NAME

FanXE − Fantastic XML Editor.

SYNOPSIS

fanxe *[CLASS OPTIONS --]* **method** *[METHOD OPTIONS]*

package require FanXE
FanXE *objName [CLASS OPTIONS]*
objName **method** *[METHOD OPTIONS]*

DESCRIPTION

FanXE(n) is an XML editor. XML files are loaded into the Document Object Model (DOM), an in-memory representation of the XML file. Multiple DOMs can be edited simultaneously.

FanXE(n) must first be trained in the layout of the XML files. The fanxe.xml file dictates which XML files will be loaded, which branches will be displayed, and which style sheets will be applied when browsing the nodes.

CLASS RELATIONSHIPS

FanXE(n) can include **Pgsql**(n) objects if the database is specified in the fanxe.xml configuration file. Use the **pgid** method to access the **Pgsql**(n) object(s).

FanXE(n) includes an **RCS**(n) object (stored in the protected variable *_rcs*) for access files under revision control.

GUI INTERFACE

Menu/Status Bar

Across the top of the GUI are three buttons and a status bar. The File and Edit buttons on the left will open menus. The status bar displays the most recent status message. The Exit button on the right exit (if unsaved changes exist, you will have an option to cancel). The File menu will allow you to reload or save the various DOMs that FanXE is editing. If the XML files are stored in **RCS**(n), the revision control will be automatic. The *-rev* option will specify the name of the revision to check out. When files are checked in, the new revision will be given the *-rev* name.

The "Fork Revision" entry will allow you to mark all revisions with the current *-rev* name into a new name. This will allow you to maintain, for example, a Production and Development version of the XML files. All style sheets stored in **RCS**(n) will also be tagged and **FanXE**(n) will always read the *-rev* revision unless *-useRCS* is set to 0.

The "Reload Style Sheets" entry will reload all the XSLT style sheets. The "Exit" entry is the same as the "Exit" button.

Search Bar

The **Search Bar** will run a regular expression search over all DOMs. Selecting the **Trim** option will close all branches in the tree pane except for those that contain matches. Otherwise, matching nodes will be opened but currently opened branches will remain in view.

Tree (XPath) Bar

The **Tree (XPath) Bar** works like the Search Bar except that you must select which DOM to search and the search expression must be a valid XPath expression. The **Trim** option works the same as in the Search Bar.

Tree Pane

The **Tree Pane** presents multiple DOMs in a tree. Not all branches of the DOM are shown in the tree, just those configured in the fanxe.xml configuration file.

View Pane

Selecting any node in the **Tree Pane** will run a style sheet associated with that node and display the HTML results in the **View Pane**. Clicking on the Help ("?") icon will display help information (the URL for the help file is configured in the fanxe.xml file). The **View Pane** is a nearly full-featured HTML browser. It does not support HTML forms except for editing the nodes. It also does not support Cascading Style Sheets (CSS). It does support special fanxe:// URIs (links to other nodes) as well as http://FanXE/method?opts links (which will run the **FanXE**(n) method) and various built-in images. The

left and right arrows will scroll through the browser history.

Tabbed Pane

The **Tabbed Pane** contains the log messages and tabbed HTML windows. Middle clicking on a node in the **Tree Pane** or middle-clicking on any URL will open the link in a new HTML tab instead of using the **View Pane**.

Stylesheets associated with nodes may contain HTML form elements for editing the node. More commonly, the style sheet will present a summary of the node and include an "Edit" button that will run a second style sheet with form elements for editing the node. Clicking the "Edit" button will run this second style sheet, but will always display the results in a new HTML tab. If the node is already being edited, the previous tab will be opened.

Editing windows will commonly have "Save", "Refresh", "XML" and "Cancel" buttons. Note that "Save" only saves the changes in memory, not on the XML file. You will be warned before exiting if you have in-memory changes that have not been saved to a file. The "Refresh" button reruns the style sheet. The "XML" button will display the current node as pretty-print XML in a new tab. "Cancel" will abort editing.

Clicking on the "Pin" icon or right-clicking on the tab will detach that tab and open it in a new window allowing you view multiple tabs simultaneously.

CLASS OPTIONS

-debug	Debugging level (default **2**).
-loadxml	Load XML files at startup (default **0**).
-xmlDir	Location of FanXE XML files.
-tmpDir	Location of Temp files.
-browser	External web browser.
-proxy	Web proxy.
-baseFontSize	Base font size for Tkhtml browser.
-helpURL	Location of default help file (in HTML format).
-rev	Default revision name to check in/out (default **Production**).
-useRCS	Use RCS(n) for files stored under revision control (default **1**).
-usePG	Use PostgreSQL for dom(n)s stored in PostgreSQL (default **1**).
-publicIDs	Map PUBLIC ids to local files.
-logupdate	Update log window (default **1**).
-geometry	Geometry of toplevel window.
-tearoffGeometry	Geometry of (detached) edit windows.
-layoutFlip	Should the outer pane be swapped with the inner frame.
-layoutP1	Which widget (tree, view or edit) goes into pane1.
-layoutP2	Which widget (tree, view or edit) goes into pane2.
-layoutP3	Which widget (tree, view or edit) goes into pane3.
-layoutType	How should panes be oriented (1=p1 and p2 are vertical; 2=p1 is vertical and p2 is horizontal; 3=p1 is horizontal and p2 is vertical; 4=p1 and p2 are horizontal).
-layoutFraction	What portion should each pane use (two numbers: first is fraction for outer pane, second is inner pane).

