

NATPAGE

The NATPAGE control is always the top node of a Natural page's layout definition. The Natural page, on the one hand, generates the visible container in which all the contained elements are placed; on the other hand, some Natural-specific settings are defined on page level.

Properties

Basic

<p>translationreference</p>	<p>This is the "translation reference" that is passed to the multi language management.</p> <p>The "translation reference" is a logical term representing a group of textids together with their translation. If using the standard file based multi language management that comes with CIS as default then a "translation reference" represents one file containing text-ids and translations in a comma separated format.</p> <p>Translation information is loaded by the multi language management "per translation reference". I.e. if a page links to a certain translation reference then all the translation information that is available through this reference is loaded in one step and is also buffered.</p> <p>You can set up different scenarios: either each page may address an own translation reference. E.g. if your page is named "abc.xml" then it references to "abc" - as consequence there is (per language) one abc.csv file holding translation information for this page. If you have a second page "def.xml" then you may define "def" accordingly. In this case each page is independent from the other. - On the other side you are required to translate certain "common text-ids" multiple times.</p> <p>If you on the other hand define one translation reference for multiple pages then you can share text-ids throughout the various pages.</p> <p>Please set up a strategy for using translation references when starting using the multi language management. The strategy should also include a structured way of naming text-ids. Text-ids may only be shared in an efficient way if it is clear what they stand for. E.g. you may names of buttons in the following way: "btn_save" and "btn_saveas".</p>	<p>Sometimes obligatory</p>	
-----------------------------	---	-----------------------------	--

stylesheetfile	<p>URL of a style sheet file used for control rendering.</p> <p>Typically the style sheet file used for control rendering is set dynamically e.g. the style depends on the user who is currently logged on. When defining the style sheet file by this property, the style sheet file is not set dynamically but defined in a fix way for this page.</p> <p>The style sheet file must be defined as URL, relative to the generated page. A valid value may be <code>../softwareag/styles/CIS_DEFAULT.css</code>.</p> <p>If not using the "hard setting" of the style sheet file via this property then the style sheet is determined by the runtime in the following way:</p> <p>(1) The adapter object provides for a "String <code>getStyle()</code>" method that return the URL. You can override the default method and pass back your own URL.</p> <p>(2) When using the default implementation derived from <code>com.softwareag.cis.server.Model</code> then the <code>getStyle()</code> method accesses the CIS session context. You can set the session's style by calling <code>"findCISessionContext()"</code> in your adapter and calling <code>"setStyle()"</code> in the session context's object.</p>	Optional	css
addstylesheetfile	<p>URL of an additional style sheet file.</p> <p>You may use this additional style sheet file in order to define more styles than are provided in the "normal" style sheet file. Typical situations are:</p> <p>(A) Some controls offer the possibility to render defined content by style-class definitions (e.g. inside a TEXTGRID you can dynamically define which style-class is used for a certain cell).</p> <p>(B) If you define own controls by using the control extension framework and if these controls require own style classes then these style classes may be provided inside the additional style sheet file.</p> <p>By using the additional style sheet file you are able to avoid doing manipulations to the "normal" style sheet files that come from CIS or that are generated inside the tool "Style Sheet Editor".</p>	Optional	css

