl:choose statements to test the data and display it conditionally. This xsl:choose statement works in Internet Explorer 5, but may not be legal in XSL served-based transforms.

To see the code for the XSL page, go to the **XSL page sidebar**.

Using an island within a document

To use an island like this example in the midst of a document, simply place the XML island followed by the JavaScript, anywhere within a rich text field and set the tags as PassThru HTML. Enter the actual data for each tag as text instead of using a field between the start and end tags.

Hiding an island from non-XML compatible browsers

When a browser that doesn't handle XML sees this island, it simply streams the text from the unknown tags into the middle of the HTML, ignoring the XSL formatting. This raw HTML isn't something you want to display to users.

Fortunately, you can use the R5 @BrowserInfo function to detect inappropriate browsers and hide the XML. We used this formula for the computed subform (called "island") that holds the data island.

```
@If(@BrowserInfo("BrowserType")="Microsoft" & @BrowserInfo("Version")=5
& TopicType="function";"island";"")
```

In addition to showing the XML only to Microsoft Internet Explorer 5, the formula shows the XML only on documents where the TopicType field contains "function". Otherwise the island would appear in all documents that use the form, regardless of whether those documents had actual data for the fields that the XML uses. In the example, the default value of the fields would be used and everything in the table would display as "yes."

Finally, remember to set all the XML text to hide from the Notes client.

The XSL page

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
        <xsl:apply-templates/>
    </xsl:template>

<xsl:template match="atfunc">
    <H1><xsl:value-of select="atfunc_name"/></H1>
        <xsl:apply-templates />
    </xsl:template>

<!-- BEGIN TABLE -->

<xsl:template match="usecontext">
<!--This line sets the table border to 1 pixel, inset, black-->
<TABLE STYLE="border:1px inset black">

<!--The first row of the table. The font of the text in this row is set 7 point, boldface, Verdana, and the background color of the row to light grey.-->
<TR STYLE="font-size:7pt; background-color: lightgrey; font-family:Verdana; font-weight:bold">

<!--This is the text that will appear spanning the first two cells of the first row of the table. -->
<TD COLSPAN="2">Works in formulas on...?</TD>
<!--The end of the first row in the table.-->
</TR>

<!--The subsequent rows of the table. The font is set to Verdana, 7 point. The cell padding is set to 0 pixels and 0 pixels.-->
<TR STYLE="font-family:Verdana; font-size:7pt; padding:0px 0px">

<xsl:for-each select="use_smicon">
    <TD>SmartIcon:
        <xsl:choose>
        <xsl:when test=".[. = 'no']" ><FONT COLOR="RED"><B><xsl:value-of
        select="."/></B></FONT></xsl:when>
        <xsl:otherwise><FONT COLOR="gray"><B><xsl:value-of select="."/></B></FONT></xsl:otherwise>
        </xsl:choose>
    </TD>
</xsl:for-each>

<xsl:for-each select="use_button">
    <TD>Button:
        <xsl:choose>
        <xsl:when test=".[. = 'no']" ><FONT COLOR="RED"><B><xsl:value-of
        select="."/></B></FONT></xsl:when>
        <xsl:otherwise><FONT COLOR="gray"><B><xsl:value-of select="."/></B></FONT></xsl:otherwise>
        </xsl:choose>
    </TD>
```

```
</xsl:for-each>

<xsl:for-each select="use_sel_form">
    <TD>Selection formula:
        <xsl:choose>
        <xsl:when test=".[. = 'no']" ><FONT COLOR="RED"><B><xsl:value-of
        select="."/></B></FONT></xsl:when>
        <xsl:otherwise><FONT COLOR="gray"><B><xsl:value-of select="."/></B></FONT></xsl:otherwise>
        </xsl:choose>
    </TD>
</xsl:for-each>

<xsl:for-each select="use_col_form">
    <TD>View column:
        <xsl:choose>
        <xsl:when test=".[. = 'no']" ><FONT COLOR="RED"><B><xsl:value-of
        select="."/></B></FONT></xsl:when>
        <xsl:otherwise><FONT COLOR="gray"><B><xsl:value-of select="."/></B></FONT></xsl:otherwise>
        </xsl:choose>
    </TD>
</xsl:for-each>

<xsl:for-each select="use_agent_man">
    <TD>Agent (manual):
        <xsl:choose>
        <xsl:when test=".[. = 'no']" ><FONT COLOR="RED"><B><xsl:value-of
        select="."/></B></FONT></xsl:when>
        <xsl:otherwise><FONT COLOR="gray"><B><xsl:value-of select="."/></B></FONT></xsl:otherwise>
        </xsl:choose>
    </TD>
</xsl:for-each>

<xsl:for-each select="use_agent_mail">
    <TD>Agent (mail):
        <xsl:choose>
        <xsl:when test=".[. = 'no']" ><FONT COLOR="RED"><B><xsl:value-of
        select="."/></B></FONT></xsl:when>
        <xsl:otherwise><FONT COLOR="gray"><B><xsl:value-of select="."/></B></FONT></xsl:otherwise>
        </xsl:choose>
    </TD>
</xsl:for-each>

<!--This line ends this row of the table. -->
</TR>

<TR STYLE="font-family:Verdana; font-size:7pt; padding:0px 3px">

<xsl:for-each select="use_agent_sched">
    <TD>Agent (scheduled):
        <xsl:choose>
        <xsl:when test=".[. = 'no']" ><FONT COLOR="RED"><B><xsl:value-of
        select="."/></B></FONT></xsl:when>
        <xsl:otherwise><FONT COLOR="gray"><B><xsl:value-of select="."/></B></FONT></xsl:otherwise>
        </xsl:choose>
    </TD>
</xsl:for-each>