PUBLIC METHODS

Command Line Methods

version

Return the version number. The RCS version number is returned.

install

First time setup of fanxe.xml. fanxe.xml will be loaded, the @Path attribute will be set, and the file will be saved again. **FanXE**(n) should be shipped without an @Path in the default fanxe.xml file.

transform *xmlFile xslFile -parameters -indent -html -xml -dom -document -gui -print -ps*

Run an XSLT transformation. The *xmlFile* and *xslFile* will be parsed. If *xmlFile* begins with an "at" sign (@), it will use that **domDoc**(n) instead of loading from an external file. The *xslFile* will be applied against the *xmlFile*.

If *-document* is specified, all <xsl:document/> elements will be written to separate files. The filename is specified in the @href attribute, and the serialization method is specified in the @method attribute. The <xsl:document/> element is then deleted from the tree. The remaining tree is handled normally.

If *-html* is specified, the results will be serialized using HTML rules (HTML elements are recognized regardless of case, without end tags for emtpy HTML elements etc.). If *-xml* is specified, the results will be serialized as XML. If *-gui* is specified, *-html* will be set and the results will be displayed in the GUI. If neither *-html*, *-xml* or *-gui* are specified, the defaultOutputMethod will be used. Nothing will be returned if *-gui* is specified. If *-dom* is specified, the resulting **domDoc**(n) will be returned. Otherwise the serialized resulting tree will be returned.

METHOD OPTIONS

xmlFile	Name of the XML file.
xslFile	Name of the XSL file.
-parameters	Parameters to pass to the xslt method (optional).
-indent	Output indentation (optional, default is 4).
-html	Return results as HTML (optional).
-xml	Return results as XML (optional).
-dom	Return the result DOM (optional).
-document	Support the xsl:document extension.
-gui	Return results to the GUI with specified title (optional).
-print	Convert HTML results to PostScript and send to the default printer (optional).
-ps	Convert HTML results to PostScript and save in file (optional).

xmlverbatim *-domDoc -domNode -xml -file -return -htmlWidget*

Call the xmlverbatim command. You must specify either a **domDoc**(n), **domNode**(n), XML text or an XML file to transform. If multiple of the above are specified, the first (in order listed here) will be used.

The XML will be transformed into HTML with syntax highlighting using Oliver Becker's xmlverbatim.xsl style sheet. If -return is true (or the GUI is not available), the HTML will be returned. Otherwise, the HTML will be displayed in the *htmlWidget* and nothing will be returned.

METHOD OPTIONS

-domDoc	A **domDoc**(n) to transform.
-domNode	A **domNode**(n) to transform.
-xml	Raw XML text to transform.
-file	An XML file to transform.

-return Return results as HTML (rather than displaying).

-htmlWidget Name (hierarchy) of the htmlWidget to display results in (default is a new window).

asXML *domName*

Return the **dom**(n) as XML text. If the *domName* is not already loaded, it will be loaded. The *domName* will be returned as an XML string.

METHOD OPTIONS

domName Name of the **dom**(n) (/xmlFile/File@Name in fanxe.xml).

compare *domName key oldRev newRev xpath*

Compare any two revisions of the *domName* file. The *xpath* of each RCS revision (*oldRev* and *newRev*) of *domName* will be compared. The *xpath* must contain the *key* attribute. XML text will be returned with each original node (marked as <Change Type="Replace"> if replaced or <Change Type="Delete">) and each new node (marked as <Change Type="With"> if replaced or <Change Type="Add">).

METHOD OPTIONS

domName Name of the **dom**(n) (/xmlFile/File@Name in fanxe.xml).

key Attribute to uniquely identify a node.

oldRev Old revision (usually symbolic name).

newRev New revision (usually symbolic name).

xpath XPath to top element to compare.

search *domName xpath pattern xslFile*

Search elements for a string. The *domName* will be searched and each *xpath* node that includes *pattern* will be added to a temporary **dom**(n) with a root element of **<Search>**. The *xslFile* xsl style sheet will be applied to the **<Search> dom**(n) and the results will be returned as HTML.

METHOD OPTIONS

domName Name of the **dom**(n) (/xmlFile/File@Name in fanxe.xml).

xpath XPath to top element to search.

pattern Pattern (regular expression) to search for.

xslFile Style sheet used to transform results into HTML.

fork *oldRev newRev*

Fork all RCS files from one symbolic name to another. All paths specified in the fanxe.xml configuration file will be searched, and all subdirectories (one level only) will be searched for RCS files. All RCS files with a symbolic name *oldRev* will be given the symbolic name *newRev*. Both *oldRev* and *newRev* will therefore reference the same revision at the time of the fork.

If *oldRev* and newRev are not specified, *oldRev* will be set to the current -rev and the user will be prompted for *newRev*. If *oldRev* is specified but *newRev* is null, the symbolic name *oldRev* will be removed from all RCS files.

Nothing is returned.

METHOD OPTIONS

oldRev Old revision name (optional).

newRev New revision name (optional).

