

ZK Component Reference

For ZK 9.6.1

Contents

Articles

ZK Component Reference	1
Introduction	1
Example Source Code	2
Base Components	2
AbstractComponent	2
FooterElement	3
FormatInputElement	4
HeaderElement	5
HeadersElement	6
HtmlBasedComponent	7
HtmlMacroComponent	9
InputElement	11
LabelElement	20
LabelImageElement	21
LayoutRegion	24
NumberInputElement	25
XulElement	27
Containers	28
Caption	28
Div	30
Drawer	32
Fragment	36
Groupbox	42
Idspace	46
Inputgroup	48
Nodom	50
Panel	51
Panelchildren	58
Span	60
Tabbox	61
Tab	66
Tabs	69
Tabpanel	71
Tabpanels	73

Window	75
Data	84
Grid	84
Column	112
Columns	114
Detail	118
Foot	120
Footer	122
Group	124
Groupfoot	126
Row	129
Rows	131
Listbox	134
Listcell	164
Listfoot	165
Listfooter	167
Listgroup	169
Listgroupfoot	171
Listhead	174
Listheader	178
Listitem	180
Tree	182
Treecell	196
Treechildren	198
Treecol	201
Treecols	203
Treefoot	205
Treefooter	208
Treeitem	210
Treerow	213
Biglistbox	215
Diagrams and Reports	220
Chart	220
Flashchart	257
Fusionchart	259
Gmaps	278
Gimage	286
Ginfo	287

Gmarker	289
Gpolyline	290
Gpolygon	292
Gscreen	294
Gcircle	295
Jasperreport	296
Timeline	301
Bandinfo	303
Hotzone	305
Timeplot	307
Plotinfo	310
Essential Components	313
A	314
Anchornav	317
Applet	320
Button	322
Captcha	329
Combobutton	331
Dropupload	335
Filedownload	342
Fileupload	346
Fisheye	351
Fisheyebar	355
Html	357
Iframe	359
Include	363
Image	370
Imagemap	373
Area	375
Label	377
Menu	382
Menubar	384
MenuItem	386
Menupopup	390
Menuseparator	393
Nav	395
Navbar	397
NavItem	400

Navseparator	403
Popup	404
Progressmeter	408
Rating	410
Selectbox	412
Separator	415
Space	417
Script	418
Style	420
Timer	422
Toolbar	423
Toolbarbutton	427
Input	430
Bandbox	430
Bandpopup	435
Calendar	437
Cascader	440
Checkbox	443
Chosenbox	446
CKEditor	451
Colorbox	458
Combobox	461
Comboitem	469
Datebox	470
Decimalbox	478
Doublebox	480
Doublespinner	482
Intbox	485
Longbox	487
Multislider	489
Radio	492
Radiogroup	493
Rangeslider	497
Searchbox	500
Signature	503
Slider	506
Sliderbuttons	511
Spinner	513

Tbeditor	516
Textbox	518
Timebox	521
Timepicker	524
Layouts	526
Absolutelayout	526
Absolutechildren	529
Anchorlayout	531
Anchorchildren	534
Borderlayout	537
Center	544
East	547
North	551
South	554
West	558
Box	562
Cardlayout	568
Columnlayout	571
Columnchildren	576
GoldenLayout	579
GoldenPanel	585
Hbox	588
Hlayout	592
Linelayout	595
Lineitem	599
Organigram	601
Orgchildren	605
Orgitem	606
Orgnode	610
Portallayout	611
Portalchildren	615
Rowlayout	619
Rowchildren	622
Splitlayout	623
Splitter	627
Tablelayout	631
TableChildren	634
Vbox	637

Vlayout	641
Multimedia and Miscellaneous	643
Audio	643
Barcode	645
BarcodeScanner	648
Cropper	653
Camera	657
Flash	662
Pdfviewer	663
Video	666
Track	672
Supplementary	674
Auxhead	674
Auxheader	677
Cell	682
Coachmark	685
Frozen	689
Paging	691
Stepbar	694
Step	697
Events	699
AfterSizeEvent	700
BandScrollEvent	701
CheckEvent	702
ColSizeEvent	703
CreateEvent	704
DropEvent	706
ErrorEvent	707
Event	708
HistoryPopStateEvent	709
InfoChangeEvent	712
InputEvent	713
KeyEvent	714
MapDropEvent	715
MapMouseEvent	716
MapMoveEvent	717
MapTypeChangeEvent	718
MapZoomEvent	719

MouseEvent	720
MoveEvent	721
OccurEventSelectEvent	722
OpenEvent	723
OverPlotEvent	724
PageSizeEvent	725
PagingEvent	726
PortalMoveEvent	727
ScrollEvent	728
SelectEvent	729
SelectionEvent	731
SizeEvent	732
VisibilityChangeEvent	733
UploadEvent	735
ZIndexEvent	736
Supporting Classes	737
AbstractListModel	737
Constraint	738
Constrained	739
ListitemRenderer	740
ListModel	741
Messagebox	742
RendererCtrl	747
SimpleConstraint	748
SimpleListModel	749
XHTML Components	750
Encoding URLs	750
In Pure Java	751
The Difference Between XUL and XHTML Components	752
XML Components	752
Transformer	753
Annotation	755
Data Binding	755
Tablet Devices	758
Configuration	758
Viewport	758
Components	759
ScrollView	759

Events	762
SwipeEvent	762
ClientInfoEvent	764
UI Enhancements	766
Borderlayout	766
Calendar	768
Cardlayout	769
Colorbox	770
Combobox	771
Datebox	772
Doublespinner	773
Grid	774
Groupbox	775
Listbox	776
Paging	777
Panel	778
Tabbox	779
Timebox	781
Tree	782
Spinner	783
Window	784
Unsupported Molds	786
Limitation	787
Accessibility	788
Containers	788
Drawer	788
Tabbox	789
Data	789
Biglistbox	790
Grid	790
Listbox	791
Tree	791
Essential Components	792
Include	792
Menubar	793
Navbar	794
Rating	794
Toolbar	795

Input	796
Bandbox	796
Calendar	797
Cascader	798
Chosenbox	798
Colorbox	799
Datebox	800
Multislider	800
Rangeslider	801
Searchbox	802
Slider	803
Timebox	803
Timepicker	804
Layouts	804
Borderlayout	805
Cardlayout	805
Linelayout	806
Organigram	807
Portallayout	807
Splitter	808
Splitlayout	809
Multimedia and Miscellaneous	809
Barcode	810
Barcodescanner	810
Pdfviewer	811
Supplementary	811
Stepbar	812
Unsupported Components	812

References

Article Sources and Contributors	814
Image Sources, Licenses and Contributors	822

ZK Component Reference

Documentation:Books/ZK_Component_Reference

If you have any feedback regarding this book, please leave it here.

<comment>http://books.zkoss.org/wiki/ZK_Component_Reference</comment>

Introduction

This guide acts as a reference of components. We outline each of the XUL components and include discussions of XHTML and XML components.

- For introductory concepts, please refer to ZK Developer's Reference: UI Composing.
- For general operations, such as drag-and-drop, hflex/vflex, tooltips and context menus, please refer to ZK Developer's Reference: UI Patterns.

The guide follows a distinctive pattern outlining the following items:

- A link to a demonstration of the component
- A link to the component's Java API
- A link to the component's JavaScript API
- The employment and purpose of the component
- An example with source of the component
- The component's supported events and children
- Component's use cases
- The version history of the component

Example Source Code

Some source code introduced in this book is at GitHub^[1].

References

[1] <https://github.com/zkoss/zkbooks/tree/master/componentreference>

Base Components

This section outlines the base components on which other ZK components are built.

AbstractComponent

Abstract Component

- Demonstration: N/A
- Java API: AbstractComponent^[1]
- JavaScript API: N/A

Employment/Purpose

A skeletal implementation of Component. Though it is OK to implement Component from scratch, this class simplifies some of the chores.

Example

N/A

Supported Events

Name	Event Type
None	None

Supported Children

* ALL

Use cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
5.0.4	August 2010	<code>stubonly</code> is introduced to ZK EE to minimize the memory footprint for large applications. Refer to here for details.

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/AbstractComponent.html#>

FooterElement

Footer Element

- Demonstration: N/A
- Java API: FooterElement ^[1]
- JavaScript API: FooterWidget ^[2]
- Style Guide: N/A

Employment/Purpose

A skeletal implementation for a footer.

Example

N/A

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: LabelImageElement

Supported Children

*All

Version History

Version	Date	Content
5.0.4	July, 2010	new added component

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/impl/FooterElement.html#>

[2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/mesh/FooterWidget.html#>

FormatInputElement

Format Input Element

- Demonstration: N/A
- Java API: FormatInputElement ^[1]
- JavaScript API: N/A

Employment/Purpose

A skeletal implementation for an input box with format

Example

N/A

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: InputElement

Supported Children

*NONE

Use cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/impl/FormatInputElement.html#>

HeaderElement

Header Element

- Demonstration: N/A
- Java API: HeaderElement ^[1]
- JavaScript API: N/A

Employment/Purpose

A skeletal implementation for a header

Example

N/A

Supported Events

Name	Event Type
onColSize	Event: ColSizeEvent ^[2] Notifies the parent of a group of headers that the widths of two of its children are changed by the user.

- Inherited Supported Events: XulElement

Supported Children

* ALL

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/impl/HeaderElement.html#>
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/event/ColSizeEvent.html#>

HeadersElement

Headers Element

- Demonstration: Headers Element ^[1]
- Java API: HeadersElement ^[2]
- JavaScript API: HeadWidget ^[3]

Employment/Purpose

A skeletal implementation for headers, the parent of a group of HeaderElement ^[4].

Example

N/A

Supported Events

Name	Event Type
onColSize	Event: ColSizeEvent ^[2] Denotes user has resized one of the columns.

- Inherited Supported Events: XulElement

Supported Children

Header Element

Use cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/zkdemo/userguide/#g13>
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/impl/HeadersElement.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/mesh/HeadWidget.html#>
- [4] <http://www.zkoss.org/javadoc/2.4.1/zul/org/zkoss/zul/impl/HeaderElement.html>

HtmlBasedComponent

Html Based Component

- Demonstration: N/A
- Java API: HtmlBasedComponent ^[1]
- JavaScript API: Widget ^[2]

Employment/Purpose

A skeletal implementation for HTML based components. It simplifies to implement methods common to HTML based components.

Example

N/A

Supported Events

Name	Event Type
onDrop	Event: DropEvent ^[3] Denotes user has dropped the dragged target to the component.
onClick	Event: MouseEvent ^[4] Denotes user has clicked the component.
onDoubleClick	Event: MouseEvent ^[4] Denotes user has double-clicked the component.
onRightClick	Event: MouseEvent ^[4] Denotes user has right-clicked the component.
onMouseOver	Event: MoveEvent ^[5] Denotes user has hovered over the component.
onMouseOut	Event: MoveEvent ^[5] Denotes user has moved out the component.
onOK	Event: KeyEvent ^[6] Denotes user has pressed the ENTER key.
onCancel	Event: KeyEvent ^[6] Denotes user has pressed the ESC key.
onCtrlKey	Event: KeyEvent ^[6] Denotes user has pressed a special key, such as PgUp, Home and a key combined with the Ctrl or Alt key. Refer to the ctrlKeys Property section below for details.

Supported Children

* ALL

Use cases

Version	Description	Example Location

Version History

Version	Date	Content
5.0.3	June 2010	The onMouseOver and onMouseOut events are supported.

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/HtmlBasedComponent.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zk/Widget.html#>
- [3] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/DropEvent.html#>
- [4] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/MouseEvent.html#>
- [5] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/MoveEvent.html#>
- [6] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/KeyEvent.html#>

HtmlMacroComponent

Html Macro Component

- Demonstration: N/A
- Java API: AbstractComponent^[1]
- JavaScript API: N/A

Employment/Purpose

The base class for macro components.

Since ZK 5.0.4, by default invoking `HtmlMacroComponent.afterCompose()`^[1] supports auto forward events and wire accessible variables to this component.

For example, (usemacro.zul)

```
<?init zscript="macro.zs"?>
<?component name="username" macroURI="macro.zul" class="Username"?>
<window id="wnd">
    <username id="ua"/>
    <username label="Account"/>
</window>
```

(macro.zs)

```
import org.zkoss.zk.ui.*;
import org.zkoss.zul.*;

public class Username extends HtmlMacroComponent {
    Button btn; // auto wire

    // auto forward
    public void onClick$btn () {
        System.out.println("success... and btn variable is not null
: " + (btn != null));
    }
}
```

(macro.zul)

```
<grid id="mc_grid">
    <rows>
        <row id="r">
            <button label="${empty arg.label ? 'Username' : arg.label}" id="btn"/>
        </row>
    </rows>
</grid>
```

If you want to turn off the auto wiring mechanism, please refer to the following steps:

Turn off auto wire mechanism by specifying the Library Property "org.zkoss.zk.ui.macro.autowire.disabled" to "true" in WEB-INF/zk.xml. If you did not specify the Library Property, the default is false.

```
<library-property>
    <name>org.zkoss.zk.ui.macro.autowire.disabled</name>
        <value>true</value>
</library-property>
```

or turn off auto forward events by specifying the Library Property "org.zkoss.zk.ui.macro.autoforward.disabled" to "true" in WEB-INF/zk.xml. If you did not specify the Library Property, the default is false.

```
<library-property>
    <name>org.zkoss.zk.ui.macro.autoforward.disabled</name>
        <value>true</value>
</library-property>
```

In the early version, if you want to apply the auto-wiring, you can invoke `java.lang.Object` Components.wireVariables(`org.zkoss.zk.ui.Component`, ^[2] `java.lang.Object`) ^[2] in `HtmlMacroComponent.afterCompose()` ^[1] as follows.

```
public void afterCompose() {
    super.afterCompose(); //create components

    Components.wireVariables(this, this);
    Components.addForward(this, this);
}
```

Example

N/A

Supported Events

Name	Event Type
None	None

See also events inherited from `HtmlBasedComponent`'s Supported Events.

Supported Children

* ALL

Use cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
5.0.3	June 2010	The corresponding DOM element is customizable. It is default to SPAN (the same as prior version) but you can change it to any tag by use of HtmlMacroComponent.setEnclosingTag(java.lang.String) ^[3] .
5.0.4	August 2010	By default, invoking HtmlMacroComponent.afterCompose() ^[1] supports auto forward events and wire accessible variables to this component.

References

- [1] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/HtmlMacroComponent.html#afterCompose\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/HtmlMacroComponent.html#afterCompose())
- [2] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/Components.html#wireVariables\(org.zkoss.zk.ui.Component](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/Components.html#wireVariables(org.zkoss.zk.ui.Component),
- [3] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/HtmlMacroComponent.html#setEnclosingTag\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/HtmlMacroComponent.html#setEnclosingTag(java.lang.String)

InputElement

Input Element

- Demonstration: N/A
- Java API: InputElement^[1]
- JavaScript API: InputWidget^[2]

Employment/Purpose

InputElement is a super class for components which prove user key input, such as `textbox`, `intbox`, `decimalbox`, `doublebox`, `datebox`, `timebox`, `spinner`, `combobox`, and `bandbox`.

Some features are implemented in this class, such as constraint, disabled, maxlength, name, readonly, and so on.

You should not use this class directly, please use the inherited class.

Example

```
<grid>
    <rows>
        <row>
            UserName <textbox value="Jerry" width="150px" />
        </row>
        <row>
```

```

        Password <textbox type="password" value="foo" width="150px" />
</row>
<row>
    Phone: <intbox constraint="no negative,no zero" width="150px" value="12345678" />
</row>
<row>
    Weight: <decimalbox format="###.##" value="154.32" width="150px" />
</row>
<row>
    Birthday: <datebox id="db" width="150px" />
</row>
<row>
    E-mail:
    <textbox width="150px" value="zk@zkoss.org"
        constraint="/.+@.+\.+[a-z]+/: Please enter an
e-mail address" />
</row>
</rows>
</grid>

```

UserName	<input type="text" value="Jerry"/>
Password	<input type="password" value="***"/>
Phone:	<input type="text" value="12345678"/>
Weight:	<input type="text" value="154.32"/>
Birthday:	<input type="text"/> 
E-mail:	<input type="text" value="zk@zkoss.org"/>

Validation

There are two ways to validate the value entered by an user: implementing Constraint^[3] or throwing WrongValueException^[4].

Constraint

An input element can be associated with a constraint (Constraint^[3]) to validate the value entered by an user. There is a default implementation called SimpleConstraint^[5] that can handle many conditions. If it is not enough, you can implement your own constraint, or throwing WrongValueException^[4] as described in the next sections.

Built-in Constraints

To use the default constraint, you could specify a list of conditions in InputElement.setConstraint(java.lang.String)^[6], such as no positive and no empty. For example,

```
<textbox constraint="no empty"/>
<intbox constraint="no negative,no zero"/>
```

Condition	+ You can specify following values at constraint attribute to apply them
no empty	Empty is not allowed.
no future	Date in the future is not allowed.
no negative	Negative numbers are not allowed.
no past	Date in the past is not allowed.
no positive	Positive numbers are not allowed.
no today	Today is not allowed.
no zero	Zero numbers are not allowed.
between yyyyMMdd and yyyyMMdd	Date only allowed between the specified range. The format must be yyyyMMdd, such as <datebox constraint="between 20071225 and 20071203"/>
after yyyyMMdd	Date only allowed after (and including) the specified date. The format must be yyyyMMdd, such as <datebox constraint="after 20071225"/>
before yyyyMMdd	Date only allowed before (and including) the specified date. The format must be yyyyMMdd, such as <datebox constraint="before 20071225"/>
end_before end_after after_start after_end ...	Specifies the position of the error box. Please refer to Popup for all allowed position. <textbox constraint="no empty, end_after"/> <textbox constraint="no empty, start_before"/>

Regular Expression

To specify a regular expression, you could have to use / to enclose the regular expression as follows.

```
<textbox constraint="/.+@.+\.+[a-z]+/"/>
```

Notice that the above statement is XML, so do not use \\ to specify a backslash. On the other hand, it is required, if writing in Java:

```
new Textbox().setConstraint("/.+@.+\\\.+[a-z]+/");
```

Flags

To specify the flags to the regular expression, you could add the flags after the ending slash of the regular expression.

For example, If you want to enable case-insensitive matching, you could add the flag as bellow.

```
<textbox constraint="/[A-Z]{3}/i"/>
```

The flags supported:

flags	Description
i	ignore case
m	multiline
s	dotAll
u	unicode

Notice: the regular expression will always using global match no matter the `g` flag is added or not.

Multiple Constraints

Notice that it is allowed to mix regular expression with other constraints by separating them with a comma.

If you prefer to display an application dependent message instead of default one, you could append the constraint with colon and the message you want to display when failed.

```
<textbox constraint="/.+\@\.[a-z]+/: e-mail address only"/>
<datebox constraint="no empty, no future: now or never"/>
of course, it supports multiple custom messages
<intbox constraint="no negative: forbid negative, no positive: forbid positive" />
```

i18n Error Message

To support multilingual, you could use the `l` function as depicted in ZK Developer's Reference.

```
<?taglib uri="http://www.zkoss.org/dsp/web/core" prefix="c"?>
<textbox constraint="/.+\@\.[a-z]+/: ${c:l('err.email.required')}"/>
```

Escape a Comma

If you want to write a longer sentence with comma separator, you can enclose your customized sentence with curly braces.

[Since 8.0.0]

```
<textbox constraint="no empty: {Sorry, no empty allowed}, /.+\@\.[a-z]+/: email only"></textbox>
```

Scollable Error Messag

If you want to move an error message box by scrolling, please use data-scrollable attribute ^[7].

Custom Constraint

If you want a custom constraint, you could implement Constraint ^[3] and specify it in the constraint property (InputElement.setConstraint(org.zkoss.zul.Constraint) ^[8]).

```
public class EvenNumberConstraint implements Constraint {
    public void validate(Component comp, Object value) throws WrongValueException {
        if (value != null && new Integer(value.toString()).intValue() % 2 == 1)
            throw new WrongValueException(comp, "Only even numbers are allowed, not "+value);
    }
}
```

- Line 4: If the validation fails, just throw WrongValue ^[9]. Notice that you have to specify which component causes the exception.

To specify it to the constraint property, you have to instantiate it first by use of the new function as shown below

```
<?taglib uri="http://www.zkoss.org/dsp/web/core" prefix="c"?>
<textbox constraint="${c:new('foo.EvenNumberConstraint')}" />
```

Display Error Message in Custom Way

Instead of the default error box, you could provide a custom approach by implementing CustomConstraint ^[10] (with Constraint ^[3]). Then, org.zkoss.zk.ui.WrongValueException) CustomConstraint.showError(org.zkoss.zk.ui.Component, org.zkoss.zk.ui.WrongValueException) ^[11] will be invoked when an exception is caught. For example,

```
<window title="Custom Constraint" border="normal">
    <zscript><![CDATA[
        class MyConst implements Constraint, CustomConstraint {
            //Constraint//
            public void validate(Component comp, Object value) {
                if (value == null || ((Integer)value).intValue() < 100)
                    throw new WrongValueException(comp, "At least 100 must be specified");
            }
            //CustomConstraint//
            public void showError(Component comp,
WrongValueException ex) {
                errmsg.setValue(ex != null ? ex.getMessage() :
                " ");
            }
        }
        Constraint ctt = new MyConst();
    ]]>
```

```

</zscript>
<hlayout>
    Enter a number at least 100:
    <intbox constraint="${ctt}" />
    <label id="errmsg" />
</hlayout>
</window>

```

And, here is the result

Validate at Client for Better Responsiveness

Responsiveness could be improved by validating more constraints at the client side^[12]. To do it, you have to implement ClientConstraint^[13] (with Constraint^[3]).

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/impl/InputElement.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/inp/InputWidget.html#>
- [3] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Constraint.html#>
- [4] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/WrongValueException.html#>
- [5] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/SimpleConstraint.html#>
- [6] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/impl/InputElement.html#setConstraint\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/impl/InputElement.html#setConstraint(java.lang.String))
- [7] http://books.zkoss.org/wiki/ZUML_Reference/ZUML/Namespaces/Client_Attribute/Data-Scrollable
- [8] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/impl/InputElement.html#setConstraint\(org.zkoss.zul.Constraint\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/impl/InputElement.html#setConstraint(org.zkoss.zul.Constraint))
- [9] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/WrongValue.html#>
- [10] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/CustomConstraint.html#>
- [11] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/CustomConstraint.html#showCustomError\(org.zkoss.zk.ui.Component\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/CustomConstraint.html#showCustomError(org.zkoss.zk.ui.Component))
- [12] The default constraint (InputElement (<http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/impl/InputElement.html#>)) validates all constraints at the client side
- [13] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/ClientConstraint.html#>

WrongValueException

In addition to throwing WrongValueException in java.lang.Object) Constraint.validate(org.zkoss.zk.ui.Component, java.lang.Object) ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Constraint.html#validate\(org.zkoss.zk.ui.Component, java.lang.Object\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Constraint.html#validate(org.zkoss.zk.ui.Component, java.lang.Object))), you can throw WrongValueException in other situations. For example, you can validate the user name and password when the user presses the login button. For example,

```

public class FooComposer extends SelectorComposer {
    @Wire
    private Textbox username;
    @Wire
    private Textbox password;

    @Listen("onClick = #login")
    public void doLogin() {
        username.clearErrorMessage(); //important to clear the previous
        error, if any
        if (examine(username, password)) {
            //success
    }
}

```

```
    } else {
        throw new WrongValueException(username, "Not a valid username or
password. Please retry.");
    }
}
```

However, notice that you have to clear the error message manually by invoking InputElement.clearErrorMessage() ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/impl/InputElement.html#clearErrorMessage\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/impl/InputElement.html#clearErrorMessage())). Otherwise, the error message will remain there unless Textbox.setValue(java.lang.String) ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Textbox.html#setValue\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Textbox.html#setValue(java.lang.String))) is called.

Properties

Inplace

All input elements can have the in-place-editing functionality, like the combobox, textbox, datebox, and so on.

```
<grid width="500px">
    <rows>
        <row>
            Textbox:
            <textbox inplace="true" value="Click me" />
        </row>
        <row>
            Combobox:
            <combobox inplace="true" value="Click me" />
        </row>
    </rows>
</grid>
```

Instant

since 6.0.0

When the instant mode is on, the `onChange` event will be fired as soon as possible as a user is typing in the input (like `onChanging` event). The value will also be updated to the component (server-side) immediately. Notice that it would automatically synchronize the value (including format) in the client and server. If you want to do something in the server while changing the value, please use `onChanging` event.

Placeholder

since 6.5.0

ZK 6.5 introduces support for HTML5 placeholder text, a very useful feature for telling users what they should enter in a textbox. This is a widely regarded UI pattern.

The following image and code show the look of the placeholder as well as the code to replicate it.



```
<textbox placeholder="Please type some text" />
```

InputAttributes

since 8.6.1

The feature is available since 8.6.1. All input elements can set some additional attributes to the input html tag in the component. The inputAttributes can take a Map with attribute names as the keys or a String separate by ";" and follow the name=value rule.

```
<bandbox inputAttributes="${map}" /></bandbox>
<datebox inputAttributes="autocorrect=off;spellcheck=true" /></datebox>
```

Supported Events

Name	Event Type
onChange	Event: InputEvent (http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/InputEvent.html#) Denotes the content of an input component has been modified by the user.
onChanging	Event: InputEvent (http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/InputEvent.html#) Denotes that user is changing the content of an input component. Notice that the component's content (at the server) won't be changed until onChange is received. Thus, you have to invoke the <code>InputEvent.getValue()</code> to retrieve the changed value.
onSelection	Event: SelectionEvent (http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/SelectionEvent.html#) Denotes that user is selecting a portion of the text of an input component. You can retrieve the start and end position of the selected text by use of the <code>getStart</code> and <code>getEnd</code> methods.
onFocus	Event: Event (http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/Event.html#) Denotes when a component gets the focus. Remember event listeners execute at the server, so the focus at the client might be changed when the event listener for onFocus got executed.
onBlur	Event: Event (http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/Event.html#) Denotes when a component loses the focus. Remember event listeners execute at the server, so the focus at the client might be already changed when executing the onBlur listener.
onError	Event: ErrorEvent (http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/ErrorEvent.html#) Denotes when a component caused a validation error.

Supported Children

*None

Use cases

Version	Description	Example Location
---------	-------------	------------------

Browser Limitations

Browser	description
iOS Safari / Chrome	<pre><textbox id="test"/> <button label="java focus (no keyboard)" onClick="test.focus()"></button> <button xmlns:w="client" label="client focus" w:onClick="zk.Widget.\$('test').focus();"></button></pre> <p>In mobile Safari, the user must explicitly tap the elements in the web view to display the keyboard.</p>

Version History

Version	Date	Content
5.0.8	June, 2011	Allow user to specify the position of error-box
6.0.0	Sep, 2011	Add instant mode, which sends onChange event and update value to component as soon as possible.
6.5.0	Sep, 2012	Support HTML5 placeholder attribute for input elements
8.5.2	May, 2018	ZK-3774 (http://tracker.zkoss.org/browse/ZK-3774): focus() doesn't work on mobile device
8.6.1	Jan, 2019	ZK-4111 (http://tracker.zkoss.org/browse/ZK-4111): Add autocorrect and spellcheck DOM attributes toggles to input-based components

LabelElement

Label Element

- Demonstration: N/A
- Java API: LabelElement^[1]
- JavaScript API: N/A

Employment/Purpose

A HTML element with a label.

Example

N/A

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

*All

Version History

Version	Date	Content

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/impl/LabelElement.html#>

LabelImageElement

Label Image Element

- Demonstration: N/A
- Java API: LabelImageElement^[1]
- JavaScript API: LabelImageWidget^[2]

Employment/Purpose

An HTML element with a label and an image.

Preload Image

The feature is applied to all of the LabelImageElement and Image components.

By default the preload function is disabled, so users have to specify the *custom-attributes* and set it to true. For example,

```
<button image="xxx.png">
  <custom-attributes org.zkoss.zul.image.preload="true"/>
</button>
```

Or specify it just below the root component.

For example,

```
<window>
  <custom-attributes org.zkoss.zul.image.preload="true"/>
  <button image="xxx.png"/>
  <image src="xxx.png"/>
</window>
```

As you can see, the *custom-attributes* will be checked recursively (see also Scope.getAttribute(java.lang.String,boolean)^[3]).

The feature can also applied from zk.xml as a library property.

For example,

```
<!-- zk.xml -->
<zk>
  <library-property>
    <name>org.zkoss.zul.image.preload</name>
    <value>true</value>
  </library-property>
</zk>
```

IconSclass

Font Awesome Bundled

ZK 7.0.0 integrates Font Awesome 4.0.1^[4] with the prefix **z-icon**. To use it, just specify the `iconSclass` attribute. For a complete list of icons, please refer to FontAwesome Cheatsheet^[5].

For example, to add a home icon on a Button,

```
<window>
  <button iconSclass="z-icon-home" />
</window>
```

If you want to use other Font Awesome function such as the animation icon, you can include the external font awesome CSS link and add the CSS class to `iconSclass`. For example,

```
<?link href="http://netdna.bootstrapcdn.com/font-awesome/4.0.1/css/font-awesome.css" rel="stylesheet"?>
<window>
  <button iconSclass="z-icon-bell fa-spin" />
  <button iconSclass="z-icon-home fa-2x fa-rotate-90" />
</window>
```

ZK 8.0.0 integrates Font Awesome 4.3.0. Also, with ZK 8 there is no need to include external font awesome CSS link to use an animation icon. Therefore the example above becomes

```
<window>
  <button iconSclass="z-icon-bell z-icon-spin" />
  <button iconSclass="z-icon-home z-icon-2x z-icon-rotate-90" />
</window>
```

ZK 8.5.2 integrates Font Awesome 4.7.0^[6].

Use Other Icons

The `iconSclass` not only applies to font awesome icons ("z-icon-" classes): Any css class can be applied to customize. Notice you need to manually include related CSS or font file. For example, the bootstrap glyphicons can be used here too:

```
<window>
  <button iconSclass="glyphicon glyphicon-envelope" />
</window>
```

Since the web font is loaded after the character is being displayed if there is no cache, ZK doesn't know if the web font is ready when initializing. Therefore using `hflex="min"` with `iconSclass` may not get the desired result. Moreover, the final width of icons might not be the same. To make the width of icons always be fixed, add `z-icon-fw`.

```
<window hflex="min">
  <button iconSclass="z-icon-fw z-icon-home" />
  <button iconSclass="z-icon-fw z-icon-bell z-icon-spin" />
</window>
```

```
</window>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: LabelElement

Supported Children

*All

Version History

Version	Date	Content
8.6.2	May 2019	ZK-4243: The result of hflex="min" is not sufficient if the content has Font Awesome icons ^[7]
7.0.0	October 2012	Add iconScss attribute with FontAwesome supported.
6.0.0	September 2011	A way to pre-load images since many UIs depend on the size of an image ^[8]

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/impl/LabelImageElement.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/LabelImageWidget.html#>
- [3] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/ext/Scope.html#getAttribute\(java.lang.String,boolean\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/ext/Scope.html#getAttribute(java.lang.String,boolean))
- [4] <https://fontawesomelib.com/releases/4.0.1/list/all/index.html>
- [5] <http://fontawesome.io/cheatsheet/>
- [6] <https://fontawesome.com/v4.7.0/icons/>
- [7] <https://tracker.zkoss.org/browse/ZK-4243>
- [8] [http://tracker.zkoss.org/browse/ZK-314](https://tracker.zkoss.org/browse/ZK-314)

LayoutRegion

Layout Region

- Demonstration: N/A
- Java API: LayoutRegion ^[1]
- JavaScript API: LayoutRegion ^[2]

Employment/Purpose

This class represents a region in a layout manager.

Example

N/A

Supported events

Name	Event Type
onOpen	Event: OpenEvent ^[3] When a layout is collapsed or opened by a user, the onOpen event is sent to the application.
onSize	Event: SizeEvent ^[4] When a layout is resized by a user, the onSize event is sent to the application.
onSlide	Event: SlideEvent ^[5] When a collapsed layout is滑动 (preview) by a user, the onSlide event is sent to the application.

- Inherited Supported Events: XulElement

Supported Children

*ALL

Version History

Version	Date	Content
8.0.3	2016/04/22	ZK-3166 ^[6] : BorderLayout slide action server-side support

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/LayoutRegion.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/layout/LayoutRegion.html#>
- [3] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/OpenEvent.html#>
- [4] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/SizeEvent.html#>
- [5] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/SlideEvent.html#>
- [6] <http://tracker.zkoss.org/browse/ZK-3166>

NumberInputElement

Number Input Element

- Demonstration: Number Input Element ^[1]
- Java API: NumberInputElement ^[2]
- JavaScript API: N/A

Employement/Purpose

A skeletal implementation for number-type input box.

Example

N/A

Per-component Locale

[since 5.0.8]

You can add a locale per component for all of the NumberInputElement.

For example,

Locale(Correct Result)	Doublebox	Decimalbox	Doublespinner
TW (Taiwan): 2,000.02	2,000.02	2,000.02	2,000.02 ▾
FR (French): 2 000,02	2 000,02	2 000,02	2 000,02 ▾
IT (Italian): 2.000,02	2.000,02	2.000,02	2.000,02 ▾

Change all locales to Taiwan

```
<grid width="550px">
    <columns>
        <column hflex="min" label="Locale(Correct Result)" />
        <column hflex="min" label="Doublebox" />
        <column hflex="min" label="Decimalbox" />
        <column hflex="min" label="Doublespinner" />
    </columns>
    <rows id="rows">
        <row>
            TW (Taiwan): 2,000.02
            <doublebox format="#,###.00" locale="zh_TW"
                value="2000.02" />
            <decimalbox format="#,###.00" locale="zh_TW" value="2000.02"/>
            <doublespinner format="#,###.00" locale="zh_TW" value="2000.02" step="0.5"/>
        </row>
    </rows>
</grid>
```

```

<row>
    FR (French) : 2 000,02
    <doublebox format="#,###.00" locale="fr"
        value="2000.02" />
    <decimalbox format="#,###.00" locale="fr" value="2000.02"/>
    <doublespinner format="#,###.00" locale="fr" value="2000.02" step="0.5"/>
</row>
<row>
    <label pre="true">IT (Italian): 2.000,02</label>
    <doublebox format="#,###.00" locale="it"
        value="2000.02" />
    <decimalbox format="#,###.00" locale="it" value="2000.02"/>
    <doublespinner format="#,###.00" locale="it" value="2000.02" step="0.5"/>
</row>
</rows>
</grid>
<button label="Change all locales to Taiwan">
    <attribute name="onClick"><![CDATA[
        for(Iterator it = rows.getChildren().iterator();
it.hasNext();) {
            for(Iterator itt =
itt.next().getChildren().iterator(); itt.hasNext();) {
                Component c = itt.next();
                if (c instanceof
org.zkoss.zul.impl.NumberInputElement)
                    c.setLocale("zh_TW");
            }
        }
    ]]></attribute>
</button>

```

Supported Events

Name	Inherited From
None	None

- Inherited Supported Events: FormatInputElement

Supported Children

*ALL

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
5.0.8	May 2011	Support locale per component.

References

- [1] <http://www.zkoss.org/zkdemo/userguide/#f2>
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/impl/NumberInputElement.html#>

XulElement

Xul Element

- Demonstration: N/A
- Java API: XulElement ^[1]
- JavaScript API: Widget ^[2]

Employement/Purpose

The fundamental class for XUL elements.

Example

N/A

Supported Events

Name	Inherited From
None	None

- Inherited Supported Events: HtmlBasedComponent

Supported Children

*ALL

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/impl/XulElement.html#>

[2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/Widget.html#>

Containers

This section outlines components which are specifically designed to be able to contain other components.

For introduction, please refer to ZK Developer's Reference: Layouts and Containers.

Caption

Caption

- Demonstration: Groupbox ^[1]
- Java API: Caption ^[2]
- JavaScript API: Caption ^[3]
- Style Guide: Caption

Employement/Purpose

A header for a Groupbox. It may contain either a text label, using LabelElement.setLabel(java.lang.String) ^[4], or child elements for a more complex caption.

Preload Image

since 6.0.0

The feature is applied to all of the LabelImageElement and Image components.

By default the preload function is disabled, so users have to specify the *custom-attributes* and set it to true. For example,

```
<caption image="xxx.png" label="caption">
  <custom-attributes org.zkoss.zul.image.preload="true"/>
</caption>
```

Or specify it just below the root component.

For example,

```
<window>
  <custom-attributes org.zkoss.zul.image.preload="true"/>
  <caption image="xxx.png" label="caption">
    <image src="xxx.png"/>
  </caption>
</window>
```

As you can see, the *custom-attributes* will be checked recursively.

Example

This is a caption

fruits

Apple Orange Banana

```
<zkb>
  <window border="normal" width="350px">
    <caption label="This is a caption"/>
    <groupbox width="300px">
      <caption label="fruits"/>
      <radiogroup onCheck="fruit.value = self.selectedItem.label">
        <radio label="Apple"/>
        <radio label="Orange"/>
        <radio label="Banana"/>
      </radiogroup>
    </groupbox>
  </window>
</zkb>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: LabelImageElement

Supported Children

*ALL

Use Cases

Version	Description	Example Location
5.0	How to use the title and caption inside a Window	Title and Caption

Version History

Version	Date	Content

References

- [1] http://www.zkoss.org/zkdemo/layout/group_box
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Caption.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Caption.html#>
- [4] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/impl/LabelElement.html#setLabel\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/impl/LabelElement.html#setLabel(java.lang.String))

Div

Div

- Demonstration: N/A
- Java API: Div ^[1]
- JavaScript API: Div ^[2]

Employment/Purpose

Div is one of the most lightweight container to group child component for, say, assigning CSS or making more sophisticated layout. It is the same as HTML DIV tag. Div is displayed as block that the following sibling won't be displayed in the same vertical position; as if there is a line break before and after it.

Example



```
<div align="left" width="300px">
    <doublebox />
</div>
<div align="right" width="300px">
    <doublebox />
</div>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

*ALL

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Div.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Div.html#>

Drawer

Drawer

- Demonstration: Introduce a new ZK Addon: Drawer ^[1]
- Java API: Drawer ^[2]
- JavaScript API: Drawer ^[3]
- Available for ZK:
- CE PE EE

Employment/Purpose

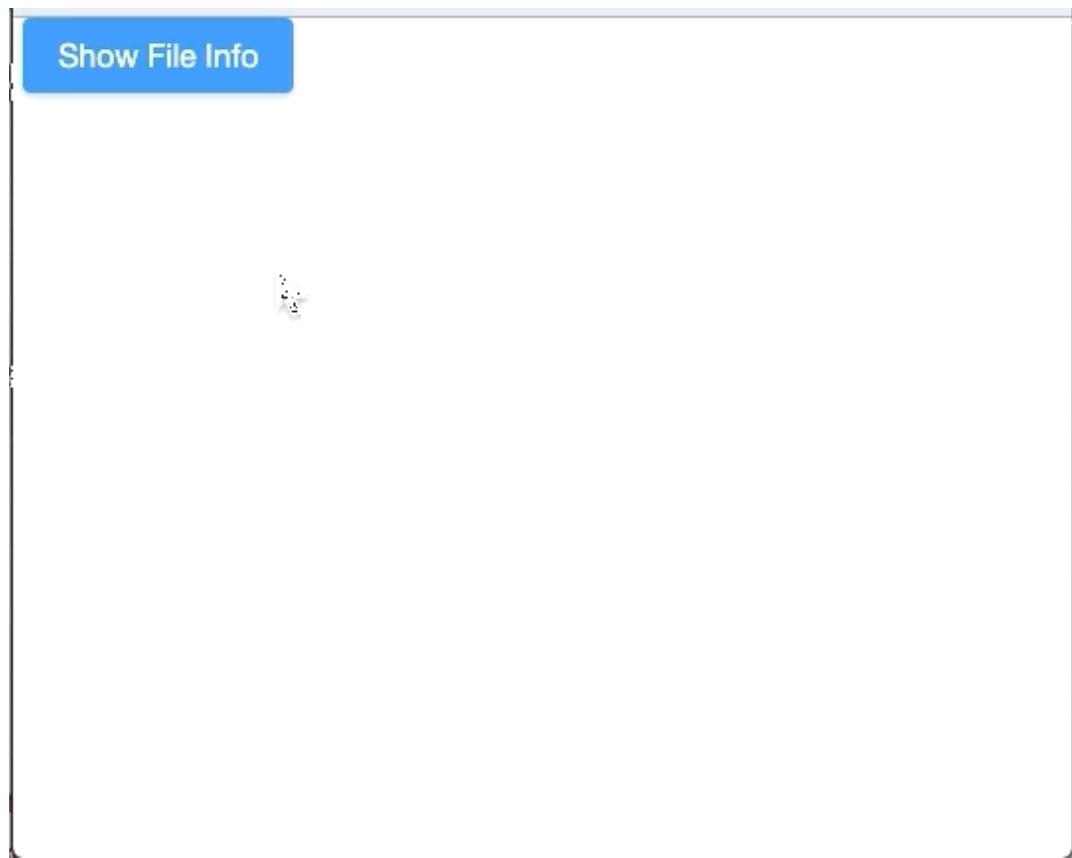
A Drawer is a component that acts as a panel but sticks to the boundary of a web page. With this, you can make the page cleaner and put details into the Drawer for a better user experience.

Example

In this example we dock a drawer on the right side and put detailed file information into the drawer.

The screenshot shows a web application interface. On the left, there is a dark grey sidebar with some placeholder text: "please leave it here." and "sentials</comment>". The main content area has a light blue background and contains the word "Drawer" in blue text. A modal window titled "File information" is open on the right side. It displays three pieces of data in a table-like format:

Name	1.jpg
Size	1.8 Megabytes
Dimensions	1920x1080



```
<button label="Show File Info" onClick="fi.open()"/>
<drawer id="fi" title="File information">
  <grid>
    <columns>
      <column />
      <column />
    </columns>
    <rows>
      <row>
        <label value="Name"/>
        1.jpg
      </row>
      <row>
        <label value="Size"/>
        1.8 Megabytes
      </row>
      <row>
        <label value="Dimensions"/>
        1920x1080
      </row>
    </rows>
  </grid>
</drawer>
```

Open / Close the Drawer

Both `visible` attribute and `open/close` methods allow you to open or close the Drawer.

Data-AnimationSpeed

Article: ZUML Reference/ZUML/Namespaces/Client Attribute/Data-AnimationSpeed

This component respects the `data-animationspeed` attribute.

Properties

Autodrop



When enabled, the drawer will be opened automatically when the mouse cursor is near the page edge.

This feature is not yet supported on mobile devices.

Closable

Sets whether it is closeable by a user (displays the close button). If enabled, there is a button for users to close the drawer.

Note that even if `closable` is false, users can still click outside the drawer to close it.

Mask

Sets whether it is masked when the drawer is opened. By default, there is a translucent dark gray full-screen mask.

Note that even if `mask` is false, users can still click outside the drawer to close it.

Position

Sets the position of the drawer. Valid values are "left", "right", "top" and "bottom".

Title

Sets the title of this drawer. `null` means no title.

Supported Events

Name	Event Type
onOpen	Event: OpenEvent ^[3] Denotes that the user has opened or closed a component. Note: unlike onClose, this event is only a notification. The client sends this event after opening or closing the component.

- Inherited Supported Events: XulElement

Supported Children

*ALL

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
9.0.0	September 2019	ZK-4365 ^[4] : Provide a drawer component

References

- [1] <https://blog.zkoss.org/2019/04/17/introduce-a-new-zk-addon-drawer/>
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Drawer.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/wgt/Drawer.html#>
- [4] <https://tracker.zkoss.org/browse/ZK-4365>

Fragment

Fragment

- Demonstration: N/A
- Java API: Fragment ^[1]
- JavaScript API: Fragment ^[2]
- Style Guide: N/A
- Available for ZK:
- CE PE EE

Purpose

Fragment is a ZK component which developers can combine native HTML elements with ZK data binding syntax to make the static page to be dynamic.

Example

This is a simple example

prop2 error

Submit

```
<zk>
    <fragment viewModel="@id('vm') @init('org.zkoss.fragment.demo.VM2') validationMessages="@id('vmsgs')"
        form="@id('fx') @load(vm) @save(vm, before='submit')
        @validator(vm.formValidator)"
        prop1="@bind(fx.prop1)" prop1err="@bind(vmsgs['fkey1'])"
        prop2="@bind(fx.prop2)" prop2err="@bind(vmsgs['fkey2'])"><! [CDATA[
            <p><input type="text" value="@bind(prop1)"/><span textContent="@load(prop1err)"/></p>
            <p><input type="text" value="@bind(prop2)"/><span textContent="@load(prop2err)"/></p>
            <button onclick="@command('submit')>Submit</button>
        ]]></fragment>
</zk>
```

Data Binding

With Fragment Component, you can bind the properties of ViewModel. For instance, you can use ZK MVVM data binding to access the ViewModel indirectly on the native HTML elements.

These are the supported annotations for now:

- **@save**

```
syntax: @save([limited EL-expression])
```

- **@load**

```
syntax: @load(limited EL-expression)
```

- **@bind**

```
syntax: @bind(limited EL-expression)
```

- **@command**

```
syntax: @command(mybean.myproperty, [arbitraryKey]=[limited EL-expression])
```

- **@global-command**

```
syntax: @global-command(mybean.myproperty, [arbitraryKey]=[limited EL-expression])
```

Limited EL-expression

Inside a fragment, it only supports partially, limited data binding expression including bean dot notation, arithmetic operator, !, and ==

Put HTML as text content

This component also provides a virtual `textContent` attribute for HTML elements to insert data into the tag.

```
<zk>
    <fragment viewModel="@id('vm') @init('org.zkoss.zktest.test2.F85_ZK_3681_Command_VM')"
               status="@bind(vm.status)"><! [CDATA[
        <div>
            <input type="checkbox" onchange="@command('onCheck', checked=event.checked)" />
            Check this checkbox: <span textContent="@load(status)" />
        </div>

        <div>
            <button onclick="@global-command('callGlobal', text='Hello', num=1)">Call global (1)</button>
            <button onclick="@global-command('callGlobal', text='World', num=2)">Call global (2)</button>
        </div>
    ]]></fragment>
</zk>
```

Shadow Elements

In this example, we use `if` and `forEach` tags together for condition and collection rendering.

```
<zk>
    <fragment viewModel="@id('vm') @init('org.zkoss.zktest.test2.F85_ZK_3681_Shadow_VM')"
        issues="@bind(vm.issues)"><! [CDATA[
            <section>
                <h1>My Issue List</h1>
                <ul>
                    <forEach items="@load(issues)">
                        <!-- There's a pre-defined variable "each" for convenience. -->
                        <li>
                            <!-- @bind(each) is wrong because each is just a temp variable in loops. -->
                            <input type="checkbox" checked="@load(each.isDone)" />
                            <if test="@load(each.isDone)">
                                <strike>[<span textContent="@load(each.id)" />]
                                <span textContent="@load(each.description)" /></strike>
                            </if>
                            <!-- No else for now. -->
                            <if test="@load(!each.isDone)">
                                [<span textContent="@load(each.id)" />]
                                <span textContent="@load(each.description)" />
                            </if>
                        </li>
                    </forEach>
                </ul>
            <section>
        ]]></fragment>
</zk>
```

- For further details, please refer to Shadow components ^[3] directly.

Data Validation

Server-side Property/Form Validation

To ensure data is correct and useful, we can leverage ZK's validators.

```
<zk>
    <fragment viewModel="@id('vm') @init('org.zkoss.fragment.demo.VM1')"
        validationMessages="@id('vmsgs')"
        prop1="@bind(vm.prop1) @validator(vm.validator1)"
        prop1err="@bind(vmsgs['prop1'])"><! [CDATA[
            <input type="text" value="@bind(prop1)" />
            <span textContent="@load(prop1err)" />
        ]]></fragment>
</zk>
```

You can get the invalid message by assigning a self-defined key as an alias. In order to access invalidate messages by HTML elements, you can simply bind the messages onto Fragment properties.

Here we can use form-binding and form validators to validate all the fields.

```
<zk>
<fragment viewModel="@id('vm') @init('foo.BarVM')" validationMessages="@id('vmsgs')"
    form="@id('fx') @load(vm.currentUser) @save(vm.currentUser,
before='submit') @validator('formBeanValidator', prefix='p_')"
    name="@bind(fx.name)" nameerror="@bind(vmsgs['p_name'])"><! [CDATA[
<input type="text" value="@bind(name)"/><span textContent="@load(nameerror)"/>
<button onclick="@command('submit')">Submit</button>
]]></fragment>
</zk>
```

Client-side Property Validation

This component also provides a new `@jsvalidator` running at client side, accepting custom JavaScript functions for validation. The benefit is that there is no need to send requests to the server for each validation. However, since the validation logic will be exposed at client side, some simple check, such as empty checking or range checking, is recommended. The usage is like `@validator` but it is effective only when applying HTML elements.

`@jsvalidator`

syntax: `@jsvalidator(validation_function_name)`

The following is the definition of custom JavaScript function.

```
ValidationFunction(val, vmsgs)
* val: The input data.
* vmsgs:
    The validation message holder object. You can add a invalidate message by adding a new property.
    If you want to clear the specific message, assign an empty string to the property.
* Returns: Boolean. True if the data is valid.
```

You can use an implicit object (`vmsgs`) to get the client-side invalid messages. The `@jsvalidator` has its own validation message holder not shared with server-side.

```
<zk>
<fragment viewModel="@id('vm') @init('foo.BarVM')" someprop="@bind(vm.prop1)"><! [CDATA[
<input type="text" value="@bind(someprop) @jsvalidator('validateExample')"/>
<span textContent="@load(vmsgs['foo'])"/>
<script type="text/javascript">
function validateExample(val, vmsgs) {
    var isValid = someValidationProcess(val);
    vmsgs['foo'] = isValid ? '' : 'Invalid value';
    return isValid;
}
</script>
]]></fragment>
</zk>
```

The Differences Between @validator and @jsvalidator

Catalogue	@validator	@jsvalidator
Validate at	Server side	Client side
ZK form validation	Supported	Not supported
Validation message holder	Initialized in validationMessages	An implicit vmsgs object

1. `@validator` relies on the server, while `@jsvalidator` relies on the browser.

2. `@jsvalidator` does not support form validation.

3. The validation message holders are not the same.

For security concerns, we recommend you to use server-side `@validator` in most cases and choose client-side `@jsvalidator` if the validation needs an instant feedback such as password strength, number range, and so on.

Event Handling

The command of ViewModel can be invoked by attaching DOM events with `@command` or `@global-command` on HTML elements. Once the DOM event is triggered (i.g. clicked or changed), the command of ViewModel will be executed and receive the corresponding event object.

You can get more details from the event object such as mouse cursor position, pressed keys, entered text, and selected text.

ZK Event object	DOM event
MouseEvent [4]	onclick oncontextmenu ondbleclick onmousedown onmouseenter onmouseleave onmouseover onmouseout onmouseup ondrag
KeyEvent [6]	onkeydown onkeypress onkeyup
InputEvent [4]	onchange oninput
CheckEvent [5]	onchange (checkbox) oninput (checkbox)
SelectionEvent [6]	onselect
DropEvent [3]	ondrop

Event [7]	onblur
	onfocus
	onfocusin
	onfocusout

- For further details about how to retrieve the event object, please refer to Retrieve Event Object [8] directly.

Properties

- content**: specify the content of this component.
- src**: specify the URI of an external content file. The file encoding is assumed to be UTF-8.
- recoverId**: specify the recover ID.

Supported Events

Name	Event Type
onRecover	Event: Event [7] Represents an event sent back to the server caused by an offline recovery.

Supported Children

None

Use Cases

Version	Description	Example Location
8.5+	Data binding, Shadow elements	Client Binding with ZK MVVM for your eyes only [9]
8.5+	Data validation, Event handling	Advanced Usage of Fragment Component [10]

Version History

Version	Date	Content
8.5	2017/09/21	Add the new Fragment component

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Fragment.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/wgt/Fragment.html#>
- [3] http://books.zkoss.org/zkessentials-book/master/shadow_components/index.html
- [4] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/InputEvent.html#>
- [5] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/CheckEvent.html#>
- [6] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/SelectionEvent.html#>
- [7] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/Event.html#>
- [8] <http://books.zkoss.org/zk-mvvm-book/8.0/advanced/parameters.html#retrieve-event-object>

- [9] <http://blog.zkoss.org/2016/11/15/client-binding-with-zk-mvvm-for-your-eyes-only/>
 [10] https://www.zkoss.org/wiki/Small_Talks/2017/July/Advanced_Usage_of_Fragment_Component

Groupbox

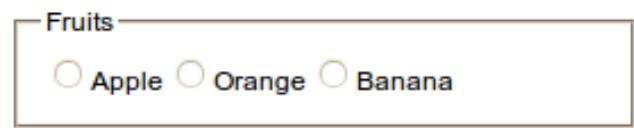
Groupbox

- Demonstration: Groupbox ^[1]
- Java API: Groupbox ^[1]
- JavaScript API: Groupbox ^[2]
- Style Guide: Groupbox

Employment/Purpose

A group box is used to group components together. A border is typically drawn around the components to show that they are related. The label across the top of the group box can be created by using Caption component. It works much like the HTML legend element. Unlike Window, a group box is not an owner of the ID space. It cannot be overlapped or popup.

Example



```
<groupbox width="250px">
  <caption label="Fruits"/>
  <radiogroup>
    <radio label="Apple"/>
    <radio label="Orange"/>
    <radio label="Banana"/>
  </radiogroup>
</groupbox>
```

Java Example

```
Groupbox gb = new Groupbox();

new Caption("Here is Caption").setParent(gb);

gb.setMold("3d");
gb.setWidth("200px");
gb.appendChild(new Label("Here is Content"));

// register an onOpen event.
gb.addEventListener(Events.ON_OPEN, new EventListener() {
  public void onEvent(Event event) throws Exception {
```

```
    if ( ((OpenEvent)event).isOpen() )
        //do something you want.
    }
} );
gb.setParent(outer);
```

Properties

ContentStyle

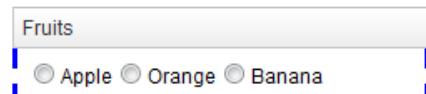
Specify the CSS style for the content block of the groupbox.



```
<groupbox width="250px" mold="3d"
    contentStyle="border: 3px blue dashed; border-top:0px">
    <caption label="Fruits"/>
    <radiogroup>
        <radio label="Apple"/>
        <radio label="Orange"/>
        <radio label="Banana"/>
    </radiogroup>
</groupbox>
```

ContentSclass

Specify the CSS class for the content block of the groupbox.



```
<zk>
<style>
.mygroupbox-cnt {
    border: 3px blue dashed;
    border-top:0px
}
</style>
<groupbox width="250px" mold="3d"
    contentSclass="mygroupbox-cnt">
    <caption label="Fruits"/>
    <radiogroup>
        <radio label="Apple"/>
        <radio label="Orange"/>
        <radio label="Banana"/>
    </radiogroup>
</groupbox>
</zk>
```

Closable

Default: **true**

Specify the groupbox whether can be collapsed or not.

For example,

```
<groupbox width="250px" mold="3d" closable="true">
```

Note: the function can only be applied when the Caption exists.

Open/Close

Default: **true**

Specify the display of the groupbox whether is open or close. For example,

```
<groupbox width="250px" mold="3d" open="false">
```

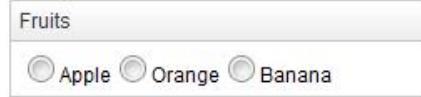
Note: the false means the groupbox is closed, i.e. no content can appear.

Limitation of the Default Mold

The default mold of groupbox uses HTML FIELDSET to represent a groupbox visually. It is efficient, but it has some limitations:

1. The look might be different from one browser to another
2. The real width and height might not be exactly the same as the specified value in some browsers, such as Firefox.

If it is an issue, you could use the 3d mold instead.



```
<groupbox width="250px" mold="3d">
    <caption label="Fruits"/>
    <radiogroup>
        <radio label="Apple"/>
        <radio label="Orange"/>
        <radio label="Banana"/>
    </radiogroup>
</groupbox>
```

[since 6.0.0]

The default mold use the same method of 3d mold to represent a groupbox, the limitation is gone.

Configure to Use the 3d Mold as Default

If you prefer to use the 3d mold as default, you could configure ZK by adding the following to /WEB-INF/zk.xml

```
<library-property>
  <name>org.zkoss.zul.Groupbox.mold</name>
  <value>3d</value>
</library-property>
```

Supported Events

Name	Event Type
onOpen	Event: OpenEvent [3] Denotes user has opened or closed a component. Note: unlike onClose, this event is only a notification. The client sends this event after opening or closing the component. It is useful to implement load-on-demand by listening to the onOpen event, and creating components when the first time the component is opened.

- Inherited Supported Events: XulElement

Supported Molds

Available molds of a component are defined in lang.xml embedded in zul.jar.

Name	Snapshot
default	
3d	

Supported Children

* ALL

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Groupbox.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Groupbox.html#>

Idspace

Idspace

- Demonstration: N/A
- Java API: Idspace ^[1]
- JavaScript API: Idspace ^[2]

Employment/Purpose

`Idspace` just like a Div but implements the ID space, all descendant components of Idspace (including the Idspace itself) form an independent ID space. Thus, you could use a idspace as the topmost component to group components. This way developers only need to maintain the uniqueness of each subset separately.

since 8.0.3

To group components without render a Div, `Idspace` provides "nodom" mold. It would render no-dom widget in client-side. It only render comment nodes for positioning.

Notice that it's not recommended to use hflex/vflex in the children of nodom element.

Example



```
<idspace>
    <window border="normal">
        <button id="btn" label="button" />
    </window>
    <div>
        <button id="btn" label="button" />
    </div>
</idspace>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

*ALL

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Idspace.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Idspace.html#>

Inputgroup

Inputgroup

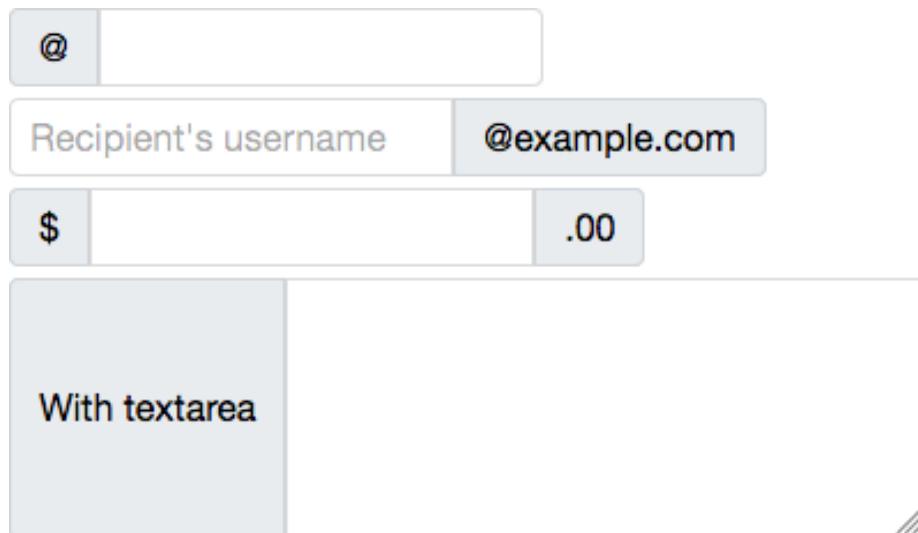
- Demonstration: Inputgroup – organize your input components ^[1]
- Java API: Inputgroup ^[2]
- JavaScript API: Inputgroup ^[3]
- Style Guide: N/A

[since 9.0.0]

Employment/Purpose

Inputgroup can be used for prepending or appending some components to the input component, merging them like a new form-input component.

Example



```
<zk>
    <inputgroup>
        @<textbox />
    </inputgroup>

    <inputgroup>
        <textbox placeholder="Recipient's username"/>@example.com
    </inputgroup>

    <inputgroup>
        $<textbox/>.00
    </inputgroup>

    <inputgroup>
        With textarea
    </inputgroup>
```

```
<textbox multiline="true" rows="5" cols="30"/>
</inputgroup>
</zk>
```

Properties

Orient

Specify the orientation.

A screenshot of a web application interface. It features a light gray header bar with the text "First and last name". Below this is a large, empty rectangular input field. At the bottom of the page, there is a blue sidebar containing four white rectangular buttons, each labeled "Button".

```
<inputgroup orient="vertical">
  First and last name<textbox/><textbox/>
</inputgroup>
```

```
<inputgroup orient="vertical">
  <button label="Button"/>
  <button label="Button"/>
  <button label="Button"/>
  <button label="Button"/>
</inputgroup>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

- * Label
- * InputElement
- * LabelImageElement

Version History

Version	Date	Content
9.0.0	October 2019	ZK-4391 [4]: Provide an inputgroup component

References

- [1] <https://blog.zkoss.org/2019/08/16/zk-9-preview-inputgroup-organize-your-input-components/>
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Inputgroup.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Inputgroup.html#>
- [4] <https://tracker.zkoss.org/browse/ZK-4391>

Nodom

Nodom

- Demonstration: N/A

Employment/Purpose

`<nodom>` is a ZK Component but only has server-side Java object and doesn't render any DOM elements and JavaScript widget at the client-side. It only renders comment nodes for positioning. Thus, if you want to control a group of components without unnecessary DOM elements, you can use a `<nodom>` as the outermost component to group components under a controller (composer/ViewModel) instead of a `Window` or a `Div`.

Limitation

`<nodom>` does not support using `hflex/vflex` in itself and its children component.

Example



```
<nodom viewModel="@id('vm')@init('foo.MyViewModel')">
    <window border="normal">
        <button id="btn" label="@init(vm.label)" />
    </window>
    <div>
        <button id="btn" label="@init(vm.label)" />
    </div>
</nodom>
```

Supported Events

Name	Event Type
None	None

Supported Children

*ALL

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content
8.0.3	2016/09/21	Add the new Nodom component

Panel

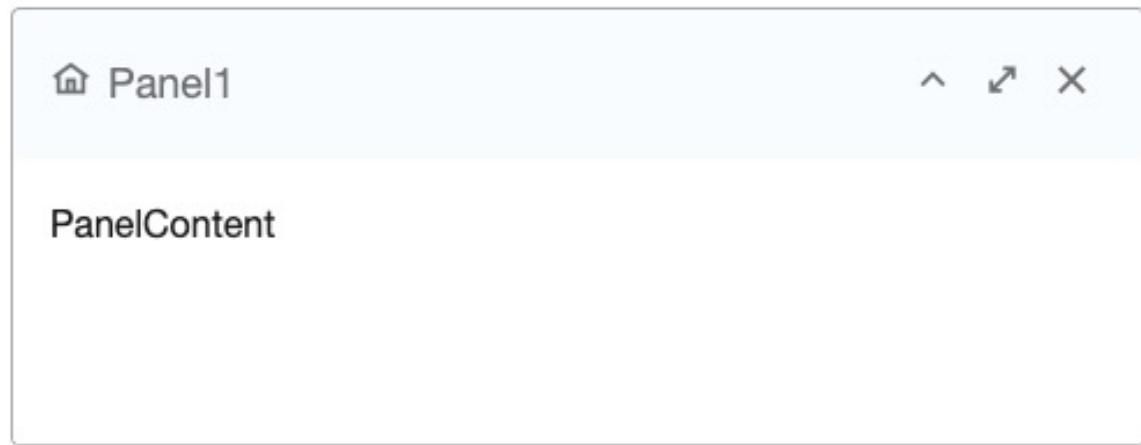
Panel

- Demonstration: Panel ^[1]
- Java API: Panel ^[2]
- JavaScript API: Panel ^[3]
- Style Guide: Panel

Employment/Purpose

Panel is a container that has specific functionality and structural components that make it the perfect building block for application-oriented user interfaces. The Panel contains bottom, top, and foot toolbars, along with separate header, footer and body sections. It also provides built-in collapsible, closable, maximizable, and minimizable behavior, along with a variety of pre-built tool buttons that can be wired up to provide other customized behavior. Panels can be easily embedded into any kind of ZUL component that is allowed to have children or layout component. Panels also provide specific features like float and move. Unlike Window, Panels can only be floated and moved inside its parent node, which is not using zk.setVParent() function at client side. In other words, if Panel's parent node is an relative position, the floated panel is only inside its parent, not the whole page. The second difference of Window is that Panel is not an independent ID space (by implementing IdSpace), so the ID of each child can be used throughout the panel.

Example



```
<panel height="20%" style="margin-bottom:10px"
       title="Panel1" border="normal" maximizable="true"
closable="true"
       collapsible="true">
  <caption iconSclass="z-icon-home"/>
  <panelchildren>PanelContent</panelchildren>
</panel>
```

Java Example

```
Panel panel = new Panel();
panel.setTitle("Here is Title");
panel.setBorder("normal");
panel.setFramable(true);

Panelchildren pc = new Panelchildren();
pc.setParent(panel);
pc.appendChild(new Label("Here is Content"));
```

Properties

Sizable

The panel can now be resized as long as the attribute sizable is set to true. The example ZUL below shows a panel which can be resized and the image displays a panel which is in the process of being resized.

```
<panel sizable="true" id="panel" framable="true" width="500px" height="400px"
       title="Panel"
       maximizable="true" minimizable="true" border="normal"
       collapsible="true" closable="true">
  <panelchildren>
    <textbox width="100%" height="100%" />
  </panelchildren>
</panel>
```

[Since 5.0.0]

Draggable

When used with Portallayout, the draggable property (HtmlBasedComponent.setDraggable(java.lang.String) [4]) can be used to control whether the panel is draggable under the portal layout.

```
<portallayout>
    <portalchildren style="padding: 5px" width="30%">
        <panel height="150px" title="Google Tools" border="normal"
            collapsible="true" closable="true" maximizable="true"
            style="margin-bottom:10px">
            <panelchildren>

                </panelchildren>
            </panel>
        <panel height="300px" title="LabPixies Clock" border="normal"
            collapsible="true" closable="true" maximizable="true"
            style="margin-bottom:10px"
            draggable="false">
            <panelchildren>
                This is not draggable.
            </panelchildren>
        </panel>
    </portalchildren>
</portallayout>
```

[Since 5.0.3]

Border

It specifies whether to display the border. Currently, it supports none, normal, rounded and rounded+. The default is none, i.e., no border.

Here is the effect with different borders:



Backward Compatibility: ZK 5.0.5 and prior shall use the combination of the border and framable property as follows.

Border in 5.0.6	The equivalent combination in 5.0.5 and prior	Description
border="none"	border="none"	framable is default to false
border="normal"	border="normal"	framable is default to false
border="rounded"	framable="true"	border is default to none
border="rounded+"	border="normal" framable="true"	

- Notice that the use of the border and framable combination still works in 5.0.6 (backward compatible).

Title

Besides this attribute, you could use Caption to define a more sophisticated caption (aka., title). If the panel has a caption whose label `Caption.getLabel()`^[5] is not empty, then this attribute is ignored. (Default: empty).

Closable

Specify the panel whether to show a close button on the title bar or not. If closable, a button is displayed and the `onClose` event (OpenEvent^[3]) is sent if a user clicks the button. (Default: false)

Miscellaneous

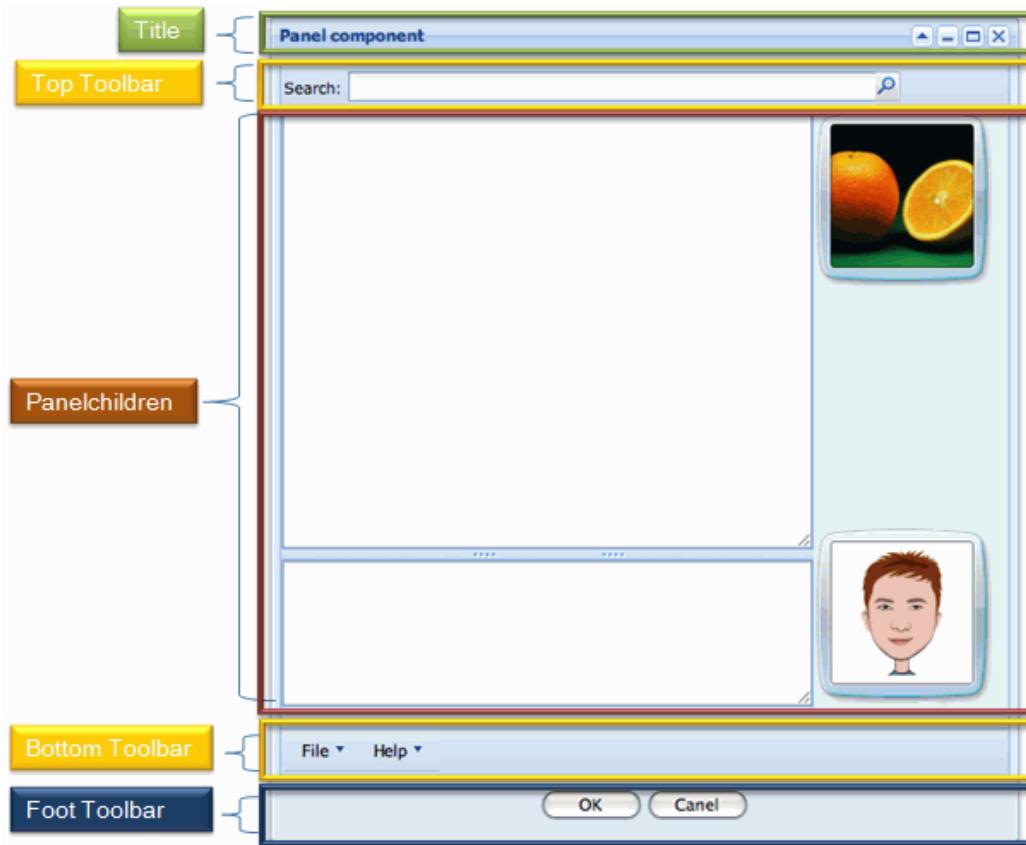
Scollable Panel

To make the scrollbar appear when content exceeds panel height, specify `style="overflow: auto"` on `Panelchildren`.

```
<panel height="200px">
    <panelchildren style="overflow: auto">
        <div style="background: #999966" height="195px" />
        <div style="background: #669999">Div Content</div>
    </panelchildren>
</panel>
```

Toolbar positions

Panel supports three kinds of Toolbar [6] positions: Top, Bottom and Foot. For example:



```
<panel id="panel" framable="true" width="500px" height="550px"
    title="Panel component" floatable="true" movable="true"
    maximizable="true" minimizable="true" border="normal"
    collapsible="true" closable="true">
    <toolbar>
        ... // Top Toolbar of the panel
    </toolbar>
    <panelchildren>
        ... // Each added child will show into the body content of
the panel
    </panelchildren>
    <toolbar>
        ... // Bottom Toolbar of the panel
    </toolbar>
    <toolbar>
        ... // Foot Toolbar of the panel
    </toolbar>
</panel>
```

- Top Toolbar (Line 5): It is used to show the top toolbar close to the body content of the panel. (It is an option)
- Bottom Toolbar (Line 11): It is used to show the bottom toolbar close to the body content of the panel. (It is an option)

- Foot Toolbar (Line 14): It is used to show the operating button under the body content with a few padding. (It is an option)

Please refer Small_Talks/2008/July/Using_Panel_to_Lay_out_Your_Website for details.

Supported Events

Name	Event Type
onMove	Event: MoveEvent [5] Denotes the position of the window is moved by a user.
onOpen	Event: OpenEvent [3] Denotes user has opened or closed a component. Note: Unlike <code>onClose</code> , this event is only a notification. The client sends this event after opening or closing the component. It is useful to implement load-on-demand by listening to the <code>onOpen</code> event, and creating components when the first time the component is opened.
onMaximize	Event: MaximizeEvent [7] Denotes user has maximize a component.
onMinimize	Event: MinimizeEvent [8] Denotes user has minimize a component.
onClose	Event: OpenEvent [3] Denotes the close button is pressed by a user, and the component shall detach itself.
onSize	Event: SizeEvent [4] Denotes the panel's size is updated by a user.
onZIndex	Event: ZIndexEvent [9] Denotes the panel's zindex is updated by a user.

- Inherited Supported Events: XulElement

Supported Children

* Panelchildren

Use Cases

Version	Description	Example Location
5.0	Portallayout, panels and events	[10]

Version History

Version	Date	Content
5.0.3	July, 2010	The draggable property can be used to control the drag-ability in a portal layout.
5.0.6	January, 2010	The framable property was deprecated. Please refer to #Border for details.

References

- [1] <http://www.zkoss.org/zkdemo/window/panel>
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Panel.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wnd/Panel.html#>
- [4] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/HtmlBasedComponent.html#setDraggable\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/HtmlBasedComponent.html#setDraggable(java.lang.String))
- [5] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Caption.html#getLabel\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Caption.html#getLabel())
- [6] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Toolbar.html#>
- [7] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/MaximizeEvent.html#>
- [8] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/MinimizeEvent.html#>
- [9] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/ZIndexEvent.html#>
- [10] <http://www.zkoss.org/forum/listComment/10765>

Panelchildren

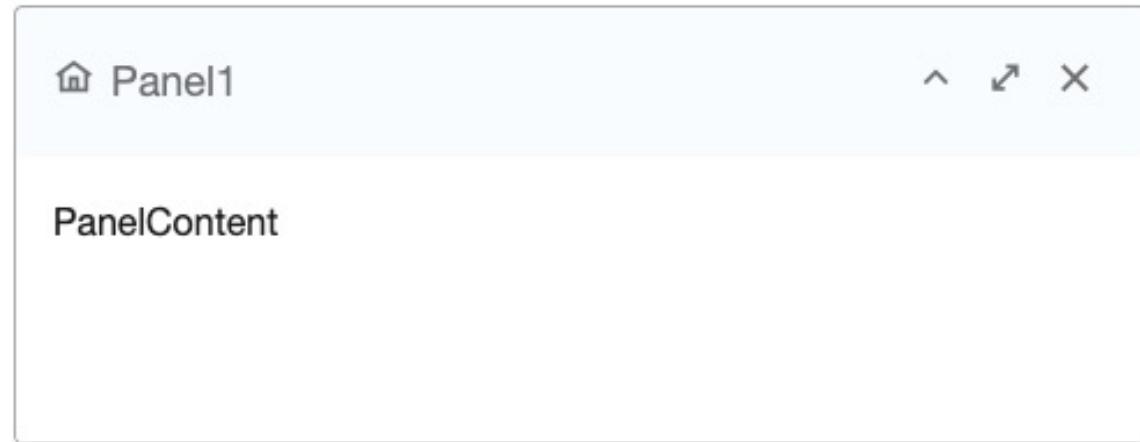
Panel Children

- Demonstration: Panel ^[1]
- Java API: Panelchildren ^[1]
- JavaScript API: Panelchildren ^[2]
- Style Guide: Panel

Employment/Purpose

Panelchildren is used for Panel component to manage each child who will be shown in the body of Panel. Note that the size of Panelchildren is automatically calculated by Panel so both setWidth(String) and setHeight(String) are read-only.

Example



```
<panel height="100px" width="200px" style="margin-bottom:10px"
       title="Panel1" border="normal" maximizable="true"
       collapsible="true">
    <panelchildren>PanelContent1</panelchildren>
</panel>
<panel height="100px" width="200px" framable="true" title="Panel2"
       border="normal" maximizable="true" style="margin-bottom:10px">
    <panelchildren>PanelContent2</panelchildren>
</panel>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

*ALL

Use Cases

Panel

Version History

Version	Date	Content

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Panelchildren.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wnd/Panelchildren.html#>

Span

Span

- Demonstration: N/A
- Java API: Span [1]
- JavaScript API: Span [2]

Employment/Purpose

Span is one of the most lightweight container to group child component for, say, assigning CSS or making more sophisticated layout. It is the same as HTML SPAN tag. Span is displayed inline with other siblings; as if there is *no* line breaks between them.

Example

Name:

```
<span>
    Name :
    <textbox />
</span>
```

Supported Events

Name	Event Type

- Inherited Supported Events: XulElement

Supported Children

*ALL

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Span.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Span.html#>

Tabbox

Tabbox

- Demonstration: Tabbox ^[1]
- Java API: Tabbox ^[2]
- JavaScript API: Tabbox ^[3]
- Style Guide: Tabbox

Employment/Purpose

A tabbox is a container used to display a set of tabbed groups of components. A row of tabs is displayed at the top (or left or other location) of tabbox which may be used to switch between each group. It allows developers to separate a large number of components into several groups (each group is contained in atabpanel). Only one group is visible at the time, such that the user interface won't be too complicate to read. Once the tab of an invisible group is clicked, it becomes visible and the previous visible group becomes invisible.

The visible group is called *selected*, which can be retrieved by use of Tabbox.getSelectedPanel() ^[4] or Tabbox.getSelectedIndex() ^[5].

Example

The screenshot shows a Tabbox component with three tabs labeled "Tab 1", "Tab 2", and "Tab 3". The "Tab 1" tab is currently selected, and its correspondingtabpanel contains the text "This is panel 1". The "Tab 3" tab is also visible below the first two, indicating it is another tab in the same group.

```
<zk>
    <tabbox width="400px">
        <tabs>
            <tab label="Tab 1" />
            <tab label="Tab 2" />
        </tabs>
        <tabpanel>
```

```

        <tabpanel>This is panel 1</tabpanel>
        <tabpanel>This is panel 2</tabpanel>
    </tabpanel>
</tabbox>
<space />
<tabbox width="400px" mold="accordion">
    <tabs>
        <tab label="Tab 3" />
        <tab label="Tab 4" />
    </tabs>
    <tabpanel>
        <tabpanel>This is panel 3</tabpanel>
        <tabpanel>This is panel 4</tabpanel>
    </tabpanel>
</tabbox>
</zk>

```

Properties and Features

Toolbar in Tabbox

The Tabbox supports the inclusion of other controls within its tab bar, thus allowing more freedom and layout options when creating layouts which include a toolbar. The screenshot below demonstrates an example Tabbox which includes extra controls in the tab bar acting like a menu system.

Note: Toolbar in Tabbox only works in a horizontal(top/bottom) orient Tabbox.



```

<tabbox width="250px">
    <tabs>
        <tab label="Tab 1" closable="true" />
        <tab label="Tab 2" closable="true" />
        <tab label="Tab 3" closable="true" />
        <tab label="Tab 4" closable="true" />
        <tab label="Tab 5" closable="true" />
    </tabs>
    <toolbar>
        <toolbarbutton image="/img/live.gif" onClick='alert("Live")' />
        <toolbarbutton image="/img/defender.gif"
            onClick='alert("Defender")' />
        <toolbarbutton image="/img/battery.gif"
            onClick='alert("Battery")' />
    </toolbar>
    <tabpanel>
        <tabpanel>This is panel 1</tabpanel>
        <tabpanel>This is panel 2 The second panel</tabpanel>
        <tabpanel>This is panel 3</tabpanel>
    </tabpanel>

```

```
<tabpanel>This is panel 4</tabpanel>
<tabpanel>This is panel 5</tabpanel>
</tabpanelles>
</tabbox>
```

MaximalHeight

since 7.0.0

In order to solve the problem where each tabpanel have different heights, we offer this feature called **MaximalHeight**. With this feature, every Tabpanel will be applied the maximum height among all thetabpanelles i.e. if onetabpanel's height is at 300px while the rest is at 240px, all of thetabpanelles will be applied with a height of 300px. This feature only works in the initial phase. The screenshot below demonstrates an example Tabbox which includes 3tabpanelles and all of them use the maximum height.

Note: The Client ROD feature will be disabled if it is set to true.



```
<tabbox maxHeight="true" width="300px">
  <tabs id="tabs0">
    <tab label="Tab1" />
    <tab label="Tab2" />
    <tab label="Tab3" />
  </tabs>
  <tabpanelles id="pnls0">
    <tabpanel>
      <div>Tabpanel Content 1</div>
      <div>Tabpanel Content 1</div>
      <div>Tabpanel Content 1</div>
    </tabpanel>
    <tabpanel>
      <div>Tabpanel Content 2</div>
      <div>Tabpanel Content 2</div>
    </tabpanel>
    <tabpanel>
      <div>Tabpanel Content 3</div>
      <div>Tabpanel Content 3</div>
      <div>Tabpanel Content 3</div>
      <div>Tabpanel Content 3</div>
    </tabpanel>
  </tabpanelles>
</tabbox>
```

Supported Events

Name	Event Type
onSelect	Event: SelectEvent [6] Denotes user has selected a tab. onSelect is sent to both tab and tabbox.

- Inherited Supported Events: XulElement

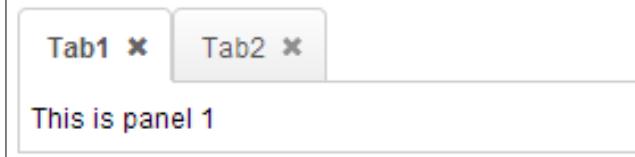
Supported Molds

Available molds of a component are defined in lang.xml embedded in zul.jar.

Name	Snapshot
default	 This is panel 1
accordion	 This is panel 1 Tab 2
accordion-lite	 Tab 1 This is panel 1 Tab 2

- accordion-lite mold is deprecated since 7.0.0

Supported Orients

Name	Snapshot
top	 This is panel 1
left	
right	



- Rename orient "horizontal" to "top", "vertical" to "left" and add extra two orients named "bottom" and "right" since 7.0.0

Supported Children

* Tabs, Tabpanels, Toolbar

Use Cases

Version	Description	Example Location
5.0	Tabbox can be used to display information on separate panels and show only one panel at a time.	[7]

Version History

Version	Date	Content

References

- [1] <http://www.zkoss.org/zkdemo/tabbox>
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Tabbox.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/tab/Tabbox.html#>
- [4] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Tabbox.html#getSelectedPanel\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Tabbox.html#getSelectedPanel())
- [5] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Tabbox.html#getSelectedIndex\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Tabbox.html#getSelectedIndex())
- [6] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event>SelectEvent.html#>
- [7] <http://www.zkoss.org/zkdemo/userguide/#l10>

Tab

Tab

- Demonstration: Tabbox ^[1]
- Java API: Tab ^[1]
- JavaScript API: Tab ^[2]
- Style Guide: Tabbox

Employment/Purpose

A specific tab. Clicking on the tab brings the tab panel to the front. You could put a label and an image on it by label and image properties.

Example



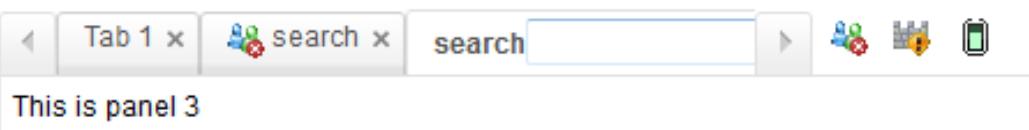
```
<tabbox width="400px">
    <tabs>
        <tab label="Tab 1" image="/img/folder.gif" />
        <tab label="Tab 2" image="/img/folder.gif" closable="true" />
    </tabs>
    <tabpanels>
        <tabpanel>This is panel 1</tabpanel>
        <tabpanel>This is panel 2</tabpanel>
    </tabpanels>
</tabbox>
```

Properties and Features

Caption

A tab might have a caption, which is specified by declaring a child component called caption.

[ZK EE]
[Since 6.5.0]



```
<tabbox width="400px">
    <tabs>
        <tab label="Tab 1" image="/img/folder.gif" />
```

```
<tab label="Tab 2" image="/img/folder.gif" closable="true" />
<tab>
    <caption hflex="min" label="search">
        <textbox />
    </caption>
</tab>
</tabs>
<tabpanels>
    <tabpanel>This is panel 1</tabpanel>
    <tabpanel>This is panel 2</tabpanel>
    <tabpanel>This is panel 3</tabpanel>
</tabpanels>
</tabbox>
```

Closable

By setting the `closable` property to true, a close button is shown on a tab, such that a user could close the tab and the corresponding tab panel by clicking the button. Once a user clicks on the close button, an `onClose` event is sent to the tab. It is processed by the `onClose` method of Tab. Then, `onClose`, by default, detaches the tab itself and the corresponding tab panel.

Dynamically-created Tab

since 7.0.0

If you assign a model to a Tabbox, it will do nothing for an `onClose` event. Therefore, developers have to listen an `onClose` event to remove the corresponding item in the model instead of Tab itself.

When using `model`, Tabs are dynamically created, so you **can't just listen to `onClose` event** like `@Listen("onClose = tab")` because a Tab is not created when wiring a listener. You can forward `onClose` event to its parent like:

```
<tabbox id="tabbox">
    <template name="model:tab">
        <tab closable="true" forward="onClose=tabbox.onTabClose(${each})"/>
    </template>
```

Supported Events

Name	Event Type
onSelect	Event: SelectEvent [6] Denotes user has selected a tab. onSelect is sent to both tab and tabbox.
onClose	Event: Event [3] Denotes the close button is pressed by a user, and the component shall detach itself.

- Inherited Supported Events: LabelImageElement

Supported Children

*NONE

Use Cases

Tabbox

Version History

Version	Date	Content
6.5.0	June, 2012	ZK-970 [4]: The Tab component support caption component as it's label

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Tab.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/tab/Tab.html#>
- [3] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/ui/zk/ui/event/Event.html#>
- [4] <http://tracker.zkoss.org/browse/ZK-970>

Tabs

-
- ## Tabs
- Demonstration: Tabbox ^[1]
 - Java API: Tabs ^[1]
 - JavaScript API: Tabs ^[2]
 - Style Guide: Tabbox

Employment/Purpose

A `tabs` is the container for the `tab` components.

Example



The screenshot displays a tabs component with four tabs labeled "Tab 1", "Tab 2", "Tab 3", and "Tab 4". "Tab 1" is the active tab, showing its corresponding panel content: "This is panel 1". "Tab 3" and "Tab 4" are also visible, each showing their respective panel content: "This is panel 3" and "This is panel 4".

```
<zk>
    <tabbox width="400px">
        <tabs>
            <tab label="Tab 1" />
            <tab label="Tab 2" />
        </tabs>
        <tabpanels>
            <tabpanel>This is panel 1</tabpanel>
            <tabpanel>This is panel 2</tabpanel>
        </tabpanels>
    </tabbox>
    <space />
    <tabbox width="400px" mold="accordion">
        <tabs>
            <tab label="Tab 3" />
            <tab label="Tab 4" />
        </tabs>
        <tabpanels>
            <tabpanel>This is panel 3</tabpanel>
            <tabpanel>This is panel 4</tabpanel>
        </tabpanels>
    </tabbox>
```

```
</zk>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

Tab

Use Cases

Tabbox

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Tabs.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/tab/Tabs.html#>

Tabpanel

Tabpanel

- Demonstration: Tabbox ^[1]
- Java API: Tabpanel ^[1]
- JavaScript API: Tabpanel ^[2]
- Style Guide: Tabbox

Employment/Purpose

A `tabpanel` is the body of a single tab panel. You would place the content for a group of components within a tab panel. The first `tabpanel` corresponds to the first `tab`, the second `tabpanel` corresponds to the second `tab` and so on.

Example

The screenshot shows a tab panel with four tabs. The first tab, "Tab 1", is active and displays the content "This is panel 1". The other three tabs, "Tab 2", "Tab 3", and "Tab 4", are inactive and show their respective labels. The tabs are arranged horizontally at the top of the panel.

