

::ISUG

PowerBuilder

techcast
series

- Easy XML with
PowerBuilder
Document Object Model

::ISUG

PowerBuilder

techcast
series

Easy XML with PBDOM



Terry Voth

www.techno-kitten.com

Moderator

TeamSybase Liaison Director

International Sybase User Group

::ISUG

PowerBuilder

techcast
series

Easy XML with PBDOM



John Strano

Presenter

Sybase Technical Evangelist

- **This session provides a technical introduction to PBDOM**
 - This session is not an introduction to XML
 - Terms you should be comfortable with:
 - Document, DTD, schema
 - Element, attribute, processing instruction, entity

- **This presentation will teach you...**
 - Why (and how) PBDOM was created
 - Where PBDOM is most useful
 - How to program with PBDOM
 - How PBDOM compares to other XML parsing technologies

- **Document Object Model**
 - As a W3C specification, the objective for the XML DOM has been to provide a standard programming interface to a wide variety of applications. The XML DOM is designed to be used with any programming language and any operating system.
 - With the XML DOM, a programmer can create an XML document, navigate its structure, and add, modify, or delete its elements.

- **Document Object Model**
 - A program called an *XML parser* can be used to load an XML document into the memory of your computer. When the document is loaded, its information can be retrieved and manipulated by accessing the DOM.
 - *The DOM represents a tree view of the XML document.* The `documentElement` is the top-level of the tree. This element has one or many `childNodes` that represent the branches of the tree.

- ***For more on XML Document Object Model***
 - <http://www.w3schools.com/dom/>

- ***When you need to...***
 - Move elements of the document within the document or to an entirely different document.
 - Create new elements and introduce them at any point within the document
 - Manipulate elements of the document and then re-insert them into the document.
 - Import nested structures.
 - The XML capabilities of the DataWindow that were introduced in PowerBuilder 9 can currently only *export* nested data structures.

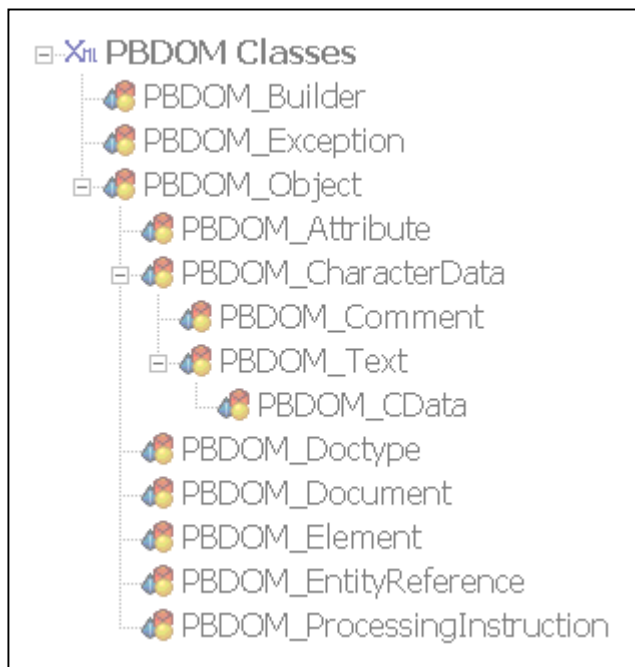
- **PowerBuilder Document Object Model**
 - PBDOM is a PBNI extension for XML data manipulations, optimized for PowerScript
 - A programming model to represent XML data – an abstraction of DOM
 - Uses Apache Xerces/C++ under the hood

- **PBDOM was created to...**
 - Be straightforward for PowerBuilder programmers
 - Use the power of the PowerScript language (objects, method overloading, arrays)
 - Hide the complexities of XML wherever possible
 - DOM is unintuitive to a PowerBuilder programmer (anyone ever used DOM in PowerDynamo?)