GUI Methods

These methods all make use of the GUI interface.

GUI

Open the GUI editor. The Graphical User Interface (GUI) is created. All XML/XSL files are loaded. Nothing is returned.

ShowImages

Display built-in images. All built-in images (including images imported from imageLib) will be displayed in the *_html*(view) window.

ShowURL *url -trim -browse -external -htmlWidget -nostack*

Show any URL. If no URL is set, the URL and trim will be read from the GUI.

For fanxe:// URLs: (includes the domName and a full XPath expression)

All nodes whose ancestors match the XPath are selected in the Tree widget.

If *-trim* is set, other branches of the Tree widget for the domName will be collapsed. Otherwise, these nodes will be opened in the Tree and selected while other nodes already open will still be displayed.

If *-browse* is 0, no nodes will be displayed in the *_html*() browser windows. If *-browse* is 1, the first match will be displayed in the *_html*(*-htmlWidget*) window. If *-browse* is 2, the first 10 matches will be displayed in new *_html*() browser windows.

-external has no effect.

The Tree nodes that match the XPath expression will be returned.

For non-fanxe:// URLs:

If the URL passed is an http anchor with no path, the *_html*(*-htmlWidget*) widget will move to display that anchor.

If *-external* is not set and the URL scheme is supported, the URL will be fetched and displayed (if supported MIME type).

If the URL scheme or MIME type are not supported or *-external* was set, the URL will be passed to the external browser.

-browse and *-trim* have no effect.

Nothing will be returned.

For all URLs: If *-nostack* is set, the stack will not be modified. Otherwise, the url will be added to the *_viewStack*.

METHOD OPTIONS

url	URL to show. Optional, default is URL entered in the GUI.
-trim	Whether to collapse the tree and show only matching nodes or to simply select nodes.
-browse	How should the fanxe:// URLs be displayed in a browser (default=0).
-external	Should the external browser be used for non-fanxe URLs.
-htmlWidget	Name (hierarchy) of the htmlWidget to display URL in (default=view).
-nostack	Do not modify the stack.

Other Public Methods

readRCS *-cache -rev filename*

Read a file from RCS. If *-useRCS* is false the *filename* will simply be read. Otherwise the *-rev* will be read directly from **RCS**(n). If the -rev revision does not exist, a warning will be issued and the flat file will be read instead. The data from the file will be returned.

METHOD OPTIONS

-cache Use results from a cache to avoid rereading from RCS (optional).

-rev Revision to read, usually a symbolic name. Default is to use the class-level *-rev* variable (optional).

filename Name of file to read.

externalEntityRef *base systemId publicId*

Handle external entity references by the parser. Returns a list with three elements. The first element is the literal "string". The second element is the URI of the file that is parsed. The third element is the data that was read. If the URI is a file, the data is read via the readRCS method (with the -cache option). If the URI is an http, the geturl package is used (results are also cached). No other schemes are currently supported and an error will be generated.

METHOD OPTIONS

base Base URI of the entity.

systemId The system identifier of the entity.

publicId The public identifier of the entity.

xsltMessage *msg terminate*

Handle <xsl:message> elements. If terminate is true, *msg* will be displayed as a warning. Otherwise, *msg* will be displayed as info. Nothing is returned.

METHOD OPTIONS

msg Message

terminate Terminate option

pgid *domName request*

Open PostgreSQL database for domName and return Pgsql object, channel, database, or any combination of these. If *domName* stores the XML in a PostgreSQL database (/xmlFile/File@PGSQL != ''), open the database (if not already opened). Valid requests are "Pgsql" (name of the Pgsql object), "pgid" (channel ID to access database directly), "dbname" (name of the database). Multiple domNames that use the same dbname will also use the same Pgsql object and pgid channel. The Pgsql object will be stored in *_pg*(*dbname*) and the pgid will be stored in *_pg*(*dbname*,id).

If *domName* does NOT store XML in PostgreSQL or *usePG* is false, nothing will be opened or returned.

METHOD OPTIONS

domName Name of the **dom**(n) (/xmlFile/File@Name in fanxe.xml).

request List of what should be returned (optional, default "Pgsql").

loadXML *-rev domName(s)*

Load an XML file. All xmlFiles passed will be read and stored in the _tdom(*xmlFile*,doc) and _tdom(*xmlFile*,root) array. If no xmlFiles are specified, all files are loaded. The "fanxe" xmlFile must be loaded before any others (if no xmlFiles are specified, this happens automatically). If the GUI has been created (the _top variable exists), the Tree widget will be updated as each xmlFile is loaded. If errors are detected parsing the xmlFiles, an error will be propogated, but only after all xmlFiles have been processed. If the **dom**(n) was loaded, 0 will be returned. Otherwise 1 will be returned.

METHOD OPTIONS

-rev Revision (usually symbolic name) to read (optional).

domName(s) List of names of **dom**(n)s to be loaded (optional).

saveXML *-rev -indent xmlFile*

Save an XML file. The xmlFile is saved in the location specified in the fanxe.xml configuration. If the configuration specifies a @Save style sheet, the xmlFile will be transformed before saving. The file will use the indentation specified by the -indent argument. Nothing will be returned.

