JSTabPanel

Extends

JSComponent

Constants Summary

Number	ACCORDION_PANEL Constant used for creating accordion panel from tab panel, by setting its tabOrientation.
Number	DEFAULT_ORIENTATION Constant used for restoring a tab panel orientation to it's initial state.
Number	HIDE Constant used for creating a tab panel that does not show tabs, by setting its tabOrientation.
Number	SPLIT_HORIZONTAL Constant used for creating horizontal split pane from tab panel, by setting its tabOrientation.
Number	SPLIT_VERTICAL Constant used for creating vertical split pane from tab panel, by setting its tabOrientation.

Property Summary

Number	anchors Enables a component to stick to a specific side of form and/or to grow or shrink when a window is resized.
String	background The background color of the component.
String	borderType The type, color and style of border of the component.
Boolean	enabled The enable state of the component, default true.
String	fontType The font type of the component.
String	foreground The foreground color of the component.
Number	formIndex The Z index of this component.
String	groupID A String representing a group ID for this component.
Number	height The height in pixels of the component.
String	name The name of the component.
JSMethod	onChange Method to be executed when the selected tab is changed in the tab panel or divider position is changed in split pane.
Number	printSliding Enables an element to resize based on its content and/or move when printing.
Boolean	printable Flag that tells if the component should be printed or not when the form is printed.
Boolean	scrollTabs Flag that tells how to arrange the tabs if they don't fit on a single line.
String	styleClass The name of the style class that should be applied to this component.
Number	tabOrientation Specifies either the position of the tabs related to the tab panel or the type of tab-panel.
Number	tabSeq An index that specifies the position of the component in the tab sequence.
Boolean	transparent Flag that tells if the component is transparent or not.
Boolean	visible The visible property of the component, default true.
Number	width The width in pixels of the component.
Number	x The x coordinate of the component on the form.
Number	y The y coordinate of the component on the form.

Method Summary

Object	getDesignTimeProperty() Get a design-time property of an element.
String[]	getDesignTimePropertyNames() Get the design-time properties of an element.
JSTab	getTab(name) Returns a JSTab instance representing the tab which has the specified name.
JSTab[]	getTabs() Returns an array of JSTab instances holding the tabs of the tab panel.
UUID	getUUID() Returns the UUID of this component.
JSTab	newTab(name, text, form) Adds a new tab with the text label and JSForm.
JSTab	newTab(name, text, form, relation) Adds a new tab with the text label and JSForm and JSRelation (can be null for unrelated).
Object	putDesignTimeProperty() Set a design-time property of an element.
Object	removeDesignTimeProperty() Clear a design-time property of an element.
void	removeTab(name) Removes the tab with the specified name from the tab panel.

Constants Details

ACCORDION_PANEL

Constant used for creating accordion panel from tab panel, by setting its tabOrientation.

Returns

Number

Sample

```
var accordion = myForm.newTabPanel('accordion', 10, 10, 620, 460);
accordion.tabOrientation = JSTabPanel.ACCORDION_PANEL;
```

DEFAULT_ORIENTATION

Constant used for restoring a tab panel orientation to it's initial state.

Returns

Number

Sample

```
var splitPane = myForm.newTabPanel('splitPane', 10, 10, 620, 460);
splitPane.tabOrientation = JSTabPanel.SPLIT_HORIZONTAL;
// (...) some code when you decide it's better to revert the orientation
splitPane.tabOrientation = JSTabPanel.DEFAULT_ORIENTATION;
```

HIDE

Constant used for creating a tab panel that does not show tabs, by setting its tabOrientation.

Returns

Number

Sample

```
var splitPane = myForm.newTabPanel('splitPane', 10, 10, 620, 460);
splitPane.tabOrientation = JSTabPanel.HIDE;
```

SPLIT_HORIZONTAL

Constant used for creating horizontal split pane from tab panel, by setting its tabOrientation.

Returns

Number

Sample

```
var splitPane = myForm.newTabPanel('splitPane', 10, 10, 620, 460);
splitPane.tabOrientation = JSTabPanel.SPLIT_HORIZONTAL;
```

SPLIT_VERTICAL

Constant used for creating vertical split pane from tab panel, by setting its tabOrientation.

Returns

Number

Sample

```
var splitPane = myForm.newTabPanel('splitPane', 10, 10, 620, 460);
splitPane.tabOrientation = JSTabPanel.SPLIT_VERTICAL;
```

Property Details

anchors

Enables a component to stick to a specific side of form and/or to grow or shrink when a window is resized.