imagestopreload	<p>Semicolon separated list of image-URLs that are directly preloaded in an invisible area of the page. If images are used inside a tree or a text grid then they are loaded by dynamically generated HTML that is placed into a corresponding area of the page. In order to optimise the loading you can preload such images by listing them in this property.</p> <p>The URL of the images must be relative to your generated HTML page.</p> <p>Example: if your page has a tree with certain node images then you may define: "images/nodeopened.gif" images/nodeclosed.gif; images/nodeendnode.gif".</p>	Optional	
darkbackground	<p>Normally a page background is in light colour (white if using CIS_DEFAULT style sheet). CIS style sheets also have a dark(er) grey colour to be used.</p> <p>If DARKBACKGROUND is set to true then the darker background colour is chosen. This property typically is used if using the SUBCISPAGE tag or ROWTABSUBPAGES tag to seamlessly integrate inner pages into darker container areas.</p>	Optional	true false
helpid	<p>This is the id that is passed into the help management for the page.</p> <p>If a user clicks F1 inside the page and if there is no specific context sensitive control help available (e.g. help for field) then the help for the page is popped up.</p>	Optional	
visiblevalueifundefined	<p>Several CIS controls support a VISIBLEPROP property. The VISIBLEPROP contains the binding to an adapter property that decides at runtime if a control is visible or not.</p> <p>This property defines how these controls behave if there is no implementation available for the property.</p> <p>Example: the VISIBLEPROP of a CHECKBOX is binding to a property "cbvisible" but there is not corresponding implementation "getCbvisible". If set to "true" then all controls with undefined visibility are displayed. If set to "false" then they are hidden.</p>	Optional	true false
contextmenumethod	<p>Name of the event that is sent to the adapter when the user clicks into the page with the right mouse button and no other control (e.g. texgrid, tree,...) handled the click so far.</p>	Optional	

immediatedisplay	Flag that indicates if the screen is visible within the initial loading phase. Default is false. When using the default you see a light HTML page showing a "just loading" image. Use property "justloadingurl" to specify a page of choice.	Optional	true false
addjavascriptlibs	Comma separated list of URLs of additional javascript libraries. Example: <code>"/yourproject/js/yourlib.js"</code> . Used to include non-CIS javascript. Example of Usage: with the DATEINPUT control you can run own rules to convert and validate user input.	Optional	
flushmethod	Name of the event that is sent to the adapter in case the page loses the focus.	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
adapterlisteners	Semicolon separated list of classes which connect to the server side adapter processing as adapter listeners (each one supporting the interface IAdapterListener).	Optional	
framebufferpriority	Priority (integer) that is used to manage the page within the CIS frame buffer. Use value "-1" to indicate that the page should not be buffered at all (typically used when having a FILEUPLOAD2 control on the page). Default is "0". Use any other integer value to indicate higher priority.	Optional	0 -1
centralcontextmenu	If set to 'true' then the context menu is rendered in a central frame. This central frame can be specified via the "popupdivframe" setting in cisconfig.	Optional	true false
usexmlhttprequest	By default CIS framework is using hidden frame communication (asynchronous server communication). Use this attribute in order to use "XMLHttpRequests". Typical usage is with timer pages (to avoid seeing ongoing communication to server on browser's statusbar).	Optional	
withownborder	If set to "true" the page will be surrounded by an additional border.	Optional	true false
userinputprop	Name of the adapter parameter which will have a value of "true" if some userinput in the page or one of its subpages has been done since the last server-roundtrip.	Optional	
Natural			

natsource	Specifies a name for the Natural adapter object that will later be generated from your page layout. During adapter generation, this name is checked to match the Natural naming conventions for objects. If you do not specify a name here, the adapter name is taken from the layout name. This might result in names that are not valid for Natural objects. These adapters can only be used in Natural for Eclipse.	Optional	
natsinglebyte	Specifies whether string properties of the page are to be mapped to Unicode strings (U) or code page strings (A) in Natural. The value "true" means code page strings. The value "false" means Unicode strings (default).	Optional	true false
natrecursion	Properties of controls used in the page might have a recursive structure. These structures are mapped to multi-dimensional arrays in the Natural adapter. Natural arrays are limited to three dimensions. Therefore, the recursion depth of these structures can be limited using this property.	Optional	1 2 3 int-value
natdc	Specifies the character that is to be used as the decimal character in the format specifications of variables with decimal format in the parameter data area of the Natural adapter. For example, if a comma (,) is specified, "(N7,2)" is generated. If a period (.) is specified, "(N7.2)" is generated. The default is the period (.).	Optional	, .
natsss	The controls ROWTABLEAREA2 and MGDGRID support server-side scrolling and sorting. The corresponding data structures are generated into the parameter data area of the Natural adapter only if this attribute has been set to true. The default is false. This is for compatibility with earlier versions. For the control TEXTGRIDSSS2, the server-side scrolling data structures are always generated.	Optional	true false
natcv	Name of a Natural control variable that shall be assigned to the page. The control variable must be defined in a Data Definition (XCIDATADEF) control on the same page. The application can use the control variable to check the modification status of the page.	Optional	
xmlns:njx	Internal use only. Do not modify this.	Optional	
Popup			