METHOD OPTIONS

-rev	Symbolic name of revision to check in (optional, default is current -rev).
-indent	XML indentation (optional, default is 2).
xmlFile	**dom**(n) name of the XML file to save.

loadXSL *domName xmlverbatim*

Load all XSL Style Sheets. If *domName* is specified, all style sheets for that **dom**(n) will be loaded, otherwise style sheets for all **dom**(n)s will be loaded.

Style sheets referenced in the fanxe.xml or myPreferences.xml configuration will be parsed (via readRCS) and their **dom**(n) handles will be stored in the _xsl array. If Errors parsing style sheets will propogate, but only after all style sheets have been read. If xmlverbatim is true, only the xmlverbatim style sheet will be loaded. Nothing is returned.

METHOD OPTIONS

domName	Name of the **dom**(n) (optional, default is null).
xmlverbatim	Only load the xmlverbatim style sheet (optional, default false).

xsl2cmd *filename*

Load an XSL Style Sheet and convert to an XML command. The *filename* will be loaded and converted to a command. The command name will be returned.

METHOD OPTIONS

filename	Name of the XSL file.

domNodeList *xpath*

Find all DOMs in myPreferences.xml and fanxe.xml. Every //File*XPath* node will be returned from either the myPreferences or fanxe **dom**(n)s.

METHOD OPTIONS

xpath	Optional continuation of the XPath.

domIndex *domName*

Find the domName in the fanxe or myPreferences DOMs. The domName will be searched for first in myPreferences (if loaded) followed by fanxe. If a matching node is found, the **domNode**(n) will be returned.

METHOD OPTIONS

domName	Name of the **dom**(n) (/xmlFile/File@Name in fanxe.xml).

domInfo *domName obj*

Return info from the _tdom array. The **dom**(n) handle of the document or root node of the *domName* will be returned.

METHOD OPTIONS

domName	Name of the **dom**(n) (/xmlFile/File@Name in fanxe.xml).
obj	Object you want returned (doc or root, optional, default is root).

domAttr *domName attr*

Return attributes from fanxe.xml configuration for *domName*. The *@attr* attribute from /xmlFile/File[@Name=*domName*] will be returned.

METHOD OPTIONS

domName	Name of the **dom**(n) (/xmlFile/File@Name in fanxe.xml).
attr	Attribute to return (optional, default Path).

domClone *domName filename new*

Create a new **dom**(n) from an existing **dom**(n). The existing structure of *domName* will be cloned in the fanxe **dom**(n). If no *filename* was specified, the user will be prompted for a file to create (if *new*) or a file to load (if not *new*). If *new* is true, the newly created *filename* will be copied from the *domName*.

The *filename* will be loaded with the new structure. The new **dom**(n) will share the same *_copy*() buffer. The new *domName* and **domNode**(n) will be returned.

METHOD OPTIONS

domName	Name of the **dom**(n) (/xmlFile/File@Name in fanxe.xml).
filename	Filename (and full path) of the new **dom**(n) (optional, user will be prompted if null).
new	Boolean to determine if we are creating a new file or loading an existing file (optional, default is false).

domRemove *domName treeOnly*

Remove a **dom**(n) from the fanxe or myPreferences configuration and remove from the tree. The *domName* will be removed from memory and the tree widget. Unless *treeOnly* is true, the node will also be removed from its configuration file. Nothing will be returned.

METHOD OPTIONS

domName	Name of the **dom**(n) (/xmlFile/File@Name in fanxe.xml).
treeOnly	Remove the **dom**(n) only from the tixTree (boolean, default false).

exit

Quit the GUI. If no **dom**(n)s currently have unsaved changes, *this* is destroyed. Otherwise, the user is given a chance to abort before *this* is destroyed.

preview *title txt*

Display text in a preview window. The preview window will be opened and *txt* displayed.

METHOD OPTIONS

title	Title of the window.
txt	Text to display.

previewSave *w*

Save text from a preview window. The user will be prompted for a filename and the text from the preview widget *w* will be saved. Nothing will be returned.

METHOD OPTIONS

w	Preview widget.

print *html -printer -outfile -landscape*

Print HTML. The HTML will be converted to PostScript and sent to the default printer.

METHOD OPTIONS

html	HTML to print.

-printer Printer to send results to (optional, default $*PRINTER*).

-outfile File to send PostScript to instead of printer (optional).

-landscape Generate landscape output.

QuickSearch *pattern trim*

Search all **dom**(n)s and show matches in the tree. The *pattern* regular expression will be searched for in all nodes of the Tree widget. Any node that matches will be selected. If *trim* is true, all other branches will be collapsed. If the pattern was found in any node, 1 is returned, otherwise 0 is returned.

METHOD OPTIONS

pattern Regular expression to search.

trim Collapse branches not found (optional boolean, default false).

findTreeNodes *url*

Find matching tree nodes for a fanxe:// URL. The *url* must be a fanxe:// URL. All nodes in the Tree widget that match will be returned.

METHOD OPTIONS

url URL to search.

pickFont *size attrs*

Font selector for htmlwidget. The HTML font names will be converted to a Tcl font family for the Tkhtml widget. The name of the Tcl font family will be returned.