<xsl:for-each select="use_hidewhen">
    <TD>Hide-when property:
        <xsl:choose>
        <xsl:when test=".[. = 'no']" ><FONT COLOR="RED"><B><xsl:value-of
```



```
            select="."/></B></FONT></xsl:when>
            <xsl:otherwise><FONT COLOR="gray"><B><xsl:value-of select="."/></B></FONT></xsl:otherwise>
            </xsl:choose>
      </TD>
</xsl:for-each>

<xsl:for-each select="use_section">
      <TD>Section:
            <xsl:choose>
            <xsl:when test=".[. = 'no']" ><FONT COLOR="RED"><B><xsl:value-of
            select="."/></B></FONT></xsl:when>
            <xsl:otherwise><FONT COLOR="gray"><B><xsl:value-of select="."/></B></FONT></xsl:otherwise>
            </xsl:choose>
      </TD>
</xsl:for-each>

<xsl:for-each select="use_window">
      <TD>Window title:
            <xsl:choose>
            <xsl:when test=".[. = 'no']" ><FONT COLOR="RED"><B><xsl:value-of
            select="."/></B></FONT></xsl:when>
            <xsl:otherwise><FONT COLOR="gray"><B><xsl:value-of select="."/></B></FONT></xsl:otherwise>
            </xsl:choose>
      </TD>
</xsl:for-each>

<xsl:for-each select="use_hot_act">
      <TD>Action hotspot:
            <xsl:choose>
            <xsl:when test=".[. = 'no']" ><FONT COLOR="RED"><B><xsl:value-of
            select="."/></B></FONT></xsl:when>
            <xsl:otherwise><FONT COLOR="gray"><B><xsl:value-of select="."/></B></FONT></xsl:otherwise>
            </xsl:choose>
      </TD>
</xsl:for-each>

<xsl:for-each select="use_hot_pop">
      <TD>Popup hotspot:
            <xsl:choose>
            <xsl:when test=".[. = 'no']" ><FONT COLOR="RED"><B><xsl:value-of
            select="."/></B></FONT></xsl:when>
            <xsl:otherwise><FONT COLOR="gray"><B><xsl:value-of select="."/></B></FONT></xsl:otherwise>
            </xsl:choose>
      </TD>
</xsl:for-each>

</TR>


<TR STYLE="font-family:Verdana; font-size:7pt; padding:0px 3px">

<xsl:for-each select="use_field">
      <TD>Field:
            <xsl:choose>
            <xsl:when test=".[. = 'no']" ><FONT COLOR="RED"><B><xsl:value-of
            select="."/></B></FONT></xsl:when>
            <xsl:otherwise><FONT COLOR="gray"><B><xsl:value-of select="."/></B></FONT></xsl:otherwise>
            </xsl:choose>
      </TD>
</xsl:for-each>
```



```
<xsl:for-each select="use_form_form">
    <TD>Form:
        <xsl:choose>
        <xsl:when test=".[. = 'no']" ><FONT COLOR="RED"><B><xsl:value-of
        select="."/></B></FONT></xsl:when>
        <xsl:otherwise><FONT COLOR="gray"><B><xsl:value-of select="."/></B></FONT></xsl:otherwise>
        </xsl:choose>
    </TD>
</xsl:for-each>

<xsl:for-each select="use_form_act">
    <TD>Form action:
        <xsl:choose>
        <xsl:when test=".[. = 'no']" ><FONT COLOR="RED"><B><xsl:value-of
        select="."/></B></FONT></xsl:when>
        <xsl:otherwise><FONT COLOR="gray"><B><xsl:value-of select="."/></B></FONT></xsl:otherwise>
        </xsl:choose>
    </TD>
</xsl:for-each>

<xsl:for-each select="use_view_act">
    <TD>View action:
        <xsl:choose>
        <xsl:when test=".[. = 'no']" ><FONT COLOR="RED"><B><xsl:value-of
        select="."/></B></FONT></xsl:when>
        <xsl:otherwise><FONT COLOR="gray"><B><xsl:value-of select="."/></B></FONT></xsl:otherwise>
        </xsl:choose>
    </TD>
</xsl:for-each>

<xsl:for-each select="use_navig">
    <TD>Navigator:
        <xsl:choose>
        <xsl:when test=".[. = 'no']" ><FONT COLOR="RED"><B><xsl:value-of
        select="."/></B></FONT></xsl:when>
        <xsl:otherwise><FONT COLOR="gray"><B><xsl:value-of select="."/></B></FONT></xsl:otherwise>
        </xsl:choose>
    </TD>
</xsl:for-each>

<xsl:for-each select="use_layout">
    <TD>Layout region:
        <xsl:choose>
        <xsl:when test=".[. = 'no']" ><FONT COLOR="RED"><B><xsl:value-of
        select="."/></B></FONT></xsl:when>
        <xsl:otherwise><FONT COLOR="gray"><B><xsl:value-of select="."/></B></FONT></xsl:otherwise>
        </xsl:choose>
    </TD>
</xsl:for-each>

</TR>
</TABLE>
</xsl:template>

<!-- END TABLE -->

</xsl:stylesheet>
```