```
<zk>
    <tabbox width="400px">
        <tabs>
            <tab label="Tab 1" />
            <tab label="Tab 2" />
        </tabs>
        <tabpanels>
            <tabpanel>This is panel 1</tabpanel>
            <tabpanel>This is panel 2</tabpanel>
        </tabpanels>
    </tabbox>
    <space />
    <tabbox width="400px" mold="accordion">
        <tabs>
            <tab label="Tab 3" />
            <tab label="Tab 4" />
        </tabs>
        <tabpanels>
            <tabpanel>This is panel 3</tabpanel>
            <tabpanel>This is panel 4</tabpanel>
        </tabpanels>
    </tabbox>

```

```
</tabpanels>
</tabbox>
</zk>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

* ALL

Use Cases

Version	Description	Example Location
3.6	How to put a scrollbar inside a tabpanel	[3] [3]
3.6	How to make a tabpanel loaded on demand	[4] [4]

See also: Tabbox

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Tabpanel.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/tab/Tabpanel.html#>
- [3] <http://www.zkoss.org/forum/listComment/9889>
- [4] <http://www.zkoss.org/forum/listComment/6236>

Tabpanels

Tabpanels

- Demonstration: Tabbox [1]
- Java API: Tabpanels [1]
- JavaScript API: Tabpanels [2]
- Style Guide: Tabbox

Employment/Purpose

A `tabpanels` is the container for the tab panels, i.e., a collection of `tabpanel` components.

Example

The screenshot shows a `tabbox` component with five tabs labeled "Tab 1", "Tab 2", "Tab 3", "Tab 4", and "Tab 5". "Tab 1" and "Tab 3" are active, displaying content boxes. "Tab 1" contains "This is panel 1". "Tab 3" contains "This is panel 3". "Tab 4" contains "This is panel 4". "Tab 5" contains "This is panel 5". A horizontal separator line is located between the tabs and the content boxes.

```
<zk>
    <tabbox width="400px">
        <tabs>
            <tab label="Tab 1" />
            <tab label="Tab 2" />
        </tabs>
        <tabpanel>This is panel 1</tabpanel>
        <tabpanel>This is panel 2</tabpanel>
    </tabpanel>
    <space />
    <tabbox width="400px" mold="accordion">
        <tabs>
            <tab label="Tab 3" />
            <tab label="Tab 4" />
        </tabs>
        <tabpanel>This is panel 3</tabpanel>
        <tabpanel>This is panel 4</tabpanel>
    </tabpanel>
</tabbox>
```

```
</zk>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

Tabpanel

Use Cases

Tabbox

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Tabpanels.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/tab/Tabpanels.html#>

Window

Window

- Demonstration: Window ^[1]
- Java API: Window ^[2]
- JavaScript API: Window ^[3]
- Style Guide: Window

Employement/Purpose

A window is, like HTML DIV tag, used to group components. Unlike other components, a window has the following characteristics.

- A window is an owner of an ID space. Any component contained in a window, including itself, could be found by use of Component.getFellow(java.lang.String) ^[4], if it is assigned with an identifier.
- A window could be overlapped, popup, and embedded.
- A window could be a modal dialog.

Example



```
<window title="Embedded Style" border="normal" width="200px">Hello,  
    Embedded!  
</window>  
<window title="Overlapped Style" mode="overlapped" border="normal"  
    width="200px">Hello, Overlapped!  
</window>
```

Window Modes

A window could be in one of five different modes:

- embedded (**default**)
- overlapped
- popup
- modal
- highlighted and

You could change the mode by the use of Window.setMode(java.lang.String) ^[5].

```
<window title="Hi, I'm Overlapped" border="normal" mode="overlapped">  
...  
</window>
```

Alternatively, you could invoke one of Window.doOverlapped() ^[6], Window.doPopup() ^[7], Window.doModal() ^[8], Window.doHighlighted() ^[9], and Window.doEmbedded() ^[10], as shown below.

```
<zk>
<window id="win" title="Hi!" border="normal" width="200px">
  <caption>
    <toolbarbutton label="Help"/>
  </caption>
  <checkbox label="Hello, Wolrd!"/>
</window>

<button label="Overlap" onClick="win.doOverlapped();"/>
<button label="Popup" onClick="win.doPopup();"/>
<button label="Modal" onClick="win.doModal();"/>
<button label="Embed" onClick="win.doEmbedded();"/>
<button label="Highlighted" onClick="win.doHighlighted();"/>
</zk>
```

Embedded

An embedded window is placed inline with other components. In this mode, you cannot change its position, since the position is decided by the browser. It is the default mode since it is the most common appearance.

Overlapped

An overlapped window is overlapped with other components, such that users could drag it around and developer could set its position by Window.setLeft(java.lang.String)^[11] and Window.setTop(java.lang.String)^[12].

```
<window title="My Overlapped" width="300px" mode="overlapped">
</window>
```

An overlapped window is typically used to display the information that should co-exist with the current operation and should appear for a long time. You might have multiple overlapped windows and each for different set of information. If you want to show the information that will appear only temporarily (dismissed as soon as a user clicks somewhere else), you could use the popup mode as described in the next section, or the Popup component.

Popup

A popup window is similar to overlapped windows, except it is automatically closed when user clicks on any component other than the popup window itself or any of its descendants. Of course, you could dismiss it manually by making it invisible or detaching it.

As its name suggested, it is designed to implement the popup windows. A typical application is to display information that won't obscure the current operation and are easy to close. A popup window is usually position around the focal point (such as a button). It can be done by use of Window.setPosition(java.lang.String)^[13] with parent.

For example, we could display a popup window right after a button as depicted below.

```
<zk>
<toolbarbutton label="More info" onClick="info.doPopup()"/><span>
<window id="info" visible="false" width="120px" border="normal" position="parent">
  Here is more information
</window>
</span>
```

```
</zk>
```

where we specify `position="parent"`, and make it as a child of a `span` component. The `span` component acts as an anchor point and the window is positioned based on it.

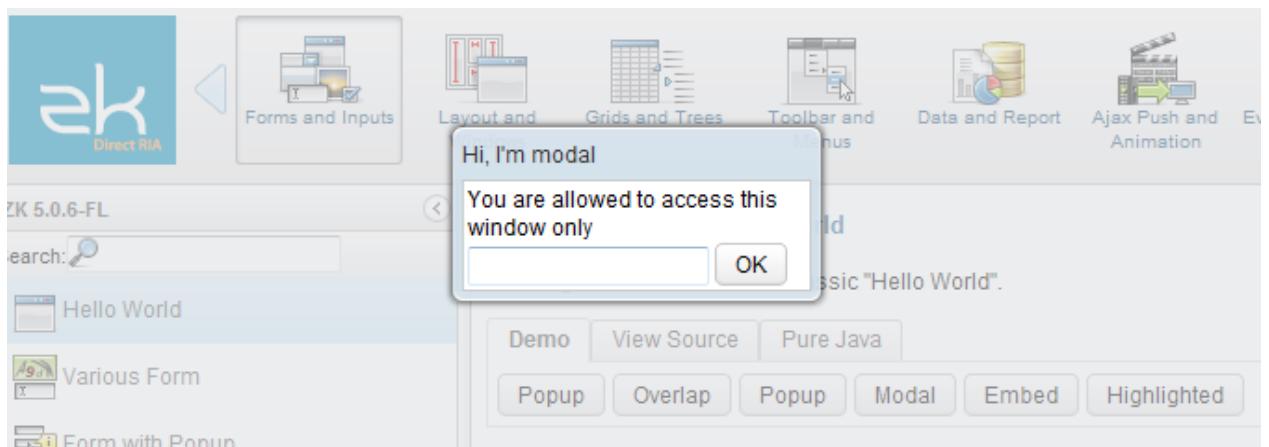
In addition to `popup` windows, you could use `Popup` for displaying a `popup`. The `popup` component has more control how to position it (by the use of `java.lang.String`) `Popup.open(org.zkoss.zk.ui.Component, java.lang.String)` [14]).

Modal and Highlighted

By default, a modal window is the same as a highlighted window. Unless you configure ZK to enable the event thread (see the next section), you could consider them as exactly same.

A modal window provides the so-called *modal* effect that limits a user from accessing components other than the modal window. Users cannot access anything outside of the modal window, including clicking or tabbing.

For instance, you could access only the textbox and button in the following example:



To dismiss a modal window, you could make it invisible (`Window.setVisible(boolean)`) [15]), or detach it from a page.

By default, a modal window is positioned at the center of the browser window. You could change the position by `Window.setPosition(java.lang.String)` [13].

You could have multiple modal windows at the same time, and the user could only access the last modal window. Once the last modal is dismissed (invisible or detached), the previous modal window will become the *active* modal window until it is dismissed.

Modal Windows and Event Processing Threads

Notice: Event processing thread is disabled by default since 5.0. For the older version, it is enabled by default

By default, events are processed in the same thread that serves the HTTP request (so-called Servlet thread). However, you could configure ZK to process events in an individual thread, such that the event listener could suspend the execution at any time, and resume later. For how to enable the event processing thread, please refer to ZK Configuration Reference.

Notice that, for better integration with other frameworks, such as Spring, it is suggested to *disable* the event processing thread (default). For more information, please refer to the Event Threads section.

Once the event thread is enabled, a modal window will behave differently from other modes: `Window.doModal()`^[8] will suspend the execution until dismissed (invisible, detached or mode changed). It is convenient to implement something that has to wait for user's further input.

As depicted in the following example, `f1()` is called only after `win1` is dismissed, while `g1()` is called immediately right after `win2` becomes highlighted:

```
win1.doModal(); //the execution is suspended until win1 is closed
f1();
win2.doHighlighted(); //the execution won't be suspended
g1()
```

Properties and Features

Border

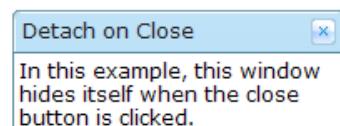
The `border` property (`Window.setBorder(java.lang.String)`^[16]) specifies whether to display a border for window. The default style sheets support only `normal` and `none`. The default value is `none`, i.e., no border.

Closable

By setting the `closable` property (`Window.setClosable(boolean)`^[17]) to true, a close button is shown for the window, which enables a to close the window by clicking the button. Once the user clicks on the `close` button, an `onClose` event is sent to the window which is processed by the `onClose` method of the `Window` component. Then, `onClose`, by default, detaches the window itself.

The `onClose` Event

You can override it to do whatever you want. Or, you can register your own listener to change the default behavior. For example, you might choose to hide the window rather than close it.



```
<window closable="true" title="Detach on Close" border="normal" width="200px"
onClose="self.visible = false; event.stopPropagation();">
    In this example, this window hides itself when the close button is
    clicked.
</window>
```

Notice that `event.stopPropagation()` (`Event.stopPropagation()`^[18]) must be called to prevent the default `onClose` handler (`Window.onClose()`^[19]) being called.

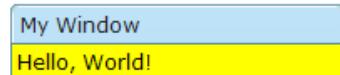
Tip: If the window is a popup, the `onOpen` event will be sent to the window with `open=false`, when the popup is closed due to the user clicking outside of the window, or pressing ESC.

The `onClose` is sent to ask the server to detach or to hide the window. By default, the window is detached. Of course, the application can override this behavior and do whatever it wants as described above.

On the other hand, `onOpen` is a notification. It is sent to notify the application that the client has hidden the window. The application cannot prevent it from hiding or changing the behavior to be detached.

ContentStyle and ContentClass

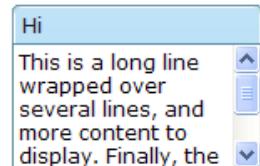
You can customize the look and feel of window's content block by specifying the `contentStyle` property (`Window.setContentStyle(java.lang.String)`)^[20].



```
<zk>
<window title="My Window" border="normal" width="200px" contentStyle="background:yellow">
    Hello, World!
</window>
</zk>
```

Scrollable Window

A typical use of the `contentStyle` attribute is to make a window scrollable as follows.



```
<window id="win" title="Hi" width="150px" height="100px" contentStyle="overflow:auto" border="normal">
    This is a long line wrapped over several lines, and more content to display. Finally, the
    scrollbars become visible.
    This is another line.
</window>
```

Note: For IE 7's overflow bug, also use **position:relative** with `overflow:auto`

Position

In addition to the `left` and `top` properties, you can control the position of an **overlapped/popup/modal/highlighted** window by the use of the `position` attribute. For example, the following code snippet positions the window to the right-bottom corner.

```
<window width="300px" mode="overlapped" position="right,bottom">
    ...
```

The value of the `position` attribute can be a combination of the following constants by separating them with comma (,).

Constant	Description
center	Position the window at the center. If <code>left</code> or <code>right</code> is also specified, it means the vertical center. If <code>top</code> or <code>bottom</code> is also specified, it means the horizontal center. If none of <code>left</code> , <code>right</code> , <code>top</code> and <code>bottom</code> is specified, it means the center in both directions. Both the <code>left</code> and <code>top</code> property are ignored.
left	Position the window at the left edge. The <code>left</code> property is ignored.
right	Position the window at the right edge. The <code>left</code> property is ignored.
top	Position the window at the top. The <code>top</code> property is ignored.
bottom	Position the window at the bottom. The <code>top</code> property is ignored.

By default, its value is null. That is, the overlapped and popup window is positioned by the `left` and `top` properties, while the modal window is positioned at the center.

Sizable

If you allow users to resize the window, you can set the `sizable` attribute to true as follows.

```
<window id="win" title="Sizable Window" border="normal" width="200px" sizable="true">
    This is a sizable window.
    <button label="Change Sizable" onClick="win.sizable = !win.sizable"/>
</window>
```

Once allowed, users can resize the window by dragging the borders.

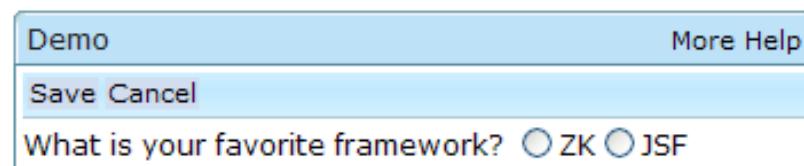
The onSize Event

Once a user resizes the window, the `onSize` event is sent with an instance of the `org.zkoss.zul.event.SizeEvent`. Notice that the window is resized before the `onSize` event is sent. In other words, the event serves as a notification that you generally ignore. Of course, you can do whatever you want in the event listener.

Note: If the user drags the upper or left border, the `onMove` event is also sent since the position has changed, too.

Title and Caption

A window might have a title, a caption and a border. The title is specified by the `title` attribute. The caption is specified by declaring a child component called `caption`. All children of the `caption` component will appear on right hand side of the title.



```
<zkc>
<window title="Demo" border="normal" width="350px">
```

```

<caption>
    <toolbarbutton label="More" />
    <toolbarbutton label="Help" />
</caption>
<toolbar>
    <toolbarbutton label="Save" />
    <toolbarbutton label="Cancel" />
</toolbar>
What is your favorite framework?
<radiogroup>
    <radio label="ZK" />
    <radio label="JSF" />
</radiogroup>
</window>
</zk>

```

You are also able to specify a label and an image within a caption, and then the appearance is as follows.



```

<zk>
    <window id="win" title="Main" border="normal" width="200px">
        <caption image="/images/ZK-Logo.PNG" label="Hi there!" />
        <checkbox label="Hello, World!" />
    </window>
</zk>

```

Trouble shooting with browser issue

- There's a issue for Listbox/Grid in window , please reference Grid in window get wrong display in IE7/IE6 [21]

Common Dialogs

The XUL component set supports the following common dialogs to simplify some common tasks.

- Messagebox
- Fileupload
- Filedownload

Supported Events

Name	Event Type
onMove	Event: Event [7] Denotes the position of the window is moved by a user.
onOpen	Event: OpenEvent [3] Denotes user has opened or closed a component. Note: Unlike <code>onClose</code> , this event is only a notification. The client sends this event after opening or closing the component. It is useful to implement load-on-demand by listening to the <code>onOpen</code> event, and creating components when the first time the component is opened.
onClose	Event: Event [7] Denotes the close button is pressed by a user, and the component shall detach itself.
onMaximize	Event: MaximizeEvent [7] Denotes user has maximize a component.
onMinimize	Event: MinimizeEvent [8] Denotes user has minimize a component.
onSize	Event: SizeEvent [4] Denotes the panel's size is updated by a user.
onZIndex	Event: ZIndexEvent [9] Denotes the panel's zindex is updated by a user.

- Inherited Supported Events: XulElement

Supported Children

*ALL

Use Cases

Version	Description	Example Location
5.0+	How to create a modal Window and communicate with it	[22]
3.6+	Best practises on creating a pop-up window to display PDF reports	[23]

Version History

Version	Date	Content
---------	------	---------

References

- ```
[1] http://www.zkoss.org/zkdemo/window/positioning
[2] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Window.html#
[3] http://www.zkoss.org/javadoc/latest/jsdoc/zul/wnd/Window.html#
[4] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/Component.html#getFellow(java.lang.String)
[5] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Window.html#setMode(java.lang.String)
[6] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Window.html#doOverlapped()
[7] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Window.html#doPopup()
[8] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Window.html#doModal()
[9] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Window.html#doHighlighted()
[10] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Window.html#doEmbedded()
[11] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Window.html#setLeft(java.lang.String)
[12] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Window.html#setTop(java.lang.String)
[13] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Window.html#setPosition(java.lang.String)
[14] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Popup.html#open(org.zkoss.zk.ui.Component,
[15] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Window.html#setVisible(boolean)
[16] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Window.html#setBorder(java.lang.String)
[17] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Window.html#setClosable(boolean)
[18] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/Event.html#stopPropagation()
[19] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Window.html#onClose()
[20] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Window.html#setContentStyle(java.lang.String)
[21] https://sourceforge.net/tracker/?func=detail&atid=785191&aid=3291179&group_id=15276213291179
[22] http://www.zkoss.org/forum/listComment/9785
[23] http://www.zkoss.org/forum/listComment/9305
```

# Data

---

This section outlines components which are designed to handle data display.

Grid and listbox are both used to display tabular data, while tree to display hierarchical data. However, both listbox and tree allow the user to select one or multiple items it displays, while grid doesn't.

Grid, listbox and tree all allow developers to separate long content into multiple pages. For more information, please refer to the Paging component.

## Grid

---

### Grid

- Demonstration: Grid <sup>[1]</sup>
- Java API: Grid <sup>[2]</sup>
- JavaScript API: Grid <sup>[3]</sup>
- Style Guide: Grid

### Employment/Purpose

Components: `grid`, `columns` and `column`, `rows` and `row`, `Auxhead` and `Auxheader`

A `grid` contains components that are aligned in rows like tables. Inside a `grid` you place `columns`, `rows`, and headers.

`Columns` defines the header and `column` attributes. They are declared with the `columns` component and assigned as a child element of the `grid`. `Column` declares the common attributes of each `column` such as the width and alignment. An optional `column` header may contain objects such as a label and/or image. Although `columns` is optional, if it exists, notice that the number of its child (`column`) should equal the number of child in a `row`. If `columns` contains no child, the `grid` will display nothing in its content.

`Rows` hold the content. They are declared with the `rows` component and assigned as a child element of `grid`. Inside `rows` you should add one `row` component for each `row` of the `grid`. Inside each `row` element you should place child components which contain the content that you want inside that `row`. Each child component should align with a `column` of the specific `row`.

Additional auxiliary headers may be added with the `auxhead` component. The `auxhead` must be closely tied with the `columns` declaration. For more details see the section on Axillary Headers below.

## Example



```

<window title="Grid Demo" border="normal" width="360px">
 <zscript>
 class Comp implements Comparator {
 private boolean _asc;
 public Comp(boolean asc) {
 _asc = asc;
 }
 public int compare(Object o1, Object o2) {
 String s1 = o1.getChildren().get(0).getValue(),
 s2 = o2.getChildren().get(0).getValue();
 int v = s1.compareTo(s2);
 return _asc ? v: -v;
 }
 }
 Comp asc = new Comp(true), dsc = new Comp(false);
 </zscript>
 <grid>
 <columns sizable="true">
 <column label="Type" sortAscending="#{asc}" sortDescending="#{dsc}" width="50px"/>
 <column label="Content"/>
 </columns>
 <rows>
 <row>
 <label value="File:"/>
 <textbox width="99%"/>
 </row>
 <row>
 <label value="Type:"/>
 <hbox>
 <listbox rows="1" mold="select">
 <listitem label="Java Files, (*.java)"/>
 <listitem label="All Files, (*.*)"/>
 </listbox>
 </hbox>
 </row>
 </rows>
 </grid>

```

```

 </listbox>
 <button label="Browse...."/>
 </hbox>
</row>
<row>
 <label value="Options:"/>
 <textbox rows="3" width="99%"/>
</row>
</rows>
</grid>
</window>

```



row count: 3

```

<zk>
 <zscript><![CDATA[
 ListModelList lm = new ListModelList(Arrays.asList(new
String[] { "David",
 "Thomas", "Steven" }));
]]></zscript>

 <grid model="${lm}" width="300px"
 onAfterRender='gridCount.setValue("row count:
"+self.getRows().getChildren().size()+"");' />
 <label id="gridCount" />
</zk>

```

## Paging

The `listbox` and `grid` components support the paging intrinsically, so you don't need to specify a paging component explicitly as above unless you want to have different visual layout or to control multiple `listbox` and `grid` controls with one paging component.

## Data Model with Paging

[Since 8.0.0]

If `listbox` or `grid` component is used with a data model, we recommend you control paging from the model directly. Although specifying page size and active page on the component is still supported, model now provides API for paging control, which makes using model more convenient.

```

<zscript><![CDATA[
import org.zkoss.zul.ext.Pageable;

List model = new ListModelList(Locale.getAvailableLocales());

```

```

public void next() {
 int activePage = ((Pageable)model).getActivePage();
 ((Pageable)model).setActivePage(activePage + 1);
}

public void previous() {
 int activePage = ((Pageable)model).getActivePage();
 if (--activePage < 0){
 activePage = 0;
 }
 ((Pageable)model).setActivePage(activePage);
}

]]></zscript>
<grid model="${model}" mold="paging" pageSize="5">
 <columns>
 <column label="Locale"/>
 </columns>
 <template name="model">
 <row>
 ${each}
 </row>
 </template>
</grid>
<button label="Next" onClick="next () "/>
<button label="Previous" onClick="previous () "/>

```

## Grids with Paging

There are two ways to handle large content in a grid: scrolling and paging. Scrolling is enabled by setting the `height` attribute as discussed in the previous section. Paging is enabled by setting the `mold` attribute to `paging`. Once paging is enabled, the grid separates the content into several pages and displays one page at a time as depicted below.

| Left                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | Right         |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| Item 1.1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Item 1.2      |
| Item 2.1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Item 2.2      |
| Item 3.1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Item 3.2      |
| Item 4.1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Item 4.2      |
|    /    | [ 1 - 4 / 7 ] |

```

<grid width="300px" mold="paging" pageSize="4">
 <columns>
 <column label="Left"/>
 <column label="Right"/>
 </columns>

```

```
</columns>
<rows>
 <row>
 <label value="Item 1.1"/><label value="Item 1.2"/>
 </row>
 <row>
 <label value="Item 2.1"/><label value="Item 2.2"/>
 </row>
 <row>
 <label value="Item 3.1"/><label value="Item 3.2"/>
 </row>
 <row>
 <label value="Item 4.1"/><label value="Item 4.2"/>
 </row>
 <row>
 <label value="Item 5.1"/><label value="Item 5.2"/>
 </row>
 <row>
 <label value="Item 6.1"/><label value="Item 6.2"/>
 </row>
 <row>
 <label value="Item 7.1"/><label value="Item 7.2"/>
 </row>
</rows>
</grid>
```

Once the paging mold is set, the grid creates an instance of a `Paging` component **as the Grid's child** and the paging component in turn handles the Grid's paging. Therefore, the number of the Grid's children includes the paging component. Also, if you remove all children of the `grid`, the `paging` is also removed.

## The PageSize Property

Having set the `paging` mold, you can specify how many rows are visible at a time (i.e., the page size) by setting the `pageSize` attribute to a numeric value. By default, it is 20.

## The Paginal Property

If you prefer to place the `paging` component in a different location or if you want to control two or more grids with the same `paging` component, you can assign the `paginal` attribute explicitly. Note: if it is not set explicitly, it is the same as the `paging` property.

Left	Right
Item 1.1	Item 1.2
Item 2.1	Item 2.2
Item 3.1	Item 3.2
Item 4.1	Item 4.2
Item A.1	Item A.2
Item B.1	Item B.2
Item C.1	Item C.2
Item D.1	Item D.2

```
<vbox>
 <paging id="pg" pageSize="4"/>
 <hbox>
 <grid width="300px" mold="paging" paginal="${pg}">
 <columns>
 <column label="Left"/><column label="Right"/>
 </columns>
 <rows>
 <row>
 <label value="Item 1.1"/><label value="Item 1.2"/>
 </row>
 <row>
 <label value="Item 2.1"/><label value="Item 2.2"/>
 </row>
 <row>
 <label value="Item 3.1"/><label value="Item 3.2"/>
 </row>
 <row>
 <label value="Item 4.1"/><label value="Item 4.2"/>
 </row>
 <row>
 <label value="Item 5.1"/><label value="Item 5.2"/>
 </row>
 <row>
 <label value="Item 6.1"/><label value="Item 6.2"/>
 </row>
 <row>
 <label value="Item 7.1"/><label value="Item 7.2"/>
 </row>
 </rows>
 </grid>
 <grid width="300px" mold="paging" paginal="${pg}">
 <columns>
 <column label="Left"/><column label="Right"/>
 </columns>
 <rows>
```

```
<row>
 <label value="Item A.1"/><label value="Item A.2"/>
</row>
<row>
 <label value="Item B.1"/><label value="Item B.2"/>
</row>
<row>
 <label value="Item C.1"/><label value="Item C.2"/>
</row>
<row>
 <label value="Item D.1"/><label value="Item D.2"/>
</row>
<row>
 <label value="Item E.1"/><label value="Item E.2"/>
</row>
<row>
 <label value="Item F.1"/><label value="Item F.2"/>
</row>
</rows>
</grid>
</hbox>
</vbox>
```

## The Paging Property

It is a read-only attribute representing the child `paging` component that is created automatically. It is null if you assign external paging via the `paginal` attribute. You rarely need to access this attribute as it is generally better to use the `paginal` attribute.

## The PagingDisabled Property

[Since 8.0.3]

Once the `pagingDisabled` is set to `true`, users will be blocked from navigating through the pagination.

## The onPaging Event and Method

Once a user clicks the page number of the `paging` component, an `onPaging` event is sent the grid. It is then processed by the `onPaging` method. By default, the method invalidates, i.e., redraws, the content of `rows`.

If you want to implement "create-on-demand" feature, you can add a event listener to the grid for the `onPaging` event. The line below demonstrates how to add an `EventListener`.

```
grid.addEventListener(org.zkoss.zul.event.ZulEvents.ON_PAGING, new
MyListener());
```

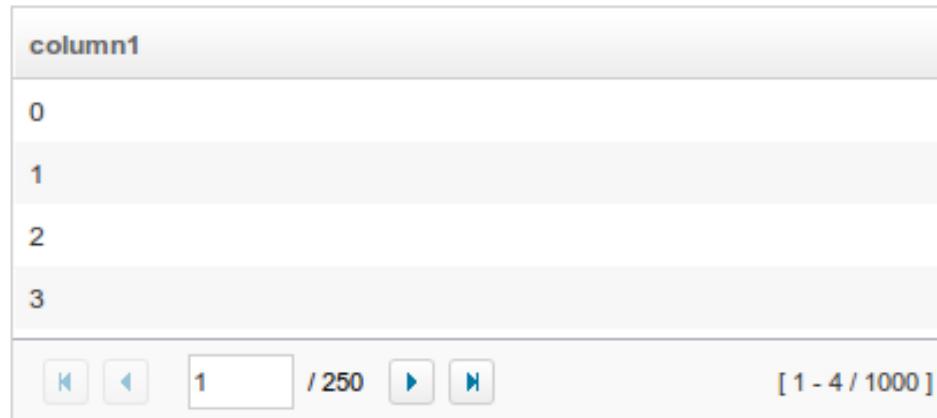
## Autopaging

[Since 5.0.2]

When using the paging mold and vflex, you could also turn on autopaging (`Grid.setAutopaging(boolean)` [4]) such that the page size will be adjusted automatically based on the available space.

For example,

```
<grid id="grid" autopaging="true" mold="paging" vflex="1">
 <columns>
 <column label="column1"/>
 </columns>
 <rows>
 <row forEach="${items}">
 ${each}
 </row>
 </rows>
</grid>
```



**Note:** Autopaging depends on a fixed row height for all rows (i.e. you can't use line wrapping text in cells or the `<details>` component). Because of that, once enabled, ZK will apply fixed height for each row with the following CSS by default. If you want to change the height, please overwrite the CSS rule as your preference. But this feature only works when each row has the same height.

```
.z-grid-autopaging .z-row-cnt {
 height: 30px;
 overflow: hidden;
}
```

[Since 5.0.8]

**Note:** In ZK 7, we change the naming `.z-row-cnt` to `.z-row-content`.

[Since 7.0.3]

## Sort

### Sorting

Grids support the direct sorting of rows. To enable ascending order sorting for a particular column, you need to assign a `java.util.Comparator` instance to the `sortAscending` attribute of the column. Similarly, you assign a comparator to the `sortDescending` property to enable the descending order.

As illustrated below, you first implement a comparator that compares any two rows of the grid, and then assign its instances to the `sortAscending` and/or `sortDescending` attributes. Notice: the `compare` method is passed two `Row`<sup>[5]</sup> instances.

```
<zk>
<zscript>
 class MyRowComparator implements Comparator {
 public MyRowComparator(boolean ascending) {
 ...
 }
 public int compare(Object o1, Object o2) {
 Row r1 = (Row)o1, r2 = (Row)o2;

 }
 }
 Comparator asc = new MyRowComparator(true);
 Comparator dsc = new MyRowComparator(false);
</zscript>
<grid>
 <columns>
 <column sortAscending="${asc}" sortDescending="${dsc}"/>
 ...

```

### The `onSort` Event

When you assign at least one comparator to a column, an `onSort` event is sent to the server if user clicks on it. The `column` component implements a listener to automatically sort rows based on the assigned comparator.

If you prefer to handle this manually, you can add your own listener to the column for the `onSort` event. To prevent the default listener to invoking the `sort` method, you have to call the `stopPropagation` method for the event being received. Alternatively, you can override the `sort` method, see below.

## The sortDirection Property

The `sortDirection` property controls whether to show an icon to indicate the order of a particular column. If rows are sorted before being added to the grid, you should set this property explicitly.

```
<column sortDirection="ascending"/>
```

It is then maintained automatically by the grid as long as you assign the comparators to the corresponding column.

## The sort Method

The `sort` method is the underlying implementation of the default `onSort` event listener. It is also useful if you want to sort the rows using Java code. For example, you might have to call this method after adding rows (assuming they not in the proper order).

```
Row row = new Row();
row.setParent(rows);
row.appendChild(...);
...
if (!"natural".equals(column.getSortDirection()))
 column.sort("ascending".equals(column.getSortDirection()));
```

The default sorting algorithm is quick-sort (by use of the `sort` method from the Components<sup>[6]</sup> class). You can override it with your own implementation.

Note: the `sort` method checks the sort direction (by calling `getSortDirection`). It sorts the rows only if the sort direction is different. To enforce the sorting, do as follows.

```
column.setSortDirection("natural");
sort(myorder);
```

The above code is equivalent to the following.

```
sort(myorder, true);
```

**Update:** see more about sorting Multiple Field Sorting on Listbox.

## Live Data

Like list boxes, grids support *live data*. With live data, developers are able to separate the data from the view. In other words, developers only need to provide the data by implementing the `ListModel`<sup>[7]</sup> interface, rather than manipulating the grid directly. The benefits are twofold.

- It is easier to use different views to show the same set of data.
- The grid sends the data to the client only if it is visible. It saves a lot of network traffic if the amount of data is large.

There are three steps to make use of live data.

1 Prepare the data in the form of a `ListModel`. ZK has a concrete implementation called `SimpleListModel`<sup>[8]</sup> for representing an array of objects. 2 Implement the `RowRenderer`<sup>[9]</sup> interface for rendering a row of data into the grid.

- This is optional. If it is not specified the default renderer is used to render the data into the first column.
- You can implement different renderers for representing the same data in different views.

3 Set the data in the `model` attribute and, optionally, the renderer in the `rowRenderer` attribute.

In the following example, we prepared a list model called `strset`, assign it to a grid using the `model` attribute. Then, the grid will do the rest.



```
<window title="Live Grid" border="normal" width="100px">
 <zscript><! [CDATA[
 String[] data = new String[30];
 for(int j=0; j < data.length; ++j) {
 data[j] = "option "+j;
 }
 ListModel strset = new SimpleListModel(data);
]]></zscript>
 <grid height="100px" model="${strset}">
 <columns>
 <column label="options"/>
 </columns>
 </grid>
</window>
```

## Sorting with Live Data

If you allow users to sort a grid with live data, you have to implement the interface, `ListModelExt`<sup>[10]</sup>, in addition to the `ListModel`<sup>[7]</sup>.

```
class MyListModel implements ListModel, ListModelExt {
 public void sort(Comparator cmpr, boolean ascending) {
 //do the real sorting
 //notify the grid (or listbox) that data is changed by use
 of ListDataEvent
 }
}
```

When a user wants to sort the grid, the grid will invoke the `sort` method of `ListModelExt` to sort the data. In other words, the sorting is done by the list model, rather than the grid.

After sorting, the list model will notify the grid by invoking the `onChange` method of the grid's registered `ListDataListener`<sup>[11]</sup> instances. These are registered by the `addListDataListener` method. In most cases, all the data is changed, so the list model usually sends the following event:

```
new ListDataEvent(this, ListDataEvent.CONTENT_CHANGED, -1, -1)
```

## Scroll to a Specific Item

since 8.5.2

You can call `scrollToIndex(int)` to scroll a Grid to a specific item, and this even works under ROD.

## Properties

### Scrollable Grid



A grid can be scrollable and has a frozen header if you specify the `height` attribute and there is not enough space to display all data.

```
<grid span="true" sizedByContent="true" width="500px" height="130px">
 <columns>
 <column label="Head 1"/>
 <column label="Head 2" align="center"/>
 <column label="Head 3" align="right"/>
 </columns>
 <rows>
 <row>
 <listbox mold="select">
 <listitem label="Faster"/>
 <listitem label="Fast"/>
 <listitem label="Average"/>
 </listbox>
 <datebox/>
 <textbox rows="2"/>
 </row>
 <row>
 <checkbox checked="true" label="Option 1"/>
 <checkbox label="Option 2"/>
 <radiogroup>
 <radio label="Apple"/>
 <radio label="Orange" checked="true"/>
 <radio label="Lemon"/>
 </radiogroup>
 </row>
 <row>
 <checkbox checked="true" label="Option 1"/>
 </row>
 </rows>
</grid>
```

```

<checkbox label="Option 2"/>
<radiogroup orient="vertical">
 <radio label="Apple"/>
 <radio label="Orange" checked="true"/>
 <radio label="Lemon"/>
</radiogroup>
</row>
</rows>
</grid>

```

[Since 7.0.0]

The browser's default scrollbar is replaced by floating scrollbar and it is not visible unless user mouse over on the content. To turn off the floating scrollbar and use original scrollbar, please add the following configuration in zk.xml.

```

<library-property>
 <name>org.zkoss.zul.nativebar</name>
 <value>true</value>
</library-property>

```

**Note:** the value of org.zkoss.zul.nativebar is true by default (since 7.0.2)

## Auxiliary Headers

In addition to columns headers, you can specify auxiliary headers with the auxhead and auxheader components as follows.

H1'07						H2'07					
Q1			Q2			Q3			Q4		
Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1,000	1,100	1,200	1,300	1,400	1,500	1,600	1,700	1,800	1,900	2,000	2,100

```

<grid>
 <auxhead>
 <auxheader label="H1'07" colspan="6"/>
 <auxheader label="H2'07" colspan="6"/>
 </auxhead>
 <auxhead>
 <auxheader label="Q1" colspan="3"/>
 <auxheader label="Q2" colspan="3"/>
 <auxheader label="Q3" colspan="3"/>
 <auxheader label="Q4" colspan="3"/>
 </auxhead>
 <columns>
 <column label="Jan"/><column label="Feb"/><column label="Mar"/>
 <column label="Apr"/><column label="May"/><column label="Jun"/>
 <column label="Jul"/><column label="Aug"/><column label="Sep"/>
 <column label="Oct"/><column label="Nov"/><column label="Dec"/>
 </columns>

```

```

<rows>
 <row>
 <label value="1,000"/><label value="1,100"/><label value="1,200"/>
 <label value="1,300"/><label value="1,400"/><label value="1,500"/>
 <label value="1,600"/><label value="1,700"/><label value="1,800"/>
 <label value="1,900"/><label value="2,000"/><label value="2,100"/>
 </row>
</rows>
</grid>

```

The auxiliary headers support the `colspan` and `rowspan` attributes that the column headers don't. Auxiliary headers must be placed in relation to a column. Without the specific declaration of `columns` auxiliary headers are ignored.

Unlike `column/columns`, which can only be used with `grid`, `auxhead/auxheader` can be used with `grid`, `listbox` and `tree`.

## The Limitation of rowspan

For better performance, every instance of Column will create an invisible HTML TH element called *faker*. However, with some complex combination of `rowspan` and `colspan`, Grid might not be able to generate the correct number of *faker* to represent each column.

For example, it is wrong if the number of the column components are not the same as the number of columns in each row as shown below:

```

<grid width="200px">
 <auxhead>
 <auxheader label="A" rowspan="2" />
 <auxheader label="BC" colspan="2" />
 <auxheader label="D" rowspan="2" />
 </auxhead>
 <columns><!-- this is wrong since the number of column components is smaller -->
 <column label="B"/>
 <column label="C"/>
 </columns>
 <rows>
 <row>
 <label forEach="E,F,G,H" value="${each}" /><!-- four columns -->
 </row>
 </rows>
</grid>

```

A	BC	D
	B	
E	F	H

As shown above, the column with label C will be invisible, because the fakers are not created correctly. Here is the result but wrong DOM structure:

Table

A	BC		D
	B		
faker(B)	faker(C)	faker(flex)	

There is a simple workaround: specify all columns. If you don't want to show all columns, you could use Auxheader instead of Column, and then add an empty Columns. For example, the code in the previous example can be fixed as follows:

```
<grid width="200px">
 <auxhead>
 <auxheader label="A" rowspan="2" />
 <auxheader label="BC" colspan="2" />
 <auxheader label="D" rowspan="2" />
 </auxhead>
 <auxhead>
 <auxheader label="B"/>
 <auxheader label="C"/>
 </auxhead>
 <columns/> <!-- use an empty columns to make fakers created correctly -->
 <rows>
 <row>
 <label forEach="E,F,G,H" value="${each}" />
 </row>
 </rows>
</grid></pre>

```

A	BC		D
	B	C	
E	F	G	H

The other limitation is that the width of the Auxheader component will depend the Column component. Thus, if you'd like to specify the width in the Column component, please note that it will take some space even if there are no label in all Column components. The workaround is simple: make the empty Columns component invisible. For example,

```

<grid width="350px">
 <auxhead>
 <auxheader label="A" rowspan="2" />
 <auxheader label="BC" colspan="2" />
 <auxheader label="D" rowspan="2" />
 </auxhead>
 <auxhead>
 <auxheader label="B" />
 <auxheader label="C" />
 </auxhead>
 <columns visible="false"><!-- make it invisible -->
 <column width="100px"/><!-- specify width here -->
 <column width="150px"/>
 <column width="50px"/>
 <column width="50px"/>
 </columns>
 <rows>
 <row>
 <label forEach="E, F, G, H" value="${each}" />
 </row>
 </rows>
</grid>

```

A	BC		D
	B	C	
E	F	G	H

## SizedByContent

By default, the widths of columns have to be specified explicitly, or it will be split equally among columns regardless what content they might have. If you want to have the minimal width (that fit the content), you could specify `hflex="min"` at the column (not the grid).

However, the grid has a special mode called sized-by-content (`Grid.setSizedByContent(boolean)`<sup>[12]</sup>). By specifying it to true, the column width will be adjusted automatically. However, as it is controlled by the browser, you will have no 100% control of it. For example, if an user resized a column, the final width might not be exactly the same as what he resized.

In general, we suggest to specify `hflex` in column, rather than specifying `sizedByContent` at grid for much more predictable result.

## Span

By default, when sizedByContent is true, column only take required space.

Time Message	Level	Source	Message
6/28/10 4:19:18 PM	Info, long content.....	Server	Merging recovery point 52 created 20 6/27/10 10 :11 PM
6/28/10 4:19:18 PM	Info, long content.....	Server	Merging recovery point 52
6/28/10 4:19:18 PM	Info, long content.....	Server	Merging recovery point 52

If wanna to span the width of the columns to occupy the whole grid, you could specify true to this attribute

Time Message	Level	Source	Message
6/28/10 4:19:18 PM	Info, long content.....	Server	Merging recovery point 52 created 20 6/27/10 10 :11 PM
6/28/10 4:19:18 PM	Info, long content.....	Server	Merging recovery point 52
6/28/10 4:19:18 PM	Info, long content.....	Server	Merging recovery point 52

```

<grid sizedByContent="true" span="true" width="800px">
 <columns>
 <column label="Time Message" />
 <column label="Level" />
 <column label="Source " />
 <column label="Message" />
 </columns>
 <rows>
 <row>
 <label value="6/28/10 4:19:18 PM" />
 <label value="Info, long content....." />
 <label value="Server" />
 <label value="Merging recovery point 52 created 20 6/27/10 10 :11 PM" />
 </row>
 </rows>
</grid>

```

## Show messages when empty

The emptyMessage attribute is used to show a message when we have no items. [Since 5.0.7]

```

<grid id="test1" emptyMessage="No items match your search">
 <columns sizable="true">
 <column label="Type" width="520px" />
 <column label="Content" hflex="min" />
 <column label="Content" hflex="1" />
 </columns>
</grid>

```

## Sizable

If you allow users to resize the width of your columns, you can set the `sizable` attribute of your columns as `true`. Once allowed, users can resize the widths of columns by dragging the border between adjacent column components.

```
<window>
 <grid>
 <columns id="cs" sizable="true">
 <column label="AA"/>
 <column label="BB"/>
 <column label="CC"/>
 </columns>
 <rows>
 <row>
 <label value="AA01"/>
 <label value="BB01"/>
 <label value="CC01"/>
 </row>
 <row>
 <label value="AA01"/>
 <label value="BB01"/>
 <label value="CC01"/>
 </row>
 <row>
 <label value="AA01"/>
 <label value="BB01"/>
 <label value="CC01"/>
 </row>
 </rows>
 </grid>
 <checkbox label="sizeable" checked="true" onCheck="cs.sizeable = self.checked"/>
</window>
```

## Auto Fitting Columns

When you want to resize a column of a Grid or Listbox, all you now need to do is double click the column when the mouse is over where the columns meet and the column will automatically resize to fit its contents. To enable this functionality Grid's columns need the attribute `sizable="true"`. In other words, all sizable column provides the auto-fitting functionality.

[Since 5.0.0]

## The onColSize Event

Once a user resizes the width, the `onColSize` event is sent with an instance of `ColSizeEvent` [2]. Notice that the column's width is adjusted before the `onColSize` event is sent. In other words, the event serves as a notification that you can ignore. Of course, you can do whatever you want in the event listener.

## Spans

It is a list of comma separated integers, controlling whether to span a cell over several columns. The first number in the list denotes the number of columns the first cell shall span. The second number denotes the number of columns the second cell will span and so on. If a number is omitted, 1 is assumed.

For example,

```
<grid>
 <columns>
 <column label="Left" align="left"/><column label="Center" align="center"/>
 <column label="Right" align="right"/><column label="Column 4"/>
 <column label="Column 5"/><column label="Column 6"/>
 </columns>
 <rows>
 <row>
 <label value="Item A.1"/><label value="Item A.2"/>
 <label value="Item A.3"/><label value="Item A.4"/>
 <label value="Item A.5"/><label value="Item A.6"/>
 </row>
 <row spans="1,2,2">
 <label value="Item B.1"/><label value="Item B.2"/>
 <label value="Item B.4"/><label value="Item B.6"/>
 </row>
 <row spans="3">
 <label value="Item C.1"/><label value="Item C.4"/>
 <label value="Item C.5"/><label value="Item C.6"/>
 </row>
 <row spans=",,2,2">
 <label value="Item D.1"/><label value="Item D.2"/>
 <label value="Item D.3"/><label value="Item D.5"/>
 </row>
 </rows>
</grid>
```

## Visible Rows

The `visibleRows` attribute is used to control how many rows are visible. By setting it to zero, the grid will resize itself to hold as many as items if possible.

## Sticky Header

After adding a sclass "z-sticky-header", when we scroll down a page and make a Grid's header out of visible range in a viewport, the Grid's header becomes floating and sticky on the top of the page.

```
<grid sclass="z-sticky-header">
 <!-- columns, rows... -->
</grid>
```

## Master Detail

Grid supports master-detail concepts which enables developers to add more information on each row. For example,

	Product Name	Price	Item location
-	Apple 20-inch Aluminum Cinema Display M9177/A   <b>Item Specifics - Item Condition</b> Condition: Used Brand: Apple Technology: DVI Monitor Type: LCD/Flat Panel	US \$202.50	tulsa, ok, United States
+	Kyocera Strobe K612B MetroPCS Metro PCS Cell Phone L@@K	US \$74.99	Speedy Shipping, USA, United States

```
<rows>
 <row>
 <detail open="true">
 <hlayout>
 <image sclass="myimg" width="100px" height="100px"
 src="/img/item1.jpg" />
 // omitted...
 </hlayout>
 </detail>
 </row>
</rows>
```

- Available in ZK PE and EE only <sup>[13]</sup>
- For further details, please refer to Detail component directly.

## Columns Menu

For example,

Author	Title
Philip Hensher	
Philip Hensher	
Philip Hensher	
Michael Greenberg	
Michael Greenberg	
Rick Perlstein	
Rick Perlstein	Nixonland

```
<zk>
```

```
 <grid>
 <columns menupopup="auto">
 <column label="Author" sort="auto"/>
 <column label="Title" sort="auto"/>
 <column label="Publisher" sort="auto"/>
 <column label="Hardcover" sort="auto"/>
 </columns>
 // omitted...
 </grid>
</zk>
```

- For further details, please refer to Columns component directly.

## Ungroup Column Menu

When the user groups the content of the grid, the column's menu will show an ungroup icon for user to reset the group.

[ZK EE]

[Since 6.5.0]

Author	Title
▲ Michael Greenberg	<input type="button" value="Group"/>
Michael Greenberg	<input type="button" value="Ungroup"/>
Michael Greenberg	<input type="button" value="Sort Ascending"/>
▲ Philip Hensher	<input type="button" value="Sort Descending"/>
Philip Hensher	<input checked="" type="checkbox"/> Author
Philip Hensher	<input checked="" type="checkbox"/> Title
Philip Hensher	<input checked="" type="checkbox"/> Publisher
▲ Rick Perlstein	<input checked="" type="checkbox"/> Hardcover

**Note:** If the Grid contains with Model, *GroupsModel*, you have to register an *onUngroup* event for column to show an ungroup icon and then replace the current model with a *ListModel* to reset the group.

For example,

```
<zk>
<zscript><![CDATA[
int cnt = 0;

Object[][] foods = new Object[][] {
 new Object[] { "Vegetables", "Asparagus", "Vitamin K", 115, 43},
 new Object[] { "Vegetables", "Beets", "Folate", 33, 74},
 new Object[] { "Vegetables", "Tomatoes", "Vitamin C", 57, 37},
 new Object[] { "Seafood", "Salmon", "Tryptophan", 103, 261},
 new Object[] { "Seafood", "Cod", "Tryptophan", 90, 119}
};

public class FoodGroupRenderer implements RowRenderer {
 public void render(Row row, Object obj, int index) {
 if (row instanceof Group) {
 row.appendChild(new Label(obj.toString()));
 } else {
 Object[] data = (Object[]) obj;
 row.appendChild(new Label(data[0].toString()));
 row.appendChild(new Label(data[1].toString()));
 row.appendChild(new Label(data[2].toString()));
 row.appendChild(new Label(data[3].toString()));
 row.appendChild(new Label(data[4].toString()));
 }
 }
}
```

```

 }

}

ListModelList listmodel = new ListModelList();

for (int i = 0; i < foods.length; i++)
 listmodel.add(foods[i]);

RowRenderer renderer = new FoodGroupRenderer();

GroupsModel model = new GroupsModelArray(foods, new ArrayComparator(0,
true));

]]></zscript>

<grid id="grid" model="${model}" rowRenderer="${renderer}">

<columns menupopup="auto">
 <column label="Category" sort="auto(0)" onGroup='grid.setModel(model)' onUngroup='grid.setModel(listmodel);' />
 <column label="Name" sort="auto(1)" />
 <column label="Top Nutrients" sort="auto(2)" />
 <column label="% of Daily" sort="auto(3)" />
 <column label="Calories" sort="auto(4)" />
</columns>
</grid>
</zk>
```

## Cell Component

In ZK5, we have introduced a new component named Cell which can be embedded into a Grid or Box (Hbox and Vbox) to fully control the layout and the style. You can now use the rowspan or the colspan property to layout your Grid, for example a content cell can now cross over multiple rows. The code below demonstrates how to do this:

```
<row>
 <cell sclass="years" rowspan="12">
 ...
 </cell>
</row>
```

[Since 5.0.0]

- For further details, please refer to Cell component directly.

## Group Component

Both Grid, and Listbox support Grouping concept, it enables developers to display data in an advanced way. Moreover, live data are also supported in Grouping Grid, and Listbox with the GroupsModel<sup>[14]</sup> interface..

Brand	Processor Type	Memory (RAM)	Price	Hard Drive Capacity
<b>Dell</b>				
Dell E4500 2.2GHz	Intel Core 2 Duo	2GB RAM	\$261.00	500GB
XP-Pro Slim Dell-Inspiron-530-s	Intel Core 2 Duo	2GB RAM	\$498.93	500GB
Dell P4 3.2 GHz	Intel Pentium 4	4GB RAM	\$377.99	500GB
<b>Compaq</b>				
Compaq SR5113WM	Intel Core Duo	1GB RAM	\$279.00	160GB
Compaq HP XW4200	Intel Pentium 4	4GB RAM	\$980	500GB
This a summary about Compaq Desktop PCs				

```
<zkb>

<zscript>
import org.zkoss.zkdemo.userguide.*;
Comparator asc = new RowLabelComparator(true),
dsc = new RowLabelComparator(false);
</zscript>
<grid>
 <columns sizable="true">
 <column label="Brand" sortAscending="#{asc}" sortDescending="#{dsc}"/>
 <column label="Processor Type" width="150px"/>
 <column label="Memory (RAM)" width="120px"/>
 <column label="Price" width="100px"/>
 <column label="Hard Drive Capacity" width="150px"/>
 </columns>
 <rows>
 <group label="Dell">
 <row>
 <label style="padding-left:15px" value="Dell E4500 2.2GHz"/>
 <label value="Intel Core 2 Duo"/>
 <label value="2GB RAM"/>
 <label value="$261.00" style="color:green"/>
 <label value="500GB"/>
 </row>
 <row>
 <label style="padding-left:15px" value="XP-Pro Slim Dell-Inspiron-530-s"/>
 <label value="Intel Core 2 Duo"/>
 <label value="2GB RAM"/>
 <label value="$498.93" style="color:green"/>
 <label value="500GB"/>
 </row>
 <row>
 <label style="padding-left:15px" value="Dell P4 3.2 GHz"/>
 <label value="Intel Pentium 4"/>
 <label value="4GB RAM"/>
 <label value="$377.99" style="color:green"/>
 <label value="500GB"/>
 </row>
 <group label="Compaq">
 <row>
 <label style="padding-left:15px" value="Compaq SR5113WM"/>
 <label value="Intel Core Duo"/>
 <label value="1GB RAM"/>
 <label value="$279.00" style="color:green"/>
 <label value="160GB"/>
 </row>
 <row>
 <label style="padding-left:15px" value="Compaq HP XW4200"/>
 </row>
 </group>
 </row>
 </rows>
</grid>
```

```

 <label value="Intel Pentium 4"/>
 <label value="4GB RAM"/>
 <label value="$980" style="color:green"/>
 <label value="500GB"/>
 </row>
 <groupfoot spans="5">
 <label value="This a summary about Compaq Desktop PCs"/>
 </groupfoot>
</rows>
</grid>
</zk>
```

\*Available in ZK PE and EE only [13]

For more information, please take a look at these smalltalks,

- Learn About Grouping with Listbox and Grid
- About How Grouping Works with Live Data
- Add Summary Field For Grouping.

Or refer to Group component directly.

## Frozen Component

In ZK 5 you are now able to freeze columns within a Grid and Listbox. This mirrors functionality seen within Excel and makes data in these components easier to read, interpret and handle.

The following code demonstrates how to freeze a column within a Grid:

```

<grid>
 <frozen style="background: #dfded8" columns="3">
 ...
 </frozen>
</grid>
```

[Since 5.0.0]

- For further details, please refer to Frozen component directly.

## Custom Attributes

### **org.zkoss.zul.grid.rod**

[default: false]  
[inherit: true] [15]

It specifies whether to enable ROD (render-on-demand). For more information, please refer to ZK Developer's Reference: Performance Tips.

### **org.zkoss.zul.grid.autoSort**

[default: false]  
[inherit: true] [16]

[since 5.0.7]

Specifies whether to sort the model when the following cases:

- Grid.setModel(ListModel)<sup>[17]</sup> is called and Column.setSortDirection(String)<sup>[18]</sup> is set.
- Column.setSortDirection(String)<sup>[18]</sup> is called.
- Model receives ListDataEvent<sup>[19]</sup> and Column.setSortDirection(String)<sup>[18]</sup> is set.

If you want to ignore sort when receiving ListDataEvent<sup>[19]</sup>, you can specifies the value as **ignore.change**.

## org.zkoss.zul.grid.preloadSize

[default: 50]  
[inherit: true]<sup>[20]</sup>  
[since 6.0.1]

Specifies the number of rows to preload when receiving the rendering request from the client. It is used only if live data (Grid.setModel(ListModel)<sup>[17]</sup>) and not paging (Grid.getPagingChild()<sup>[21]</sup>).

## org.zkoss.zul.grid.initRodSize

[default: 50]  
[inherit: true]<sup>[22]</sup>  
[since 6.0.1]

Specifies the number of rows rendered when the Grid first render. It is used only if live data (Grid.setModel(ListModel)<sup>[17]</sup>) and not paging (Grid.getPagingChild()<sup>[21]</sup>).

## org.zkoss.zul.grid.autohidePaging

[default: true]  
[inherit: true]<sup>[23]</sup>  
[since: 7.0.1]

It specifies whether to enable autohide property for internal paging components.

- 
- [1] <http://www.zkoss.org/zkdemo/grid/>
  - [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Grid.html#>
  - [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/grid/Grid.html#>
  - [4] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Grid.html#setAutopaging\(boolean\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Grid.html#setAutopaging(boolean))
  - [5] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Row.html#>
  - [6] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/Components.html#>
  - [7] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/ListModel.html#>
  - [8] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/SimpleListModel.html#>
  - [9] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/RowRenderer.html#>
  - [10] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/ListModelExt.html#>
  - [11] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/event/ListDataListener.html#>
  - [12] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Grid.html#setSizedByContent\(boolean\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Grid.html#setSizedByContent(boolean))
  - [13] <http://www.zkoss.org/product/edition.dsp>
  - [14] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/GroupsModel.html#>
  - [15] The custom attribute could be specified in this component, or any of its ancestor. In addition, it could be specified as a library property to enable or disable it for the whole application.
  - [16] Same as above.
  - [17] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Grid.html#setModel\(ListModel\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Grid.html#setModel(ListModel))
  - [18] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Column.html#setSortDirection\(String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Column.html#setSortDirection(String))
  - [19] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/event/ListDataEvent.html#>

- [20] Same as above.
- [21] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Grid.html#getPagingChild\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Grid.html#getPagingChild())
- [22] Same as above.
- [23] Same as above.

## Supported Events

Name	Event Type
onAfterRender	<b>Event:</b> Event ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/Event.html#">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/Event.html#</a> ) Notifies one that the model's data has been rendered.
onPageSize	<b>Event:</b> PageSizeEvent ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/event/PageSizeEvent.html#">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/event/PageSizeEvent.html#</a> ) Notifies the paging size has been changed when the autopaging ( Grid.setAutopaging(boolean) ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Grid.html#setAutopaging(boolean)">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Grid.html#setAutopaging(boolean)</a> )) is enabled and user changed the size of the content.

- Inherited Supported Events: XulElement

## Supported Molds

Available molds of a component are defined in lang.xml embedded in zul.jar.

Name	Snapshot
default	
paging	

## Supported Children

\* Columns, Rows, Foot

## Use Cases

Version	Description	Example Location
---------	-------------	------------------

## Version History

Version	Date	Content
5.0.2	May 2010	Support the autopaging
5.0.4	July 2010	Support onAfterRender event
5.0.5	October 2010	The span property was introduced to span the columns to occupy the whole grid.
5.0.7	April 2011	Grid shall sort model based on current state.
5.0.7	April 2011	The emptyMessage attribute supported
5.0.7	April 2011	The onPageSize event was introduced.
5.0.8	June 2011	Deprecated setPreloadSize, instead with a custom attributes "org.zkoss.zul.grid.preloadSize".
5.0.8	June 2011	Add a custom attributes "org.zkoss.zul.grid.initRodSize" for control ROD render size.
6.5.0	June 2012	ZK-147 ( <a href="http://tracker.zkoss.org/browse/ZK-147">http://tracker.zkoss.org/browse/ZK-147</a> ): Support ungroup for grid's column menu
7.0.1	January 2014	ZK-2079 ( <a href="http://tracker.zkoss.org/browse/ZK-2079">http://tracker.zkoss.org/browse/ZK-2079</a> ): Add a custom attributes "org.zkoss.zul.grid.autohidePaging" for control autohide in internal paging component
7.0.2	April 2014	Due to the better user-firendly for the scrollbar layout, we changed the org.zkoss.zul.nativebar of the library property to true by default for Grid, Listbox, Tree and Borderlayout component.
7.0.3	July 2014	ZK-2359 ( <a href="http://tracker.zkoss.org/browse/ZK-2359">http://tracker.zkoss.org/browse/ZK-2359</a> ): Since ZK 7, the style class naming of autopaging has changed.
8.5.0	Oct 2017	ZK-3690 ( <a href="http://tracker.zkoss.org/browse/ZK-3690">http://tracker.zkoss.org/browse/ZK-3690</a> ): Added visibleRows property for Grid (the same as rows property of Listbox and Tree)
9.6.0	Mar 2021	ZK-4795 ( <a href="http://tracker.zkoss.org/browse/ZK-4795">http://tracker.zkoss.org/browse/ZK-4795</a> ): Grid/Listbox/Tree supports sticky column headers

# Column

## Column

- Demonstration: Grid (Simple Grid) [1]
- Java API: Column [2]
- JavaScript API: Column [3]
- Style Guide: Column

## Employment/Purpose

A single column in a Columns element. Each child of the Column element is placed in each successive cell of the grid. The column with the most child elements determines the number of rows in each column. The use of column is mainly to define attributes for each cell in the grid.

## Example

Grid Demo

Type	Content
File:	<input type="text"/>
Type:	<input type="text"/> Java Files,(*.java) <input type="button" value="Browse..."/>
Option:	<input type="text"/>

```
<window title="Grid Demo" border="normal" width="50%">
 <grid>
 <columns sizable="true">
 <column label="Type" hflex="min"/>
 <column label="Content" />
 </columns>
 <rows>
 <row>
 <label value="File:" />
 <textbox width="99%" />
 </row>
 <row>
 <label value="Type:" />
 <hbox>
 <listbox mold="select">
 <listitem label="Java Files, (*.java)" />
 </listbox>
 </hbox>
 </row>
 </rows>
 </grid>
</window>
```

```

 <listitem label="All Files, (*.*)" />
 </listbox>
 <button label="Browse..." />
</hbox>
</row>
<row>
 <label value="Options:" />
 <textbox rows="3" width="99%" />
</row>
</rows>
</grid>
</window>

```

## Supported Events

Name	Event Type
onSort	<b>Event:</b> SortEvent [4] Denotes user has sorted the row of this column.
onGroup	<b>Event:</b> SortEvent [4] <ul style="list-style-type: none"> <li>Available for ZK:</li> <li><b>CE PE EE</b></li> </ul> Denotes user has grouped all the cells under a column.
onUngroup	<b>Event:</b> SortEvent [4] <ul style="list-style-type: none"> <li>Available for ZK:</li> <li><b>CE PE EE</b></li> </ul> <p>[Since 6.5.0]</p> Denotes user has ungrouped all the cells under a column.

- Inherited Supported Events: HeaderElement

## Supported Children

\*ALL

## Use Cases

Grid

## Version History

Version	Date	Content
6.5.0	June 2012	ZK-147 [5]: Support ungroup for grid's column menu

## References

- [1] <http://www.zkoss.org/zkdemo/grid/simple>
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Column.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/grid/Column.html#>
- [4] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/SortEvent.html#>
- [5] <http://tracker.zkoss.org/browse/ZK-147>

# Columns

## Columns

- Demonstration: Grid (Simple Grid) <sup>[1]</sup>
- Java API: Columns <sup>[1]</sup>
- JavaScript API: Columns <sup>[2]</sup>
- Style Guide: Columns

## Employment/Purpose

Defines the columns of a grid.

Each child of a columns element should be a org.zkoss.zul.Column element.

## Example



```
<window title="Grid Demo" border="normal" width="360px">
 <zscript> class Comp implements Comparator { private boolean _asc;
 public Comp(boolean asc) { _asc = asc; } public int
 compare(Object o1,
 Object o2) { String s1 =
 o1.getChildren().get(0).getValue(), s2 =
 o2.getChildren().get(0).getValue(); int v =
 s1.compareTo(s2); return
 _asc ? v: -v; } } Comp asc = new Comp(true), dsc = new
 Comp(false);
```

```
</zscript>
<grid>
 <columns sizable="true">
 <column label="Type" sortAscending="asc" sortDescending="dsc" width="50px"/>
 <column label="Content" />
 </columns>
 <rows>
 <row>
 <label value="File:>" />
 <textbox width="99%" />
 </row>
 <row>
 <label value="Type:>" />
 <hbox>
 <listbox rows="1" mold="select">
 <listitem label="Java Files, (*.java)" />
 <listitem label="All Files, (*.*)" />
 </listbox>
 <button label="Browse..." />
 </hbox>
 </row>
 <row>
 <label value="Options:>" />
 <textbox rows="3" width="99%" />
 </row>
 </rows>
</grid>
</window>
```

## Properties

### Menupopup

By default, the *none* is assumed, you can specify the *auto* to appear a default menu for the columns. Or you can provide your own menupopup for the component.

#### Auto

Author	Title
Philip Hensher	Group
Philip Hensher	Sort Ascending
Philip Hensher	Sort Descending
Michael Greenberg	Author
Michael Greenberg	Title
Rick Perlstein	Publisher
Rick Perlstein	Hardcover
	Nixonland

```
<grid>
 <columns menupopup="auto">
 <column label="Author" sort="auto"/>
 <column label="Title" sort="auto"/>
 <column label="Publisher" sort="auto"/>
 <column label="Hardcover" sort="auto"/>
 </columns>
```

#### Customized Menupopup

Type	Type1
File:	Group
File:	Sort Ascending
File:	Sort Descending
File:	File1s

```
<window title="Column's Menu Demo" border="normal" width="500px">
 <menupopup id="editPopup">
 <menuitem label="Group" image="~/zul/img/grid/menu-group.png"/>
 <menuitem label="Sort Ascending" image="~/zul/img/grid/menu-arrowup.png"/>
 <menuitem label="Sort Descending" image="~/zul/img/grid/menu-arrowdown.png"/>
 </menupopup>
</window>
```

```

<columns sizable="true" menupopup="editPopup">
 <column id="col" label="Type" sortAscending="asc" sortDescending="dsc"/>
 <column id="col1" label="Type1" sortAscending="asc" sortDescending="dsc"/>
 <column id="col2" label="Content"/>
</columns>
</grid>
</window>

```

As you can see, the example above specify a customized menupopup to the columns as its column menu.

## Sizable

Specifies whether the user is allowed to size a column of a grid (by dragging the vertical bar between two adjacent columns).

Notice that, in addition to dragging, the end user could double-click on the vertical bar between two adjacent columns, such that the grid will automatically resize the column to fit its contents. In other words, all sizable column provides the auto-fitting functionality.

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: HeadersElement

## Supported Children

\* Column

## Use Cases

Grid

## Version History

Version	Date	Content
---------	------	---------

## References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Columns.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/grid/Columns.html#>

# Detail

## Detail

- Demonstration: Grid (Master detail) <sup>[1]</sup>
- Java API: Detail <sup>[2]</sup>
- JavaScript API: Detail <sup>[3]</sup>
- Style Guide: Detail
- Available for ZK:
- CE PE EE

## Employment/Purpose

The detail component is used to display a detailed section where a master row and multiple detail rows are on the same row.

## Example

Please open/close the +/- button, and the layout of this page shows properly.

	Product Name	Price	Item location
<input type="button" value="-"/>	Apple 20-inch Aluminum Cinema Display M9177/A	US \$202.50	tulsa, ok, United States

**Item Specifics - Item Condition**

Condition: **Used**

Brand: **Apple**

Technology: **DVI**

Monitor Type: **LCD/Flat Panel**



```
<?xml version="1.0" encoding="UTF-8"?>
<zkb>
 Please open/close the +/- button, and the layout of this page
 shows
 properly.

 <grid fixedLayout="true" width="600px">
 <columns>
 <column width="40px" />
 <column>Product Name</column>
 <column>Price</column>
 <column>Item location</column>
 </columns>
 </grid>

```

```

<rows>

<row>
 <detail>
 <hlayout>
 <image width="200px" height="200px" src="/img/icon_update.png" />
 <vlayout>
 <label value="Item Specifics - Item Condition" style="font-weight:bold;font-style: italic;" />
 <hlayout>
 <label value="Condition:>" />
 <label value="Used" style="font-weight:bold;" />
 </hlayout>
 <hlayout>
 <label value="Brand:>" />
 <label value="Apple" style="font-weight:bold;" />
 </hlayout>
 <hlayout>
 <label value="Technology:>" />
 <label value="DVI" style="font-weight:bold;" />
 </hlayout>
 <hlayout>
 <label value="Monitor Type:>" />
 <label value="LCD/Flat Panel" style="font-weight:bold;" />
 </hlayout>
 </vlayout>
 </detail>
 <label value="Apple 20-inch Aluminum Cinema Display M9177/A" />
 <label style="color:green;float:right;" value="US $202.50" />
 <label value="tulsa, ok, United States" />
</row>
</rows>
</grid>
</zk>

```

## Supported Events

Name	Event Type
onOpen	<b>Event:</b> OpenEvent [3] Denotes user has opened or closed a component. Note: unlike onClose, this event is only a notification. The client sends this event after opening or closing the component. It is useful to implement load-on-demand by listening to the onOpen event, and creating components when the first time the component is opened.

- Inherited Supported Events: XulElement

## Supported Children

\*ALL

## Use Cases

Version	Description	Example Location
---------	-------------	------------------

## Version History

Version	Date	Content
---------	------	---------

## References

- [1] [http://www.zkoss.org/zkdemo/grid/master\\_detail](http://www.zkoss.org/zkdemo/grid/master_detail)
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Detail.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zkex/grid/Detail.html#>

## Foot

---

### Foot

- Demonstration: Grid (Header and footer) <sup>[1]</sup>
- Java API: Foot <sup>[2]</sup>
- JavaScript API: Foot <sup>[3]</sup>
- Style Guide: Footer

## Employment/Purpose

Defines a set of footers ( Footer) for a grid ( Grid).

## Example

Type	Content
File:	<input type="text"/>
Type:	<input type="text" value="Java Files (*.java)"/> <span style="border: 1px solid #ccc; padding: 2px;">Browse...</span>
footer	footer2

```
<grid width="300px">
 <columns>
 <column label="Type" width="50px"/>
 <column label="Content"/>
```

```
</columns>
<rows>
 <row>
 <label value="File:>" />
 <textbox width="99%" />
 </row>
 <row>
 <label value="Type:>" />
 <hbox>
 <listbox rows="1" mold="select">
 <listitem label="Java Files, (*.java)" />
 <listitem label="All Files, (*.*)" />
 </listbox>
 <button label="Browse..." />
 </hbox>
 </row>
</rows>
<foot>
 <footer>footer1</footer>
 <footer>footer2</footer>
</foot>
</grid>
```

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

## Supported Children

\* Footer

## Use Cases

Grid

## Version History

Version	Date	Content
---------	------	---------

## References

- [1] [http://www.zkoss.org/zkdemo/grid/header\\_and\\_footer](http://www.zkoss.org/zkdemo/grid/header_and_footer)
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Foot.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/grid/Foot.html#>

## Footer

---

### Footer

- Demonstration: Grid (Header and footer) <sup>[1]</sup>
- Java API: Footer <sup>[1]</sup>
- JavaScript API: Footer <sup>[2]</sup>
- Style Guide: Footer

### Employment/Purpose

A column of the footer of a grid ( Grid). Its parent must be Foot.

Unlike Column, you could place any child in a grid footer.

### Example

Type	Content
File:	<input type="text"/>
Type:	<input type="text" value="Java Files (*.java)"/> <span style="border: 1px solid #ccc; padding: 2px;">▼</span> <span style="border: 1px solid #ccc; padding: 2px; background-color: #f0f0f0;">Browse...</span>
footer	footer2

```

<grid width="300px">
 <columns>
 <column label="Type" width="50px"/>
 <column label="Content"/>
 </columns>
 <rows>
 <row>
 <label value="File:"/>

```

```
<textbox width="99%"/>
</row>
<row>
 <label value="Type :"/>
 <hbox>
 <listbox rows="1" mold="select">
 <listitem label="Java Files, (*.java)"/>
 <listitem label="All Files, (*.*)"/>
 </listbox>
 <button label="Browse..."/>
 </hbox>
</row>
</rows>
<foot>
 <footer>footer1</footer>
 <footer>footer2</footer>
</foot>
</grid>
```

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: FooterElement

## Supported Children

\*ALL

## Use Cases

Grid

## Version History

Version	Date	Content
---------	------	---------

## References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Footer.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/grid/Footer.html#>

# Group

---

## Group

- Demonstration: Group <sup>[1]</sup>
- Java API: Group <sup>[2]</sup>
- JavaScript API: Group <sup>[3]</sup>
- Style Guide: Group
- Available for ZK:
- CE PE EE

## Employment/Purpose

Adds the ability for single level grouping to the Grid.

Default getSclass(): the same as grid's sclass.

## Example

Grid support Groupfoot in Group

Type	Content
▲ Group1: (gp1)	<b>Group1:</b>
File:	File:
Type:	Java Files (*.java) <input type="button" value="Browse..."/>
<b>2 Java Files</b>	<b>10 Files</b>
▲ Group 2 (gp2)	
Options:	Options:
<b>2 Options</b>	<b>10 Options</b>

```
<?xml version="1.0" encoding="UTF-8"?>
<zkb>
 Grid support Groupfoot in Group

 <grid id="grid" width="500px">
 <columns id="h" sizable="true">
 <column id="col1" label="Type"/>
 <column id="col2" label="Content"/>
 </columns>
 <rows id="rows">
 <group id="gp1">
 <label value="Group1: (gp1)" />
 <label value="Group1:" />
 </group>
 </rows>
 </grid>
</zkb>
```

```

</group>
<row>
 <label value="File:"/>
 <label value="File:"/>
</row>
<row id="row1">
 <label value="Type:"/>
 <hbox>
 <listbox rows="1" mold="select">
 <listitem label="Java Files, (*.java)"/>
 <listitem label="All Files, (*.*)"/>
 </listbox>
 <button label="Browse..."/>
 </hbox>
</row>
<groupfoot>
 <label value="2 Java Files"/>
 <label value="10 Files"/>
</groupfoot>
<group id="gp2" label="Group 2 (gp2)" onOpen='alert("Group is open: "+self.open);'>
<row>
 <label value="Options:"/>
 <label value="Options:"/>
</row>
<groupfoot>
 <label value="2 Options"/>
 <label value="10 Options"/>
</groupfoot>
</rows>
</grid>
</zk>

```

## Supported Events

Name	Event Type
onOpen	<b>Event:</b> OpenEvent <sup>[3]</sup> Denotes user has opened or closed a component. Note: unlike onClose, this event is only a notification. The client sends this event after opening or closing the component. It is useful to implement load-on-demand by listening to the onOpen event, and creating components when the first time the component is opened.

- Inherited Supported Events: Row

## Supported Children

\* ALL

## Use Cases

Grid

## Version History

Version	Date	Content

## References

- [1] <http://www.zkoss.org/zkdemo/grid/grouping>
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Group.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zkex/grid/Group.html#>

## Groupfoot

---

### Groupfoot

- Demonstration: Group <sup>[1]</sup>
- Java API: Groupfoot <sup>[1]</sup>
- JavaScript API: Groupfoot <sup>[2]</sup>
- Style Guide: Groupfoot
- Available for ZK:
- CE PE EE

## Employment/Purpose

Adds the ability for single level grouping to the Grid. Default getSclass(): the same as grid's sclass.

## Example

Grid support Groupfoot in Group

Type	Content
▲ Group1: (gp1)	<b>Group1:</b>
File:	File:
Type:	Java Files,(*.java) ▾ <b>Browse...</b>
<b>2 Java Files</b>	<b>10 Files</b>
▲ Group 2 (gp2)	
Options:	Options:
<b>2 Options</b>	<b>10 Options</b>

```
<?xml version="1.0" encoding="UTF-8"?>
<zkb>
 Grid support Groupfoot in Group

 <grid id="grid" width="500px">
 <columns id="h" sizable="true">
 <column id="col1" label="Type"/>
 <column id="col2" label="Content"/>
 </columns>
 <rows id="rows">
 <group id="gp1">
 <label value="Group1: (gp1)" />
 <label value="Group1:" />
 </group>
 <row>
 <label value="File:" />
 <label value="File:" />
 </row>
 <row id="row1">
 <label value="Type:" />
 <hbox>
 <listbox rows="1" mold="select">
 <listitem label="Java Files, (*.java)" />
 <listitem label="All Files, (*.*)" />
 </listbox>
 <button label="Browse..." />
 </hbox>
 </row>
 </rows>
 </grid>
</zkb>
```

```

</row>
<groupfoot>
 <label value="2 Java Files"/>
 <label value="10 Files"/>
</groupfoot>
<group id="gp2" label="Group 2 (gp2)" onOpen='alert("Group is open: "+self.open);'>
<row>
 <label value="Options:"/>
 <label value="Options:"/>
</row>
<groupfoot>
 <label value="2 Options"/>
 <label value="10 Options"/>
</groupfoot>
</rows>
</grid>
</zk>

```

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: Listitem

## Supported Children

\*ALL

## Use Cases

Version	Description	Example Location

## Version History

Version	Date	Content

## References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Groupfoot.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkex/grid/Groupfoot.html#>

# Row

## Row

- Demonstration: Grid (Simple Grid) [1]
- Java API: Row [5]
- JavaScript API: Row [1]
- Style Guide: Row

## Employment/Purpose

A single row in a Rows element. Each child of the Row element is placed in each successive cell of the grid. The row with the most child elements determines the number of columns in each row.

Default getScss(): the same as grid's scss.

## Example

Grid Demo

Type	Content
File:	<input type="text"/>
Type:	<input type="text"/> Java Files (*.java) ▾ <input type="button" value="Browse..."/>
Option:	<input type="text"/>

```
<window title="Grid Demo" border="normal" width="360px">
<zscript>
 class Comp implements Comparator {
 private boolean _asc;
 public Comp(boolean asc) {
 _asc = asc;
 }
 public int compare(Object o1, Object o2) {
 String s1 = o1.getChildren().get(0).getValue(),
 s2 = o2.getChildren().get(0).getValue();
 int v = s1.compareTo(s2);
 return _asc ? v: -v;
 }
 }
 Comp asc = new Comp(true), dsc = new Comp(false);
</zscript>
```

```

<grid>
 <columns sizable="true">
 <column label="Type" sortAscending="${asc}"
 sortDescending="${dsc}" width="50px"/>
 <column label="Content" />
 </columns>
 <rows>
 <row>
 <label value="File:" />
 <textbox width="99%" />
 </row>
 <row>
 <label value="Type:" />
 <hbox>
 <listbox rows="1" mold="select">
 <listitem label="Java Files, (*.java)" />
 <listitem label="All Files, (*.*)" />
 </listbox>
 <button label="Browse..." />
 </hbox>
 </row>
 <row>
 <label value="Options:" />
 <textbox rows="3" width="99%" />
 </row>
 </rows>
</grid>
</window>

```

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

## Supported Children

\*ALL

## Use Cases

Version	Description	Example Location
---------	-------------	------------------

## Version History

Version	Date	Content
---------	------	---------

## References

[1] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/grid/Row.html#>

# Rows

---

## Rows

- Demonstration: Grid (Simple Grid) <sup>[1]</sup>
- Java API: Rows <sup>[1]</sup>
- JavaScript API: Rows <sup>[2]</sup>
- Style Guide: Rows

## Employment/Purpose

Defines the rows of a grid. Each child of a rows element should be a Row <sup>[5]</sup> element.

## Example

Grid Demo

Type	Content
File:	<input type="text"/>
Type:	Java Files (*.java) <input type="button" value="Browse..."/>
Option:	<input type="text"/>

```
<window title="Grid Demo" border="normal" width="360px">
<zscript>
 class Comp implements Comparator {
 private boolean _asc;
```

```
public Comp(boolean asc) {
 _asc = asc;
}
public int compare(Object o1, Object o2) {
 String s1 = o1.getChildren().get(0).getValue(),
 s2 = o2.getChildren().get(0).getValue();
 int v = s1.compareTo(s2);
 return _asc ? v : -v;
}
}
Comp asc = new Comp(true), dsc = new Comp(false);
</zscript>
<grid>
 <columns sizable="true">
 <column label="Type" sortAscending="#{asc}" sortDescending="#{dsc}" width="50px"/>
 <column label="Content" />
 </columns>
 <rows>
 <row>
 <label value="File:" />
 <textbox width="99%" />
 </row>
 <row>
 <label value="Type:" />
 <hbox>
 <listbox rows="1" mold="select">
 <listitem label="Java Files, (*.java)" />
 <listitem label="All Files, (*.*)" />
 </listbox>
 <button label="Browse..." />
 </hbox>
 </row>
 <row>
 <label value="Options:" />
 <textbox rows="3" width="99%" />
 </row>
 </rows>
</grid>
</window>
```

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

## Supported Children

\* Row, Group, Groupfoot

## Use Cases

Version	Description	Example Location

## Version History

Version	Date	Content

## References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Rows.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/grid/Rows.html#>

# Listbox

## Listbox

- Demonstration: Listbox <sup>[1]</sup>
- Java API: Listbox <sup>[2]</sup>
- JavaScript API: Listbox <sup>[3]</sup>
- Style Guide: Listbox

## Employment/Purpose

Components: `listbox`, `listitem`, `listcell`, `listhead` and `listheader`.

A list box is used to display a number of items in a list. The user may select an item from the list. Although `listhead` is optional, if it exists, notice that the number of `listheader` should equal the number of `listcell`, so that `listbox` can display its content correctly. If `listhead` contains no `listheader`, the `listbox` will display nothing in its content.

## Example

listbox demo	
name	gender
Mary	FEMALE
John	MALE
Jane	FEMALE
Henry	MALE
This is footer1	This is footer2

```
<window title="listbox demo" border="normal" width="250px">
 <listbox id="box">
 <listhead sizable="true">
 <listheader label="name" sort="auto" />
 <listheader label="gender" sort="auto" />
 </listhead>
 <listitem>
 <listcell label="Mary" />
 <listcell label="FEMALE" />
 </listitem>
 <listitem>
 <listcell label="John" />
 <listcell label="MALE" />
 </listitem>
 <listitem>
 <listcell label="Jane" />
 <listcell label="FEMALE" />
 </listitem>
 </listbox>
</window>
```

```

<listitem>
 <listcell label="Henry" />
 <listcell label="MALE" />
</listitem>
<listfoot>
 <listfooter>
 <label value="This is footer1" />
 </listfooter>
 <listfooter>
 <label value="This is footer2" />
 </listfooter>
</listfoot>
</listbox>
</window>

```

## Listboxes Contain Buttons

In theory, a list cell can contain any other components, as demonstrated below.

Population	Percentage
A. Graduate	20%
B. College	<input checked="" type="checkbox"/> 23%
C. High School	<input type="text" value="40%"/>

```

<zk>
 <listbox width="250px">
 <listhead>
 <listheader label="Population"/>
 <listheader label="Percentage"/>
 </listhead>
 <listitem value="A">
 <listcell><textbox width="90%" value="A. Graduate"/></listcell>
 <listcell label="20%"/>
 </listitem>
 <listitem value="B">
 <listcell><checkbox label="B. College"/></listcell>
 <listcell><button label="23%"/></listcell>
 </listitem>
 <listitem value="C">
 <listcell label="C. High School"/>
 <listcell><textbox cols="8" value="40%"/></listcell>
 </listitem>
 </listbox>
</zk>

```

Notes:

1. Don't use a list box, when a grid is a better choice. The appearances of list boxes and grids are similar, but the listbox should only be used to represent a list of selectable items.
2. Users are usually confused if a listbox contains editable components such as a `textbox` or a `checkbox`.
3. Due to the limitation of the browsers, users cannot select a range of characters from text boxes.

## Mold

The Listbox has two molds: `default` and `select`.

### Select Mold

If the `select` mold is used, Listbox renders an HTML `<select>` tag instead.

Matthew ▾

```
<listbox mold="select">
 <listitem label="Matthew"/>
 <listitem label="Macus"/>
 <listitem label="Lucas"/>
 <listitem label="John"/>
</listbox>
```

Note: if the `mold` is "select", `rows` is "1", and none of the items are marked as selected, the browser will display the `listbox` as if the first item is selected. Worst of all, if user selects the first item in this case, no `onSelect` event is sent. To avoid this confusion, developers should select at least one item when using `mold="select"` and `rows="1"`.

In addition to each items label, you can assign an application-specific value to each item using the `setValue` method.

Listgroups are allowed in this mold.



```
<listbox mold="select">
 <listgroup label="Support">
 <listitem label="Matthew"/>
 <listitem label="Macus"/>
 </listgroup>
 <listgroup label="R&D">
 <listitem label="Lucas"/>
 <listitem label="John"/>
 </listgroup>
</listbox>
```

**Notice:** Listbox doesn't send `onClick` event for `listitem` under this mold.

## Mouseless Entry Listbox

- Press UP and DOWN to move the selection up and down by one list item.
- Press PgUp and PgDn to move the selection up and down by one page.
- Press HOME to move the selection to the first item, and END to move to the last item.
- Press Ctrl+UP and Ctrl+DOWN to move the focus up and down by one list item without changing the selection.
- Press SPACE to select the item in focus.

## Paging

Like grids, you can use multiple pages to represent large content by setting the mold to paging. Similarly, you can control how many items each page displays, whether to use an external paging component and whether to customize the behavior when a page is selected.

The `listbox` and `grid` components support the paging intrinsically, so you don't need to specify a paging component explicitly as above unless you want to have different visual layout or to control multiple `listbox` and `gridcontrols` with one paging component.

Please refer to the Grid for more details.

## Autopaging

When using the paging mold and vflex, you could also turn on autopaging (`Listbox.setAutopaging(boolean)`<sup>[4]</sup>) such that the page size will be adjusted automatically based on the available space.

**Note:** If the autopaging is enabled, the height of each row will be applied the following CSS by default. If you want to change the height, please overwrite the CSS rule as your preference. But this feature only works when each row has the same height.

```
.z-listbox-autopaging .z-listcell-cnt {
 height: 30px;
 overflow: hidden;
}
```

**Note:** In ZK 7, we change the naming `.z-listcell-cnt` to `.z-listcell-content`.

```
.z-listbox-autopaging .z-listitem {
 height: 80px; /* set custom height */
}
.z-listbox-autopaging .z-listcell-content {
 height: auto; /* remove the default height */
/*
 max-height: 58px; /* limit the height to avoid long
text increasing the height */
}
```

## PagingDisabled

Once the `pagingDisabled` is set to `true`, users will be blocked from navigating through the pagination.

## Selection

### Nonselectable Tags

By default, when a user clicks on an HTML `<button>`, `<input>`, `<textarea>` or `<a>` tag, it doesn't change the selection. For example, when a user clicks the textbox in the following example, the selection state of the item won't be changed (only the textbox gains the focus).

```
<listitem>
 <listcell>
 <textbox/>
 </listcell>
</listitem>
```

### Click to Select Anyway

Sometimes it is not intuitive, such as using with inplace editing (`InputElement.isInplace()`<sup>[5]</sup>). If you want to have more control of whether to select an item, you could specify a list of tags in the `nonselectableTags` property (`Listbox.setNonselectableTags(java.lang.String)`<sup>[6]</sup>). For example, if you want to select the item, no matter what tag the user clicks, you could specify an empty string as follows.

```
<listbox nonselectableTags="">
 <listitem><listcell><textbox/></listcell></listitem>
 <listitem><listcell><button label="button"/></listcell></listitem>
 <listitem><listcell><h:input xmlns:h="native"/></listcell></listitem>
 <listitem><listcell><datebox/></listcell></listitem>
</listbox>
```

If you only want to ignore BUTTON and INPUT only, you could specify:

```
<!-- The tag here means HTML tag, not ZUL tag -->
<listbox nonselectableTags="button, input"/>
```

### Click Checkmark to Select Only

If you want to toggle the selection only when the user clicks on the checkmark, you could specify `*`. Notice that you have to specify `checkmark="true"` as well (otherwise, no item is selectable).

This setting also allows to select and copy a text in a listcell with `ctrl+c`.

```
<listbox checkmark="true" nonselectableTags="*>
 <listitem>
 <listcell>
 you can copy the text with ctrl+c
 </listcell>
```

```
</listitem>
</listbox>
```

## Multiple Selection

When a user clicks on a list item, the whole row is selected and the `onSelect` event is sent back to the server to notify the application. You are able to enable multiple selections by setting the `multiple` attribute to true. The default value is `false`.

If there is no checkmark (by default `checkmark="false"`), then **click to select one item will deselect others**, just like you select a file in a file browser in a OS.

To select multiple items, you can:

- press `Ctrl` to select separate items:

Name	Gender	Age
Mary	FEMALE	18
John	MALE	20
Jane	FEMALE	32
Henry	MALE	29
Mark	MALE	15
Jeffery	MALE	17
Rebecca	FEMALE	21

- press `Shift` to select consecutive items:

Name	Gender	Age
Mary	FEMALE	18
John	MALE	20
Jane	FEMALE	32
Henry	MALE	29
Mark	MALE	15
Jeffery	MALE	17
Rebecca	FEMALE	21

## Enable with a ListModel

If you assign a ListModel to a Listbox, then you should enable the multiple selection with the **ListModel**

. Please **do not** set **multiple** on listbox directly, and set **multiple** on the model instead.

```
...
List Items = new ArrayList();
for (int i = 0; i < 1000; i++) {
 Items.add("data "+i);
}

ListModelList model = new ListModelList(Items);
model.setMultiple(true);
...