METHOD OPTIONS

size Size of the requested font.

attrs Attributes of the requested font.

getFont *family size slant weight*

Font converter from htmlwidget. The htmlwidget uses pickFont to select family, size, slant and weight. This method will convert those into a custom font. If the font does not yet exist, getFont will create it. The name of the matching font is returned.

METHOD OPTIONS

family Font family.

size Font size.

slant Font slant.

weight Font weight.

PROTECTED VARIABLES

NOTE: Protected variables are only accessible by sub-classes of FanXE.

version Version (set by RCS).

_fontSizes Array of font sizes calculated from $-baseFontSize.

_rcs RCS(n) object.

_rcsCache RCS(n) cache.

_pg Array of Pgsql(n) objects.

_safeInterp Safe interpreter .

DOM Handles

_tdom dom(n) documents and root nodes.

_xsl Style Sheets.

_xslFilename Style Sheet filenames.

Tix Tree

_treeXPath Array mapping of XPath to Tix Tree path.

_treeNodes Array mapping a dom(n) node to Tix Tree path.

_copy Cut & Paste buffer.

_domChanged Track changes of dom(n)s.

GUI Components

_top Top level GUI window.

_buttons Buttons frame.

_pane Outer pane widget.

_subpane Innter pane widget.

_tree tixTree(n) widget.

_hlist HList(n) widget.

_hlistStatus Status of HList(n) (default **0**).

_tabset BLT(n) tabset(n) widget.

_txt Text widget.

_menu Menu widget.

_status Status widget.

_info HTML info widget.

_xpath XPath search widget.

_search Regular expression search widget.

_styles Tix display styles (DItem(n)).

_preview Preview widget counter (default **0**).

HTML Widget Array

_html Index of HTML windows.

_viewStack Stack of URLs in the _html(view) window.

_htmlContents Map of HTML widgets to dom(n) nodes/style sheets.

_formButtons Radio/check button variables.

_formNames Form widget names (used to reconstruct XML when processing a form).

_tkhtmlPriv Widget selection.

_images HTML images.

_imgCount Image counter (default **0**).

_font HTML fonts.

PROTECTED METHODS

NOTE: Protected methods are only accessible by sub-classes of FanXE.

saveNode *command domName domNode newNode*

Execute the saveNode script (if any) for a given **domNode**(n). If a saveNode script exists for **domNode**(n), it will be executed.

METHOD OPTIONS

command Command to send to script (replace, append or delete).

domName Name of the **dom**(n) (/xmlFile/File@Name in fanxe.xml).

domNode Node to find saveNode script for.

newNode If replacing, the name of the new node (optional).

deleteNull *domNode*

Delete empty text nodes. All childNodes of *domNode* that are TEXT_NODEs and contain only whitespace will be deleted from the **dom**(n). Nothing is returned.

METHOD OPTIONS

domNode Root node.

busy *state*

Set or clear busy state for GUI. If the GUI exists (the _top variable is set), the blt::busy method is called on the *_top* widget and set to *state*. Nothing is returned.

METHOD OPTIONS

state State to set the busy mode (hold or release). Default is "hold".

unsavedChanges *domName*

Handle unsaved changes in an xml file. This method is called whenever unsaved changes exist in a **dom**(n). The user is offered a chance to abort the action. If the user chooses to abort, 1 is returned, otherwise 0 is returned.

METHOD OPTIONS

domName Name of the **dom**(n) (/xmlFile/File@Name in fanxe.xml).

status *level text*

Display log messages. This method is called by the log package to send all messages. If no GUI is present, messages will be sent to stderr. If the GUI is present, the *msg* will be displayed int the *_status* window and written to the *_txt* window (with appropriate colors defined by the log package). Nothing will be returned.

METHOD OPTIONS

level Message level.

text Text of the log message.

Menu *m*

Create the menu (context sensitive). The *_menu*(*m*) will be built with all commands currently available.

METHOD OPTIONS

m Name of the menu.

progress *htmlWidget token total current*

Feedback command to show progress (e.g. loading a URL). The percent of the file that has been loaded will be displayed in either the *_info*(*htmlWidget*) window or sent to stderr.

METHOD OPTIONS

htmlWidget Name (hierarchy) of the htmlWidget.

token The token from ::http::geturl.

total Total expected size of download.

current Current size of download.

preferences *override*

Initialize preferences. Sets default Tk resources and options from FanXE.xml and ~/.tk/fanxe/[info class].xml. Nothing is returned.

METHOD OPTIONS

override Should preferences override command line options (optional boolean, default false).

displayStyles

Initialize or reset DisplayStyles. All *_styles* will be created or updated from the **<DisplayStyles>** elements in the fanxe and myPreferences **dom**(n)s.

TixTree Methods

loadTree *domName refresh*

Load the GUI Tree. The Tree widget is updated with a view of the *domName* **dom**(n). If *refresh* is false, the root is deleted and the tree is rebuilt. Otherwise, the **dom**(n) is parsed and only nodes that changed are updated.

METHOD OPTIONS

domName Name of the **dom**(n) (/xmlFile/File@Name in fanxe.xml).

refresh Boolean to determine if tree should be loaded from scratch or merely updated. Default is false (reload from scratch).

fillTree *refresh domName domNode*

Recursively populate the Tix Tree. The *domNode* in *domName* and all children will be updated in the Tix tree. If *refresh* is false, all children will be deleted first when *domNode* is the root node of *domName*. If *refresh* is true, each node will be inspected to determine if it is already in the Tix tree and it will only be updated if it already exists (thus, the state (opened or closed) of each branch will not be altered.