- **Setup**
 - Just add pbdom90.pbd to (%SYBASE%\Shared\PowerBuilder) your library list
 - %SYBASE%\Shared\PowerBuilder should be on your PATH or AppPath registry entry (for pbdom90.dll, pbxerces90.dll and xerces_2_1_0.dll)

- **PBDOM Class Hierarchy**

- All PBDOM classes (except PBDOM_Builder and PBDOM_Exception) inherit from PBDOM_Object



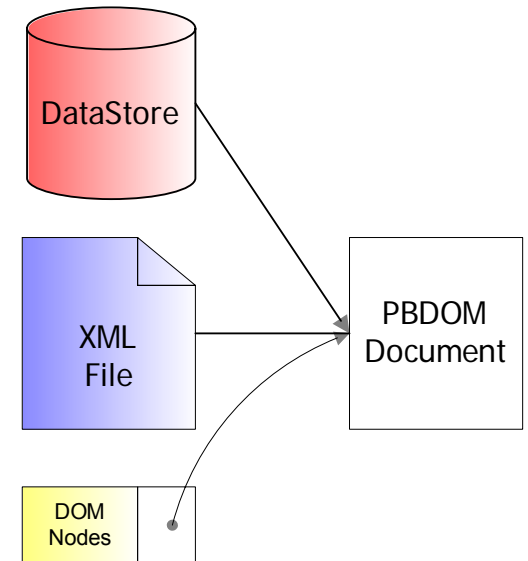
• Construction

- XML documents can be created from scratch

```
PBDOM_Document doc
PBDOM_Element root
doc = CREATE PBDOM_Document
root = CREATE PBDOM_Element
root.SetName( "root" )
root.SetText( "this is the root" )
doc.AddContent( root )
```

- Or built from a file, a string or a datastore

```
PBDOM_Builder builder
doc = builder.BuildFromString( "<foo>bar</foo>" )
doc = builder.BuildFromFile( "c:\foo\bar.xml" )
doc = builder.BuildFromDataStore( I_ds )
```



- **Navigating the element tree**

```
PBDOM_Element root, children[], first
// Get the root element of the document
root = doc.GetRootElement()
// Get an array of all child elements
root.GetChildElements( children )
// Get only elements with a given name
root.GetChildElements( "name", children )
// Get the first element with a given name
first = root.GetChildElement( "name" )
```

- **The element array is live!**

- Changes to the elements affects the parent document
- Once the array is returned, it is now bounded
- Adding new elements to the array requires a `SetContent()` call
☹

- **Moving elements is easy with PBDOM**

```
// PBDOM_Document docOne, docTwo
PBDOM_Element     movabl e

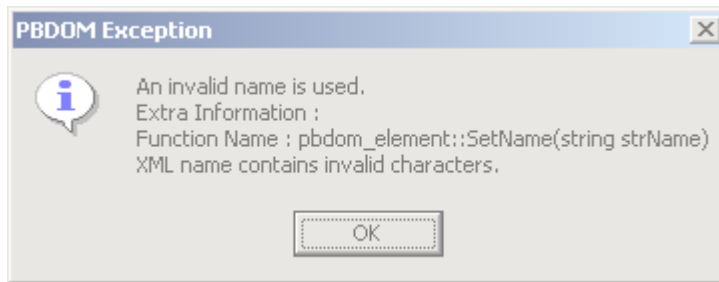
movabl e = CREATE PBDOM_Element
Movabl e.SetName( "movabl e" )
docOne.AddContent( movabl e )           // add
movabl e.Detach()                       // remove
docTwo.addContent( movabl e )          // add again
```

- Detach() may be called for anything that is derived from PBDOM_Object
 - Comments
 - ProcessingInstructions
 - Elements (and their content)
 - Etc...
- PBDOM elements aren't permanently tied to their parent document