<listbox model="${model}" ... />
```

## The Checkmark Property

The `checkmark` attribute controls whether to display a checkbox or a radio button in front of each list item.

Population	Percentage
A. Graduate	20%
B. College	23%
C. High School	40%
D. Others	17%

=> <=>

<input type="checkbox"/>	Population	Percentage
<input type="checkbox"/>	E. Supermen	21%

In the following example, you will notice how a checkbox is added automatically when you move a list item from the left listbox to the right one. The checkbox is then removed when you move a list item from the right listbox to the left listbox.

```
<zkb>
<hbox>
 <listbox id="src" rows="0" multiple="true" width="200px">
 <listhead>
 <listheader label="Population"/>
 <listheader label="Percentage"/>
 </listhead>
 <listitem id="a" value="A">
 <listcell label="A. Graduate"/>
 <listcell label="20%"/>
 </listitem>
 <listitem id="b" value="B">
 <listcell label="B. College"/>
 <listcell label="23%"/>
 </listitem>
 <listitem id="c" value="C">
 <listcell label="C. High School"/>
 </listitem>
 </listbox>
 <listbox id="tgt" rows="0" multiple="true" width="200px">
 <listitem id="d" value="D">
 <listcell label="D. Others"/>
 <listcell label="17%"/>
 </listitem>
 </listbox>
</hbox>
```

```

 <listcell label="40%"/>
 </listitem>
 <listitem id="d" value="D">
 <listcell label="D. Others"/>
 <listcell label="17%"/>
 </listitem>
</listbox>
<vbox>
 <button label="=>" onClick="move(src, dst)"/>
 <button label="=<" onClick="move(dst, src)"/>
</vbox>
<listbox id="dst" checkmark="true" rows="0" multiple="true" width="200px">
 <listhead>
 <listheader label="Population" width="120px"/>
 <listheader label="Percentage"/>
 </listhead>
 <listitem id="e" value="E">
 <listcell label="E. Supermen"/>
 <listcell label="21%"/>
 </listitem>
</listbox>
<zscript>
 void move(Listbox src, Listbox dst) {
 Listitem s = src.getSelectedItem();
 if (s == null)
 Messagebox.show("Select an item first");
 else
 s.setParent(dst);
 }
</zscript>
</hbox>
</zk>
```

Population	Percentage
<input checked="" type="radio"/> E. Supermen	21%
<input checked="" type="radio"/> B. College	23%

Note: If the `multiple` attribute is false, radio buttons are displayed instead, as demonstrated by the right hand listbox.

To enable Select all feature, there are some constraints as below:

Select all checkbox in listheader is only available if ROD is false.

If paging mold is enabled, the Select all checkbox in listheader is only available when using a ListModel implementation as model. (e.g. ListModelList)

The Select all checkbox on listheader now support onCheckSelectAll event that can determine whether it is checked or not.

```
<listbox checkmark="true" multiple="true" width="350px">
 <custom-attributes org.zkoss.zul.listbox.rod="false"/>
 <attribute name="onCheckSelectAll"><![CDATA[
 if (event.isChecked()) {
 System.out.println("Select All Checked");
 } else {
 System.out.println("Select All Un-Checked");
 }
]]></attribute>
 <listhead>
 <listheader label="col 1" />
 <listheader label="col 2" />
 </listhead>
 <listitem id="a" value="A">
 <listcell label="A. Graduate"/>
 <listcell label="20%"/>
 </listitem>
 <listitem id="b" value="B">
 <listcell label="B. College"/>
 <listcell label="23%"/>
 </listitem>
</listbox>
```

## Deselect Others when Clicking an Item with Checkmark

If a listbox's checkmark (Listbox.isCheckmark() [7]) is set to **true**, the selection will be toggled when an user clicks an item. In other words, all other items will remain their selection state.

If you prefer to **deselect all other items** and select the item being clicked (which the behavior of ZK 5.0.4 and earlier), you could specify true to this library property called `org.zkoss.zul.listbox.checkmarkDeselectOthers` in WEB-INF/zk.xml:

```
<library-property>
 <name>org.zkoss.zul.listbox.checkmarkDeselectOthers</name>
 <value>true</value>
</library-property>
```

## Toggle Selection when Right Clicking an Item with Checkmark

If a listbox's checkmark (Listbox.isCheckmark()<sup>[7]</sup>) is enabled, the selection will be toggled when user right click on item.

If you prefer not to select/deselect item on right click, you could specify false to this library property called org.zkoss.zul.listbox.rightSelect in WEB-INF/zk.xml:

```
<library-property>
 <name>org.zkoss.zul.listbox.rightSelect</name>
 <value>false</value>
</library-property>
```

## Sorting

Listboxes support the sorting of list items directly. When you enable sorting, a user can click a Listheader to switch the sorting order between **ascending** and **descending**. There are a few ways to enable the sorting of a particular column.

### Sort

The simplest way is sort="auto". Then, when a user clicks a listheader, listbox sorts the column based on the 'label' of each listcell in a **case-insensitive** way.

name	gender
Henry	MALE
Jane	FEMALE
John	MALE
Mary	FEMALE

```
<zk>
 <listbox width="200px">
 <listhead>
 <listheader label="name" sort="auto"/>
 <listheader label="gender" sort="auto"/>
 </listhead>
 ...
 </listbox>
</zk>
```

## Auto-sorting on Fields

If `ListModel` contains non-String object, you need to specify its property to sort. By default, it sorts in a case-sensitive way with `FieldComparator`<sup>[8]</sup>.

In the following example, we demonstrate how to sort a person object based on its First Name, Last Name, or Age.

Full Name	Age
Tom Cheng	43
Henri Smith	41
Jim Xavier	39

```
<zscript>
<! [CDATA[

class Person {
 private String firstName;
 private String lastName;
 private int age;

 public Person(String f, String l, int a) {
 firstName = f;
 lastName = l;
 age = a;
 }

 public String getFirstName() {
 return firstName;
 }
 public String getLastname() {
 return lastName;
 }
 public String getFullName() {
 return firstName + " " + lastName;
 }
 public int getAge() {
 return age;
 }
}

ListModelList persons = new ListModelList();
persons.add(new Person("Tom", "Cheng", 43));
persons.add(new Person("Henri", "Smith", 41));
```

```
persons.add(new Person("Jim", "Xavier", 39));
]]>
</zscript>
<listbox model="${persons}">
 <listhead>
 <listheader label="Full Name" sort="auto(lastName, firstName)" />
 <listheader label="Age" sort="auto(age)" />
 </listhead>
 <template name="model">
 <listitem>
 <listcell label="${each.fullName}" />
 <listcell label="${each.age}" />
 </listitem>
 </template>
</listbox>
```

## Case-insensitive

To sort in case-insensitive, you can apply one of the functions below:

- UPPER()
- LOWER()

```
<listheader label="First Name" sort="auto(UPPER(firstName))" />
```

## The SortAscending and SortDescending Properties

If you prefer to sort list items in different ways, you can assign a `java.util.Comparator` instance to the `sortAscending` and/or `sortDescending` attributes. Once assigned, the list items can be sorted in the ascending and/or descending order with the specified comparator.

The invocation of the `sort` attribute with `auto` automatically assigns two comparators to the `sortAscending` and `sortDescending` attributes. You can override any of them by assigning another comparator.

For example, assume you want to sort based on the value of list items, rather than list cell's label, then you assign an instance of `ListitemComparator` to these attributes as follows.

```
<zscript>
 import org.zkoss.zul.*;
 Comparator asc = new ListitemComparator(-1, true, true);
 Comparator dsc = new ListitemComparator(-1, false, true);
</zscript>
<listbox width="200px" model="${model}">
 <listhead>
 <listheader label="ID" sortAscending="${asc}" sortDescending="${dsc}" />
 </listhead>
</listbox>
```

## The SortDirection Property

The `sortDirection` attribute determines **the display of the sorting direction icon** to indicate "ascending" or "descending". It **doesn't sort** the data. If list items are sorted before adding to the listbox, you should set this attribute explicitly.

```
<listheader sortDirection="ascending"/>
```

Sorting is maintained automatically by the listboxes as long as you assign the comparator to the corresponding list header.

## The onSort Event

When you assign at least one comparator to a `Listheader`, an `onSort` [9] event is sent to the server if users clicks on it. The `Listheader` implements a listener to sort.

If you prefer to handle sorting manually, you can add your own listener to a `Listheader` for the `onSort` event. To prevent the default listener invoking the `sort` method, you have to call the `stopPropagation` method. Alternatively, you can override the `sort` method, please see below.

## The Sort Method

The `sort` method is the underlying implementation of the default `onSort` event listener. It is also useful if you want to sort the list items using Java code. For example, you may have to call this method after adding items (assuming that they are not added in the proper order).

```
new Listem("New Stuff").setParent(listbox);
if (!"natural".header.getSortDirection())
 header.sort("ascending").equals(header.getSortDirection()));
```

The default sorting algorithm is quick-sort (by use of the `sort` method from the `org.zkoss.zk.ui.Components` class). You can override it with your own implementation or listen to the `onSort` event as described in the previous section.

**Tip:** Sorting a large amount of live data could degrade the performance significantly. It is better to intercept the `onSort` event or the `sort` method to handle it effectively. Please refer to the **Sort Live Data** section further down.

## Live Data

Like grid<sup>[10]</sup>, listbox supports *live data*. With live data, developers can separate data from the view. In other words, developers need only to provide the data by implementing the `ListModel` [7] interface, rather than manipulating the list box directly.

The benefits are twofold:

- It is easier to use different views to display the same set of data.
- The list box sends the data to the client only if it is visible. This saves a lot of network traffic if there is a large amount of data.

There are three steps to make use of live data.

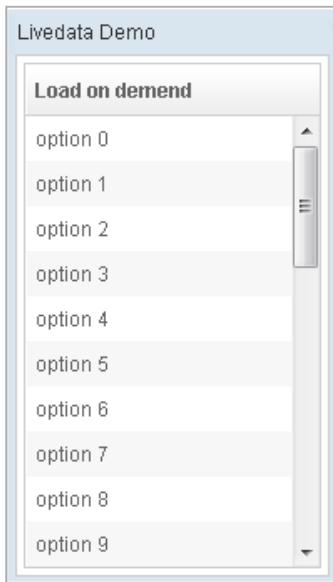
1 Prepare the data in the form of a `ListModel` [7]. ZK has a concrete implementation called `SimpleListModel` [8] for representing an array of objects.

2 Implement the `ListitemRenderer` [11] interface for rendering a item of data into the listbox.

- This is optional. If it is not specified the default renderer is used to render the data into the first column.

- You can implement different renderers for representing the same data in different views.

3 Set the data in the `model` attribute and, optionally, the renderer in the `itemRenderer` attribute.



In the following example, we prepared a list model called `strset`, assigned it to a list box through the `model` attribute. Then, the listbox will do the rest.

```
<window title="Livedata Demo" border="normal" width="200px">
<zscript><! [CDATA[
 String[] data = new String[30];
 for(int j=0; j < data.length; ++j) {
 data[j] = "option "+j;
 }
 ListModel strset = new SimpleListModel(data);
]]></zscript>
<listbox rows="10" model="${strset}">
 <listhead>
 <listheader label="Load on demand"/>
 </listhead>
</listbox>
</window>
```

- 
- [1] <http://www.zkoss.org/zkdemo/listbox>
  - [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Listbox.html#>
  - [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/sel>Listbox.html#>
  - [4] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Listbox.html#setAutopaging\(boolean\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Listbox.html#setAutopaging(boolean))
  - [5] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/impl/InputElement.html#isInplace\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/impl/InputElement.html#isInplace())
  - [6] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul>Listbox.html#setNonselectableTags\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul>Listbox.html#setNonselectableTags(java.lang.String))
  - [7] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul>Listbox.html#isCheckmark\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul>Listbox.html#isCheckmark())
  - [8] <https://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/FieldComparator.html>
  - [9] <https://www.zkoss.org/javadoc/7.0.3/zk/org/zkoss/zk/ui/event/SortEvent.html>
  - [10] The concept is similar to Swings (`javax.swing.ListModel`).
  - [11] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/ListItemRenderer.html#>

## Sorting with Live Data

If you allow users to sort a listbox with live data, you have to implement the interface, Sortable (<http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/ext/Sortable.html#>), in addition to the ListModel (<http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/ListModel.html#>).

```
class MyListModel implements ListModel, Sortable {
 public void sort(Comparator cmpr, boolean ascending) {
 //do the real sorting
 //notify the listbox (or grid) that data is changed by use
 of ListDataEvent
 }
 ...
}
```

When a user wants to sort the listbox, the listbox will invoke sort(java.util.Comparator,boolean) ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/ext/Sortable.html#sort\(java.util.Comparator,boolean\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/ext/Sortable.html#sort(java.util.Comparator,boolean))) to sort the data. In other words, the sorting is done by the list model, rather than the listbox.

After sorting, the list model will notify the listbox by invoking the ListDataListener.onChange(org.zkoss.zul.event.ListDataEvent) ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/event/ListDataListener.html#onChange\(org.zkoss.zul.event.ListDataEvent\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/event/ListDataListener.html#onChange(org.zkoss.zul.event.ListDataEvent))) method of the listbox' registered ListDataListener (<http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/event/ListDataListener.html#>) instances. These are registered by ListModel.addListDataListener(org.zkoss.zul.event.ListDataListener) ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/ListModel.html#addListDataListener\(org.zkoss.zul.event.ListDataListener\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/ListModel.html#addListDataListener(org.zkoss.zul.event.ListDataListener))). In most cases, all the data is changed, so the list model usually sends the following event:

```
new ListDataEvent(this, ListDataEvent.CONTENT_CHANGED, -1, -1)
```

**Note:** the implementation of the ListModel (<http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/ListModel.html#>) and Sortable (<http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/ext/Sortable.html#>) is independent of the visual presentation. In other words, they can be used with grids, listboxes and any other components supporting ListModel (<http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/ListModel.html#>).

If you require maximum flexibility, you should not depend on the actual component used, and instead use ListDataEvent (<http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/event>ListDataEvent.html#>) to communicate model changes.

## Scroll a Listitem into Current View

When a Listbox is scrollable, if you want to scroll a Listitem out of the visible area into the current view (visible area), you can call scrollToIndex() (<https://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Listbox.html#scrollToIndex-int>).

## Properties

### Single-Column Listboxes

The simplest format is as follows. It is a single-column and single-selection list box.

Butter Pecan
Chocolate Chip
Raspberry Ripple

```
<zk>
 <listbox width="200px">
 <listitem label="Butter Pecan"/>
 <listitem label="Chocolate Chip"/>
 <listitem label="Raspberry Ripple"/>
 </listbox>
</zk>
```

### Multi-Column Listboxes

The list box also supports multiple columns. When a user selects an item, the entire row is selected.

To define a multi-column list, the number of listcells must match the number of columns with a row. For example if there are 4 columns then each row must contain 4 listcells.

George	House Painter
Mary Ellen	Candle Maker
Roger	Swashbuckler

```
<zk>
 <listbox width="200px">
 <listitem>
 <listcell label="George"/>
 <listcell label="House Painter"/>
 </listitem>
 <listitem>
 <listcell label="Mary Ellen"/>
 <listcell label="Candle Maker"/>
 </listitem>
 <listitem>
 <listcell label="Roger"/>
 <listcell label="Swashbuckler"/>
 </listitem>
 </listbox>
</zk>
```

```
</listbox>
</zk>
```

## Column Headers

You can specify column headers by using `listhead` and `listheader`, please see the code below<sup>[1]</sup>. In addition to label, you can specify an image as the header by use of the `image` attribute.

Name	Occupation
George	House Painter
Mary Ellen	Candle Maker
Roger	Swashbuckler

```
<zk>
 <listbox width="200px">
 <listhead>
 <listheader label="Name"/>
 <listheader label="Occupation"/>
 </listhead>

 ...
 </listbox>
</zk>
```

[1] This feature is a bit different from XUL, where `listhead` and `listheader` are used.

## Column Footers

You could specify the column footers by using `listfoot` and `listfooter`. Please note, each time a `listhead` instance is added to a list box, it must be the first child, and a `listfoot` instance the last child.

Population	%
A. Graduate	20%
B. College	23%
C. High School	40%
D. Others	17%
More or less	100%

```
<zk>
 <listbox width="200px">
 <listhead>
 <listheader label="Population"/>
 <listheader align="right" label="%"/>
 </listhead>
 <listitem id="a" value="A">
 <listcell label="A. Graduate"/>
 </listitem>
 </listbox>
</zk>
```

```
 <listcell label="20%"/>
 </listitem>
 <listitem id="b" value="B">
 <listcell label="B. College"/>
 <listcell label="23%"/>
 </listitem>
 <listitem id="c" value="C">
 <listcell label="C. High School"/>
 <listcell label="40%"/>
 </listitem>
 <listitem id="d" value="D">
 <listcell label="D. Others"/>
 <listcell label="17%"/>
 </listitem>
 <listfoot>
 <listfooter label="More or less"/>
 <listfooter label="100%"/>
 </listfoot>
</listbox>
</zk>
```

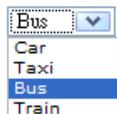
## Auxiliary Headers

Like grids, you can specify auxiliary headers with the `auxhead` and `auxheader` components.

Please refer to the Grid for more details.

## Drop-Down List

You can create a drop-down list by setting the listbox's mold to select and making the box a single row. Notice you cannot use multi-column for the drop-down list.



```
<zk>
 <listbox mold="select" rows="1">
 <listitem label="Car"/>
 <listitem label="Taxi"/>
 <listitem label="Bus" selected="true"/>
 <listitem label="Train"/>
 </listbox>
</zk>
```

## Scollable Listboxes

A list box will be scrollable if it has a defined or automatically calculated height (e.g. by specifying the rows, height or vflex attribute) and there is not enough space to display all the list items.

Name	Gender
Mary	FEMALE
John	MALE
Jane	FEMALE
Henry	MALE

```
<zk>
<listbox width="250px" rows="4">
 <listhead>
 <listheader label="Name" sort="auto"/>
 <listheader label="Gender" sort="auto"/>
 </listhead>
 <listitem>
 <listcell label="Mary"/>
 <listcell label="FEMALE"/>
 </listitem>
 <listitem>
 <listcell label="John"/>
 <listcell label="MALE"/>
 </listitem>
 <listitem>
 <listcell label="Jane"/>
 <listcell label="FEMALE"/>
 </listitem>
 <listitem>
 <listcell label="Henry"/>
 <listcell label="MALE"/>
 </listitem>
 <listitem>
 <listcell label="Michelle"/>
 <listcell label="FEMALE"/>
 </listitem>
</listbox>
</zk>
```

The browser's default scrollbar is replaced by floating scrollbar and it is not visible unless user mouse over on the content. To turn off the floating scrollbar and use original scrollbar, please add the following configuration in zk.xml.

```
<library-property>
 <name>org.zkoss.zul.nativebar</name>
 <value>true</value>
```

```
</library-property>
```

Note: the value of org.zkoss.zul.nativebar is true by default ( )

## SizedByContent

By default, the widths of columns have to be specified explicitly, or it will be split equally among columns regardless what content they might have. If you want to have the minimal width (that fit the content), you could specify hflex="min" at the column (not the listbox).

However, the listbox has a special mode called sized-by-content ( Listbox.setSizedByContent(boolean) ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Listbox.html#setSizedByContent\(boolean\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Listbox.html#setSizedByContent(boolean)))). By specifying it to true, the column width will be adjusted automatically. However, it is controlled by the browser, so you will have no 100% control of it. For example, if an user resized a column, the final width might not be exactly the same as what he resized.

In general, we suggest to specify hflex in column, rather than specifying sizedByContent at listbox for much more predictable result.

## Span

By default, when sizedByContent is true, column only take required space.

Time	Message	Level	Source	Message
6/28/10 4:19:18 PM	Info, long content.....	Server	Merging recovery point 52 created 20 6/27/10 10 :11 PM	
6/28/10 4:19:18 PM	Info, long content.....	Server	Merging recovery point 52	
6/28/10 4:19:18 PM	Info, long content.....	Server	Merging recovery point 52	

If wanna to span the width of the columns to occupy the whole listbox, you could specify true to this attribute

Time	Message	Level	Source	Message
6/28/10 4:19:18 PM	Info, long content.....	Server	Merging recovery point 52 created 20 6/27/10 10 :11 PM	
6/28/10 4:19:18 PM	Info, long content.....	Server	Merging recovery point 52	
6/28/10 4:19:18 PM	Info, long content.....	Server	Merging recovery point 52	

```
<listbox sizedByContent="true" span="true" width="800px">
 <listhead>
 <listheader label="Time Message" />
 <listheader label="Level" />
 <listheader label="Source" />
 <listheader label="Message" />
 </listhead>
 <listitem>
 <listcell label="6/28/10 4:19:18 PM" />
 <listcell label="Info, long content....." />
 <listcell label="Server" />
 <listcell label="Merging recovery point 52 created 20 6/27/10 10 :11 PM" />
 </listitem>
</listbox>
```

**Equally Distributed: sizedByContent="false" without specified width**

default	default	default
Row 1	Hello World	Hello World

**Partially Fixed: sizedByContent="false" with specified width**

Fixed	Flexed	Flexed
Row 1	Hello World	Hello World

**Partially Fixed: fixed width with hflex**

Fixed	Proportional Flex 1/3	Proportional Flex 2/3
Row 1	Hello World	Hello World

**sizedByContent="true" span="false"**

Fixed	Flexed	Flexed
Row 1	Hello World	Hello World

**Fully Auto-Width: sizedByContent="true" span="true", specified width on a listheader has no effect**

Fixed	Flexed	Flexed
Row 1	Hello World	Hello World

## Rows

The `rows` attribute is used to control how many rows are visible. By setting it to zero, the list box will resize itself to hold as many as items if possible.

## Vflex

The `vflex` property controls whether the listbox will grow or shrink vertically to fit the given space. It is named vertical flexibility. For example, if the list is too big to fit in the browser window, its height will decrease to make the whole list control visible in the browser window.

This property is ignored if the `rows` attribute is specified.

## Show messages when empty

The `emptyMessage` attribute is used to show a message when we have no items.

```
<listbox id="test1" emptyMessage="No items match your search">

 <listhead sizable="true">
 <listheader label="Type" width="520px" />
 <listheader label="Content" hflex="min" />
 <listheader label="Content" hflex="1" />
 </listhead>
</listbox>
```

## Maxlength

The `maxlength` property defines the maximum number of characters visible at the browser. By setting this attribute, you are able to create a narrower list box.

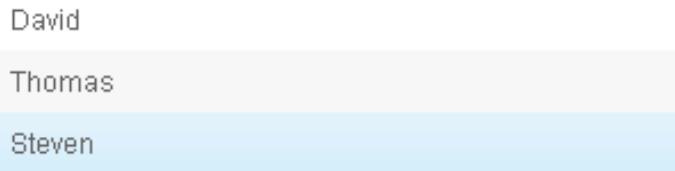
## Sizable

Like `columns`, you can set the `sizable` attribute of the `listhead` to `true` to allow users to resize the width of list headers. The `onColSize` event is also sent when a user resizes listbox.

## Auto Fitting Columns

When you want to resize a column of a Grid or Listbox, all you now need to do is double click the column when the mouse is over where the columns meet and the column will automatically resize to fit its contents. To enable this functionality Listbox's Listhead need the attribute `sizable="true"`. In other words, all sizable column provides the auto-fitting functionality.

## The `onAfterRender` Event



```
<zk>
<zscript><![CDATA[
 ListModelList lm = new ListModelList(Arrays.asList(new
String[] { "David",
 "Thomas", "Steven" }));
]]></zscript>

<listbox width="300px" model="${lm}" onAfterRender="self.setSelectedIndex(2)"/>
</zk>
```

## Sticky Header

After adding a sclass "z-sticky-header", when we scroll down a page and make a Listbox's header out of visible range in a viewport, the Listbox's header becomes floating and sticky on the top of the page.

```
<listbox sclass="z-sticky-header">
 <!-- listhead, listitem... -->
</listbox>
```

## Columns Menu

For example,

The screenshot shows a ZK Listbox component with two columns: 'Author' and 'Title'. The 'Title' column has a context menu open, listing options: 'Group', 'Sort Ascending', 'Sort Descending', 'Author' (selected), 'Title' (selected), 'Publisher' (selected), and 'Hardcover' (selected). The menu also includes a 'More...' option at the bottom.

Author	Title
Philip Hensher	
Philip Hensher	
Philip Hensher	
Michael Greenberg	
Michael Greenberg	
Rick Perlstein	
Rick Perlstein	

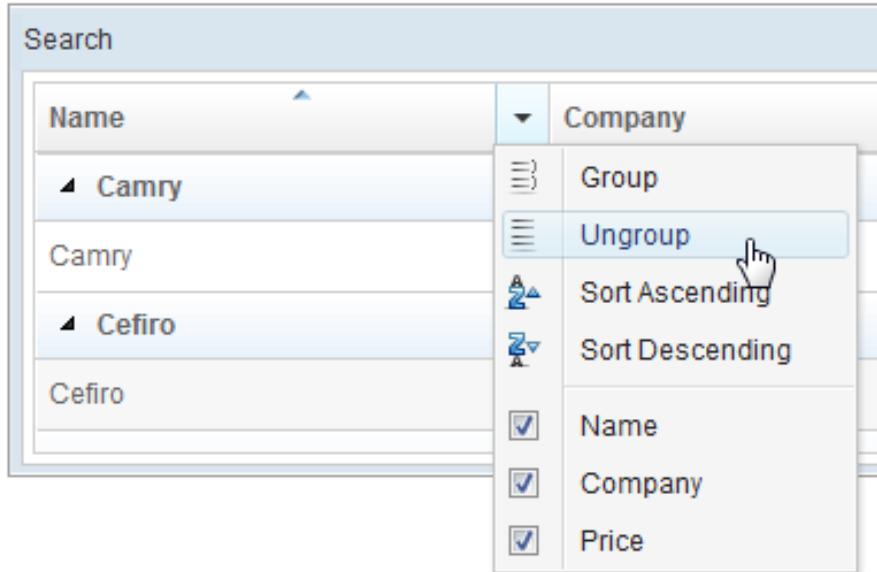
```
<zk>
 <listbox>
 <listhead menupopup="auto">
 <listheader label="Author" sort="auto"/>
 <listheader label="Title" sort="auto"/>
 <listheader label="Publisher" sort="auto"/>
 <listheader label="Hardcover" sort="auto"/>
 </listhead>
 // omitted...
 </listbox>
</zk>
```

- For further details, please refer to Listhead component directly.

## Ungroup Column Menu

When the user groups the content of the listbox, the column's menu will show an ungroup icon for user to reset the group.

[ZK EE]



**Note:** If the Listbox contains with Model, *GroupsModel*, you have to register an *onUngroup* event for listheader to show an ungroup icon and then replace the current model with a *ListModel* to reset the group.

For example,

```
<zk>
<zscript><! [CDATA[
int cnt = 0;
Object[][] foods = new Object[][] {
 new Object[] { "Vegetables", "Asparagus", "Vitamin K", 115, 43},
 new Object[] { "Vegetables", "Beets", "Folate", 33, 74},
 new Object[] { "Vegetables", "Tomatoes", "Vitamin C", 57, 37},
 new Object[] { "Seafood", "Salmon", "Tryptophan", 103, 261},
 new Object[] { "Seafood", "Cod", "Tryptophan", 90, 119}
};
public class FoodGroupRenderer implements ListitemRenderer {
 public void render(Listitem row, Object obj, int index) {
 if (row instanceof Listgroup) {
 row.setLabel(obj.toString());
 } else {
 Object[] data = (Object[]) obj;
 row.appendChild(new Listcell(data[0].toString()));
 row.appendChild(new Listcell(data[1].toString()));
 row.appendChild(new Listcell(data[2].toString()));
 row.appendChild(new Listcell(data[3].toString()));
 row.appendChild(new Listcell(data[4].toString()));
 }
 }
}]>
```

```

 }

 }

ListModelList listmodel = new ListModelList();
for (int i = 0; i < foods.length; i++)
 listmodel.add(foods[i]);
ListitemRenderer renderer = new FoodGroupRenderer();
GroupsModel model = new GroupsModelArray(foods, new ArrayComparator(0,
true));
]]></zscript>
<listbox id="listbox" model="${model}" itemRenderer="${renderer}">
 <listhead menupopup="auto">
 <listheader label="Category" sort="auto(0)" onGroup='listbox.setModel(model) '
onUngroup='listbox.setModel(listmodel);' />
 <listheader label="Name" sort="auto(1)" />
 <listheader label="Top Nutrients" sort="auto(2)" />
 <listheader label="% of Daily" sort="auto(3)" />
 <listheader label="Calories" sort="auto(4)" />
 </listhead>
</listbox>
</zk>
```

## Listgroup Component

Both Grid, and Listbox support Grouping concept, it enables developers to display data in an advanced way. Moreover, live data are also supported in Grouping Grid, and Listbox with the GroupsModel (<http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/GroupsModel.html#>) interface..

Brand	Processor Type	Memory (RAM)	Price	Hard Drive Capacity
<b>Dell</b>				
Dell E4500 2.2GHz	Intel Core 2 Duo	2GB RAM	\$261.00	500GB
XP-Pro Slim Dell-Inspiron-530-s	Intel Core 2 Duo	2GB RAM	\$498.93	500GB
Dell P4 3.2 GHz	Intel Pentium 4	4GB RAM	\$377.99	500GB
<b>Compaq</b>				
Compaq SR5113WM	Intel Core Duo	1GB RAM	\$279.00	160GB
Compaq HP XW4200	Intel Pentium 4	4GB RAM	\$980	500GB
This a summary about Compaq Desktop PCs				

```

<zk>
 <listbox>
 <listhead sizable="true">
 <listheader label="Brand"/>
 <listheader label="Processor Type" width="150px"/>
 <listheader label="Memory (RAM)" width="120px"/>
 <listheader label="Price" width="100px"/>
 <listheader label="Hard Drive Capacity" width="150px"/>
 </listhead>
 <listgroup label="Dell"/>
 <listitem>
 <listcell style="padding-left:15px" label="Dell E4500 2.2GHz"/>
 </listitem>
 </listbox>
</zk>
```

```

<listcell label="Intel Core 2 Duo"/>
<listcell label="2GB RAM"/>
<listcell label="$261.00" style="color:green"/>
<listcell label="500GB"/>
</listitem>
</listitem>
<listcell style="padding-left:15px" label="XP-Pro Slim Dell-Inspiron-530-s"/>
<listcell label="Intel Core 2 Duo"/>
<listcell label="2GB RAM"/>
<listcell label="$498.93" style="color:green"/>
<listcell label="500GB"/>
</listitem>
</listitem>
<listcell style="padding-left:15px" label="Dell P4 3.2 GHz"/>
<listcell label="Intel Pentium 4"/>
<listcell label="4GB RAM"/>
<listcell label="$377.99" style="color:green"/>
<listcell label="500GB"/>
</listitem>
<listgroup label="Compaq"/>
<listitem>
<listcell style="padding-left:15px" label="Compaq SR5113WM"/>
<listcell label="Intel Core Duo"/>
<listcell label="1GB RAM"/>
<listcell label="$279.00" style="color:green"/>
<listcell label="160GB"/>
</listitem>
<listitem>
<listcell style="padding-left:15px" label="Compaq HP XW4200"/>
<listcell label="Intel Pentium 4"/>
<listcell label="4GB RAM"/>
<listcell label="$980" style="color:green"/>
<listcell label="500GB"/>
</listitem>
<listgroupfoot>
<listcell span="5" label="This a summary about Compaq Desktop PCs"/>
</listgroupfoot>
</listbox>
</zk>

```

\* Available in ZK PE and EE only (<http://www.zkoss.org/product/edition.dsp>)

For more information, please take a look at these smalltalks,

- Learn About Grouping with Listbox and Grid
- About How Grouping Works with Live Data
- Add Summary Field For Grouping.

Or refer to Listgroup component directly.

## Frozen Component

In ZK 5 you are now able to freeze columns within a Grid and Listbox. This mirrors functionality seen within Excel and makes data in these components easier to read, interpret and handle.

The following code demonstrates how to freeze a column within a Grid:

```
<listbox>
 <listhead>
 <listheader label="header 1"/>
 <listheader label="header 2"/>
 <listheader label="header 3"/>
 <listheader label="header 4"/>
 </listhead>
 <frozen columns="2"/>
 <listitem>
 <listcell label="cell 1"/>
 <listcell label="cell 2"/>
 <listcell label="cell 3"/>
 <listcell label="cell 4"/>
 </listitem>
</listbox>
```

- For further details, please refer to Frozen component directly.

## Custom Attributes

### org.zkoss.zul.listbox.rightSelect

```
[default: true]
[inherit: true][1]
```

It specifies that the selection will be toggled when user right clicks on an item, if the checkmark is enabled (`Listbox.isCheckmark()` ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Listbox.html#isCheckmark\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Listbox.html#isCheckmark()))).

If it is turned off, right clicking on an item won't change its selection state.

### org.zkoss.zul.listbox.groupSelect

```
[default: false]
[inherit: true][2]
```

It specifies whether Listgroups are selectable under this Listbox. (Similar to above, it can also be specified as a library property, which will be in effect for the whole application.)

## org.zkoss.zul.listbox.autoSort

```
[default: false]
[inherit: true][3]
```

Specifies whether to sort the model when the following cases:

- Listbox.setModel(ListModel) ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Listbox.html#setModel\(ListModel\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Listbox.html#setModel(ListModel))) is called and Listheader.setSortDirection(String) ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul>Listheader.html#setSortDirection\(String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul>Listheader.html#setSortDirection(String))) is set.
- Listheader.setSortDirection(String) ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul>Listheader.html#setSortDirection\(String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul>Listheader.html#setSortDirection(String))) is called.
- Model receives ListDataEvent (<http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/event>ListDataEvent.html#>) and Listheader.setSortDirection(String) ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul>Listheader.html#setSortDirection\(String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul>Listheader.html#setSortDirection(String))) is set.

If you want to ignore sort when receiving ListDataEvent (<http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/event>ListDataEvent.html#>), you can specifies the value as **ignore.change**.

## org.zkoss.zul.listbox.rod

```
[default: false]
[inherit: true][4]
```

It specifies whether to enable ROD (render-on-demand). For more information, please refer to ZK Developer's Reference: Performance Tips.

## org.zkoss.zul.listbox.preloadSize

```
[default: 50]
[inherit: true][5]
```

It specifies the number of items to preload when receiving the rendering request from the client. It is used only if live data ( Listbox.setModel(ListModel) ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul>Listbox.html#setModel\(ListModel\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul>Listbox.html#setModel(ListModel)))) and not paging ( Listbox.getPagingChild() ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul>Listbox.html#getPagingChild\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul>Listbox.html#getPagingChild()))).

## org.zkoss.zul.listbox.initRodSize

```
[default: 50]
[inherit: true][6]
```

Specifies the number of items rendered when the Listbox first render. It is used only if live data ( Listbox.setModel(ListModel) ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul>Listbox.html#setModel\(ListModel\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul>Listbox.html#setModel(ListModel)))) and not paging ( Listbox.getPagingChild() ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul>Listbox.html#getPagingChild\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul>Listbox.html#getPagingChild()))).

## org.zkoss.zul.listbox.autohidePaging

```
[default: true]
[inherit: true][7]
```

It specifies whether to enable autohide property for internal paging component.

- [1] The custom attribute could be specified in this component, or any of its ancestor. In addition, it could be specified as a library property to enable or disable it for the whole application.
- [2] Same as above.
- [3] Same as above.
- [4] Same as above.
- [5] Same as above.
- [6] Same as above.
- [7] Same as above.

## Supported Events

Name	Event Type
onSelect	<b>Event:</b> SelectEvent ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event&gt;SelectEvent.html#">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event&gt;SelectEvent.html#</a> ) Notifies one that the user has selected a new item in the listbox.
onFocus	<b>Event:</b> Event ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event</a> #) Denotes when a component gets the focus. Remember event listeners execute at the server, so the focus at the client might be changed when the event listener for onFocus got executed.
onBlur	<b>Event:</b> Event ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event</a> #) Denotes when a component loses the focus. Remember event listeners execute at the server, so the focus at the client might be changed when the event listener for onBlur got executed.
onAfterRender	<b>Event:</b> Event ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event</a> #) Notifies one that the model's data has been rendered.
onPageSize	<b>Event:</b> PageSizeEvent ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/event">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/event</a> #) Notifies the paging size has been changed when the autopaging ( Listbox.setAutopaging(boolean) ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul&gt;Listbox.html#setAutopaging(boolean)">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul&gt;Listbox.html#setAutopaging(boolean)</a> )) is enabled and user changed the size of the content.
onCheckSelectAll	<b>Event:</b> CheckEvent ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event</a> #) (Since 6.5.6) Notifies the checkbox on a listheader is checked to select all checkable items.

- Inherited Supported Events: XulElement

## Supported Molds

Available molds of a component are defined in lang.xml embedded in zul.jar.

Name	Snapshot												
default	<table border="1"> <thead> <tr> <th>name</th><th>gender</th></tr> </thead> <tbody> <tr> <td>Mary</td><td>FEMALE</td></tr> <tr> <td>John</td><td>MALE</td></tr> <tr> <td>Jane</td><td>FEMALE</td></tr> <tr> <td>Henry</td><td>MALE</td></tr> <tr> <td>This is footer1</td><td>This is footer2</td></tr> </tbody> </table>	name	gender	Mary	FEMALE	John	MALE	Jane	FEMALE	Henry	MALE	This is footer1	This is footer2
name	gender												
Mary	FEMALE												
John	MALE												
Jane	FEMALE												
Henry	MALE												
This is footer1	This is footer2												
select	<div style="border: 1px solid #ccc; padding: 5px; width: fit-content;"> <p>Mary</p> <p>Mary</p> <p>John</p> <p>Jane</p> <p>Henry</p> </div>												
paging	<table border="1"> <thead> <tr> <th>name</th><th>gender</th></tr> </thead> <tbody> <tr> <td>Mary</td><td>FEMALE</td></tr> <tr> <td>John</td><td>MALE</td></tr> <tr> <td>This is footer1</td><td>This is footer2</td></tr> </tbody> </table> <p style="text-align: center;"> <span>◀</span> <span>◀</span> <span>1</span> /2 <span>▶</span> <span>▶</span> [1 - 2 / 4]     </p>	name	gender	Mary	FEMALE	John	MALE	This is footer1	This is footer2				
name	gender												
Mary	FEMALE												
John	MALE												
This is footer1	This is footer2												

## Supported Children

Listitem, Listhead, Listfoot, Listgroup, Listgroupfoot

## Use Cases

Version	Description	Example Location
---------	-------------	------------------

## Version History

Version	Date	Content
5.0.2	May 2010	Support the autopaging
5.0.4	July 2010	Support onAfterRender event
5.0.5	September 2010	The nonselectabletag property was introduced to enhance the control of when to select an item
5.0.5	September 2010	When a listbox's checkmark is enabled and an item is clicked, it will toggle the selection of the item and the other remains the same.
5.0.5	October 2010	When a listbox's checkmark is enabled and an item is right clicked, it will toggle the selection of the item.
5.0.5	October 2010	The span property was introduced to span the columns to occupy the whole listbox.
5.0.6	February 2011	The nonselectableTags property supported "*".
5.0.7	April 2011	Listbox shall sort model based on current state.

5.0.7	April 2011	The emptyMessage attribute supported
5.0.7	April 2011	The onPageSize event was introduced.
5.0.8	June 2011	Deprecated setPreloadSize, instead with a custom attributes "org.zkoss.zul.listbox.preloadSize".
5.0.8	June 2011	Add a custom attributes "org.zkoss.zul.listbox.initRodSize" for control ROD render size.
5.0.11	February 2012	ZK-873 ( <a href="http://tracker.zkoss.org/browse/ZK-873">http://tracker.zkoss.org/browse/ZK-873</a> ): Select all checkbox in listheader is only available if ROD is false.
6.5.0	June 2012	ZK-120 ( <a href="http://tracker.zkoss.org/browse/ZK-120">http://tracker.zkoss.org/browse/ZK-120</a> ): Provide menupopup="auto" for listbox
6.5.0	June 2012	ZK-147 ( <a href="http://tracker.zkoss.org/browse/ZK-147">http://tracker.zkoss.org/browse/ZK-147</a> ): Support ungroup for grid's column menu
7.0.1	January 2014	ZK-2079 ( <a href="http://tracker.zkoss.org/browse/ZK-2079">http://tracker.zkoss.org/browse/ZK-2079</a> ): Add a custom attributes "org.zkoss.zul.listbox.autohidePaging" for control autohide in internal paging component
7.0.2	April 2014	Due to the better user-firendly for the scrollbar layout, we changed the org.zkoss.zul.nativebar of the library property to true by default for Grid, Listbox, Tree and Borderlayout component.
7.0.3	July 2014	ZK-2359 ( <a href="http://tracker.zkoss.org/browse/ZK-2359">http://tracker.zkoss.org/browse/ZK-2359</a> ): Since ZK 7, the style class naming of autopaging has changed.
8.6.0	Oct 2018	ZK-2756 ( <a href="http://tracker.zkoss.org/browse/ZK-2756">http://tracker.zkoss.org/browse/ZK-2756</a> ): Listbox supports listgroup like optgroup in select mold
9.6.0	Mar 2021	ZK-4795 ( <a href="http://tracker.zkoss.org/browse/ZK-4795">http://tracker.zkoss.org/browse/ZK-4795</a> ): Grid/Listbox/Tree supports sticky column headers

# Listcell

---

## Listcell

- Demonstration: Listbox (Keystroke Command) <sup>[1]</sup>
- Java API: Listcell <sup>[2]</sup>
- JavaScript API: Listcell <sup>[3]</sup>
- Style Guide: Listitem

## Employment/Purpose

A list cell.

## Example

Listbox

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: LabelImageElement

## Supported Children

\* ALL

## Use Cases

Version	Description	Example Location
---------	-------------	------------------

## Version History

Version	Date	Content
---------	------	---------

## References

- [1] [http://www.zkoss.org/zkdemo/listbox/keystroke\\_command](http://www.zkoss.org/zkdemo/listbox/keystroke_command)
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Listcell.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/sel>Listcell.html#>

## Listfoot

---

### Listfoot

- Demonstration: N/A
- Java API: Listfoot <sup>[1]</sup>
- JavaScript API: Listfoot <sup>[2]</sup>
- Style Guide: Listfooter

## Employment/Purpose

Like Listhead, each listbox has at most one Listfoot.

## Example

name	gender
Mary	FEMALE
John	MALE
Jane	FEMALE
Henry	MALE
This is footer1	This is footer2

```
<window title="listbox demo" border="normal" width="250px">
 <listbox id="box">
 <listhead sizable="true">
```

```
<listheader label="name" sort="auto"/>
<listheader label="gender" sort="auto"/>
</listhead>
<listitem>
 <listcell label="Mary"/>
 <listcell label="FEMALE"/>
</listitem>
<listitem>
 <listcell label="John"/>
 <listcell label="MALE"/>
</listitem>
<listitem>
 <listcell label="Jane"/>
 <listcell label="FEMALE"/>
</listitem>
<listitem>
 <listcell label="Henry"/>
 <listcell label="MALE"/>
</listitem>
<listfoot>
 <listfooter><label value="This is footer1"/></listfooter>
 <listfooter><label value="This is footer2"/></listfooter>
</listfoot>
</listbox>
</window>
```

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

## Supported Children

\* Listfooter

## Use Cases

Version	Description	Example Location
---------	-------------	------------------

## Version History

Version	Date	Content
---------	------	---------

## References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Listfoot.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/sel>Listfoot.html#>

## Listfooter

---

### Listfooter

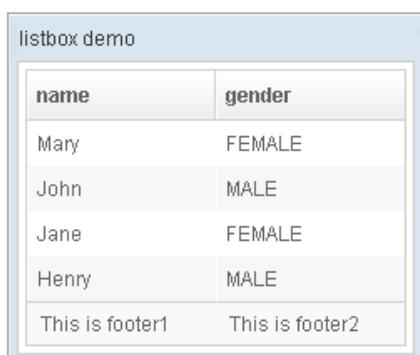
- Demonstration: N/A
- Java API: Listfooter <sup>[1]</sup>
- JavaScript API: Listfooter <sup>[2]</sup>
- Style Guide: Listfooter

### Employment/Purpose

A column of the footer of a list box (Listbox). Its parent must be Listfoot. Unlike Listheader, you could place any child in a list footer.

Note: Listcell also accepts children.

### Example



```
<window title="listbox demo" border="normal" width="250px">
 <listbox id="box">
 <listhead sizable="true">
 <listheader label="name" sort="auto"/>
 <listheader label="gender" sort="auto"/>
```

```
</listhead>
<listitem>
 <listcell label="Mary"/>
 <listcell label="FEMALE"/>
</listitem>
<listitem>
 <listcell label="John"/>
 <listcell label="MALE"/>
</listitem>
<listitem>
 <listcell label="Jane"/>
 <listcell label="FEMALE"/>
</listitem>
<listitem>
 <listcell label="Henry"/>
 <listcell label="MALE"/>
</listitem>
<listfoot>
 <listfooter><label value="This is footer1"/></listfooter>
 <listfooter><label value="This is footer2"/></listfooter>
</listfoot>
</listbox>
</window>
```

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: FooterElement

## Supported Children

\*ALL

## Use Cases

Version	Description	Example Location
---------	-------------	------------------

## Version History

Version	Date	Content
---------	------	---------

## References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Listfooter.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/sel>Listfooter.html#>

## Listgroup

### Listgroup

- Demonstration: N/A
- Java API: Listgroup <sup>[1]</sup>
- JavaScript API: Listgroup <sup>[2]</sup>
- Style Guide: Listgroup
- Available for ZK:
- CE PE EE

### Employment/Purpose

Adds the ability for single level grouping to the Listbox.

### Example

Listbox support Grouping

name	gender
▷ Group1	Group2
◀ Grouping 2	
Jane	FEMALE
Henry	MALE

```
<?xml version="1.0" encoding="UTF-8"?>
<zkc>
 Listbox support Grouping
 <listbox id="listbox" width="250px">
 <listhead sizable="true" id="h">
 <listheader id="h1" label="name" sort="auto" />
 <listheader id="h2" label="gender" sort="auto" />
 </listhead>
```

```

<listgroup id="gp1" open="false">
 <listcell label="Group1"/>
 <listcell label="Group2"/>
</listgroup>
<listitem>
 <listcell label="a Mary" />
 <listcell label="a FEMALE" />
</listitem>
<listitem>
 <listcell label="b Mary" />
 <listcell label="b FEMALE" />
</listitem>
<listitem id="li1">
 <listcell label="c Mary1" />
 <listcell label="c FEMALE1" />
</listitem>
<listitem>
 <listcell label="d Mary" />
 <listcell label="d FEMALE" />
</listitem>
<listitem>
 <listcell label="e John" />
 <listcell label="e MALE" />
</listitem>
<listgroup id="g2" label="Grouping 2" />
<listitem>
 <listcell label="Jane" />
 <listcell label="FEMALE" />
</listitem>
<listitem>
 <listcell label="Henry" />
 <listcell label="MALE" />
</listitem>

</listbox>
</zk>

```

## Supported Events

Name	Event Type
onOpen	<b>Event:</b> OpenEvent <sup>[3]</sup> Denotes user has opened or closed a component. Note: unlike onClose, this event is only a notification. The client sends this event after opening or closing the component. It is useful to implement load-on-demand by listening to the onOpen event, and creating components when the first time the component is opened.

- Inherited Supported Events: Listitem

## Supported Children

\* Listcell

## Use Cases

Version	Description	Example Location
---------	-------------	------------------

## Version History

Version	Date	Content
---------	------	---------

## References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Listgroup.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkex/sel>Listgroup.html#>

## Listgroupfoot

---

### Listgroupfoot

- Demonstration: N/A
- Java API: Listgroupfoot <sup>[1]</sup>
- JavaScript API: Listgroupfoot <sup>[2]</sup>
- Style Guide: Listgroupfoot
- Available for ZK:
- CE PE EE

## Employment/Purpose

GroupFooter serves as a summary listitem of listgroup.

## Example

Listbox support Grouping	
name	gender
▷ Group1	Group2
◀ Grouping 2	
Jane	FEMALE
Henry	MALE

```
<?xml version="1.0" encoding="UTF-8"?>
<zk>
 Listbox support Grouping
 <listbox id="listbox" width="250px">
```

```
<listhead sizable="true" id="h">
 <listheader id="h1" label="name" sort="auto" />
 <listheader id="h2" label="gender" sort="auto" />
</listhead>
<listgroup id="gp1" open="false">
 <listcell label="Group1"/>
 <listcell label="Group2"/>
</listgroup>
<listitem>
 <listcell label="a Mary" />
 <listcell label="a FEMALE" />
</listitem>
<listitem>
 <listcell label="b Mary" />
 <listcell label="b FEMALE" />
</listitem>
<listitem id="li1">
 <listcell label="c Maryl" />
 <listcell label="c FEMALE1" />
</listitem>
<listitem>
 <listcell label="d Mary" />
 <listcell label="d FEMALE" />
</listitem>
<listitem>
 <listcell label="e John" />
 <listcell label="e MALE" />
</listitem>
<listgroupfoot id="f1">
 <listcell label="10 emails" />
 <listcell label="zk1" />
</listgroupfoot>
<listgroup id="g2" label="Grouping 2" />
<listitem>
 <listcell label="Jane" />
 <listcell label="FEMALE" />
</listitem>
<listitem>
 <listcell label="Henry" />
 <listcell label="MALE" />
</listitem>

</listbox>
</zk>
```

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: Listitem

## Supported Children

Listcell [2]

## Use Cases

Version	Description	Example Location
---------	-------------	------------------

## Version History

Version	Date	Content
---------	------	---------

## References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Listgroupfoot.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkex/sel>Listgroupfoot.html#>

# Listhead

## Listhead

- Demonstration: Listbox (Sorting) <sup>[1]</sup>
- Java API: Listhead <sup>[2]</sup>
- JavaScript API: Listhead <sup>[3]</sup>
- Style Guide: Listhead

## Employment/Purpose

A list headers used to define multi-columns and/or headers. Can only support Listheader as its child.

## Example

listbox demo	
name	gender
Mary	FEMALE
John	MALE
Jane	FEMALE
Henry	MALE
This is footer1	This is footer2

```
<window title="listbox demo" border="normal" width="250px">
 <listbox id="box">
 <listhead sizable="true">
 <listheader label="name" sort="auto"/>
 <listheader label="gender" sort="auto"/>
 </listhead>
 <listitem>
 <listcell label="Mary"/>
 <listcell label="FEMALE"/>
 </listitem>
 <listitem>
 <listcell label="John"/>
 <listcell label="MALE"/>
 </listitem>
 <listitem>
 <listcell label="Jane"/>
 <listcell label="FEMALE"/>
 </listitem>
 <listitem>
 <listcell label="Henry"/>
 <listcell label="MALE"/>
 </listitem>
 </listbox>
</window>
```

```

<listfoot>
 <listfooter><label value="This is footer1"/></listfooter>
 <listfooter><label value="This is footer2"/></listfooter>
</listfoot>
</listbox>
</window>

```

## Properties

### Menupopup

By default, the *none* is assumed, you can specify the *auto* to appear a default menu for the listhead. Or you can provide your own menupopup for the component.

[Since 6.5.0]

### Auto

Author	Title
Philip Hensher	Group
Philip Hensher	Sort Ascending
Philip Hensher	Sort Descending
Michael Greenberg	<input checked="" type="checkbox"/> Author
Michael Greenberg	<input checked="" type="checkbox"/> Title
Rick Perlstein	<input checked="" type="checkbox"/> Publisher
Rick Perlstein	<input checked="" type="checkbox"/> Hardcover

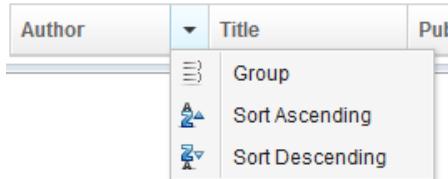
```

<zk>
 <listbox>
 <listhead menupopup="auto">
 <listheader label="Author" sort="auto"/>
 <listheader label="Title" sort="auto"/>
 <listheader label="Publisher" sort="auto"/>
 <listheader label="Hardcover" sort="auto"/>
 </listhead>
 // omitted...
 </listbox>
</zk>

```

listItems will be categorized by the label of its children (listcells). You could extend (ListitemComparator [4]) and assign to the listheader (sorting properties) to change this default behavior.

## Customized Menupopup



```
<menupopup id="editPopup">
 <menuitem label="Group" image="~/zul/img/grid/menu-group.png"/>
 <menuitem label="Sort Ascending" image="~/zul/img/grid/menu-arrowup.png"/>
 <menuitem label="Sort Descending" image="~/zul/img/grid/menu-arrowdown.png"/>
</menupopup>
<listbox>
 <listhead menupopup="editPopup">
 <listheader label="Author" sort="auto"/>
 <listheader label="Title" sort="auto"/>
 <listheader label="Publisher" sort="auto"/>
 <listheader label="Hardcover" sort="auto"/>
 </listhead>
 // omitted...
</listbox>
```

As you can see, the example above specifies a customized menu popup to the columns as its column menu.

## Invisible Listhead for Alignment

Sometimes you want to use the listheader with a size or hflex value, but you don't want to show it up on the page, you can specify all the listheaders in the same listhead with an empty string.

For example,

```
<listbox width="200px">
 <listhead>
 <listheader hflex="1" />
 <listheader hflex="2" />
 <listheader hflex="1" />
 </listhead>
 <auxhead>
 <auxheader colspan="3">
 auxheader (listheaders hidden)
 </auxheader>
 </auxhead>
 <listitem>
 <listcell>hflex 1</listcell>
 <listcell>hflex 2</listcell>
 <listcell>hflex 1</listcell>
 </listitem>
 <listitem>
 <listcell>hflex 1</listcell>
 <listcell>hflex 2</listcell>
```

```

<listcell>hflex 1</listcell>
</listitem>
</listbox>

```

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: HeadersElement

## Supported Children

\* Listheader

## Use Cases

Version	Description	Example Location
5.0	Multiline Listheader	[5]

## Version History

Version	Date	Content
6.5.0	June 2012	ZK-120 [6]: Provide menupopup="auto" for listbox

## References

- [1] <http://www.zkoss.org/zkdemo/listbox/sorting>
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Listhead.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/sel>Listhead.html#>
- [4] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/ListitemComparator.html#>
- [5] <http://www.zkoss.org/forum/listComment/6864>
- [6] <http://tracker.zkoss.org/browse/ZK-120>

# Listheader

## Listheader

- Demonstration: Listbox (Sorting) <sup>[1]</sup>
- Java API: Listheader <sup>[1]</sup>
- JavaScript API: Listheader <sup>[2]</sup>
- Style Guide: Listheader

## Employment/Purpose

The list header which defines the attributes and header of a column of a list box. Its parent must be Listhead.

## Example

listbox demo	
name	gender
Mary	FEMALE
John	MALE
Jane	FEMALE
Henry	MALE
This is footer1	This is footer2

```
<window title="listbox demo" border="normal" width="250px">
 <listbox id="box">
 <listhead sizable="true">
 <listheader label="name" sort="auto"/>
 <listheader label="gender" sort="auto"/>
 </listhead>
 <listitem>
 <listcell label="Mary"/>
 <listcell label="FEMALE"/>
 </listitem>
 <listitem>
 <listcell label="John"/>
 <listcell label="MALE"/>
 </listitem>
 <listitem>
 <listcell label="Jane"/>
 <listcell label="FEMALE"/>
 </listitem>
 <listitem>
 <listcell label="Henry"/>
 <listcell label="MALE"/>
 </listitem>
 </listbox>
</window>
```

```

<listfoot >
 <listfooter><label value="This is footer1"/></listfooter>
 <listfooter><label value="This is footer2"/></listfooter>
</listfoot>
</listbox>
</window>

```

## Supported Events

Name	Event Type
onSort	<b>Event:</b> SortEvent [4] Denotes user has sorted the row of this column.
onGroup	<b>Event:</b> SortEvent [4]  [ZK PE] [Since 6.5.0]  Denotes user has grouped all the cells under a column.
onUngroup	<b>Event:</b> SortEvent [4]  [ZK EE] [Since 6.5.0]  Denotes user has ungrouped all the cells under a column.

- Inherited Supported Events: HeaderElement

## Supported Children

\*ALL

## Use Cases

Version	Description	Example Location
5.0	Multiline Listheader	[5]

## Version History

Version	Date	Content
6.5.0	June 2012	ZK-120 [6]: Provide menupopup="auto" for listbox
6.5.0	June 2012	ZK-147 [5]: Support ungroup for grid's column menu

## References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Listheader.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/sel>Listheader.html#>

# Listitem

---

## Listitem

- Demonstration: Listbox (Keystroke Command) [\[1\]](#)
- Java API: Listitem [\[1\]](#)
- JavaScript API: Listitem [\[2\]](#)
- Style Guide: Listitem

## Employment/Purpose

A list item.

## Example

name	gender
Mary	FEMALE
John	MALE
Jane	FEMALE
Henry	MALE
This is footer1	This is footer2

```
<window title="listbox demo" border="normal" width="250px">
 <listbox id="box">
 <listhead sizable="true">
 <listheader label="name" sort="auto" />
 <listheader label="gender" sort="auto" />
 </listhead>
 <listitem>
 <listcell label="Mary" />
 <listcell label="FEMALE" />
 </listitem>
 <listitem>
 <listcell label="John" />
 </listitem>
 </listbox>
</window>
```

```

 <listcell label="MALE" />
 </listitem>
 <listitem>
 <listcell label="Jane" />
 <listcell label="FEMALE" />
 </listitem>
 <listitem>
 <listcell label="Henry" />
 <listcell label="MALE" />
 </listitem>
 <listfoot>
 <listfooter>
 <label value="This is footer1" />
 </listfooter>
 <listfooter>
 <label value="This is footer2" />
 </listfooter>
 </listfoot>
</listbox>
</window>

```

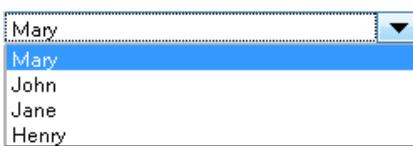
## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

## Supported Molds

Available molds of a component are defined in lang.xml embedded in zul.jar. The mold of listitem is decided by the mold of listbox.

Name	Snapshot												
default	<table border="1"> <thead> <tr> <th>name</th> <th>gender</th> </tr> </thead> <tbody> <tr> <td>Mary</td> <td>FEMALE</td> </tr> <tr> <td>John</td> <td>MALE</td> </tr> <tr> <td>Jane</td> <td>FEMALE</td> </tr> <tr> <td>Henry</td> <td>MALE</td> </tr> <tr> <td>This is footer1</td> <td>This is footer2</td> </tr> </tbody> </table>	name	gender	Mary	FEMALE	John	MALE	Jane	FEMALE	Henry	MALE	This is footer1	This is footer2
name	gender												
Mary	FEMALE												
John	MALE												
Jane	FEMALE												
Henry	MALE												
This is footer1	This is footer2												
select													

## Supported Children

\* Listcell

## Use Cases

See Listbox.

## Version History

Version	Date	Content
---------	------	---------

## References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul>Listitem.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/sel>Listitem.html#>

## Tree

---

### Tree

- Demonstration: Tree <sup>[1]</sup>
- Java API: Tree <sup>[2]</sup>
- JavaScript API: Tree <sup>[3]</sup>
- Style Guide: Tree

## Employment/Purpose

A tree consists of three parts, the set of columns, the set of footers, and the tree body. The set of columns is defined by a number of treecol components, one for each column. Each column will appear as a header at the top of the tree. The second part, The set of footers is defined by a number of treefooter components, one for each column also. Each column will appear as a footer at the bottom of the tree. The third part, the tree body, contains the data to appear in the tree and is created with a treechildren component.

Although treecols is optional, if it exists, notice that the number of its child (treecol) should equal the number of treecell, so that tree can display its content correctly. If treecols contains no treecol, the tree will display nothing in its content.

## Example

### 2-Column Tree

tree demo	
Name	Description
Item 1	Item 1 description
▲ Item 2	Item 2 description
Item 2.1	
Item 2.2	Item 2.2 is something who cares
Item 3	
Count	Summary

```
<window title="tree demo" border="normal" width="400px" >
 <tree id="tree" rows="5">
 <treetable sizable="true">
 <treetablecol label="Name" />
 <treetablecol label="Description" />
 </treetable>
 <treetablechildren>
 <treetableitem>
 <treetablerow>
 <treetablecell label="Item 1" />
 <treetablecell label="Item 1 description" />
 </treetablerow>
 </treetableitem>
 <treetableitem>
 <treetablerow>
 <treetablecell label="Item 2" />
 <treetablecell label="Item 2 description" />
 </treetablerow>
 <treetablechildren>
 <treetableitem>
 <treetablerow>
 <treetablecell label="Item 2.1" />
 </treetablerow>
 </treetableitem>
 <treetableitem>
 <treetablerow>
 <treetablecell label="Item 2.2" />
 <treetablecell label="Item 2.2 is something who cares" />
 </treetablerow>
 </treetableitem>
 </treetablechildren>
 </treetableitem>
 </treetablechildren>
 </tree>
</window>
```

```

 </treechildren>
 </treeitem>
 <treeitem label="Item 3" />
</treechildren>
<treefoot>
 <treefooter label="Count" />
 <treefooter label="Summary" />
</treefoot>
</tree>
</window>

```

## Default Selection

```

David
Thomas
Steven

```

```

<zscript><![CDATA[
DefaultTreeModel stm = new DefaultTreeModel(new
DefaultTreeNode("ROOT",
Arrays.asList(new DefaultTreeNode[]{
 new DefaultTreeNode("David",new ArrayList()),
 new DefaultTreeNode("Thomas",new ArrayList()),
 new DefaultTreeNode("Steven",new ArrayList())))));
]]></zscript>

<tree width="300px" model="${stm}" onAfterRender="self.setSelectedItem(self.getTreechildren().getLastChild())"/>

```

## Change Style

To change the style of tree icon, you may call `setZclass(String style)`. Four built in style includes "z-tree", "z-dottree", "z-filetree", "z-vfiletree"

```
Since ZK 7.0.0
```

Since ZK 7.0.0, the dottree, filetree and vfiletree styles are deprecated because designs are changed.

## Mouseless Entry Tree

- Press UP and DOWN to move the selection up and down by one tree item.
- Press PgUp and PgDn to move the selection up and down by one page.
- Press HOME to move the selection to the first item, and END to the last item.
- Press RIGHT to open a tree item, and LEFT to close a tree item.
- Press Ctrl+UP and Ctrl+DOWN to move the focus up and down by one tree item without changing the selection.
- Press SPACE to select the item in focus.

# Paging

## Autopaging

When using the paging mold and vflex, you could also turn on autopaging (boolean) Tree.setAutopaging( boolean) [4]) such that the page size will be adjusted automatically based on the available space.

[Since 5.0.2]

**Note:** If enable the autopaging, the height of each row will be applied the following CSS by default. If you want to change the height, please overwrite the CSS rule as your preference.

```
.z-tree-autopaging .z-treecell-cnt,
.z-dottree-autopaging .z-treecell-cnt,
.z-filetree-autopaging .z-treecell-cnt,
.z-vfiletree-autopaging .z-treecell-cnt {
 height: 30px;
 overflow: hidden;
}
```

[Since 5.0.8]

**Note:** In ZK 7, we change the naming *.z-treecell-cnt* to *.z-treecell-content*.

[Since 7.0.3]

## PagingDisabled

[Since 8.0.3]

Once the `pagingDisabled` is set to `true`, users will be blocked from navigating through the pagination.

## The onPaging and onPageSize Event

When a user clicks to scroll up and down the page, the `onPaging` event is sent along with a `org.zkoss.zul.event.PagingEvent` instance. Similarly, the `onPageSize` event is sent along with an `org.zkoss.zul.event.PageSize` instance.

## Frozen Component

In ZK 7 you are now able to freeze columns within a Tree. This mirrors functionality seen within Excel and makes data in these components easier to read, interpret and handle.

The following code demonstrates how to freeze a column within a Tree:

```
<tree id="tree" rows="5" width="600px">
 <frozen columns="2" start="1" />
 <treetable>
 <treetablecols sizable="true">
 <treetablecol width="100px">ID</treetablecol>
 <treetablecol width="50px">Priority</treetablecol>
 <treetablecol width="50px">Status</treetablecol>
 <treetablecol width="150px">Summary</treetablecol>
 <treetablecol width="250px">Detail</treetablecol>
```

```
</treecols>
<treechildren>
 <treeitem>
 <treerow>
 <treecell>0001</treecell>
 <treecell>1</treecell>
 <treecell>closed</treecell>
 <treecell>Fix login issue</treecell>
 <treecell>Login does not work at all</treecell>
 </treerow>
 </treeitem>
 <treeitem>
 <treerow>
 <treecell>0002</treecell>
 <treecell>3</treecell>
 <treecell>open</treecell>
 <treecell>Button style broken</treecell>
 <treecell>Check main.css</treecell>
 </treerow>
 <treechildren>
 <treeitem>
 <treerow>
 <treecell>00021</treecell>
 <treecell>1</treecell>
 <treecell>closed</treecell>
 <treecell>Fix logout issue</treecell>
 <treecell>Logout does not work at all</treecell>
 </treerow>
 </treeitem>
 </treechildren>
</treeitem>
<treeitem>
 <treerow>
 <treecell>0003</treecell>
 <treecell>2</treecell>
 <treecell>open</treecell>
 <treecell>Client search result</treecell>
 <treecell>Search service returns incomplete result</treecell>
 </treerow>
</treeitem>
</treechildren>
</tree>
```

[Since 7.0.0]

- For further details, please refer to Frozen component directly.

## Selection

### Nonselectable Tags

[ 5.0.5 ]

By default, when an user clicks on a BUTTON, INPUT, TEXTAREA or A tag, the selection state of the item won't be changed. For example, when an user clicks the textbox in the following example, the selection state of the item won't be changed (only the textbox gains the focus).

```
<treeitem>
 <treerow>
 <treetcell><textbox/></treetcell>
```

Sometimes it is not intuitive, such as using with inplace editing (`InputElement.isInplace()`<sup>[5]</sup>). If you want to have more control of whether to select an item, you could specify a list of tags in the nonselectableTags property (`Tree.setNonselectableTags(java.lang.String)`<sup>[5]</sup>). For example, if you want to select the item, no matter what tag the user clicks, you could specify an empty string as follows.

```
<tree nonselectableTags="">
 <treetchildren>
 <treeitem>
 <treerow>
 <treetcell><textbox/></treetcell>
```

If you want to toggle the selection only when the user clicks on the checkmark, you could specify \* ([since 5.0.6]). Notice that you have to specify `checkmark="true"` as well (otherwise, no item is selectable).

```
<tree checkmark="true" nonselectableTags="*">
```

### Multiple Selection

When the user clicks on a tree item, the whole item is selected and the `onSelect` event is sent back to the server to notify the application. You can control whether a tree control allows multiple selections by setting the `multiple` attribute to true. The default value is false.

**Note:** If you use a model, the `multiple` attribute should be set to the model instead of the tree itself.

### The Checkmark Property

Name	Description
<input checked="" type="checkbox"/> Item 1	Item 1 description
<input checked="" type="checkbox"/> ▲ Item 2	Item 2 description
<input type="checkbox"/> ▲ Item 2.1	
<input type="checkbox"/> Item 2.1.1	
<input type="checkbox"/> Item 2.1.2	

The `checkmark` attribute controls whether to display a checkbox or a radio button in front of each tree item. If the `multiple` attribute is set to true and the `checkmark` is set to true then a checkbox will be displayed in front of every item. However, if `multiple` is set to false then a radio button will be displayed instead.

## Deselect Others when Clicking an Item with Checkmark

If a tree's checkmark (Tree.isCheckmark() [6]) is enabled, the selection will be toggled when an user clicks an item. In other words, all other items will remain the same.

If you prefer to deselect all other items and select the item being clicked (which the behavior of ZK 5.0.4 and earlier), you could specify true to this library property called org.zkoss.zul.tree.checkmarkDeselectOthers in WEB-INF/zk.xml:

```
<library-property>
 <name>org.zkoss.zul.tree.checkmarkDeselectOthers</name>
 <value>true</value>
</library-property>
```

## Toggle Selection when Right Clicking an Item with Checkmark

If a tree's checkmark (Tree.isCheckmark() [6]) is enabled, the selection will be toggled when user right click on item.

If you prefer not to select/deselect item on right click, you could specify false to this library property called org.zkoss.zul.tree.rightSelect in WEB-INF/zk.xml:

```
<library-property>
 <name>org.zkoss.zul.tree.rightSelect</name>
 <value>false</value>
</library-property>
```

[Since 5.0.5]

# Properties

## Auxiliary Headers

Like grids, you can specify auxiliary headers with the auxhead and auxheader components.

Please refer to the Grid for more details.

## SizedByContent

By default, the widths of columns have to be specified explicitly, or it will be split equally among columns regardless what content they might have. If you want to have the minimal width (that fit the content), you could specify hflex="min" at the column (not the tree).

However, the tree has a special mode called sized-by-content (Tree.setSizedByContent(boolean) [7]). By specifying it to true, the column width will be adjusted automatically. However, it is controlled by the browser, so have have no 100% control of it. For example, if an user resized a column, the final width might not be exactly the same as what he resized.

In general, we suggest to specify hflex in column, rather than specifying sizedByContent at tree for much more predictable result.

## Rows

The `rows` attribute is used to control how many rows are visible. By setting it to zero, the tree control will resize itself to hold as many as items as possible.

## Vflex

The `vflex` attribute controls whether to grow or shrink vertically to fit their given space. It is so-called vertical flexibility. For example, if the tree is too big to fit in the browser window, it will shrink to make the whole tree visible in the browser window.

This property is ignored if the `rows` attribute is specified.

## Maxlength

The `maxlength` attribute defines the maximal allowed characters being visible at the browser. By setting this property, you can make a narrower tree control.

## Sizable

Like `columns`, you can set the `sizable` attribute of `treecols` to `true` in order to allow users to resize the width of the tree headers. Similarly, the `onColSize` event is sent when a user resizes the widths.

## Auto Fitting Columns

[Since 5.0.0]

When you want to resize a column of a Tree or Listbox, all you now need to do is double click the column when the mouse is over where the columns meet and the column will automatically resize to fit its contents. To enable this functionality Tree's `treecols` need the attribute `sizable="true"`. In other words, all sizable column provides the auto-fitting functionality.

## Scollable Tree

A tree will be scrollable if you specify the `rows` attribute or the `height` attribute and there is not enough space to display all the tree items.

Name	Description	
Item 1	Item 1 description	▲
▼ Item 2	Item 2 description	☰
▼ Item 2.1		▼
Item 2.1.1		

```
<tree rows="4">
 <treecols>
 <treecol label="Name"/>
 <treecol label="Description"/>
 </treecols>
 <treechildren>
 <treeitem>
 <treerow>
 <treecell label="Item 1"/>
```

```
 <treecell label="Item 1 description"/>
 </treerow>
</treeitem>
<treeitem>
 <treerow>
 <treecell label="Item 2"/>
 <treecell label="Item 2 description"/>
 </treerow>
 <treechildren>
 <treeitem>
 <treerow>
 <treecell label="Item 2.1"/>
 </treerow>
 <treechildren>
 <treeitem>
 <treerow>
 <treecell label="Item 2.1.1"/>
 </treerow>
 </treeitem>
 <treeitem>
 <treerow>
 <treecell label="Item 2.1.2"/>
 </treerow>
 </treeitem>
 </treechildren>
 </treeitem>
 <treeitem>
 <treerow>
 <treecell label="Item 2.2"/>
 <treecell label="Item 2.2 is something who cares"/>
 </treerow>
 </treeitem>
 </treechildren>
</treeitem>
<treeitem label="Item 3"/>
</treechildren>
</tree>
```

[Since 7.0.0]

The browser's default scrollbar is replaced by floating scrollbar and it is not visible unless user mouse over on the content. To turn off the floating scrollbar and use original scrollbar, please add the following configuration in zk.xml.

```
<library-property>
 <name>org.zkoss.zul.nativebar</name>
 <value>true</value>
</library-property>
```

**Note:** the value of org.zkoss.zul.nativebar is true by default (since 7.0.2)

## The Open Property and the onOpen Event

Each tree item contains the `open` property which is used to control whether to display its child items. The default value is true. By setting this property to false, you are able to control what part of the tree is invisible.

```
<treeitem open="false">
```

When a user clicks on the +/- button, he opens the tree item and makes its children visible. The `onOpen` event is then sent to the server to notify the application.

For sophisticated applications, you can defer the creation of the content of the tree item or manipulate its content dynamically until the `onOpen` event is received. Refer to the **Load on Demand** section in the **ZK User Interface Markup Language** chapter for details.

## Sticky Header

After adding a sclass "z-sticky-header", when we scroll down a page and make a Tree's header out of visible range in a viewport, the Tree's header becomes floating and sticky on the top of the page.

```
<tree sclass="z-sticky-header">
 <!-- treecols, treeitem... -->
</tree>
```

## Create-on-Open for Tree Controls

As illustrated below, you could listen to the `onOpen` event, and then load the children of a tree item. You can do the same thing using group boxes.

```
<zk>
 <tree width="200px">
 <treecols>
 <treecol label="Subject"/>
 <treecol label="From"/>
 </treecols>
 <treechildren>
 <treeitem open="false" onOpen="load()">
 <treerow>
 <treecell label="Intel Snare XML"/>
 <treecell label="David Needle"/>
 </treerow>
 <treechildren/>
 </treeitem>
 </treechildren>
 <zscript>
 void load() {
 Treechildren tc = self.getTreechildren();
 if (tc.getChildren().isEmpty()) {
 Treeitem ti = new Treeitem();
```

```
 ti.setLabel("New added");
 ti.setParent(tc);
 }
}

</zscript>
</tree>
</zk>
```

## Custom Attributes

### org.zkoss.zul.tree.rightSelect

[default: true]  
[inherit: true]<sup>[8]</sup>

It specifies the selection should be toggled when user right clicks on an item, if the checkmark is enabled (`Tree.isCheckmark()`<sup>[6]</sup>). If it is turned off, right clicking on an item will change its selection state.

- 
- [1] <http://www.zkoss.org/zkdemo/tree>
  - [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Tree.html#>
  - [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/sel/Tree.html#>
  - [4] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Tree.html#setAutopaging>(  
[5] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Tree.html#setNonselectableTags\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Tree.html#setNonselectableTags(java.lang.String))  
[6] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Tree.html#isCheckmark>(  
[7] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Tree.html#setSizedByContent\(boolean\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Tree.html#setSizedByContent(boolean))  
[8] The custom attribute could be specified in this component, or any of its ancestor. In addition, it could be specified as a library property to enable or disable it for the whole application.

### org.zkoss.zul.tree.autoSort

[default: false]  
[inherit: true]<sup>[1]</sup>  
[since 5.0.7]

It specifies whether to sort the model when the following cases:

- (`TreeModel`) `Tree.setModel(TreeModel)` (<http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Tree.html#setModel>) is called and `Treecol.setSortDirection(String)` ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Treecol.html#setSortDirection\(String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Treecol.html#setSortDirection(String))) is set.
- `Treecol.setSortDirection(String)` ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Treecol.html#setSortDirection\(String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Treecol.html#setSortDirection(String))) is called.
- Model receives `TreeDataEvent` (<http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/event/TreeDataEvent.html#>) and `Treecol.setSortDirection(String)` ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Treecol.html#setSortDirection\(String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Treecol.html#setSortDirection(String))) is set.

If you want to ignore sort when receiving `TreeDataEvent` (<http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/event/TreeDataEvent.html#>), you can specifies the value as **ignore.change**.

## org.zkoss.zul.tree.autohidePaging

[default: true]  
 [inherit: true]<sup>[2]</sup>  
 [since: 7.0.1]

It specifies whether to enable autohide property for internal paging component.

- [1] The custom attribute could be specified in this component, or any of its ancestor. In addition, it could be specified as a library property to enable or disable it for the whole application.
- [2] Same as above.

## Supported Events

Name	Event Type
onSelect	<b>Event:</b> SelectEvent ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event&gt;SelectEvent.html#">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event&gt;SelectEvent.html#</a> ) Notifies one that the user has selected a new item in the tree.
onFocus	<b>Event:</b> Event ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/Event.html#">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/Event.html#</a> ) Denotes when a component gets the focus. Remember event listeners execute at the server, so the focus at the client might be changed when the event listener for onFocus got executed.
onBlur	<b>Event:</b> Event ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/Event.html#">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/Event.html#</a> ) Denotes when a component loses the focus. Remember event listeners execute at the server, so the focus at the client might be changed when the event listener for onBlur got executed.
onAfterRender	<b>Event:</b> Event ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/Event.html#">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/Event.html#</a> ) Notifies one that the model's data has been rendered.
onPageSize	<b>Event:</b> PageSizeEvent ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/event/PageSizeEvent.html#">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/event/PageSizeEvent.html#</a> ) Notifies the paging size has been changed when the autopaging ( Tree.setAutopaging(boolean) ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Tree.html#setAutopaging(boolean)">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Tree.html#setAutopaging(boolean)</a> )) is enabled and user changed the size of the content.

- Inherited Supported Events: XulElement

## Supported Molds

Available molds of a component are defined in lang.xml embedded in zul.jar.

Name	Snapshot														
default	<table border="1"> <thead> <tr> <th>Name</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Item 1</td><td>Item 1 description</td></tr> <tr> <td>▲ Item 2</td><td>Item 2 description</td></tr> <tr> <td>    Item 2.1</td><td></td></tr> <tr> <td>    Item 2.2</td><td>Item 2.2 is something who cares</td></tr> <tr> <td>    Item 3</td><td></td></tr> <tr> <td>Count</td><td>Summary</td></tr> </tbody> </table>	Name	Description	Item 1	Item 1 description	▲ Item 2	Item 2 description	Item 2.1		Item 2.2	Item 2.2 is something who cares	Item 3		Count	Summary
Name	Description														
Item 1	Item 1 description														
▲ Item 2	Item 2 description														
Item 2.1															
Item 2.2	Item 2.2 is something who cares														
Item 3															
Count	Summary														

paging	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%;">Name</th><th style="width: 50%;">Description</th></tr> </thead> <tbody> <tr> <td>Item 1</td><td>Item 1 description</td></tr> <tr> <td>▲ Item 2</td><td>Item 2 description</td></tr> <tr> <td>    Item 2.1</td><td></td></tr> </tbody> </table> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Count</td><td style="width: 70%;">Summary</td></tr> <tr> <td style="text-align: center;">    /    </td><td style="text-align: right; vertical-align: bottom;"> [ 1 - 3 / 5 ] </td></tr> </table>	Name	Description	Item 1	Item 1 description	▲ Item 2	Item 2 description	Item 2.1		Count	Summary	/	[ 1 - 3 / 5 ]
Name	Description												
Item 1	Item 1 description												
▲ Item 2	Item 2 description												
Item 2.1													
Count	Summary												
/	[ 1 - 3 / 5 ]												

## Supported Children

\* Treecols, Treechildren, Treefoot

## Use Cases

Version	Description	Example Location
3.6	Smalltalk: Building a Complex Tree with Grid-in-Grid	Building a Complex Tree with Grid-in-Grid
3.6	Expand all items of a Tree at start	( <a href="http://www.zkoss.org/forum/listComment/9379">http://www.zkoss.org/forum/listComment/9379</a> ) ( <a href="http://www.zkoss.org/forum/">http://www.zkoss.org/forum/</a> listComment/9379)

## Browser Limitations

Browser	description
Chrome & Safari	<pre style="font-family: monospace; font-size: 0.9em; margin: 0; padding: 0; border: none;"> &lt;zk&gt;     &lt;hbox&gt;         &lt;tree&gt;             &lt;treecols&gt;                 &lt;treecol label="Name" /&gt;                 &lt;treecol label="Description" /&gt;             &lt;/treecols&gt;             &lt;treechildren&gt;                 &lt;treeitem&gt;                     &lt;treerow&gt;                         &lt;treetable&gt;                             &lt;treetablecell label="Item 1" /&gt;                             &lt;treetablecell label="Item 1 description" /&gt;                         &lt;/treetable&gt;                     &lt;/treerow&gt;                 &lt;/treeitem&gt;             &lt;/treechildren&gt;         &lt;/tree&gt;     &lt;/hbox&gt; &lt;/zk&gt;</pre> <p>The width of the tree will be zero with Chrome &amp; Safari. the Webkit considers the width of tree as zero. please specify the width to tree to work around.</p>

## Version History

Version	Date	Content
5.0.2	May 2010	Support the autopaging
5.0.4	July 2010	Support onAfterRender event
5.0.5	September 2010	The nonselectabletag property was introduced to enhance the control of when to select an item
5.0.5	September 2010	When a tree's checkmar is enabled and an item is clicked, it will toggle the selection of the item and the other remains the same.
5.0.5	October 2010	When a tree's checkmark is enabled and an item is right clicked, it will toggle the selection of the item.
5.0.6	February 2011	Sorting was supported
5.0.6	February 2011	The nonselectableTags property supported "*".
5.0.7	April 2011	Tree shall sort model based on current state.
5.0.7	April 2011	The onPageSize event was introduced.
7.0.0	December 2013	Change Style and Scrollable Tree were updated.
7.0.0	December 2013	Frozen Component was introduced.
7.0.1	January 2014	ZK-2079 ( <a href="http://tracker.zkoss.org/browse/ZK-2079">http://tracker.zkoss.org/browse/ZK-2079</a> ): Add a custom attributes "org.zkoss.zul.tree.autohidePaging" for control autohide in internal paging component
7.0.2	April 2014	Due to the better user-firendly for the scrollbar layout, we changed the org.zkoss.zul.nativebar of the library property to true by default for Grid, Listbox, Tree and Borderlayout component.
7.0.3	July 2014	ZK-2359 ( <a href="http://tracker.zkoss.org/browse/ZK-2359">http://tracker.zkoss.org/browse/ZK-2359</a> ): Since ZK 7, the style class naming of autopaging has changed.
9.6.0	Mar 2021	ZK-4795 ( <a href="http://tracker.zkoss.org/browse/ZK-4795">http://tracker.zkoss.org/browse/ZK-4795</a> ): Grid/Listbox/Tree supports sticky column headers

# Treecell

## Treecell

- Demonstration: Tree (Dynamic Styling) <sup>[1]</sup>
- Java API: Treecell <sup>[2]</sup>
- JavaScript API: Treecell <sup>[3]</sup>
- Style Guide: Treerow

## Employment/Purpose

Treecell represents one column in a treerow by sequential. Treecell can contain any components in it, such as label, image, textbox etc.

## Example

The screenshot shows a window titled "tree demo". Inside, there is a tree view with the following structure:

Name	Description
Item 1	Item 1 description
Item 2	Item 2 description
Item 2.1	
Item 3	

Items 1 and 2 have folder icons next to them. Item 2 has a plus sign indicating it has children, and Item 2.1 has a right-pointing arrow indicating it is a child of Item 2.