If opposite anchors are activated then the component with grow or shrink with the window. For example if Top and Bottom are activated, then the component will grow/shrink when the window is vertically resized. If Left and Right are activated then the component will grow/shrink when the window is horizontally resized.

If opposite anchors are not activated, then the component will keep a constant distance from the sides of the window which correspond to the activated anchors.

Returns

Number

Sample

```
var form = solutionModel.newForm('mediaForm', 'db:/example_data/parent_table', null, false, 400, 300);
var strechAllDirectionsLabel = form.newLabel('Strech all directions', 10, 10, 380, 280);
strechAllDirectionsLabel.background = 'red';
strechAllDirectionsLabel.anchors = SM_ANCHOR.ALL;
var strechVerticallyLabel = form.newLabel('Strech vertically', 10, 10, 190, 280);
strechVerticallyLabel.background = 'green';
strechVerticallyLabel.anchors = SM_ANCHOR.WEST | SM_ANCHOR.NORTH | SM_ANCHOR.SOUTH;
var strechHorizontallyLabel = form.newLabel('Strech horizontally', 10, 10, 380, 140);
strechHorizontallyLabel.background = 'blue';
strechHorizontallyLabel.anchors = SM_ANCHOR.NORTH | SM_ANCHOR.WEST | SM_ANCHOR.EAST;
var stickToTopLeftCornerLabel = form.newLabel('Stick to top-left corner', 10, 10, 200, 100);
stickToTopLeftCornerLabel.background = 'orange';
stickToTopLeftCornerLabel.anchors = SM_ANCHOR.NORTH | SM_ANCHOR.WEST; // This is equivalent to SM_ANCHOR.
DEFAULT
var stickToBottomRightCornerLabel = form.newLabel('Stick to bottom-right corner', 190, 190, 200, 100);
stickToBottomRightCornerLabel.background = 'pink';
stickToBottomRightCornerLabel.anchors = SM_ANCHOR.SOUTH | SM_ANCHOR.EAST;
```

background

The background color of the component.

Returns

String

Sample

```
// This property can be used on all types of components.
// Here it is illustrated only for labels and fields.
var greenLabel = form.newLabel('Green',10,10,100,50);
greenLabel.background = 'green'; // Use generic names for colors.
var redField = form.newField('parent_table_text',JSField.TEXT_FIELD,10,110,100,30);
redField.background = '#FF0000'; // Use RGB codes for colors.
```

borderType

The type, color and style of border of the component.

Returns

String

Sample

```
//HINT: To know exactly the notation of this property set it in the designer and then read it once out
through the solution model.
var field = form.newField('my_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
field.borderType = solutionModel.createLineBorder(1,'#ff0000');
```

enabled

The enable state of the component, default true.

Returns

Boolean

Sample

```
var form = solutionModel.newForm('printForm', 'db:/example_data/parent_table', null, false, 400, 300);
var field = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
field.enabled = false;
```

fontType

The font type of the component.

Returns

String

Sample

```
var label = form.newLabel('Text here', 10, 50, 100, 20);
label.fontType = solutionModel.createFont('Times New Roman',1,14);
```

foreground

The foreground color of the component.

Returns

String

Sample

```
// This property can be used on all types of components.
// Here it is illustrated only for labels and fields.
var labelWithBlueText = form.newLabel('Blue text', 10, 10, 100, 30);
labelWithBlueText.foreground = 'blue'; // Use generic names for colors.
var fieldWithYellowText = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 50, 100, 20);
fieldWithYellowText.foreground = '#FFFF00'; // Use RGB codes for colors.
```

formIndex

The Z index of this component. If two components overlap, then the component with higher Z index is displayed above the component with lower Z index.

Returns

Number

Sample

```
var labelBelow = form.newLabel('Green', 10, 10, 100, 50);
labelBelow.background = 'green';
labelBelow.formIndex = 10;
var fieldAbove = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 10, 100, 30);
fieldAbove.background = '#FF0000';
fieldAbove.formIndex = 20;
```

groupID

A String representing a group ID for this component. If several components have the same group ID then they belong to the same group of components. Using the group itself, all components can be disabled/enabled or made invisible/visible.
The group id should be a javascript compatible identifier to allow access of the group in scripting.

Returns

String

Sample

```
var form = solutionModel.newForm('someForm', 'db:/example_data/parent_table', null, false, 400, 300);
var label = form.newLabel('Green', 10, 10, 100, 20);
var field = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 40, 100, 20);
label.groupID = 'someGroup';
field.groupID = 'someGroup';
forms['someForm'].elements.someGroup.enabled = false;
```

height

The height in pixels of the component.