This method is initially called by loadTree and then recursively called by itself. It can also be called by methods that update domName to keep the tree view current. Nothing is returned.

METHOD OPTIONS

refresh Boolean to determine if tree is being loaded or updated (see loadTree method).

domName Name of the **dom**(n) (/xmlFile/File@Name in fanxe.xml).

domNode Node to begin filling tree.

node2branch *domName domNode*

Find a Branch Node in fanxe.xml for a given **dom**(n) node. The **//Branch** node in fanxe.xml for *domNode* will be returned.

METHOD OPTIONS

domName Name of the **dom**(n) (/xmlFile/File@Name in fanxe.xml).

domNode Node to lookup.

node2treePath *domName domNode*

Determine the treePath for a given **dom**(n) node. The treePath for *domName*/*domNode* is returned. A treePath will be returned even if the domNode is not currently in the tree, since this method is used to determine what the path should be when adding nodes.

METHOD OPTIONS

domName Name of the **dom**(n) (/xmlFile/File@Name in fanxe.xml).

domNode Node to lookup.

closeTree *treeNode*

Recursively close all children on the Tix Tree. All children below *treeNode* in the Tix Tree will be closed. Nothing will be returned.

METHOD OPTIONS

treeNode Tree node to begin closing.

copy *cut test*

Copy (and optionally cut) selected elements into the *_copy*() buffer. The currently selected nodes in the Tree widget are first tested if they can be copied (must all be from the same **dom**(n) and must all be at the same level). If the currently selected nodes can't be copied, 1 will be returned. If *test* is true and the nodes CAN be copied, 0 will be returned (but data will NOT be copied into the copy buffer). If *test* is false (and nodes can be copied), the nodes will be copied into the copy buffer. If *cut* is true, the nodes will be deleted from the **dom**(n) and Tree widget and 0 will be returned.

METHOD OPTIONS

cut Boolean to determine if elements should be deleted from the Tree & **dom**(n). Optional, default is false.

test Boolean to determine if command can be executed based on the current selection. Optional, default is true.

paste *test*

Paste the *_copy*() buffer after *curSel*. If the current selection is invalid for pasting (too few/many selections, no copy buffer or mismatched depth), 1 will be returned. If *test* is true and the current selection IS valid for pasting, 0 will be returned (but data will NOT be pasted). If *test* is false (and selection is valid), the copy buffer will be copied into the **dom**(n), the Tree widget will be updated and 0 will be returned.

METHOD OPTIONS

test Boolean to determine if command can be executed based on the current selection. Optional, default is true.

HTML Viewer Methods

itemView *htmlWidget treeNode -xslt -style -nostack*

View an element from the **dom**(n). If -xslt is specified, then *_xsl*(*-xslt*) will be used to transform *node* into HTML. Otherwise, the style sheet to use will be determined based on the -style argument (based on the @-style attribute in the current branch). If -nostack flag is set, the _viewStack will not be modified.

The resulting HTML will be displayed in the *_html*(*htmlWidget*) widget. If htmlWidget is empty, a new browser will be added with an automatically generated name. If the item is sucessfully displayed, the htmlWidget will be returned, otherwise nothing will be returned.

METHOD OPTIONS

htmlWidget Widget to display results into.

treeNode Tree node to display.

-xslt Name of the style sheet used to transform *node* into HTML (optional, default derived from fanxe.xml and -style).

-style Attribute from fanxe.xml to determine the name of the style sheet to use (optional, default "Browse").

-nostack Do not modify the stack.

viewStack *direction htmlWidget*

Move forward or backwards the *_viewStack*. The *_html*(view) window will redisplay the previous (-1) or next (+1) node on the *_viewStack*. If that node is no longer available, it will be skipped and removed from the stack. Nothing will be returned.

METHOD OPTIONS

direction	Direction to move in the stack (-1 or +1).
htmlWidget	Name (hierarchy) of the htmlWidget (optional, default view).

checkTabs

Check that all nodes in the _tabset still have live nodes. All tabs in the _tabset will be checked to verify that any nodes displayed are still available in the **dom**(n).

addBrowser *htmlWidget title close*

Add a htmlWidget to the tabset. If *htmlWidget* already exists, it will be raised. If not, it will be created and added to the *_tabset*. If *button* is true, a row of buttons will be added above the tkhtml widget. The value htmlWidget (which may be auto generated) will be returned.

METHOD OPTIONS

htmlWidget	Name (hierarchy) of the htmlWidget (optional, default will be auto generated).
title	Title/description of the htmlWidget (optional, default will be auto generated).
close	Boolean: should the widget include a close button (optional, default true)?

detachBrowser *index*

Detach a browser from the tabset. The *index* of tabset will be detached and resized.

METHOD OPTIONS

index	Tabset index.

raiseBrowser *htmlWidget*

Raise a htmlWidget in the tabset. If *htmlWidget* exists, it will be raised. If the htmlWidget does not exist, 1 will be returned (otherwise 0).