```
<window title="tree demo" border="normal" width="400px">
 <tree id="tree" width="90%">
 <treecols sizable="true">
 <treecol label="Name" />
 <treecol label="Description" />
 </treecols>
 <treechildren>
 <treeitem>
 <treerow>
 <treecell>
 <image src="/img/folder.gif" />
 Item 1
 </treecell>
 <treecell>
 <textbox value="Item 1 description" />
 </treecell>
 </treerow>
 </treeitem>
 <treeitem>
 <treerow>
```

```
<treecell label="Item 2" />
<treecell label="Item 2 description" />
</treerow>
<treechildren>
<treeitem open="false">
<treerow>
<treecell label="Item 2.1">
<image src="/img/folder.gif" />
</treecell>
</treerow>
<treechildren>
<treeitem>
<treerow>
<treecell label="Item 2.1.1" />
</treerow>
</treeitem>
</treechildren>
</treeitem>
</treechildren>
</treeitem>
<treeitem label="Item 3" />
</treechildren>
</tree>
</window>
```

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: LabelImageElement

## Supported Children

\*ALL

## Use Cases

Version	Description	Example Location
---------	-------------	------------------

## Version History

Version	Date	Content
---------	------	---------

## References

- [1] [http://www.zkoss.org/zkdemo/tree/dynamic\\_styling](http://www.zkoss.org/zkdemo/tree/dynamic_styling)
- [2] [#](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Treecell.html)
- [3] [#](http://www.zkoss.org/javadoc/latest/jsdoc/zul/sel/Treecell.html)

# Treechildren

---

## Treechildren

- Demonstration: Tree (Dynamic Styling) <sup>[1]</sup>
- Java API: Treechildren <sup>[1]</sup>
- JavaScript API: Treechildren <sup>[2]</sup>
- Style Guide: Treechildren

## Employment/Purpose

Treechildren contains a collection of treeitem components. It is main body of the Tree and it also the main body of a Treeitem's children.

You can change the page size of each treechildren instance by modifying the pageSize property

## Example

Name	Description
Item 1	Item 1 description
Item 2	Item 2 description
Item 2.1	
Item 3	

```
<window title="tree demo" border="normal" width="400px">
 <tree id="tree" width="90%">
 <treetable sizable="true">
 <treetablecol label="Name" />
```

```
<treecol label="Description" />
</treecols>
<treechildren>
 <treeitem>
 <treerow>
 <treecell>
 <image src="/img/folder.gif" />
 Item 1
 </treecell>
 <treecell>
 <textbox value="Item 1 description" />
 </treecell>
 </treerow>
 </treeitem>
 <treeitem>
 <treerow>
 <treecell label="Item 2" />
 <treecell label="Item 2 description" />
 </treerow>
 <treechildren>
 <treeitem open="false">
 <treerow>
 <treecell label="Item 2.1">
 <image src="/img/folder.gif" />
 </treecell>
 </treerow>
 <treechildren>
 <treeitem>
 <treerow>
 <treecell label="Item 2.1.1" />
 </treerow>
 </treeitem>
 </treechildren>
 </treeitem>
 </treechildren>
 </treeitem>
 <treeitem label="Item 3" />
</treechildren>
</tree>
</window>
```

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

## Supported Children

\* Treeitem

## Use Cases

Version	Description	Example Location

## Version History

Version	Date	Content

## References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Treechildren.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/sel/Treechildren.html#>

# Treecol

## Treecol

- Demonstration: Tree (Dynamic Styling) <sup>[1]</sup>
- Java API: Treecol <sup>[1]</sup>
- JavaScript API: Treecol <sup>[2]</sup>
- Style Guide: Treecol

## Employment/Purpose

A treecol is a top column of tree, Its parent must be Treecols .

## Example

tree demo	
Name	Description
Item 1	Item 1 description
Item 2	Item 2 description
Item 2.1	
Item 3	

```
<window title="tree demo" border="normal" width="400px">
 <tree id="tree" width="90%">
 <treecols sizable="true">
 <treecol label="Name" />
 <treecol label="Description" />
 </treecols>
 <treechildren>
 <treeitem>
 <treerow>
 <treecell>
 <image src="/img/folder.gif" />
 Item 1
 </treecell>
 <treecell>
 <textbox value="Item 1 description" />
 </treecell>
 </treerow>
 </treeitem>
 <treeitem>
 <treerow>
 <treecell label="Item 2" />
```

```

 <treecell label="Item 2 description" />
 </treerow>
 <treechildren>
 <treeitem open="false">
 <treerow>
 <treecell label="Item 2.1">
 <image src="/img/folder.gif" />
 </treecell>
 </treerow>
 <treechildren>
 <treeitem>
 <treerow>
 <treecell label="Item 2.1.1" />
 </treerow>
 </treeitem>
 </treechildren>
 </treeitem>
 </treechildren>
</treeitem>
<treeitem label="Item 3" />
</treechildren>
</tree>
</window>

```

## Supported Events

Name	Event Type
onSort	Event: Event [7] Denotes user has sorted the treeitem of this treecol.

- Inherited Supported Events: HeaderElement

## Supported Children

\*ALL

## Use Cases

Version	Description	Example Location
---------	-------------	------------------

## Version History

Version	Date	Content
5.0.6	Feb 2011	Support onSort event

## References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Treecol.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/sel/Treecol.html#>

# Treecols

## Treecols

- Demonstration: Tree (Dynamic Styling) <sup>[1]</sup>
- Java API: Treecols <sup>[1]</sup>
- JavaScript API: Treecols <sup>[2]</sup>
- Style Guide: Treecols

## Employment/Purpose

A treecols is main part of tree which contains set of columns. The set of columns is defined by a number of treecol components. Each column will appear as a column at the top of the tree.

## Example

Name	Description
Item 1	Item 1 description
Item 2	Item 2 description
Item 2.1	
Item 3	

```
<window title="tree demo" border="normal" width="400px">
 <tree id="tree" width="90%">
 <treecols sizable="true">
 <treecol label="Name" />
 <treecol label="Description" />
 </treecols>
 <treetable>
 <treeitem>
 <treerow>
 <treecell>
 <image src="/img/folder.gif" />
 Item 1
 </treecell>
 <treecell>
 <textbox value="Item 1 description" />
 </treecell>
 </treerow>
 </treeitem>
 </treetable>
 </tree>
</window>
```

```
</treerow>
</treeitem>
<treeitem>
 <treerow>
 <treecell label="Item 2" />
 <treecell label="Item 2 description" />
 </treerow>
 <treetruechildren>
 <treeitem open="false">
 <treerow>
 <treecell label="Item 2.1">
 <image src="/img/folder.gif" />
 </treecell>
 </treerow>
 <treetruechildren>
 <treeitem>
 <treerow>
 <treecell label="Item 2.1.1" />
 </treerow>
 </treeitem>
 </treetruechildren>
 </treeitem>
 </treetruechildren>
</treeitem>
<treeitem label="Item 3" />
</treetruechildren>
</tree>
</window>
```

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: HeadersElement

## Supported Children

\* Treecol

## Use cases

Version	Description	Example Location
---------	-------------	------------------

## Version History

Version	Date	Content
---------	------	---------

## References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Treecols.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/sel/Treecols.html#>

## Treefoot

---

### Treefoot

- Demonstration: N/A
- Java API: Treefoot <sup>[1]</sup>
- JavaScript API: Treefoot <sup>[2]</sup>
- Style Guide: Treefooter

### Employment/Purpose

A treefoot is main part of tree which contains set of footers. The set of footers is defined by a number of treefooter components. Each column will appear as a footer at the bottom of the tree.

### Example

tree demo

Name	Description
Item 1	Item 1 description
Item 2	Item 2 description
Item 2.1	
Item 2.2	Item 2.2 is something who cares
Item 3	
Count	Summary

```
<window title="tree demo" border="normal" width="400px" >
 <tree id="tree" rows="5">
 <treecols sizable="true">
 <treecol label="Name" />
```

```
<treecol label="Description" />
</treecols>
<treetable>
 <treetableheader>
 <treeitem>
 <treerow>
 <treecell label="Item 1" />
 <treecell label="Item 1 description" />
 </treerow>
 </treeitem>
 <treeitem>
 <treerow>
 <treecell label="Item 2" />
 <treecell label="Item 2 description" />
 </treerow>
 <treetablechildren>
 <treeitem>
 <treerow>
 <treecell label="Item 2.1" />
 </treerow>
 </treeitem>
 <treeitem>
 <treerow>
 <treecell label="Item 2.2" />
 </treerow>
 </treeitem>
 </treetablechildren>
 </treeitem>
 </treetableheader>
 <treetablebody>
 <treeitem>
 <treerow>
 <treecell label="Item 2.2 is something who cares" />
 <treecell label="Item 2.2 is something who cares" />
 </treerow>
 </treeitem>
 </treetablebody>
</treetable>
<treetablefooter>
 <treetablechildren>
 <treeitem label="Item 3" />
 </treetablechildren>
 <treetablefoot>
 <treetablefootitem>
 <treetablefootcell label="Count" />
 <treetablefootcell label="Summary" />
 </treetablefootitem>
 </treetablefoot>
</treetablefoot>
</tree>
</window>
```

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

## Supported Children

\* Treefooter

## Use Cases

Tree

## Version History

Version	Date	Content

## References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Treefoot.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/sel/Treefoot.html#>

# Treefooter

## Treefooter

- Demonstration: N/A
- Java API: Treefooter <sup>[1]</sup>
- JavaScript API: Treefooter <sup>[2]</sup>
- Style Guide: Treefooter

## Employment/Purpose

A treefooter is a bottom column of tree, Its parent must be Treefoot. You colud place any child in a tree footer.

## Example

tree demo

Name	Description
Item 1	Item 1 description
▲ Item 2	Item 2 description
Item 2.1	
Item 2.2	Item 2.2 is something who cares
Item 3	
Count	Summary

```
<window title="tree demo" border="normal" width="400px" >
 <tree id="tree" rows="5">
 <treetable>
 <treetablecols sizable="true">
 <treetablecol label="Name" />
 <treetablecol label="Description" />
 </treetablecols>
 <treetablechildren>
 <treetableitem>
 <treetablerow>
 <treetablecell label="Item 1" />
 <treetablecell label="Item 1 description" />
 </treetablerow>
 </treetableitem>
 <treetableitem>
 <treetablerow>
 <treetablecell label="Item 2" />
 <treetablecell label="Item 2 description" />
 </treetablerow>
 </treetableitem>
 </treetablechildren>
 </treetable>
 </tree>
</window>
```

```
<treerow>
 <treetcell label="Item 2.1" />
</treerow>
</treeitem>
<treeitem>
 <treerow>
 <treetcell label="Item 2.2" />
 <treetcell
 label="Item 2.2 is
something who cares" />
 </treerow>
</treeitem>
</treechildren>
</treeitem>
<treeitem label="Item 3" />
</treechildren>
<treetfoot>
 <treetfooter label="Count" />
 <treetfooter label="Summary" />
</treetfoot>
</tree>
</window>
```

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: FooterElement

## Supported Children

\* ALL

## Use Cases

Tree

## Version History

Version	Date	Content

## References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Treefooter.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/sel/Treefooter.html#>

## Treeitem

### Treeitem

- Demonstration: Tree (Dynamic Styling) <sup>[1]</sup>
- Java API: Treeitem <sup>[1]</sup>
- JavaScript API: Treeitem <sup>[2]</sup>
- Style Guide: N/A

### Employment/Purpose

Treeitem contains a row of data (treerow), and an optional treechildren.

If the component doesn't contain a treechildren, it is a leaf node that doesn't accept any child items.

If it contains a treechildren, it is a branch node that might contain other items.

For a branch node, an +/- button will appear at the beginning of the row, such that user could open and close the item by clicking on the +/- button.

### Example

The screenshot shows a window titled "tree demo". Inside, there is a table with two columns: "Name" and "Description". The "Name" column contains icons indicating folder or file status, and the "Description" column contains text descriptions. The tree structure is as follows:

Name	Description
Item 1	Item 1 description
Item 2	Item 2 description
Item 2.1	
Item 3	

```
<window title="tree demo" border="normal" width="400px">
 <tree id="tree" width="90%">
 <treecols sizable="true">
 <treecol label="Name" />
```

```
<treecol label="Description" />
</treecols>
<treechildren>
 <treeitem>
 <treerow>
 <treecell>
 <image src="/img/folder.gif" />
 Item 1
 </treecell>
 <treecell>
 <textbox value="Item 1 description" />
 </treecell>
 </treerow>
 </treeitem>
 <treeitem>
 <treerow>
 <treecell label="Item 2" />
 <treecell label="Item 2 description" />
 </treerow>
 <treechildren>
 <treeitem open="false">
 <treerow>
 <treecell label="Item 2.1">
 <image src="/img/folder.gif" />
 </treecell>
 </treerow>
 <treechildren>
 <treeitem>
 <treerow>
 <treecell label="Item 2.1.1" />
 </treerow>
 </treeitem>
 </treechildren>
 </treeitem>
 </treechildren>
 </treeitem>
 <treeitem label="Item 3" />
</treechildren>
</tree>
</window>
```

More examples please refer to Tree

## Label and Image

Treeitem provides Treeitem.setImage(java.lang.String) [3] and Treeitem.setLabel(java.lang.String) [4] to simplify the assignment of image and label to a treeitem. However, they are actually placed in the first treecell (of the child treerow). Furthermore, if the treecell or treerow are not created, they will be created automatically. For example,

```
<treeitem label="hello"/>
```

is equivalent to

```
<treeitem>
 <treerow>
 <treecell label="hello"/>
 </treerow>
```

It also means you cannot attach a treerow child to the treeitem, after setImage or setLabel was invoked. It means, though a bit subtle, the following will cause an exception:

```
<treeitem label="hello"> <!-- treerow is created automatically because of setLabel -->
 <treerow> <!-- exception since only one treerow is allowed per treeitem -->
</treeitem>
```

## Supported Events

Name	Event Type
onOpen	<b>Event:</b> OpenEvent [3] Denotes user has opened or closed a component. It is useful to implement load-on-demand by listening to the onOpen event, and creating components when the first time the component is opened.

- Inherited Supported Events: XulElement

## Supported Children

\* Treerow, Treechildren

## Use Cases

Version	Description	Example Location
---------	-------------	------------------

## Version History

Version	Date	Content
---------	------	---------

## References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Treeitem.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/sel/Treeitem.html#>
- [3] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Treeitem.html#setImage\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Treeitem.html#setImage(java.lang.String))
- [4] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Treeitem.html#setLabel\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Treeitem.html#setLabel(java.lang.String))

# Treerow

## Treerow

- Demonstration: Tree (Dynamic Styling) <sup>[1]</sup>
- Java API: Treerow <sup>[1]</sup>
- JavaScript API: Treerow <sup>[2]</sup>
- Style Guide: Treerow

## Employment/Purpose

Treerow is a single row in the tree. It is the main content of treeitem. Treerow can contains multiple treecell, each treecell represent one column in this row by sequencial. A treecell can contains any component in it, such as label, image, textbox etc.

## Example

Name	Description
Item 1	Item 1 description
Item 2	Item 2 description
Item 2.1	
Item 3	

```
<window title="tree demo" border="normal" width="400px">
 <tree id="tree" width="90%">
 <treecols sizable="true">
 <treecol label="Name" />
 <treecol label="Description" />
 </treecols>
 <treechildren>
 <treeitem>
 <treerow>
 <treecell>
 <image src="/img/folder.gif" />
 Item 1
 </treecell>
 </treerow>
 </treeitem>
 </treechildren>
 </tree>
</window>
```

```

<treecell>
 <textbox value="Item 1 description" />
</treecell>
</treerow>
</treeitem>
<treeitem>
 <treerow>
 <treecell label="Item 2" />
 <treecell label="Item 2 description" />
 </treerow>
 <treechildren>
 <treeitem open="false">
 <treerow>
 <treecell label="Item 2.1">
 <image src="/img/folder.gif" />
 </treecell>
 </treerow>
 <treechildren>
 <treeitem>
 <treerow>
 <treecell label="Item 2.1.1" />
 </treerow>
 </treeitem>
 </treechildren>
 </treeitem>
 </treechildren>
</treeitem>
<treeitem label="Item 3" />
</treechildren>
</tree>
</window>

```

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

## Supported Children

\* Treecell

## Use Cases

Tree

## Version History

Version	Date	Content
5.0.1	March 2010	Allow the context, tooltip and popup properties to be specified in Treerow.

## References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Treerow.html#>

[2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/sel/Treerow.html#>

## Biglistbox

---

### Biglistbox

- Demonstration: Demo
- Java API: Biglistbox <sup>[1]</sup>
- JavaScript API: Biglistbox <sup>[2]</sup>
- Style Guide: Biglistbox
- Available for ZK:
-   

## Employment/Purpose

A component to handle a huge data sets and provides the same and as many as the functionalities of Listbox including selection, sorting, keystroke navigation, ROD(rendering-on-demand), and so on..

## Example

Big Listbox Demo for 1 Trillion Data					
Header x = 0	a Header x = 1	Header x = 2	Header x = 3	Header x = 4	Header x = 5
y = 0	a y = 0	y = 0	y = 0	y = 0	b y = 0
y = 1	a y = 1	y = 1	y = 1	y = 1	b y = 1
y = 2	a y = 2	y = 2	y = 2	y = 2	b y = 2
y = 3	a y = 3	y = 3	y = 3	y = 3	b y = 3
y = 4	a y = 4	y = 4	y = 4	y = 4	b y = 4
y = 5	a y = 5	y = 5	y = 5	y = 5	b y = 5
y = 6	a y = 6	y = 6	y = 6	y = 6	b y = 6
y = 7	a y = 7	y = 7	y = 7	y = 7	b y = 7
y = 8	a y = 8	y = 8	y = 8	y = 8	b y = 8
y = 9	a y = 9	y = 9	y = 9	y = 9	b y = 9
v = 10	a v = 10	v = 10	v = 10	v = 10	b v = 10

```
<biglistbox hflex="1" vflex="1">
 <!-- Template example
 <template name="heads">
 <html><![CDATA[
 <div class="images_${matrixInfo[0]}28" title="x=${matrixInfo[0]},y=${matrixInfo[1]}>${each[matrixInfo[0]]}</div>
]]></html>
 </template>
 <template name="rows">
 <html><![CDATA[
 <div class="images_${matrixInfo[0]}28" title="x=${matrixInfo[0]},y=${matrixInfo[1]}>${each[matrixInfo[0]]}</div>
]]></html>
 </template> -->
</biglistbox>
```

As you can see, we utilize two attributes - *rowIndex* & *colIndex* from the *matrixInfo* object to receive the current index during template rendering phase.

## Mouseless Entry Listbox

- Press UP and DOWN to move the selection up and down by one item.
- Press LEFT and RIGHT to move the selection left and right by one item.
- Press PgUp and PgDn to move the selection up and down by one page.
- Press HOME to move the selection to the first item, and END to move to the last item.

## Model/Renderer

### MatrixModel

By default, ZK does not provide a built-in model implementation class for MatrixModel<sup>[3]</sup> because Biglistbox is designed to handle unlimited data set, therefore, there is no need to handle model data in memory. This usage is application-dependent and varies from case to case. However, you can extend your own implementation from the AbstractListModel<sup>[4]</sup> skeleton class. For more details, please refer to this smalltalk.

### MatrixRenderer

Here is an implementation example of MatrixRenderer<sup>[5]</sup>

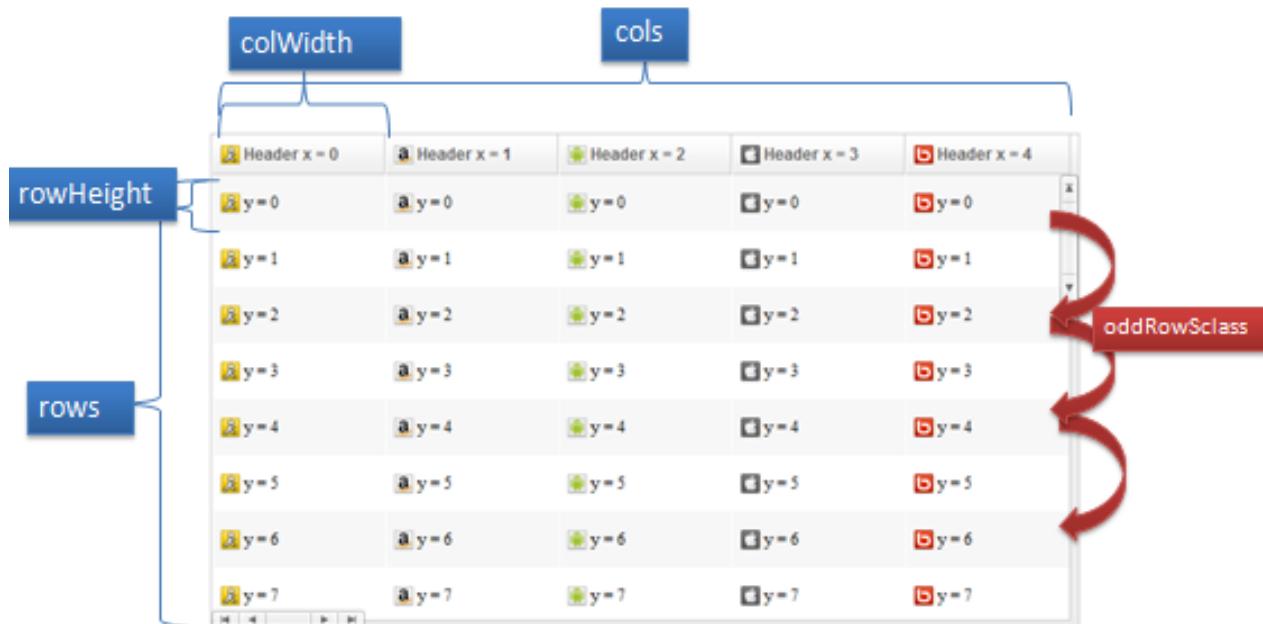
```
new MatrixRenderer<List<String>>() {

 @Override
 public String renderCell(Component owner, List<String> data,
 int rowIndex, int colIndex) throws Exception {
 String d = data.get(colIndex);
 d = d.replace("ZK", "ZK");
 d = d.replace("Hello", "Hello");
 return "<div class='images_" + (colIndex%28) + "' title='x=" +
 colIndex + ",y=" + rowIndex + "'>" + d + "</div>";
 }

 @Override
 public String renderHeader(Component owner, List<String> data,
 int rowIndex, int colIndex) throws Exception {
 return "<div class='images_" + (colIndex % 28) + "' title='"
 + images[colIndex % 28] + "'>" +
 data.get(colIndex)
 + "</div>";
 }
}
```

As you can see, we must implement two methods - *renderCell* and *renderHeader* defined by **MatrixRenderer** to generate the HTML output string for cells and headers. In the **MatrixRenderer** interface, the two methods we designed allow developers to generate any kind of HTML result for display, you can use either **MatrixRenderer** interface or **Template** function to render the content for Biglistbox.

## Properties



- **cols:** specify column size for viewport.
- **rows:** specify row size for viewport.
- **colWidth:** specify column width for each column.
- **rowHeight:** specify row height for each row.
- **matrixRenderer:** specify matrix renderer.
- **model:** specify matrix model.
- **oddRowSclass:** specify sclass for this component to stripe.
- **frozenCols:** specify how many columns are frozen.
- **fixFrozenCols:** specify whether to enable the fix frozen columns that user cannot change the columns size dynamically.
- **autoCols:** specify whether to enable the auto-adjusting cols' size. Default: **true**
- **autoRows:** specify whether to enable the auto-adjusting rows' size. Default: **true**

## Custom Attributes

### org.zkoss.zkmax.zul.biglistbox.preloadSize

```
[default: 50]
[inherit: true][6]
```

It specifies the number of items to preload when receiving the rendering request from the client. It is used only if live data (Biglistbox.setModel(org.zkoss.zkmax.zul.MatrixModel))<sup>[7]</sup>.

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Biglistbox.html#>

[2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/big/Biglistbox.html#>

[3] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/MatrixModel.html#>

[4] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/AbstractListModel.html#>

[5] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/MatrixRenderer.html#>

[6] The custom attribute could be specified in this component, or any of its ancestor. In addition, it could be specified as a library property to enable or disable it for the whole application.

[7] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Biglistbox.html#setModel\(org.zkoss.zkmax.zul.MatrixModel\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Biglistbox.html#setModel(org.zkoss.zkmax.zul.MatrixModel))

## Supported Events

Name	Event Type
onSelect	<b>Event:</b> SelectEvent ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event&gt;SelectEvent.html#">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event&gt;SelectEvent.html#</a> ) Represents an event caused by user's list selection is changed at the client.
onSort	<b>Event:</b> SortEventExt ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/event/SortEventExt.html#">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/event/SortEventExt.html#</a> ) Represents an event that indicates a sorting request to data for Biglistbox and provides more information about the column index.
onScroll	<b>Event:</b> ScrollEventExt ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/event/ScrollEventExt.html#">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/event/ScrollEventExt.html#</a> ) Represents an event caused by that user is scrolling or has scrolled at the client for Biglistbox component and provides more information about the position X and Y data.
onScrollX	<b>Event:</b> ScrollEventExt ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/event/ScrollEventExt.html#">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/event/ScrollEventExt.html#</a> ) Represents an event caused by that user is scrolling or has scrolled the X-axis at the client for Biglistbox component and provides more information about the position X and Y data.
onScrollY	<b>Event:</b> ScrollEventExt ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/event/ScrollEventExt.html#">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/event/ScrollEventExt.html#</a> ) Represents an event caused by that user is scrolling or has scrolled the Y-axis at the client for Biglistbox component and provides more information about the position X and Y data.
onCellClick	<b>Event:</b> CellClickEvent ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/event/CellClickEvent.html#">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/event/CellClickEvent.html#</a> ) Represents an event that indicates a clicking on a cell data for a matrix data component like Biglistbox, and provides more information about the row index and the column index.
onAfterRender	<b>Event:</b> Event ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/Event.html#">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/Event.html#</a> ) Notifies one that the model's data has been rendered.

- Inherited Supported Events: XulElement

## Supported Molds

- The default mold

## Supported Children

None

## Use Cases

Version	Description	Example Location
6.0.1+	Handling a Trillion Data Using ZK	Small Talks

# Diagrams and Reports

---

This section outlines components which are used to create interactive charts and reports.

## Chart

---

### Chart

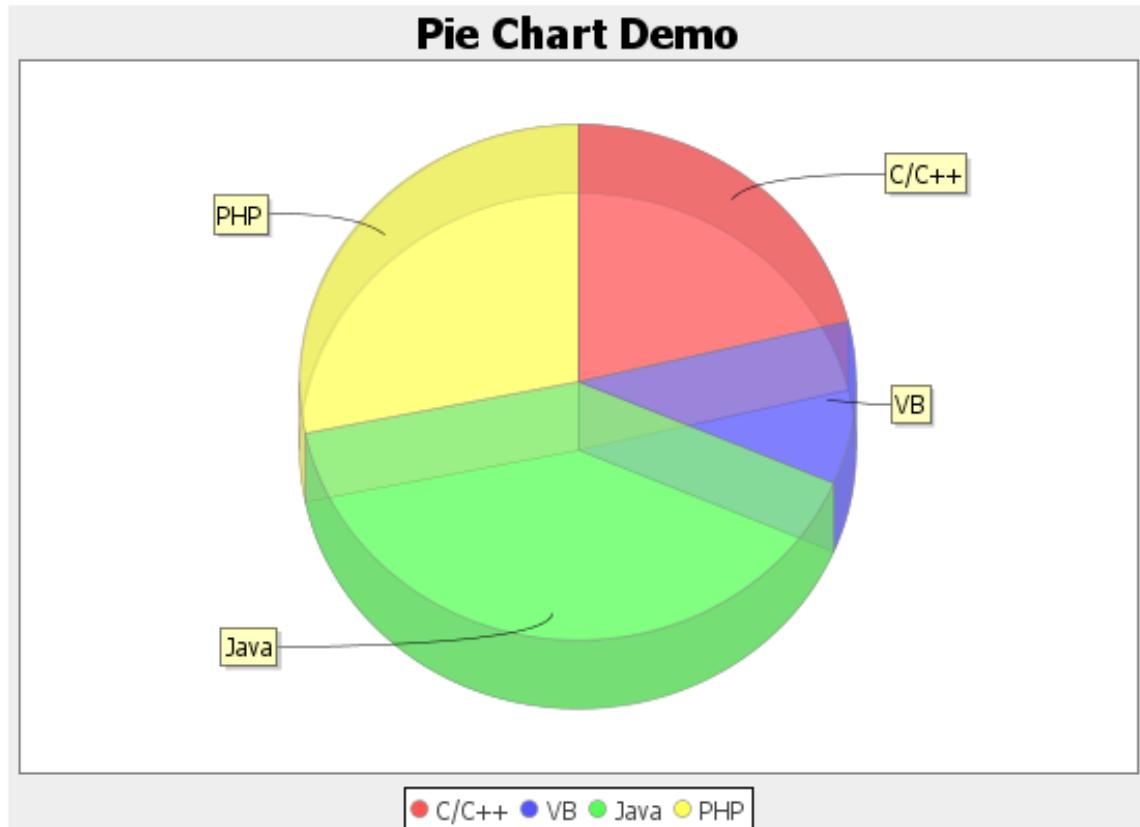
- Demonstration: Chart <sup>[1]</sup>
- Java API: Chart <sup>[2]</sup>
- JavaScript API: Chart <sup>[3]</sup>
- Style Guide: N/A
- JfreeChart Engine Available in ZK PE and EE only <sup>[13]</sup>

## Employment/Purpose

A chart is used to show a set of data as a graph. It helps users to judge things with a snapshot. To use a chart component, developers must prepare a ChartModel and a ChartEngine. Developers also set proper chart type, and the threeD (3D) attribute to draw proper chart. The model and type must match to each other; or the result is unpredictable. The 3D chart is not supported on all chart type.

## Example

### Pie Chart

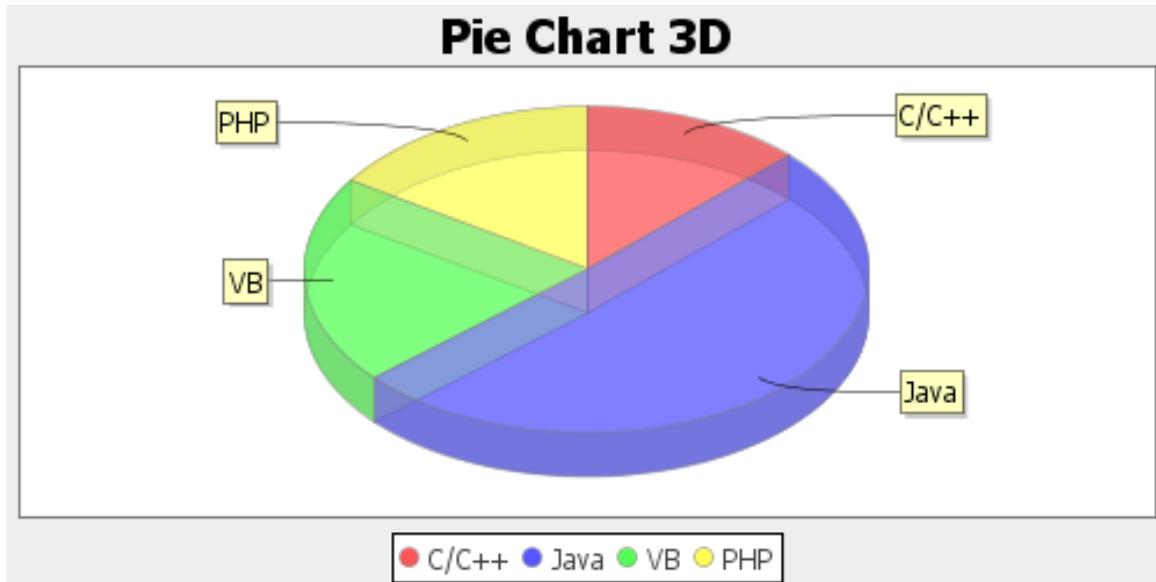


```
<chart id="mychart" title="Pie Chart Demo" width="550" height="400"
 type="pie" threeD="true" fgAlpha="128">
 <attribute name="onClick"><! [CDATA[
 Area area = event.getAreaComponent();
 if (area != null)

 alert(""+area.getAttribute("entity")+": "+area.getTooltiptext());
]]></attribute>
 <zscript><! [CDATA[
 PieModel model = new SimplePieModel();
 model.setValue("C/C++", new Double(21.2));
 model.setValue("VB", new Double(10.2));
 model.setValue("Java", new Double(40.4));
 model.setValue("PHP", new Double(28.2));
 mychart.setModel(model);
]]>
```

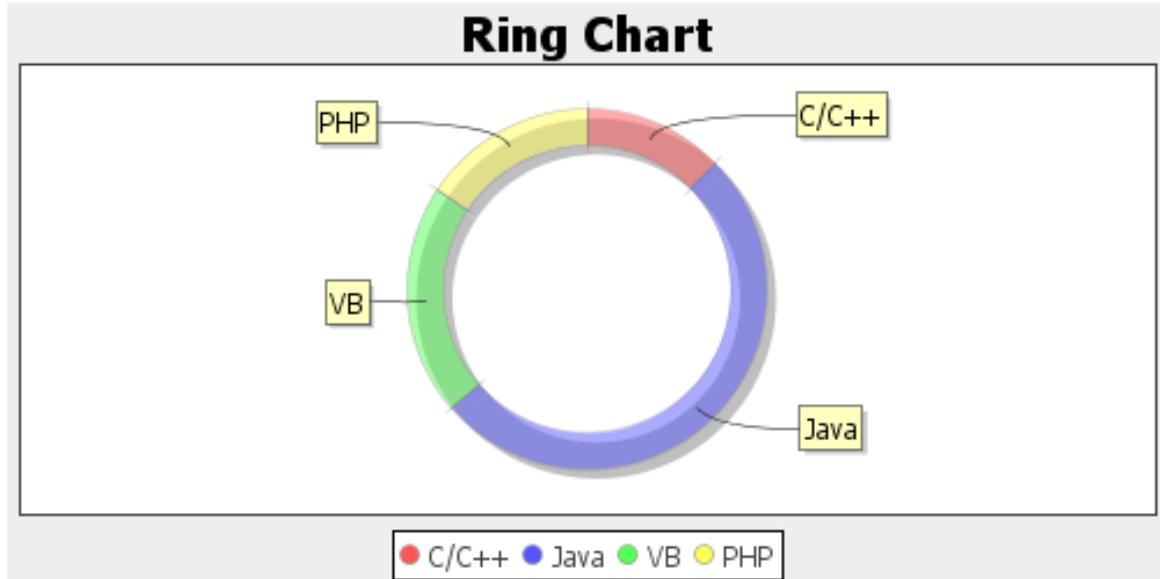
```
]]></zscript>
</chart>
```

## Pie Chart 3D



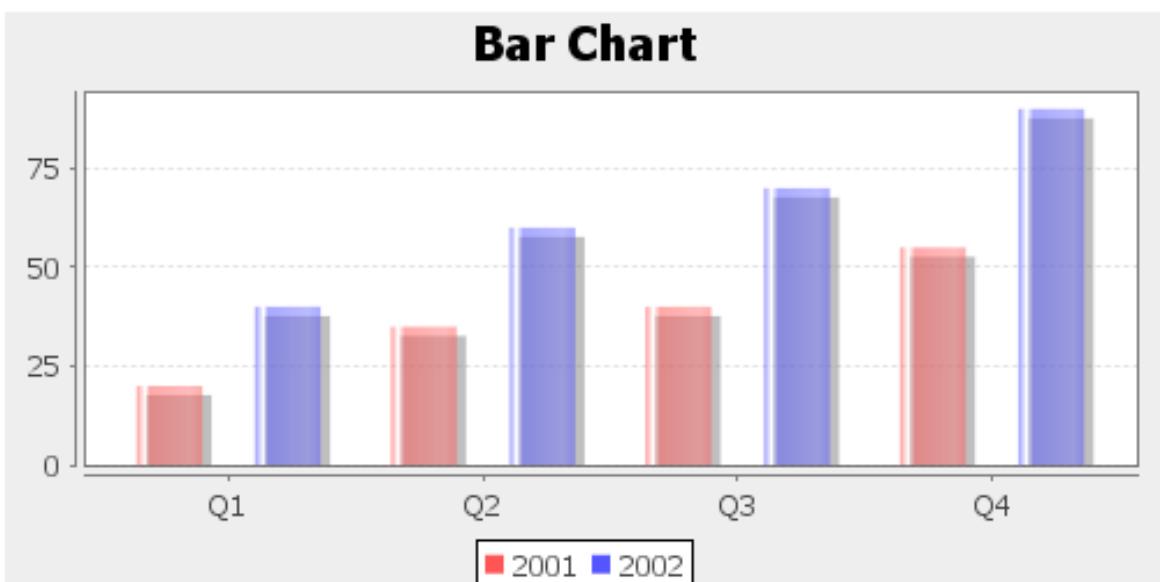
```
<chart id="piechart3d" title="Pie Chart 3D" width="500" height="250"
 type="pie" threeD="true" fgAlpha="128">
<zscript><![CDATA[
 PieModel piemodel = new SimplePieModel();
 piemodel.setValue("C/C++", new Double(12.5));
 piemodel.setValue("Java", new Double(50.2));
 piemodel.setValue("VB", new Double(20.5));
 piemodel.setValue("PHP", new Double(15.5));
 piechart3d.setModel(piemodel);
]]></zscript>
</chart>
```

## Ring Chart



```
<chart id="ringchart" title="Ring Chart" width="500" height="250"
 type="ring" threeD="false" fgAlpha="128">
<zscript><![CDATA[
 PieModel piemodel = new SimplePieModel();
 piemodel.setValue("C/C++", new Double(12.5));
 piemodel.setValue("Java", new Double(50.2));
 piemodel.setValue("VB", new Double(20.5));
 piemodel.setValue("PHP", new Double(15.5));
 ringchart.setModel(piemodel);
]]></zscript>
</chart>
```

## Bar Chart



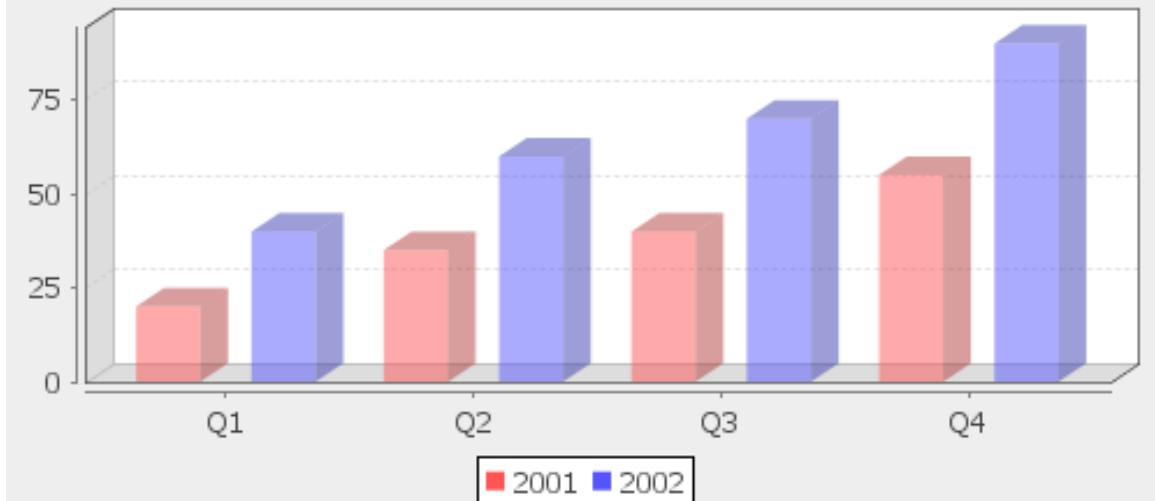
```
<chart id="barchart" title="Bar Chart" width="500" height="250"
 type="bar" threeD="false" fgAlpha="128">
```

```

<zscript><![CDATA[
 CategoryModel catmodel = new SimpleCategoryModel();
 catmodel.setValue("2001", "Q1", new Integer(20));
 catmodel.setValue("2001", "Q2", new Integer(35));
 catmodel.setValue("2001", "Q3", new Integer(40));
 catmodel.setValue("2001", "Q4", new Integer(55));
 catmodel.setValue("2002", "Q1", new Integer(40));
 catmodel.setValue("2002", "Q2", new Integer(60));
 catmodel.setValue("2002", "Q3", new Integer(70));
 catmodel.setValue("2002", "Q4", new Integer(90));
 barchart.setModel(catmodel);
]]></zscript>
</chart>
```

## Bar Chart 3D

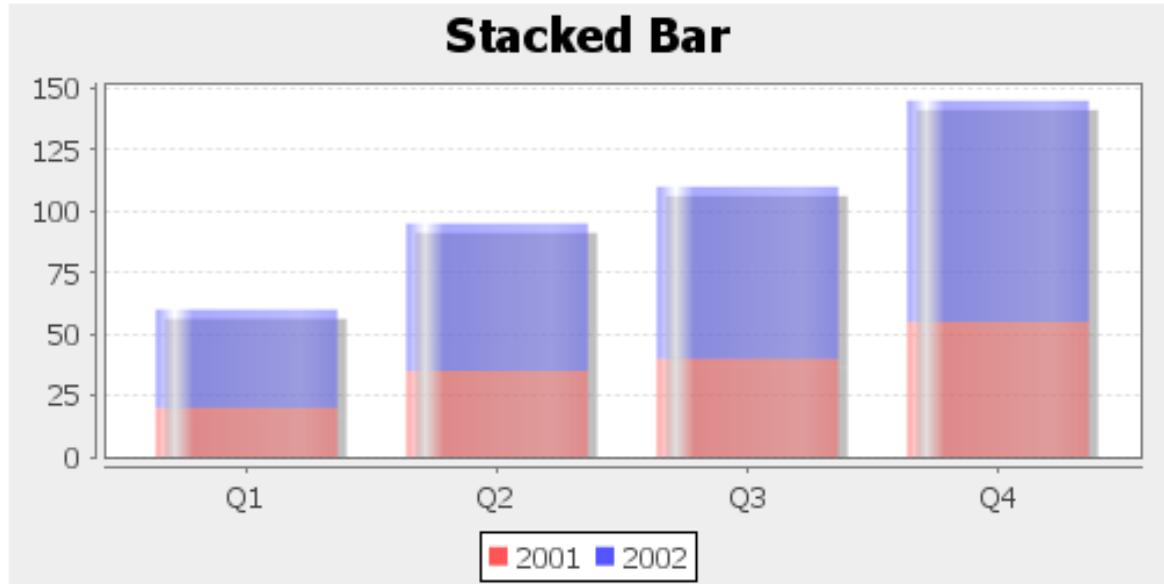
**Bar Chart 3D**



```

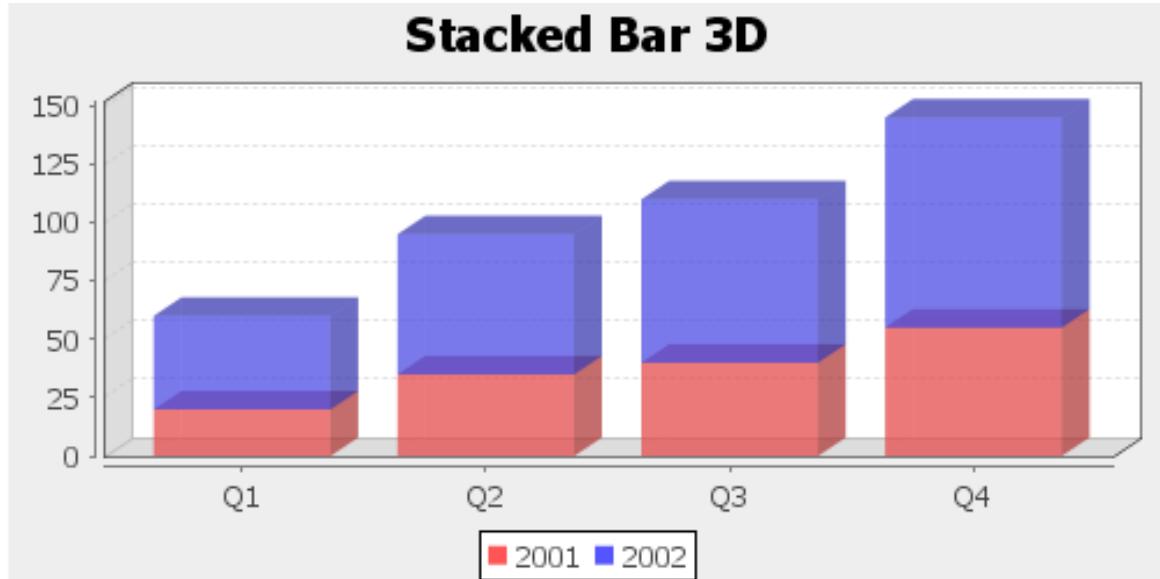
<chart id="barchart3d" title="Bar Chart 3D" width="500" height="250"
 type="bar" threeD="true" fgAlpha="128">
<zscript><![CDATA[
 CategoryModel catmodel = new SimpleCategoryModel();
 catmodel.setValue("2001", "Q1", new Integer(20));
 catmodel.setValue("2001", "Q2", new Integer(35));
 catmodel.setValue("2001", "Q3", new Integer(40));
 catmodel.setValue("2001", "Q4", new Integer(55));
 catmodel.setValue("2002", "Q1", new Integer(40));
 catmodel.setValue("2002", "Q2", new Integer(60));
 catmodel.setValue("2002", "Q3", new Integer(70));
 catmodel.setValue("2002", "Q4", new Integer(90));
 barchart3d.setModel(catmodel);
]]></zscript>
</chart>
```

## Stacked Bar



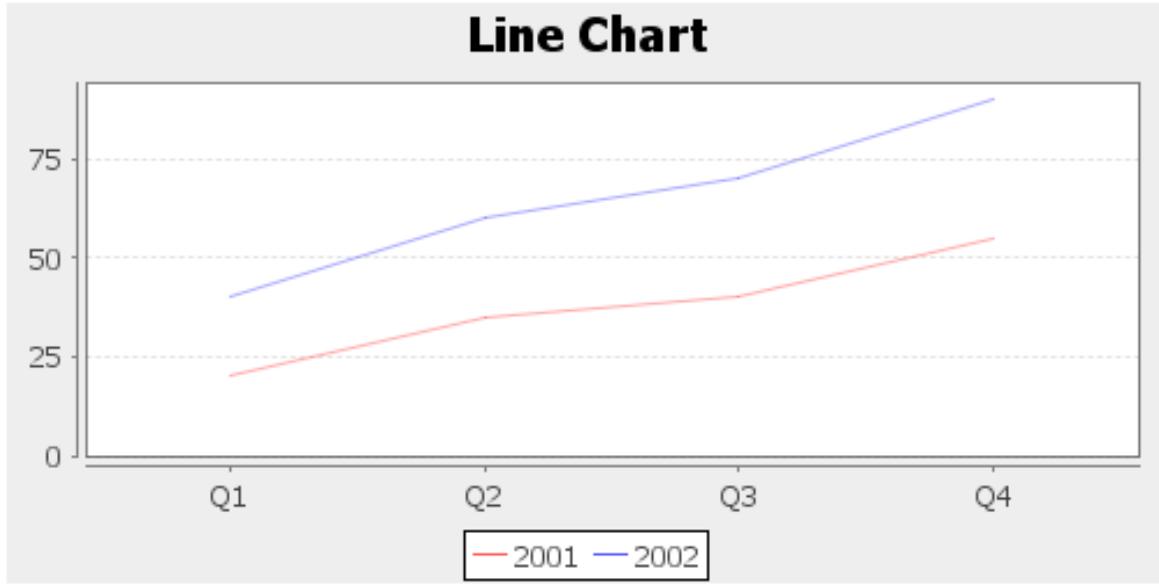
```
<chart id="sbarchart" title="Stacked Bar" width="500" height="250"
 type="stacked_bar" threeD="false" fgAlpha="128">
<zscript><![CDATA[
 CategoryModel catmodel = new SimpleCategoryModel();
 catmodel.setValue("2001", "Q1", new Integer(20));
 catmodel.setValue("2001", "Q2", new Integer(35));
 catmodel.setValue("2001", "Q3", new Integer(40));
 catmodel.setValue("2001", "Q4", new Integer(55));
 catmodel.setValue("2002", "Q1", new Integer(40));
 catmodel.setValue("2002", "Q2", new Integer(60));
 catmodel.setValue("2002", "Q3", new Integer(70));
 catmodel.setValue("2002", "Q4", new Integer(90));
 sbarchart.setModel(catmodel);
]]></zscript>
</chart>
```

## Stacked Bar 3D



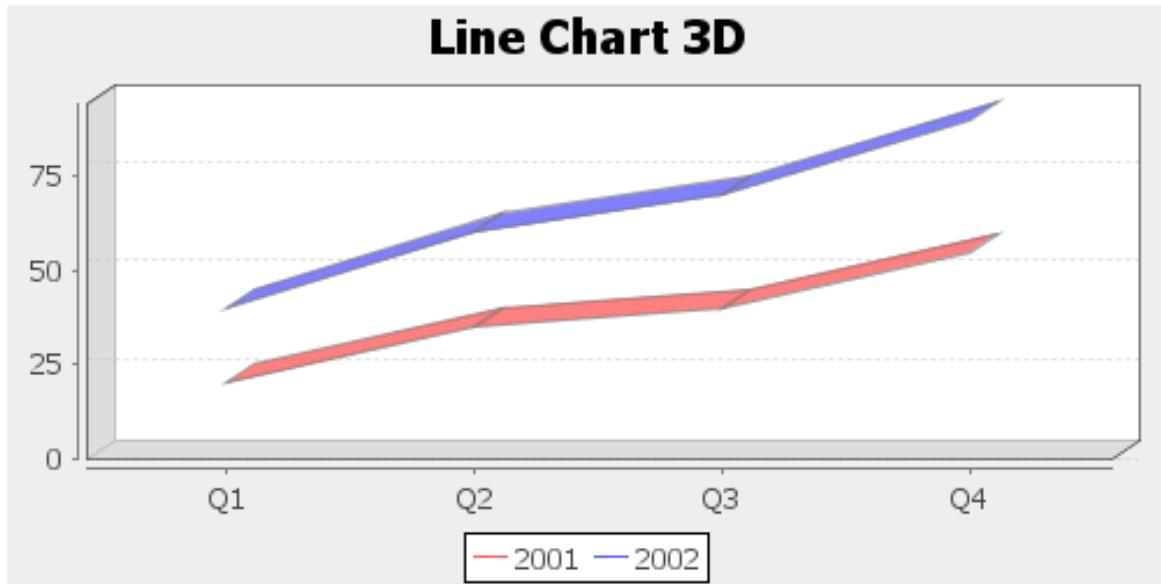
```
<chart id="sbarchart3d" title="Stacked Bar 3D" width="500"
 height="250" type="stacked_bar" threeD="true" fgAlpha="128">
<zscript><![CDATA[
 CategoryModel catmodel = new SimpleCategoryModel();
 catmodel.setValue("2001", "Q1", new Integer(20));
 catmodel.setValue("2001", "Q2", new Integer(35));
 catmodel.setValue("2001", "Q3", new Integer(40));
 catmodel.setValue("2001", "Q4", new Integer(55));
 catmodel.setValue("2002", "Q1", new Integer(40));
 catmodel.setValue("2002", "Q2", new Integer(60));
 catmodel.setValue("2002", "Q3", new Integer(70));
 catmodel.setValue("2002", "Q4", new Integer(90));
 sbarchart3d.setModel(catmodel);
]]></zscript>
</chart>
```

## Line Chart



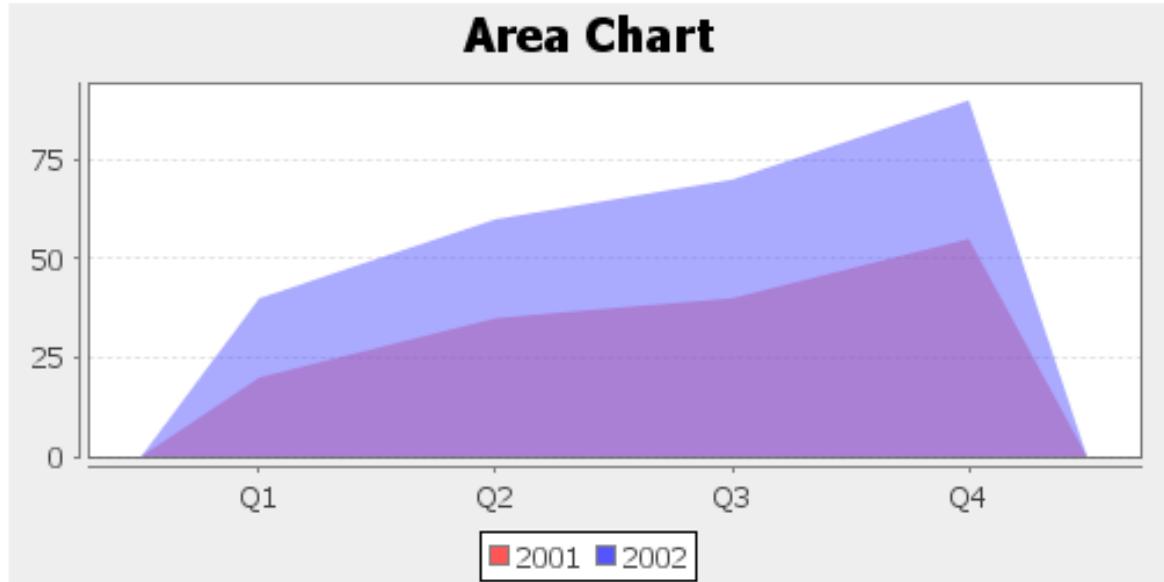
```
<chart id="linechart" title="Line Chart" width="500" height="250">
 type="line" threeD="false" fgAlpha="128">
<zscript><![CDATA[
 CategoryModel catmodel = new SimpleCategoryModel();
 catmodel.setValue("2001", "Q1", new Integer(20));
 catmodel.setValue("2001", "Q2", new Integer(35));
 catmodel.setValue("2001", "Q3", new Integer(40));
 catmodel.setValue("2001", "Q4", new Integer(55));
 catmodel.setValue("2002", "Q1", new Integer(40));
 catmodel.setValue("2002", "Q2", new Integer(60));
 catmodel.setValue("2002", "Q3", new Integer(70));
 catmodel.setValue("2002", "Q4", new Integer(90));
 linechart.setModel(catmodel);
]]></zscript>
</chart>
```

## Line Chart 3D



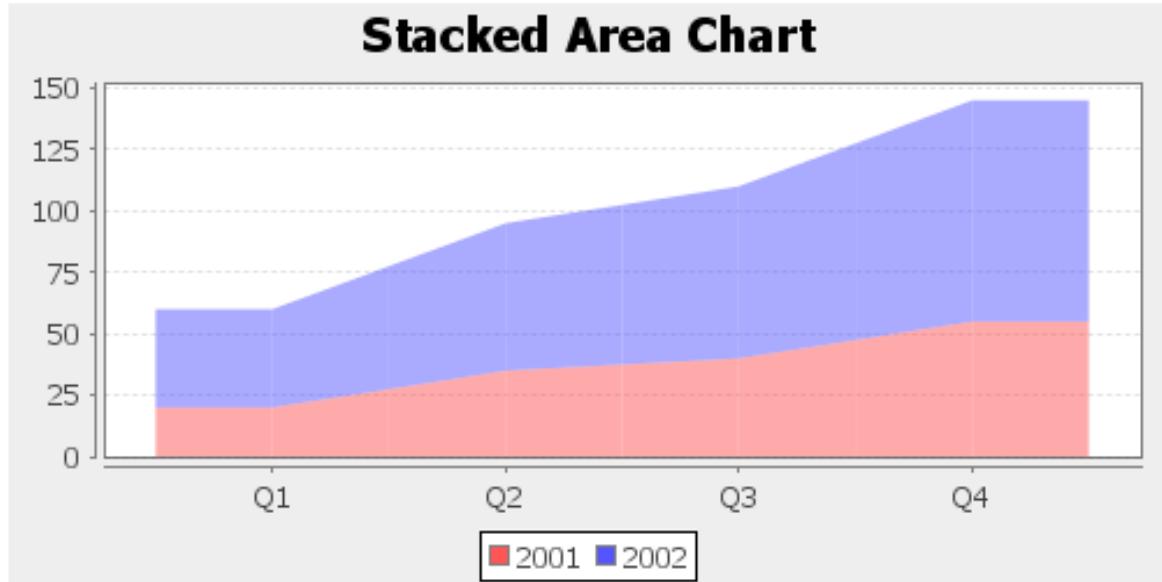
```
<chart id="linechart3d" title="Line Chart 3D" width="500"
 height="250" type="line" threeD="true" fgAlpha="128">
<zscript><![CDATA[
 CategoryModel catmodel = new SimpleCategoryModel();
 catmodel.setValue("2001", "Q1", new Integer(20));
 catmodel.setValue("2001", "Q2", new Integer(35));
 catmodel.setValue("2001", "Q3", new Integer(40));
 catmodel.setValue("2001", "Q4", new Integer(55));
 catmodel.setValue("2002", "Q1", new Integer(40));
 catmodel.setValue("2002", "Q2", new Integer(60));
 catmodel.setValue("2002", "Q3", new Integer(70));
 catmodel.setValue("2002", "Q4", new Integer(90));
 linechart3d.setModel(catmodel);
]]></zscript>
</chart>
```

## Area Chart



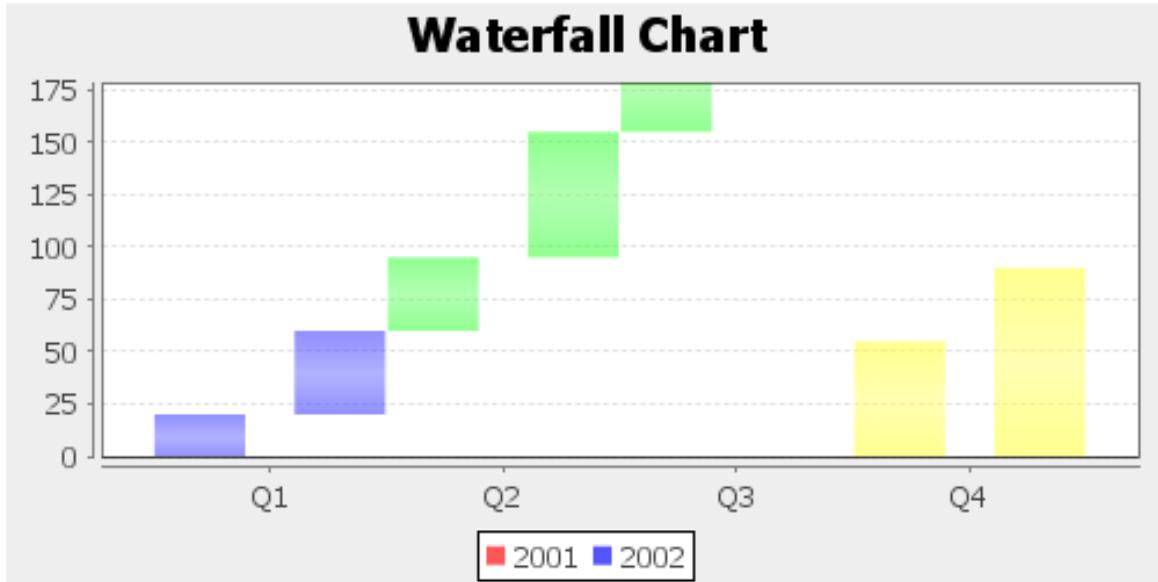
```
<chart id="areachart" title="Area Chart" width="500" height="250">
 type="area" threeD="false" fgAlpha="128">
<zscript><![CDATA[
 CategoryModel catmodel = new SimpleCategoryModel();
 catmodel.setValue("2001", "Q1", new Integer(20));
 catmodel.setValue("2001", "Q2", new Integer(35));
 catmodel.setValue("2001", "Q3", new Integer(40));
 catmodel.setValue("2001", "Q4", new Integer(55));
 catmodel.setValue("2002", "Q1", new Integer(40));
 catmodel.setValue("2002", "Q2", new Integer(60));
 catmodel.setValue("2002", "Q3", new Integer(70));
 catmodel.setValue("2002", "Q4", new Integer(90));
 areachart.setModel(catmodel);
]]></zscript>
</chart>
```

## Stacked Area Chart



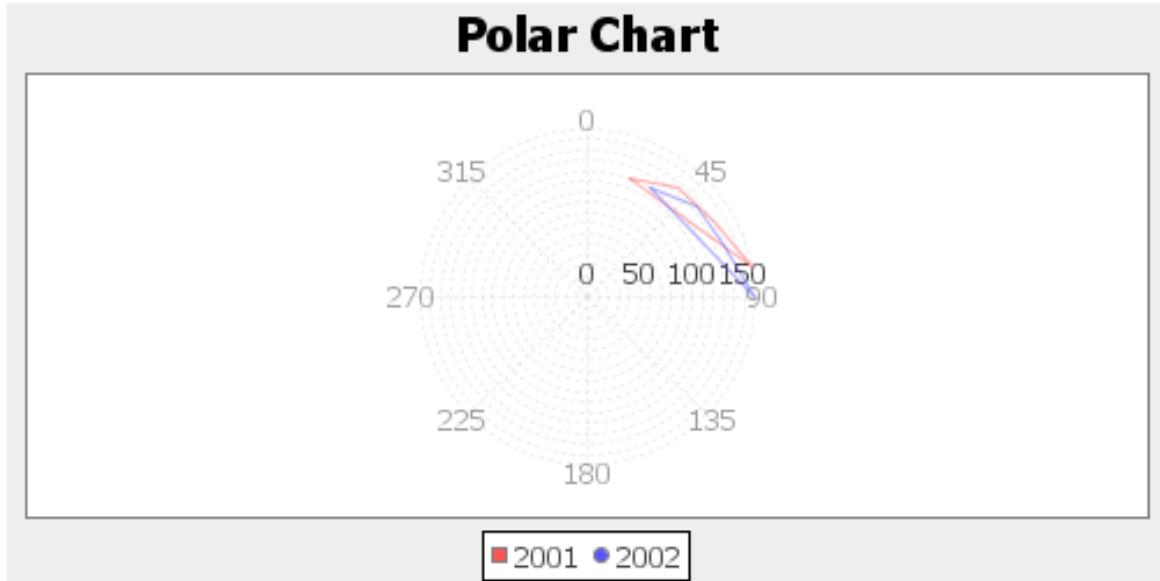
```
<chart id="sareachart" title="Stacked Area Chart" width="500"
 height="250" type="stacked_area" threeD="false"
 fgAlpha="128">
 <zscript><![CDATA[
 CategoryModel catmodel = new SimpleCategoryModel();
 catmodel.setValue("2001", "Q1", new Integer(20));
 catmodel.setValue("2001", "Q2", new Integer(35));
 catmodel.setValue("2001", "Q3", new Integer(40));
 catmodel.setValue("2001", "Q4", new Integer(55));
 catmodel.setValue("2002", "Q1", new Integer(40));
 catmodel.setValue("2002", "Q2", new Integer(60));
 catmodel.setValue("2002", "Q3", new Integer(70));
 catmodel.setValue("2002", "Q4", new Integer(90));
 sareachart.setModel(catmodel);
]]></zscript>
</chart>
```

## Waterfall Chart



```
<chart id="waterfall" title="Waterfall Chart" width="500"
 height="250" type="waterfall" threeD="false" fgAlpha="128">
<zscript><![CDATA[
 CategoryModel catmodel = new SimpleCategoryModel();
 catmodel.setValue("2001", "Q1", new Integer(20));
 catmodel.setValue("2001", "Q2", new Integer(35));
 catmodel.setValue("2001", "Q3", new Integer(40));
 catmodel.setValue("2001", "Q4", new Integer(55));
 catmodel.setValue("2002", "Q1", new Integer(40));
 catmodel.setValue("2002", "Q2", new Integer(60));
 catmodel.setValue("2002", "Q3", new Integer(70));
 catmodel.setValue("2002", "Q4", new Integer(90));
 waterfall.setModel(catmodel);
]]></zscript>
</chart>
```

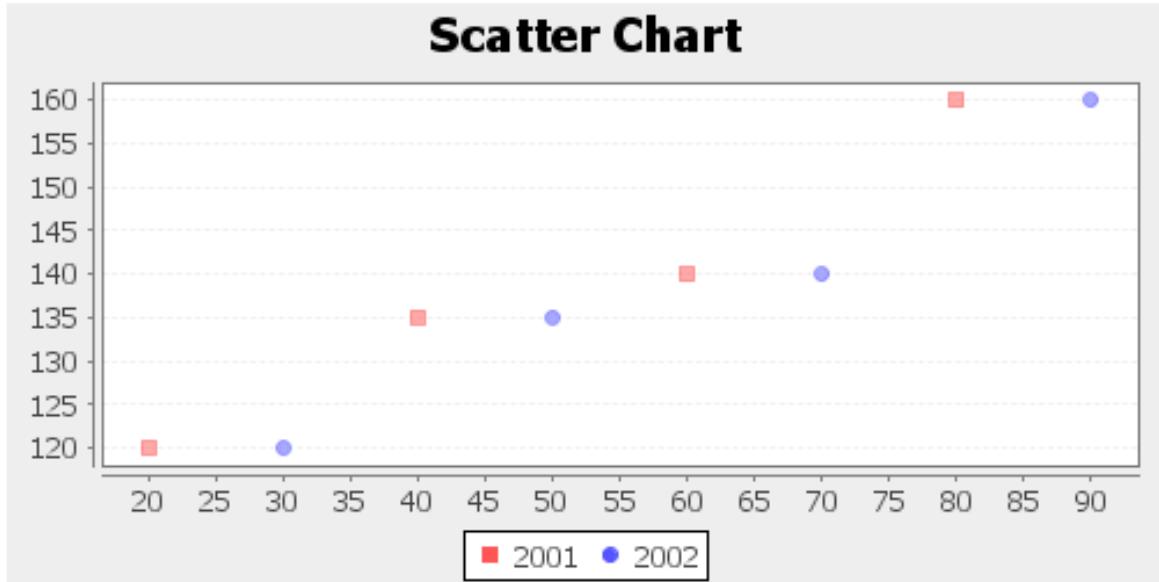
## Polar Chart



```
<chart id="polar" title="Polar Chart" width="500" height="250"
 type="polar" threeD="false" fgAlpha="128">
<zscript><![CDATA[
 XYModel xymodel = new SimpleXYModel();
 xymodel.addValue("2001", new Integer(20), new Integer(120));
 xymodel.addValue("2001", new Integer(40), new Integer(135));
 xymodel.addValue("2001", new Integer(60), new Integer(140));
 xymodel.addValue("2001", new Integer(80), new Integer(160));

 xymodel.addValue("2002", new Integer(30), new Integer(120));
 xymodel.addValue("2002", new Integer(50), new Integer(135));
 xymodel.addValue("2002", new Integer(70), new Integer(140));
 xymodel.addValue("2002", new Integer(90), new Integer(160));
 polar.setModel(xymodel);
]]></zscript>
</chart>
```

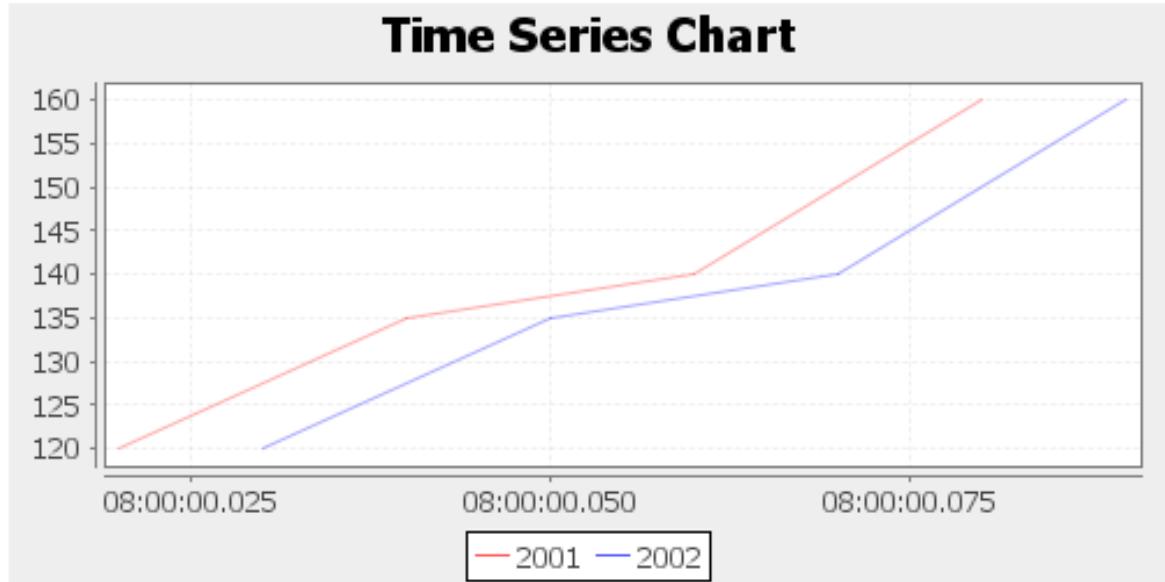
## Scatter Chart



```
<chart id="scatter" title="Scatter Chart" width="500" height="250">
 type="scatter" threeD="false" fgAlpha="128">
<zscript><![CDATA[
 XYModel xymodel = new SimpleXYModel();
 xymodel.addValue("2001", new Integer(20), new Integer(120));
 xymodel.addValue("2001", new Integer(40), new Integer(135));
 xymodel.addValue("2001", new Integer(60), new Integer(140));
 xymodel.addValue("2001", new Integer(80), new Integer(160));

 xymodel.addValue("2002", new Integer(30), new Integer(120));
 xymodel.addValue("2002", new Integer(50), new Integer(135));
 xymodel.addValue("2002", new Integer(70), new Integer(140));
 xymodel.addValue("2002", new Integer(90), new Integer(160));
 scatter.setModel(xymodel);
]]></zscript>
</chart>
```

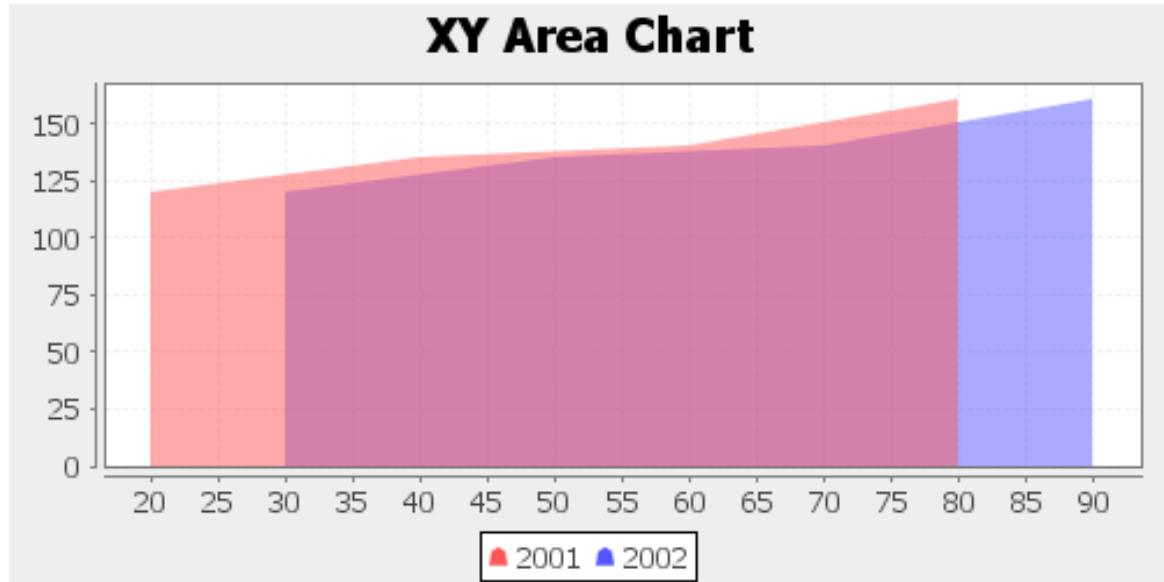
## Time Series Chart



```
<chart id="timeseries" title="Time Series Chart" width="500"
 height="250" type="time_series" threeD="false"
 fgAlpha="128">
 <zscript><![CDATA[
 XYModel xymodel = new SimpleXYModel();
 xymodel.addValue("2001", new Integer(20), new
 Integer(120));
 xymodel.addValue("2001", new Integer(40), new
 Integer(135));
 xymodel.addValue("2001", new Integer(60), new
 Integer(140));
 xymodel.addValue("2001", new Integer(80), new
 Integer(160));

 xymodel.addValue("2002", new Integer(30), new
 Integer(120));
 xymodel.addValue("2002", new Integer(50), new
 Integer(135));
 xymodel.addValue("2002", new Integer(70), new
 Integer(140));
 xymodel.addValue("2002", new Integer(90), new
 Integer(160));
 timeseries.setModel(xymodel);
]]></zscript>
</chart>
```

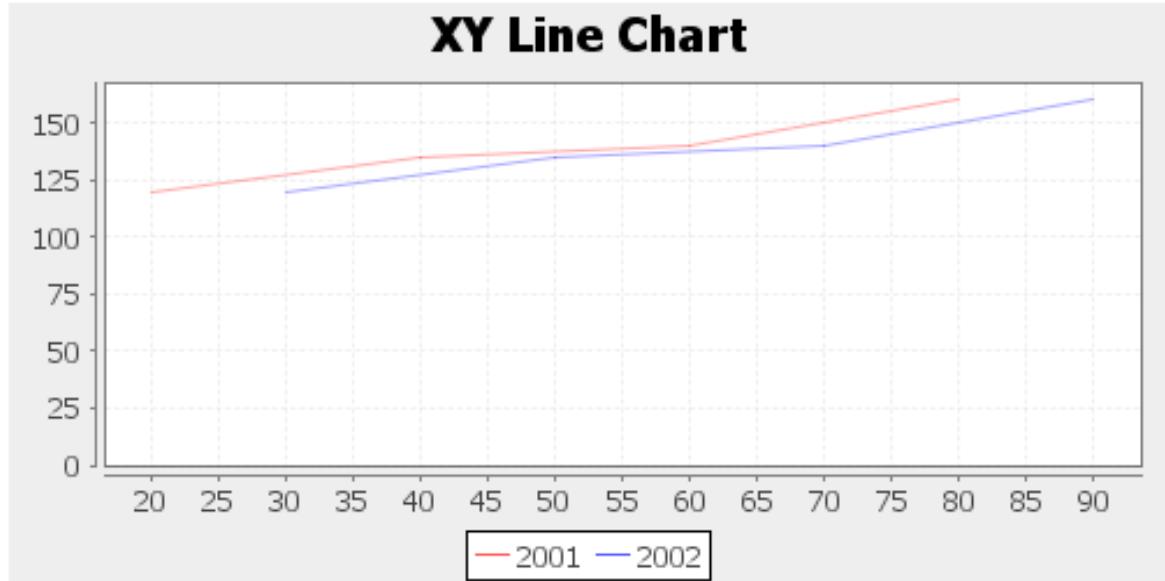
## XY Area Chart



```
<chart id="xyarea" title="XY Area Chart" width="500" height="250">
 type="area" threeD="false" fgAlpha="128">
<zscript><![CDATA[
 XYModel xymodel = new SimpleXYModel();
 xymodel.addValue("2001", new Integer(20), new Integer(120));
 xymodel.addValue("2001", new Integer(40), new Integer(135));
 xymodel.addValue("2001", new Integer(60), new Integer(140));
 xymodel.addValue("2001", new Integer(80), new Integer(160));

 xymodel.addValue("2002", new Integer(30), new Integer(120));
 xymodel.addValue("2002", new Integer(50), new Integer(135));
 xymodel.addValue("2002", new Integer(70), new Integer(140));
 xymodel.addValue("2002", new Integer(90), new Integer(160));
 xyarea.setModel(xymodel);
]]></zscript>
</chart>
```

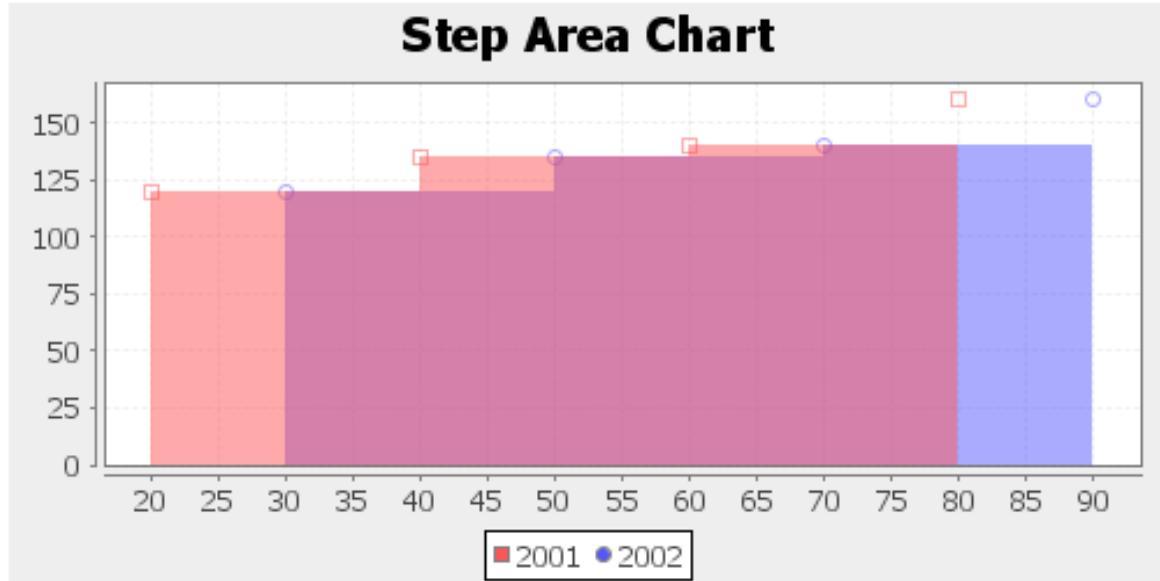
## XY Line Chart



```
<chart id="xyline" title="XY Line Chart" width="500" height="250">
 type="line" threeD="false" fgAlpha="128">
<zscript><![CDATA[
 XYModel xymodel = new SimpleXYModel();
 xymodel.addValue("2001", new Integer(20), new Integer(120));
 xymodel.addValue("2001", new Integer(40), new Integer(135));
 xymodel.addValue("2001", new Integer(60), new Integer(140));
 xymodel.addValue("2001", new Integer(80), new Integer(160));

 xymodel.addValue("2002", new Integer(30), new Integer(120));
 xymodel.addValue("2002", new Integer(50), new Integer(135));
 xymodel.addValue("2002", new Integer(70), new Integer(140));
 xymodel.addValue("2002", new Integer(90), new Integer(160));
 xyline.setModel(xymodel);
]]></zscript>
</chart>
```

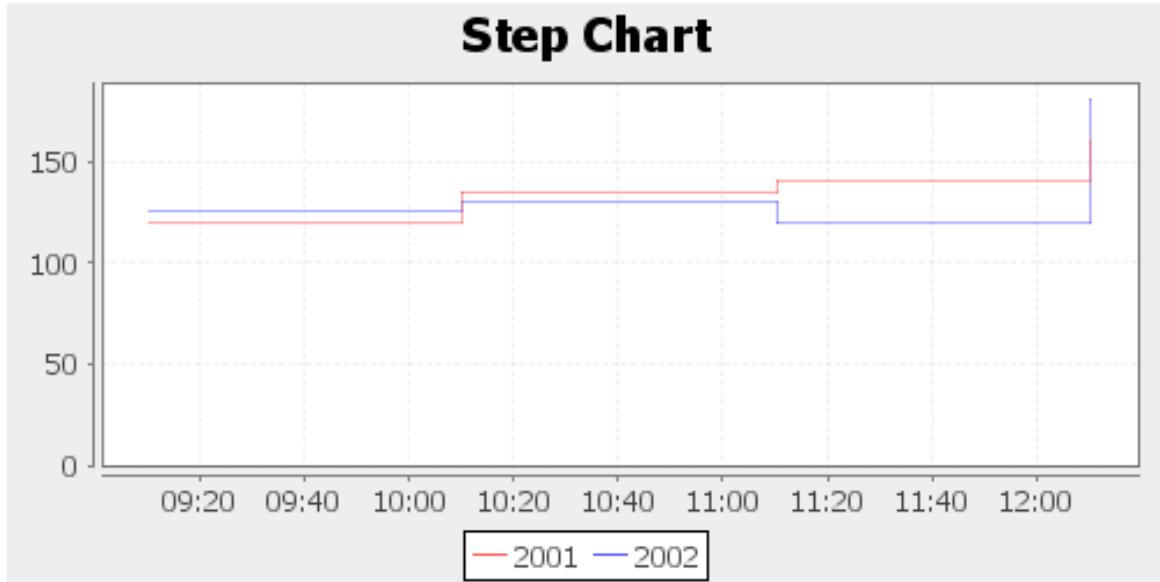
## Step Area Chart



```
<chart id="steparea" title="Step Area Chart" width="500"
 height="250" type="step_area" threeD="false" fgAlpha="128">
<zscript><![CDATA[
 XYModel xymodel = new SimpleXYModel();
 xymodel.addValue("2001", new Integer(20), new Integer(120));
 xymodel.addValue("2001", new Integer(40), new Integer(135));
 xymodel.addValue("2001", new Integer(60), new Integer(140));
 xymodel.addValue("2001", new Integer(80), new Integer(160));

 xymodel.addValue("2002", new Integer(30), new Integer(120));
 xymodel.addValue("2002", new Integer(50), new Integer(135));
 xymodel.addValue("2002", new Integer(70), new Integer(140));
 xymodel.addValue("2002", new Integer(90), new Integer(160));
 steparea.setModel(xymodel);
]]></zscript>
</chart>
```

## Step Chart



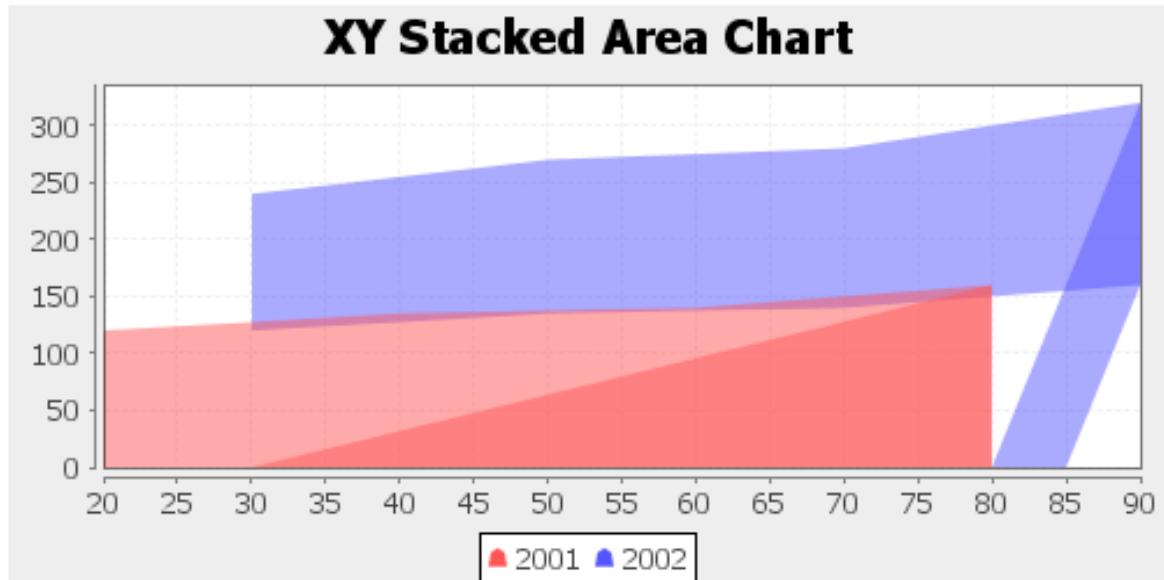
```
<chart id="step" title="Step Chart" width="500" height="250">
 type="step" threeD="false" fgAlpha="128">
<zscript><![CDATA[
 public Date time(int year, int month, int day, int hour, int
minute, int second) {
 final java.util.Calendar calendar =
java.util.Calendar.getInstance(TimeZone.getTimeZone("GMT"));
 calendar.set(year, month-1, day, hour, minute,
second);
 final Date result = calendar.getTime();
 return result;
 }
 XYModel datemode1 = new SimpleXYModel();
 datemode1.addValue("2001", new Long(time(2001, 5, 2, 1, 10,
15).getTime()), new Integer(120));
 datemode1.addValue("2001", new Long(time(2001, 5, 2, 2, 10,
15).getTime()), new Integer(135));
 datemode1.addValue("2001", new Long(time(2001, 5, 2, 3, 10,
15).getTime()), new Integer(140));
 datemode1.addValue("2001", new Long(time(2001, 5, 2, 4, 10,
15).getTime()), new Integer(160));

 datemode1.addValue("2002", new Long(time(2001, 5, 2, 1, 10,
20).getTime()), new Integer(125));
 datemode1.addValue("2002", new Long(time(2001, 5, 2, 2, 10,
20).getTime()), new Integer(130));
 datemode1.addValue("2002", new Long(time(2001, 5, 2, 3, 10,
20).getTime()), new Integer(120));
 datemode1.addValue("2002", new Long(time(2001, 5, 2, 4, 10,
20).getTime()), new Integer(180));
 step.setModel(datemode1);

```

```
]]></zscript>
</chart>
```

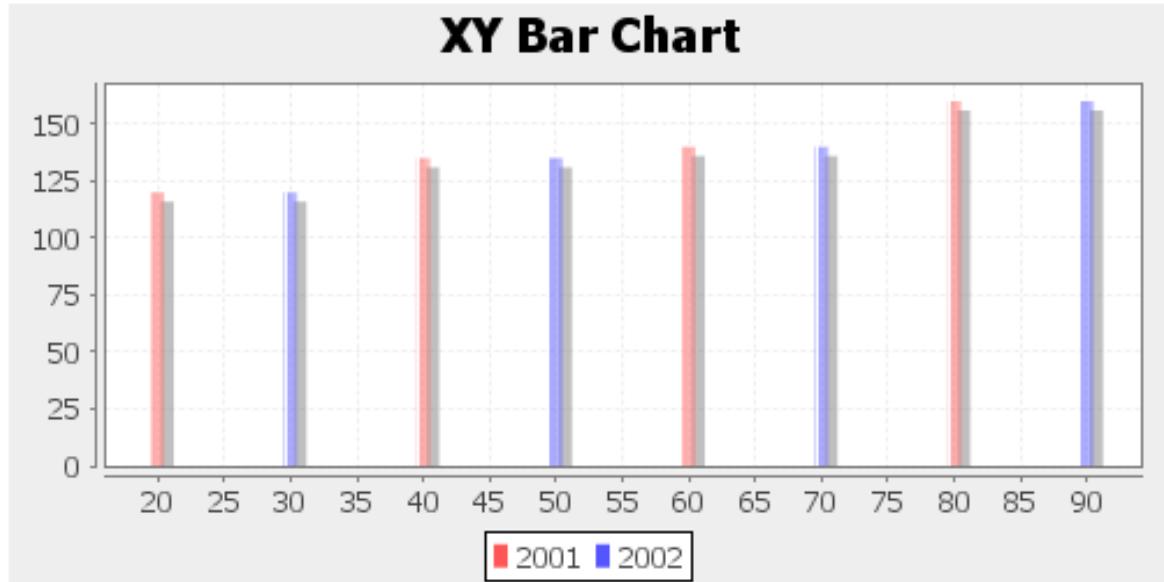
## XY Stacked Area Chart



```
<chart id="xystackedarea" title="XY Stacked Area Chart" width="500"
 height="250" type="stacked_area" threeD="false"
 fgAlpha="128">
 <zscript><![CDATA[
 XYModel xymodel = new SimpleXYModel();
 xymodel.addValue("2001", new Integer(20), new
 Integer(120));
 xymodel.addValue("2001", new Integer(40), new
 Integer(135));
 xymodel.addValue("2001", new Integer(60), new
 Integer(140));
 xymodel.addValue("2001", new Integer(80), new
 Integer(160));

 xymodel.addValue("2002", new Integer(30), new
 Integer(120));
 xymodel.addValue("2002", new Integer(50), new
 Integer(135));
 xymodel.addValue("2002", new Integer(70), new
 Integer(140));
 xymodel.addValue("2002", new Integer(90), new
 Integer(160));
 xystackedarea.setModel(xymodel);
]]></zscript>
</chart>
```

## XY Bar Chart

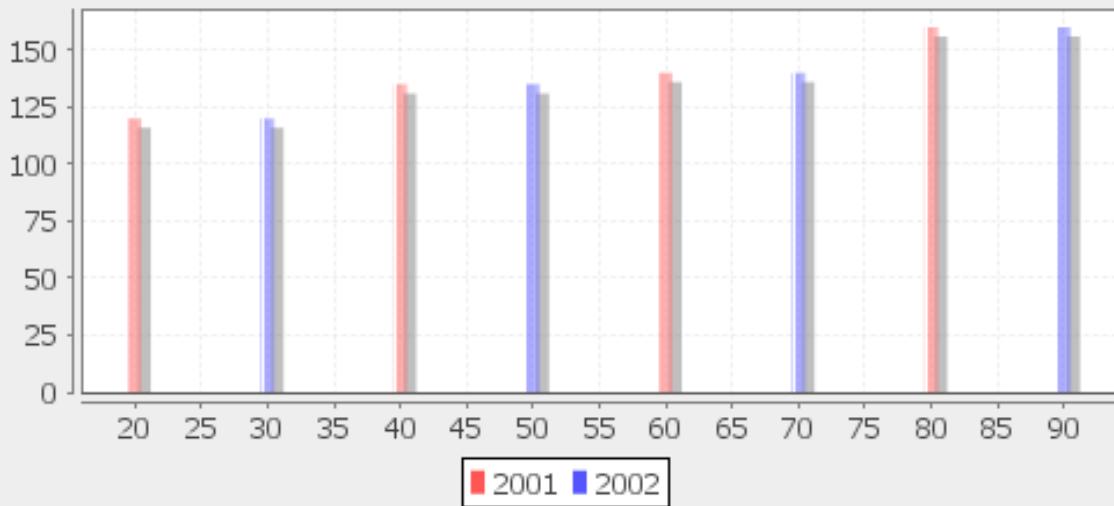