Returns

Number

Sample

```
var field = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
application.output('original width: ' + field.width);
application.output('original height: ' + field.height);
field.width = 200;
field.height = 100;
application.output('modified width: ' + field.width);
application.output('modified height: ' + field.height);
```

name

The name of the component. Through this name it can also accessed in methods.

Returns

String

Sample

```
var form = solutionModel.newForm('someForm', 'db:/example_data/parent_table', null, false, 620, 300);
var label = form.newLabel('Label', 10, 10, 150, 150);
label.name = 'myLabel'; // Give a name to the component.
forms['someForm'].controller.show()
// Now use the name to access the component.
forms['someForm'].elements['myLabel'].text = 'Updated text';
```

onChange

Method to be executed when the selected tab is changed in the tab panel or divider position is changed in split pane.

Returns

JSMethod

Sample

```
var onChangeMethod = form.newMethod('function onTabChange(previousIndex, event) { application.output("Tab
changed from previous index " + previousIndex + " at " + event.getTimestamp()); }');
var tabPanel = form.newTabPanel('tabs', 10, 10, 620, 460);
tabPanel.newTab('tab1', 'Child Two', childOne);
tabPanel.newTab('tab2', 'Child Two', childTwo);
tabPanel.onChange = onChangeMethod;
```

printSliding

Enables an element to resize based on its content and/or move when printing. The component can move horizontally or vertically and can grow or shrink in height and width, based on its content and the content of neighboring components.

Returns

Number

Sample

```
var form = solutionModel.newForm('printForm', 'db:/example_data/parent_table', null, false, 400, 300);
var slidingLabel = form.newLabel('Some long text here', 10, 10, 5, 5);
slidingLabel.printSliding = SM_PRINT_SLIDING.GROW_HEIGHT | SM_PRINT_SLIDING.GROW_WIDTH;
slidingLabel.background = 'gray';
forms['printForm'].controller.showPrintPreview();
```

printable

Flag that tells if the component should be printed or not when the form is printed.

By default components are printable.

Returns

Boolean

Sample

```
var form = solutionModel.newForm('printForm', 'db:/example_data/parent_table', null, false, 400, 300);
var printedField = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
var notPrintedField = form.newField('parent_table_id', JSField.TEXT_FIELD, 10, 40, 100, 20);
notPrintedField.printable = false; // This field won't show up in print preview and won't be printed.
forms['printForm'].controller.showPrintPreview()
```

scrollTabs

Flag that tells how to arrange the tabs if they don't fit on a single line. If this flag is set, then the tabs will stay on a single line, but there will be the possibility to scroll them to the left and to the right. If this flag is not set, then the tabs will be arranged on multiple lines.

Returns

Boolean

Sample

```
var tabPanel = form.newTabPanel('tabs', 10, 10, 200, 200);
tabPanel.newTab('tab1', 'Child Two', childOne, parentToChild); // The first form uses the relation.
tabPanel.newTab('tab2', 'Child Two', childTwo);
tabPanel.scrollTabs = true;
```

styleClass

The name of the style class that should be applied to this component.

When defining style classes for specific component types, their names must be prefixed according to the type of the component. For example in order to define a class names 'fancy' for fields, in the style definition the class must be named 'field.fancy'. If it would be intended for labels, then it would be named 'label.fancy'. When specifying the class name for a component, the prefix is dropped however. Thus the field or the label will have its styleClass property set to 'fancy' only.

Returns

String

Sample

```
var form = solutionModel.newForm('printForm', 'db:/example_data/parent_table', null, false, 400, 300);
var field = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
var style = solutionModel.newStyle('myStyle','field.fancy { background-color: yellow; }');
form.styleName = 'myStyle'; // First set the style on the form.
field.styleClass = 'fancy'; // Then set the style class on the field.
```

tabOrientation

Specifies either the position of the tabs related to the tab panel or the type of tab-panel.
Can be one of SM_ALIGNMENT.(TOP, RIGHT, BOTTOM, LEFT), DEFAULT_ORIENTATION, HIDE, SPLIT_HORIZONTAL, SPLIT_VERTICAL, ACCORDION_PANEL.