popupwidth	Each CIS page can be opened as a popup dialog. This properties define the pixel width preferred for the page. - See the property "popupheight" for more information.	Optional	100px 200px 300px 400px
popupheight	Each CIS page can be opened as a popup dialog. This property defines the pixel height preferred for the page. A popup is typically opened by calling the "openPopup"-method in your adapter code. If no further definition is done then the popup will open in the height that is defined by this value. You can also dynamically manipulate the size and position of the popup by using the Model-method "setPopupFeatures" - please read corresponding documentation inside the Java API documentation.	Optional	100px 200px 300px 400px

popupfeatures	<p>In addition to POPUPWIDTH and POPUPHEIGHT you can control the appearance of the popup dialog in which the current page may be displayed. You define a string to maintain different feature aspects, separated by semi-colon.</p> <p>center:yes no</p> <p>edge:sunken raised</p> <p>resizable:yes no</p> <p>scroll:yes no</p> <p>status:yes no (to display or hide a status bar)</p> <p>An example string looks as follows: "dialogLeft:100px"</p> <p>There is one special function built in by which you can position a popup relative to its caller's window (the dialogLeft and dialogTop definition normally refer to absolute coordinates of the screen): by specifying "dialogLeft: SCRX(100)px" you define that the position is 100 pixels right from the left top corner of the current window. - Use "dialogTop: SCRY(100)px" in the same way for vertical positioning.</p> <p>Please also pay attention to the methods "setPopupTitle()" and "setPopupPageFeatures()" in the com.casabac.server.Model class. By using these method you can define popup parameters in a dynamic way inside your adapter implementation.</p>	Optional	<p>dialogLeft: 200px</p> <p>dialogTop: 100px</p> <p>edge: sunken</p> <p>resizable: yes</p> <p>status: no</p>
Occupied			
occupiedimage	<p>URL of the image that is displayed to indicate that the screen is just communicating to the server. This is the image that is located in the top left corner and which by default is a flashing hour glass.</p> <p>You can specify any image, e.g. also animated GIF files. If you want your image not to be visible in the top left corner but "somewhere" in the screen then draw an image with some transparent area on the left and above the image that you want to show.</p>	Optional	
occupiedpixelheight	<p>When the screen is busy, because the client is exchanging information with the server, an hour glass image is displayed at the top left corner. With this property you define the pixel height of this hour glass image.</p>	Optional	

occupiedpixelwidth	When the screen is busy, because the client is exchanging information with the server, an hour glass image is displayed at the top left corner. With this property you define the pixel width of this hour glass image.	Optional	
Hot Keys			
hotkeys	<p>Comma separated list of hot keys. A hotkey consists of a list of keys and a method name. Separate the keys by "-" and the method name again with a comma</p> <p>Example:</p> <p>ctrl-alt-65;onCtrlAltA;13;onEnter ...defines two hot keys. Method onCtrlAltA is invoked if the user presses Ctrl-Alt-A. Method "onEnter" is called if the user presses the ENTER key.</p> <p>Use the popup help within the Layout Painter to input hot keys.</p>	Optional	
Loading			
justloadingurl	URL of the page that is displayed to indicate that screen is just loading. Typically this is a light HTML page showing a loading image of choice. Use plain HTML - not a generated CIS page.	Optional	