```
<chart id="xybar" title="XY Bar Chart" width="500" height="250"
 type="bar" threeD="false" fgAlpha="128">
<zscript><![CDATA[
 XYModel xymodel = new SimpleXYModel();
 xymodel.addValue("2001", new Integer(20), new Integer(120));
 xymodel.addValue("2001", new Integer(40), new Integer(135));
 xymodel.addValue("2001", new Integer(60), new Integer(140));
 xymodel.addValue("2001", new Integer(80), new Integer(160));

 xymodel.addValue("2002", new Integer(30), new Integer(120));
 xymodel.addValue("2002", new Integer(50), new Integer(135));
 xymodel.addValue("2002", new Integer(70), new Integer(140));
 xymodel.addValue("2002", new Integer(90), new Integer(160));
 xybar.setModel(xymodel);
]]></zscript>
</chart>
```

## Histogram Chart

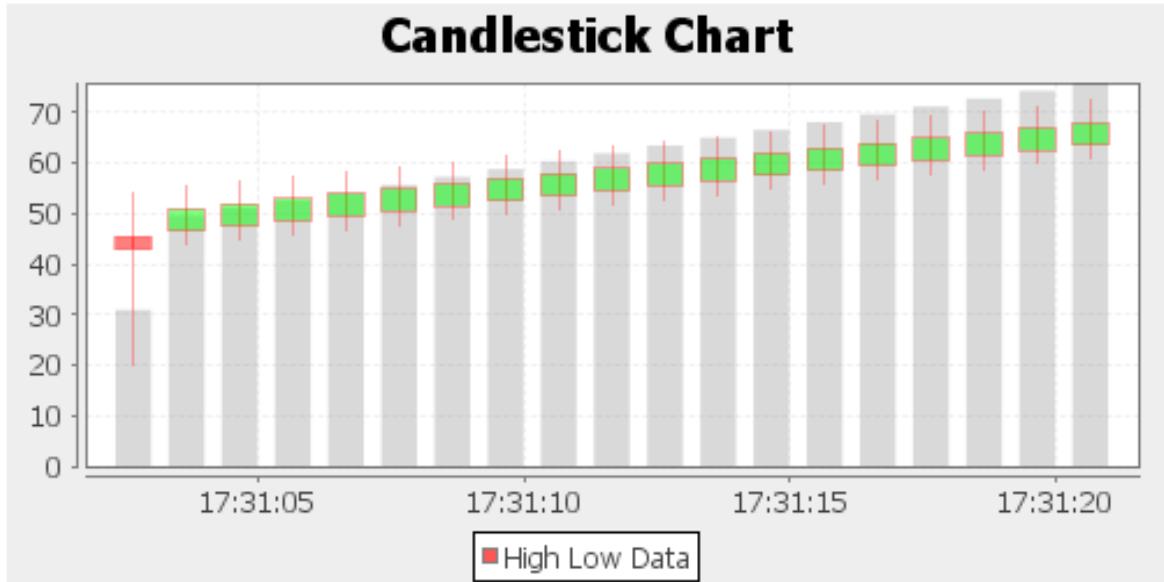
**Histogram Chart**



```
<chart id="histogram" title="Histogram Chart" width="500"
 height="250" type="histogram" threeD="false" fgAlpha="128">
<zscript><![CDATA[
 XYModel xymodel = new SimpleXYModel();
 xymodel.addValue("2001", new Integer(20), new Integer(120));
 xymodel.addValue("2001", new Integer(40), new Integer(135));
 xymodel.addValue("2001", new Integer(60), new Integer(140));
 xymodel.addValue("2001", new Integer(80), new Integer(160));

 xymodel.addValue("2002", new Integer(30), new Integer(120));
 xymodel.addValue("2002", new Integer(50), new Integer(135));
 xymodel.addValue("2002", new Integer(70), new Integer(140));
 xymodel.addValue("2002", new Integer(90), new Integer(160));
 histogram.setModel(xymodel);
]]></zscript>
</chart>
```

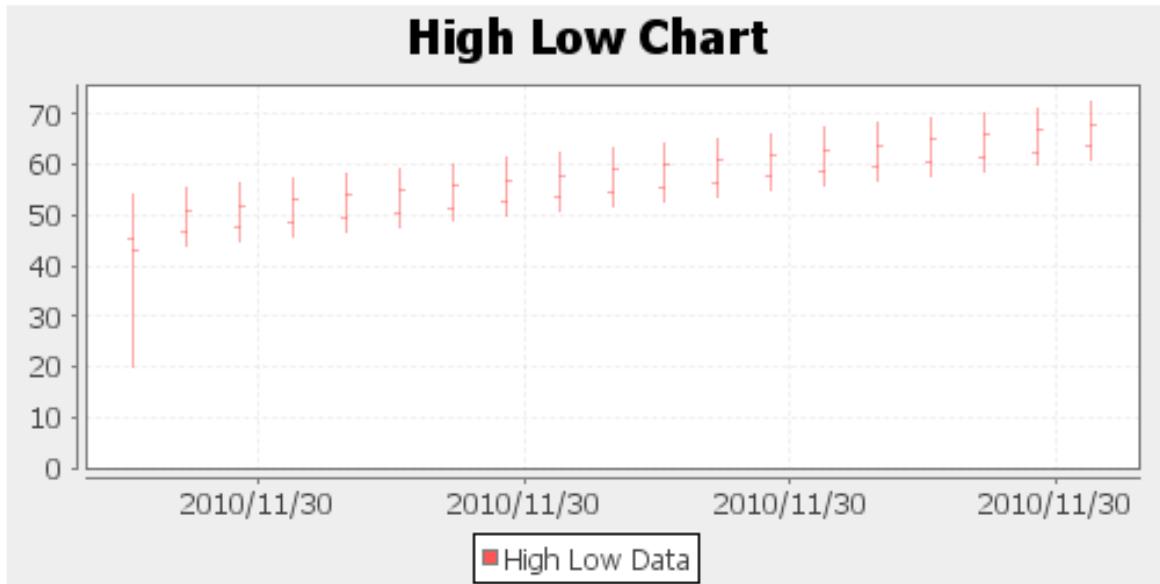
## Candlestick Chart



```
<chart id="candlestick" title="Candlestick Chart" width="500"
 height="250" type="candlestick" threeD="false"
 fgAlpha="128">
<zscript><![CDATA[
 HiLoModel hilomodel = new SimpleHiLoModel();
 long d = System.currentTimeMillis();
 hilomodel.addValue(new Date(d), new Double(45.5), new
Double(54.2), new Double(19.9), new Double(42.8), new Double(20));
 hilomodel.addValue(new Date(d+1000), new Double(46.5),
new Double(55.2), new Double(43.8), new Double(50.9), new
Double(32));
 hilomodel.addValue(new Date(d+2000), new Double(47.5),
new Double(56.2), new Double(44.8), new Double(51.9), new
Double(33));
 hilomodel.addValue(new Date(d+3000), new Double(48.5),
new Double(57.2), new Double(45.8), new Double(52.9), new
Double(34));
 hilomodel.addValue(new Date(d+4000), new Double(49.5),
new Double(58.2), new Double(46.8), new Double(53.9), new
Double(35));
 hilomodel.addValue(new Date(d+5000), new Double(50.5),
new Double(59.2), new Double(47.8), new Double(54.9), new
Double(36));
 hilomodel.addValue(new Date(d+6000), new Double(51.5),
new Double(60.2), new Double(48.8), new Double(55.9), new
Double(37));
 hilomodel.addValue(new Date(d+7000), new Double(52.5),
new Double(61.2), new Double(49.8), new Double(56.9), new
Double(38));
 hilomodel.addValue(new Date(d+8000), new Double(53.5),
new Double(62.2), new Double(50.8), new Double(57.9), new
Double(39));
]]>
```

```
Double(39));
 hilomodel.addValue(new Date(d+9000), new Double(54.5),
new Double(63.2), new Double(51.8), new Double(58.9), new
Double(40));
 hilomodel.addValue(new Date(d+10000), new Double(55.5),
new Double(64.2), new Double(52.8), new Double(59.9), new
Double(41));
 hilomodel.addValue(new Date(d+11000), new Double(56.5),
new Double(65.2), new Double(53.8), new Double(60.9), new
Double(42));
 hilomodel.addValue(new Date(d+12000), new Double(57.5),
new Double(66.2), new Double(54.8), new Double(61.9), new
Double(43));
 hilomodel.addValue(new Date(d+13000), new Double(58.5),
new Double(67.2), new Double(55.8), new Double(62.9), new
Double(44));
 hilomodel.addValue(new Date(d+14000), new Double(59.5),
new Double(68.2), new Double(56.8), new Double(63.9), new
Double(45));
 hilomodel.addValue(new Date(d+15000), new Double(60.5),
new Double(69.2), new Double(57.8), new Double(64.9), new
Double(46));
 hilomodel.addValue(new Date(d+16000), new Double(61.5),
new Double(70.2), new Double(58.8), new Double(65.9), new
Double(47));
 hilomodel.addValue(new Date(d+17000), new Double(62.5),
new Double(71.2), new Double(59.8), new Double(66.9), new
Double(48));
 hilomodel.addValue(new Date(d+18000), new Double(63.5),
new Double(72.2), new Double(60.8), new Double(67.9), new
Double(49));
candlestick.setModel(hilomodel);
]]></zscript>
</chart>
```

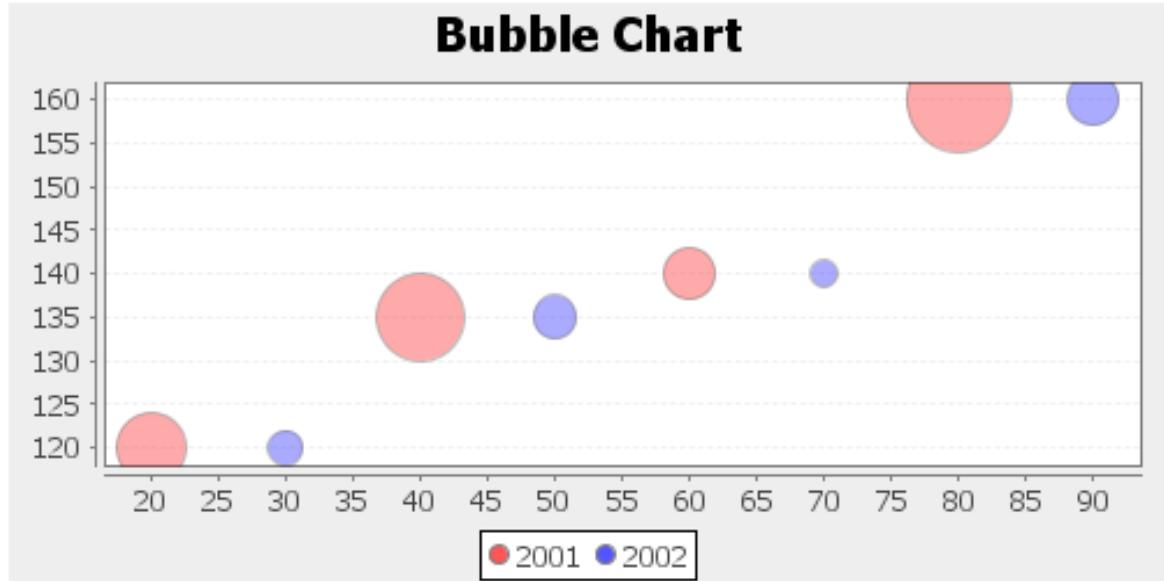
## High Low Chart



```
<chart id="highlow" title="High Low Chart" width="500" height="250"
 type="highlow" threeD="false" fgAlpha="128"
 dateFormat="yyyy/MM/dd">
 <zscript><![CDATA[
 HiLoModel hilomodel = new SimpleHiLoModel();
 long d = System.currentTimeMillis();
 hilomodel.addValue(new Date(d), new Double(45.5), new
Double(54.2), new Double(19.9), new Double(42.8), new Double(20));
 hilomodel.addValue(new Date(d+1000), new Double(46.5),
new Double(55.2), new Double(43.8), new Double(50.9), new
Double(32));
 hilomodel.addValue(new Date(d+2000), new Double(47.5),
new Double(56.2), new Double(44.8), new Double(51.9), new
Double(33));
 hilomodel.addValue(new Date(d+3000), new Double(48.5),
new Double(57.2), new Double(45.8), new Double(52.9), new
Double(34));
 hilomodel.addValue(new Date(d+4000), new Double(49.5),
new Double(58.2), new Double(46.8), new Double(53.9), new
Double(35));
 hilomodel.addValue(new Date(d+5000), new Double(50.5),
new Double(59.2), new Double(47.8), new Double(54.9), new
Double(36));
 hilomodel.addValue(new Date(d+6000), new Double(51.5),
new Double(60.2), new Double(48.8), new Double(55.9), new
Double(37));
 hilomodel.addValue(new Date(d+7000), new Double(52.5),
new Double(61.2), new Double(49.8), new Double(56.9), new
Double(38));
 hilomodel.addValue(new Date(d+8000), new Double(53.5),
new Double(62.2), new Double(50.8), new Double(57.9), new
Double(39));
]]>
```

```
Double(39));
 hilomodel.addValue(new Date(d+9000), new Double(54.5),
new Double(63.2), new Double(51.8), new Double(58.9), new
Double(40));
 hilomodel.addValue(new Date(d+10000), new Double(55.5),
new Double(64.2), new Double(52.8), new Double(59.9), new
Double(41));
 hilomodel.addValue(new Date(d+11000), new Double(56.5),
new Double(65.2), new Double(53.8), new Double(60.9), new
Double(42));
 hilomodel.addValue(new Date(d+12000), new Double(57.5),
new Double(66.2), new Double(54.8), new Double(61.9), new
Double(43));
 hilomodel.addValue(new Date(d+13000), new Double(58.5),
new Double(67.2), new Double(55.8), new Double(62.9), new
Double(44));
 hilomodel.addValue(new Date(d+14000), new Double(59.5),
new Double(68.2), new Double(56.8), new Double(63.9), new
Double(45));
 hilomodel.addValue(new Date(d+15000), new Double(60.5),
new Double(69.2), new Double(57.8), new Double(64.9), new
Double(46));
 hilomodel.addValue(new Date(d+16000), new Double(61.5),
new Double(70.2), new Double(58.8), new Double(65.9), new
Double(47));
 hilomodel.addValue(new Date(d+17000), new Double(62.5),
new Double(71.2), new Double(59.8), new Double(66.9), new
Double(48));
 hilomodel.addValue(new Date(d+18000), new Double(63.5),
new Double(72.2), new Double(60.8), new Double(67.9), new
Double(49));
highlow.setModel(hilomodel);
]]></zscript>
</chart>
```

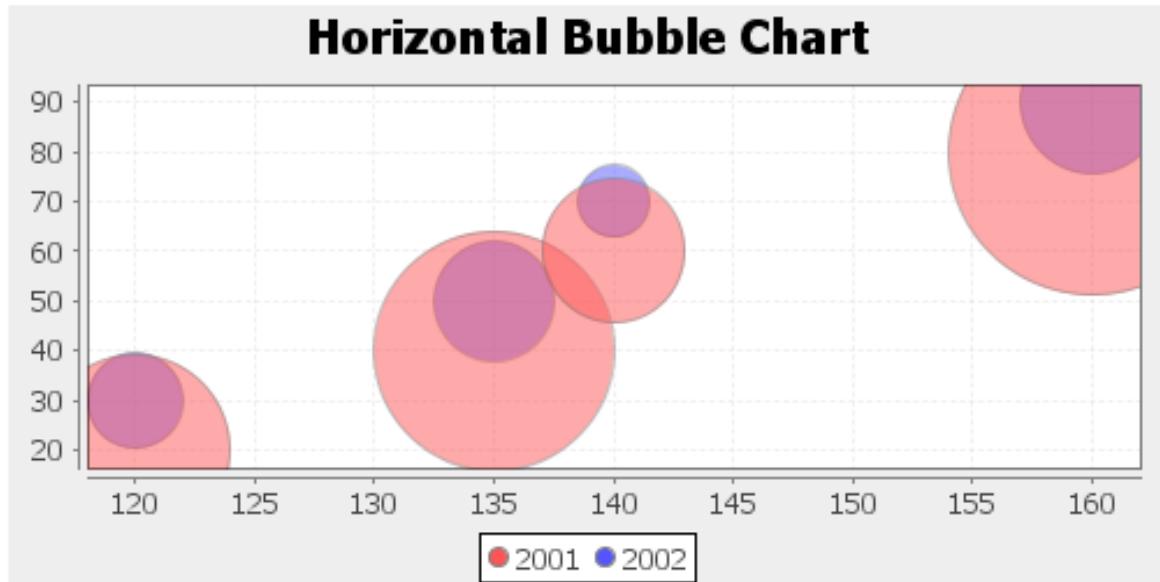
## Bubble Chart



```
<chart id="bubble" title="Bubble Chart" width="500" height="250">
 type="bubble" threeD="false" fgAlpha="128">
<zscript><![CDATA[
 XYZModel xyzmodel = new SimpleXYZModel();
 xyzmodel.addValue("2001", new Integer(20), new
 Integer(120), new Integer(8));
 xyzmodel.addValue("2001", new Integer(40), new
 Integer(135), new Integer(10));
 xyzmodel.addValue("2001", new Integer(60), new
 Integer(140), new Integer(6));
 xyzmodel.addValue("2001", new Integer(80), new
 Integer(160), new Integer(12));

 xyzmodel.addValue("2002", new Integer(30), new
 Integer(120), new Integer(4));
 xyzmodel.addValue("2002", new Integer(50), new
 Integer(135), new Integer(5));
 xyzmodel.addValue("2002", new Integer(70), new
 Integer(140), new Integer(3));
 xyzmodel.addValue("2002", new Integer(90), new
 Integer(160), new Integer(6));
 bubble.setModel(xyzmodel);
]]></zscript>
</chart>
```

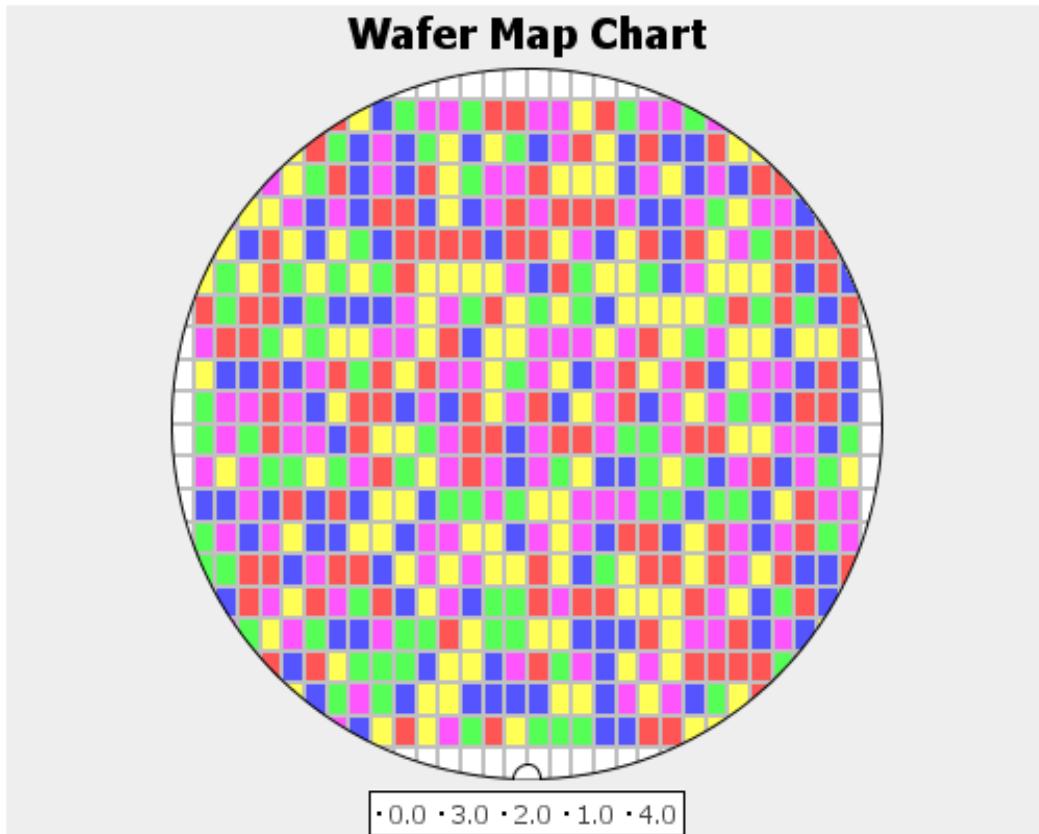
## Horizontal Bubble Chart



```
<chart id="hbubble" title="Horizontal Bubble Chart" width="500"
 height="250" type="bubble" threeD="false" fgAlpha="128"
 orient="horizontal">
 <zscript><![CDATA[
 XYZModel xyzmodel = new SimpleXYZModel();
 xyzmodel.addValue("2001", new Integer(20), new
 Integer(120), new Integer(8));
 xyzmodel.addValue("2001", new Integer(40), new
 Integer(135), new Integer(10));
 xyzmodel.addValue("2001", new Integer(60), new
 Integer(140), new Integer(6));
 xyzmodel.addValue("2001", new Integer(80), new
 Integer(160), new Integer(12));

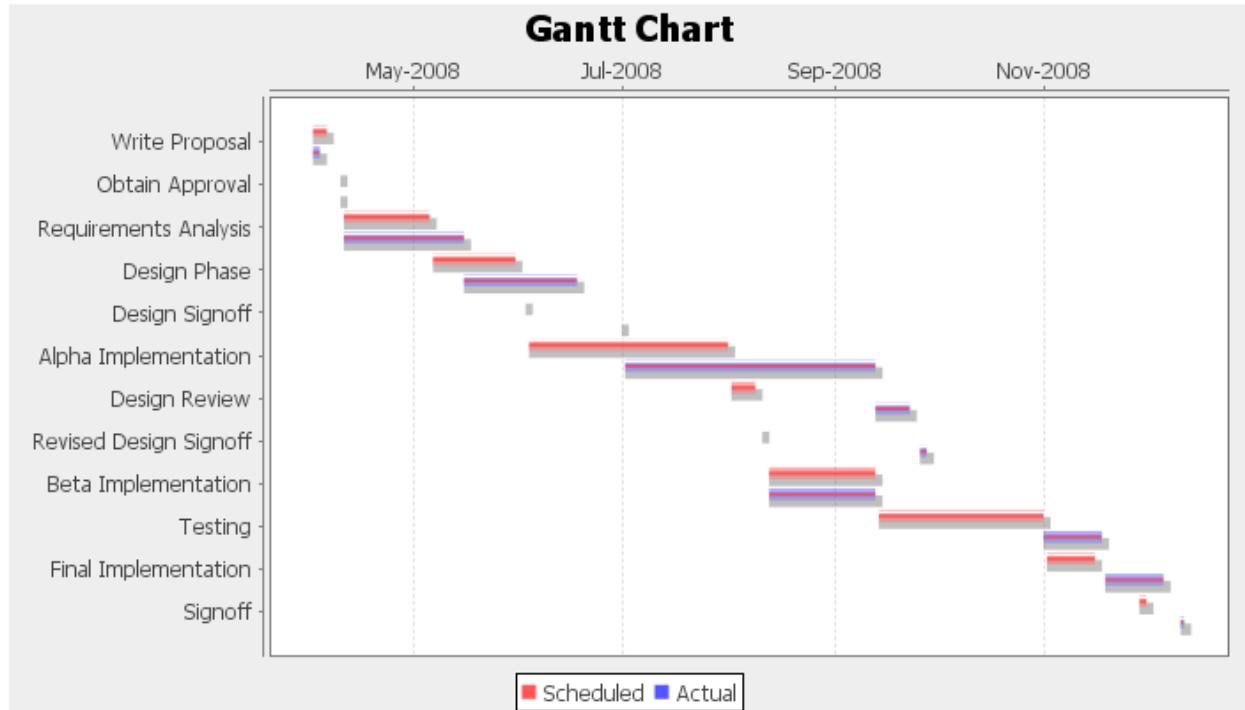
 xyzmodel.addValue("2002", new Integer(30), new
 Integer(120), new Integer(4));
 xyzmodel.addValue("2002", new Integer(50), new
 Integer(135), new Integer(5));
 xyzmodel.addValue("2002", new Integer(70), new
 Integer(140), new Integer(3));
 xyzmodel.addValue("2002", new Integer(90), new
 Integer(160), new Integer(6));
 hbubble.setModel(xyzmodel);
]]></zscript>
</chart>
```

## Wafer Map Chart



```
<chart id="wafermap" title="Wafer Map Chart" width="500"
 height="400" type="wafermap" threeD="false" fgAlpha="128">
<zscript><![CDATA[
 final int xdim = 30;
 final int ydim = 20;
 final Random random = new Random();
 WaferMapModel wafermodel = new WaferMapModel(xdim, ydim);
 for (int x = 0; x < xdim; x++) {
 for (int y = 0; y < ydim; y++) {
 wafermodel.addValue(random.nextInt(5), x, y);
 }
 }
 wafermap.setModel(wafermodel);
]]></zscript>
</chart>
```

## Gantt Chart



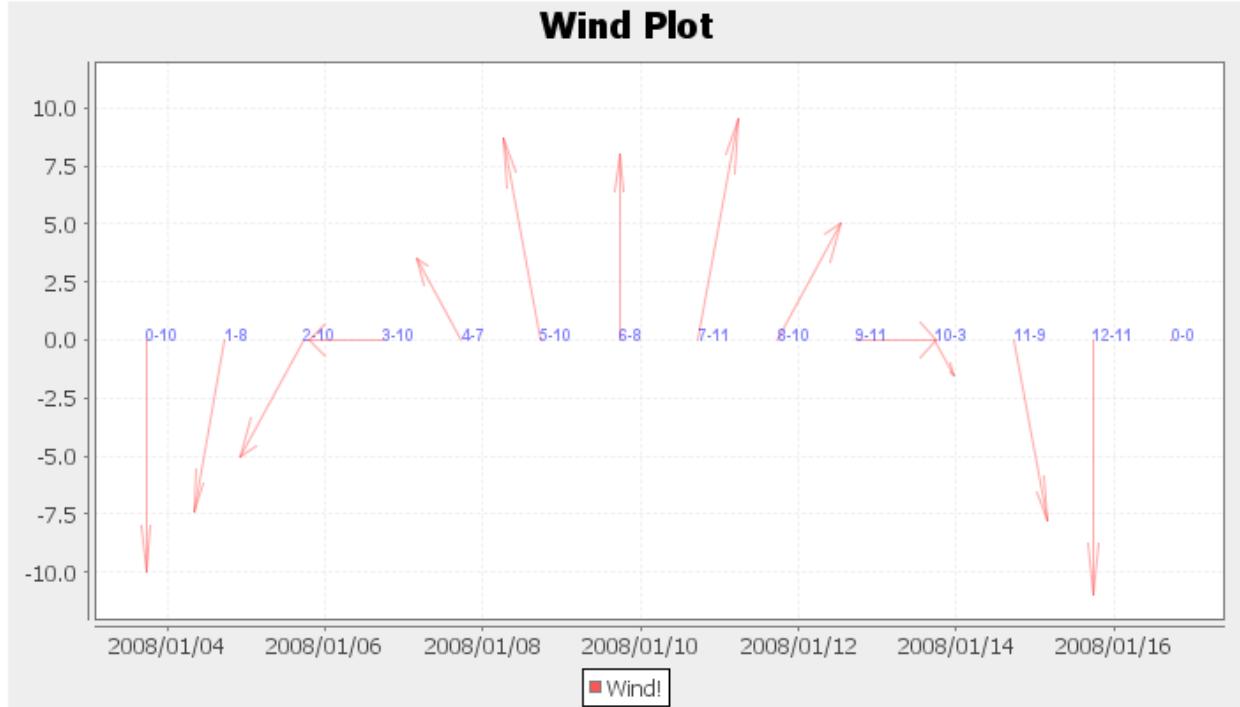
```
<chart id="gantt" title="Gantt Chart" width="700" height="400"
 type="gantt" threeD="false" fgAlpha="128"
 dateFormat="yyyy/MM/dd" >
<zscript><![CDATA[
 import org.zkoss.zul.GanttModel.GanttTask;

 public Date date(int year, int month, int day) {
 final java.util.Calendar calendar =
java.util.Calendar.getInstance();
 calendar.set(year, month-1, day);
 final Date result = calendar.getTime();
 return result;
 }
 //series, task (task description, start, end, complete
percentage)
 GanttModel ganttmmodel = new GanttModel();
 ganttmmodel.addValue("Scheduled", new GanttTask("Write
Proposal", date(2008,4,1), date(2008,4,5), 0.0));
 ganttmmodel.addValue("Scheduled", new GanttTask("Obtain
Approval", date(2008,4,9), date(2008,4,9), 0.0));
 ganttmmodel.addValue("Scheduled", new
GanttTask("Requirements Analysis", date(2008,4,10), date(2008,5,5),
0.0));
 ganttmmodel.addValue("Scheduled", new
GanttTask("Design Phase", date(2008,5,6), date(2008,5,30), 0.0));
 ganttmmodel.addValue("Scheduled", new
GanttTask("Design Signoff", date(2008,6,2), date(2008,6,2), 0.0));
```

```
 ganntmodel.addValue("Scheduled", new GanttTask("Alpha
Implementation", date(2008,6,3), date(2008,7,31), 0.0));
 ganntmodel.addValue("Scheduled", new GanttTask("Design
Review", date(2008,8,1), date(2008,8,8), 0.0));
 ganntmodel.addValue("Scheduled", new GanttTask("Revised
Design Signoff", date(2008,8,10), date(2008,8,10), 0.0));
 ganntmodel.addValue("Scheduled", new GanttTask("Beta
Implementation", date(2008,8,12), date(2008,9,12), 0.0));
 ganntmodel.addValue("Scheduled", new GanttTask("Testing",
date(2008,9,13), date(2008,10,31), 0.0));
 ganntmodel.addValue("Scheduled", new GanttTask("Final
Implementation", date(2008,11,1), date(2008,11,15), 0.0));
 ganntmodel.addValue("Scheduled", new GanttTask("Signoff",
date(2008,11,28), date(2008,11,30), 0.0));

 ganntmodel.addValue("Actual", new GanttTask("Write
Proposal", date(2008,4,1), date(2008,4,3), 0.0));
 ganntmodel.addValue("Actual", new GanttTask("Obtain
Approval", date(2008,4,9), date(2008,4,9), 0.0));
 ganntmodel.addValue("Actual", new GanttTask("Requirements
Analysis", date(2008,4,10), date(2008,5,15), 0.0));
 ganntmodel.addValue("Actual", new GanttTask("Design Phase",
date(2008,5,15), date(2008,6,17), 0.0));
 ganntmodel.addValue("Actual", new GanttTask("Design
Signoff", date(2008,6,30), date(2008,6,30), 0.0));
 ganntmodel.addValue("Actual", new GanttTask("Alpha
Implementation", date(2008,7,1), date(2008,9,12), 0.0));
 ganntmodel.addValue("Actual", new GanttTask("Design
Review", date(2008,9,12), date(2008,9,22), 0.0));
 ganntmodel.addValue("Actual", new GanttTask("Revised Design
Signoff", date(2008,9,25), date(2008,9,27), 0.0));
 ganntmodel.addValue("Actual", new GanttTask("Beta
Implementation", date(2008,8,12), date(2008,9,12), 0.0));
 ganntmodel.addValue("Actual", new GanttTask("Testing",
date(2008,10,31), date(2008,11,17), 0.0));
 ganntmodel.addValue("Actual", new GanttTask("Final
Implementation", date(2008,11,18), date(2008,12,5), 0.0));
 ganntmodel.addValue("Actual", new GanttTask("Signoff",
date(2008,12,10), date(2008,12,11), 0.0));
 gannt.setModel(ganntmodel);
]]></zscript>
</chart>
```

## Wind Chart

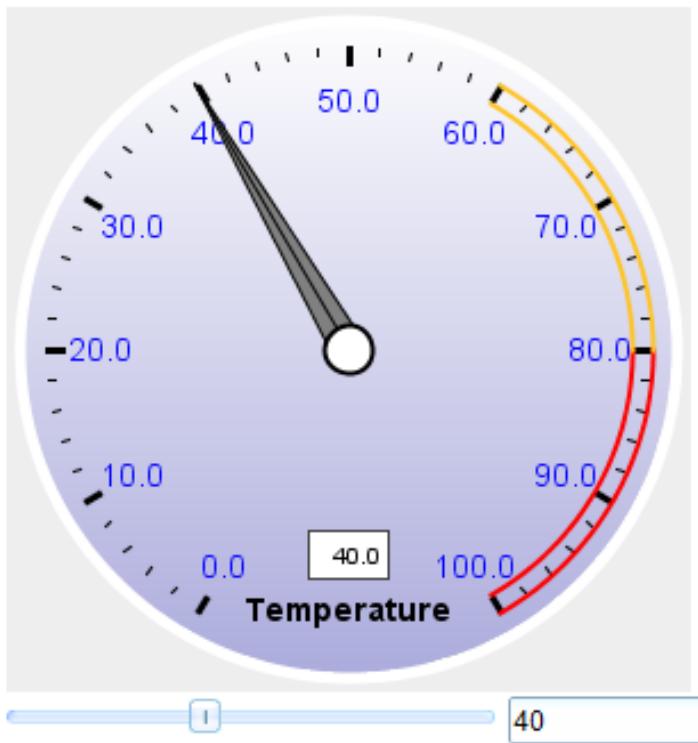


```
<chart id="wind" title="Wind Plot" width="700" height="400">
 type="wind" threeD="false" fgAlpha="128"
 dateFormat="yyyy/MM/dd" >
<zscript><![CDATA[
 public Date date(int year, int month, int day) {
 final java.util.Calendar calendar =
java.util.Calendar.getInstance();
 calendar.set(year, month-1, day);
 final Date result = calendar.getTime();
 return result;
 }
 XYZModel windmodel = new SimpleXYZModel();
 //series, date, direction (0-12), force(0-12)
 windmodel.addValue("Wind!", new Long(date(2008, 1,
3).getTime()), new Double(0d), new Double(10.0));
 windmodel.addValue("Wind!", new Long(date(2008, 1,
4).getTime()), new Double(1d), new Double(8.5));
 windmodel.addValue("Wind!", new Long(date(2008, 1,
5).getTime()), new Double(2.0), new Double(10.0));
 windmodel.addValue("Wind!", new Long(date(2008, 1,
6).getTime()), new Double(3.0), new Double(10.0));
 windmodel.addValue("Wind!", new Long(date(2008, 1,
7).getTime()), new Double(4.0), new Double(7.0));
 windmodel.addValue("Wind!", new Long(date(2008, 1,
8).getTime()), new Double(5.0), new Double(10.0));
 windmodel.addValue("Wind!", new Long(date(2008, 1,
```

```

9).getTime(), new Double(6.0), new Double(8.0));
 windmodel.addValue("Wind!", new Long(date(2008, 1,
10).getTime()), new Double(7.0), new Double(11.0));
 windmodel.addValue("Wind!", new Long(date(2008, 1,
11).getTime()), new Double(8.0), new Double(10.0));
 windmodel.addValue("Wind!", new Long(date(2008, 1,
12).getTime()), new Double(9.0), new Double(11.0));
 windmodel.addValue("Wind!", new Long(date(2008, 1,
13).getTime()), new Double(10.0), new Double(3.0));
 windmodel.addValue("Wind!", new Long(date(2008, 1,
14).getTime()), new Double(11.0), new Double(9.0));
 windmodel.addValue("Wind!", new Long(date(2008, 1,
15).getTime()), new Double(12.0), new Double(11.0));
 windmodel.addValue("Wind!", new Long(date(2008, 1,
16).getTime()), new Double(0.0), new Double(0.0));
 wind.setModel(windmodel);
]]></zscript>
</chart>
```

## Dial Chart



```

<window onOK="doOK()" width="350px">
 <chart id="dial" title="Dial Plot" width="300" height="300" type="dial" threeD="false" fgAlpha="128">
 <zscript><![CDATA[
 import org.zkoss.zul.DialModel;
 import org.zkoss.zul.DialModelScale;

 int val= 40;
```

```
DialModel dialmodel = new DialModel();

DialModelScale scale = dialmodel.newScale(0.0, 100.0,
-120.0, -300.0, 10.0, 4); //scale's configuration data

scale.setText("Temperature");

scale.newRange(80, 100, "#FF0000", 0.83, 0.89);

scale.newRange(60, 80, "#FFC426", 0.83, 0.89);

scale.setValue(val);

doOK() {

 val = dbx.getValue();

 if (val > 100) {

 val = 100;

 } else if (val < 0) {

 val = 0;

 }

 dbx.value = val;

 slx.curpos = val;

 scale.setValue(val);

 if (val > 80) {

 scale.setNeedleColor(scale.getRange(0).getRangeColor());

 } else if (val > 60) {

 scale.setNeedleColor(scale.getRange(1).getRangeColor());

 } else {

 scale.setNeedleColor(dialmodel.getFrameFgColor());

 }

}

dial.setModel(dialmodel);

]]></zscript>

</chart>

<hbox>

<slider id="slx" curpos="${val}" onScroll="dbx.value=self.curpos; doOK()"/>

<intbox id="dbx" value="${val}" onChange="doOK()"/>

</hbox>

</window>
```

## Properties

### Type and Model

Type	Model	3D
pie	PieModel	o
ring	PieModel	x
bar	CategoryModel or XYModel	o
line	CategoryModel or XYModel	o
area	CategoryModel or XYModel	x
stacked_bar	CategoryModel	o
stacked_area	CategoryModel or XYModel	x
waterfall	CategoryModel	x
polar	XYModel	x
scatter	XYModel	x
time_series	XYModel	x
polar	XYModel	x
step_area	XYModel	x
step	XYModel	x
histogram	XYModel	x
bubble	XYModel	x
wind	XYModel	x
candlestick	HiLoModel	x
highlow	HiLoModel	x
wafermap	WaferMapModel	x
gantt	GanttModel	x
dial	DialModel	x

## Clicked Area: Series, Legend

In an onClick listener, you can call MouseEvent.getX() [4] and MouseEvent.getY() [5] to get coordinates.

Call MouseEvent.getAreaComponent() [6] to retrieve the area component (Area [7]) which users click on.

There are several attributes you can get from the clicked Area:

- series: series name
- category: category name, e.g. GanttTask's description
- entity: could be DATA or LEGEND
- percent: percentage of complete of a gantt task
- start: start value of a gannt task
- end: end value of a gannt task

```

@Listen("onClick = #gantt")
public void drilldown(MouseEvent event) {
 final Area area = (Area)event.getAreaComponent();
 if (area != null)

```

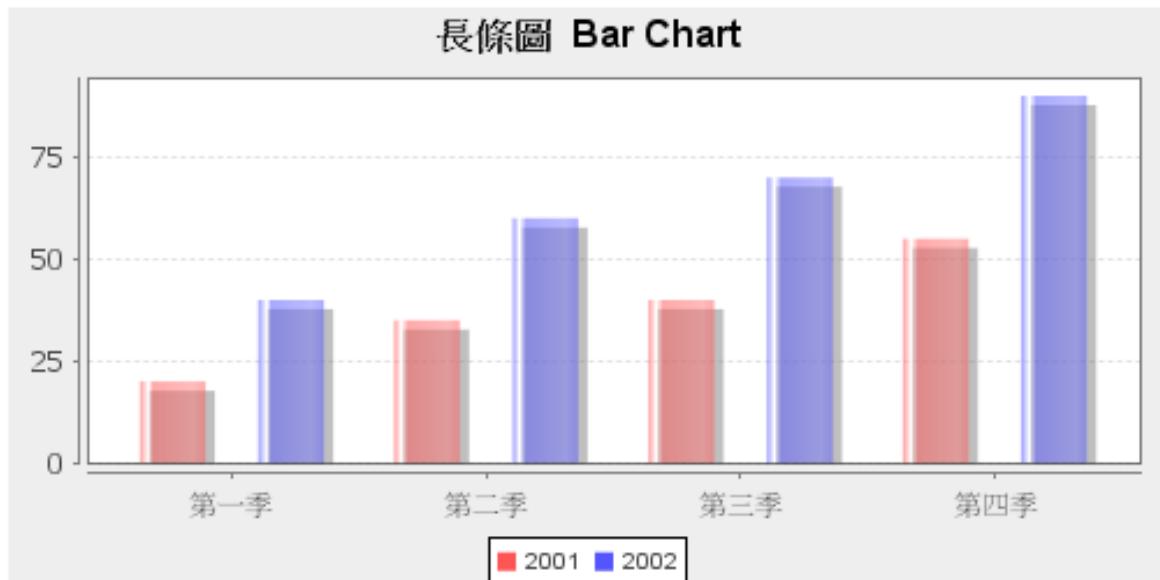
```

 Messagebox.show(area.getAttribute("category") + :
"+area.getTooltiptext());
}

```

## Charts Support Setting the Font

We can now specify what font we want a chart to use. This is useful when employing a foreign language within your chart. The example demonstrates setting a font and using it with a foreign language, in this case Chinese.



```

<zscript>
import java.awt.Font;
String fontname = "Dialog";
Font tfont = new Font(fontname, Font.BOLD, 16); //for title
Font lfont = new Font(fontname, Font.PLAIN, 10); //for
legend
Font lbfont = new Font(fontname, Font.PLAIN, 12); //for
label

CategoryModel catmodel = new SimpleCategoryModel();
catmodel.setValue("2001", "第一季", new Integer(20));
catmodel.setValue("2001", "第二季", new Integer(35));
catmodel.setValue("2001", "第三季", new Integer(40));
catmodel.setValue("2001", "第四季", new Integer(55));
catmodel.setValue("2002", "第一季", new Integer(40));
catmodel.setValue("2002", "第二季", new Integer(60));
catmodel.setValue("2002", "第三季", new Integer(70));
catmodel.setValue("2002", "第四季", new Integer(90));

</zscript>

<chart id="barchart" title="長條圖 Bar Chart" width="500" height="250" type="bar" threeD="false" fgAlpha="128"
titleFont="${tfont}" legendFont="${lfont}" xAxisFont="${lbfont}"
xAxisTickFont="${lbfont}" model="${catmodel}"/>

```

## Supported Events

Name	Inherited From
None	None

- Inherited Supported Events: Imagemap

## Supported Children

\*NONE

## Use Cases

Version	Description	Example Location
5.0	Make a Chart fill 100% width in parent panel	[8]
5.0	Dual axis in Chart	[9]

## Troubleshooting

### Linux

Chart depends on Java Swing that might not work under some version of JVM. For the information to make it work under Linux, please refer to ZK Installation Guide: Linux.

## Version History

Version	Date	Content
5.0.4	August 2010	<p>MouseEvent introduced a new method, MouseEvent.getAreaComponent()<sup>[6]</sup>, which simplifies the retrieval of the area component.</p> <pre>Area area = (Area) event.getAreaComponent(); //must be Area or null when used with chart if (area != null)     ... </pre>
5.0.3	June 2010	<p>The area sent with the click event becomes UUID of the area component. Thus, use desktop.getComponentByUuid(event.getArea()). To write a program compatible with any version of ZK:</p> <pre>String areaid = event.getArea(); if (areaid != null) {     Area area = desktop.getComponentByUuidIfAny(areaid);     if (area == null)         area = chart.getFellow(areaid); //fall back to older version     ... }</pre>

## References

- [1] [http://www.zkoss.org/zkdemo/chart/pie\\_chart](http://www.zkoss.org/zkdemo/chart/pie_chart)
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Chart.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Chart.html#>
- [4] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/MouseEvent.html#getX\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/MouseEvent.html#getX())
- [5] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/MouseEvent.html#getY\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/MouseEvent.html#getY())
- [6] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/MouseEvent.html#getAreaComponent\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/MouseEvent.html#getAreaComponent())
- [7] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Area.html#>
- [8] <http://www.zkoss.org/forum/listComment/10761>
- [9] <http://www.zkoss.org/forum/listComment/8752>

# Flashchart

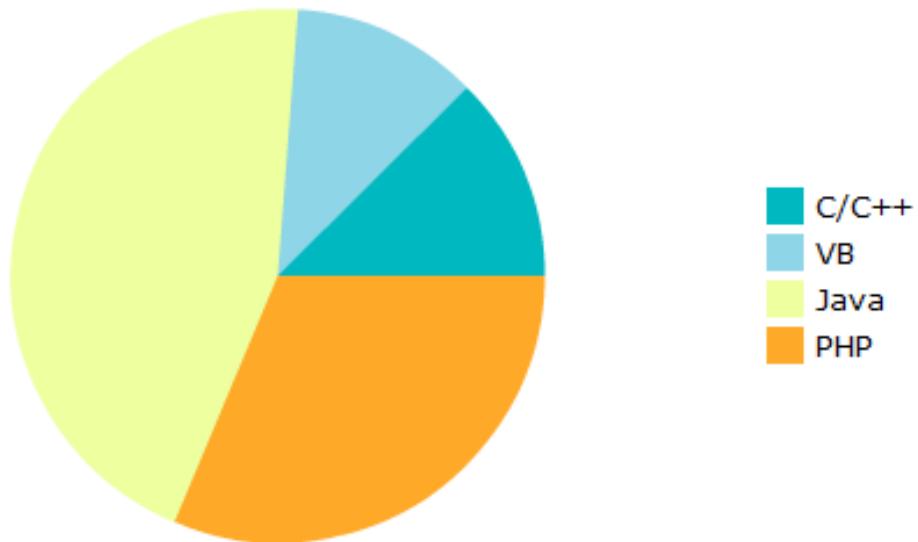
## Flashchart

- Demonstration: Flash Pie Chart <sup>[1]</sup>, Other Flash Charts <sup>[2]</sup>
- Java API: Flashchart <sup>[3]</sup>
- JavaScript API: Flashchart <sup>[4]</sup>

## Employment/Purpose

The generic flash chart component.

## Example



```
<flashchart id="mychart" width="500" height="250" type="pie">
<zscript><![CDATA[
 PieModel model = new SimplePieModel();
 model.setValue("C/C++", new Double(56));
 model.setValue("VB", new Double(51));
 model.setValue("Java", new Double(202));
 model.setValue("PHP", new Double(141));
 mychart.setModel(model);
]]>
```

```
]]></zscript>
</flashchart>
```

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: Flash

## Supported Children

\*ALL

## Use Cases

Version	Description	Example Location

## Version History

Version	Date	Content
5.0.12, 6.0.3.1, and 6.5.1		Decouple Flashchart component from the zul to the addon package. Please download it from the Github [5].

## References

- [1] [http://www.zkoss.org/zkdemo/chart/flash\\_pie\\_chart](http://www.zkoss.org/zkdemo/chart/flash_pie_chart)
- [2] [http://www.zkoss.org/zkdemo/chart/other\\_flash\\_charts](http://www.zkoss.org/zkdemo/chart/other_flash_charts)
- [3] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Flashchart.html#>
- [4] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/fchart/Flashchart.html#>
- [5] <https://github.com/zkoss/flashchart>

# Fusionchart

---

## Fusionchart

- Demonstration:
- Java API: Fusionchart <sup>[1]</sup>
- JavaScript API: Fusionchart <sup>[2]</sup>
- Style Guide: N/A
- Available for ZK:
  - CE
  - PE
  - EE
- ZK Fusionchart has now entered maintenance mode as of Q2, 2014 as the 3rd party library that it's based on - Fusioncharts free, has moved out of flash and stopped supporting it.

## Employment/Purpose

The Fusionchart which integrates the FusionCharts Free <sup>[3]</sup> with ZK. The technology makes use of Flash to draw charts and enables the user to customize the style of charts such as bar or line colors.

Fusionchart separates the presentation layer from the data, providing the users with a API to supply data in a clean MVC based manner. In addition to updating the data dynamically, it provides a pleasant user experience as the visual display is updated immediately.

## Common Attributes

Name	Description
title	Title of chart
threeD	Set true to show three dimensional graph(Note: Some types may not support 3D)
type	Type of chart
fgAlpha	Foreground alpha
model	Model of chart

## Supported Model

Type	Model	3D
bar	CategoryModel	*
line	CategoryModel, XYModel	X
pie	PieModel, SingleValueCategoryModel [Since 6.5.3]	O
funnel	SingleValueCategoryModel [Since 6.5.3]	X
combination	CategoryModel	O
stacked_bar	CategoryModel	O
area	CategoryModel	X
gantt	GanttModel	X

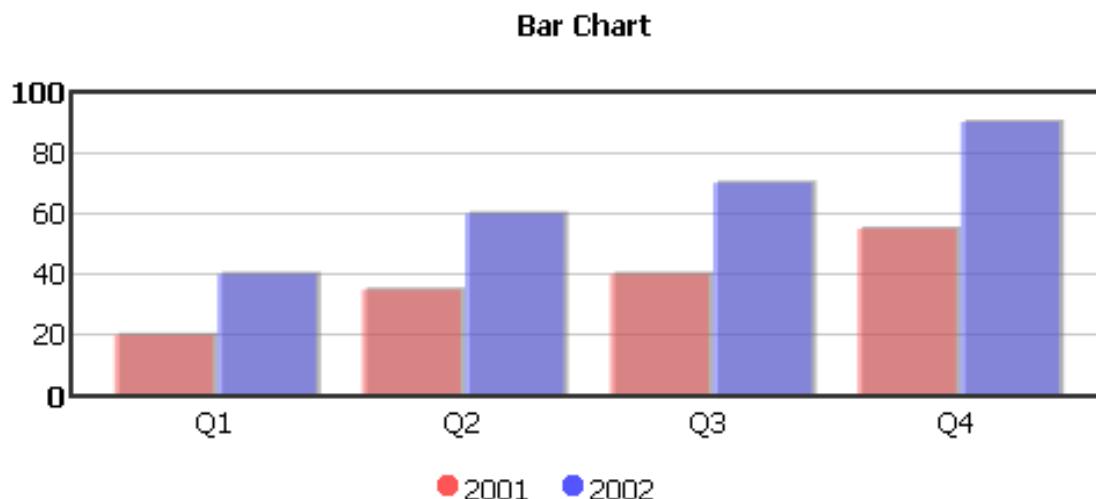
**Note:** 3D bar chart of Fusionchart does not support horizontal orientation now. If you want to use horizontal 3D bar chart, please use Chart component.

## Example

### Bar Chart

#### 2D Bar Chart

##### Vertical Bar Chart



```
<zk>
<div width="100%" apply="demo.BarchartComposer">
 <fusionchart id="mychart" title="Bar Chart" type="bar" width="500" height="250" />
</div>
</zk>
```

```
public class BarchartComposer extends SelectorComposer<Div> {

 @Wire
 private Fusionchart mychart;
```

```

public void doAfterCompose(Div comp) throws Exception {
 super.doAfterCompose(comp);

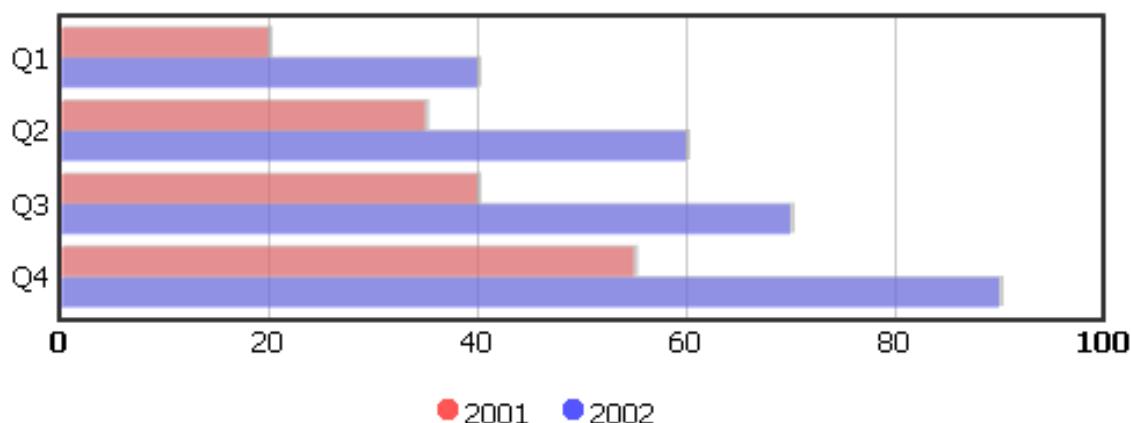
 CategoryModel catmodel = new SimpleCategoryModel();
 catmodel.setValue("2001", "Q1", new Integer(20));
 catmodel.setValue("2001", "Q2", new Integer(35));
 catmodel.setValue("2001", "Q3", new Integer(40));
 catmodel.setValue("2001", "Q4", new Integer(55));
 catmodel.setValue("2002", "Q1", new Integer(40));
 catmodel.setValue("2002", "Q2", new Integer(60));
 catmodel.setValue("2002", "Q3", new Integer(70));
 catmodel.setValue("2002", "Q4", new Integer(90));

 mychart.setModel(catmodel);
}
}

```

## Horizontal Bar Chart

**Horizontal Bar Chart**



```

<zk>
<div width="100%" apply="demo.BarchartComposer">
 <fusionchart id="mychart" title="Horizontal Bar Chart" type="bar" orient="horizontal" width="500" height="250" />
</div>
</zk>

```

```

public class BarchartComposer extends SelectorComposer<Div> {

 @Wire
 private Fusionchart mychart;

 public void doAfterCompose(Div comp) throws Exception {
 super.doAfterCompose(comp);
 }
}

```

```

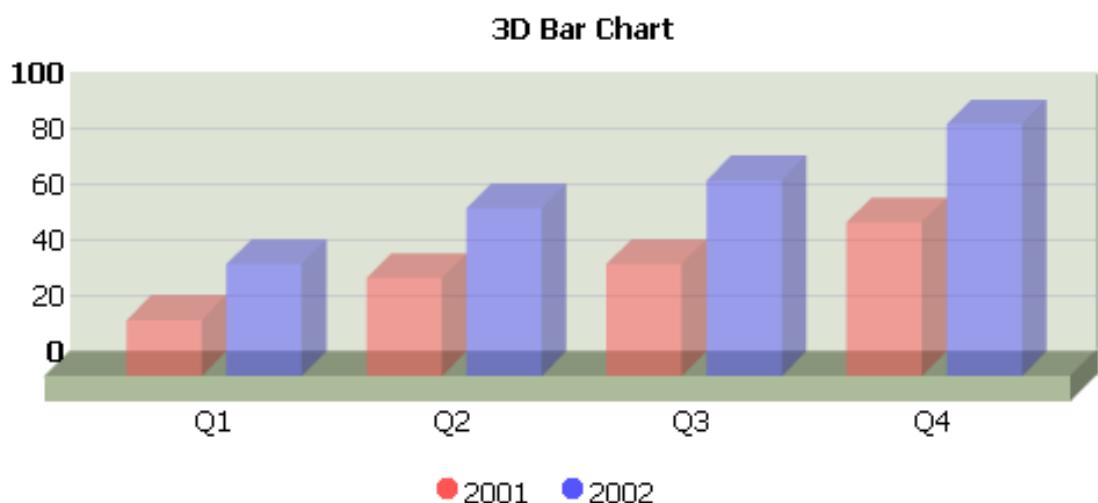
CategoryModel catmodel = new SimpleCategoryModel();
catmodel.setValue("2001", "Q1", new Integer(20));
catmodel.setValue("2001", "Q2", new Integer(35));
catmodel.setValue("2001", "Q3", new Integer(40));
catmodel.setValue("2001", "Q4", new Integer(55));
catmodel.setValue("2002", "Q1", new Integer(40));
catmodel.setValue("2002", "Q2", new Integer(60));
catmodel.setValue("2002", "Q3", new Integer(70));
catmodel.setValue("2002", "Q4", new Integer(90));

mychart.setModel(catmodel);
}

}

```

### 3D Vertical Bar Chart



```

<zk>
 <div width="100%" apply="demo.BarchartComposer">
 <fusionchart id="mychart" title="3D Bar Chart" type="bar" threeD="true" width="500" height="250" />
 </div>
</zk>

```

```

public class BarchartComposer extends SelectorComposer<Div> {

 @Wire
 private Fusionchart mychart;

 public void doAfterCompose(Div comp) throws Exception {
 super.doAfterCompose(comp);

 CategoryModel catmodel = new SimpleCategoryModel();
 catmodel.setValue("2001", "Q1", new Integer(20));
 catmodel.setValue("2001", "Q2", new Integer(35));
 }
}

```

```

 catmodel.setValue("2001", "Q3", new Integer(40));
 catmodel.setValue("2001", "Q4", new Integer(55));
 catmodel.setValue("2002", "Q1", new Integer(40));
 catmodel.setValue("2002", "Q2", new Integer(60));
 catmodel.setValue("2002", "Q3", new Integer(70));
 catmodel.setValue("2002", "Q4", new Integer(90));

 mychart.setModel(catmodel);
 }
}

```

## Bar Chart Configurations

You could set the color of all categories in the CategoryChartConfig, here's a sample code for setting first one and second one to different color .

```

CategoryChartConfig config = new CategoryChartConfig();
config.setAnimation(false); //You could turn off the animation
,which default is enabled.

SeriesPropertiesMap seriesPropertiesMap =
config.getSeriesPropertiesMap();
// Use the index to config the chart
// The color value is required to be HEX code without # due to
the restriction of Fuscionchart core.

seriesPropertiesMap.createSeriesProperties(0).addProperty("color", "AFD8F8");

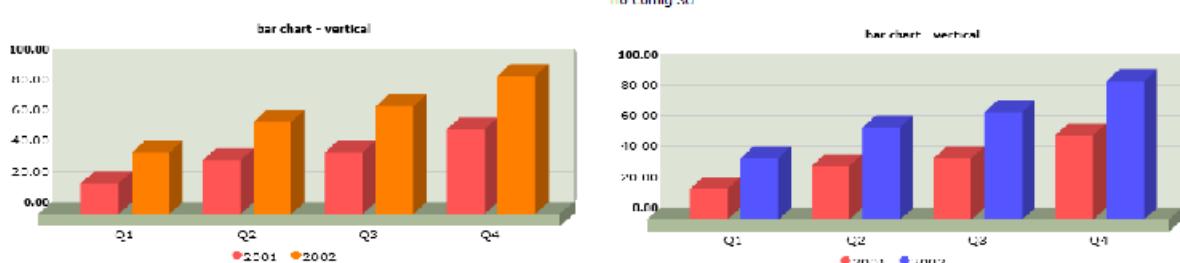
seriesPropertiesMap.createSeriesProperties(1).addProperty("color", "FF8000");

// ===== Or another way , use category name instead of index ===== //
//
seriesPropertiesMap.createSeriesProperties("2001").addProperty("color", "AFD8F8");
//
seriesPropertiesMap.createSeriesProperties("2002").addProperty("color", "FF8000");

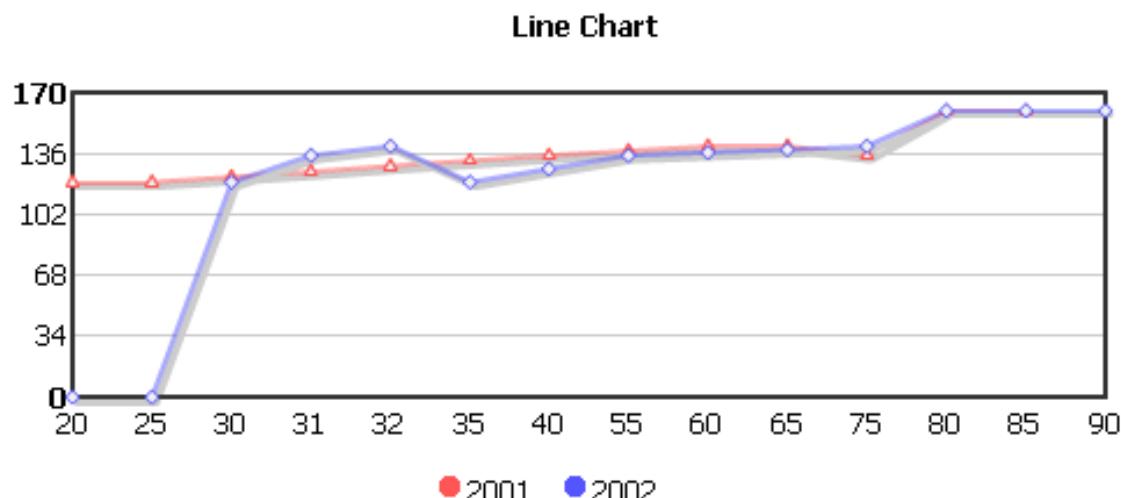
fusionchart.setChartConfig(config);

```

The color is changed after we set the config ,here's the demo image.



## Line Chart



```
<zk>
 <div width="100%" apply="demo.LinechartComposer">
 <fusionchart id="mychart" title="Line Chart" type="line" width="500" height="250" />
 </div>
</zk>

public class LinechartComposer extends SelectorComposer<Div> {

 @Wire
 private Fusionchart mychart;

 public void doAfterCompose(Div comp) throws Exception {
 super.doAfterCompose(comp);

 XYModel xymodel = new SimpleXYModel();
 xymodel.addValue("2001", new Integer(20), new Integer(120));
 xymodel.addValue("2001", new Integer(25), new Integer(120));
 xymodel.addValue("2001", new Integer(30), new Integer(125));
 xymodel.addValue("2001", new Integer(31), new Integer(130));
 xymodel.addValue("2001", new Integer(32), new Integer(132));
 xymodel.addValue("2001", new Integer(35), new Integer(133));
 xymodel.addValue("2001", new Integer(40), new Integer(135));
 xymodel.addValue("2001", new Integer(55), new Integer(136));
 xymodel.addValue("2001", new Integer(60), new Integer(137));
 xymodel.addValue("2001", new Integer(65), new Integer(138));
 xymodel.addValue("2001", new Integer(75), new Integer(135));
 xymodel.addValue("2001", new Integer(80), new Integer(160));
 xymodel.addValue("2001", new Integer(85), new Integer(160));

 xymodel.addValue("2002", new Integer(20), new Integer(0));
 xymodel.addValue("2002", new Integer(25), new Integer(0));
 }
}
```

```

 xymodel.addValue("2002", new Integer(30), new Integer(120));
 xymodel.addValue("2002", new Integer(31), new Integer(135));
 xymodel.addValue("2002", new Integer(32), new Integer(140));
 xymodel.addValue("2002", new Integer(35), new Integer(120));
 xymodel.addValue("2002", new Integer(40), new Integer(135));
 xymodel.addValue("2002", new Integer(55), new Integer(135));
 xymodel.addValue("2002", new Integer(60), new Integer(140));
 xymodel.addValue("2002", new Integer(65), new Integer(140));
 xymodel.addValue("2002", new Integer(75), new Integer(140));
 xymodel.addValue("2002", new Integer(80), new Integer(160));
 xymodel.addValue("2002", new Integer(85), new Integer(160));
 xymodel.addValue("2002", new Integer(90), new Integer(160));

 mychart.setModel(xymodel);
 }
}

```

## Line Chart Configurations

```

CategoryChartConfig config = new CategoryChartConfig();
config.setAnimation(false); //You could turn off the
animation ,which default is enabled.
config.getChartProperties().addProperty("rotateNames",
"1");

SeriesPropertiesMap seriesPropertiesMap =
config.getSeriesPropertiesMap();

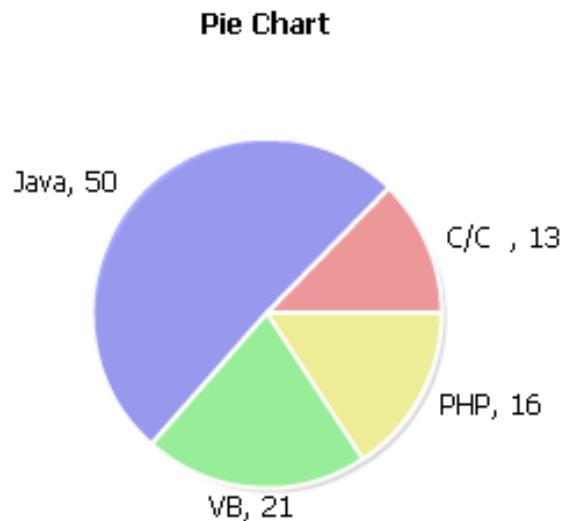
//Create Series Properties by index , you could use
category name , too
//ex. seriesPropertiesMap.createSeriesProperties("2001")

seriesPropertiesMap.createSeriesProperties(0)
 .addProperty("color", "AFD8F8"); //change
color
seriesPropertiesMap.createSeriesProperties(1)
 .addProperty("anchorBorderColor", "FF8000")
//change series color
 .addProperty("lineThickness", "5"); //Change
line weight
fusionchart.setChartConfig(config);

```

## Pie Chart

### 2D Pie Chart



```
<zk>
 <div width="100%" apply="demo.PiechartComposer">
 <fusionchart id="mychart" title="2D Pie Chart" type="pie" width="500" height="250" />
 </div>
</zk>

public class PiechartComposer extends SelectorComposer<Div> {

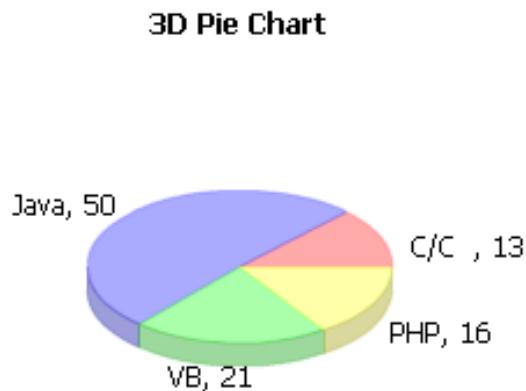
 @Wire
 private Fusionchart mychart;

 @Override
 public void doAfterCompose(Div comp) throws Exception {
 super.doAfterCompose(comp);

 PieModel piemodel = new SimplePieModel();
 piemodel.setValue("C/C++", new Double(12.5));
 piemodel.setValue("Java", new Double(50.2));
 piemodel.setValue("VB", new Double(20.5));
 piemodel.setValue("PHP", new Double(15.5));

 mychart.setModel(piemodel);
 }
}
```

### 3D Pie Chart



```
<zk>
 <div width="100%" apply="demo.PiechartComposer">
 <fusionchart id="mychart" title="3D Pie Chart" type="pie" threeD="true" width="500" height="250" />
 </div>
</zk>

public class PiechartComposer extends SelectorComposer<Div> {

 @Wire
 private Fusionchart mychart;

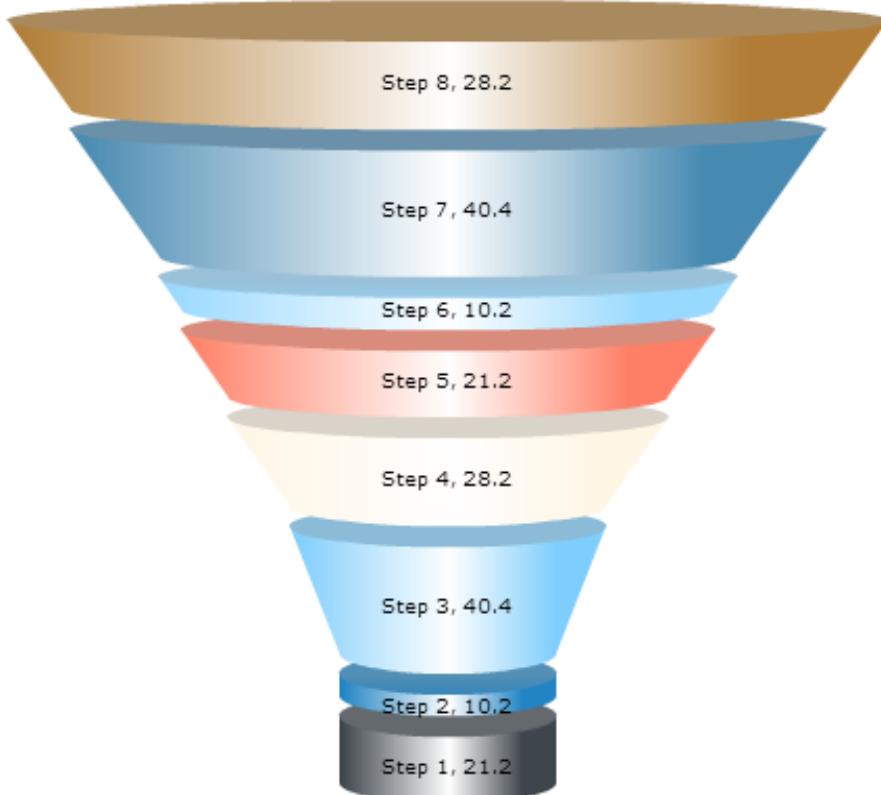
 @Override
 public void doAfterCompose(Div comp) throws Exception {
 super.doAfterCompose(comp);

 PieModel piemodel = new SimplePieModel();
 piemodel.setValue("C/C++", new Double(12.5));
 piemodel.setValue("Java", new Double(50.2));
 piemodel.setValue("VB", new Double(20.5));
 piemodel.setValue("PHP", new Double(15.5));

 mychart.setModel(piemodel);
 }
}
```

## Funnel Chart

[Since 6.5.3]



```
<zK>
 <div width="100%" apply="demo.FunnelchartComposer ">
 <fusionchart id="mychart" width="550" height="400" type="funnel" />
 </div>
</zK>
```

```
public class FunnelchartComposer extends SelectorComposer<Div> {

 @Wire
 private Fusionchart mychart;

 @Override
 public void doAfterCompose(Div comp) throws Exception {
 super.doAfterCompose(comp);

 SingleValueCategoryModel model = new
SimpleSingleValueCategoryModel();
 model.setValue("Step 1", new Double(21.2));
 model.setValue("Step 2", new Double(10.2));
 model.setValue("Step 3", new Double(40.4));
 model.setValue("Step 4", new Double(28.2));
 model.setValue("Step 5", new Double(21.2));
 model.setValue("Step 6", new Double(10.2));
```

```

 model.setValue("Step 7", new Double(40.4));
 model.setValue("Step 8", new Double(28.2));

 mychart.setModel(model);
 }
}

```

## Combination Chart

### 2D Combination Chart

**2D Combination Charts**



```

<zkc>

<fusionchart id="mychart" title="2D Combination Charts" type="combination" width="500" height="250" />

</zkc>

```

```

public class CombinationchartComposer extends SelectorComposer<Div> {

 @Wire
 private Fusionchart mychart;

 @Override
 public void doAfterCompose(Div comp) throws Exception {
 super.doAfterCompose(comp);

 CategoryModel catmodel = new SimpleCategoryModel();
 catmodel.setValue("Product A", "08/01", new Integer(20));
 catmodel.setValue("Product A", "08/02", new Integer(35));
 catmodel.setValue("Product A", "08/03", new Integer(40));
 catmodel.setValue("Product A", "08/04", new Integer(55));
 catmodel.setValue("Product B", "08/01", new Integer(40));
 catmodel.setValue("Product B", "08/02", new Integer(60));
 catmodel.setValue("Product B", "08/03", new Integer(70));
 }
}

```

```

 catmodel.setValue("Product B", "08/04", new Integer(90));
 catmodel.setValue("Product C", "08/01", new Integer(90));
 catmodel.setValue("Product C", "08/02", new Integer(30));
 catmodel.setValue("Product C", "08/03", new Integer(60));
 catmodel.setValue("Product C", "08/04", new Integer(10));

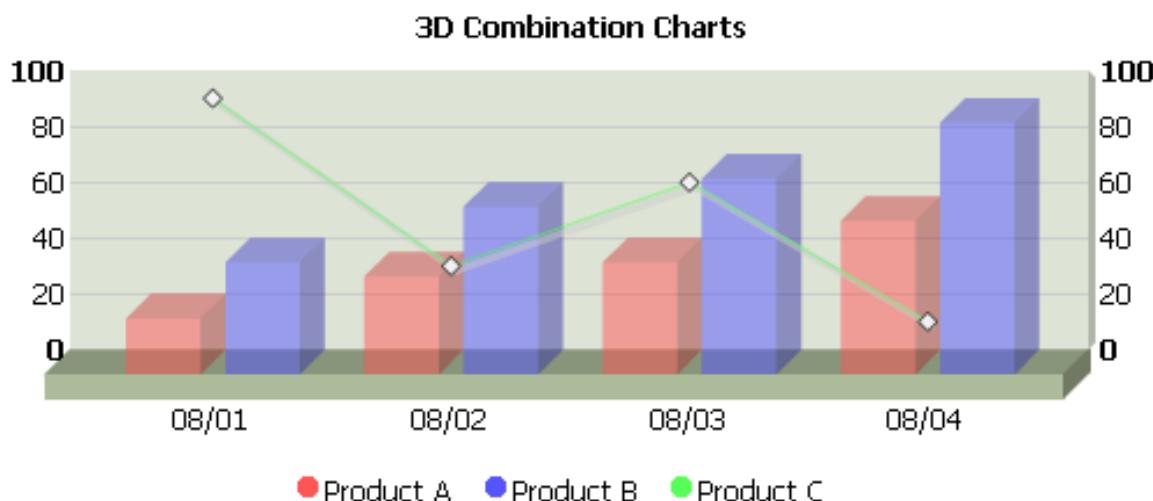
 CategoryChartConfig config = new CategoryChartConfig();
 SeriesPropertiesMap sPropertiesMap =
 config.getSeriesPropertiesMap();

 sPropertiesMap.createSeriesProperties("Product
A").addProperty("parentYAxis", "P");
 sPropertiesMap.createSeriesProperties("Product
B").addProperty("parentYAxis", "S");
 sPropertiesMap.createSeriesProperties("Product
C").addProperty("parentYAxis", "S");

 mychart.setChartConfig(config);
 mychart.setModel(catmodel);
 }
}

```

### 3D Combination Chart



```

<zkc>
<div width="100%" apply="demo.ComboChartComposer">
 <fusionchart id="mychart" title="3D Combination Charts" type="combination" threeD="true" width="500" height="250" />
</div>
</zkc>

```

```

public class CombinationchartComposer extends SelectorComposer<Div> {

 @Wire
 private Fusionchart mychart;
}

```

```
 @Override
 public void doAfterCompose(Div comp) throws Exception {
 super.doAfterCompose(comp);

 CategoryModel catmodel = new SimpleCategoryModel();
 catmodel.setValue("Product A", "08/01", new Integer(20));
 catmodel.setValue("Product A", "08/02", new Integer(35));
 catmodel.setValue("Product A", "08/03", new Integer(40));
 catmodel.setValue("Product A", "08/04", new Integer(55));
 catmodel.setValue("Product B", "08/01", new Integer(40));
 catmodel.setValue("Product B", "08/02", new Integer(60));
 catmodel.setValue("Product B", "08/03", new Integer(70));
 catmodel.setValue("Product B", "08/04", new Integer(90));
 catmodel.setValue("Product C", "08/01", new Integer(90));
 catmodel.setValue("Product C", "08/02", new Integer(30));
 catmodel.setValue("Product C", "08/03", new Integer(60));
 catmodel.setValue("Product C", "08/04", new Integer(10));

 CategoryChartConfig config = new CategoryChartConfig();
 SeriesPropertiesMap sPropertiesMap =
 config.getSeriesProperties();

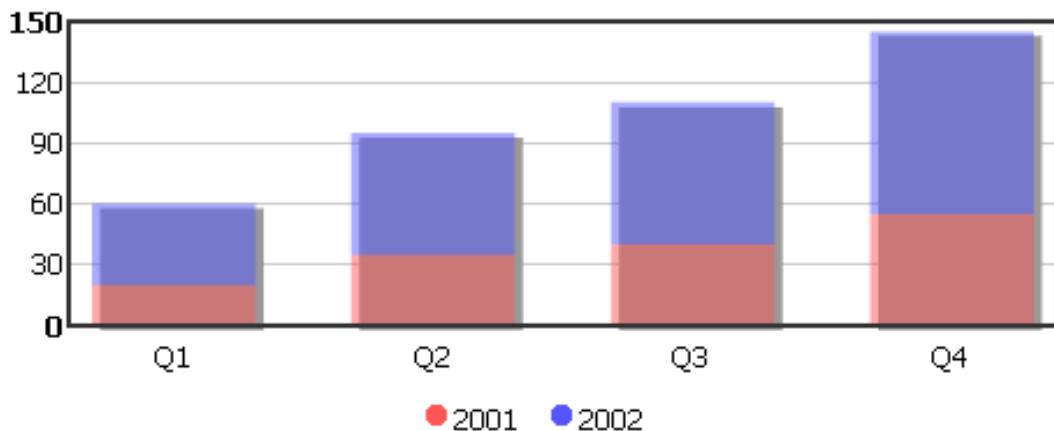
 sPropertiesMap.createSeriesProperties("Product
A").addProperty("parentYAxis", "P");
 sPropertiesMap.createSeriesProperties("Product
B").addProperty("parentYAxis", "S");
 sPropertiesMap.createSeriesProperties("Product
C").addProperty("parentYAxis", "S");

 mychart.setChartConfig(config);
 mychart.setModel(catmodel);
 }
}
```

## Stacked Chart

### 2D Stacked Chart

2D Stacked Bar Charts



```
<zk>
 <div width="100%" apply="demo.StackedchartComposer">
 <fusionchart id="mychart" title="2D Stacked Charts" type="stacked_bar" width="500" height="250" />
 </div>
</zk>

public class StackedchartComposer extends SelectorComposer<Div> {

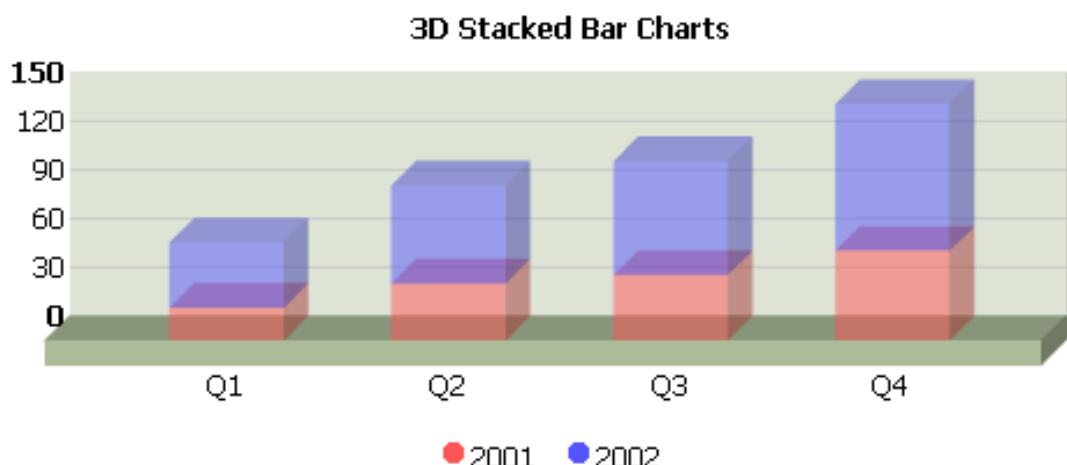
 @Wire
 private Fusionchart mychart;

 @Override
 public void doAfterCompose(Div comp) throws Exception {
 super.doAfterCompose(comp);

 CategoryModel catmodel = new SimpleCategoryModel();
 catmodel.setValue("2001", "Q1", new Integer(20));
 catmodel.setValue("2001", "Q2", new Integer(35));
 catmodel.setValue("2001", "Q3", new Integer(40));
 catmodel.setValue("2001", "Q4", new Integer(55));
 catmodel.setValue("2002", "Q1", new Integer(40));
 catmodel.setValue("2002", "Q2", new Integer(60));
 catmodel.setValue("2002", "Q3", new Integer(70));
 catmodel.setValue("2002", "Q4", new Integer(90));

 mychart.setModel(catmodel);
 }
}
```

### 3D Stacked Chart



```

<zk>
 <div width="100%" apply="demo.StackedchartComposer">
 <fusionchart id="mychart" title="3D Stacked Charts" type="stacked_bar" threeD="true" width="500" height="250" />
 </div>
</zk>

public class StackedchartComposer extends SelectorComposer<Div> {

 @Wire
 private Fusionchart mychart;

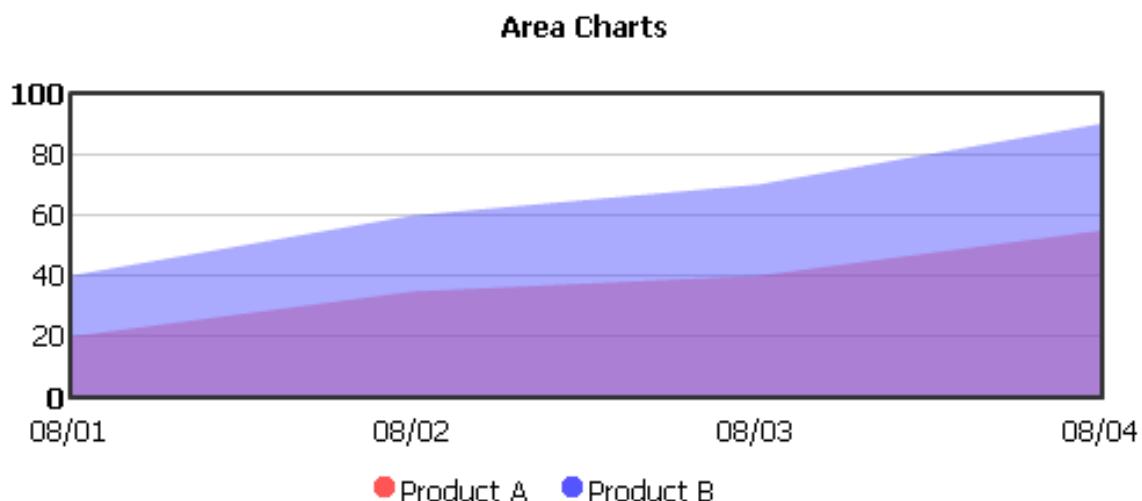
 @Override
 public void doAfterCompose(Div comp) throws Exception {
 super.doAfterCompose(comp);

 CategoryModel catmodel = new SimpleCategoryModel();
 catmodel.setValue("2001", "Q1", new Integer(20));
 catmodel.setValue("2001", "Q2", new Integer(35));
 catmodel.setValue("2001", "Q3", new Integer(40));
 catmodel.setValue("2001", "Q4", new Integer(55));
 catmodel.setValue("2002", "Q1", new Integer(40));
 catmodel.setValue("2002", "Q2", new Integer(60));
 catmodel.setValue("2002", "Q3", new Integer(70));
 catmodel.setValue("2002", "Q4", new Integer(90));

 mychart.setModel(catmodel);
 }
}

```

## Area Chart



```
<zk>
 <window border="none" width="100%" apply="demo.AreachartComposer">
 <fusionchart id="mychart" title="Area Charts" type="area" width="500" height="250" />
 </window>
</zk>

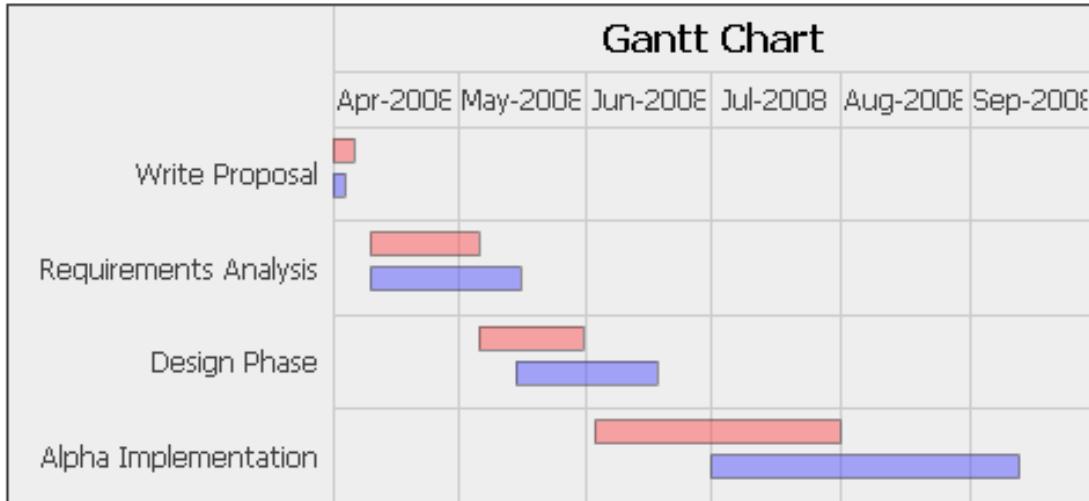
public class AreachartComposer extends SelectorComposer<Div> {
 @Wire
 private Fusionchart mychart;

 @Override
 public void doAfterCompose(Div comp) throws Exception {
 super.doAfterCompose(comp);

 CategoryModel catmodel = new SimpleCategoryModel();
 catmodel.setValue("2001", "Q1", new Integer(20));
 catmodel.setValue("2001", "Q2", new Integer(35));
 catmodel.setValue("2001", "Q3", new Integer(40));
 catmodel.setValue("2001", "Q4", new Integer(55));
 catmodel.setValue("2002", "Q1", new Integer(40));
 catmodel.setValue("2002", "Q2", new Integer(60));
 catmodel.setValue("2002", "Q3", new Integer(70));
 catmodel.setValue("2002", "Q4", new Integer(90));

 mychart.setModel(catmodel);
 }
}
```

## Gantt Chart



```
<zk>
 <div width="100%" apply="demo.GanttchartComposer">
 <fusionchart id="mychart" title="Gantt Chart" type="gantt" width="500" height="250" />
 </div>
</zk>
```

```
public class GanttchartComposer extends SelectorComposer<Div> {
 @Wire
 private Fusionchart mychart;

 @Override
 public void doAfterCompose(Div comp) throws Exception {
 super.doAfterCompose(comp);

 GanttModel ganttmodel = new GanttModel();

 String scheduled = "Scheduled";
 String actual = "Actual";

 ganttmodel.addValue(scheduled, new GanttTask("Write
Proposal", date(2008, 4, 1), date(2008, 4, 5), 0.0));
 ganttmodel.addValue(scheduled, new GanttTask("Requirements
Analysis", date(2008, 4, 10), date(2008, 5, 5), 0.0));
 ganttmodel.addValue(scheduled, new GanttTask("Design
Phase", date(2008, 5, 6), date(2008, 5, 30), 0.0));
 ganttmodel.addValue(scheduled, new GanttTask("Alpha
Implementation", date(2008, 6, 3), date(2008, 7, 31), 0.0));

 ganttmodel.addValue(actual, new GanttTask("Write Proposal",
date(2008, 4, 1), date(2008, 4, 3), 0.0));
 ganttmodel.addValue(actual, new GanttTask("Requirements
Analysis", date(2008, 4, 10), date(2008, 5, 15), 0.0));
 ganttmodel.addValue(actual, new GanttTask("Design Phase",
```

```
date(2008, 5, 15), date(2008, 6, 17), 0.0));
 ganttmodel.addValue(actual, new GanttTask("Alpha
Implementation", date(2008, 7, 1), date(2008, 9, 12), 0.0));

 mychart.setModel(ganttmodel);
 }

 @Listen("onClick = #mychart")
 public void show(Event event) {
 Map data = (Map) event.getData();
 Messagebox.show(data.toString());
 }

 private Date date(int year, int month, int day) {
 final java.util.Calendar calendar =
java.util.Calendar.getInstance();
 calendar.set(year, month - 1, day);
 return calendar.getTime();
 }
}
```

## Event Handling

Fusionchart supports mouse clicking event. You can receive information from event object, including clicked series, category, and value.

The following sample will show how to get an event object and display information on a message dialog.