Returns

Number

Sample

```
var tabPanel = form.newTabPanel('tabs', 10, 10, 620, 460);
tabPanel.newTab('tab1', 'Child Two', childOne, parentToChild); // The first form uses the relation.
tabPanel.newTab('tab2', 'Child Two', childTwo);
// The SM_ALIGNMENT constants TOP, RIGHT, BOTTOM and LEFT can be used to put the
// tabs into the needed position. Use HIDE to hide the tabs. Use DEFAULT_ORIENTATION to restore it to it's
initial state.
// The constants SPLIT_HORIZONTAL, SPLIT_VERTICAL can be used to create a split pane,
// where the first tab will be the first component and the second tab will the second component.
// ACCORDION_PANEL can be used to create an accordion pane.
tabPanel.tabOrientation = SM_ALIGNMENT.BOTTOM;
```

tabSeq

An index that specifies the position of the component in the tab sequence. The components
are put into the tab sequence in increasing order of this property. A value of 0 means
to use the default mechanism of building the tab sequence (based on their location on the form).
A value of -2 means to remove the component from the tab sequence.

Returns

Number

Sample

```
// Create three fields. Based on how they are placed, by default they will come one
// after another in the tab sequence.
var fieldOne = form.newField('parent_table_id', JSField.TEXT_FIELD, 10, 10, 100, 20);
var fieldTwo = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 40, 100, 20);
var fieldThree = form.newField('parent_table_id', JSField.TEXT_FIELD, 10, 70, 100, 20);
// Set the third field come before the first in the tab sequence, and remove the
// second field from the tab sequence.
fieldOne.tabSeq = 2;
fieldTwo.tabSeq = SM_DEFAULTS.IGNORE;
fieldThree.tabSeq = 1;
```

transparent

Flag that tells if the component is transparent or not.

The default value is "false", that is the components
are not transparent.

Returns

Boolean

Sample

```
// Load an image from disk an create a Media object based on it.
var imageBytes = plugins.file.readFile('d:/ball.jpg');
var media = solutionModel.newMedia('ball.jpg', imageBytes);
// Put on the form a label with the image.
var image = form.newLabel('', 10, 10, 100, 100);
image.imageMedia = media;
// Put two fields over the image. The second one will be transparent and the
// image will shine through.
var nonTransparentField = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 20, 100, 20);
var transparentField = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 50, 100, 20);
transparentField.transparent = true;
```

visible

The visible property of the component, default true.

Returns

Boolean

Sample

```
var form = solutionModel.newForm('printForm', 'db:/example_data/parent_table', null, false, 400, 300);
var field = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
field.visible = false;
```

width

The width in pixels of the component.

Returns

Number

Sample

```
var field = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
application.output('original width: ' + field.width);
application.output('original height: ' + field.height);
field.width = 200;
field.height = 100;
application.output('modified width: ' + field.width);
application.output('modified height: ' + field.height);
```

x

The x coordinate of the component on the form.

Returns

Number

Sample

```
var field = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
application.output('original location: ' + field.x + ', ' + field.y);
field.x = 90;
field.y = 90;
application.output('changed location: ' + field.x + ', ' + field.y);
```

y

The y coordinate of the component on the form.

Returns

Number

Sample

```
var field = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
application.output('original location: ' + field.x + ', ' + field.y);
field.x = 90;
field.y = 90;
application.output('changed location: ' + field.x + ', ' + field.y);
```

Method Details

getDesignTimeProperty

Object **getDesignTimeProperty** ()

Get a design-time property of an element.

Returns

Object

Sample

```
var frm = solutionModel.getForm('orders')
var fld = frm.getField('fld')
var prop = fld.getDesignTimeProperty('myprop')
```

getDesignTimePropertyNames

String[] **getDesignTimePropertyNames** ()

Get the design-time properties of an element.

Returns

String[]

Sample

```
var frm = solutionModel.getForm('orders')
var fld = frm.getField('fld')
var propNames = fld.getDesignTimePropertyNames()
```

getTab

JSTab **getTab** (name)

Returns a JSTab instance representing the tab which has the specified name.

Parameters

{String} name - The name of the tab that should be returned.

Returns

JSTab - A JSTab instance represented the requested tab.

Sample

```
var tabPanel = form.newTabPanel('tabs', 10, 10, 620, 460);
tabPanel.newTab('tab1', 'Child Two', childOne);
tabPanel.newTab('tab2', 'Child Two', childTwo);
tabPanel.getTab('tab2').text = 'Child Two Changed';
```

getTabs

JSTab[] **getTabs** ()

Returns an array of JSTab instances holding the tabs of the tab panel.

Returns

JSTab[] - An array of JSTab instances representing all tabs of this tabpanel.