METHOD OPTIONS

htmlWidget	Name (hierarchy) of the htmlWidget.

deleteBrowser *htmlWidget*

Delete an htmlWidget from the tabset. If *htmlWidget* is "view", do nothing. Otherwise, destroy the widget and delete it from the *_tabset*. Nothing is returned.

METHOD OPTIONS

htmlWidget	Name (hierarchy) of the htmlWidget.

HTML Form Editor Methods

submit *name value w*

Form submit button. Called whenever an input element of type submit or image are pressed. *w* is the name of the button pressed. The *treeNode* (which includes both the *domName* and *domNode*) is determined by referencing the _htmlContents array and the *htmlWidget* is the third generation parent of *w*.

The behavior of the button depends on the *name* and the *value*. If the *name* is:

edit	If the *domNode* still exists, it will be opened in a new htmlWidget and displayed using the @Edit style sheet. If *domNode* does not exist, nothing will happen.
cancel	The window will be closed.
preview	If the *domNode* still exists, the form will be processed (see **form2xml**) and the XML results will be displayed in a preview window. If the *domNode* does not exist, the window will be closed.

refresh_save
The form first be refreshed (see Refresh below) and then saved (see Save below).

save If the *domNode* still exists, the form will be processed (see **form2xml**). saveNode will be executed with a "replace" command and the new node. If saveNode returns an error, the changes will be aborted. If saveNode does not return an error, all children and attributes of *domNode* will be deleted and replaced with the children and attributes of the new node. If *domNode* does not exist, the window will be closed.

If *name* is none of the above and the first word of *value* is Add, Delete, Copy, Duplicate, Insert or Refresh (and *domNode* still exists), **form2xml** will be called with the first word of *value* and *name* as the action. The window will then rerun the style sheet with the results of **form2xml**. If the *domNode* does not exist, the window will be closed.

If the *name* and *value* are not recognized, a warning will be displayed.

Nothing is returned.

METHOD OPTIONS

name	Name of the submit button.
value	Value of the submit button.
w	Name of the widget associated with this button.

form2xml *htmlWidget action*

Convert an HTML form to XML. This method converts the values in the HTML form elements to XML and, optionally, allows elements to be added, deleted, copied (aka duplicated) or inserted.

Each form widget, the children of "*_html*(*htmlWidget*).x", may contain a portion of an XML element. The *name* of each element (i.e. <input *name*=>) is stored in the _formNames array.

A *name* is made up of 1 or more **tag:ID** pairs separated by a forward slash ("/") followed optionally by an at sign ("@") and an attribute name. The full path of the node must be included in the *name*.

A **tag:ID** pair includes the element *name* (**tag**) and a unique **ID** associated with that element (usually created by the XSLT generate-id() function or the fanxe:path() function). The **ID** can be any string as long as it does not include a colon, slash or at sign (":", "/" or "@"). Also, if the **ID** begins with "COPY", it will be treated specially (see below).

For example, the following XML:

<**Root**>
 <**Branch**>
  <**Leaf** *Attribute*=""/>
 </**Branch**>
 <**Branch**/>
</**Root**>

contains four elements. It could be represented by four HTML <**form**> elements with the following names:

Root:*r1*
Root:*r1*/**Branch**:*b1*
Root:*r1*/**Branch**:*b1*/**Leaf**:*l1*@Attribute
Root:*r1*/**Branch**:*b2*

Note that you do NOT have to have an HTML <**form**> element for each XML element. Referencing **Root**:*r1*/**Branch**:*b1* implicitly creates the **Root**:*r1* element.

Refer to the **formCommand** documentation for a list of supported <**form**> elements and their syntax.

As in all XML documents, you can only have a single root element.

The value of an attribute or text node are determined by the current contents of the HTML <**form**> elements.

Nodes with an **ID** that begins with "COPY" are used to copy all child elements of the original **dom**(n) into the new **dom**(n) without creating an HTML <**form**> element for each node. The value of the HTML <**form**> element will be appended to the XPath of the original node and the entire subtree will be replicated in the new **dom**(n).

In addition to generating the **dom**(n), an additional *action* can be applied to the new **dom**(n). The *action* should include the *action* command followed by the node *name* (e.g. **Root**:*r1*/**Branch**:*b1*). Possible values for *action* are:

Add The node will be added to the **dom**(n). The node *name* can include one more attribute/value pairs. And attribute/value pair is seperated by an equal sign ("="). Multiple attribute/value pairs are seperated by at sign ("@"). For example: **Root**:*r1*/**Branch**:*b1*@first=1@second=2.

Delete The node will be deleted.

Copy, **Duplicate**
These commands are identical. The node and all children and attributes will be duplicated. The new node will appear immediately before the original node.

Insert Insert behaves identical to Copy except that all children and attributes of the new node are deleted.

METHOD OPTIONS

htmlWidget Name (hierarchy) of the htmlWidget.

action Action (if any) requested (Add, Delete, Copy, Duplicate or Insert).

formCommand *n cmd args*

HTML widget form command. Called by the Tkhtml widget when a form element is encountered. This method will build an appropriate widget in the Tkhtml widget. *n* is the token (not used), *cmd* is the command (input and textarea are the only currently supported commands, all others are silently ignored). *args* are the remaining parameters and depend on the *cmd*.

