

ROWTABLE0 and COLTABLE0

The ROWTABLE0 or COLTABLE0 container is not visible. Normally, it is just used for arranging controls. The following example shows how to define two columns - inside a ROWAREA - to arrange controls:

```
<pagebody>
  <rowarea name="Area 1">
    <itr takefullwidth="true">
      <coltable0 width="50%" takefullheight="true">
        <itr>
          <label name="Factor 1" width="100">
            </label>
            <field valueprop="factor1" length="5">
              </field>
          </itr>
        </coltable0>
      <coltable0 width="50%" takefullheight="true">
        <itr>
          <label name="Factor 2" width="100">
            </label>
            <field valueprop="factor2" length="5">
              </field>
          </itr>
        </coltable0>
      </itr>
    </rowarea>
  </pagebody>
```

The result looks as follows:

Area 1	
Factor 1	<input type="text" value="0"/>
Factor 2	<input type="text" value="0"/>

Inside the ROWAREA, two COLTABLE0 tags are placed - each occupying 50% of the width. Each COLTABLE0 area builds - independently from the other - its own table rows (ITR rows in the example).

All complex field arrangements should be done by using ROWTABLE0/COLTABLE0 tags as shown in the example.

The following topics are covered below:

- ROWTABLE0 Properties
- COLTABLE0 Properties

ROWTABLE0 Properties

Basic

height	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100% ". If the parent element does not specify a width then the rendering result may not represent what you expect.</p>	Optional	<p>100</p> <p>150</p> <p>200</p> <p>250</p> <p>300</p> <p>250</p> <p>400</p> <p>50%</p> <p>100%</p>
align	<p>Alignment of the content of the ITR row.</p> <p>Background: the ITR as independent table row renders a table into its content area. Inside this table a row is opened in which the controls are placed.</p> <p>This table normally is starting on the left of the ITR row. With this ALIGN property you can explicitly define the alignment of the table.</p>	Optional	<p>left</p> <p>center</p> <p>right</p>
valign	<p>Vertical alignment of control in its column.</p> <p>Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimtes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.</p>	Optional	<p>top</p> <p>middle</p> <p>bottom</p>

fixlayout	<p>The fixlayout property is important for saving rendering performance inside your browser. To become effective it requires to have specified the height and the width (if available as property) of the control.</p> <p>If setting fixlayout to "true" then the control's area is defined as area which is not sized dependent on its content (as normally done with table rendering). Instead the size is predefined from outside without letting the browser "look" into the content of the area. If the content is not fitting into the area then it is cut.</p> <p>You typically use this control if the content of the control's area is flexibly sizable. E.g. if the content (e.g. a TEXTGRID control) is following the size of the container.</p> <p>When using vertical percentage based sizing you should pay attention to set the fixlayout-property to "true" as often as possible. - The browser as consequence will be much faster in doing its rendering because a screen consists out of "building blocks" with simple to calculate sizes.</p>	Optional	true false
tablestyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.</p>	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
comment	<p>Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.</p>	Optional	
flashprop	\$en/popupwizard/njx__attr_flashprop\$	Optional	

COLTABLE0 Properties

The properties for COLTABLE0 are very similar to those of ROWTABLE0.

Basic

width	<p>Width of the control.</p> <p>There are three possibilities to define the width:</p> <p>(A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p> <p>(C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100% ". If the parent element does not specify a width then the rendering result may not represent what you expect.</p>	Optional	<p>100</p> <p>120</p> <p>140</p> <p>160</p> <p>180</p> <p>200</p> <p>50%</p> <p>100%</p>
widthprop	\$en/popupwizard/njx_coltable0_attr_widthprop\$	Optional	
takefullheight	<p>Indicates if the content of the control's area gets the full available height.</p> <p>If you use percentage sizing inside the control's area then this property must be switched to 'true'. If you use no explicit vertical sizing at all - or you use vertical pixel sizing for your controls - the property must be switched to 'false'.</p> <p>Background information: container control's internally open up a table in which you place rows (ITR/TR) which then hold controls (e.g. LABEL/FIELD). The table that is opened up normally has no explicit height and grows with its content as consequence. By specifying "takefullheight=true" the table itself is sized to fill the maximum height of the available area.</p>	Optional	<p>true</p> <p>false</p>

fixlayout	<p>The fixlayout property is important for saving rendering performance inside your browser. To become effective it requires to have specified the height and the width (if available as property) of the control.</p> <p>If setting fixlayout to "true" then the control's area is defined as area which is not sized dependent on its content (as normally done with table rendering). Instead the size is predefined from outside without letting the browser "look" into the content of the area. If the content is not fitting into the area then it is cut.</p> <p>You typically use this control if the content of the control's area is flexibly sizable. E.g. if the content (e.g. a TEXTGRID control) is following the size of the container.</p> <p>When using vertical percentage based sizing you should pay attention to set the fixlayout-property to "true" as often as possible. - The browser as consequence will be much faster in doing its rendering because a screen consists out of "building blocks" with simple to calculate sizes.</p>	Optional	true false
tablestyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.</p>	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
comment	<p>Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.</p>	Optional	