```
public class FusionchartComposer extends SelectorComposer<Div> {
 @Wire
 private Fusionchart mychart;

 @Override
 public void doAfterCompose(Window comp) throws Exception {
 super.doAfterCompose(comp);
 // omitted...
 }

 @Listen("onClick = #mychart")
 public void show(Event event) {
 Map data = (Map) event.getData();
 Messagebox.show(data.toString());
 }
}
```

## Supported Events

Name	Event Type
onClick	Mouse clicking event. You can get information of clicked area from this event.

## Supported Children

\*None

## Use Cases

Version	Description	Example Location

## Version History

Version	Date	Content
6.0.1	April. 2012	add new component
6.5.3	June 2013	add Fusion Funnel chart integration
All	April 2014	entering maintenance mode

## References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Fusionchart.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/fusionchartz/Fusionchart.html#>
- [3] <http://www.fusioncharts.com/free/>

# Gmaps

---

## Gmaps

- Demonstration: Gmaps <sup>[1]</sup>
- Java API: in release zip <sup>[2]</sup>
- JavaScript API: N/A
- Style Guide: N/A

## Employment/Purpose

Components: gmaps, ginfo, gmarker, gpolyline, gpolygon, gimage, and gscreen.

A gmaps is a maps component that wraps the famous Google Maps service that you can control it and embedded it in your ZK web application page in pure Java. Inside the gmap, you can manipulate your maps and add contents to the maps to create convenient locality related web application. You can add ginfo to represent an anchored information window for the maps. You can add multiple gmarkers to indicate a special location. You can add gpolyline and gpolygon to indicate a path or an area. You can also overlay gimage and gscreen to indicate very special places.

## Authentication

### API Key

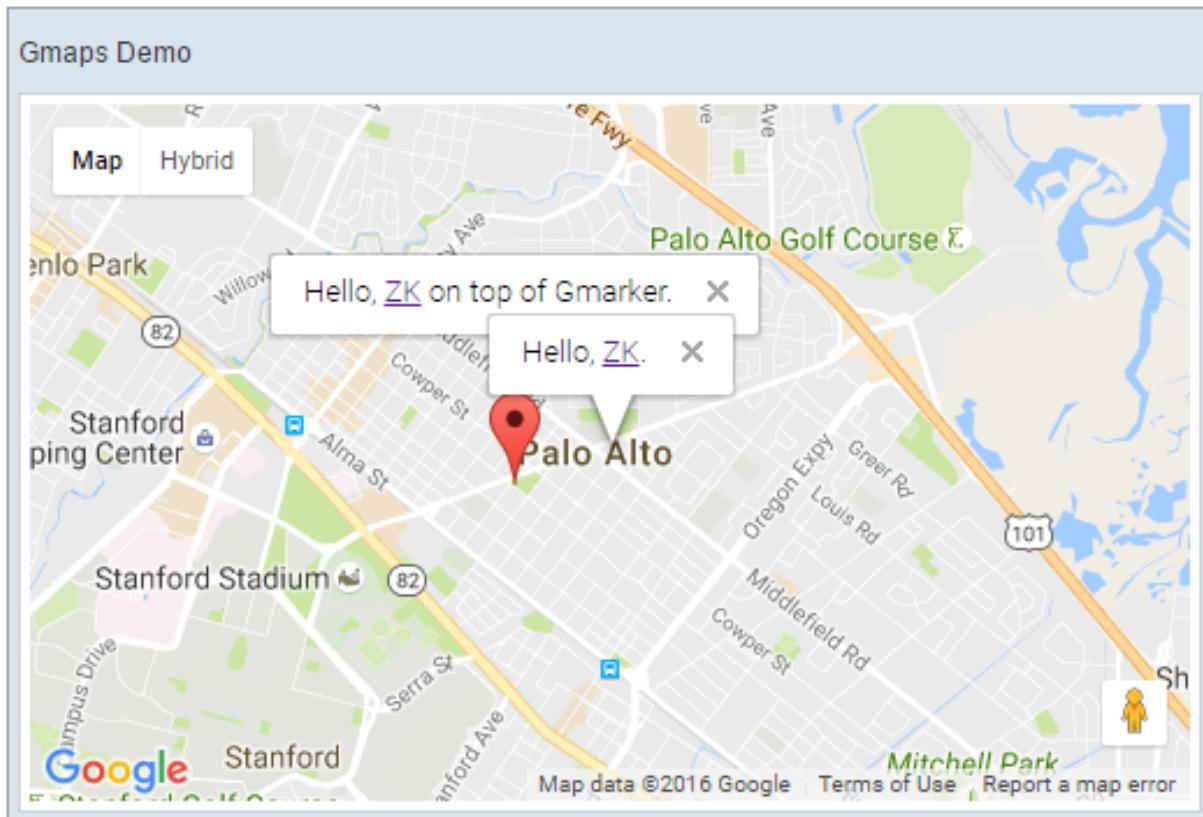
**Important:** Since 2016 an API Key is mandatory <sup>[3]</sup> in order to use the gmaps API. The APIKey is defined as a global JS variable as in the examples below. Also check out the trouble shooting section. **NOTE:** You'll have to upgrade to version **3.0.4** (older versions don't work anymore)

See Example below.

### Client ID

Alternatively you can specify a client-id using the **client**-property <gmaps client="YOUR\_CLIENT\_ID"/>

## Example



source code for gmaps version 3.0.4 (and above)

```
<window title="Gmaps Demo" border="normal" width="520px">
 <script type="text/javascript" content="zk.googleAPIkey='Your-Google-API-Key';"/>

 <!-- you may wish specify the version of google map api manually for some reason,
 use version="[version]" to do it.
 ex: <gmaps version="3.25" id="mymap" ... /> -->
 <gmaps id="mymap" width="500px" height="300px" showSmallCtrl="true">
 <ginfo id="myinfo" open="true">
 <![CDATA[
 Hello, ZK.
]]>
 </ginfo>

 <gmarker id="mymark" lat="37.4410" lng="-122.1490">
 <![CDATA[
 Hello, ZK on top of Gmarker.
]]>
 </gmarker>

 <attribute name="onMapClick">
 Gmarker gmarker = event.getGmarker();
 if (gmarker != null) {
 gmarker.setOpen(true);
 }
 </attribute>
 </gmaps>
</window>
```

```
 }
 </attribute>
</gmaps>
</window>

source code for gmaps version 2.x

<window title="Gmaps Demo" border="normal" width="520px">
<script type="text/javascript" content="zk.googleAPIkey='Your-Google-API-Key'" />

<gmaps id="mymap" width="500px" height="300px" showSmallCtrl="true">
<ginfo id="myinfo" open="true">
<! [CDATA[
Hello, ZK.
]]>
</ginfo>

<gmarker id="mymark" lat="37.4410" lng="-122.1490">
<! [CDATA[
Hello, ZK on top of Gmarker.
]]>
</gmarker>

<attribute name="onMapClick">
Gmarker gmarker = event.getGmarker();
if (gmarker != null) {
 gmarker.setOpen(true);
}
</attribute>
</gmaps>
</window>
```

## Properties

### Protocol

since 3.0.0

Specify which protocol to load the Maps API. Currently it supports `http` for insecure connections and `https` for secure connections.

### Supported Events

Name	Event Type
onSelect	<b>Event:</b> SelectEvent [6] Notifies one that the user has selected a new item(can be Ginfo, Gpolyline, or GPolygon) in the gmaps.
onInfoChange	<b>Event:</b> InfoChangeEvent Notifies that the current open information window has changed(opened/closed)
onMapDrop	<b>Event:</b> MapDropEvent Notifies that some component is dragged and dropped on the gmaps or gmarker component with latitude and longitude information.
onMapClick, onMapDoubleClick, onMapRightClick	<b>Event:</b> MapMouseEvent Notifies that some mouse action has been applied on the gmaps or gmarker component with latitude and longitude information.
onMapMove	<b>Event:</b> MapMoveEvent Notifies that the view center (latitude, longitude) of the gmaps has been moved.
onMapTypeChange	<b>Event:</b> MapTypeChangeEvent Notifies that the map type of the gmaps has been changed.
onMapZoom	<b>Event:</b> MapZoomEvent Notifies that the zoom level of the gmaps has been changed.

- Inherited Supported Events: XulElement

## Work with MVVM

3.0.4 already includes the `addon.xml` below, you don't need to add it manually.

since 6.0.0

For work with ZK6 MVVM, it is required to create an addon xml and add the server annotation as follows:

( You can download the sample project [here <sup>[4]</sup>] )

WEB-INF/gmapsz-bind-addon.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<language-addon>
 <!-- The name of this addon. It must be unique -->
 <addon-name>gmapszbind</addon-name>
 <!-- Specifies what other addon this depends
 <depends></depends>
 -->
 <!-- Which language this addon will be added to -->
 <language-name>xul/html</language-name>
 <!-- Add bind annotation to gmaps -->
 <component>
 <component-name>gmaps</component-name>
 <extends>gmaps</extends>
 <annotation>
 <annotation-name>ZKBIND</annotation-name>
 <property-name>mapType</property-name>
 <attribute>
```

```
 <attribute-name>ACCESS</attribute-name>
 <attribute-value>both</attribute-value>
 </attribute>
 <attribute>
 <attribute-name>SAVE_EVENT</attribute-name>
 <attribute-value>onMapTypeChange</attribute-value>
 </attribute>
 <attribute>
 <attribute-name>LOAD_REPLACEMENT</attribute-name>
 <attribute-value>mapType</attribute-value>
 </attribute>
 <attribute>
 <attribute-name>LOAD_TYPE</attribute-name>
 <attribute-value>java.lang.String</attribute-value>
 </attribute>
</annotation>
<annotation>
 <annotation-name>ZKBIND</annotation-name>
 <property-name>selectedItem</property-name>
 <attribute>
 <attribute-name>ACCESS</attribute-name>
 <attribute-value>both</attribute-value>
 </attribute>
 <attribute>
 <attribute-name>SAVE_EVENT</attribute-name>
 <attribute-value>onSelect</attribute-value>
 </attribute>
 <attribute>
 <attribute-name>LOAD_REPLACEMENT</attribute-name>
 <attribute-value>selectedItem</attribute-value>
 </attribute>
 <attribute>
 <attribute-name>LOAD_TYPE</attribute-name>
 <attribute-value>org.zkoss.zk.ui.Component</attribute-value>
 </attribute>
</annotation>
<annotation>
 <annotation-name>ZKBIND</annotation-name>
 <property-name>lat</property-name>
 <attribute>
 <attribute-name>ACCESS</attribute-name>
 <attribute-value>both</attribute-value>
 </attribute>
 <attribute>
 <attribute-name>SAVE_EVENT</attribute-name>
 <attribute-value>onMapMove</attribute-value>
 </attribute>
</annotation>
```

```
</annotation>
<annotation>
 <annotation-name>ZKBIND</annotation-name>
 <property-name>lng</property-name>
 <attribute>
 <attribute-name>ACCESS</attribute-name>
 <attribute-value>both</attribute-value>
 </attribute>
 <attribute>
 <attribute-name>SAVE_EVENT</attribute-name>
 <attribute-value>onMapMove</attribute-value>
 </attribute>
</annotation>
<annotation>
 <annotation-name>ZKBIND</annotation-name>
 <property-name>zoom</property-name>
 <attribute>
 <attribute-name>ACCESS</attribute-name>
 <attribute-value>both</attribute-value>
 </attribute>
 <attribute>
 <attribute-name>SAVE_EVENT</attribute-name>
 <attribute-value>onMapZoom</attribute-value>
 </attribute>
</annotation>
<annotation>
 <annotation-name>ZKBIND</annotation-name>
 <property-name>neLat</property-name>
 <attribute>
 <attribute-name>ACCESS</attribute-name>
 <attribute-value>both</attribute-value>
 </attribute>
 <attribute>
 <attribute-name>SAVE_EVENT</attribute-name>
 <attribute-value>onMapMove</attribute-value>
 </attribute>
</annotation>
<annotation>
 <annotation-name>ZKBIND</annotation-name>
 <property-name>neLng</property-name>
 <attribute>
 <attribute-name>ACCESS</attribute-name>
 <attribute-value>both</attribute-value>
 </attribute>
 <attribute>
 <attribute-name>SAVE_EVENT</attribute-name>
 <attribute-value>onMapMove</attribute-value>
 </attribute>

```

```
</attribute>
</annotation>
<annotation>
 <annotation-name>ZKBIND</annotation-name>
 <property-name>swLat</property-name>
 <attribute>
 <attribute-name>ACCESS</attribute-name>
 <attribute-value>both</attribute-value>
 </attribute>
 <attribute>
 <attribute-name>SAVE_EVENT</attribute-name>
 <attribute-value>onMapMove</attribute-value>
 </attribute>
</annotation>
<annotation>
 <annotation-name>ZKBIND</annotation-name>
 <property-name>swLng</property-name>
 <attribute>
 <attribute-name>ACCESS</attribute-name>
 <attribute-value>both</attribute-value>
 </attribute>
 <attribute>
 <attribute-name>SAVE_EVENT</attribute-name>
 <attribute-value>onMapMove</attribute-value>
 </attribute>
</annotation>
</component>
</language-addon>
```

then add it into WEB-INF/zk.xml

```
<zk>
 <language-config>
 <addon-uri>/WEB-INF/gmapsz-bind-addon.xml</addon-uri>
 </language-config>
</zk>
```

## Supported Children

\* Ginfo, Gmarker, Gpolyline, Gpolygon, Gimage, Gscreen, Gcircle

## Trouble Shooting

Starting from June 2016, an API key is required to use Google Maps APIs. If you are seeing the following error in your JS console, please obtain an API key from Google.

Google Maps API error: MissingKeyMapError

For more information please refer to this link <sup>[5]</sup>.

Note that ZK GMaps integrates ZK and Google Maps, allowing developers to include Google Maps inside a ZK application easily. Google Maps itself is a 3rd party library and is licensed under its own terms and conditions.

## Use Cases

Version	Description	Example Location
---------	-------------	------------------

## Version History

Version	Date	Content
---------	------	---------

## References

- [1] [http://www.zkoss.org/zkdemo/reporting/google\\_map](http://www.zkoss.org/zkdemo/reporting/google_map)
- [2] <https://github.com/zkoss/zkgmapsz/releases>
- [3] <https://developers.google.com/maps/faq#keysystem>
- [4] <https://zkgmapsz.googlecode.com/svn/trunk/gmapszTest>
- [5] [https://developers.google.com/maps/pricing-and-plans/standard-plan-2016-update?utm\\_source=geoblog&utm\\_medium=social&utm\\_campaign=2016-geo-na-website-gmedia-blogs-us-blogPost&utm\\_content=TBC](https://developers.google.com/maps/pricing-and-plans/standard-plan-2016-update?utm_source=geoblog&utm_medium=social&utm_campaign=2016-geo-na-website-gmedia-blogs-us-blogPost&utm_content=TBC)

# Gimage

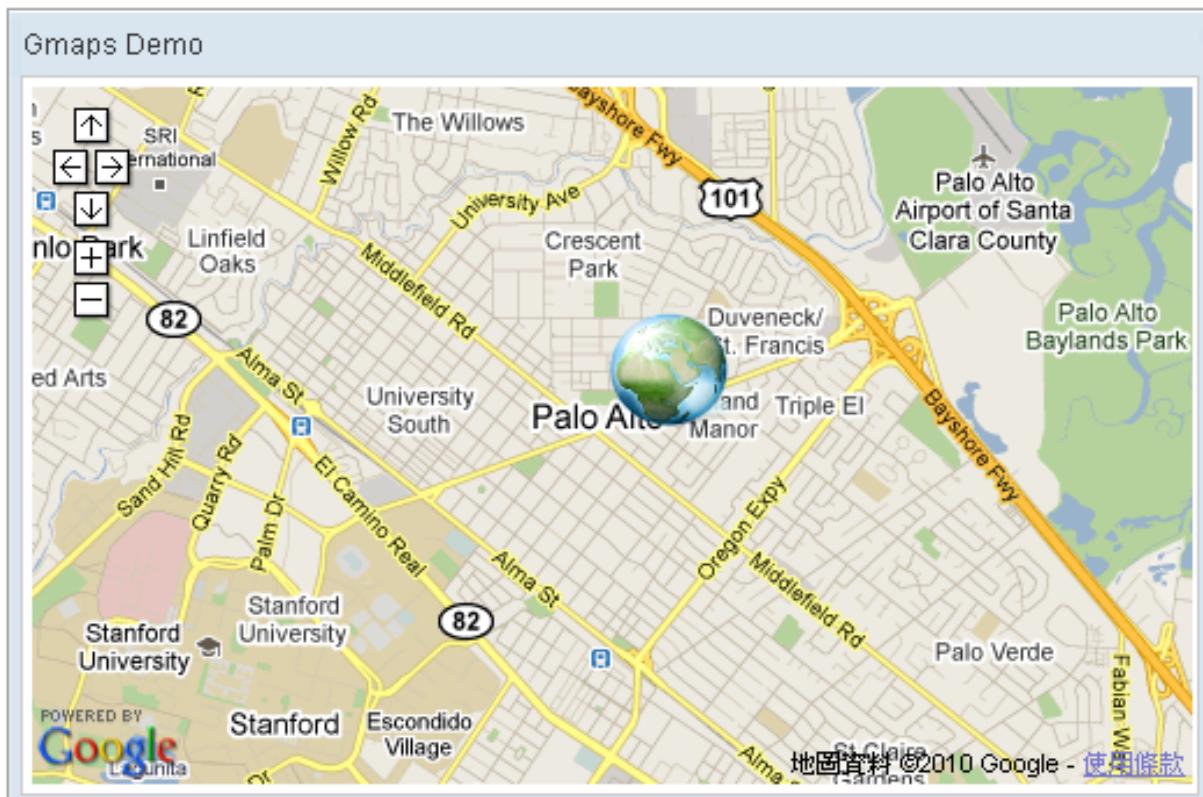
## Gimage

- Demonstration: Gmaps [1]
- Java API: N/A
- JavaScript API: N/A
- Style Guide: N/A

## Employment/Purpose

Image anchored on two specified (latitude, longitude) points(south-west and north-east) which is zoomed and moved with the Google Maps.

## Example



```
<script type="text/javascript" content="zk.googleAPIkey='Your-Google-API-Key'" />
<gmaps id="mymap" width="500px" height="300px" showSmallCtrl="true">
 <gimage src="/img/Centigrade-Widget-Icons/Globe-128x128.png"
 swlat="37.44215478" swlng="-122.14273453" nelat="37.45033195"
 nelng="-122.13191986" />
</gmaps>
```

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

## Supported Children

\*None

## Use Cases

Version	Description	Example Location
---------	-------------	------------------

## Version History

Version	Date	Content
---------	------	---------

## Ginfo

### Ginfo

- Demonstration: Gmaps [1]
- Java API: N/A
- JavaScript API: N/A
- Style Guide: N/A

### Employment/Purpose

The popup info window of the Google Maps. You can specify the content in pure text or HTML.

## Example

Gmaps

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

## Supported Children

\*None

## Use Cases

Version	Description	Example Location
---------	-------------	------------------

## Version History

Version	Date	Content
---------	------	---------

# Gmarker

## Gmarker

- Demonstration: Gmaps [1]
- Java API: N/A
- JavaScript API: N/A
- Style Guide: N/A

## Employment/Purpose

The gmarker of the Google Maps that you can used to indicate a specific (latitude, longitude) points. The icon, shadow, color, and so on are customizable.

## Example

Gmaps

## Supported Events

Name	Event Type
onMapDrop	<b>Event:</b> MapDropEvent Notifies that some component is dragged and dropped on the gmaps or gmarker component with latitude and longitude information.

- Inherited Supported Events: Ginfo

## Supported Children

\*None

## Use Cases

Version	Description	Example Location
---------	-------------	------------------

## Version History

Version	Date	Content
---------	------	---------

# G polyline

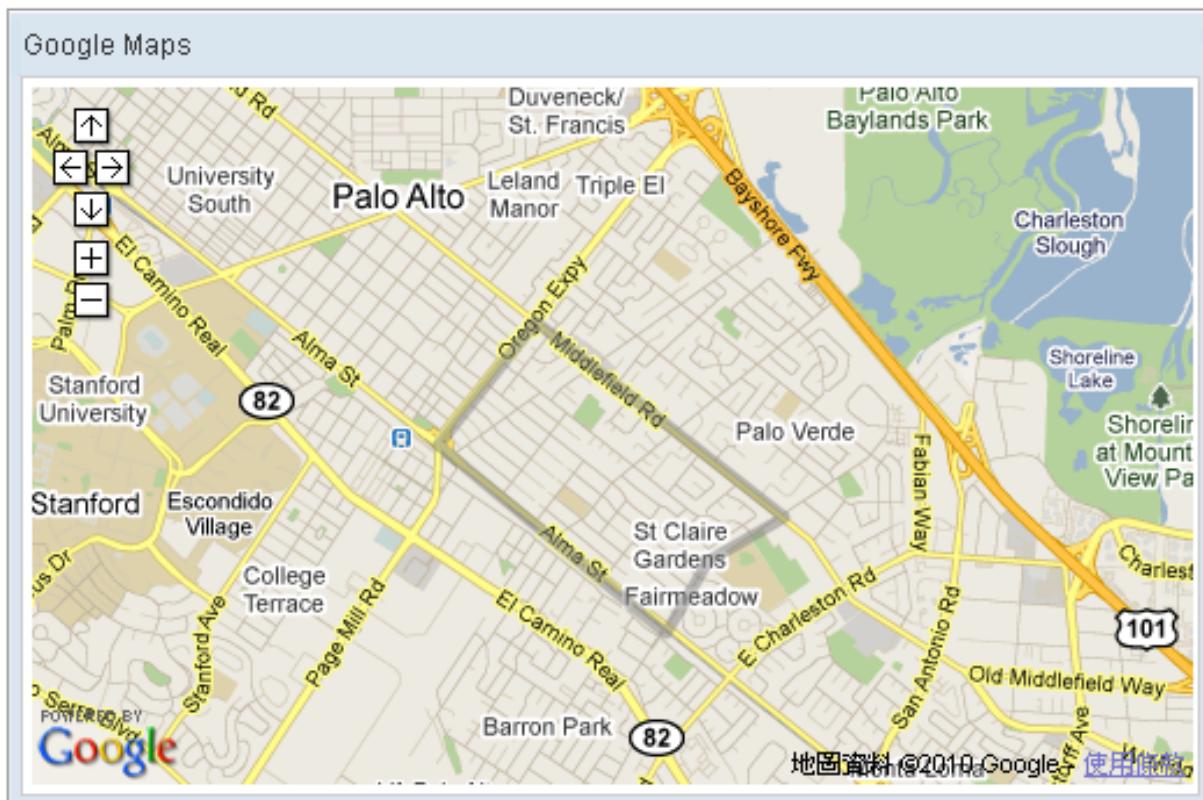
## G polyline

- Demonstration: Gmaps [1]
- Java API: N/A
- JavaScript API: N/A
- Style Guide: N/A

## Employment/Purpose

Polyline drawn on the Google Maps per the given (latitude, longitude) points and visible zoom level(0~3).

## Example



```
<window title="Google Maps" border="normal" width="520px">
 <script type="text/javascript" content="zk.googleAPIkey='Your-Google-API-Key'"/>
 <gmaps id="mymap" width="500px" height="300px" showSmallCtrl="true">
 <gpolyline points="37.42838786,-122.13998795,3,37.43561240,-122.13277816,3,37.42416187,-122.11441040,3,
 37.42157162,-122.12007522,3,37.41734524,-122.12316513,3,37.42838786,-122.13998795,3"/>
 </gmaps>
</window>
```

## Disable Path Encoding

since 3.3.0

In order to preserve bandwidth the polyline/polygon path coordinates are encoded (default) with a lossy algorithm [1] (i.e. some precision is lost - visible at high zoom levels). In cases where the highest precision is needed (e.g. when editing a polyline/polygon) this encoding can be disabled by setting `pathEncoded="false"` / `polygon.setPathEncoded(false);` - at the cost of increased network request and response sizes.

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

## Supported Children

\*None

## Use Cases

Version	Description	Example Location

## Version History

Version	Date	Content

## References

[1] <https://developers.google.cn/maps/documentation/utilities/polylinealgorithm>

# Gpolygon

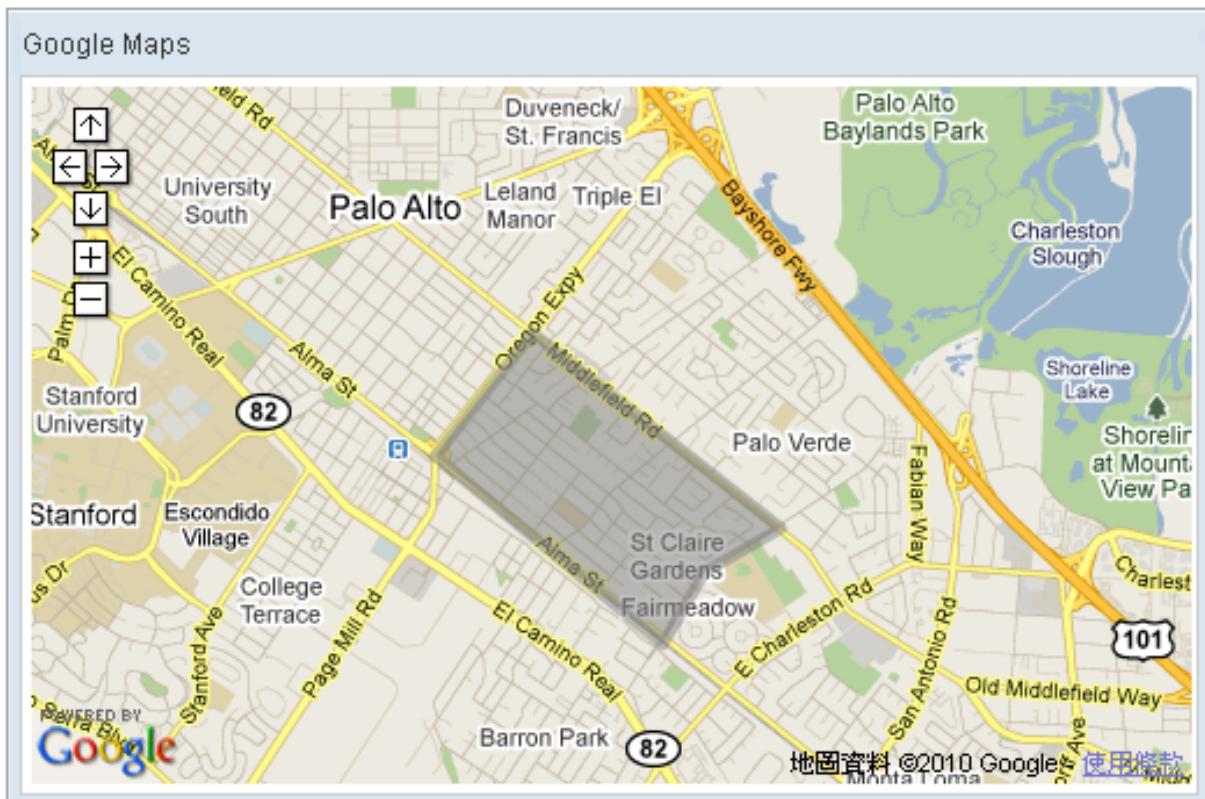
## Gpolygon

- Demonstration: Gmaps [1]
- Java API: N/A
- JavaScript API: N/A
- Style Guide: N/A

## Employment/Purpose

Polygon drawn on the Google Maps per the given (latitude, longitude) points, visible zoom level(0~3), and fill color, etc..

## Example



```
<window title="Google Maps" border="normal" width="520px">
 <script type="text/javascript" content="zk.googleAPIkey='Your-Google-API-Key'"/>
 <gmaps id="mymap" width="500px" height="300px" showSmallCtrl="true">
 <gpolygon points="37.42838786,-122.13998795,3,37.43561240,-122.13277816,3,37.42416187,-122.11441040,3,
 37.42157162,-122.12007522,3,37.41734524,-122.12316513,3"/>
 </gmaps>
</window>
```

## Disable Path Encoding

since 3.3.0

see: Gpolyline - Disable\_Path\_Encoding

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: Gpolyline

## Supported Children

\*None

## Use Cases

Version	Description	Example Location
---------	-------------	------------------

## Version History

Version	Date	Content
---------	------	---------

# Gscreen

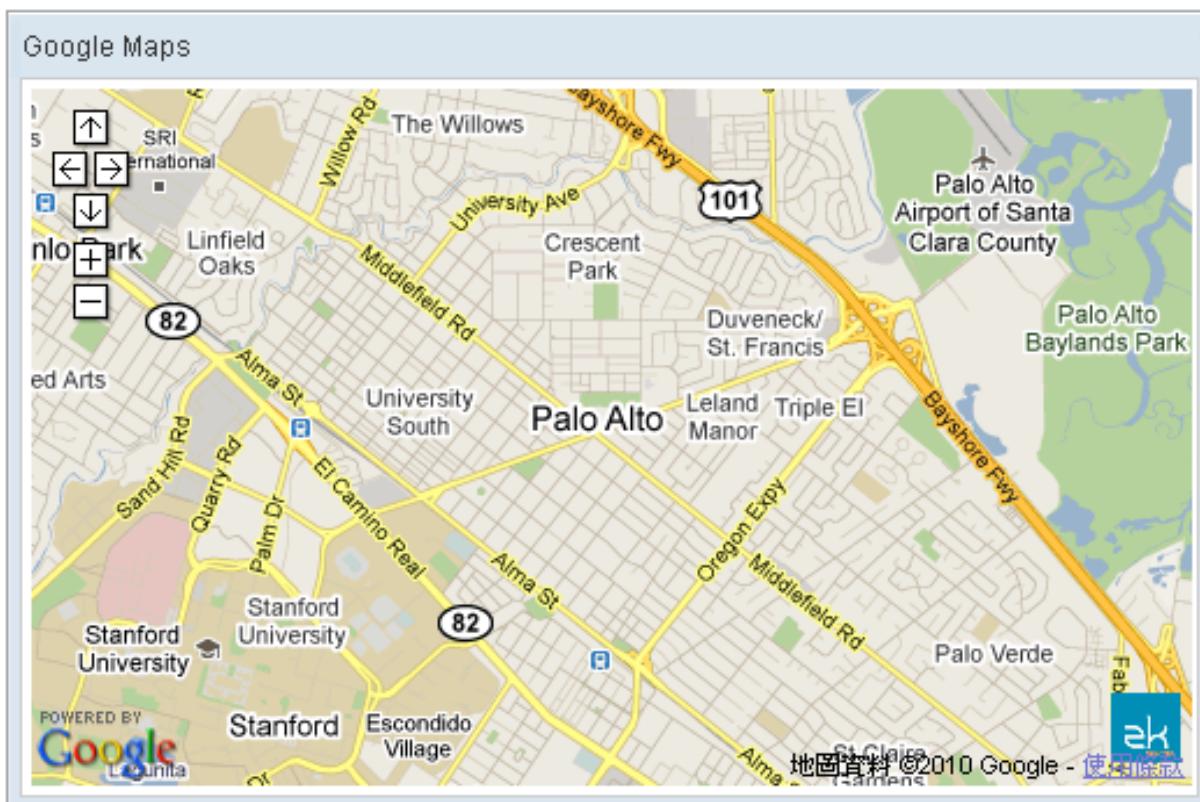
## Gscreen

- Demonstration: Gmaps [1]
- Java API: N/A
- JavaScript API: N/A
- Style Guide: N/A

## Employment/Purpose

A rectangular image on the Gmaps whose position remains fixed on the screen even you move the maps. It can be used in showing logos, heads-up display, etc.

## Example



```
<window title="Google Maps" border="normal" width="520px">
<script type="text/javascript" content="zk.googleAPIkey='Your-Google-API-Key'"/>
<gmaps id="mymap" width="500px" height="300px" showSmallCtrl="true">
<gscreen src="/img/ZK-Logo.gif" screenX="465" screenY="10" offsetX="0" offsetY="0" width="30" height="30"/>
</gmaps>
</window>
```

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

## Supported Children

\*None

## Use Cases

Version	Description	Example Location

## Version History

Version	Date	Content

## Gcircle

---

### Gcircle

- Demonstration: N/A
- Java API: N/A
- JavaScript API: N/A
- Style Guide: N/A

## Employment/Purpose

Circle drawn on the Google Maps per the given (latitude, longitude) points, fill color, etc..

## Example

```
<window title="Google Maps" border="normal" width="520px">
 <script type="text/javascript" content="zk.googleAPIkey='Your-Google-API-Key'"/>
 <gmaps id="mymap" width="500px" height="300px" showSmallCtrl="true">
 <gcircle id="cmarker" lat="0" lng="0" radius="10000" strokeColor="red" fillColor="red"></gcircle>
 </gmaps>
</window>
```

## Supported Events

Name	Event Type
onCenterChange	org.zkoss.gmaps.event.CenterChangeEvent
onRadiusChange	org.zkoss.gmaps.event.RadiusChangeEvent

- Inherited Supported Events: XulElement

## Supported Children

\*None

## Use Cases

Version	Description	Example Location

## Version History

Version	Date	Content

## Jasperreport

### Jasperreport

- Demonstration: Jasperreport <sup>[1]</sup>
- Java API: Jasperreport <sup>[2]</sup>
- JavaScript API: Jasperreport <sup>[3]</sup>
- Style Guide: N/A
- Available for ZK:
- CE PE EE

## Employment/Purpose

The JasperReport component is based on JasperReports Library, a report generator (e.g. XML generator, PDF generator etc.). This component is used to generate a Jasper report into an inline frame.

## Example

The screenshot shows the ZK Studio interface with a Jasper Report preview and its corresponding Zscript code.

**Jasper Report Preview:**

The preview shows a report titled "Address Report" generated from a "CustomDataSource from java". The report contains three sections: "1. Berne", "2. Boston", and "3. Chicago", each displaying a table of address data. The tables have columns "ID", "Name", and "Street".

ID	Name	Street
22	Bill Ott	250 - 20th Ave.
9	James Schneider	277 Seventh Av.

Count : 2

ID	Name	Street
32	Michael Ott	339 College Av.
23	Julia Heiniger	358 College Av.

Count : 2

ID	Name	Street
39	Mary Karsen	202 College Av.
35	George Karsen	412 College Av.
11	Julia White	412 Upland Pl.

Count : 3

**Zscript Code:**

```
<jasperreport id="report" height="360px" />

<zscript>
import org.zkoss.zkdemo.userguide.CustomDataSource;

//Preparing parameters
Map parameters = new HashMap();
parameters.put("ReportTitle", "Address Report");
parameters.put("DataFile", "CustomDataSource from
java");

report.setSrc("/data/jasperreport.jasper");
report.setParameters(parameters);
report.setDatasource(new CustomDataSource());
report.setType("pdf");
</zscript>
```

## Provide Export Parameters

The Jasperreport component API provides a way to specify export parameters. To do so, you should put a Map containing export parameters within the parameters Map, with key "exportParameter". For example:

```
Map parameters = new HashMap();
Map exportParams = new HashMap();
exportParams.put("net.sf.jasperreports.export.mypropertynname",
true);
parameters.put("exportParameter", exportParams);
report.setParameters(parameters); // report is the Jasperreport
component
```

## One-Page-Per-Sheet Property

In Jasperreport engine, this property is default to be false. However, the default behavior is turned on in the Jasperreport component, in which case when the report is exported as Excel format there will be one sheet generated for each page. To override this setting, set it as an export parameter as the following:

```
Map parameters = new HashMap();
Map exportParams = new HashMap();

exportParams.put(JRXlsAbstractExporterParameter.PROPERTY_ONE_PAGE_PER_SHEET.toString(),
false);
parameters.put("exportParameter", exportParams);
report.setParameters(parameters); // report is the Jasperreport
component
```

## exportName

[ since 8.6.1 ]

You can specify the export file name for the download if any, the full file name will be `exportName + "." + format`.

Note: `exportName` can not be empty or null.

Default: "report"

```
<jasperreport exportName="test"/>
```

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: Iframe

## Supported Children

\*NONE

## Supported Type

### Usage

#### Java Code

```
//Jasperreport report;
report.setType("pdf"); // report is the Jasperreport component
```

#### Zul Code

```
<jasperreport id="report" type="pdf" />
```

### Type List

Type	Version
pdf	
xml	
html	
rtf	
xls	
jxl	
csv	
odt	
xlsx	since 5.0.8
docx	since 5.0.8
graphic2d	since 5.0.8
ods	since 5.0.8
pptx	since 5.0.8
txt	since 5.0.8
xhtml	since 5.0.8

## Supported JasperReport Version

ZK	JasperReport	Transitive Dependency
9.6.0	6.14.0	optional (net.sf.jasperreports;jasperreports [4] needs to be added manually on demand)
9.5.0	6.14.0	added by default
9.0.0	6.6.0	added by default
8.6.0	6.5.1	added by default
7.0.0	4.5.1	added by default
6.0.0	4.0.1	added by default

## Use Cases

Version	Description
5.0.2	How to use Dynamic Jasper Reports [5]
6/5.0.8	Create a Report with ZK using iReport and JasperReports [6]

## Troubleshooting

### Linux

Jasperreport depends on the fonts you use in the report. For more information, please refer to ZK Installation Guide: Linux.

## Version History

Version	Date	Content
5.0.1	March 2010	Support Hibernate and SQL connections
5.0.8	Upgrade JaserReport version to 4.0 and support new JasperReport exporter	

## References

- [1] <http://www.zkoss.org/zkdemo/reporting/jasperreport>
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkex/zul/Jasperreport.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zkex/utl/Jasperreport.html#>
- [4] <https://mvnrepository.com/artifact/net.sf.jasperreports/jasperreports>
- [5] <http://www.zkoss.org/forum/listComment/10873>
- [6] [http://books.zkoss.org/wiki/Small\\_Talks/2012/April/Create\\_a\\_Report\\_with\\_ZK\\_using\\_iReport\\_and\\_JasperReports](http://books.zkoss.org/wiki/Small_Talks/2012/April/Create_a_Report_with_ZK_using_iReport_and_JasperReports)

# Timeline

---

## Timeline

Deprecated, please use new Timeline instead [1]

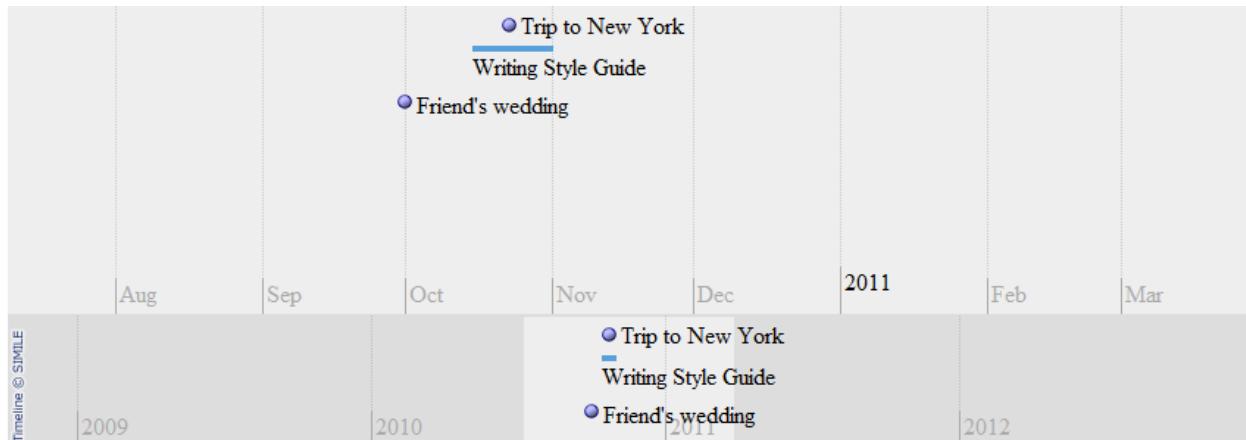
## Employment/Purpose

Timeline [2] is a DHTML-based Ajax widget for visualizing time-based events developed by SIMILE project of MIT. It is like Google Maps for time-based information.

A timeline contains one or more Bands, which can be panned infinitely by dragging with the mouse pointer, using the mouse scroll-wheel or the keyboard's arrow buttons.

A band can be configured to synchronize with another band such that panning one band also scrolls the other.

## Example



```
<timeline height="300px" width="100%">
 <bandinfo width="70%" id="b1" intervalUnit="month"
 intervalPixels="100" eventSourceUrl="misc/timeline_ex1.xml">
 </bandinfo>
 <bandinfo width="30%" intervalUnit="year" intervalPixels="200"
 syncWith="b1" eventSourceUrl="misc/timeline_ex1.xml">
 </bandinfo>
</timeline>
```

timeline\_ex1.xml

```
<data>
 <event start="Oct 1 2010 00:00:00 GMT" end="Oct 1 2010 00:00:00 GMT"
 title="Friend's wedding">
 I'm not sure precisely when my friend's wedding is.
 </event>
 <event start="Oct 15 2010 11:00:00 GMT" end="Nov 1 2010 09:00:00 GMT"
 isDuration="true" title="Writing Style Guide"
 image="http://simile.mit.edu/images/csail-logo.gif">
 A few days to write some documentation for
 </event>
</data>
```

```

Timeline

.

</event>
<event start="Oct 23 2010 06:12:33 GMT" title="Trip to New York"
 link="http://www.priceline.com/">
 Happy New Year !
</event>
</data>

```

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

## Supported Children

\* Bandinfo

## Use Cases

Version	Description	Example Location
3.6	Smalltalk: Using Timeline Component, Part I	Using Timeline Component, Part I
3.6	Smalltalk: Using Timeline Component, Part II	Using Timeline Component, Part II
3.6	How to use ListModel in Timeline component	How to use ListModel in Timeline component
3.6	Using the New Feature Of The Timeline Component	Using the New Feature Of The Timeline Component
3.6	Integrating Google Calendar with ZK Timeline Component	Integrating Google Calendar with ZK Timeline Component

## Version History

Version	Date	Content

## References

- [1] <http://blog.zkoss.org/index.php/2015/01/13/introducing-new-zk-addon-zk-timeline/>  
[2] <http://www.simile-widgets.org/timeline/>

# Bandinfo

---

## Bandinfo

Deprecated, please use new Timeline instead [1]

## Employment/Purpose

Defines the bandinfo of a timeline.

A band layer is the main building block for timeline and it's the object that is responsible for painting the band itself.

## Example



```
<timeline height="300px" width="100%">
 <bandinfo width="70%" id="b1" intervalUnit="month"
 intervalPixels="100" eventSourceUrl="misc/timeline_ex1.xml">
 </bandinfo>
 <bandinfo width="30%" intervalUnit="year" intervalPixels="200"
 syncWith="b1" eventSourceUrl="misc/timeline_ex1.xml">
 </bandinfo>
</timeline>
```

timeline\_ex1.xml

```
<data>
 <event start="Oct 1 2010 00:00:00 GMT" end="Oct 1 2010 00:00:00 GMT"
 title="Friend's wedding">
 I'm not sure precisely when my friend's wedding is.
 </event>
 <event start="Oct 15 2010 11:00:00 GMT" end="Nov 1 2010 09:00:00 GMT"
 isDuration="true" title="Writing Style Guide"
 image="http://simile.mit.edu/images/csail-logo.gif">
 A few days to write some documentation for
 Timeline
 .
 </event>
</data>
```

```
<event start="Oct 23 2010 06:12:33 GMT" title="Trip to New York"
 link="http://www.priceline.com/">
 Happy New Year !
</event>
</data>
```

## Supported Events

Name	Event Type
onOccurEventSelect	<b>Event:</b> OccurEventSelectEvent Denotes the Occur Event in the bandinfo component has been clicked by the user.
onBandScroll	<b>Event:</b> BandScrollEvent Denotes the bandinfo component has been moved by the user.

- Inherited Supported Events: XulElement

## Supported Children

\* Hotzone

## Use Cases

Timeline

## Version History

Version	Date	Content

# Hotzone

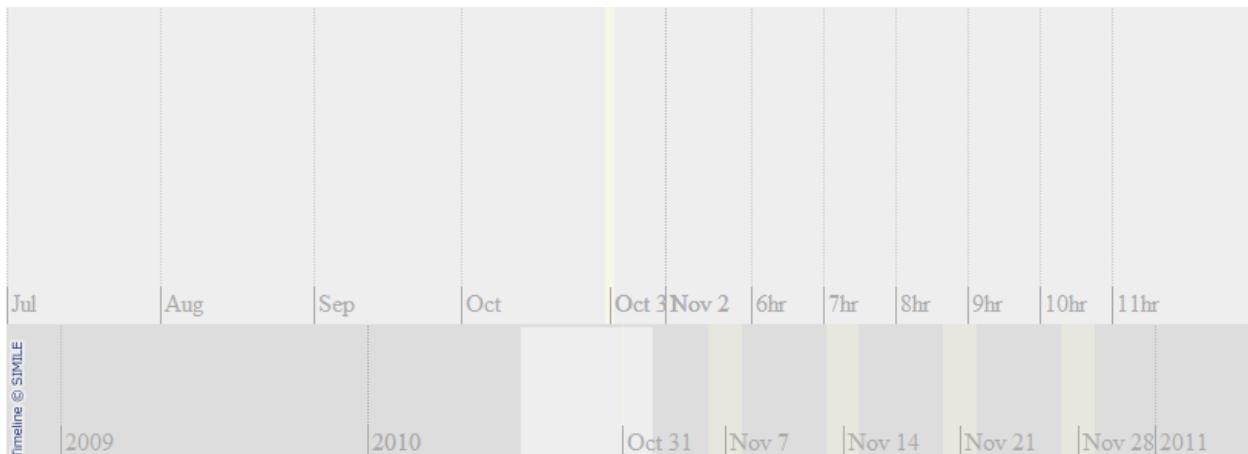
## Hotzone

Deprecated, please use new Timeline instead [1]

## Employment/Purpose

Defines the hotzone of a bandinfo .

## Example



```
<window id="win">
<zscript>
<![CDATA[
import java.util.Date;
import java.util.TimeZone;

TimeZone zone=TimeZone.getTimeZone("GMT-05");

Date current=new Date(Date.parse("Jun 28 2010 00:00:00 GMT-0500"));
//for hotzone of band #1
Date d1=new Date(Date.parse("Nov 01 2010 00:00:00 GMT-0500"));
Date d2=new Date(Date.parse("Dec 01 2010 00:00:00 GMT-0500"));
Date d3=new Date(Date.parse("Nov 02 2010 00:00:00 GMT-0500"));
Date d4=new Date(Date.parse("Nov 04 2010 00:00:00 GMT-0500"));
Date d5=new Date(Date.parse("Nov 02 2010 06:00:00 GMT-0500"));
Date d6=new Date(Date.parse("Nov 02 2010 12:00:00 GMT-0500"));

//for hotzone of band #2
Date d7=new Date(Date.parse("Nov 01 2010 00:00:00 GMT-0500"));
Date d8=new Date(Date.parse("Dec 01 2010 00:00:00 GMT-0500"));
]]>
</zscript>
```

```

<timeline height="300px" width="100%">
 <bandinfo width="70%" id="b1" intervalUnit="month"
 intervalPixels="100"
 timeZone="${zone}" date="${current}">
 <hotzone start="${d1}" end="${d2}" magnify="10" unit="week" />
 <hotzone start="${d3}" end="${d4}" magnify="7" unit="day" />
 <hotzone start="${d5}" end="${d6}" magnify="5" unit="hour" />
 </bandinfo>
 <bandinfo timeZone="${zone}" date="${current}"
 width="30%" intervalUnit="year" intervalPixels="200"
 syncWith="b1">
 <hotzone start="${d7}" end="${d8}" magnify="20" unit="week" />
 </bandinfo>
</timeline>
</window>

```

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

## Supported Children

\*NONE

## Use Cases

Timeline

## Version History

Version	Date	Content

# Timeplot

## Timeplot

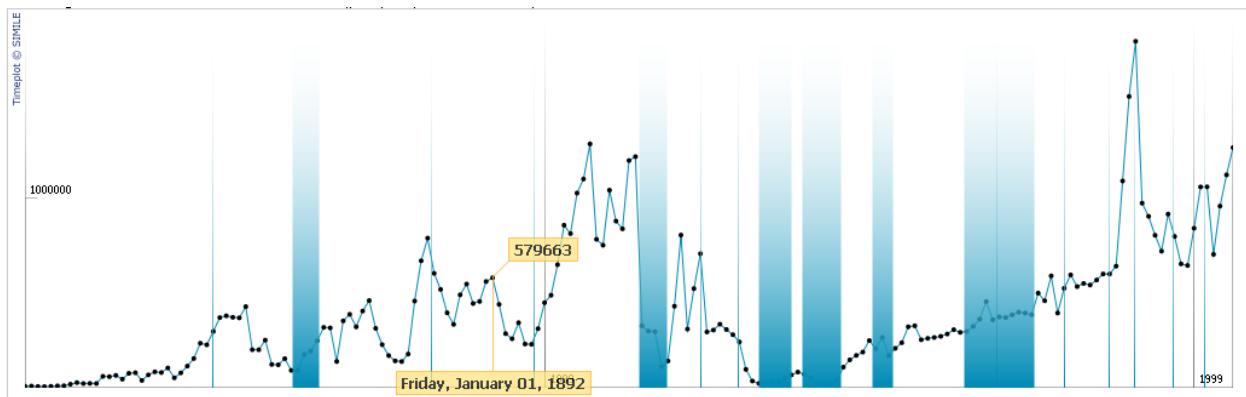
Deprecated, please use ZK Charts instead <sup>[1]</sup>

## Employment/Purpose

Timeplot <sup>[2]</sup> is a DHTML-based AJAXy widget for plotting time series and overlay time-based events over them.

You can populate Timeplot with data by pointing it to an space or comma separated file. Timeplot also supports Timeline's XML format, meaning that you can reuse the same data file of your Timeline and overlay them over a time series plot.

## Example



<zk>

```
<div width="1200px">
 New Legal Permanent Residents in the U.S. (per year) vs.
 U.S.
 History
 <zscript>
 import org.zkforge.timeplot.geometry.*;
 import org.zkforge.timeplot.data.*;
 import org.zkforge.timeplot.operator.*;
 import org.zkforge.timeline.data.OccurEvent;
 import org.zkoss.zul.*;
 PlotDataSource pds=new PlotDataSource();
 pds.setDataSourceUri("misc/immigration.txt");
 pds.setSeparator(" ");
 ValueGeometry vg=new DefaultValueGeometry();
 vg.setGridColor("#000000");
 TimeGeometry tg=new DefaultTimeGeometry();
 tg.setAxisLabelsPlacement("bottom");
 </zscript>
 <timeplot width="1000px" height="300px">
 <plotinfo id="plot 1" plotDataSource="#36; {pds}">

```

```

dotColor="#000000" showValues="true"
lineColor="#008bb6"
valueGeometry="\#36;{vg}"
timeGeometry="\#36;{tg}"
eventSourceUri="misc/us_history.xml" />
</timeplot>
</div>
</zk>
```

## immigration.txt

```

1820 8385
1821 9127
1822 6911
1823 6354
1824 7912
1825 10199
1826 10837
1827 18875
1828 27382
1829 22520
1830 23322
```

## us\_history.xml

```

<?xml version="1.0" encoding="UTF-8"?>

<data>

<event start="Jan 01 1849 00:00:00 GMT" title="California Gold Rush" link="http://en.wikipedia.org/wiki/California_Gold_Rush"></event>

<event start="Apr 12 1861 00:00:00 GMT" end="Apr 09 1865 00:00:00 GMT" title="American Civil War" link="http://en.wikipedia.org/wiki/American_Civil_War"></event>

<event start="Aug 03 1882 00:00:00 GMT" title="Immigration Act of 1882" link="http://en.wikipedia.org/wiki/1882_Immigration_Act">The U.S. Congress passes a new Immigration Act that stated that a 50 cents tax would be levied on all aliens landing at United States ports. The money collected was to be used to defray the expenses of regulating immigration and for the care of immigrants after landing. The legislation also gave powers to the authorities to deny entry to "convicts (except those convicted of political offences), lunatics, idiots and persons likely to become public charges".</event>

<event start="Apr 25 1898 00:00:00 GMT" title="Spanish-American War" link="http://en.wikipedia.org/wiki/Spanish-American_War"></event>

<event start="Jul 28 1914 00:00:00 GMT" end="Nov 11 1918 00:00:00 GMT" title="World War I" link="http://en.wikipedia.org/wiki/World_War_I"></event>

<event start="Jan 01 1924 00:00:00 GMT" title="Immigration Act of 1924" link="http://en.wikipedia.org/wiki/1924_Immigration_Act">The Immigration Act of 1924, which included the National Origins Act, Asian Exclusion Act or the Johnson-Reed Act, was a United States federal law that limited the number of immigrants who could be admitted from any country to 2% of the number of people from that country who were already living in the United States in 1890, according to the Census of 1890. It excluded immigration to the US of Asians. It superseded the 1921 Emergency Quota Act. The law was aimed at further restricting the Southern and Eastern Europeans who had begun to enter the country in large numbers beginning in the 1890s, as well as East Asians and Asian Indians, who were prohibited from immigrating
```

```

entirely. It set no limits on immigration from Latin America.</event>

<event start="Oct 29 1929 00:00:00 GMT" title="Black Tuesday" link=""></event>

<event start="Jan 01 1933 00:00:00 GMT" end="Jan 01 1938 00:00:00 GMT" title="New Deal" link="http://en.wikipedia.org/wiki/New_Deal"></event>

<event start="Sep 01 1939 00:00:00 GMT" end="Sep 02 1945 00:00:00 GMT" title="World War II" link="http://en.wikipedia.org/wiki/World_War_II"></event>

<event start="Jun 25 1950 00:00:00 GMT" end="Jul 27 1953 00:00:00 GMT" title="Korean War" link="http://en.wikipedia.org/wiki/Korean_War"></event>

<event start="Aug 02 1964 00:00:00 GMT" end="Apr 30 1975 00:00:00 GMT" title="Vietnam War" link="http://en.wikipedia.org/wiki/Vietnam_War"></event>

<event start="Jul 21 1969 00:00:00 GMT" title="Man on the Moon">Neil Armstrong becomes the first person to walk on the Moon as the
commander of the American mission Apollo 11.</event>

<event start="Jan 01 1980 00:00:00 GMT" title="Refugee Act">The Refugee Act is a 1980 United States federal law that reformed
United States immigration law and admitted refugees on systematic basis
for humanitarian reasons. A 1985 ceiling of 70,000 refugees, with
270,000 immigrants total and 20,000 from any one country, was
established.</event>

<event start="Nov 06 1986 00:00:00 GMT" title="Immigration Reform and Control Act of 1986" link="http://en.wikipedia.org/wiki/Immigration_Reform_and_Control_Act_of_1986">The Act made it illegal to knowingly hire or recruit illegal
immigrants, required employers to attest to their employees' immigration status, and granted amnesty to illegal immigrants who
entered the United States before January 1, 1982 and had resided there
continuously.</event>

<event start="Nov 29 1990 00:00:00 GMT" title="Immigration Act of 1990" link="http://en.wikipedia.org/wiki/Immigration_Act_of_1990"></event>

<event start="Sep 30 1996 00:00:00 GMT" title="Illegal Immigration Reform and Immigrant Responsibility Act of 1996" link="http://en.wikipedia.org/wiki/Illegal_Immigration_Reform_and_Immigrant_Responsibility_Act_of_1996"></event>

<event start="Sep 11 2001 00:00:00 GMT" title="Attack at the World Trade Center" link="http://en.wikipedia.org/wiki/Vietnam_War"></event>

</data>

```

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: HtmlBasedComponent

## Supported Children

- \* Plotinfo

## Use Cases

Version	Description	Example Location
3.6	Smalltalk: Using Timeplot component-Part I	Using Timeplot component-Part I
3.6	Smalltalk: Using Timeplot component-Part II	Using Timeplot component-Part II

## Version History

Version	Date	Content
	July, 2014	Deprecate this component, please use ZK Charts <sup>[1]</sup> instead

## References

[1] <http://blog.zkoss.org/index.php/2014/07/15/zk-charts-paint-timeplot-chart-with-zk-charts/>

[2] <http://www.simile-widgets.org/timeplot/>

## Plotinfo

---

### Plotinfo

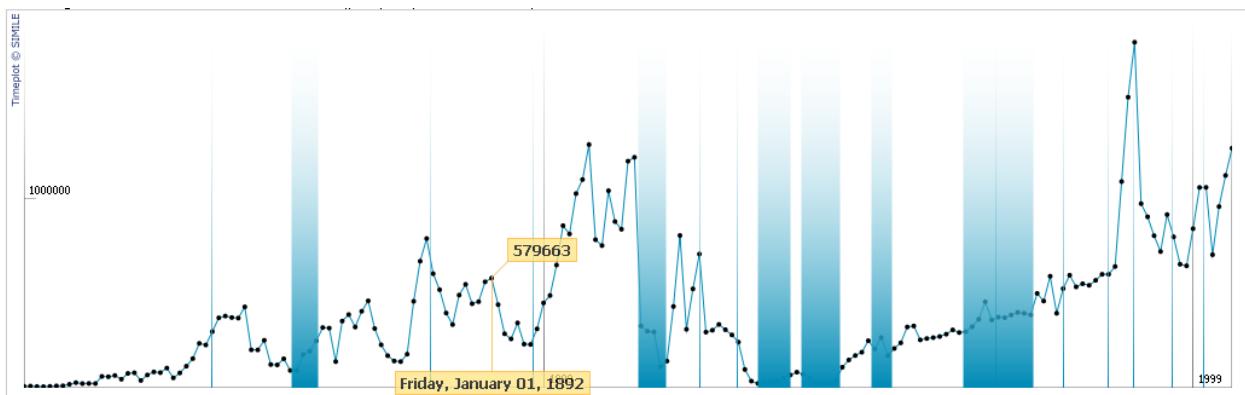
- Demonstration: SIMILE Timeplot <sup>[1]</sup>
- Java API: N/A
- JavaScript API: N/A
- Style Guide: N/A

## Employment/Purpose

Defines the plotinfo of a timeplot.

A plot layer is the main building block for timeplots and it's the object that is responsible for painting the plot itself. Each plot needs to have a time geometry, either a DataSource (for time series plots) or an EventSource (for event plots) and a value geometry in case of time series plots.

## Example



```

<zk>
 <div width="1200px">
 New Legal Permanent Residents in the U.S. (per year) vs.
 U.S.
 History
 <zscript>
 import org.zkforge.timeplot.geometry.*;
 import org.zkforge.timeplot.data.*;
 import org.zkforge.timeplot.operator.*;
 import org.zkforge.timeline.data.OccurEvent;
 import org.zkoss.zul.*;
 PlotDataSource pds=new PlotDataSource();
 pds.setDataSourceUri("misc/immigration.txt");
 pds.setSeparator(" ");
 ValueGeometry vg=new DefaultValueGeometry();
 vg.setGridColor("#000000");
 TimeGeometry tg=new DefaultTimeGeometry();
 tg.setAxisLabelsPlacement("bottom");
 </zscript>
 <timeplot width="1000px" height="300px">
 <plotinfo id="plot 1" plotDataSource="#{pds}"
 dotColor="#000000" showValues="true"
 lineColor="#008bb6"
 valueGeometry="#{vg}">
 timeGeometry="#{tg}"
 eventSourceUri="misc/us_history.xml" />
 </plotinfo>
 </timeplot>
 </div>
</zk>

```

immigration.txt

```

1820 8385
1821 9127
1822 6911
1823 6354
1824 7912
1825 10199
1826 10837
1827 18875
1828 27382
1829 22520
1830 23322

```

us\_history.xml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<data>
```

```
<event start="Jan 01 1849 00:00:00 GMT" title="California Gold Rush" link="http://en.wikipedia.org/wiki/California_Gold_Rush"></event>

<event start="Apr 12 1861 00:00:00 GMT" end="Apr 09 1865 00:00:00 GMT" title="American Civil War" link="http://en.wikipedia.org/wiki/American_Civil_War"></event>

<event start="Aug 03 1882 00:00:00 GMT" title="Immigration Act of 1882" link="http://en.wikipedia.org/wiki/1882_Immigration_Act">The U.S. Congress passes a new Immigration Act that stated that a 50 cents tax would be levied on all aliens landing at United States ports. The money collected was to be used to defray the expenses of regulating immigration and for the care of immigrants after landing. The legislation also gave powers to the authorities to deny entry to "convicts (except those convicted of political offences), lunatics, idiots and persons likely to become public charge".</event>

<event start="Apr 25 1898 00:00:00 GMT" title="Spanish-American War" link="http://en.wikipedia.org/wiki/Spanish-American_War"></event>

<event start="Jul 28 1914 00:00:00 GMT" end="Nov 11 1918 00:00:00 GMT" title="World War I" link="http://en.wikipedia.org/wiki/World_War_I"></event>

<event start="Jan 01 1924 00:00:00 GMT" title="Immigration Act of 1924" link="http://en.wikipedia.org/wiki/1924_Immigration_Act">The Immigration Act of 1924, which included the National Origins Act, Asian Exclusion Act or the Johnson-Reed Act, was a United States federal law that limited the number of immigrants who could be admitted from any country to 2% of the number of people from that country who were already living in the United States in 1890, according to the Census of 1890. It excluded immigration to the US of Asians. It superseded the 1921 Emergency Quota Act. The law was aimed at further restricting the Southern and Eastern Europeans who had begun to enter the country in large numbers beginning in the 1890s, as well as East Asians and Asian Indians, who were prohibited from immigrating entirely. It set no limits on immigration from Latin America.</event>

<event start="Oct 29 1929 00:00:00 GMT" title="Black Tuesday" link=""></event>

<event start="Jan 01 1933 00:00:00 GMT" end="Jan 01 1938 00:00:00 GMT" title="New Deal" link="http://en.wikipedia.org/wiki/New_Deal"></event>

<event start="Sep 01 1939 00:00:00 GMT" end="Sep 02 1945 00:00:00 GMT" title="World War II" link="http://en.wikipedia.org/wiki/World_War_II"></event>

<event start="Jun 25 1950 00:00:00 GMT" end="Jul 27 1953 00:00:00 GMT" title="Korean War" link="http://en.wikipedia.org/wiki/Korean_War"></event>

<event start="Aug 02 1964 00:00:00 GMT" end="Apr 30 1975 00:00:00 GMT" title="Vietnam War" link="http://en.wikipedia.org/wiki/Vietnam_War"></event>

<event start="Jul 21 1969 00:00:00 GMT" title="Man on the Moon">Neil Armstrong becomes the first person to walk on the Moon as the commander of the American mission Apollo 11.</event>

<event start="Jan 01 1980 00:00:00 GMT" title="Refugee Act" link="http://en.wikipedia.org/wiki/Refugee_Act">The Refugee Act is a 1980 United States federal law that reformed United States immigration law and admitted refugees on systematic basis for humanitarian reasons. A 1985 ceiling of 70,000 refugees, with 270,000 immigrants total and 20,000 from any one country, was established.</event>

<event start="Nov 06 1986 00:00:00 GMT" title="Immigration Reform and Control Act of 1986" link="http://en.wikipedia.org/wiki/Immigration_Reform_and_Control_Act_of_1986">The Act made it illegal to knowingly hire or recruit illegal immigrants, required employers to attest to their employees' immigration status, and granted amnesty to illegal immigrants who entered the United States before January 1, 1982 and had resided there continuously.</event>

<event start="Nov 29 1990 00:00:00 GMT" title="Immigration Act of 1990" link="http://en.wikipedia.org/wiki/Immigration_Act_of_1990"></event>

<event start="Sep 30 1996 00:00:00 GMT" title="Illegal Immigration Reform and Immigrant Responsibility Act of 1996" link="http://en.wikipedia.org/wiki/Illegal_Immigration_Reform_and_Immigrant_Responsibility_Act_of_1996"></event>

<event start="Sep 11 2001 00:00:00 GMT" title="Attack at the World Trade Center" link="http://en.wikipedia.org/wiki/Vietnam_War"></event>

</data>
```

## Supported Events

Name	Event Type
onOverPlotData	<b>Event:</b> OverPlotEvent Denotes the plot is hovered by a user.

- Inherited Supported Events: HtmlBasedComponent

## Supported Children

\*NONE

## Use Cases

Timeplot

## Version History

Version	Date	Content

## References

[1] <http://www.zkoss.org/zkdemo/reporting/timeplot>

## Essential Components

---

This section outlines components which are considered a "must have" by ZK developers.

# A

---

## A

- Demonstration: N/A
- Java API: A <sup>[1]</sup>
- JavaScript API: A <sup>[2]</sup>
- Style Guide: A

## Employment/Purpose

The same as HTML A tag.

## Properties

### Autodisable

`A.setAutodisable(java.lang.String)` <sup>[3]</sup> is used to disable an anchor automatically, when it is clicked. It is useful to prevent the user from clicking it twice (and firing redundant requests), which is common if the request takes long to serve.

The simplest use is to specify it with `self` as follows. Then, the anchor is disabled when it is clicked.

```

```

If you'd like to disable several anchors, you could specify all of them in this property by separating with a comma. For example, the following disables both anchors, when one of them is clicked.

```


```

The anchor will be enabled automatically, after the request has been served (i.e., the response has been sent back to the client). If you prefer to enable them manually (i.e., by calling `A.setDisabled(boolean)` <sup>[4]</sup> explicitly), you could prefix the ID with a plus (+). For example,

```

```

Then, you could enable them manually under the situation depending on your application's requirement, such as

```
if (something_happens) {
 ok.setDisabled(false);
 cancel.setDisabled(false);
}
```

## Enable Autodisable for All Anchors

As described in ZK Developer's Reference: Customization, you could customize ZK to enable `autodisable` for all anchors by specifying the following in the custom language addon:

```
<language-addon>
 <language-name>xul/html</language-name>
 <component>
 <component-name>a</component-name>
 <extends>a</extends>
 <property>
 <property-name>autodisable</property-name>
 <property-value>self</property-value>
 </property>
 </component>
</language-addon>
```

## File Download Link

There are several correct ways to create a file download link:

### open a new tab

specify `_blank` and your browser will produce a new tab and close immediately.

```
download
```

After clicking a link like `<a href="report.pdf" target='_self'>download</a>`, you will find ZK doesn't send AJAX requests anymore to the server.

### download attribute

A browser will prompt the user to save the linked URL instead of navigating to it.

```
<zk xmlns:c="client/attribute">
download
</zk>
```

Please refer to download attribute <sup>[5]</sup>

## Example

[Visit ZK!](#)

```

```

In addition, you could add child components to A <sup>[1]</sup> too:

```

 <grid>
 <rows>
 <row>What ever content</row>
 </rows>
```

```
</grid>

```

Notice that a child component might also handle the mouse click, so the final result of clicking on a child component is really up to which child component is used.

The href attribute can be an URI. For example,

```


```

If the URI starts with "/", ZK will encode it with the application's context path. Otherwise the path is relative to the path given by Desktop.getDirectory().

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: LabelImageElement

## Supported Children

\* ALL

## Use Cases

Version	Description	Example Location
---------	-------------	------------------

## Version History

Version	Date	Content
5.0.5	October, 2010	A [1] supports any children.
7.0.2	May, 2014	Support autodisable property for A component [6]

## References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/A.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/A.html#>
- [3] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/A.html#setAutodisable\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/A.html#setAutodisable(java.lang.String))
- [4] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/A.html#setDisabled\(boolean\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/A.html#setDisabled(boolean))
- [5] <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/a#attr-download>
- [6] <http://tracker.zkoss.org/browse/ZK-2237>

# Anchornav

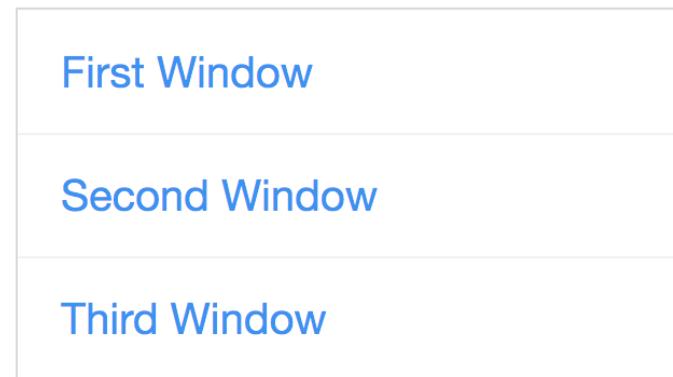
## Anchornav

- Demonstration: Anchornav: A new ZK Addon for scrolling within a page <sup>[1]</sup>
- Java API: Anchornav <sup>[2]</sup>
- JavaScript API: Anchornav <sup>[3]</sup>
- Available for ZK:
- CE PE EE

## Employment/Purpose

This component synchronizes the scrolling position on a page or within ZK containers (Div, Window, etc.) with hyperlinks and buttons. It allows you to both navigate to desired ZK components in a page and to highlight the current navigation link based on the current scroll position.

## Example



```
<zk xmlns:ca="client/attribute">
 <anchornav name="a1" width="250px">
 <listbox>
 <listitem><listcell><a ca:data-anchornav-target="$win1"> First Window </listcell></listitem>
 <listitem><listcell><a ca:data-anchornav-target="$win2"> Second Window </listcell></listitem>
 <listitem><listcell><a ca:data-anchornav-target="$win3"> Third Window </listcell></listitem>
 </listbox>
 </anchornav>
 <window id="win1" title="1. First Window">
 Hello world.
 </window>
 <window id="win2" title="2. Second Window">
 Welcome to ZK world.
 </window>
</zk>
```

By default, the Anchornav component will watch the scroll position of a page. Other scrollable containers can be associated with a named Anchornav by setting the client-attribute: ca:data-anchornav-scroll.

For example:

```
<div ca:data-anchornav-scroll="a1">
 <!-- other components -->
</div>
```

A or Button components can be used as links and targets are specified by the ZK client-attribute ca:data-anchornav-target=[selector]. For example, in the first example, Line 3 and Line 4, \$win1 / \$win2 select widgets by zk component id, win1 / win2.

Here the JQuery-based selector syntax (#domId / .class / elementName) is extended by ZK specific selectors using \$componentId or @ componentName.

## Properties

### Name

Set the name of Anchornav, it is only required when we want to watch scrolling in ZK containers, instead of watching the whole page.

The name declared on Anchornav must be used on the scrolling container with the ca:data-anchornav-scroll attribute such as:

Note: This is necessary when using a scrollbar inside a component, rather than the document-level scrollbar

```
<zk xmlns:ca="client/attribute">
 <anchornav name="a1" width="250px">
 ...
 </anchornav>
 <div id="scrollableDiv" style="overflow:auto" ca:data-anchornav-scroll="a1">
 <window id="win1" title="1. First Window">
 Hello world.
 </window>
 <window id="win2" title="2. Second Window">
 Welcome to ZK world.
 </window>
 </div>
</zk>
```

## PositionFixed

Sets whether to enable position fixed when anchornav is out of current view. When it is set to true, Anchornav will stay (float) on the same position of the page. (Default: true)

## Supported Events

- Inherited Supported Events: LabelImageElement

## Supported Children

\*ALL

## Use Cases

Version	Description	Example Location
---------	-------------	------------------

## Version History

Version	Date	Content
9.0.0	November, 2019	Anchornav <sup>[2]</sup> was introduced.

## References

[1] <https://blog.zkoss.org/2019/08/29/anchornav-a-new-zk-addon-for-scrolling-within-a-page/>

[2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Anchornav.html>

[3] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/nav/Anchornav.html>

# Applet

---

## Applet

- Demonstration: N/A
- Java API: Applet <sup>[1]</sup>
- JavaScript API: Applet <sup>[2]</sup>
- Style Guide: N/A

## Employment/Purpose

A generic applet component.

If the properties are not enough, you can use the Client-Attribute namespace to specify them.

```
<applet xmlns:ca="client/attribute"
 ca:whatever_name="whatever_value"/>
```

## Archive and Codebase

Since 5.0.3, both `archive` and `codebase` properties are encoded with the application's context path and URL rewriting, so you don't and should not have to encode it again.

## Example



```
<applet codebase="img/" code="ticker.class" msg="ZK is Simple and Rich!" width="580px" />
```

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: HtmlBasedComponent

## Supported Children

\*NONE

## Use Cases

Version	Description	Example Location

## Version History

Version	Date	Content
5.0.3	June 2010	The archive, myscript, align, hspace, and vspace properties are supported

## References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Applet.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/med/Applet.html#>

# Button

## Button

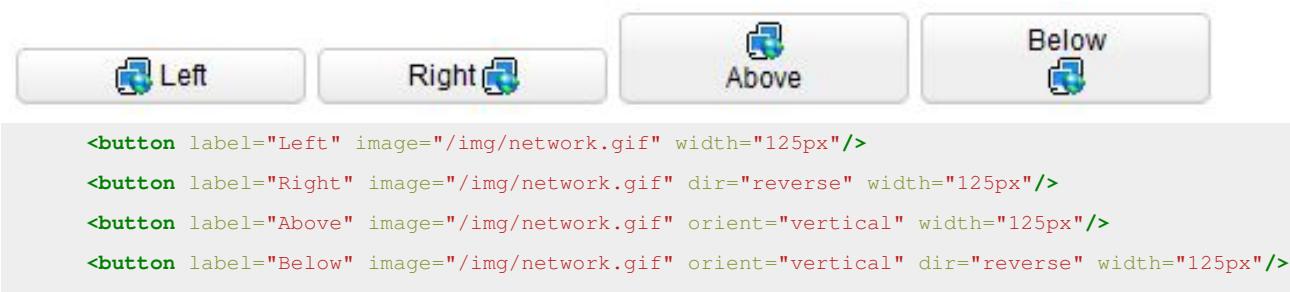
- Demonstration: Button <sup>[1]</sup> and Fileupload
- Java API: Button <sup>[2]</sup>
- JavaScript API: Button <sup>[3]</sup>
- Style Guide: Button

## Employment/Purpose

You could assign a `label` and an `image` to a button by the `label` and `image` properties. If both are specified, the `dir` property controls which is displayed up front, and the `orient` property controls whether the layout is horizontal or vertical.

Within ZK 5, the file upload has been redesigned so it can be integrated with any widget. For example, the button now can be used to upload a file. In addition to this, the display of the upload status has been enhanced and can be customized easily.

## Example



The image shows four buttons arranged horizontally. Each button has a small blue icon of a computer monitor with a network connection. The buttons are labeled "Left", "Right", "Above", and "Below".

```
<button label="Left" image="/img/network.gif" width="125px"/>
<button label="Right" image="/img/network.gif" dir="reverse" width="125px"/>
<button label="Above" image="/img/network.gif" orient="vertical" width="125px"/>
<button label="Below" image="/img/network.gif" orient="vertical" dir="reverse" width="125px"/>
```

In addition to employing URLs to specify images, you can dynamically assign a generated image to a button using the `setImageContent` method. Refer to the following section for details.

**Tip:** The `setImageContent` method is supplied by all components that have an `image` property. Simply put, `setImageContent` is used for dynamically generated images, while `image` is used for images identifiable by a URL.

## File Upload

Any button<sup>[4]</sup> can be used to upload files. All you need to do is:

1. Specify the `upload` attribute with `true`
2. Handles the `onUpload` event.

```
<button upload="true" label="Fileupload" onUpload="myProcessUpload(event.getMedia())"/>
```

When the file is uploaded, an instance of `UploadEvent`<sup>[5]</sup> is sent to the button. Then, the event listener can retrieve the uploaded content by examining the return value of `UploadEvent.getMedia()`<sup>[6]</sup>.

---

```
[1] http://www.zkoss.org/zkdemo/input/button
[2] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Button.html#
[3] http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Button.html#
[4] Any Button (http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Button.html#) can be used to upload files too.
[5] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/UploadEvent.html#
[6] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/UploadEvent.html#getMedia()
```

## Custom Error Message When Fileupload Over Maxsize

1. Write your own AuLoader

```
package test;

import
org.apache.commons.fileupload.FileUploadBase.SizeLimitExceededException;

import org.zkoss.zk.au.http.AuUploader;

public class MyUploader extends AuUploader {

 protected String handleError(Throwable ex) {

 if(ex instanceof SizeLimitExceededException) {
 SizeLimitExceededException e = (SizeLimitExceededException)
ex;
 return e.getActualSize() + " is over our limit";
 }
 return super.handleError(ex);
 }
}
```

2. Apply it in the web.xml

```
<servlet>
 <description>The asynchronous update engine for ZK</description>
 <servlet-name>auEngine</servlet-name>
 <servlet-class>org.zkoss.zk.au.http.DHtmlUpdateServlet</servlet-class>
 <init-param>
 <param-name>extension0</param-name>
 <param-value>/upload=test.MyUploader</param-value>
 </init-param>
</servlet>
```

- Result Video Demo
  - <http://screencast.com/t/MZIxSR9JzG>

## Limitation of the Default Mold

The default mold of a button uses HTML BUTTON tag to represent it visually. It is efficient, but it has some limitations:

1. The look might be different from one browser to another.
2. It doesn't support the file upload. In fact, it will become the trendy mold automatically if `upload` is specified.
3. If it is disabled it can not receive mouse events and hence can not use ZK popup component as tooltip.

If it is an issue, you could use the trendy mold instead.

```
<button label="OK" mold="trendy"/>
```

## Configure to Use the Trendy Mold as Default

If you prefer to use the trendy mold as default, you could configure ZK by adding the following to `/WEB-INF/zk.xml`

```
<library-property>
 <name>org.zkoss.zul.Button.mold</name>
 <value>trendy</value>
</library-property>
```

## File Download

Similar to file download link problem, if you specify `href` to make a button for downloading, you also need to specify `target`:

```
<button label="download" href="/myfile.pdf" target="_blank"/>
```

- If no specifying `target`, you will find event firing doesn't work anymore after you download a file.

## Properties

### Autodisable

`Button.setAutodisable(java.lang.String)` ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Button.html#setAutodisable\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Button.html#setAutodisable(java.lang.String))) is used to disable a button automatically, when it is clicked. It is useful to prevent the user from clicking it twice (and firing redundant requests), which is common if the request takes long to serve.

The simplest use is to specify it with `self` as follows. Then, the button is disabled when it is clicked.

```
<button id="ok" label="OK" autodisable="self" />
```

If you'd like to disable several buttons, you could specify all of them in this property by separating with a comma. For example, the following disables both buttons, when one of them is clicked.

```
<button id="ok" label="OK" autodisable="ok, cancel" />
<button id="cancel" label="Cancel" autodisable="ok, cancel" />
```

The button will be enabled automatically, after the request has been served (i.e., the response has been sent back to the client). If you prefer to enable them manually (i.e., by calling `Button.setDisabled(boolean)` ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Button.html#setDisabled\(boolean\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Button.html#setDisabled(boolean))) explicitly), you could prefix the ID with a plus (+). For example,

```
<button id="ok" label="OK" autodisable="+self, +cancel" />
```

Then, you could enable them manually under the situation depending on your application's requirement, such as

```
if (something_happens) {
 ok.setDisabled(false);
 cancel.setDisabled(false);
}
```

## Enable Autodisable for All Buttons

As described in ZK Developer's Reference: Customization, you could customize ZK to enable `autodisable` for all button by specifying the following in the custom language addon:

```
<language-addon>
 <component>
 <component-name>button</component-name>
 <extends>button</extends>
 <property>
 <property-name>autodisable</property-name>
 <property-value>self</property-value>
 </property>
 </component>
</language-addon>
```

## Href

In addition to handling the `onClick` event, you could specify the URL in the `href` property (`Button.setHref(java.lang.String)` ([`http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Button.html#setHref\(java.lang.String\)`](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Button.html#setHref(java.lang.String)))), such that the browser will navigate to the URL you specified directly (without sending back any request to the server). If you prefer to visit the URL in another browser window, you could specify the name in `Button.setTarget(java.lang.String)` ([`http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Button.html#setTarget\(java.lang.String\)`](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Button.html#setTarget(java.lang.String))) (just like using a HTML A tag).

Notice that the end user could hold the `Control` key and click on the button to visit the link in a new browser window (like a HTML A tag does).

## Href and the onClick Event

There are two ways to add behavior to a `button` and `toolbarbutton`. Firstly, you can specify a listener to the `onClick` event. Secondly, you could specify a URL for the `href` property (`Button.setHref(java.lang.String)` ([`http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Button.html#setHref\(java.lang.String\)`](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Button.html#setHref(java.lang.String)))). If both are specified, the `href` property has the higher priority, i.e., the `onClick` event won't be sent.

```
<zk>
 <window title="example">
 <button label="click me" onClick="do_something_in_Java()" />
 <button label="don't click that one, click me" href="/another_page.zul"/>
 </window>
</zk>
```

## Href and SendRedirect

The href property is processed at the client. In other words, the browser will jump to the URL specified in the href property, so your application running at the server has no chance to process it.

If you have to process it at the server or you have to decide whether to jump to another URL based on certain condition, you could listen to the onClick event, process it, and then invoke Executions.sendRedirect(java.lang.String) ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/Executions.html#sendRedirect\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/Executions.html#sendRedirect(java.lang.String))) if it shall jump to another URL.

For end users, there is no difference between the use of Button.setHref(java.lang.String) ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Button.html#setHref\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Button.html#setHref(java.lang.String))) and Executions.sendRedirect(java.lang.String) ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/Executions.html#sendRedirect\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/Executions.html#sendRedirect(java.lang.String))).

```
<zk>
 <window>
 <button label="redirect" onClick="Executions.sendRedirect ("another.zul")" />
 <button label="href" href="another.zul"/>
 </window>
</zk>
```

Since the onClick event is sent to the server for processing, you are able to perform additional tasks before invoking Executions.sendRedirect(java.lang.String) ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/Executions.html#sendRedirect\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/Executions.html#sendRedirect(java.lang.String))), such as redirecting to another page only if certain conditions are satisfied.

On the other hand, the href property is processed at the client side. Your application won't be notified when users click the button.

## Type

[Since 5.0.4]

Button.setType(java.lang.String) ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Button.html#setType\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Button.html#setType(java.lang.String))) sets the button's type. It is designed to work with the HTML FORM tag. You rarely need it unless you want to work with HTML FORM and Servlets. For example,

```
<n:form action="/foo/my_handler" xmlns:n="native">
 <textbox/>
 <button type="submit" label="Submit"/>
 <button type="reset" label="Reset"/>
</n:form>
```

## Upload

By specifying the upload property ( `Button.setUpload(java.lang.String)` ([`http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Button.html#setUpload\(java.lang.String\)`](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Button.html#setUpload(java.lang.String)))), you could make a button used for uploading files. For example,

```
<button label="Upload" upload="true" onUpload="handle(event.media)" />
```

Once the file(s) are uploaded, the `onUpload` event will be sent with an instance of `UploadEvent` ([`http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/UploadEvent.html#`](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/UploadEvent.html#)). And, you could retrieve the uploaded files from `UploadEvent.getMedia()` ([`http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/UploadEvent.html#getMedia\(\)`](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/UploadEvent.html#getMedia())) and `UploadEvent.getMedias()` ([`http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/UploadEvent.html#getMedias\(\)`](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/UploadEvent.html#getMedias()))

If you want to customize the handling of the file upload at the client, you can specify a JavaScript class when calling this method:

```
<button upload="foo.Upload"/> <!-- assume you implement a JavaScript class: foo.Upload -->
```

## Options

Another options for the upload can be specified as follows:

```
<button label="Upload"
upload="true,maxsize=-1,multiple=true,accept=audio/*|video/*|image/*|MIME_type,
native"/>
```

- `maxsize`: the maximal allowed upload size of the component, in kilobytes, or a negative value if no limit.
- `native`: treating the uploaded file(s) as binary, i.e., not to convert it to image, audio or text files.
- `multiple`: treating the file chooser allows multiple files to upload, the setting only works with HTML5 supported browsers (since ZK 6.0.0)
- `accept`: specifies the MIME types of files that the server accepts, the setting only works with HTML5 supported browsers (since ZK 7.0.0). MIME type list ([`http://www.iana.org/assignments/media-types/media-types.xhtml`](http://www.iana.org/assignments/media-types/media-types.xhtml)).

## Customize Upload Size Exceeding Message

If you want to customize the message about the file size exceeding max size, please refer to ZK Developer's Reference/Internationalization.

For Example, (in WEB-INF/zk-label.properties)

```
MZul.2105=The request was rejected because its size {{0}} exceeds the
configured maximum {{1}}
```

Notice that you can change the index `{0}` & `{1}` to choose the file size unit. (Auto:{0},{1} Byte:{2},{3} KB:{4},{5} MB:{6},{7})

## Inherited Functions

Please refer to LabelImageElement for inherited functions.

## Supported Events

Name	Event Type
onFocus	<b>Event:</b> Event ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/Event.html#">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/Event.html#</a> ) Denotes when a component gets the focus.
onBlur	<b>Event:</b> Event ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/Event.html#">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/Event.html#</a> ) Denotes when a component loses the focus.
onUpload	<b>Event:</b> UploadEvent ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/UploadEvent.html#">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/UploadEvent.html#</a> ) Denotes user has uploaded a file to the component.

- Inherited Supported Events: LabelImageElement

## Supported Molds

Available molds of a component are defined in lang.xml embedded in zul.jar.

Name	Snapshot
default	<a href="#">Demo</a>
trendy	<a href="#">Demo</a>
os	<a href="#">Demo</a>

## Supported Children

\*NONE

## Use Cases

Version	Description	Example Location
3.6	Get dynamically generated Button reference in onClick Event	( <a href="http://www.zkoss.org/forum/listComment/8780">http://www.zkoss.org/forum/listComment/8780</a> ) ( <a href="http://www.zkoss.org/forum/listComment/8780">http://www.zkoss.org/forum/listComment/8780</a> )
3.6	How to fire onClick Event on a Button	( <a href="http://www.zkoss.org/forum/listComment/1716">http://www.zkoss.org/forum/listComment/1716</a> ) ( <a href="http://www.zkoss.org/forum/listComment/1716">http://www.zkoss.org/forum/listComment/1716</a> )

## Version History

Version	Date	Content
5.0.4	August 2010	Button.setType(java.lang.String) ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Button.html#setType(java.lang.String)">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Button.html#setType(java.lang.String)</a> ) was introduced to allow a button able to submit or reset a form. <pre>&lt;n:form action="a_uri" xmlns:n="native"&gt;   &lt;button type="submit" label="Submit"/&gt;   &lt;button type="reset" label="Reset"/&gt; &lt;/n:form&gt;</pre>
6.0.0	December 2011	Button.setUpload(java.lang.String) ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Button.html#setUpload(java.lang.String)">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Button.html#setUpload(java.lang.String)</a> ) the <i>multiple</i> setting was introduced to allow to choose multiple files to upload at the same time. (HTML5 supported browsers only) <pre>&lt;button upload="true,multiple=true" label="Fileupload"/&gt;</pre>
7.0.0	September 2013	Button.setUpload(java.lang.String) ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Button.html#setUpload(java.lang.String)">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Button.html#setUpload(java.lang.String)</a> ) the <i>accept</i> setting was introduced to allow to specify the types of files that the server accepts. (HTML5 supported browsers only) <pre>&lt;button upload="true,accept=audio/* video/* image/* MIME_type" label="Fileupload"/&gt;</pre>

## Captcha

### Captcha

- Demonstration: Capcha <sup>[1]</sup>
- Java API: CapTCHA <sup>[2]</sup>
- JavaScript API: CapTCHA <sup>[3]</sup>
- Style Guide: N/A
- Available in ZK PE and EE only <sup>[13]</sup>

### Employment/Purpose

A `captcha` component can generate a special distortion image, also called a CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) image. Developers could set `height` and `width` for dimension of captcha. By default, captcha render the image with a randomly generated text, and developers can set `value` to assign a purposive text.

### Example



```
<vbox>
 <captcha id="cpa" length="5" width="200px" height="50px"/>
</vbox>
```

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: Image

## Supported Children

\*NONE

## Use Cases

Version	Description	Example Location
---------	-------------	------------------

## Troubleshooting

### Linux

Captcha depends on Java Swing that might not work under some version of JVM. For the information to make it work under Linux, please refer to ZK Installation Guide: Linux.

## Version History

Version	Date	Content
---------	------	---------

## References

- [1] [http://www.zkoss.org/zkdemo/input/form\\_sample](http://www.zkoss.org/zkdemo/input/form_sample)
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Captcha.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Captcha.html#>

# Combobutton

## Combobutton

- Demonstration:
- Java API: Combobutton [1]
- JavaScript API: Combobutton [2]
- Style Guide: Combobutton

## Employment/Purpose

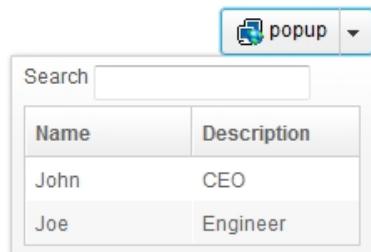
A Combobutton is a special Button that embeds a popup or menupopup child.

You could assign a `label` and an `image` to a Combobutton by the `label` and `image` properties. If both are specified, the `dir` property control which is displayed up front, and the `orient` property controls whether the layout is horizontal or vertical, the `autodrop` property control whether the child popup/menupopup open while mouseover and close while mouseout the right side drop down icon of Combobutton automatically.

When the user clicks the drop down icon of Combobutton, the child popup/menupopup of the Combobutton will be displayed.

## Example

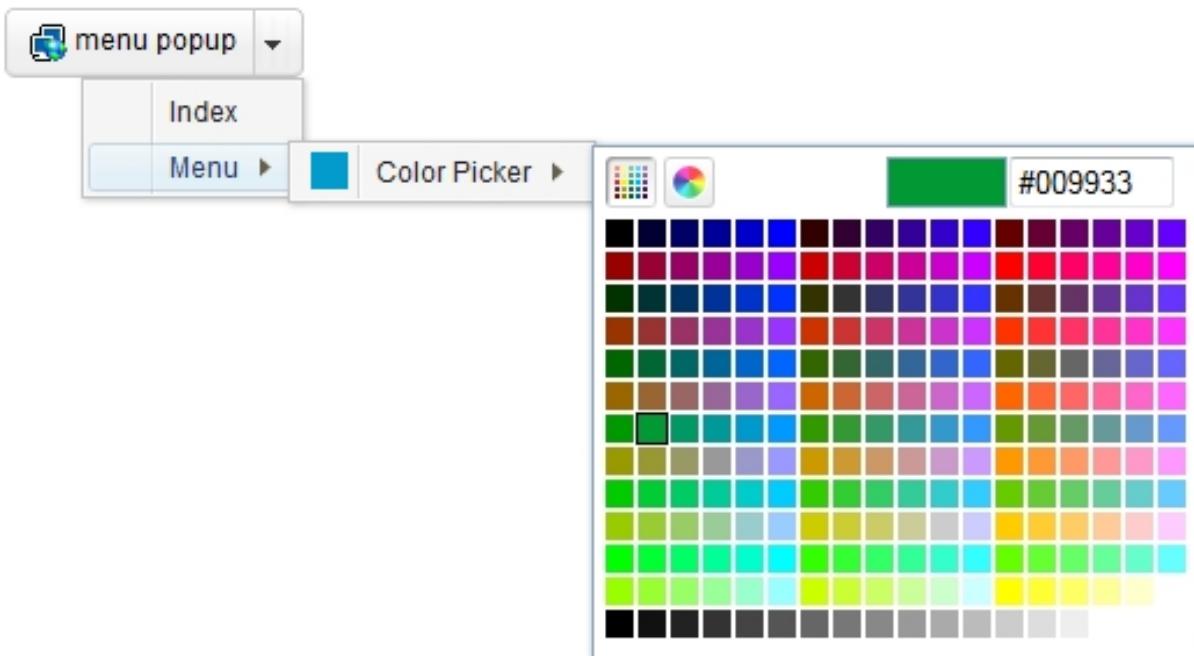
- Combobutton with Popup



```
<combobutton label="popup" image="/img/network.gif">
 <popup>
 <vbox>
 <hbox>
 Search
 <textbox />
 </hbox>
 <listbox width="200px">
 <listhead>
 <listheader label="Name" />
 <listheader label="Description" />
 </listhead>
 <listitem>
 <listcell label="John" />
 <listcell label="CEO" />
 </listitem>
 <listitem>
```

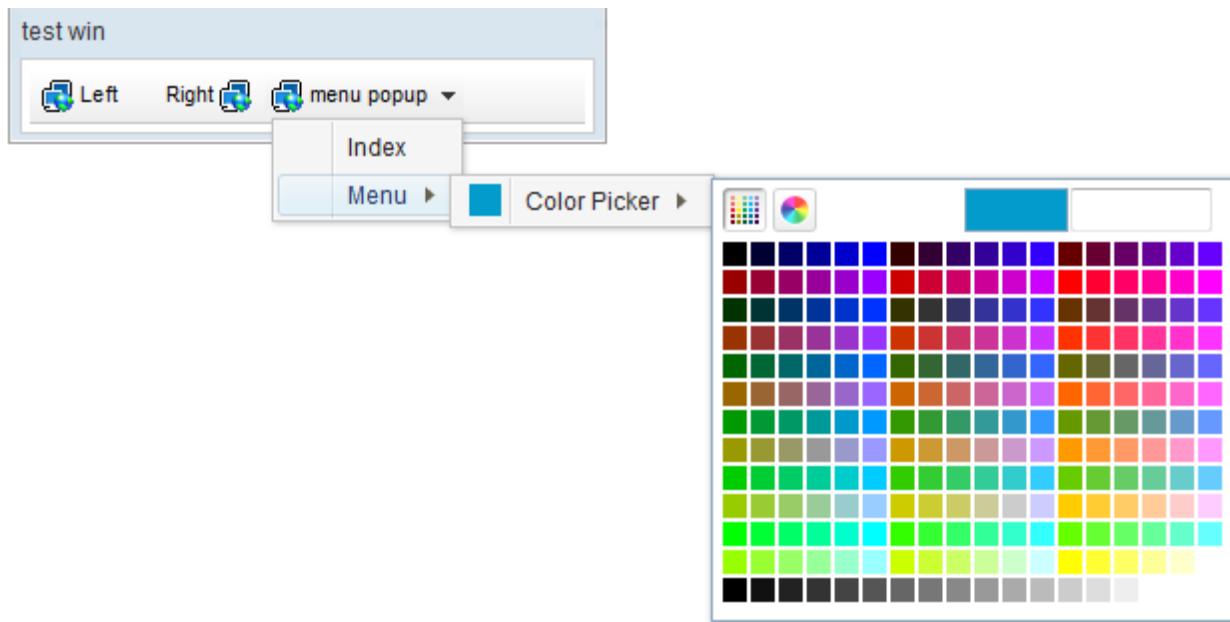
```
 <listcell label="Joe" />
 <listcell label="Engineer" />
 </listitem>
</listbox>
</vbox>
</popup>
</combobutton>
```

- Combobutton with Menupopup



```
<combobutton label="menu popup" image="/img/network.gif">
<menupopup>
 <menuitem label="Index"/>
 <menu label="Menu">
 <menupopup>
 <menu label="Color Picker" content="#color=#029BCB" />
 </menupopup>
 </menu>
</menupopup>
</combobutton>
```

- Combobutton as Toolbarbutton



```
<zk>
<window border="normal" title="test win"
 width="300px">
 <toolbar>
 <toolbarbutton label="Left" image="/img/network.gif" />
 <space />
 <toolbarbutton label="Right" image="/img/network.gif"
 dir="reverse" />
 <combobutton label="menu popup" image="/img/network.gif"
 mold="tbbtn">
 <menupopup>
 <menuitem label="Index"/>
 <menu label="Menu">
 <menupopup>
 <menu label="Color Picker" content="#color=#029BCB" />
 </menupopup>
 </menu>
 </menupopup>
 </combobutton>
 </toolbar>
</window>
</zk>
```

## Properties

### Autodrop

Combobutton.setAutodrop(boolean)<sup>[3]</sup> is used to set whether the child popup should drop down automatically while mouseover the right drop down icon of Combobutton.

The simplest use is to specify it with `self` as follows. Then, the button is disabled when it is clicked.