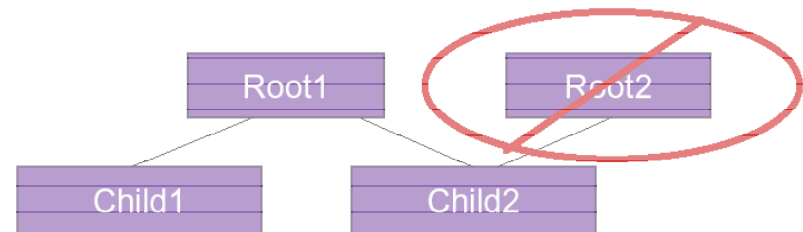
- **Always well-formed**

- The PBDOM_Element constructor and setter methods check the element for well-formedness

```
el em. SetName( "Spaces are illegal" )
```



- The **AddContent()** method also checks for:
 - Structure – no loops in any tree
 - One and only one root element
 - Consistent namespaces



- Elements may have attributes

```
<table width="100%" border="0"></table>
```

```
// Get an attribute
```

```
ls_width = table.GetAttributeValue( "width" ) // or
```

```
ls_width = table.GetAttribute( "width" ).GetText()
```

```
// Attributes can be typed
```

```
li_border = table.GetAttribute( "width" ).GetIntValue()
```

```
// Set an attribute
```

```
table.SetAttribute( "cellspacing", "0" )
```

```
// Remove an attribute
```

```
table.RemoveAttribute( "cellspacing" )
```

```
// Remove all attributes
```

```
PBDOM_Attribute empty[]
```

```
table.SetAttributes( empty ) // the PowerScript way
```

- **Elements may have content**

```
<description>
```

```
    cool demo
```

```
</description>
```

```
// the text is directly available – returns
```

```
// "~r~ncool    demo~r~n"
```

```
Is_desc = elem.GetText()
```

```
// two convenience methods
```

```
Is_desc = elem.GetTextTrim()    // returns "cool    demo"
```

```
Is_desc = elem.GetTextNormalize() // returns "cool demo"
```

```
// text can be changed directly
```

```
elem.SetText( "a new description" )
```

- Elements may have mixed content

```
<description>  
  <!-- comment -->  
  <?convert units="metric" ?>  
  cool demo  
</description>
```

```
PBDOM_Object content[]  
desc.GetContent( content )  
FOR i = 1 TO UpperBound( content )  
  CHOOSE content[i].GetObjectClassString()  
    CASE "pbdom_comment"  
      // ...  
    CASE "pbdom_processinginstruction"  
      // ...  
  END CHOOSE  
NEXT
```

- **Processing instructions look like this**

```
<?  $\left( \begin{array}{c} \text{xml -styl esheet} \\ \text{Target} \end{array} \right) \left( \begin{array}{c} \text{type="text/xsl " href="foo.xml " } \\ \text{Data} \end{array} \right) ?>$ 
```

```
// Get target (e.g., "xsl -styl esheet")
```

```
Is_target = pi.GetTarget()
```

```
// Get data (e.g., 'type="text/xsl " href="foo.xml "')
```

```
Is_data = pi.GetText()
```

```
// Get individual values as attributes
```

```
String names[]
```

```
pi.GetNames( names )
```

```
FOR i = 1 TO UpperBound( names )
```

```
    MessageBox( names[i], pi.GetValue( names[i] ) )
```

```
NEXT
```

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:variable name="ffsection" select="//SITE_SECTION">
  <xsl:template name="TopNav">
    ...
  </xsl:template>
</xsl:stylesheet>
```

```
String      Is_element
PBDOM_Element  template
```

```
// get element name and namespace – return "xsl:template"
template = root.GetChildElement( "template" )
Is_element= template.GetNamespacePrefix() + ":" +
  template.GetName()
```