Supported form elements are:

<**input**
type="submit"
name=REQUIRED
value=REQUIRED>

<**input**
type="image"
name=REQUIRED
value=REQUIRED
src=REQUIRED>

<**input**
type="checkbox"
name=REQUIRED
value=REQUIRED
showvalue=OPTIONAL **(default false; nonstandard HTML)**
checked=OPTIONAL **(default false)**
disabled=OPTIONAL **(default false)**>

<**input**
type="radio"
name=REQUIRED

value=REQUIRED
showvalue=OPTIONAL **(default false;**
nonstandard HTML)
checked=OPTIONAL **(default false)**
disabled=OPTIONAL **(default false)**>

<**input**
type="text"
name=REQUIRED
value=REQUIRED
disabled=OPTIONAL **(default false)**
size=OPTIONAL **(default 20)**>

<**input**
type="hidden"
name=REQUIRED
value=REQUIRED>

<**textarea**
name=REQUIRED
rows=OPTIONAL **(default 5)**
cols=OPTIONAL **(default 80)**>
TEXT
</**textarea**>

<**select**
name=REQUIRED
size=OPTIONAL **(default is number of options)**
multiple=OPTIONAL **(default false)**
disabled=OPTIONAL **(default false)**>
<**option**
value=REQUIRED>
TEXT
</**option**>
</**select**>

The *showvalue* attribute of the radio and checkbox input types is not part of any HTML standards. Setting it to true will add the text of the value attribute to the button widget.

Forms are used by FanXE to edit XML nodes. The *name* attributes are used to determine the structure of the final node. An element should be uniquely identified in the form of **tag:ID** (where **tag** is the XML element tag and **ID** is a unique node name). See documentation of **form2xml** command for details on naming.

METHOD OPTIONS

n	Index of form elements.
cmd	Type of form element being passed.
args	Arguments to the form element.

STATIC PROCEDURES

FanXE::darken *color*

Return a slightly darker color. A slightly darker version of *color* will be returned.

PROC OPTIONS

color	Original color.

FanXE::parseArg *type value*

Parse the arguments from **dom**(n) function calls. Called by **FanXE**(n) Tcl XSLT extensions to parse arguments passed by by function calls. If *type* is **attrnodes**, return the last element of the *value* list. If *type* is **nodes**, return the text of the *value* node. Otherwise, return the *value*.

PROC OPTIONS

type	Type of value (literal or reference) being passed.
value	The value or reference to the value.

FanXE::safeDOM *node xpath cmd attr*

Safely query any node in the **dom**(n). This command is aliased to the *_safeInterp* as **domCmd** before evaluating the Title code. The *node* is included in the alias. Code in the Title node can use **domCmd** to query attributes or text of any other node in the current **dom**(n). Valid commands are **getAttribute** (which must include the *attr* to query) or **text**.

The text or attribute of all nodes matching the *xpath* expression are returned as a list.

PROC OPTIONS

node	Name of the **domNode**(n).
xpath	XPath of node to query.
cmd	Query command (**getAttribute** or **text**).
attr	Attribute to query (optional).

FanXE XSLT Extended Functions

These are extended XSLT functions available only for style sheets processed with **FanXE**(n). These functions will not operate in any other XSLT engine. To use these functions, you will need to define the *fanxe* namespace in your style sheet:

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fanxe="FanXE"
  extension-element-prefixes="fanxe">
```

fanxe:tcl *(string)*

Execute a Tcl command. The *string* will be executed in a safe **interp**(n). If an error is throw, it will be displayed via the **log**(n) command as an error and an empty string will be returned. Otherwise, the return value of the command will be returned as a string.

PROC OPTIONS

(string)	Command to be executed.

fanxe:mtime *(string)*

Calculate modification time of a file. The results of **file mtime** *string* will be returned as a string (the fanxe:tcl() function does not include the **file**(n) command).

PROC OPTIONS

(string)	Full path to filename.

fanxe:path *(node)*

Calculate a path string for **form2xml**. The full path as a series of **tag:ID** pairs separated by forward slashes (/) will be returned as a string. This path is used by the **form2xml** method.

PROC OPTIONS

(node)	The node to generate a path for (optional, default is current node).

fanxe:copyPath *(node)*

Calculate a path string for **form2xml** COPY values. The XPath of the element (excluding the parent node) will be returned. This function can be used as a value attribute input elements with a COPY id (see **form2xml**).

PROC OPTIONS

(node) The node to generate a path for (optional, default is current node).

Extended domNode(n) methods

These methods extend the **domNode**(n) methods.

::dom::domNode::fanxePath

Replacement for the **domNode**(n) **toXPath** method. If the *domNode* includes a default namespace, nodes in that namespace will use the tail of the namespaceURI as the node prefix. In all other aspects, this method is the same as the toXPath method (if the *domNode* does not include a default namespace, the toXPath method is simply called).

If the **dom**(n) was loaded with the **loadXML** method, the path returned by **fanxePath** will be a valid *xpathQuery* for the **domNode**(n) **selectNodes** method.

The modified XPath is returned.

SEE ALSO

FanXE(n), **Pgsql**(n), **RCS**(n), **domDoc**(n), **domNode**(n), **dom**(n), **file**(n), **interp**(n), **log**(n).

AUTHOR

Written by Tom Allard <tallard@frb.gov>