```
<combobutton label="popup" autodrop="true" />
```

Moreover, it support other properties inherited from Button in stead of upload.

## Supported Events

Name	Event Type
onClick	<b>Event:</b> Event <sup>[7]</sup> Denotes when left button of Combobutton is clicked.
onOpen	<b>Event:</b> Event <sup>[7]</sup> Denotes when the child popup is opened or closed, it will not be fired if open or close child popup by call Combobutton.setOpen(boolean) <sup>[4]</sup> directly.

- Inherited Supported Events: Button

## Supported Molds

- The default mold
- The tbbtn mold

since 6.5.0

The **tbbtn** mold is renamed to **toolbar** mold

## Supported Children

Popup Menupopup

## Use Cases

Version	Description	Example Location
6.0.0+	Combobutton with Colorbox	blog post <sup>[5]</sup>

## Version History

Version	Date	Content
6.5.0	September 2012	The <b>tbbtn</b> mold is renamed to <b>toolbar</b> mold

## References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Combobutton.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Combobutton.html#>
- [3] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Combobutton.html#setAutodrop\(boolean\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Combobutton.html#setAutodrop(boolean))
- [4] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Combobutton.html#setOpen\(boolean\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Combobutton.html#setOpen(boolean))
- [5] <http://blog.zkoss.org/index.php/tag/combobutton/>

# Dropupload

---

## Dropupload

- Demonstration: N/A
- Java API: Dropupload <sup>[1]</sup>
- JavaScript API: Dropupload <sup>[2]</sup>
- Style Guide: N/A
- Available in ZK EE only <sup>[13]</sup>

## Employment/Purpose

Dropupload leverages HTML 5 technology to handle file uploading where users can simply drag and drop the file(s) they want to upload into Dropupload and the uploading process will start automatically. The behaviour and operation of this Dropupload component is similar to ZK's **file upload button** <sup>[3]</sup> but with better user experience and performance.

## Example

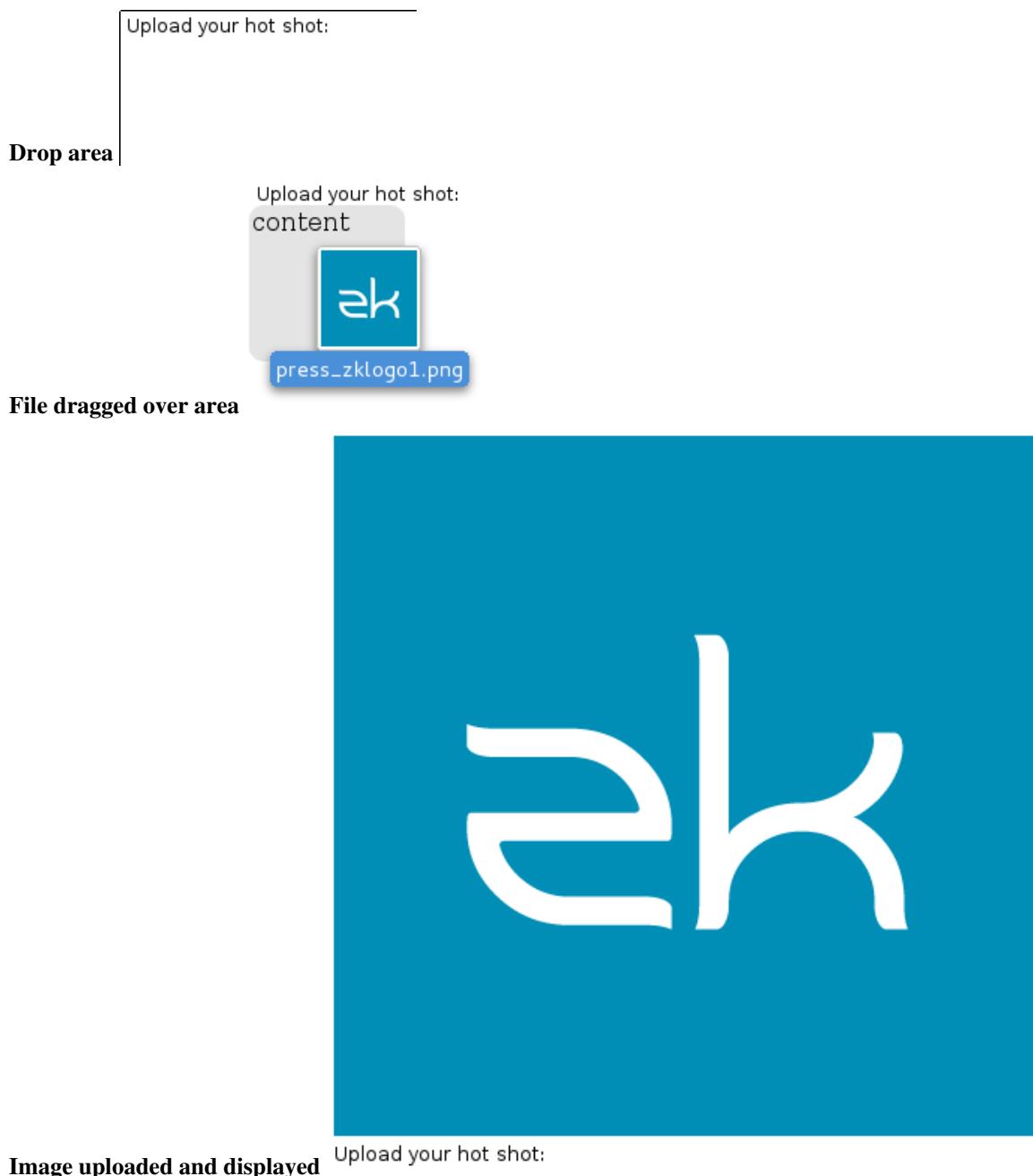
Following is a typical example of its implementation, it will always show component and limit the upload file size.

```
<dropupload maxsize="5120" detection="none" onUpload="doSomething(event)">
 <attribute name="content"><! [CDATA[
 Drop Here

 size < 5MB
]]></attribute>
</dropupload>
```

Another example, it will detect the drag action:

```
<zk>
 <vlayout>
 <image id="img" />
 Upload your hot shot:
 <dropupload maxsize="-1" content="content" detection="browser" onUpload="img.setContent(event.media)" />
 </vlayout>
</zk>
```



## Maxsize

The `maxsize` attribute is used for limiting the file size of a single file in which users are allowed to upload. Users are allowed to drag in two or more files at once but each of them has to be smaller than the size set by `Maxsize`. If one of the files is larger than the size set by `Maxsize`, an error message will occur and nothing will be uploaded.

For example, in the case of the previous sample code, you can upload multiple files, say, four files that are smaller than 5120KB at once but if one of them exceeds 5120KB, then an exception will occur and none of the four files will be uploaded to the server.

The unit of `MaxsizeM` attribute is in KB. If it is not assigned a value, it will use the value of `Configuration.getMaxUploadSize()` automatically while a negative value would mean that the file size is set as unlimited.

## Detection

This attribute will define what users see when they drag and drop files into the application i.e. how the Dropupload component and its content will appear according to their action.

There are four valid values of detection :

- `none` : Ignore users' drag action, always show Dropupload and its content.
- `browser` (default setting) : Dropupload is not visible in the application initially but shows up along with the content when users drag files into the browser.
- `self` : Dropupload is visible in the application initially but the content only appears when users drag files into the component.
- `id` of another component : Behaviour of this value is almost identical to `self`, except that the trigger area is inside the component of the appointed id.

The `content` value can be any HTML string and remember to surround the content value by `CDATA` block .

**Note :** A Dropupload with `detection="browser"` cannot be used with another Dropupload component that has a different detection value; users won't be able to drag a file into the component successfully.

## Anchor

since 7.0.2

This attribute allows the dropupload component to anchor to another component and overlay that component when the user drag & drops files to the browser. Much like how Gmail works when dropping attachments to emails.

## Example



```
<zk>
 <div height="100px"></div>
 <tabbox height="100px">
 <tabs>
 <tab id="A" label="Tab A" />
```

```

<tab id="B" label="Tab B" />
</tabs>
<tabpanels id="tps">
 <tabpanel>This is panel A</tabpanel>
 <tabpanel>This is panel B</tabpanel>
</tabpanels>
</tabbox>
<dropupload anchor="\${tps}"></dropupload>
</zk>

```

## MaxFileCount

Default: -1

Set the maximum number of files a user can upload at once, -1 means no limit. When the number of upload files exceeds the max file count, nothing will be uploaded and `onMaxFileCountExceed` event will be fired, developers can listen to that event and get the number of uploaded files by calling `event.getData()`.

For example (MVVM pattern):

```

<dropupload onMaxFileCountExceed="@command('maxFileCountExceed', filesCount=event.data)" />

@Command
public void maxFileCountExceed(@BindingParam("filesCount") Integer
filesCount) {
 Messagebox.show(filesCount + " files exceed the number of upload
files limitation.");
}

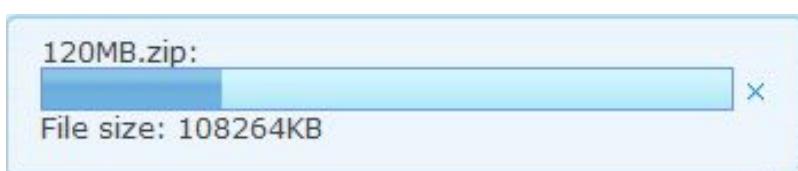
```

## Do not Convert File

By default, ZK will convert upload file to image, audio and text file if possible. Developer can use `native="true"` to demand ZK don't convert file.

## Customized File Viewer

Similar to **file upload button**<sup>[3]</sup>, the default file viewer will show the uploading progress via a pop-up bar as illustrated below.



Alternatively, developers can also design customized File Viewer by implementing a JavaScript class to handle the display screen when uploading files. Below is an example of a customized file viewer where the progress bar is shown at the bottom of the browser.

A screenshot of a customized file upload progress dialog. It shows the file name 'FileName: 120MB.zip Cancel' at the top left. To its right, the text 'File size: : 6496KB of 108264KB' is displayed. Below these, there is some JavaScript code defining a class 'foo.MyFileViewer'.

```

foo.MyFileViewer = zk.$extends(zk.Object, {
 updated: null,
}

```

```

$init: function (uplder, file) {
 this._uplder = uplder;
 var id = uplder.id,
 uri = zk.ajaxURI('/web/zk/img/progress2.gif',
{au:true}),
 html = '<div id="' + id + '" class="viewer"><image class="float-left" src="' + uri + '"/>' +
 '+ '<div class="float-left">fileName: ' + file.name
 + 'Cancel</div><div class="float-right">' +
 + msgzk.FILE_SIZE + ': 0 of '
 + '0</div><div class="clear"></div></div>';

 jq(uplder.getWidget().getPage()).append(html);

 this.viewer = jq('#'+ id)[0];
 jq('#' + id + '-cancel').click(function() {
 uplder.cancel();
 });
},
update: function (sent, total) {
 jq('#'+ this._uplder.id +
'-sent').html(Math.round(sent/1000) + msgzk.KBYTES);
 if (!this.updated) {
 this.updated = true;
 jq('#'+ this._uplder.id +
'-total').html(Math.round(total/1024)+msgzk.KBYTES);
 }
},
destroy: function () {
 jq(this.viewer).remove();
}
});

```

In the code snippet above, you can see that there are three functions - *\$init*, *update*, and *destroy*.

1. **\$init(uplder, file)**: When the user selects a file from the file chooser, this function will be invoked.
  - *uplder*: An uploader object
  - *file*: The file user uploads. It is a File [4] object.
2. **update(send, total)**: After the uploading engine receives the size that has already been uploaded, this function will be invoked.
  - *sent*: An integer of the uploaded size.
  - *total*: An integer of the total uploaded size.
3. **destroy()**: After the file has been uploaded or if the uploading has been canceled or if the uploading has caused an error, this function will be invoked.

After customizing your JavaScript class which in this case is `foo.MyFileViewer`, assign it to Dropupload using the `viewerClass` attribute as demonstrated below:

```
<dropupload viewClass="foo.MyFileViewer" content="custom viewer" detection="none" />
```

## Uploader

Below is a summarised description table of the *Uploader* when passed a selected file from the user.

Method	Usage
getWidget	Indicate which component the widget belongs to
cancel	Stops the uploading process.

## Transforming the original File Viewer

Customized File Viewers written in the past can also be applied to Dropupload with only some slight changes :

- Originally, the second parameter of \$init() is filenm (type: String). To apply it to the new Dropupload component, change the second parameter to file (type: File) object and add another line of filenm = file.name to solve the issue.

```
//before
$init: function (uplder, filenm) {
 //routine
}

//after
$init: function (uplder, file) {
 var filenm = file.name;
 //routine
}
```

- The first parameter of update(), send would originally pass an integer value ranging from 0 to 100, representing the percentage of the uploading process whereas now it will pass the value of the already uploaded size of data in Bytes.

## Customize Upload Size Exceeding Message

Please refer to ZK\_Component\_Reference/Essential\_Components/Button#Customize\_Upload\_Size\_Exceeding\_Message

## Event For Completed Uploads

After the upload is finished, the uploaded files can be retrieved from the companion event, which is an instance of UploadEvent [5]. For example,

```
<zscript><![CDATA[
public void showFileName(org.zkoss.zk.ui.event.UploadEvent event) {
 org.zkoss.util.media.Media[] medias = event.getMedias();
 StringBuffer sb = new StringBuffer();
 for (org.zkoss.util.media.Media m : medias) {
 sb.append(m.getName() + "\n");
 }
 Messagebox.show(sb.toString());
```

```

}
]]></zscript>
<dropupload detection="none" onUpload="showFileName(event)" />

```

## Browser Support

As Dropupload leverages HTML5 technology, some browsers don't support it. Currently, it works normally on Firefox (v.13), Chrome (v.19) and Safari (v.5.1.x), but doesn't function in IE 9, Opera v.11.x, and Microsoft Edge.

Moreover, the `detection` setting cannot be displayed on some older machines.

## Supported Events

Name	Event Type
onUpload	<b>Event:</b> UploadEvent <sup>[5]</sup> This event will be triggered once a user has uploaded a file.
onMaxFileCountExceed	<b>Event:</b> Event <sup>[7]</sup> This event will be triggered when number of upload files exceed the maxFileCount.

- Inherited Supported Events: LabelImageElement

## Supported Children

\*NONE

## Use Cases

Version	Description	Example Location

## Version History

Version	Date	Content
6.5.0	June, 2012	Dropupload <sup>[1]</sup> was introduced.
7.0.2	March, 2014	ZK-2207 <sup>[5]</sup> : Dropupload support anchor

## References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Dropupload.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/wgt/Dropupload.html#>
- [3] [http://books.zkoss.org/wiki/ZK\\_Component\\_Reference/Essential\\_Components/Button#File\\_Upload](http://books.zkoss.org/wiki/ZK_Component_Reference/Essential_Components/Button#File_Upload)
- [4] <http://www.w3.org/TR/FileAPI/>
- [5] <http://tracker.zkoss.org/browse/ZK-2207>

# Filedownload

## Filedownload

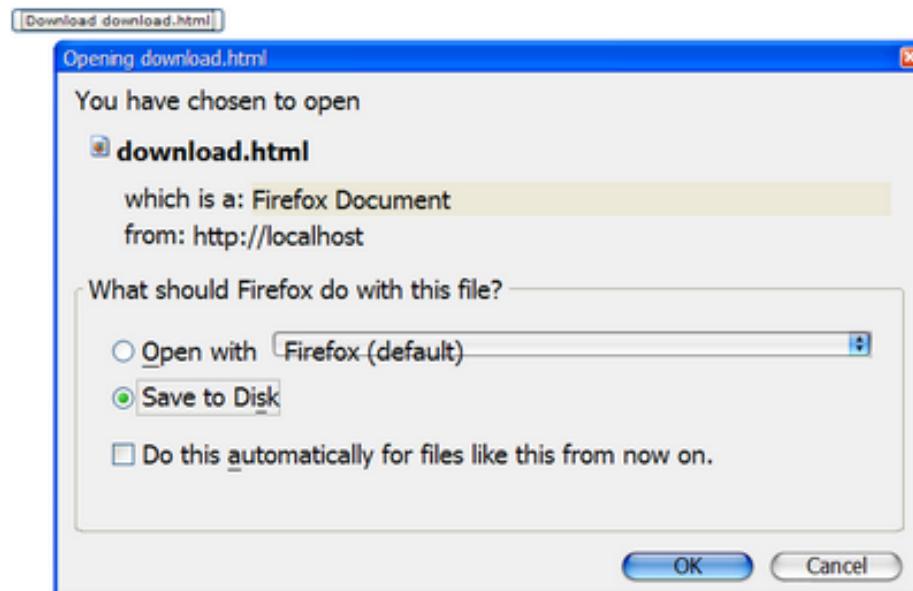
- Demonstration: File Download [1]
- Java API: Filedownload [2]
- JavaScript API: N/A
- Style Guide: N/A

## Employment/Purpose

Filedownload [2] provides a set of utilities to prompt a user for downloading a file from the server to the client.

Notice that Filedownload [2] is not a component. Rather, it is a collection of utilities for file download.

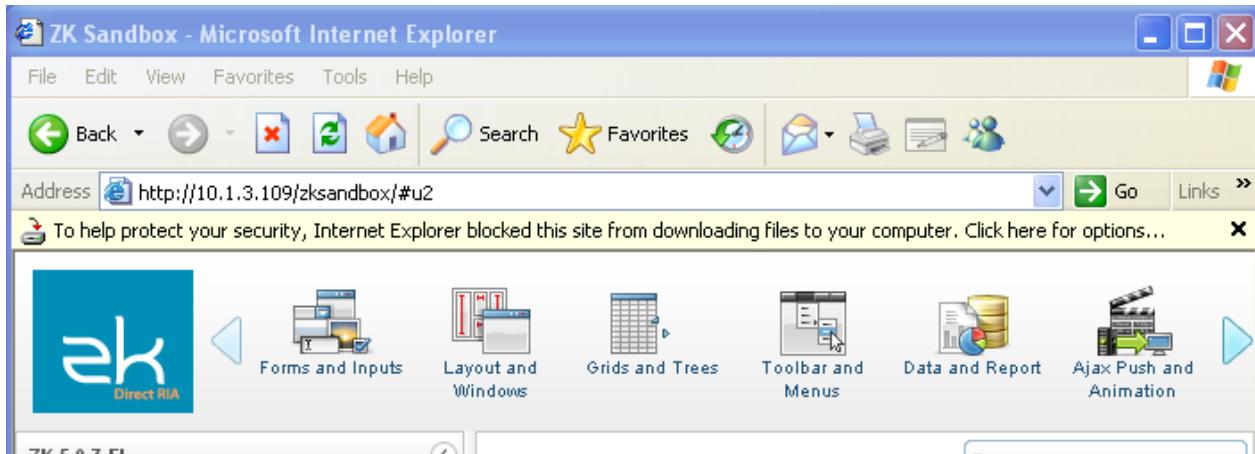
Unlike the `iframe` component that displays the file in the browser window, a file download dialog is shown at the browser if one of the `save` methods is called. Then, the user can specify the location in his local file system to save the file.



```
<button label="Download">
 <attribute name="onClick">
 java.io.InputStream is =
 desktop.getWebApp().getResourceAsStream("/test/download.html");
 if (is != null)
 Filedownload.save(is, "text/html", "download.html");
 else
 alert("/test/download.html not found");
 </attribute>
</button>
```

## Limitation of IE 6/7/8

With this approach (Filedownload<sup>[2]</sup>), Internet Explorer 6, 7 and 8<sup>[3]</sup> will show up an warning message on top of the browser as snapshot below ("To help protect your security, Internet Explorer...").



If the user allows the download (right click and select Download File...), IE will eventually reload the page. It means the whole content of the page is reloaded and the downloading does not take place. And, the user have to click the download button or link again. It is a tedious user experience.

To work around it, we have to prepare another page for real download, and then use FORM submit, instead of invoking Filedownload<sup>[2]</sup>, to redirect to the page for real download. For example,

```
<!-- download.zul: the page that guides users to download -->
<h:form xmlns:h="native" action="real-download.jsp" target="_blank"> <!-- a form -->
 <button label="Download" type="submit"/> <!-- use a submit button -->
</h:form>
```

And, the page for real download could be implemented with, say, JSP or a servlet. ZK provides utilities to simplify the task: javax.servlet.http.HttpServletResponse, org.zkoss.util.media.Media, boolean, boolean) Https.write(javax.servlet.http.HttpServletRequest, javax.servlet.http.HttpServletResponse, org.zkoss.util.media.Media, boolean, boolean)<sup>[4]</sup> and AMedia<sup>[5]</sup>. For example,

```
<%-- real-download.jsp: the page really downloads the file to response --%>

<%@ page import="org.zkoss.web.servlet.http.Https, org.zkoss.util.media.AMedia" %>
<%
 Https.write(request, response,
 new AMedia("B1896797.pdf", null, null,
 application.getResourceAsStream("/test2/B1896797.pdf")),
 true/*download*/, false);
%>
```

If the user could input more information to select the file to download, we could enhance download.zul in the above example by adding more input components inside HTML FORM. For example,

```
<h:form xmlns:h="native" action="real-download.jsp" target="_blank">
 <datebox name="when"/>
 <button label="Download" type="submit"/>
</h:form>
```

Notice we have to specify the name property such that its value will be sent with the given name. For more information of using HTML FORM, please refer to ZK Developer's Reference/Integration/Use ZK in JSP#HTML\_Form:ZK Developer's Reference: HTML Form.

## Side Effect: Chrome and Safari

This HTML FORM approach works fine under all Internet Explorer and Firefox. Unfortunately, it has another unpleasant side effect under Chrome and Safari: it opens additional blank browser window (so the user has to close it manually). Thus, it is better to detect the browser first and apply the HTML Form approach only if it is Internet Explorer.

```
<?taglib uri="http://www.zkoss.org/dsp/web/core" prefix="c"?>
<h:form xmlns:h="native" action="real-download.jsp" target="_blank"
if="${c:browser('ie')}">
...

```

- 
- [1] [http://www.zkoss.org/zkdemo/file\\_handling/file\\_download](http://www.zkoss.org/zkdemo/file_handling/file_download)
  - [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Filedownload.html#>
  - [3] Internet Explorer 9 and other browsers all work fine without this limitation.
  - [4] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/web/servlet/http/Https.html#write\(javax.servlet.http.HttpServletRequest,java.io.OutputStream\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/web/servlet/http/Https.html#write(javax.servlet.http.HttpServletRequest,java.io.OutputStream))
  - [5] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/util/media/AMedia.html#>

## The Resumable Download

- Available for ZK:
- CE PE EE

In certain situations, you might prefer to generate an URL that can be used even if the desktop becomes invalid. For example, you want to allow users to use a download manager (such as wget and DownThemAll). Another example is related to the blocking feature found in some browsers -- which confirm the download with the user and causes the page to reload (and then the previous desktop is lost).

To solve this, you have to use the so-called resumable download. By resumable we mean the user can bookmark the URL and download it later (and even resume the download in the middle). On the other hand, the download URL of the save method becomes obsolete as soon as the desktop (or session) is gone.

To use resumable download, you have to invoke the saveResumable method of Filedownload (<http://www.zkoss.org/javadoc/latest/zk/org/zkmax/zul/Filedownload.html#>) instead of save as depicted below:

```
<window title="Save Resumable" border="normal">
 <button label="download"
 onClick='Filedownload.saveResumable("foo.txt", "text/plain",
null)' />
</window>
```

Then, the URL generated by saveResumable can be copied to the download manager the user prefers.

## Control Resumable Download

Since the resumable download can be used in any session or without any session, or with a different client (such flashget), you might want to limit the download under certain condition. There are two library properties that can control the number of allowed resumable downloads.

- `org.zkoss.zk.download.resumable.lifetime`
  - Specifies when the download URL will be expired (unit: second).
  - Default: 14400 (i.e., 4 hours).
- `org.zkoss.zk.download.resumable.maxsize`

Specifies the maximal allowed number of resumable downloads.

(( Default: 4096.

If you want more advanced control, you can implement `FiledownloadListener` (<http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/FiledownloadListener.html#>) and specify it in a library property called `org.zkoss.zkmax.zul.FiledownloadListener.class`. For example, in `zk.xml`, you can do:

```
<library-property>
 <name>org.zkoss.zkmax.zul.FiledownloadListener.class</name>
 <value>com.foo.MyDownloadListener</value>
</library-property>
```

## Handling download target and window unload

Triggering a file download in the main context of a ZK page can cause the client-engine to terminate while the page is still open.

Refer to the Developer Reference guide for in-depth details.

## Version History

Version	Date	Content
---------	------	---------

# Fileupload

## Fileupload

- Demonstration: File Upload <sup>[1]</sup>
- Java API: Fileupload <sup>[2]</sup>
- JavaScript API: Fileupload <sup>[3]</sup>
- Style Guide: N/A

## Employment/Purpose

There are two ways to use Fileupload <sup>[2]</sup> as a component to upload files, or invoke Fileupload.get() <sup>[4]</sup> to open a dialog to upload files.

## Use as a Component

Fileupload <sup>[2]</sup> itself is a component. You can use it directly as follows.

```
<fileupload label="Upload">
 <attribute name="onUpload">
 org.zkoss.util.media.Media media = event.getMedia();
 //then, you can process media here
 </attribute>
</fileupload>
```

Fileupload <sup>[2]</sup> is actually a button with upload=true. In other words, the above is equivalent to

```
<button label="Upload" upload="true">
 ...
```

Please refer to Button: Upload for details.

## Invoke the Static Method: get

Fileupload provides a set of static methods to simplify file uploading, such as Fileupload.get() <sup>[4]</sup>, java.lang.String) Fileupload.get(java.lang.String, java.lang.String) <sup>[5]</sup>, and so on.

The behavior is a little bit different depending on if the event thread is enabled (default: it is disabled<sup>[6]</sup>).

[1] [http://www.zkoss.org/zkdemo/file\\_handling/file\\_upload](http://www.zkoss.org/zkdemo/file_handling/file_upload)

[2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Fileupload.html#>

[3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Fileupload.html#>

[4] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Fileupload.html#get\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Fileupload.html#get())

[5] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Fileupload.html#get\(java.lang.String,java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Fileupload.html#get(java.lang.String,java.lang.String))

[6] Prior to 5.0, it is default to enabled. Refer to ZK Configuration Reference: disable-event-thread.

## Creating a custom template for the Static Method: get

When using the static method Fileupload.get(...) to display a generic upload popup, the popup content is defined by a ZUL file, so you could customize it by replacing it with your own implementation. It can be done easily by invoking Fileupload.setTemplate(java.lang.String) ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Fileupload.html#setTemplate\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Fileupload.html#setTemplate(java.lang.String))). Notice that it affects all Fileupload popups subsequently created in an application. It is typically called when the application starts (i.e., in WebAppInit.init(org.zkoss.zk.ui.WebApp) ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/util/WebAppInit.html#init\(org.zkoss.zk.ui.WebApp\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/util/WebAppInit.html#init(org.zkoss.zk.ui.WebApp))) -- for more information, please refer to ZK Developer's Reference: Init and Cleanup).

To implement a custom template, please take a look at the default template (<https://github.com/zkoss/zk/blob/master/zul/src/archive/web/zul/html/fileuploaddlg.zul>).

## Example Usage

When the event thread is disabled (default), the execution won't be suspended when Fileupload.get() ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Fileupload.html#get\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Fileupload.html#get())) is called. In other words, the returned value is always null. To retrieve the uploaded files, the developer has to listen the `onUpload` event, which is sent when the uploading is completed.

By default, the `onUpload` event is sent to all root components. For example, Div (<http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Div.html#>) will, in the following example, receive the `onUpload` event since it is the root component:

```
<div onUpload="processMedia(event.getMedias());">
 <zscript deferred="true"><![CDATA[
 import org.zkoss.util.media.Media;

 public void processMedia(Media[] media) {
 if (media != null) {
 for (int i = 0; i < media.length; i++) {
 if (media[i] instanceof org.zkoss.image.Image) {
 image.setContent(media[i]);
 } else {
 Messagebox.show("Not an image: " + media[i],
 "Error",
 Messagebox.OK, Messagebox.ERROR);
 break; //not to show too many errors
 }
 }
 }
 }
]]></zscript>
 <vbox>
 <button label="Upload" onClick="Fileupload.get(-1);"/>
 <image id="image" />
 </vbox>
</div>
```

## Specify the Callback Event Listener

[Since 6.5.3]

If you prefer the event being sent to the callback event listener, specify the event listener when invoke Fileupload.get().

**Note:** the target of the upload event is always null.

For example:

```
<zk>
 <vbox>
 <button label="Upload">
 <attribute name="onClick">
 Fileupload.get(new EventListener() {
 public void onEvent(UploadEvent event) {
 org.zkoss.util.media.Media media =
 event.getMedia();
 if (media instanceof org.zkoss.image.Image) {
 org.zkoss.image.Image img =
 (org.zkoss.image.Image) media;
 if (img.getWidth() > img.getHeight()) {
 if (img.getHeight() > 300) {
 pics.setHeight("300px");
 pics.setWidth(img.getWidth() * 300
/ img.getHeight() + "px");
 }
 }
 if (img.getHeight() > img.getWidth()) {
 if (img.getWidth() > 400) {
 pics.setWidth("400px");
 pics.setHeight(img.getHeight() *
400 / img.getWidth() + "px");
 }
 }
 image.setContent(img);
 } else {
 Messagebox.show("Not an image: "+media,
 "Error", Messagebox.OK, Messagebox.ERROR);
 }
 }
 })
 </attribute>
 </button>
 <image id="image" />
 </vbox>
</zk>
```

## Specify the Target Component

[Since 5.0.2]

If you prefer the event being sent to a particular component, specify the component in the desktop's attribute called `org.zkoss.zul.Fileupload.target`.

For example, we could have the button to receive the `onUpload` event as follows:

```
<zk>
 <zscript deferred="true"><![CDATA[
 import org.zkoss.util.media.Media;

 Executions.getCurrent().getDesktop().setAttribute(
 "org.zkoss.zul.Fileupload.target", uploadBtn);

 public void processMedia(Media[] media) {
 if (media != null) {
 for (int i = 0; i < media.length; i++) {
 if (media[i] instanceof org.zkoss.image.Image) {
 image.setContent(media[i]);
 } else {
 Messagebox.show("Not an image: " + media[i],
 "Error",
 Messagebox.OK, Messagebox.ERROR);
 break; //not to show too many errors
 }
 }
 }
 }
]]></zscript>
 <vbox>
 <button id="uploadBtn" label="Upload"
 onUpload="processMedia(event.getMedias());"
 onClick="Fileupload.get(-1); />
 <image id="image" />
 </vbox>
</zk>
```

## Event Thread Enabled (deprecated)

If the event thread is enable, the uploaded file will be returned directly by `Fileupload.get()` ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Fileupload.html#get\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Fileupload.html#get())) and other static methods, such as:

```
<zk>
 <button label="Upload">
 <attribute name="onClick">{
 org.zkoss.util.media.Media[] media = Fileupload.get(-1);
 if (media != null) {
 for (int i = 0; i < media.length; i++) {
 if (media[i] instanceof org.zkoss.image.Image)
```

```
{
 org.zkoss.zul.Image image = new
 org.zkoss.zul.Image();
 image.setContent(media[i]);
 image.setParent(pics);
} else {
 Messagebox.show("Not an image:
"+media[i], "Error", Messagebox.OK, Messagebox.ERROR);
 break; //not to show too many errors
}
}
}
}
} </attribute>
</button>
<vbox id="pics" />
</zk>
```

As shown, Fileupload.get(int) ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Fileupload.html#get\(int\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Fileupload.html#get(int))) won't return until the end user uploads the files (and/or closes the dialog).

## Temporary File Created During Uploading

This component depends on Apache Commons Fileupload ( DiskFileItemFactory (<https://commons.apache.org/proper/commons-fileupload/apidocs/org/apache/commons/fileupload/disk/DiskFileItemFactory.html>)), so `org.apache.commons.io.FileCleanerTracker` will delete those temporary files created during uploading. Please refer to Resource cleanup (<https://commons.apache.org/proper/commons-fileupload/using.html>)

You can verify this cleanup by enforcing garbage collecting with JVisualVM.

## Example

Here is an example that uses Fileupload (<http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Fileupload.html#>) as a component:

```
<image id="img" />
Upload your hot shot:
<fileupload label="Upload" onUpload="img.setContent(event.media)" />
```

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: Button

## Supported Children

\*NONE

## Use Cases

Version	Description	Example Location

## Version History

Version	Date	Content
5.0.2	May 2010	Able to specify a target for the onUpload event sent by Fileupload.get() ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Fileupload.html#get()">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Fileupload.html#get()</a> ). Used if the event thread is disabled.

## Fisheye

---

### Fisheye

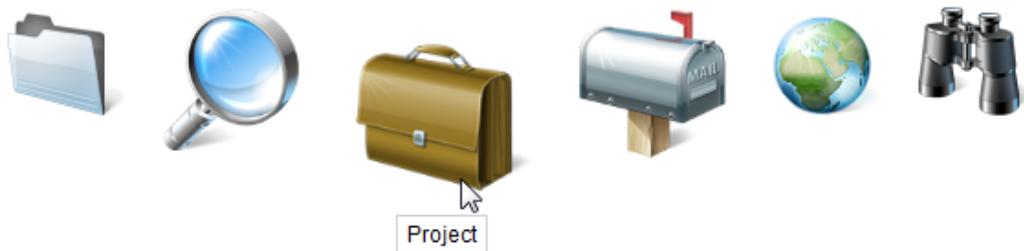
- Demonstration: Fisheye <sup>[1]</sup>
- Java API: Fisheye <sup>[2]</sup>
- JavaScript API: Fisheye <sup>[3]</sup>
- Style Guide: Fisheyebars
- Available in ZK PE and EE only <sup>[13]</sup>

## Employment/Purpose

A fisheye item

### Example

Attach icon edge at bottom  Vertical orient



```
<zk>
<div height="450px">
 <checkbox label="Attach icon edge at bottom"
 onCheck='fsb.attachEdge=self.checked?"bottom":"top"' />
 <checkbox label="Vertical orient"
 onCheck='fsb.orient=self.checked?"vertical":"horizontal"' />
 <separator bar="true" />
 <fisheyebar id="fsb" style="position:absolute;margin:80px 150px;">
 attachEdge="top" itemWidth="80" itemHeight="80"
 itemMaxHeight="160" itemMaxWidth="160">
 <fishey> image="/img/Centigrade-Widget-Icons/FolderABlue-128x128.png" label="Folder"
 onClick="alert(self.label)" />
 <fishey> image="/img/Centigrade-Widget-Icons/ReadingGlass-128x128.png" label="Reading Glasses"
 onClick="alert(self.label)" />
 <fishey> image="/img/Centigrade-Widget-Icons/Briefcase-128x128.png" label="Project"
 onClick="alert(self.label)" />
 <fishey> image="/img/Centigrade-Widget-Icons/MailboxFlag-128x128.png"
 label="Email" onClick="alert(self.label)" />
 <fishey> image="/img/Centigrade-Widget-Icons/Globe-128x128.png"
 label="Globe" onClick="alert(self.label)" />
 <fishey> image="/img/Centigrade-Widget-Icons/Spyglass-128x128.png" label="Spyglass"
 onClick="alert(self.label)" />
 </fisheyebar>
 </div>
</zk>
```

## Properties

### Dynamic Images

For example you can create an image using the Java2D libraries and then set the content of the fisheye to the created image, below is an example of how to do this.

```
<?page title="Auto Generated index.zul"?>
<zk>

 <window title="test of autodisable">
 <fisheyebar><fisheye id="fish1" /></fisheyebar>

 <zscript>
 import java.awt.*;
 import java.awt.image.*;
 import java.awt.geom.*;

 void draw() {
 BufferedImage bi = new BufferedImage(200, 200,
BufferedImage.TYPE_INT_RGB);
 Graphics2D g2d = bi.createGraphics();
 Line2D line = new Line2D.Double(0, 0, bi.getWidth(),
bi.getHeight());
 g2d.setColor(Color.blue);
 g2d.setStroke(new BasicStroke(100));
 g2d.draw(line);
 fish1.setImageContent(bi);
 }

 draw();
 </zscript>
 </window>
</zk>
```

[Since 5.0.0]

## Inherited Functions

Please refer to LabelImageElement for inherited functions.

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: LabelImageElement

## Supported Children

\*None

## Use Cases

Version	Description	Example Location

## Version History

Version	Date	Content
5.0.0	January 2010	Fisheye supports dynamic images

## References

- [1] [http://www.zkoss.org/zkdemo/menu/fisheye\\_menu](http://www.zkoss.org/zkdemo/menu/fisheye_menu)
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkex/zul/Fisheye.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zkex/menu/Fisheye.html#>

# Fisheyebar

## Fisheyebar

- Demonstration: Fisheyebar <sup>[1]</sup>
- Java API: Fisheyebar <sup>[1]</sup>
- JavaScript API: Fisheyebar <sup>[2]</sup>
- Style Guide: Fisheyebar
- Available in ZK PE and EE only <sup>[13]</sup>

## Employment/Purpose

A fisheye bar is a bar of fisheye that is a menu similar to the fish eye menu on the Mac OS.

## Example

Attach icon edge at bottom  Vertical orient



```
<zk>
 <div height="450px">
 <checkbox label="Attach icon edge at bottom"
 onCheck='fsb.attachEdge=self.checked?"bottom":"top"'/>
 />
 <checkbox label="Vertical orient"
 onCheck='fsb.orient=self.checked?"vertical":"horizontal"' />

 <separator bar="true" />
 <fisheyebar id="fsb" style="position:absolute; margin:80px 150px;">
 attachEdge="top" itemWidth="80" itemHeight="80"
 itemMaxHeight="160" itemMaxWidth="160">
 <fisheyeb>
```

```

<fisheyebar>
 <fisheyeb> image="/img/Centigrade-Widget-Icons/MailboxFlag-128x128.png"
 label="Email" onClick="alert(self.label)" />
 <fisheyeb> image="/img/Centigrade-Widget-Icons/Globe-128x128.png"
 label="Globe" onClick="alert(self.label)" />
 <fisheyeb> image="/img/Centigrade-Widget-Icons/Spyglass-128x128.png" label="Spyglass"
 onClick="alert(self.label)" />
</fisheyebar>
</div>
</zk>

```

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

## Supported Children

\* Fisheyeb

## Use Cases

Version	Description	Example Location

## Version History

Version	Date	Content

## References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkex/zul/Fisheyebar.html#>  
[2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkex/menu/Fisheyebar.html#>

# Html

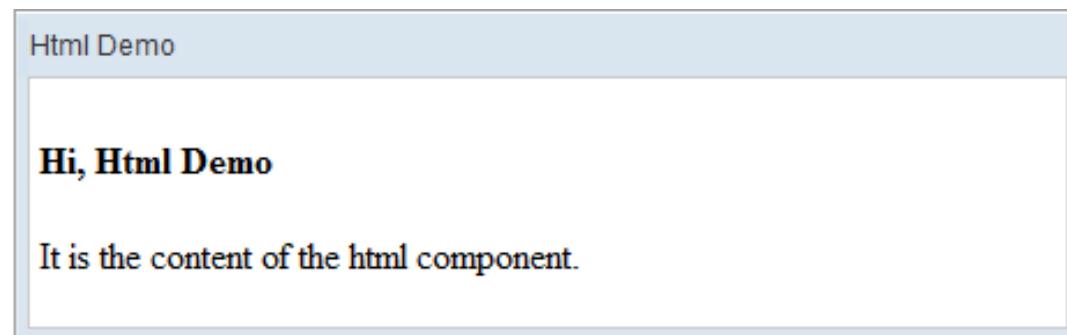
## Html

- Demonstration: Html <sup>[1]</sup>
- Java API: Html <sup>[2]</sup>
- JavaScript API: Html <sup>[3]</sup>
- Style Guide: N/A

## Employment/Purpose

The simplest way is to use an XUL component called `html` to embed whatever HTML tags you want to send directly to the browser. To avoid ZK from interpreting the HTML tags, you usually enclose them with `<! [CDATA[` and `]]>`. In other words, they are not the child component. Rather, they are stored in the `content` property. Notice you can use EL expressions in it.

## Example



```
<window id="win" title="Html Demo" border="normal">
 <html><! [CDATA[
 <h4>Hi, ${win.title}</h4>
 <p>It is the content of the html component.</p>
]]></html>
</window>
```

where `<h4>...</p>` will become the content of the `html` element (see also the `getContent` method of the `org.zkoss.zul.Html` class).

The `html` component generates the HTML SPAN tag to enclose the content. In other words, it generates the following HTML tags when rendered to the browser.

```

 <h4>Hi, Html Demo</h4>
 <p>It is the content of the html component.</p>

```

## Size issue

On Chrome, if <html>'s sibling use vflex to set height flexibly, developer must override .z-html CSS like this:

```
.z-html {
 display: block;
}
```

or

```
.z-html {
 display: inline-block;
}
```

Without this setting, offsetHeight of <html> will be zero, and sibling's height will be wrong.

## Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

## Supported Children

\*None

## Use Cases

Version	Description	Example Location
3.6	Work with HTML tags: The Html component	The Html component
3.6	herf attribute in Html component	[4] [4]
3.6	Use Html component to escape HTML characters	[5] [5]

## Version History

Version	Date	Content
---------	------	---------

## References

- [1] [http://www.zkoss.org/zkdemo/composite/html\\_element](http://www.zkoss.org/zkdemo/composite/html_element)
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Html.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Html.html#>
- [4] <http://www.zkoss.org/forum/listComment/4745>
- [5] <http://www.zkoss.org/forum/listComment/11118>

## Iframe

---

### Iframe

- Demonstration: Iframe <sup>[1]</sup>
- Java API: Iframe <sup>[2]</sup>
- JavaScript API: Iframe <sup>[3]</sup>
- Style Guide: N/A

### Employment/Purpose

The `iframe` component uses the HTML IFRAME tag to delegate a portion of the display to another URL. Though the appearance looks similar to the `include` component. The concept and meaning of the `iframe` component is different.

The content included by the `include` component is a fragment of the whole HTML page.

Because the content is part of the HTML page, the content is part of the desktop and you could access any components, if any, inside of the `include` component. The inclusion is done at the server, and the browser knows nothing about it. It means the URL specified by the `src` property could be any internal resource.

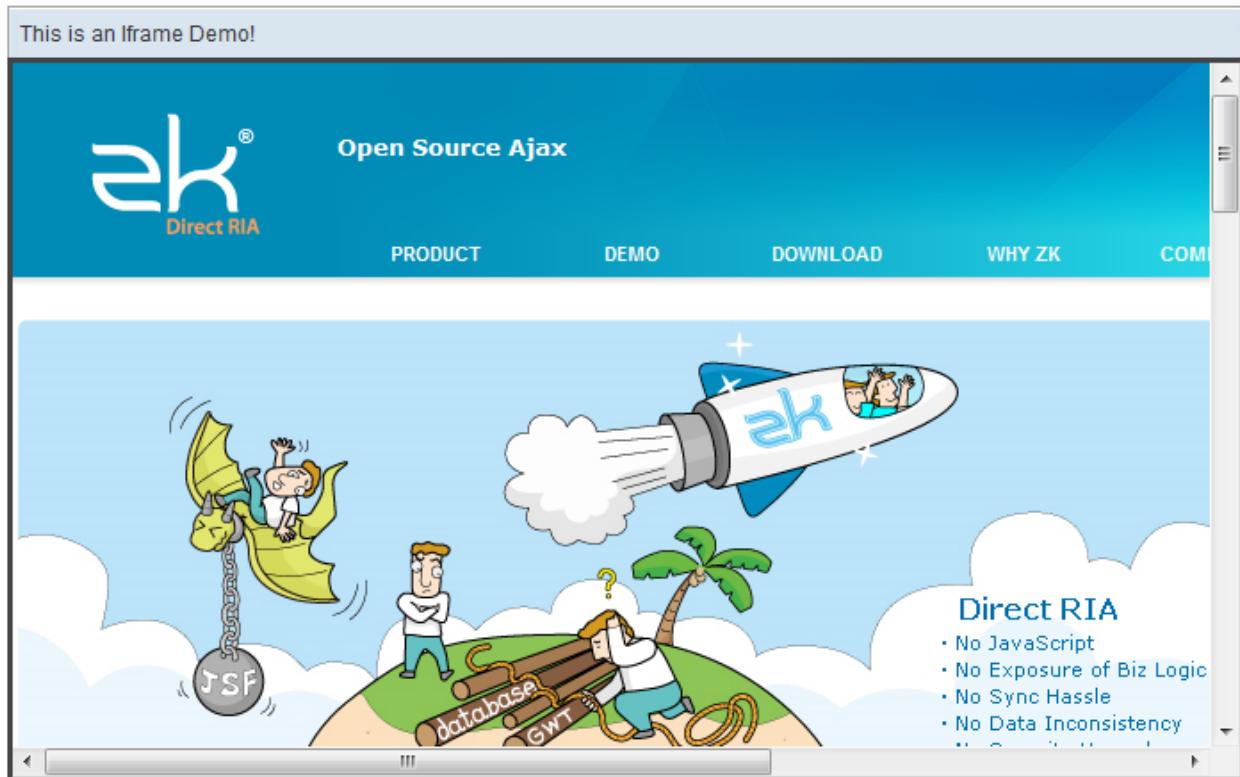
The content of the `iframe` component is loaded by the browser as a separate page. Because it is loaded as a separate page, the format of the content could be different from HTML. For example, you could embed an PDF file.

The embedding is done by the browser, when it interprets the HTML page containing the IFRAME tag. It also implies that the URL must be a resource that you can access from the browser.

Like the `image` and `audio` components, you could specify the dynamically generated content. A typical example is you could use JasperReport to generate a PDF report in a binary array or stream, and then pass the report to an `iframe` component by wrapping the result with the `org.zkoss.util.media.AMedia` class.

In the following example, we illustrate that you could embed any content by use of `iframe`, as long as the client supports its format.

## Example



```
<window id="win" title="This is an Iframe Demo!">
 <iframe style="width:99%; height:400px; border:3px inset;" src="http://www.zkoss.org" />
</window>
```

## The onURIChange Event

When the user navigates the `iframe` component to another URL (or bookmark), an object of `URIEvent`<sup>[4]</sup> is sent to the `iframe` component. This event is usually used to bookmark the status of the `iframe` component, such that the right content can be restored later.

## Integrate with Other Technologies

The `onURIChange` event won't be sent if the `iframe` component contains a non-ZK page. For example, it won't be sent if it contains a PDF page.

On the other hand, if you use other technologies to put a ZK page in an iframe, you can monitor the URL by writing a JavaScript method called `_global_.String`) `_global_.onIframeURLChange(_global_.String, _global_.String)`<sup>[5]</sup> as follows.

```
//Part of your, say, PHP page
<script type="text/script">
function onIframeURLChange(uuid, url) {
 do_whatever_you_need_in_the_technology_you_use(uuid, url);
}
</script>
```

where `uuid` is the ID of the element that you can retrieve by `document.getElementById`, and `url` is the new URL that the iframe is navigated to. Notice that `url` includes the context path, while `URIEvent.getURI()`<sup>[6]</sup> does *not*.

## Retrieving Component inside Iframe

When using `iframe`, the page is actually loaded into another browser window, aka., another desktop if a ZUML document is included.

It is illegal to access components attached to other desktops<sup>[7]</sup>. If you want to retrieve component, you may use `Include`<sup>[8]</sup> instead of `Iframe`<sup>[2]</sup>. Or, you could use Event Queues with the group scope. Of course, you could handle it manually by deliberately passing the information through session.

- 
- [1] <http://www.zkoss.org/zkdemo/composite/iframe>
  - [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Iframe.html#>
  - [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/utl/Iframe.html#>
  - [4] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/URIEvent.html#>
  - [5] [http://www.zkoss.org/javadoc/latest/jsdoc/\\_global/\\_global\\_.html#onIframeURLChange\(\\_global\\_.String,](http://www.zkoss.org/javadoc/latest/jsdoc/_global/_global_.html#onIframeURLChange(_global_.String,)
  - [6] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/URIEvent.html#getURI\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/URIEvent.html#getURI())
  - [7] For more information please refer to the Component-based UI section
  - [8] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Include.html#>

## Communication among iFrames without Server Push

[Since 5.0.4] Available in ZK EE only (<http://www.zkoss.org/product/edition.dsp>)

If your application contains multiple desktops due to iframes in a portal layout it is now possible to communicate between these instances without the need for server push or timer. It thus minimizes the network traffic.

Since ZK 5.0.4, the concept of group has been introduced to enable the use of a group-scope event queue which facilitates easy communication between the instances. The code below demonstrates some examples:

```
EventQueue que = EventQueues.lookup("groupTest", EventQueues.GROUP,
true);

que.subscribe(new EventListener() {
 public void onEvent(Event evt) {
 o.setValue(o.getValue() + evt.getData() + "\n");
 }
});

void publish() {
 String text = i.getValue();

 if (text.length() > 0) {
 i.setValue("");
 que.publish(new Event("onGroupTest", null, text));
 }
}
```

For more information please take a look at ZK Developer's Reference: Event Queues.

## onload

If you'd like to do something when iframe's content has been loaded, you could listen to the `onload` event. However, unlike `onChange` and others, you could not listen the widget-level event ([Event](http://www.zkoss.org/javadoc/latest/jsdoc/zk/Event.html#) (<http://www.zkoss.org/javadoc/latest/jsdoc/zk/Event.html#>) and listened by use of Client-side Event Listening), because the `onload` event might be fired before the widget has been bound to DOM.

Rather, you shall use the client-attribute namespace to register a DOM-level listener as follows.

```
<iframe src="http://www.google.com" width="100%" height="300px"
 xmlns:ca="client/attribute"
 ca:onload="do_whater_you_want () "/>
```

Notice that the registration of `onload` with the client-attribute namespace is DOM-level, so the `this` variable references to the DOM element (rather than the widget).

## Supported Events

Name	Event Type
<code>onURIChange</code>	<b>Event:</b> <a href="#">URIEvent</a> ( <a href="http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/URIEvent.html#">http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/URIEvent.html#</a> ) Denotes the associated URI ( <code>src</code> ) has been changed by user. Use <code>getURI()</code> to retrieve the URI being changed to.

- Inherited Supported Events: `HtmlBasedComponent`

## Supported Children

\*NONE

## Use Cases

Version	Description	Example Location
3.6 or later	Print Iframe Content	( <a href="http://www.zkoss.org/forum/listComment/6599">http://www.zkoss.org/forum/listComment/6599</a> ) ( <a href="http://www.zkoss.org/forum/listComment/6599">http://www.zkoss.org/forum/listComment/6599</a> )

## Version History

Version	Date	Content

# Include

---

## Include

- Demonstration: Include <sup>[1]</sup>
- Java API: Include <sup>[8]</sup>
- JavaScript API: Include <sup>[2]</sup>
- Style Guide: N/A

## Employment/Purpose

The `include` component is used to include the output generated by another servlet. The servlet could be anything including JSF, JSP and even another ZUML page.

```
<window title="include demo" border="normal" width="300px">
 Hello, World!
 <include src="/userguide/misc/includeHello.zul" />
 <include src="/html/frag.html?some=any" />
 <include src="mypage" argument="${anyValue}" other="${anotherValue}" />
</window>
```

Like all other properties, you could dynamically change the `src` attribute to include the output from a different servlet at run time.

If the included output is from another ZUML, this allows developers to access components in the included page as if they are part of the containing page.

If the `include` component is used to include a ZUML page, the included page will become part of the desktop. However, the included page is not visible until the request is processed completely. In other words, it is visible only in the following events, triggered by user or timer.

The reason is that the `include` component includes a page as late as the Rendering phase. On the other hand, zscript takes place at the Component Creation phase, and `onCreate` takes place at the Event Processing Phase. They both execute before the inclusion.

## Example

```
<window title="include demo" border="normal" width="300px">
 Hello, World!
 <include src="/userguide/misc/includeHello.zul" />
 <include src="/html/frag.html?some=any" />
 <include src="mypage" argument="${anyValue}" other="${anotherValue}" />
</window>
```

## Modes

There are two ways to include another ZUML document: `instant` and `defer`. It is controlled by the property called `mode` (`Include.setMode(java.lang.String)` [3]). By default, it is `auto`, i.e., the real mode is decided automatically as described below. Notice that the included file should not contain `<html>` tag.

### Auto

The `auto` mode (default) decides the mode based the page to include. If the page (`Include.setSrc(java.lang.String)` [4]) is ended with the extension named `.zul`, `.zhtml`, `.html`, or `.xhtml`, the `instant` mode is assumed. Otherwise, the `defer` mode is assumed. If it is not the case (such as your ZUML page's extension is not any of them), you could specify the mode explicitly.

Notice that if a query string is specified, the `defer` mode is assumed, too, since the `instant` mode does not support the query string (#1).

That means , the following sample is using `defer` mode so the  `${param.test}` in header.zul is working .

```
<include src="header.zul?test=5" /> <!-- default mode is auto -->
```

but the following sample is using `instant` mode , so the  `${param.test}` in the header.zul is not working.

```
<include mode="instant" src="header.zul?test=5" /> <!-- set the instant mode explicitly -->
```

If you have no any query string in src and it's ended with `".zul"` , the include will be `instant` mode by default.

```
<include src="header.zul" /> <!-- default mode is auto -->
```

The code below demonstrates how to use the `auto` mode:

```
<window title="demo" border="normal">
 <include mode="auto" src="another.zul" />
</window>
```

### Instant

In the `instant` mode, the `include` component loads the page by the use of `org.zkoss.zk.ui.Component`, `java.util.Map`) `Execution.createComponent(java.lang.String, org.zkoss.zk.ui.Component, java.util.Map)` [5]. It means that the components defined in the included page are instantiated *instantly* and added as children of the `include` component.

Unlike the `defer` mode, no additional Page [6] instance is created.

The code below demonstrates how to use the `instant` mode:

```
<window title="demo" border="normal">
 <include mode="instant" src="include.zul" />
</window>
```

Notice the `include` component itself is a ID space owner, so there is no issue of ID conflicts. But, it also means you have to use Path [7] or other techniques to retrieve the child components.

## Pass Values to the Included Page

you can pass values through `java.lang.Object`) `Include.setDynamicProperty(java.lang.String, java.lang.Object)` [8].

Unlike the `defer` mode, the query string is not supported in the `instant` mode.

For example:

```
<include src="mypage" some="something" another="${expr}" />
```

Then, you could retrieve the values by use the `arg` object as described in the Load ZUML in Java section.

---

In the `defer` mode, the values have to be retrieved by the `requestScope` object.

---

Unfortunately there is a bug [9] in 5.0.5, so it won't work unless you use 5.0.6 or later.

## Defer

In the `defer` mode, the `include` component includes the page by going through the Servlet container(the include method of `javax.servlet.RequestDispatcher`). Thus, it is OK to include any kind of pages, not limited to ZUML documents.

```
<window title="demo" border="normal">
 <include mode="defer" src="include.zul" />
</window>
```

## Differences to the Instant Mode

Here is a list of differences between the `defer` and `instant` modes if a ZUML document is included (directly or indirectly going through another, say, JSP page) to contain the components defined in the included ZUML document.

- In the `defer` mode, an instance of Page [6] will be created.
- In the `defer` mode, the instantiated components become the root of the Page [6] instance. The `include` component itself has no child component at all.
- In the `defer` mode, the page is included when the `include` component is rendered. Thus, the Page [6] instance (and its content) is not available when loading the ZUML document that the `include` component belongs to. It means you cannot access any of its content until receiving an AU request from the client.

## Pass Values to the Included Page

There are two ways to pass values to the included page in the `defer` mode. First, you can pass them with the query string.

```
<include mode="defer" src="mypage?some=something" />
```

Then, in the included page, you can access them with `Execution.getParameter(java.lang.String)` [10] or the `javax.servlet.ServletRequest` interface. In EL expressions (of the included page), you can use the `param` object or the `paramValues` object to access them.

```
${param.some}
```

Notice that you can only pass String-typed values with the query string. Alternatively, we can pass any kind of values with the so-called dynamic properties by use of `java.lang.Object`) `Include.setDynamicProperty(java.lang.String, java.lang.Object)` [8] or, in ZUL, a dynamic property as follows:

```
<include mode="defer" src="mypage" some="something" another="${expr}" />
```

With the dynamic properties, you can pass non-String-typed values. In the included page, you can access them with `Execution.getAttribute(java.lang.String)`<sup>[11]</sup> or the `javax.servlet.ServletRequest` interface. In EL expressions (of the included page), you can use the `requestScope` object to access them.

```
${requestScope.some}
```

## Include the Same Page Twice

With the `include` component, you could include any pages multiple times no matter it is in the `instant` or `defer` mode. For example,

```
<include src="/mypage.zul"/>
<include src="/mypage.zul"/>
```

However, if you are using the `defer` mode and want to access the component inside of them, you have to assign a unique identifier of the page being included. Here is what you can do.

```
<include mode="defer" src="/mypage.zul?pageId=first"/>
<include mode="defer" src="/mypage.zul?pageId=second"/>
```

In addition, in the page being included, i.e., `mypage.zul` in this example, you have to write

```
<?page id="${param.pageId}" ?>
```

Then, you could access their component by the use of Path<sup>[7]</sup> as follows.

```
Path.getComponent('//first/textbox/');
Path.getComponent('//second/textbox');
```

Notice that, in the `defer` mode, components are created as late as rendering the `include` component, so you could access them only in the event listener serving AU requests (aka., Ajax).

## Include Non-ZUML Pages

If the included page is not ZUML (such as a HTML fragment), the content is generated directly, so they might be evaluated before widgets are rendered. Technically it is OK. However, if the included page embeds some JavaScript code that depends on widgets, it might run as you expected. For example, assume we have the includer and included page as follows:

```
<!-- includer -->
<window id="main">
 <include src="frag.html"/>
</window>

<!-- included -->
<script>zK.log(jq("$main"));</script>
```

Then, `jq("$main")` will resolve nothing (an empty array) because its content is evaluated first.

There are two solutions to make it evaluate later:

1. Use `int) zk.afterMount(_global_.Function, int)`<sup>[12]</sup> in the included page
2. Or, specify this custom attribute to defer the rendering in the includer.

### Use zk.afterMount() in the included page

First, you could use int) zk.afterMount(\_global\_.Function, int) [12] to defer the evaluation as follows in the included page:

```
<!-- the included non-ZUML page -->
<script>
zk.afterMount(function () {
 zk.log(jq("$main"));
 ...//handle widgets:
});
</script>
```

### Use the org.zkoss.zul.include.html.defer attribute

[since 5.0.7]

Alternatively, you could specify a custom attribute called the org.zkoss.zul.include.html.defer in the Include [8] component (rather than using int) zk.afterMount(\_global\_.Function, int) [12] in the included component):

```
<!-- the includee -->
<include src="included.html">
 <custom-attributes org.zkoss.zul.include.html.defer="true"/>
</include>
```

## Backward Compatibility

For versions prior to 5.0, the defer mode is the default. If you prefer to keep using the defer mode, you could specify a library property called org.zkoss.zul.include.mode as follows.

```
<library-property>
 <name>org.zkoss.zul.include.mode</name>
 <value>defer</value>
</library-property>
```

## Refresh Included Pages

First, use the include component with either the instant or defer mode to include whatever page you want (ZUML, JSP, JSF or whatever) inside a ZK document. Second, you can dynamically change it by changing the src property (Include.setSrc(java.lang.String) [4]).

For example, the following code would change the included page from hello.zul to byebye.zul, when an end user press the Bye! button.

```
<include id="inner" src="hello.zul"/>
<button label="Bye!" onClick='inner.src = "byebye.zul"'>
```

If you want to reload the same page (and not to change to another page), you have to set the src property to null first; then set it back to what it was. Because ZK optimizes operations, setting the same value to the same property would be deemed to do nothing.

For example, the following code would refresh the hello.zul page when an end user press the Reload button.

```
<include id="inner" src="hello.zul"/>
<button id="reload" label="Reload" onClick="String tmp=inner.src; inner.src=null; inner.src=tmp;" />
```

Another way to reload is to invalidate the "include" component by use of Component.invalidate() [13] as follows.

```
<include id="inner" src="hello.zul"/>
<button id="reload" label="Reload" onClick="inner.invalidate();"/>
```

Notice that Component.invalidate() [13] will cause the included page to be reloaded in both the instance and defer mode.

## Access Components Inside

Since include creates ID space, to access components inside it , please refer to ZK\_Developer's\_Reference/UI\_Composing/ID\_Space#Find\_Component\_Manually

## Progressing for Slow Pages

If an included page takes too long to load, you could specify true to the progressing property (Include.setProgressing(boolean) [14]). Thus, the included page won't load in the same HTTP request, so the including page will be ready to the client as soon as possible.

```
<include page="slow.zul" progressing="true"/>
```

ZK Client Engine will show a busy message to indicate that the page is not ready yet, and this prevents users from accessing it. Then, the real loading of the included page will then take place later. Though the end user still cannot access the page, the feed back is much better (with a *semi-ready* page than totally blank).

This feature is actually done by the use of the so-called echo event. For more information, please refer to the Long Operations: Use Echo Events section.

This feature cannot be used with the instant mode. If the auto mode is used (default), it switches to the defer mode automatically.

## Custom Attributes

### org.zkoss.zul.include.html.defer

```
[default: false]
[inherit: true] [15]
[since 5.0.7]
```

It specifies whether to defer the rendering of the included non-ZUML page, until all widgets are instantiated and rendered at the client. By default, if the included page is not ZUML (i.e., HTML fragment), the content is generated directly, and they might be evaluated before widgets are rendered (depending on the browser and the complexity of a page).

For more information, please refer to the #Include Non-ZUML Pages section.

- ```
[1] http://www.zkoss.org/zkdemo/composite/include
[2] http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Include.html#
[3] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Include.html#setMode(java.lang.String)
[4] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Include.html#setSrc(java.lang.String)
[5] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/Execution.html#createComponents(java.lang.String,
[6] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/Page.html#
[7] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/Path.html#
[8] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Include.html#setDynamicProperty(java.lang.String,
[9] https://sourceforge.net/tracker/?func=detail&aid=3142583&group_id=152762&atid=785191
[10] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/Execution.html#getParameter(java.lang.String)
[11] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/Execution.html#getAttribute(java.lang.String)
[12] http://www.zkoss.org/javadoc/latest/jsdoc/_global_zk.html#afterMount(_global_Function,
[13] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/Component.html#invalidate()
[14] http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Include.html#setProgressing(boolean)
[15] The custom attribute could be specified in this component, or any of its ancestor. In addition, it could be specified as a library property to
enable or disable it for the whole application.
```

Supported Events

| Name | Event Type |
|------|------------|
| None | None |

- Inherited Supported Events: XulElement

Supported Children

*NONE

Use Cases

| Version | Description | Example Location |
|---------|-------------|------------------|
|---------|-------------|------------------|

Version History

| Version | Date | Content |
|---------|--------------|--|
| 5.0.0 | January 2010 | The mode is default to auto (rather than defer). You could configure the default mode to defer by specifying a library property called org.zkoss.zul.include.mode in WEB-INF/zk.xml. |
| 5.0.7 | April 2011 | The custom attribute called org.zkoss.zul.include.html.defer was introduced to defer the rendering of a non-ZUML page (such as HTML fragment) until all widgets are instantiated and rendered at the client. |

Image

Image

- Demonstration: Image ^[1]
- Java API: Image ^[2]
- JavaScript API: Image ^[3]
- Style Guide: N/A

Employment/Purpose

An `image` component is used to display an image at the browser. There are two ways to assign an image to an `image` component. First, you could use the `src` property to specify a URI where the image is located. This approach is similar to what HTML supports. It is useful

if you want to display a static image, or any image that can be identified by URL.

```
<image src="/my.png">
```

Locale Dependent Image

Like using any other properties that accept an URI, you can specify "*" for identifying a Locale dependent image. For example, if you have different images for different Locales, you could use the following code.

```
<image src="/my*.png"/>
```

Assuming one of your users is visiting your page with `de_DE` as the preferred Locale. Zk will try to locate the image file called `/my_de_DE.png`. If it is not found, it will try `/my_de.png` and finally `/my.png`.

Please refer to ZK Developer's Reference/Internationalization/Locale-Dependent Resources for more details.

Secondly, you could use the `setContent` method to set the content of an image to an `image` component directly. Once assigned, the image displayed at the browser will be updated automatically. This approach is useful if an image is generated dynamically.

For example, you can generate a map for the location specified by a user as demonstrated below.

```
<zk>
    Location: <textbox onChange="updateMap(self.value)" />
    Map: <image id="image"/>
    <zscript><![CDATA[
        void updateMap(String location) {
            if (location.length() > 0) {
                org.zkoss.image.AImage img = new
org.zkoss.image.AImage(location);
                image.setContent(img);
            }
        }
    ]]>
</zscript>
</zk>
```

In the above example, we assume that you have a class named `MapImage` for generating a map of the specified location.

Notice that the image component accepts the content encapsulated by the `org.zkoss.image.Image` format. If the image generated by your tool is not in this format, you can use the `org.zkoss.image.AImage` class to wrap a binary array of data, a file or an input stream into the `Image` interface.

In traditional Web applications, caching a dynamically generated image is complicated, however with the `image` component, you don't need to worry about it. Once the content of an image is assigned, it belongs to the `image` component, and the memory it occupies will be released automatically when the `image` component is no longer used.

Tip: If you want to display the content other than an image, say a PDF, you can use the `iframe` component. Please refer to the relevant section for details.

Image Supports `javax.awt.image.RenderedImage`

Since version 3.0.7 ZK allows `image`, `button` and related components to support `RenderedImage` directly without a format conversion. Here is the example code,

```
<window title="Test of Live Image">
    <image id="img"/>
    <zscript>
        import java.awt.*;
        import java.awt.image.*;
        import java.awt.geom.*;
        int x = 10, y = 10;

        void draw(int x1, int y1, int x2, int y2) {
            BufferedImage bi = new BufferedImage(400, 300,
BufferedImage.TYPE_INT_RGB);
            Graphics2D g2d = bi.createGraphics();
            Line2D line = new Line2D.Double(x1, y1, x2, y2);
            g2d.setColor(Color.blue);
            g2d.setStroke(new BasicStroke(3));
            g2d.draw(line);
            img.setContent(bi);
        }
        draw(x, y, x += 10, y += 10);
    </zscript>
    <button label="change" onClick="draw(x, y, x += 10, y += 10)" />
</window>
```

Preload Image

[since 6.0.0]

The feature is applied to all of the LabelImageElement and Image components.

By default the preload function is disabled, so users have to specify the *custom-attributes* to be true. For example,

```
<image src="xxx.png">
    <custom-attributes org.zkoss.zul.image.preload="true" />
</image>
```

Or specify just below the root component.

For example,

```
<window>
    <custom-attributes org.zkoss.zul.image.preload="true" />
    <button image="xxx.png" />
    <image src="xxx.png" />
</window>
```

As you can see, the *custom-attributes* will be checked recursively

[since 6.5.2]

The feature can also applied from zk.xml as a library property.

For example,

```
<!-- zk.xml -->
<zk>
    <library-property>
        <name>org.zkoss.zul.image.preload</name>
        <value>true</value>
    </library-property>
</zk>
```

Example

```
<image src="/my.png">
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

*None

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
6.0.0	September 2011	A way to pre-load images since many UIs depend on the size of an image [8]

References

- [1] http://www.zkoss.org/zkdemo/multimedia/dynamic_image
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Image.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Image.html#>

Imagemap

Imagemap

- Demonstration: N/A
- Java API: Imagemap ^[1]
- JavaScript API: Imagemap ^[2]
- Style Guide: N/A

Employment/Purpose

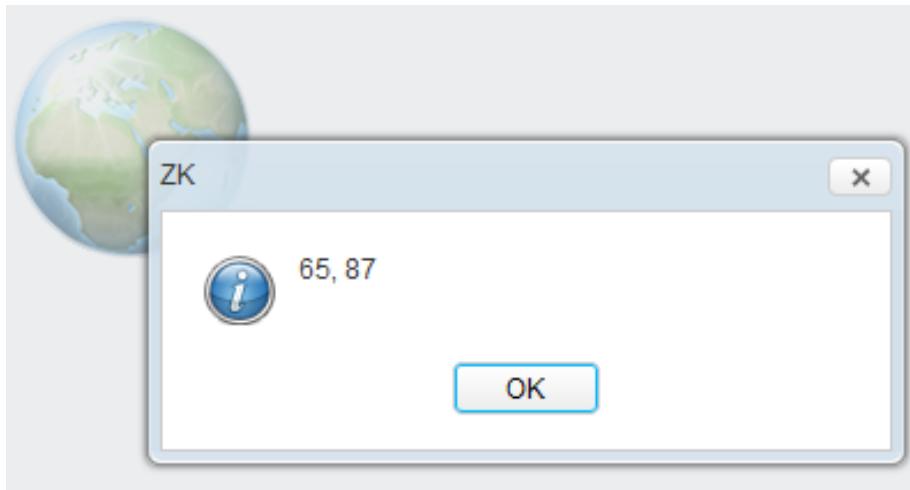
A `imagemap` component is a special image. It accepts whatever properties an `imagecomponent` accepts. However, unlike `image`, if a user clicks on the image, an `onClick` event is sent back to the server with the coordinates of the mouse position. In contrast, the `onClick` event sent by `image` doesn't contain the coordinates. The coordinates of the mouse position are screen pixels counted from the upper-left corner of the image beginning with `(0, 0)`. It is stored as instance of `org.zkoss.zk.ui.event.MouseEvent`. Once a controller receives the `onClick` event, it can get the coordinates of the mouse position by `getX()` and `getY()`.

Note: Don't try to use CSS background as your image, the image map need a real image or it won't work.

Example

```
<imagemap src="/img/sun.jpg" onClick="alert(event.x + ", " +event.y)"/>
```

For example, if a user clicks 208 pixels over and 205 pixels down from the upper-left corner of the image displayed from the following statement, then the user gets the result as depicted below.



Supported Events

Name	Event Type
None	None

- Inherited Supported Events: Image

Supported Children

* Area

Use Cases

Version	Description	Example Location
3.6	How to get area clicked from Imagemap onClick Event	[3] [3]
3.6	Imagemap with hyperlink	[4] [4]

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Imagemap.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Imagemap.html#>
- [3] <http://www.zkoss.org/forum/listComment/1336>
- [4] <http://www.zkoss.org/forum/listComment/3016>

Area

Area

- Demonstration: N/A
- Java API: Area ^[7]
- JavaScript API: Area ^[1]
- Style Guide: N/A

Employment/Purpose

An area of a image map. Instead of the application processing the coordinates, developers can add the area components as children of a imagemap component thus defining a target. The imagemap component will translate the mouse pointer coordinates into a logical name ie. The id of the area the user clicked.

Example

```
<imagemap src="/img/sun.jpg" onClick="alert(event.area)">
    <area id="First" coords="0, 0, 100, 100"/>
    <area id="Second" shape="circle" coords="200, 200, 100"/>
</imagemap>
```

The shape Property

An area component supports three kinds of shapes: circle, polygon and rectangle. The coordinates of the mouse position are screen pixels counted from the upper-left corner of the image beginning with (0, 0).

Shape	Coordinates / Description
circle	coords="x, y, r" where x and y define the position of the circle's center and r is the radius in pixels.
polygon	coords="x1, y1, x2, y2, x3, y3..." where each pair of x and y define a point of the polygon. At least three pairs of coordinates are required to define a triangle. The polygon is automatically closed, so it is not necessary to repeat the first coordinate at the end of the list to close the region.
rectangle	coords="x1, y1, x2, y2" where the first coordinate pair is one corner of the rectangle and the other pair is the corner diagonally opposite. A rectangle is just a shortened way of specifying a polygon with four vertices.

If the coordinates in one `area` component overlap with another, the first one takes precedence.

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: AbstractComponent

Supported Children

*None

Use Cases

Version	Description	Example Location
5.0.2	Area in Imagemap with href	[4]

Version History

Version	Date	Content

References

[1] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Area.html#>

Label

-
- ## Label
- Demonstration: Label ^[1]
 - Java API: Label ^[1]
 - JavaScript API: Label ^[2]
 - Style Guide: Label

Employment/Purpose

A label component represents a piece of text. A pure text on the zul will be automatically converted to a label.

Example

Label Demo	
Label(normal):	This is a Label. Next Line...
Label(color):	This is a Label. Next Line...
Label(font):	This is a Label. Next Line...
Label(size):	This is a Label. Next Line...
Label(maxlength):	This...
Label(pre):	This is a Label. Next Line...
input:	This is a Label. Next Line...

```
<window title="Label Demo" >
<grid>
    <rows>
        <row>Label(normal): <label id="lb1"/></row>
        <row>Label(color): <label id="lb2" style="color:red"/></row>
        <row>Label(font): <label id="lb3" style="font-weight:bold"/></row>
        <row>Label(size): <label id="lb4" style="font-size:14pt"/></row>
        <row>Label(maxlength): <label id="lb5" maxlength="5"/></row>
        <row>Label(pre): <label id="lb6" pre="true"/></row>
        <row>input:
            <textbox id="txt" rows="2"><attribute name="onChange">
                lb1.value=self.value;
                lb2.value=self.value;
                lb3.value=self.value;
                lb4.value=self.value;
```

```
        lb5.value=self.value;
        lb6.value=self.value;
    </attribute></textbox>
</row>
</rows>
</grid>
</window>
```

You can control how a label is displayed with the `style`, `pre` and `maxlength` Properties.

For example, if you specify `pre` to be `true`, all white spaces, such as new line, space and tab, are preserved.

A label component represents a piece of text.



Hello World

```
<window border="normal">
    Hello World
</window>
```

If you want to add an attribute to a label, it has to be written as follows:



Hello World

```
<window border="normal">
    <label style="color: red" value="Hello World" />
</window>
```

Special Character

Since ZUML is XML, not HTML, so it doesn't accept `&nbsp`. However, you can use ` ` instead. For the whole list, please refer th List of XML and HTML character entity references ^[3].

Properties

Pre, Hyphen, Maxlength and Multiline

[since 5.0.0]

You can control how a label is displayed using the `pre`, `multiline` and `maxlength` properties. For example, if you specify `pre` to be true, all white spaces, such as new lines, spaces and tabs, are preserved.

pre	multiline	maxlength	Description
true	any	any	All white spaces are preserved, including new lines, spaces and tabs.
false	true	any	New lines are preserved.
false	false	positive	The label only show its value up to the length of "maxlength".
false	false	0	The label is displayed regularly.

this thing has spaces.
next line.

this thing no space. next line.

this i...

```
<window border="normal" width="300px">
    <vbox id="result">
        <label id="lb1" pre="true"></label>
        <separator bar="true"/>
        <label id="lb2" multiline="false" />
        <separator bar="true"/>
        <label id="lb3" maxlength="10" />
        <zscript><![CDATA[
            lb1.value = "      this      thing      has      spaces.\nnext
line.";
            lb2.value = "      this      thing      no      space.\nnext
line.";
            lb3.value = "      this is more than 10 chars.";
        ]]></zscript>
    </vbox>
</window>
```

[For ZK3 users]

This displaying rule is slightly different in ZK3.

hyphen	pre	maxlength	Description
false	false	positive	Truncated the characters that exceeds the specified maxlength.
true	any	positive	If the length of a line exceeds maxlength, the line is hyphenated.
false	true	any	maxlength is ignored.
any	any	0	hyphen is ignored.

```
this is 9  
-----  
this is  
ten more  
to show  
-----  
this fram-  
ework  
-----  
performan-  
ce is eve-  
rything
```

```
<window border="normal" width="300px">  
  <vbox id="result">  
    </vbox>  
    <zscript><![CDATA[  
      String[] s = {"this is 9",  
                    "this is ten more to show",  
                    "this framework",  
                    "performance is everything"};  
      for (int j = 0; j < s.length; ++j) {  
        Label l = new Label(s[j]);  
        lmaxlength = 9;  
        lhyphen = true;  
        lparent = result;  
        Separator sep = new Separator();  
        sep.setBar(true);  
        sep.parent = result;  
      }  
    ]]>  
  </zscript>  
</window>
```

The `multiline` property is similar to the `pre` property, except it only preserves new lines and white space at the beginning of each line.

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

*NONE

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Label.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Label.html#>
- [3] https://en.wikipedia.org/wiki/List_of_XML_and_HTML_character_entity_references

Menu

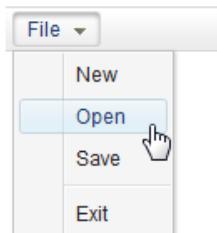
Menu

- Demonstration: Menu [1]
- Java API: Menu [2]
- JavaScript API: Menu [3]
- Style Guide: Menu in Menubar, Menu in Menupopup

Employment/Purpose

An element, much like a button, is placed on a menu bar. When the user clicks the menu element, the child Menupopup of the menu will be displayed. This element is also used to create submenus of Menupopup.

Example

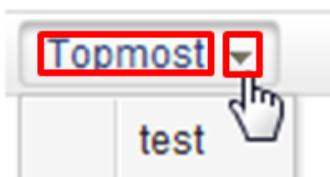


```
<menubar>
  <menu label="File">
    <menupopup>
      <MenuItem label="New" onClick="alert(self.label)" />
      <MenuItem label="Open" onClick="alert(self.label)" />
      <MenuItem label="Save" onClick="alert(self.label)" />
      <menuseparator/>
      <MenuItem label="Exit" onClick="alert(self.label)" />
    </menupopup>
  </menu>
</menubar>
```

Properties

onClick

If we take a look at the screenshot of the Menu below both the right hand side and left hand side have been outlined in red. Clicking the arrow on the right hand side will show the menu whereas clicking the main button (the left hand side) will fire the onClick event. The red outline is used to highlight the clickable areas of the Menu.



The code to register an onClick event is shown below:

```
<menubar>
    <menu label="Topmost" onClick='alert(1);'>
        <menupopup>
            <MenuItem label="test"/>
        </menupopup>
    </menu>
</menubar>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: LabelImageElement

Supported Children

* Menupopup

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/zkdemo/menu>
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Menu.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/menu/Menu.html#>

Menubar

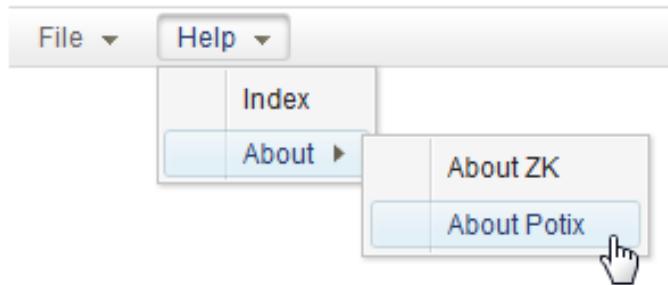
Menubar

- Demonstration: Menu ^[1]
- Java API: Menubar ^[1]
- JavaScript API: Menubar ^[2]
- Style Guide: Menubar

Employment/Purpose

A container usually contains more than one menu elements.

Example



```
<menubar id="menubar">
    <menu label="File">
        <menupopup onOpen="alert(self.id)">
            <menuitem label="New" onClick="alert(self.label)" />
            <menuitem label="Open" onClick="alert(self.label)" />
            <menuitem label="Save" onClick="alert(self.label)" />
            <menuseparator />
            <menuitem label="Exit" onClick="alert(self.label)" />
        </menupopup>
    </menu>
    <menu label="Help">
        <menupopup>
            <menuitem label="Index" onClick="alert(self.label)" />
            <menu label="About">
                <menupopup>
                    <menuitem label="About ZK" onClick="alert(self.label)" />
                    <menuitem label="About Potix" onClick="alert(self.label)" />
                </menupopup>
            </menu>
        </menupopup>
    </menu>
</menubar>
```

Properties

Scrollable

The code below demonstrates how easy it is to make the Menubar scrollable!



```
<menubar width="200px" scrollable="true">
  <menu label="Menu1">
    <menupopup>
      <menuitem label="Submenu"/>
    </menupopup>
  </menu>
  <menu label="Menu2">
    <menupopup>
      <menuitem label="Submenu"/>
    </menupopup>
  </menu>
  <menu label="Menu3">
    <menupopup>
      <menuitem label="Submenu"/>
    </menupopup>
  </menu>
</menubar>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

```
* Menu, MenuItem, Menuseparator
```

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Menubar.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/menu/Menubar.html#>

MenuItem

MenuItem

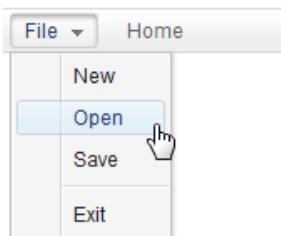
- Demonstration: Menu ^[1] and Fileupload
- Java API: MenuItem ^[1]
- JavaScript API: MenuItem ^[2]
- Style Guide: MenuItem in Menubar, MenuItem in Menupopup

Employment/Purpose

A single choice in a Menupopup element. It acts much like a button but it is rendered on a menu. Default getZclass(): z-menu-item .