```
// get element by name and namespace
template = root.GetChildElement( "template", "xsl", &
  "http://www.w3.org/1999/XSL/Transform"
```

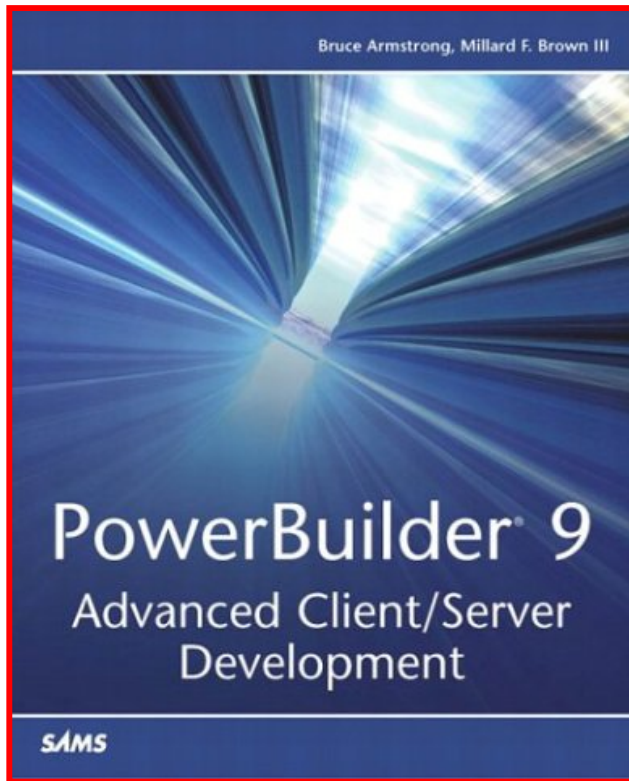
- **Demo – Code Example**

- **Apache Xerces/COM**
 - Xerces is the [current] underlying XML parser for PBDOM, but is less intuitive

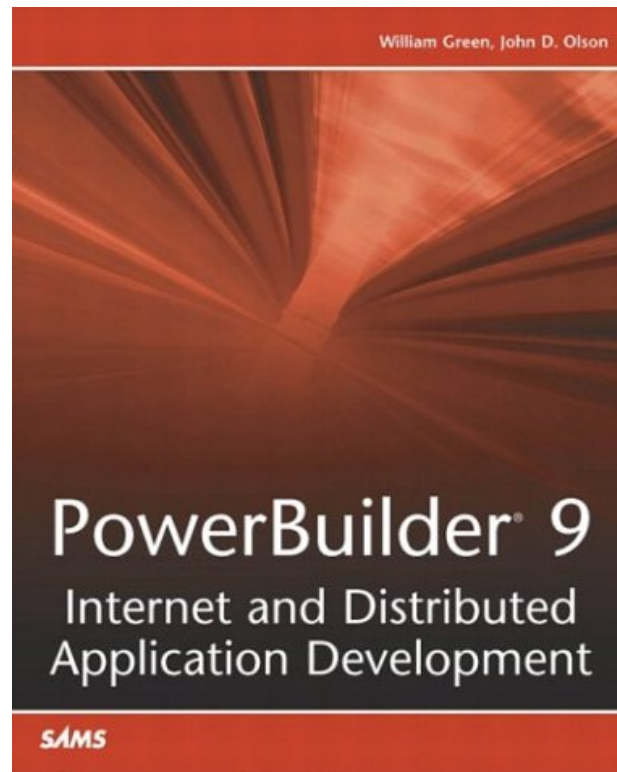
- **MSXML**
 - No deployment to UNIX possible
 - Less Intuitive
 - COM collections vs. PB arrays (if you're a casual COM user)

- **Others**
 - Expat – a C DLL (requires external function declarations...unwieldy)

Advanced Client/Server



Internet & Distributed Development



- **Forum for exchanging samples, tools, scripts, etc.**
 - Download samples created by Sybase or by external users
 - Leverage contributions of others to extend Sybase products

- **Features enable community collaboration**
 - Contribute code or start your own collaborative / open source project with input from other product experts

- **Any SDN member can participate**
 - Log in using your MySybase account via SDN
 - <http://codexchange.sybase.com>
 - Or via SDN at www.sybase.com/developer

Q & A