Sample

```
var tabPanel = form.newTabPanel('tabs', 10, 10, 620, 460);
tabPanel.newTab('tab1', 'Child Two', childOne);
tabPanel.newTab('tab2', 'Child Two', childTwo);
var tabs = tabPanel.getTabs();
for (var i=0; i<tabs.length; i++)
        application.output("Tab " + i + " has text " + tabs[i].text);
```

getUUID

UUID **getUUID** ()

Returns the UUID of this component.

Returns

UUID

Sample

```
var button_uuid = solutionModel.getForm("my_form").getButton("my_button").getUUID();
application.output(button_uuid.toString());
```

newTab

JSTab **newTab** (name, text, form)

Adds a new tab with the text label and JSForm.

Parameters

{String} name - The name of the new tab.
{String} text - The text to be displayed on the new tab.
{JSForm} form - The JSForm instance that should be displayed in the new tab.

Returns

JSTab - A JSTab instance representing the newly created and added tab.

Sample

```
// Create a parent form.
var form = solutionModel.newForm('parentForm', 'db:/example_data/parent_table', null, false, 640, 480);
// Create a first child form.
var childOne = solutionModel.newForm('childOne', 'db:/example_data/child_table', null, false, 400, 300);
childOne.newField('child_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
// Create a relation to link the parent form to the first child form.
var parentToChild = solutionModel.newRelation('parentToChild','db:/example_data/parent_table','db:
/example_data/child_table',JSRelation.INNER_JOIN);
parentToChild.newRelationItem('parent_table_id','=','child_table_parent_id');
// Create a second child form.
var childTwo = solutionModel.newForm('childTwo', 'db:/example_data/my_table', null, false, 400, 300);
childTwo.newField('my_table_image', JSField.IMAGE_MEDIA, 10, 10, 100, 100);
// Create a tab panel and add two tabs to it, with the two child forms.
var tabPanel = form.newTabPanel('tabs', 10, 10, 620, 460);
tabPanel.newTab('tab1', 'Child Two', childOne, parentToChild); // The first form uses the relation.
tabPanel.newTab('tab2', 'Child Two', childTwo);
```

newTab

JSTab **newTab** (name, text, form, relation)

Adds a new tab with the text label and JSForm and JSRelation (can be null for unrelated).

Parameters

{String} name - The name of the new tab.
{String} text - The text to be displayed on the new tab.
{JSForm} form - The JSForm instance that should be displayed in the new tab.
{Object} relation - A JSRelation object that relates the parent form with the form
that will be displayed in the new tab.

Returns

JSTab - A JSTab instance representing the newly created and added tab.

Sample

```
// Create a parent form.
var form = solutionModel.newForm('parentForm', 'db:/example_data/parent_table', null, false, 640, 480);
// Create a first child form.
var childOne = solutionModel.newForm('childOne', 'db:/example_data/child_table', null, false, 400, 300);
childOne.newField('child_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
// Create a relation to link the parent form to the first child form.
var parentToChild = solutionModel.newRelation('parentToChild','db:/example_data/parent_table','db:
/example_data/child_table',JSRelation.INNER_JOIN);
parentToChild.newRelationItem('parent_table_id','=','child_table_parent_id');
// Create a second child form.
var childTwo = solutionModel.newForm('childTwo', 'db:/example_data/my_table', null, false, 400, 300);
childTwo.newField('my_table_image', JSField.IMAGE_MEDIA, 10, 10, 100, 100);
// Create a tab panel and add two tabs to it, with the two child forms.
var tabPanel = form.newTabPanel('tabs', 10, 10, 620, 460);
tabPanel.newTab('tab1', 'Child Two', childOne, parentToChild); // The first form uses the relation.
tabPanel.newTab('tab2', 'Child Two', childTwo);
```

putDesignTimeProperty

Object **putDesignTimeProperty** ()

Set a design-time property of an element.

Returns

Object

Sample

```
var frm = solutionModel.getForm('orders')
var fld = frm.getField('fld')
fld.putDesignTimeProperty('myprop', 'strawberry')
```

removeDesignTimeProperty

Object **removeDesignTimeProperty** ()

Clear a design-time property of an element.

Returns

Object

Sample

```
var frm = solutionModel.getForm('orders')
var fld = frm.getField('fld')
fld.removeDesignTimeProperty('myprop')
```

removeTab

void **removeTab** (name)

Removes the tab with the specified name from the tab panel.

Parameters

{String} name - the name of the tab to be removed

Returns

void

Sample

```
var tabPanel = form.newTabPanel('tabs', 10, 10, 620, 460);
tabPanel.newTab('tab1', 'Child Two', childOne);
tabPanel.newTab('tab2', 'Child Two', childTwo);
tabPanel.removeTab('tab1');
```