Within ZK 5, the file upload has been redesigned so it can be integrated with any widget. For example, the toolbarbutton can now be used to upload a file. In addition to this, the display of the upload status has been enhanced and can be customized easily.

Example



```
<menubar>
  <menu label="File">
    <menupopup>
      <menuitem label="New" onClick="alert(self.label)" />
      <menuitem label="Open" onClick="alert(self.label)" />
      <menuitem label="Save" onClick="alert(self.label)" />
```

```

<menuseparator/>
<menuitem label="Exit" onClick="alert(self.label)"/>
</menupopup>
</menu>
<menuitem label="Home"/>
</menubar>

```

Fileupload Example

```
<menuitem upload="true" label="Customized Attach" onUpload='alert("File is uploaded!")' />
```

Properties

Autodisable

[Since 5.0.7]

MenuItem.setAutodisable(java.lang.String)^[3] is used to disable a menuitem automatically, when it is clicked. It is useful to prevent the user from clicking it twice (and firing redundant requests), which is common if the request takes too long to serve.

The simplest use is to specify it with self as follows. Then, the menuitem is disabled when it is clicked.

```
<menuitem id="ok" label="OK" autodisable="self" />
```

If you'd like to disable several menuitems, you could specify all of them in this property by separating with a comma. For example, the following disables both menuitems, when one of them is clicked.

```
<menuitem id="ok" label="OK" autodisable="ok, cancel" />
<menuitem id="cancel" label="Cancel" autodisable="ok, cancel" />
```

The menuitem will be enabled automatically, after the request has been served (i.e., the response has been sent back to the client). If you prefer to enable them manually (i.e., by calling MenuItem.setDisabled(boolean)^[4] explicitly), you could prefix the ID with a plus (+). For example,

```
<menuitem id="ok" label="OK" autodisable="+self, +cancel" />
```

Then, you could enable them manually under the situation depending on your application's requirement, such as

```
if (something_happens) {
    ok.setDisabled(false);
    cancel.setDisabled(false);
}
```

Enable Autodisable for All Menuitems

As described in ZK Developer's Reference: Customization, you could customize ZK to enable autodisable for all menuitem by specifying the following in the custom language addon:

```
<language-addon>
    <component>
        <component-name>menuitem</component-name>
        <extends>menuitem</extends>
    </component>

```

```
<property>
    <property-name>autodisable</property-name>
    <property-value>self</property-value>
</property>
</component>
</language-addon>
```

Href

In addition to handling the onClick event, you could specify the URL in the href property (MenuItem.setHref(java.lang.String)^[5]), such that the browser will navigate to the URL you specified directly (without sending back any request to the server). If you prefer to visit the URL in another browser window, you could specify the name in MenuItem.setTarget(java.lang.String)^[6] (just like using a HTML A tag).

Notice that the end user could hold the Control key and click on the menuitem to visit the link in a new browser window (like a HTML A tag does).

Href and the onClick Event

There are two ways to add behavior to a menuitem. Firstly, you can specify a listener for the onClick event. Secondly, you could specify a URL for the href property (MenuItem.setHref(java.lang.String)^[5]). If both are specified, the href property has the higher priority, i.e., the onClick event won't be sent.

```
<zk>
    <menubar>
        <menuitem label="click me" onClick="do_something_in_Java()"/>
        <menuitem label="don't click that one, click me" href="/another_page.zul"/>
    </menubar>
</zk>
```

Href and SendRedirect

The href property is processed at the client. In other words, the browser will jump to the URL specified in the href property, so your application running on the server has no chance to process it.

If you have to process it on the server or you have to decide whether to jump to another URL based on certain condition, you could listen to the onClick event, process it, and then invoke Executions.sendRedirect(java.lang.String)^[7] if it jumps to another URL.

For end users, there is no difference between the use of MenuItem.setHref(java.lang.String)^[5] and Executions.sendRedirect(java.lang.String)^[7].

```
<zk>
    <menubar>
        <menuitem label="redirect" onClick="Executions.sendRedirect("another.zul")" />
        <menuitem label="href" href="another.zul"/>
    </menubar>
</zk>
```

Since the onClick event is sent to the server for processing, you are able to perform additional tasks before invoking Executions.sendRedirect(java.lang.String)^[7], such as redirecting to another page only if certain conditions are satisfied.

On the other hand, the `href` property is processed at the client side. Your application won't be notified when users click the menuitem.

Upload

By specifying the upload property (`MenuItem.setUpload(java.lang.String)`^[8]), you could make a menuitem used for uploading files. For example,

```
<MenuItem upload="true" label="Upload" onUpload='alert(event.media)' />
```

Once the file(s) are uploaded, the `onUpload` event will be sent with an instance of `UploadEvent`^[5]. And, you could retrieve the uploaded files from `UploadEvent.getMedia()`^[6] and `UploadEvent.getMedias()`^[9]

If you want to customize the handling of the file upload at the client, you can specify a JavaScript class when calling this method:

```
<MenuItem upload="foo.Upload"/> <!-- assume you implement a JavaScript class: foo.Upload -->
```

Another options for the upload can be specified as follows:

```
<MenuItem label="Upload" upload="true,maxsize=-1,native"/>
```

where

- `maxsize`: the maximal allowed upload size of the component, in kilobytes, or a negative value if no limit.
- `native`: treating the uploaded file(s) as binary, i.e., not to convert it to image, audio or text files.

Supported Events

Name	Event Type
<code>onCheck</code>	Event: <code>CheckEvent</code> ^[5] Denotes user has checked the item.
<code>onUpload</code>	Event: <code>UploadEvent</code> ^[5] Denotes user has uploaded a file to the component.

- Inherited Supported Events: `LabelImageElement`

Supported Children

*NONE

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content
5.0.7	May 2011	<code>MenuItem.setAutodisable(java.lang.String)</code> ^[3] was used to disable a menuitem automatically, when it is clicked.

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/MenuItem.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/menu/MenuItem.html#>
- [3] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/MenuItem.html#setAutodisable\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/MenuItem.html#setAutodisable(java.lang.String))
- [4] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/MenuItem.html#setDisabled\(boolean\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/MenuItem.html#setDisabled(boolean))
- [5] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/MenuItem.html#setHref\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/MenuItem.html#setHref(java.lang.String))
- [6] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/MenuItem.html#setTarget\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/MenuItem.html#setTarget(java.lang.String))
- [7] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/Executions.html#sendRedirect\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/Executions.html#sendRedirect(java.lang.String))
- [8] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/MenuItem.html#setUpload\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/MenuItem.html#setUpload(java.lang.String))
- [9] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/UploadEvent.html#getMedias\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/UploadEvent.html#getMedias())

Menupopup

Menupopup

- Demonstration: Menu ^[1]
- Java API: Menupopup ^[1]
- JavaScript API: Menupopup ^[2]
- Style Guide: Menupopup

Employment/Purpose

A container is used to display menus. It should be placed inside a Menu.

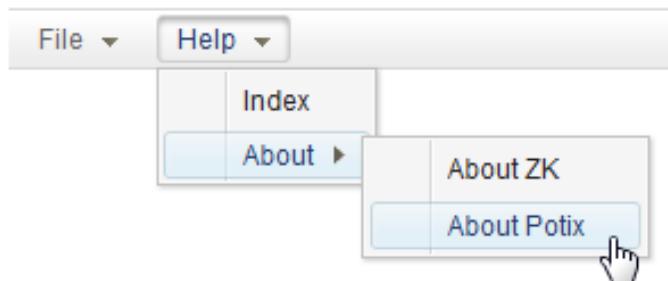
Supported event: `onOpen`.

Note: to have better performance, `onOpen` is sent only if non-deferrable event listener is registered (see `Deferrable`).

To load the content dynamically, you can listen to the `onOpen` event, and then create `MenuItem` when `OpenEvent.isOpen()` is true.

Default `HtmlBasedComponent.getSclass(): menupopup`.

Example



```
<menubar id="menubar">
    <menu label="File">
        <menupopup onOpen="alert(self.id)">
            <MenuItem label="New" onClick="alert(self.label)" />
            <MenuItem label="Open" onClick="alert(self.label)" />
            <MenuItem label="Save" onClick="alert(self.label)" />
            <menuseparator />
            <MenuItem label="Exit" onClick="System.exit(0)" />
        </menupopup>
    </menu>
    <menu label="Edit">
        <menupopup>
            <MenuItem label="Cut" onClick="alert(self.label)" />
            <MenuItem label="Copy" onClick="alert(self.label)" />
            <MenuItem label="Paste" onClick="alert(self.label)" />
            <menuseparator />
            <MenuItem label="Delete" onClick="alert(self.label)" />
        </menupopup>
    </menu>
    <menu label="View">
        <menupopup>
            <MenuItem label="Zoom In" onClick="alert(self.label)" />
            <MenuItem label="Zoom Out" onClick="alert(self.label)" />
            <menuseparator />
            <MenuItem label="Status Bar" checked="checked" onClick="alert(self.label)" />
        </menupopup>
    </menu>
    <menu label="Help">
        <menupopup>
            <MenuItem label="Index" onClick="alert(self.label)" />
            <MenuItem label="About" onClick="alert(self.label)" />
        </menupopup>
    </menu>
</menubar>
```

```

        <menupopup>
            <menuitem label="Exit" onClick="alert(self.label)" />
        </menupopup>
    </menu>
    <menu label="Help">
        <menupopup>
            <menuitem label="Index" onClick="alert(self.label)" />
            <menu label="About">
                <menupopup>
                    <menuitem label="About ZK" onClick="alert(self.label)" />
                    <menuitem label="About Potix" onClick="alert(self.label)" />
                </menupopup>
            </menu>
        </menupopup>
    </menu>
</menubar>

```

Toggle Menupopup

Since 7.0.1

If you assign a menupopup to a target component and add `type=toggle` to its popup or context attribute, it will toggle the visibility of menupopup by click. That means if you click the target component, it will cause the menupopup to show up, click on the target component again will hide the menupopup.

```

<button label="left click" popup="mp, type=toggle"/>
<menupopup id="mp">
    <menuitem label="menupopup"/>
</menupopup>

```

Highlight position

Since 8.6.0

We can highlight position in a menupopup by using `setActive(int)` method. Notice that we can only highlight `menuitem` or `menu` that is neither disabled nor invisible.

The `setActive` will not cause a menupopup to be opened. An explicit `open` is needed if the menupopup is not showed.

```

<button label="Highlight Index" onClick="mnuHelp.open(); mnuHelp.getMenupopup().setActive(0);" />
<menubar>
    <menu label="Help" id="mnuHelp">
        <menupopup>
            <menuitem label="Index" onClick="alert(self.label)" />
            <menu label="About">
                <menupopup>
                    <menuitem label="About ZK" onClick="alert(self.label)" />
                    <menuitem label="About Potix" onClick="alert(self.label)" />
                </menupopup>
            </menu>
        </menupopup>
    </menu>
</menubar>

```

```
</menupopup>
</menu>
</menubar>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: Popup

Supported Children

* Menu , MenuItem , Menuseparator

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content
7.0.1	Dec 2013	Menupopup support toggle type [3]
8.6.0	Oct 2018	ZK-3551: Menupopup active/highlight position from serverside [4]

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Menupopup.html#>

[2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/menu/Menupopup.html#>

[3] <http://tracker.zkoss.org/browse/ZK-2049>

[4] <http://tracker.zkoss.org/browse/ZK-3551>

Menuseparator

Menuseparator

- Demonstration: Menu ^[1]
- Java API: Menuseparator ^[1]
- JavaScript API: Menuseparator ^[2]
- Style Guide: Menuseparator

Employment/Purpose

Used to create a separator between menu items..

Example



```
<menubar>
  <menu label="File">
    <menupopup>
      <MenuItem label="New" onClick="alert(self.label)" />
      <MenuItem label="Open" onClick="alert(self.label)" />
      <MenuItem label="Save" onClick="alert(self.label)" />
      <menuseparator/>
      <MenuItem label="Exit" onClick="alert(self.label)" />
    </menupopup>
  </menu>
  <menuseparator/>
  <MenuItem label="Home" />
</menubar>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

*NONE

Use Cases

Menu

Menubar

Version History

Version	Date	Content

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Menuseparator.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/menu/Menuseparator.html#>

Nav

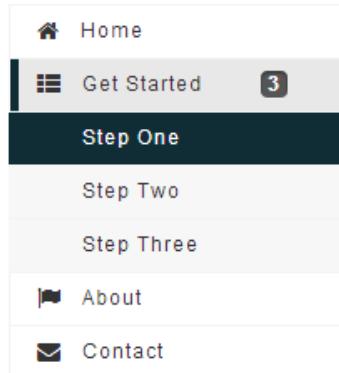
Nav

- Demonstration:
- Java API: Nav ^[1]
- JavaScript API: Nav ^[2]
- Style Guide:
- Available for ZK:
-   

Employment/Purpose

A container is used to display navitem, it should be placed inside a navbar.

Example

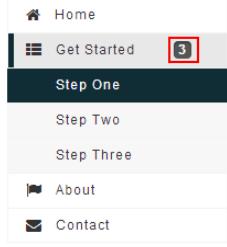


```
<navbar orient="vertical" width="200px">
    <navitem label="Home" iconSclass="z-icon-home" />
    <nav label="Get Started" iconSclass="z-icon-th-list" badgeText="3">
        <navitem label="Step One" />
        <navitem label="Step Two" />
        <navitem label="Step Three" />
    </nav>
    <navitem label="About" iconSclass="z-icon-flag" />
    <navitem label="Contact" iconSclass="z-icon-envelope"/>
</navbar>
```

Properties

Badge Text

This property set the badge text for the Nav, it is used to present more details of Nav. For example, a Nav with label "Get Started" contains three Navitem components. If we want to let user know how much items in the Nav without opening it, we can show the children numbers of current Nav by setBadgeText(java.lang.String) ^[3] API. The code snippets as shown below:

	<pre><nav label="Get Started" iconSclass="z-icon-th-list" badgeText="3"> <navitem label="Step One" /> <navitem label="Step Two" /> <navitem label="Step Three" /> </nav></pre>
---	--

Supported Events

Name	Event Type
onOpen	Event: OpenEvent ^[3] Denotes user has opened or closed a nav component.

- Inherited Supported Events: LabelImageElement

Supported Children

* Nav, Navitem, Navseparator

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
7.0.0	August, 2013	Nav ^[1] was introduced.

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Nav.html>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/nav/Nav.html>
- [3] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Nav.html#setBadgeText\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Nav.html#setBadgeText(java.lang.String))

Navbar

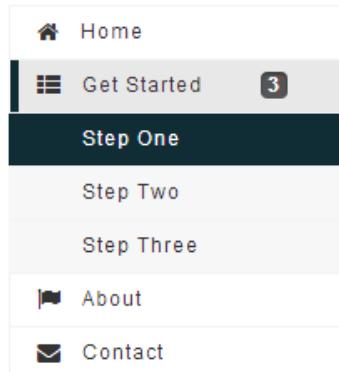
Navbar

- Demonstration:
- Java API: Navbar^[1]
- JavaScript API: Navbar^[2]
- Style Guide:
- Available for ZK:
- CE PE EE

Employment/Purpose

Provide a roadmap to help user navigate through website. It's a container that usually contains nav elements.

Example

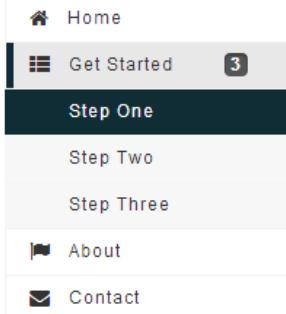


```
<navbar orient="vertical" width="200px">
    <navitem label="Home" iconSclass="z-icon-home" />
    <nav label="Get Started" iconSclass="z-icon-th-list" badgeText="3">
        <navitem label="Step One" />
        <navitem label="Step Two" />
        <navitem label="Step Three" />
    </nav>
    <navitem label="About" iconSclass="z-icon-flag" />
    <navitem label="Contact" iconSclass="z-icon-envelope"/>
</navbar>
```

Properties

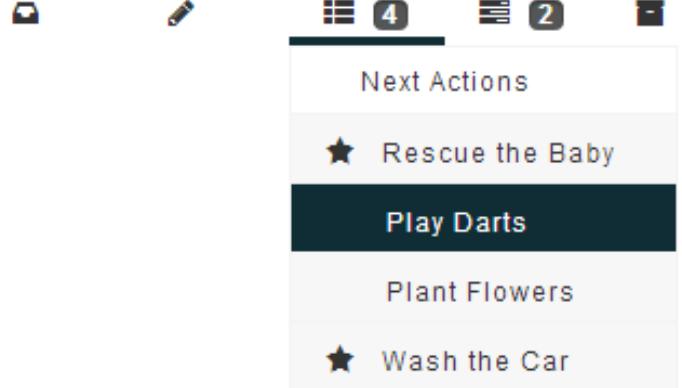
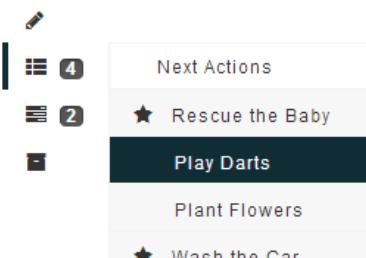
Orient

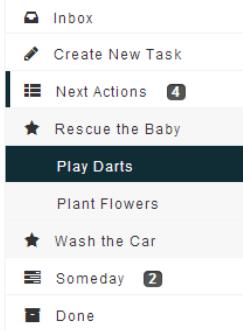
A navbar could be placed in a vertical or horizontal orientation, the `orient` attribute decides.

Orient	Snapshot
horizontal	
vertical	

Collapsed

A navbar can be collapsed, the `collapsed` attribute decides.

Collapsed	Orient	Snapshot
true	horizontal	
false	horizontal	
true	vertical	

false	vertical	
-------	----------	--

Autoclose

[since 8.0.4]

By default only a single nav-element is open at any time - automatically closing other nav-elements which are not on the current open path. This behavior can be disabled setting `autoclose="false"`, which keeps nav elements open until they are clicked again by the user.

```
<navbar orient="vertical" autoclose="false">
  <nav label="nav 1">
    <navitem label="nav 1.1"/>
    <navitem label="nav 1.2"/>
  </nav>
  <nav label="nav 2">
    <navitem label="nav 2.1"/>
    <navitem label="nav 2.2"/>
  </nav>
</navbar>
```

Supported Events

Name	Event Type
onSelect	Event: SelectEvent [6] Notifies one that the user has selected a navitem in the navbar.

- Inherited Supported Events: LabelImageElement

Supported Children

* Nav, Navitem, Navseparator

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
7.0.0	August, 2013	Navbar [1] was introduced.

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Navbar.html>

[2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/nav/Navbar.html>

Navitem

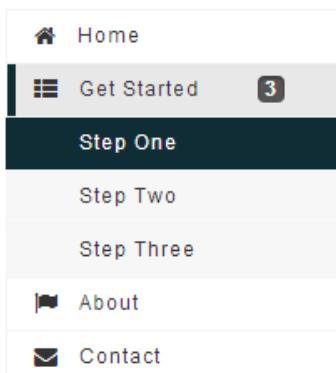
Navitem

- Demonstration:
- Java API: Navitem [1]
- JavaScript API: Navitem [2]
- Style Guide:
- Available for ZK:
- CE PE EE

Employment/Purpose

A single choice in a Navbar or Nav element. It acts much like a button but it is rendered on a navbar.

Example



```
<navbar orient="vertical" width="200px">
    <navitem label="Home" iconSclass="z-icon-home" />
    <nav label="Get Started" iconSclass="z-icon-th-list" badgeText="3">
        <navitem label="Step One" />
        <navitem label="Step Two" />
        <navitem label="Step Three" />
    </nav>
    <navitem label="About" iconSclass="z-icon-flag" />
    <navitem label="Contact" iconSclass="z-icon-envelope"/>
</navbar>
```

Properties

Href

In addition to handling the `onClick` event, you could specify the URL in the `href` property (`NavItem.setHref(java.lang.String)` ^[3]), such that the browser will navigate to the URL you specified directly (without sending back any request to the server). If you prefer to visit the URL in another browser window, you could specify the name in `NavItem.setTarget(java.lang.String)` ^[4] (just like using a HTML A tag).

Notice that the end user could hold the `Control` key and click on the `navitem` to visit the link in a new browser window (like a HTML A tag does).

Href and the `onClick` Event

There are two ways to add behavior to a `navitem`. Firstly, you can specify a listener for the `onClick` event. Secondly, you could specify a URL for the `href` property (`NavItem.setHref(java.lang.String)` ^[3]). If both are specified, the `href` property has the higher priority, i.e., the `onClick` event won't be sent.

```
<navbar>
    <navitem label="click me" onClick="do_something_in_Java()"/>
    <navitem label="don't click that one, click me" href="/another_page.zul"/>
</navbar>
```

Href and SendRedirect

The `href` property is processed at the client. In other words, the browser will jump to the URL specified in the `href` property, so your application running on the server has no chance to process it.

If you have to process it on the server or you have to decide whether to jump to another URL based on certain condition, you could listen to the `onClick` event, process it, and then invoke `Executions.sendRedirect(java.lang.String)` ^[7] if it jumps to another URL.

For end users, there is no difference between the use of `NavItem.setHref(java.lang.String)` ^[3] and `Executions.sendRedirect(java.lang.String)` ^[7].

```
<navbar>
    <navitem label="redirect" onClick="Executions.sendRedirect("another.zul")" />
    <navitem label="href" href="another.zul"/>
</navbar>
```

Since the `onClick` event is sent to the server for processing, you are able to perform additional tasks before invoking `Executions.sendRedirect(java.lang.String)` ^[7], such as redirecting to another page only if certain conditions are

satisfied.

On the other hand, the `href` property is processed at the client side. Your application won't be notified when users click the navitem.

Badge Text

This property set the badge text for the Navitem, it is used to present more details of Navitem.

```
<navitem label="Step One" badgeText="1"/>
```

Supported Events

Name	Event Type

- Inherited Supported Events: LabelImageElement

Supported Children

*NONE

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content
7.0.0	August, 2013	Navitem [1] was introduced.

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Navitem.html>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/nav/Navitem.html>
- [3] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Navitem.html#setHref\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Navitem.html#setHref(java.lang.String))
- [4] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Navitem.html#setTarget\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Navitem.html#setTarget(java.lang.String))

Navseparator

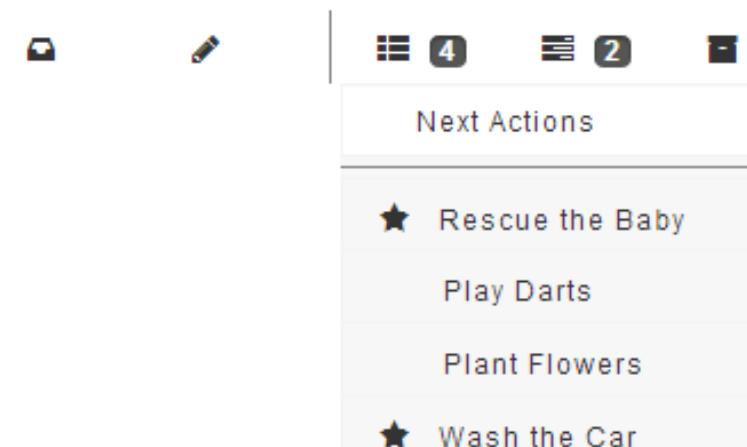
Navseparator

- Demonstration:
- Java API: Navseparator^[1]
- JavaScript API: Navseparator^[2]
- Style Guide:
- Available for ZK:
- CE PE EE

Employment/Purpose

Used to create a separator between nav items..

Example



The screenshot shows a ZK application's navigation bar. On the left, there are icons for a folder, a pencil, and a search bar. In the center, there are two items: 'Inbox' with a badge of '4' and 'Create New Task' with a badge of '2'. Below these is a 'Navseparator' element. To its right is a 'nav' element with the label 'Next Actions'. This 'nav' element contains four 'navitem' elements: 'Rescue the Baby' (with a star icon), 'Play Darts', 'Plant Flowers', and 'Wash the Car' (with a star icon). The 'Rescue the Baby' item has a badge of '4' above it.

```
<navbar orient="horizontal" collapsed="true">
    <navitem label="Inbox" iconSclass="z-icon-inbox" />
    <navitem label="Create New Task" iconSclass="z-icon-pencil"/>
    <navseparator/>
    <nav label="Next Actions" iconSclass="z-icon-th-list" badgeText="4">
        <navseparator/>
        <navitem label="Rescue the Baby" iconSclass="z-icon-star"/>
        <navitem label="Play Darts" />
        <navitem label="Plant Flowers" />
        <navitem label="Wash the Car" iconSclass="z-icon-star"/>
    </nav>
    <nav label="Someday" iconSclass="z-icon-tasks" badgeText="2"/>
    <nav label="Done" iconSclass="z-icon-archive"/>
</navbar>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

*NONE

Version History

Version	Date	Content
7.0.0	October, 2013	Navseparator ^[1] was introduced.

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Navseparator.html>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/nav/Navseparator.html>

Popup

Popup

- Demonstration: Tooltips and Popup ^[1]
- Java API: Popup ^[2]
- JavaScript API: Popup ^[3]
- Style Guide: Popup

Employment/Purpose

The popup is a container that does not have any special frame. You can associate a popup with any component by specifying the popup's id at one of the following attributes:

```
<button popup="popup_id"/>
<button tooltip="popup_id"/>
<button context="popup_id"/>

<popup id="popup_id">
    this is a popup
</popup>
```

Example

Tootip for Another Ponup

ZK simply rich.
ZK your killer Web application now!

```
<separator bar="true" />
<label value="Tootip for Another Ponup" tooltip="any" />
<popup id="any" width="300px">
    <vbox>
        ZK simply rich.
        <toolbarbutton label="ZK your killer Web application now!" href="http://www.zkoss.org" />
    </vbox>
</popup>
```

ZK simply rich.

ZK your killer Web application now!

```
<textbox popup="popup, position=after_start" />
<popup id="popup" width="300px">
    <vbox>
        ZK simply rich.
        <toolbarbutton label="ZK your killer Web application now!" href="http://www.zkoss.org" />
    </vbox>
</popup>
```

Position

You can simply specify a popup's position when attaching to a component by

- built-in position
- x, y coordinate

```
<button popup="popup_id, position=overlap_end" />
<button popup="popup_id, x=50, y=50" />
```

since 6.0.1

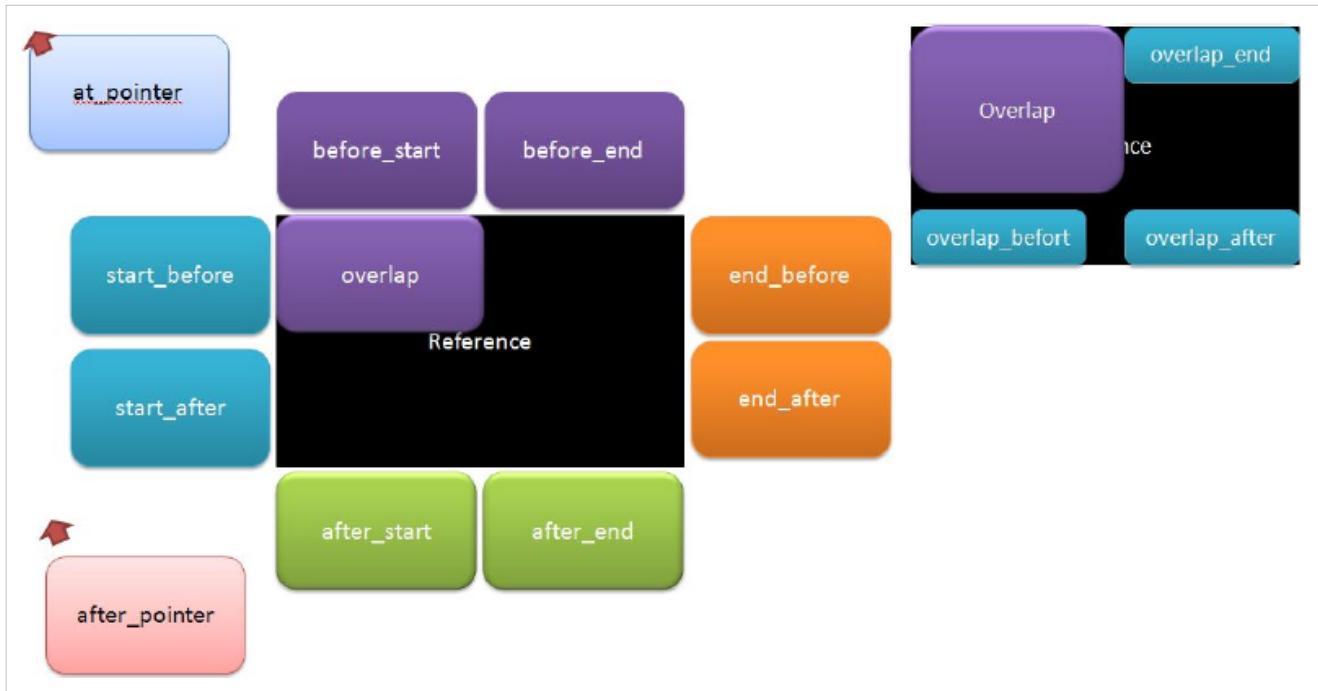
ZK supports the following position string:



overlap, overlap_end, overlap_before, overlap_after are kept (still available) for backward compatibility. They are identical with top_left, top_right, bottom_left, and bottom_right, respectively.

Before 6.0.0

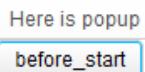
The 14 possible positions are provided below:



The following illustrates the simplicity of usage,

```
<popup id="pp">  
    Here is popup  
</popup>  
<button label="before_start" onClick='pp.open(self, "before_start");' />
```

Upon clicking the button the popup component will appear in the relative position specified. In this case the position is just above the button.



Methods

There are several overloading open() methods available, please check javadoc ^[4]

Toggle Popup

Since 7.0.0

If a popup/context menu is assigned to a target component via the popup/context attribute, the popup up will show up when the user clicks on the target component. Click the target component again, the popup will still show up by default. As of 7.0.0, The popup/context attribute supports additional toggle type, which could make target component act as a toggle switcher. If the popup has not shown up yet, click the target component will cause the popup to show up. If the popup is showing up, click on the target component again will toggle the popup to hide. The usage is in below code.

```
<button label="Popup" popup="id, type=toggle"/>
```

Tooltip Delay

The tooltip attribute can also support a delay, the following code outlines how to accomplish this.

```
<label value="Tooltip" tooltip="popup_id, position=before_start, delay=500"/>
```

Supported Events

Name	Event Type
onOpen	Event: OpenEvent ^[3] Denotes a Popup has been opened or closed (in this case OpenEvent::isOpen() returns false).

- Inherited Supported Events: XulElement

Supported Children

* ALL

Use Cases

Version	Description	Example Location
3.6	Smalltalk: Toolbar and Menus	ZK Developer's Reference: Tooltips, Context Menus and Popups
3.6	A way to specify the position of the Popup component	New Features of ZK 3.6.1
3.6	Popup, tooltip and context positions	New Features of ZK 3.6.3

Version History

Version	Date	Content
7.0.0	Nov 2013	Popup support toggle type

References

- [1] <http://www.zkoss.org/zkdemo/popup>
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Popup.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Popup.html#>
- [4] <https://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Popup.html>

Progressmeter

Progressmeter

- Demonstration: Progressmeter ^[1]
- Java API: Progressmeter ^[2]
- JavaScript API: Progressmeter ^[3]
- Style Guide: Progressmeter

Employment/Purpose

A progress meter is a bar that indicates how much of a task has been completed. The value property must be in the range between 0 and 100.

Example



```
<progressmeter value="10"/>
```

Properties

Indeterminate

since 8.6.1

If true, the progressmeter will show an indeterminate animation and the real value of the progressmeter will be hidden.(default false)

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

*NONE

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
8.6.1	Jan, 2019	ZK-3629 [4]: use the progressmeter to indicate a long operation is so hard

References

- [1] http://www.zkoss.org/zkdemo/effects/upload_effect
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Progressmeter.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Progressmeter.html#>
- [4] <https://tracker.zkoss.org/browse/ZK-3629>

Rating

Rating

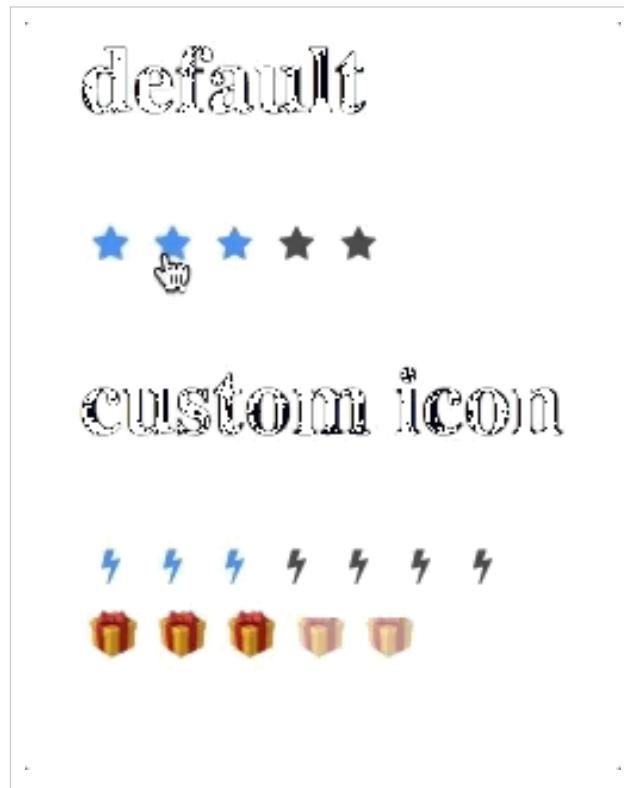
- Java API: Rating ^[1]
- JavaScript API: Rating ^[2]

[since 8.6.0]

Employment/Purpose

The rating component is a component that allows user selecting an rate that is smaller than the maximum number.

Example



```
<style>
    .myGiftIcon:before {
        content: '\f2c8';
    }
    .gift .z-rating-icon{
        opacity: 0.5;
    }
    .gift .z-rating-selected{
        opacity: 1;
    }
</style>
```

```
<vlayout style="margin: 20px">
  <n:h1>default</n:h1>
  <rating rating="3"/>
  <n:h1>custom icon</n:h1>
  <rating iconSclass="z-icon-bolt" rating="3" max="7"/>
  <div sclass="gift">
    <rating iconSclass="myGiftIcon" rating="3"/>
  </div>
</vlayout>
```

Properties and Features

IconSclass

Specify the sclass name of the rating icon.

Orient

The orientation is default to horizontal, could be changed to vertical if vertical is specified.

Rating

This is the rating value, will have a initial value if specified to an integer larger than 0.

Cancelable

If true, by clicking the previous rated icon again, the rating will be canceled and set 0.

Max

Represents the maximum number of the rating. Also, icons will be rendered as the max size.

Disabled

If disabled is true, it's not allowed to be rated. (Is allowed to have an initial rating.)

Readonly

If true, the rating is only readable, not changeable. (Is allowed to have an initial rating.)

Supported Events

Name	Event Type
onChange	Event: Event [7] Denotes user has rated.

- Inherited Supported Events: XulElement

Supported Children

* none

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Rating.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/wgt/Rating.html#>

Selectbox

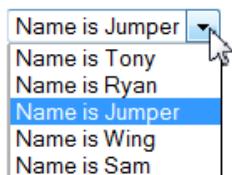
Selectbox

- Demonstration: selection_dropdown ^[1]
- Java API: Selectbox ^[2]
- JavaScript API: Selectbox ^[3]
- Style Guide: N/A

Employment/Purpose

Selectbox is a lightweight dropdown list and it can support ListModel, Renderer, and Databinding as well. The benefit of it is not to create child widgets for each data, so the memory usage is much lower at the server.

Example



```
<zk>
<zscript>
<! [CDATA[
String[] userName = { "Tony", "Ryan", "Jumper", "Wing",
"Sam" } ;
ListModelList model = new ListModelList(userName);
]]></zscript>
<selectbox model="${model}" onSelect='alert(model.getData());'>
<template name="model">
Name is ${each}
</template>
</selectbox>
```

```
</zk>
```

To give the selectbox an initial value, for example, Tony, add the following code after the model is created:

```
model.addToSelection ("Tony");
```

Data binding

Here is the MVVM way:

```
<zscript><! [CDATA[  
    public class MyUserBean {  
        private String[] userList = { "Tony", "Ryan", "Jumper", "Wing",  
        "Sam" };  
        private int index = 0;  
  
        public ListModelList getUserList() {  
            return new ListModelList( Arrays.asList(userList) );  
        }  
  
        public void setUserList() {  
        }  
  
        public void setIndex(int ind) {  
            index = ind;  
        }  
  
        public int getIndex() {  
            return index;  
        }  
    }  
    MyUserBean mybean = new MyUserBean();  
/** Implements ItemRenderer without using template  
    org.zkoss.zul.ItemRenderer render = new  
org.zkoss.zul.ItemRenderer() {  
    public String render(Component owner, Object data, int index)  
throws Exception {  
        return data.toString();  
    }  
};  
*/  
]]></zscript>  
<div apply="org.zkoss.bind.BindComposer">  
    Select User:  
    <selectbox id="box" model="@init(mybean.userList)"  
              selectedIndex="@bind(mybean.index)">  
        <template name="model">${each}</template>
```

```
</selectbox>

Selected:
<label id="val" value="@load(mybean.index)" />
</div>
```

Supported Events

Name	Event Type
onSelect	Event: SelectEvent [6] Notifies one that the user has selected a new item in the selectbox.

- Inherited Supported Events: HtmlBasedComponent

Supported Children

*NONE

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
6.0.0	October 4, 2011	Add the new Selectbox component
6.0.0-RC2	December 6, 2011	Rename OptionRenderer to ItemRenderer

References

- [1] https://www.zkoss.org/zkdemo/getting_started/selection_dropdown
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul>Selectbox.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt>Selectbox.html#>

Separator

Separator

- Demonstration: N/A
- Java API: Separator^[1]
- JavaScript API: Separator^[2]
- Style Guide: Separator

Employment/Purpose

A separator is used to insert a space between two components. There are several ways to customize the separator.

- By use of the orient attribute, you are able to specify whether the separator is vertical or horizontal. By default it is a horizontal separator, which inserts a line break. On the other hand, a vertical separator inserts white space.
- By use of the bar attribute, you can control whether to show a horizontal or vertical line between components.
- By use of the spacing attribute, you can control the size of spacing.

Example

```
line 1 by separator  
line 2 by separator  
line 3 by separator | another piece
```

```
line 4 by separator | another piece
```

```
line 1 by separator  
<separator />  
line 2 by separator  
<separator />  
line 3 by separator  
<space bar="true" />  
another piece  
<separator spacing="20px" />  
line 4 by separator  
<space bar="true" spacing="20px" />  
another piece
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

*NONE

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Separator.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Separator.html#>

Space

Space

- Demonstration: N/A
- Java API: Space ^[1]
- JavaScript API: Space ^[2]
- Style Guide: Separator

Employment/Purpose

Space is a Separator with the orient default to "horizontal". In other words, <space> is equivalent to <separator orient="horizontal">

Example

|||

```
<zkc>
  <space bar="true"/>
  <space bar="true"/>
  <space bar="true"/>
  <space bar="true"/>
</zkc>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: Separator

Supported Children

*NONE

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Space.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Space.html#>

Script

Script

- Demonstration: Script ^[1]
- Java API: Script ^[1]
- JavaScript API: Script ^[2]
- Style Guide: N/A

Employment/Purpose

The script component is used to specify the script codes running at the browser. Notice that, unlike zscript, the script codes are running at the browser. They are usually written in JavaScript which is supported by the most of browsers. The simplest format is as follows.

Example

```
change color
```

```
<zk>
    <window id="win">
        <button label="change color" onClick='Clients.evalJavaScript("myfunc()")' />
    </window>
    <script type="text/javascript">
        function myfunc() {
            jq("$win").css("backgroundColor", "blue");
        }
    </script>
</zk>
```

Alternatives

Instead of using the script component, you could use the script directive instead. It does not support defer, but it is more efficient since no component is created.

```
<?script src="~/js/zk.debug.wpd"?>  
<?script content="jq.IE6_ALPHAFIX='.png';"?>
```

where the first statement loads the debug utility and the second generates JavaScript code snippet directly.

Another alternative is HTML SCRIPT. For example, we could define global variables and functions as follows

```
<n:script xmlns:n="native"><!-- use the native namespace -->  
    var a_global_variable;  
    function a_global_function () {  
        alert("native script");  
    }  
    alert("you can not access this as widget but evaluated  
immediately");  
</n:script>
```

Defer the Evaluation

By default, the specified JavaScript code will be evaluated as soon as the page is loaded. There is an attribute called defer. By specifying true, the JavaScript code won't be evaluated until all widgets are created and bound to the DOM tree.

```
<textbox id="inp"/>  
<script defer="true">  
    this.$f("inp").setValue("initialized");  
</script>
```

The defer attribute can be used with a JavaScript file as shown below. Then, the JavaScript file will be loaded after all widgets are created and bound to the DOM tree.

```
<script src="/js/foo.js" defer="true"/>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: AbstractComponent

Supported Children

*NONE

Use Cases

Version	Description	Example Location
5.0	Overview and Tutorial	Client Side Programming ZK Client-side Reference: General Control

Version History

Version	Date	Content

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Script.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/util/Script.html#>

Style

Style

- Demonstration: Style ^[1]
- Java API: Style ^[2]
- JavaScript API: Style ^[3]
- Style Guide: N/A

Employment/Purpose

The style component used to specify CSS styles for the owner desktop.

Note:

- a `style` component can appear anywhere in a ZUML page, but it affects all components in the same desktop.
- `getContent()` simply get the string that is set by `setContent()`. If you call `setSrc()` or call the constructor of `Style()`, `getContent()` still gets null.

Example

```
+ <script type="text/javascript">
- <style id="z_c6_2" type="text/css">
  1
  2  a{
  3      color:red;
  4  }
  5
</style>
+ <script type="text/javascript" charset="UTF-8"
<style> a{ color:red; }</style>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: AbstractComponent

Supported Children

*NONE

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content
5.0.3	June 2010	The media property is supported.

References

- [1] http://www.zkoss.org/zkdemo/styling/custom_style
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Style.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/util/Style.html#>

Timer

Timer

- Demonstration: Timer ^[1]
- Java API: Timer ^[2]
- JavaScript API: Timer ^[3]
- Style Guide: N/A

Employment/Purpose

Timer is a special component that is invisible. It fires one or more org.zkoss.zk.ui.event.Event after a specified delay, notice that the timer won't fire any event until it is attached to a page.

Example

```
<label id="now" />
<timer id="timer" delay="1000" repeats="true"
       onTimer="now.setValue(new Date().toString())" />
```

Supported Events

Name	Event Type
onTimer	Event: Event ^[7] Denotes the timer you specified has triggered an event. To know which timer, invoke the <code>getTarget</code> method in the Event class.

- Inherited Supported Events: HtmlBasedComponent

Supported Children

*NONE

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content

References

- [1] <http://www.zkoss.org/zkdemo/userguide/#u3>
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Timer.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/utl/Timer.html#>

Toolbar

Toolbar

- Demonstration: Toolbar ^[1]
- Java API: Toolbar ^[6]
- JavaScript API: Toolbar ^[2]
- Style Guide: Toolbar

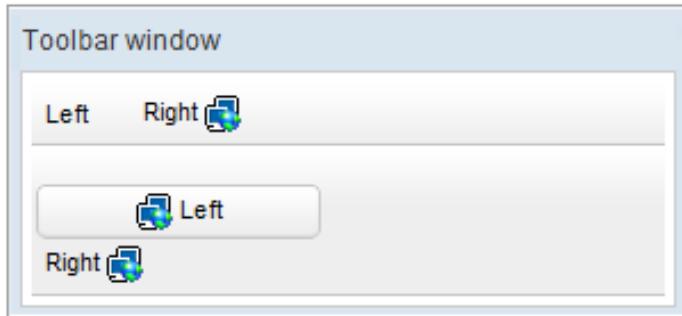
Employment/Purpose

A toolbar is used to place a series of buttons, such as `toolbarbutton` or `button`. The toolbar buttons could be used without toolbars, so a toolbar could be used without tool buttons. However, the tool buttons change their appearance if they are placed inside a toolbar.

The toolbar has two orientation: `horizontal` and `vertical`. It controls how the buttons are placed.

See also : Button, Toolbarbutton

Example



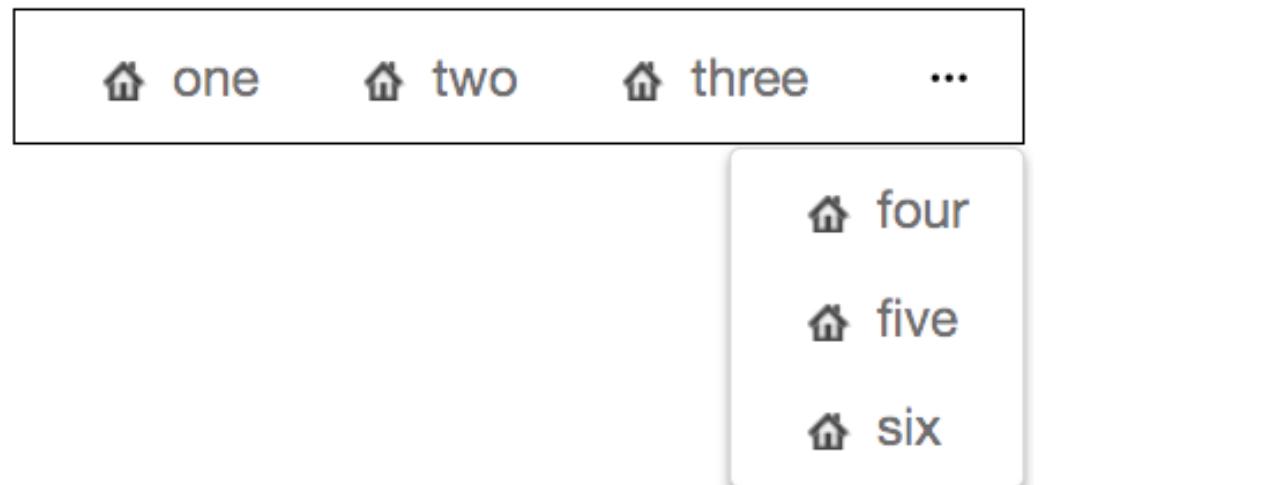
```
<window title="Toolbar window" border="normal" width="300px">
    <toolbar>
        <toolbarbutton label="Left" />
        <space />
        <toolbarbutton label="Right" image="/img/network.gif"
            dir="reverse" />
    </toolbar>
    <toolbar orient="vertical">
        <button label="Left" image="/img/network.gif" width="125px" />
        <toolbarbutton label="Right" image="/img/network.gif"
            dir="reverse" />
    </toolbar>
</window>
```

overflowPopup

[since 8.6.0]

When overflowPopup="true", a toolbar will have a . . . symbol that shows a popup which contains those buttons weren't able to fit in the toolbar.

Default: false.



```
<toolbar overflowPopup="true" width="350px" style="border: 1px black solid;">
    <toolbarbutton label="one" iconSclass="z-icon-home"/>
    <toolbarbutton label="two" iconSclass="z-icon-home"/>
    <toolbarbutton label="three" iconSclass="z-icon-home"/>
    <toolbarbutton label="four" iconSclass="z-icon-home"/>
    <toolbarbutton label="five" iconSclass="z-icon-home"/>
    <toolbarbutton label="six" iconSclass="z-icon-home"/>
</toolbar>
```

overflowPopupIconSclass

[since 9.6.0]

When `overflowPopup="true"`, you can customize a toolbar ... symbol just specify the `overflowPopupIconSclass` attribute. For a complete list of icons, please refer to FontAwesome Cheatsheet [5].

Default: ... symbol

```
<toolbar overflowPopup="true" overflowPopupIconSclass="z-icon-plus-square" width="350px" style="border: 1px black solid;">

<toolbarbutton label="one" iconSclass="z-icon-home"/>
<toolbarbutton label="two" iconSclass="z-icon-home"/>
<toolbarbutton label="three" iconSclass="z-icon-home"/>
<toolbarbutton label="four" iconSclass="z-icon-home"/>
<toolbarbutton label="five" iconSclass="z-icon-home"/>
<toolbarbutton label="six" iconSclass="z-icon-home"/>

</toolbar>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Molds

Available molds of a component are defined in lang.xml embedded in zul.jar. It is suggested to set mold to panel while toolbar is in the footer of a panel.

Name	Snapshot
default	Left Right
panel	Left Right

Supported Children

* ALL

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content

References

- [1] <http://www.zkoss.org/zkdemo/menu/toolbar>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Toolbar.html#>

Toolbarbutton

Toolbarbutton

- Demonstration: Toolbar ^[1] and Fileupload
- Java API: Toolbarbutton ^[1]
- JavaScript API: Toolbarbutton ^[2]
- Style Guide: Toolbarbutton

Employment/Purpose

The behavior of Toolbarbutton is similar to the button except the appearance is different. The button component uses HTML BUTTON tag, while the toolbarbutton component uses HTML DIV tag.

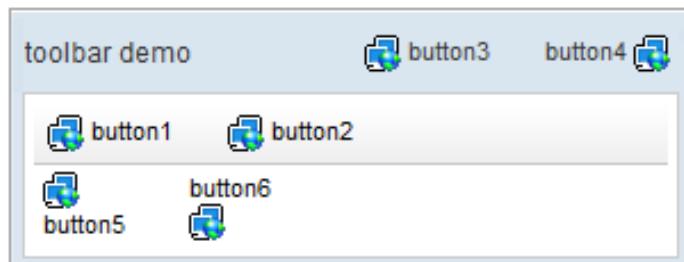
A toolbarbutton could be placed outside a toolbar. However toolbarbuttons change their appearance if they are placed inside a toolbar.

Toolbarbutton supports `getHref()`. If `getHref()` is not null, the `onClick` handler is ignored and this element is degenerated to HTML's A tag.

Within ZK 5, the file upload has been redesigned so it can be integrated with any widget. For example, the toolbarbutton can now be used to upload a file. In addition to this, the display of the upload status has been enhanced and can be customized easily.

See also : Button, Toolbar

Example



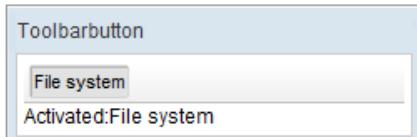
```
<window title="toolbar demo" border="normal" width="300px">
    <caption>
        <toolbarbutton label="button3" image="/img/network.gif" />
        <space />
        <toolbarbutton label="button4" image="/img/network.gif"
            dir="reverse" />
    </caption>
    <toolbar>
        <toolbarbutton label="button1" image="/img/network.gif" />
        <space />
        <toolbarbutton label="button2" image="/img/network.gif" />
    </toolbar>
    <hbox>
        <toolbarbutton label="button5" image="/img/network.gif" />
        <toolbarbutton label="button6" image="/img/network.gif" />
    </hbox>
</window>
```

```

        orient="vertical" />
    <space />
    <toolbarbutton label="button6" image="/img/network.gif"
        orient="vertical" dir="reverse" />
</hbox>
</window>
```

Toggle Mode

Toolbarbutton support toggle mode since ZK 6.0.0 . (mode="toggle")



In the toggle mode , the toolbarbutton will be persistent checked after user clicked it, and will be released after next click. It will fire CheckEvent [5] when state changed.

```

<window title="Toolbarbutton" border="normal" width="250px" >
    <toolbar >
        <toolbarbutton label="File system" mode="toggle" >
            <attribute name="onCheck"><![CDATA[
                if(event.isChecked()) {
                    result.setValue("Activated:"+event.getTarget().getLabel());
                }else{
                    result.setValue("Deactivated:"+event.getTarget().getLabel());
                }
            ]]></attribute>
        </toolbarbutton>
    </toolbar>

    <label id="result" />
</window>
```

File Upload

Any toolbarbutton^[3] can be used to upload files. All you need to do is:

1. Specify the upload attribute with true
2. Handles the onUpload event.

```
<toolbarbutton upload="true" label="Fileupload" onUpload="myProcessUpload(event.getMedia())" />
```

When the file is uploaded, an instance of UploadEvent^[5] is sent to the button. Then, the event listener can retrieve the uploaded content by examining the return value of UploadEvent.getMedia()^[6].

-
- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Toolbarbutton.html#>
 - [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Toolbarbutton.html#>
 - [3] Any Toolbarbutton (<http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Toolbarbutton.html#>) can be used to upload files too.

Supported Events

Name	Event Type
onCheck	Event: CheckEvent (http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/CheckEvent.html#) Denotes when toolbarbutton is checked , only available in toggle mode . (since ZK 6.0.0)

- Inherited Supported Events: Button

Supported Children

*NONE

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content
6.0.0	2/10	Introduce Toggle Mode to Toolbarbutton

Input

This section outlines components which are used to input application data.

Bandbox

Bandbox

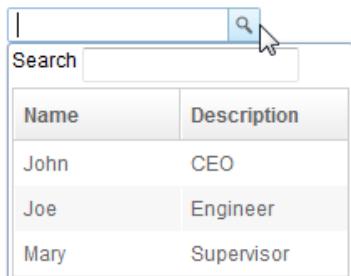
- Demonstration: Bandbox ^[1]
- Java API: Bandbox ^[2]
- JavaScript API: Bandbox ^[3]
- Style Guide: Bandbox

Employment/Purpose

A bandbox is a special text box that embeds a customizable popup window (aka., a dropdown window). Like comboboxes, a bandbox consists of an input box and a popup window. The popup window is opened automatically, when users presses Alt+DOWN or clicks the magnifier button.

Unlike comboboxes, the popup window of a bandbox could be anything. It is designed to give developers the maximal flexibility. A typical use is to represent the popup window as a search dialog.

Example



```
<bandbox id="bd">
    <bandpopup>
        <vbox>
            <hbox>
                Search
                <textbox />
            </hbox>
            <listbox width="200px"
onSelect="bd.value=self.selectedItem.label;bd.close();">
                <listhead>
                    <listheader label="Name" />
                    <listheader label="Description" />
                </listhead>
                <listitem>
```

```
        <listcell label="John" />
        <listcell label="CEO" />
    </listitem>
    <listitem>
        <listcell label="Joe" />
        <listcell label="Engineer" />
    </listitem>
    <listitem>
        <listcell label="Mary" />
        <listcell label="Supervisor" />
    </listitem>
</listbox>
</vbox>
</bandpopup>
</bandbox>
```

Mouseless Entry bandbox

- Alt+DOWN to pop up the list.
- Alt+UP or ESC to close the list.

Properties

The Close Method

A popup window could contain any components, so it is the developer's job to close the popup and copy any needed value from it.

```
<listbox width="200px"
    onSelect="bd.value=self.selectedItem.label; bd.close();">
```

In the above example, we copy the selected item's label to the bandbox, and then close the popup.

Autodrop



By default, the popup window won't be opened until user clicks the button, or presses Alt+DOWN on the keyboard. However, you can set the `autodrop` property to true and as soon as the user types a character the popup will be opened. This is helpful for novice users, but it might be annoying for experienced users.

```
<zk>
    <bandbox id="bd" autodrop="true">
        <bandpopup>
            ...
        </bandpopup>
    </bandbox>
</zk>
```

The onOpen Event

If the user opens the popup window the `onOpen` event is sent to the application. By using the `fulfill` attribute with the `onOpen` value as shown below, you can defer the creation of the popup window.

```
<bandbox id="test">
    <bandpopup fulfill="test.onOpen">
        ...
    </bandpopup>
</bandbox>
```

Alternatively, you can prepare the popup window in Java by listening to the `onOpen` event, as depicted below.

```
<zk>
    <bandbox id="band" onOpen="prepare()"/>

    <zscript>
        void prepare()
        {
            if (band.getPopup() == null) {
                //create child elements
            }
        }
    </zscript>
</zk>
```

The onChanging Event

Since a bandbox is also a text box, you are also able to listen to an `onChanging` event. By listening to this event, you can manipulate the popup window in any fashion. The code below illustrates capturing the user key and displaying information accordingly.

```
<zk>
    <bandbox id="band" autodrop="true" onChanging="suggest()"/>
    <zscript>
        void suggest()
        {
            if (event.value.startsWith("A")) {
                //do something
            } else if (event.value.startsWith("B")) {
                //do another
            }
        }
    </zscript>
</zk>
```

Notice that, when the `onChanging` event is received, the content of the bandbox has not changed. Therefore, you cannot use the `value` property of the bandbox. Instead, you should use the `value` property of the event (`org.zkoss.zk.ui.event.InputEvent`).

Constraint

You could specify what value to accept for input controls by the use of the `constraint` property. It could be a combination of `noEmpty`, and/or a regular expression.

To specify two or more constraints, use comma to separate them as follows.

```
<bandbox constraint="no empty,/^A/" />
```

To specify a regular expression, you may have to use the character / to enclose the regular expression as follows.

```
<bandbox constraint="/^A/" />
```

Notes:

- The above statement is XML, so do *not* use \\ to specify a backslash. However typing \\ is necessary, if writing in Java.

```
new Bandbox().setConstraint("/.+.+@.+.+\\.+[a-z]+/")
```

- You are allowed to mix regular expressions with other constraints by separating them with a comma.

If you prefer to display different message to the default one, you can append the error message to the constraint with a colon.

```
<bandbox constraint="/^A/: only allowed the item start with A"/>
```

Notes:

- The error message, if specified, must be the last element and start with colon.
 - To support multiple languages, you could use the `「」` function as depicted in the **Internationalization** chapter.

```
<bandbox constraint="/^A/: ${c:l('err.startwith.required')}"/>
```

IconSclass

[Since 8.6.2]

Specify the `sclass` name of the `Bandbox` button icon. For built-in icon, please see ZK Component Reference/Base Components/`LabelImageElement`.

Inherited Functions

Please refer to Texthbox for inherited functions.

Supported Events

Name	Event Type
onOpen	<p>Event: OpenEvent [3]</p> <p>Denotes user has opened or closed a component. Note: unlike onClose, this event is only a notification. The client sends this event after opening or closing the component.</p>

- Inherited Supported Events: Textbox

Supported Molds

Available molds of a component are defined in lang.xml embedded in zul.jar.

Name	Snapshot
default	
rounded	 [Since 5.0.0]

Supported Children

* Bandpopup

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

- [1] http://www.zkoss.org/zkdemo/combobox/customizable_combobox
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Bandbox.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/inp/Bandbox.html#>

Bandpopup

Bandpopup

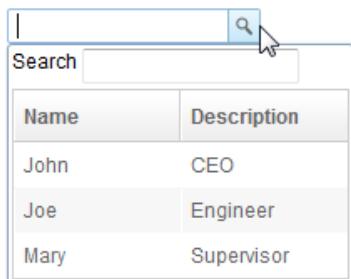
- Demonstration: Bandbox [1]
 - Java API: Bandpopup [1]
 - JavaScript API: Bandpopup [2]
 - Style Guide: Bandpopup

Employment/Purpose

The popup that belongs to a Bandbox instance.

Developers usually listen to the `onOpen` event that is sent to Bandbox and then creates proper components as children of this component.

Example



```
</listitem>
<listitem>
    <listcell label="Mary" />
    <listcell label="Supervisor" />
</listitem>
</listbox>
</vbox>
</bandpopup>
</bandbox>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

*All

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Bandpopup.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/inp/Bandpopup.html#>

Calendar

Calendar

- Demonstration: Calendar ^[1]
- Java API: Calendar ^[2]
- JavaScript API: Calendar ^[3]
- Style Guide: Calendar

Employment/Purpose

A calendar displays a 'flat' calendar and allows user to select a day from it.

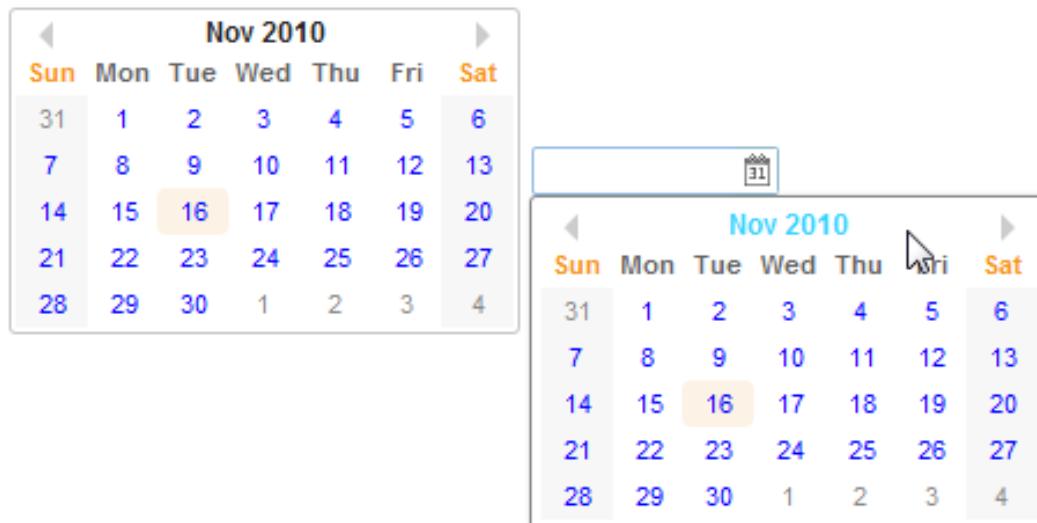
The first day of the week is decided by the locale (actually the return value of the `getFirstDayOfWeek` method in the `java.util.Calendar`).

Since 5.0.3, you can control the first day of the week by the use of the session attribute and the library property. Please refer to The First Day of the Week for details.

Customization

Since 5.0.3, the rendering of the calendar can be customized at the client by providing JavaScript code that overrides Renderer ^[4].

Example



```
<calendar id="cal" onChange="in.value = cal.value"/>
<datebox id="in" onChange="cal.value = in.value"/>
```

Date Range Selector

Start ~ End

20190522 - 20190531

May 2019							May 2019						
Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat
28	29	30	1	2	3	4	28	29	30	1	2	3	4
5	6	7	8	9	10	11	5	6	7	8	9	10	11
12	13	14	15	16	17	18	12	13	14	15	16	17	18
19	20	21	22	23	24	25	19	20	21	22	23	24	25
26	27	28	29	30	31	1	26	27	28	29	30	31	1

Check calendar.zul [5]

Calendar Day Renderer

This is achieved by overriding the default renderer at the client to customize the appearance of days on ZK's Calendar. For example,

Nov 2010						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4

```
<zk>
<script><![CDATA[
    zk.afterLoad('zul.db', function(){
        zul.db.Renderer.cellHTML = function (cal, y, m, day,
monthofs) {
            return '<a href="javascript:;" style="color:red;">' + day + '</a>';
        };
    });
]]></script>
<calendar/>
</zk>
```

[Since 5.0.3]

Show Week Number

Calendar supports to show a week number of the year.

- Available for ZK:
- **CE** **PE** **EE**

[Since 6.5.0]

With WeekOfYear

Jun 2012						
Wk	Sun	Mon	Tue	Wed	Thu	Fri
21	27	28	29	30	31	1
22	3	4	5	6	7	8
23	10	11	12	13	14	15
24	17	18	19	20	21	22
25	24	25	26	27	28	29
						30

```
<calendar weekOfYear="true" />
```

2DigitYearStart

You can control the 2DigitYearStart by the use of the library property. Please refer to org.zkoss.web.preferred.2DigitYearStart for details.

Constraint

since 8.5.2

This component also supports constraint like ZK Component Reference/Input/Datebox#Constraint

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

*NONE

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
5.0.3	June, 2010	Calendar Day Renderer
5.0.3	July, 2010	An application can control the first day of the week by use of the session attribute and the library property. Please refer to The First Day of the Week for details.
5.0.4	August, 2010	Calendar supports moving to next/prev mon by mouse scrolling.
6.5.0	June, 2012	ZK-1175 [6]: Calendar support show week number

References

- [1] http://www.zkoss.org/zkdemo/reporting/simple_calendar
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Calendar.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/db/Calendar.html#>
- [4] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/db/Renderer.html#>
- [5] <https://github.com/zkoss/zkbooks/blob/master/componentreference/src/main/webapp/input/calendar.zul#L21>
- [6] <http://tracker.zkoss.org/browse/ZK-1175>

Cascader

Cascader

- Demonstration ^[1]
- Java API: Cascader ^[2]
- JavaScript API: Cascader ^[3]
- Available for ZK:
- CE PE EE

Employment/Purpose

A Cascader is a dropdown list in a tree structure and supports TreeModel.

Example



```
<zscript><! [CDATA[DefaultTreeModel tm = new DefaultTreeModel (new DefaultTreeNode ("ROOT", Arrays.asList (new DefaultTreeNode [] { new DefaultTreeNode ("David", Arrays.asList (new TreeNode [] { new DefaultTreeNode ("Thomas", new ArrayList () ), new DefaultTreeNode ("Steven", new ArrayList ()) })) , new DefaultTreeNode ("David-1", new DefaultTreeNode ("David-2", null))) , new DefaultTreeNode ("Thomas", new ArrayList ()) , new DefaultTreeNode ("Steven", new ArrayList ()) })); ]]></zscript><cascader width="300px" model="${tm}" />
```

Users can select in layers, and the selected items are converted into text. (Default: joining by slashes, i.g. "A/B/C")

Properties

Disabled

Sets whether it is disabled. A disabled component can't interact with users.

ItemConverter

The converter generates the label text shown in the cascader. By implementing your own Converter^[4], you can generate the label that represents the selected item. The default implementation is joining all the `toString()` result of items by slashes.

ItemRenderer

See also: ZK_Developer's_Reference/MVC/View/Renderer/Cascader_Renderer

The item renderer is used to render each item.

The easiest way is to use the default implementation and use `<template name="model">`. Or, by implementing your own ItemRenderer^[5], you can generate the HTML fragment yourself for the data model.

Model

The tree model associated with this cascader.

Open

Drops down or closes the list of items.

Placeholder

When the selected item is empty, the placeholder text would be displayed. (Default: empty)

SelectedItem

Represents the selected item, or null if no item is selected.

Items are selected only if the leaf item is selected. For example, in an A - B - C structure, selected item remains null until the leaf node C is selected.

Supported Events

Name	Event Type
onAfterRender	Event: Event ^[7] Notifies one that the model's data has been rendered.
onSelect	Event: SelectEvent ^[6] Represents an event caused by user's the selection changed at the client.
onOpen	Event: OpenEvent ^[3] Represents an event that indicates an open state that is changed at the client.

- Inherited Supported Events: HtmlBasedComponent

Supported Children

* None

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content
9.0.0	November, 2019	ZK-4392 ^[6] : Provide a cascader component

References

- [1] <https://www.zkoss.org/zkdemo/combobox/cascader>
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Cascader.html#>
- [3] <http://www.zkoss.org/javadoc/latest/javadoc/zkmax/inp/Cascader.html#>
- [4] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/util/Converter.html#>
- [5] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/ItemRenderer.html#>
- [6] <https://tracker.zkoss.org/browse/ZK-4392>

Checkbox

Checkbox

- Demonstration: Checkbox ^[1]
- Java API: Checkbox ^[2]
- JavaScript API: Checkbox ^[3]
- Style Guide: Checkbox

Employment/Purpose

A checkbox.

Example



```
<window title="Checkbox demo" border="normal" width="350px">
    <checkbox id="apple" label="Apple" onCheck="doChecked()" />
    <checkbox id="orange" label="Orange" onCheck="doChecked()" />
    <checkbox id="banana" label="Banana" onCheck="doChecked()" />
    <hbox>
        You have selected :
        <label id="fruit2" />
    </hbox>
    <zscript> void doChecked() { fruit2.value = (apple.isChecked() ?
        apple.label+ ' ' : """) +
        (orange.isChecked() ? orange.label+ ' ' :
        """); }
    </zscript>
</window>
```

Mold

[Since 8.6.0]

There are two additional molds for Checkbox: switch and toggle, you can customize the mold in css by overriding class.

```
<checkbox mold="switch" />  
<checkbox mold="toggle" />
```

switch

Default:



Customized in CSS:



```
.z-checkbox-switch-off > .z-checkbox-mold {  
    background-color: red;  
}  
.z-checkbox-switch-on > .z-checkbox-mold {  
    background-color: green;  
}  
.z-checkbox-switch-off > .z-checkbox-mold:before {  
    background-color: black;  
}  
.z-checkbox-switch-on > .z-checkbox-mold:before {  
    background-color: white;  
}
```

toggle

Default:



Customized in CSS:



```
.z-checkbox-toggle-off > .z-checkbox-mold {  
    background-color: red;  
}  
.z-checkbox-toggle-on > .z-checkbox-mold {  
    background-color: green;  
}
```

tristate

Allowing users to set the indeterminate state, in addition to the checked and unchecked states. In tristate mode, when users click on the checkbox, it will switch between checked, unchecked and indeterminate states. This is different from the default mode which has only checked and unchecked states.



```
<checkbox mold="tristate"></checkbox>
```

We provide a new API `getState()` return CHECKED, UNCHECKED or INDETERMINATE since ZK 9.0.0 .

```
State state = checkbox.getState() // CHECKED, UNCHECKED or INDETERMINATE
```

Indeterminate

Indeterminate is a state that is neither checked nor unchecked.

Note: changing `indeterminate` will not affect the `checked` value, but changing `checked` attribute will set `indeterminate` to false.

```
<checkbox indeterminate="true"/>
```



Display a checkbox like:

Supported Events

Name	Event Type
onFocus	Event: Event ^[7] Denotes when a component gets the focus.
onBlur	Event: Event ^[7] Denotes when a component loses the focus.
onCheck	Event: CheckEvent ^[5] Denotes when a component is checked or unchecked.

- Inherited Supported Events: LabelImageElement

Supported Children

*None

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/zkdemo/input/checkbox>
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Checkbox.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Checkbox.html#>

Chosenbox

Chosenbox

- Demonstration ^[1]
- Java API: Chosenbox ^[2]
- JavaScript API: Chosenbox ^[3]
- Style Guide: Chosenbox
- Available for ZK:
- CE PE EE

Employment/Purpose

A component similar to Combobox but handles the multi-selection and the select order.

Example

Typical Usage

- **creatable** attribute denotes whether or not to display **createMessage** when a user inputs a value that is non-existing in the model, and sends it back to the server along with an **onSearch** event when user clicks the ENTER key or separator.
- **emptyMessage** will be displayed as a placeholder if nothing is selected or focused.
- **noResultText** will be displayed if nothing matches the input value and it cannot be created either; syntax "{0}" will be replaced with the input value at client side.
- **createMessage** will be displayed in popup if nothing matches the input value but can be created as new label; syntax "{0}" will be replaced with the input value at the client-side.

When no item is selected or focused, **emptyMessage** is visible.

Please select some items.

When there is no data to be shown in the model and data 0 already selected, **noResultText** appears.

data 0 data 0

No such item - data 0 - and it is already in the model.

When there is no item in the model but it is creatable, **createMessage** appears.

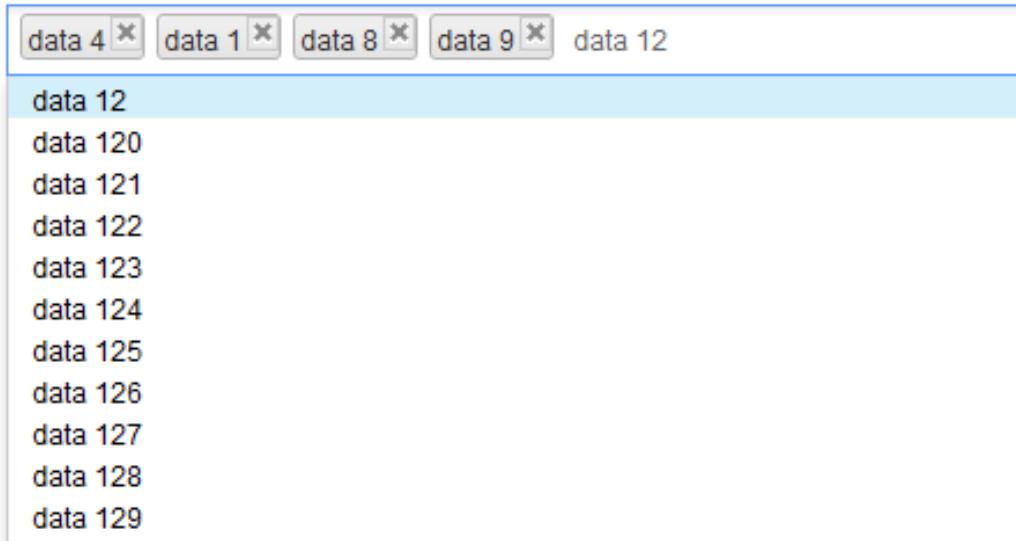
data 0 item 0

No such item -item 0 - but it is not in model either, you can try to create it.

```
<zscript>
    ListModelList model = new
ListModelList(Locale.getAvailableLocales());
</zscript>
<chosenbox width="400px"
    model="${model}" creatable="true"
    emptyMessage=" Please select some items."
    noResultsText=" No such item - {0} - and it is already
in the model."
    createMessage=" No such item -{0} - but it is not in
model either, you can try to create it.">
    <attribute name="onSearch">
        Object obj = event.getValue();
        ((ListModelList)model).add(obj);
        self.addItemToSelection(obj);
    </attribute>
</chosenbox>
```

Rendering All

Here, all the content will be sent to and processed at the client side. The rendering process is pretty fast with a few items but may cause performance issue when the model exceeds 40,000 items and rendering them all at once.



```
<zscript>
    ListModelList model = new
ListModelList(Locale.getAvailableLocales());
</zscript>
<chosenbox width="400px" model="${model}" />
```

Lazy Rendering

With `ListSubModel`, Chosenbox doesn't render any DOM elements in the drop-down list at first. Until a user enters a character, it retrieves 'matched' items from the server-side and renders them in the drop-down list. This might produce some delay at the client side because of server processing time and network latency.

See also: Combobox#Autocomplete_by_ListSubModel [4]

```
<zscript><! [CDATA [
    ListModelList model = new
ListModelList(Locale.getAvailableLocales());
    ListSubModel subModel = ListModels.toListSubModel(model);
]]></zscript>
<chosenbox width="400px" model="${subModel}" />
```

Mouseless Entry Chosenbox

- Press UP and DOWN to move the focus up and down by one option.
- Press LEFT and RIGHT to move focus between selected item(s) and the input field.
- Press ESC to clear input and close drop-down list.
- Press DELETE to delete the focused item and move focus to next item if any or input field.
- Press BACKSPACE to delete the focused item and move focus to previous item if any or input field.
- Press ENTER or specified separator to select the focused option.

Properties

- **creatable:** specify whether to send an event to server when user inputs an non-existing value by clicking ENTER or separator. Default: **false**
- **createMessage:** displayed in popup if nothing matches the input value and creatable is true; syntax "{0}" will be replaced with the input value at client side
- **disabled:** specify whether or not it is disabled. Default: **false**
- **emptyMessage:** displayed as place holder in input if nothing is selected or focused
- **model:** specify the **ListModel** of this **chosenbox**
 - If you set **ListModelList** to the model of **chosenbox**, all the content will be sent to and processed at the client-side, The rendering process is pretty fast with a few items but may cause performance issue when the model exceeds 40,000 items and rendering them all at once
 - If you set **ListSubModel** to the **chosenbox** model, the content of the drop-down list will not be rendered to the client-side, and will remain blank until user enters an input. The server will then provide a 'matched' content for the input. This will cause some delay at the client side because of server processing time and network transfer time
- **name:** specify the name of the input element of this component
- **noResultsText:** displayed in popup window if nothing matches the input value and creatable is false; syntax "{0}" will be replaced with the input value at client-side
- **open:** specify whether or not to open the drop-down list. Default: **false**
- **tabindex:** specify the tab order of the input node of this component. Default: **0**
- **separator:** the separate characters will work as 'Enter' key when clicked on; it will not be considered as an input value. Upon releasing the key, it will an send onSearch or onSelect event depending on the situation. Supports: 0-9, A-Z (case insensitive), and , . ; ' [] / \ -=

Supported Events

Name	Event Type
onSelect	Event: SelectEvent [6] Represents an event caused by user's the selection changed at the client.
onOpen	Event: OpenEvent [3] Represents an event that indicates an open state that is changed at the client.
onSearch	Event: InputEvent [4] Represents an event that indicates users inputting an non-existing value by clicking ENTER or separator.
onSearching	Event: InputEvent [4] Represents an event sent back to the server caused by user's input text.
onItemClick	Event: Event [7] Represents an event sent back to the server caused by clicking a selected tag.

- Inherited Supported Events: HtmlBasedComponent

Supported Molds

- The default mold

Supported Children

None

Use Cases

Version	Description	Example Location
6.0.1+	Creatable Chosenbox	Chosenbox – A beautiful and powerful multiple combobox [5]

Version History

Version	Date	Content
6.0.1	April 3, 2012	Add the new Chosenbox component
8.0.2	May 24, 2016	Add the new Event - onItemClick

References

- [1] https://www.zkoss.org/zkdemo/zk_pe_and_ee/combobox_chosenbox
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Chosenbox.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/inp/Chosenbox.html#>
- [4] http://books.zkoss.org/wiki/ZK_Component_Reference/Input/Combobox#Autocomplete_by_ListSubModel
- [5] <http://blog.zkoss.org/index.php/2012/02/09/>
<zk-6-0-new-feature-highlight-part-4-chosenbox-a-beautiful-and-powerful-multiple-combobox/>

CKEditor

CKEditor

- Demonstration: WYSIWYG Editor ^[1]
- Java API: N/A
- JavaScript API: N/A
- Source code: It is moved to GitHub zkoss/zkckeditor ^[2] after 3.6.0.0.

This add-on is available here ^[3] (**deprecated**)

It is moved to google code after 3.5.2.0: zkckeditor ^[4] (**deprecated**)

Maven

You need to include CKEditor jar in pom.xml before using it because it has a different group id from ZK other components.

```
<dependency>
    <groupId>org.zkoss.zkforge</groupId>
    <artifactId>ckeze</artifactId>
    <version>${ckeze.version}</version>
</dependency>
```

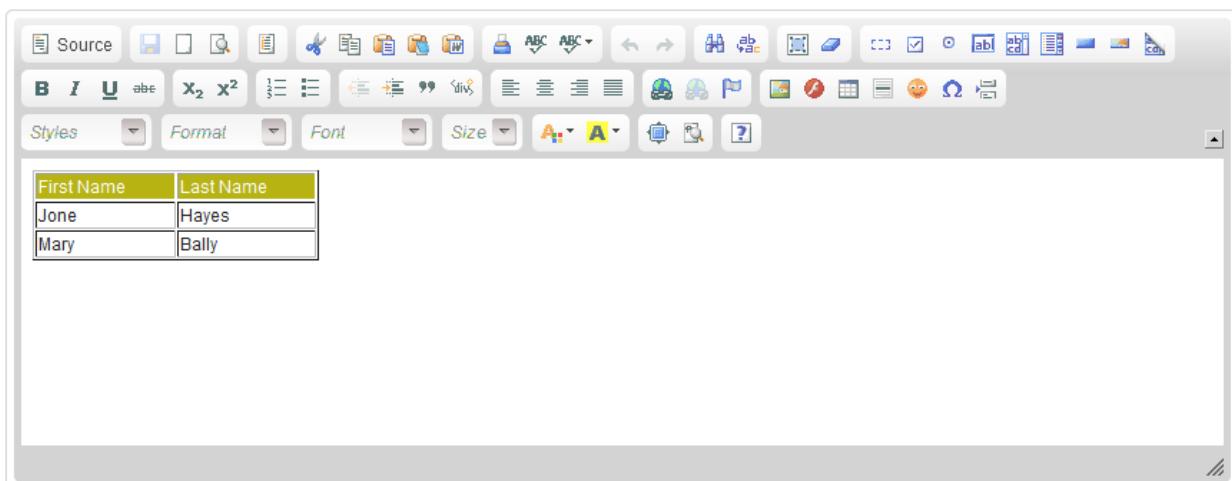
Check the latest version on CE repository ^[5].

Employment/Purpose

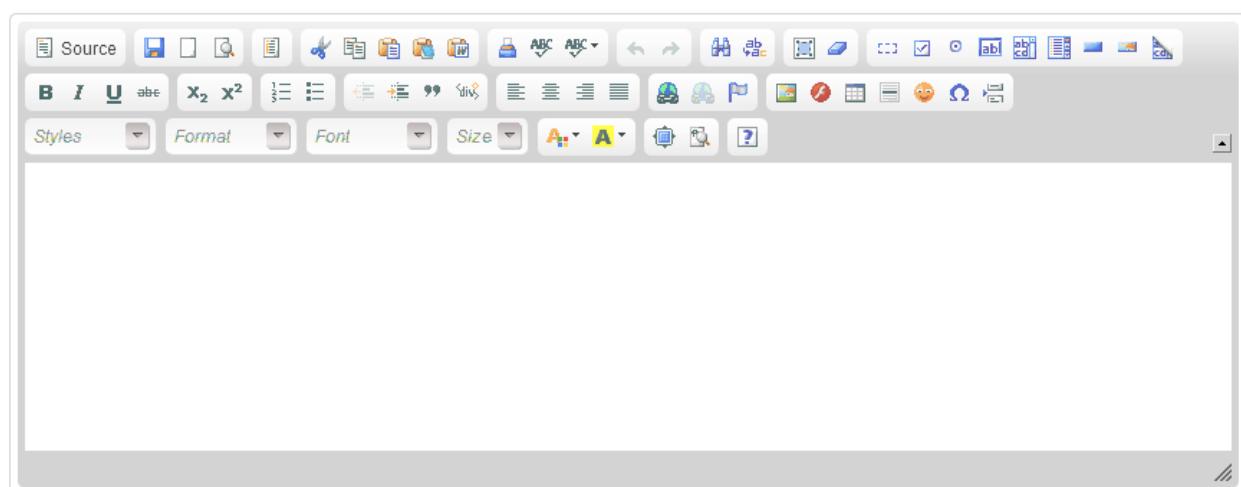
The component is a wrapper of CKEditor ^[6]

CKEditor is a popular HTML on-line text editor developed by Frederico Caldeira Knabben. It is used inside web pages. It's a WYSIWYG editor, which means that the text being edited on it looks as similar as possible to the results users have when publishing it. It brings to the web common editing features found on desktop editing applications like Microsoft Word and OpenOffice.

Example



```
<ckeditor width="850px">
<attribute name="value"><! [CDATA[
<table width="200" cellspacing="1" cellpadding="1" border="1">
    <tbody>
        <tr style="background: #B7B313; color:white;">
            <td>First Name</td>
            <td>Last Name</td>
        </tr>
        <tr>
            <td>Jone</td>
            <td>Hayes</td>
        </tr>
        <tr>
            <td>Mary</td>
            <td>Bally</td>
        </tr>
    </tbody>
</table>
]]></attribute>
</ckeditor>
```



It will turn on the save button when inside a form

```
<zk xmlns:n="http://www.zkoss.org/2005/zk/native">  
    <n:form>  
        <ckeditor width="850" />  
    </n:form>  
</zk>
```

File browser

ZK CKEditor provides a default file browser for browsing the files in a folder that you specify. You can define a target folder in index.zul and when you open the add image/flash dialog and click "Browse Server", CKEditor will open a new window, and list all the files in the file browser.

The screenshot illustrates the CKEditor file browser interface. At the top, there is a code snippet in ZK XML:

```
<zk>
    <ckeditor filebrowserImageBrowseUrl="img"/>
</zk>
```

To the left, a file tree shows the directory structure:

- webapp
 - ckezConfig
 - files
 - flash
 - img
 - Chrysanthemum.jpg
 - item1.jpg
 - item2.jpg
 - item3.jpg
 - item4.jpg
 - Lighthouse.jpg
 - Penguins.jpg
 - META-INF
 - WEB-INF- index.zul

The main window is titled "Image Properties". It contains tabs for "Image Info", "Link", and "Advanced". The "Image Info" tab is active. It has fields for "URL" (with a "Browse Server" button highlighted with a red box) and "Alternative Text". On the left, there are input fields for "Width", "Height", "Border", "HSpace", "VSpace", and "Alignment" (set to "<not set>"). The "Preview" area displays a large amount of placeholder text. At the bottom are "OK" and "Cancel" buttons.

On the right, a separate window shows a file browser with the path "/img". It lists several files:

- Chrysanthemum.jpg (image thumbnail)
- Desert.jpg (image thumbnail)
- item2.jpg (image thumbnail)
- item3.jpg (image thumbnail)

Custom File browser

since 3.6.0.2

If you wish to customize your own file browser, you can change the location by calling CKEditor.setFilebrowserImageUploadUrl(page_url), and refer to CKEditor Developers Guide [7] to create your custom file browser.

File upload

since 3.6.0.2

This feature is only enabled when you specify filebrowserImageUploadUrl attribute. ZK CKEditor provides a default file upload handler for uploading the files to the folder you specify. You can only specify a folder under the web context root because a web application can access its own folder.

The screenshot illustrates the CKEditor file browser and the 'Image Properties' dialog for file uploads.

File Browser: On the left, a tree view shows the directory structure of a web application. It includes 'webapp' (containing 'ckezConfig', 'files', 'flash', and 'img' folders), 'META-INF', 'WEB-INF', and 'index.zul'. The 'img' folder contains several image files: Chrysanthemum.jpg, item1.jpg, item2.jpg, item3.jpg, item4.jpg, Lighthouse.jpg, and Penguins.jpg.

Code Snippet: A code snippet is shown in green font:

```
<font size="8.37">
    <ckeditor filebrowserImageBrowseUrl="img" filebrowserImageUploadUrl="img"/>
</font>
```

Image Properties Dialogs: Two 'Image Properties' dialogs are displayed side-by-side.

- Left Dialog:** Shows the 'Send it to the Server' tab. The 'URL' field contains 'localhost\images\New folder\60x60_logo.png'. A 'Browse...' button is available to change the file. A 'Send it to the Server' button is highlighted with a red border.
- Right Dialog:** Shows the 'Image Info' tab. The 'URL' field contains '/addontest/img/60x60_logo.png'. The 'Preview' section displays a small logo image. The 'Width' and 'Height' fields are both set to 60. Other settings include 'Border', 'HSpace', 'VSpace', and 'Alignment'.

Custom File upload handler

since 3.6.0.2

If you wish to customize your own file upload handler, you can change the location by calling CKEDITOR.setFileUploadHandlePage(page_url), and refer to CKEditor Developers Guide^[7] to create your custom file upload handler.

Custom Configuration

Prepare a javascript file for configuration like:

config.js

```
CKEDITOR.editorConfig = function(config) {  
    //enable spell checker  
    config.disableNativeSpellChecker = false;  
    //Automatically enables "Spell Check As You Type" on editor startup  
    config.scayt_autoStartup = true;  
    //locale  
    config.language = 'de';  
};
```

Please refer to <http://docs.ckeditor.com#!/api/CKEDITOR.config> for complete configuration options.

Specify the configuration file at `customConfigurationsPath` attribute with the absolute path.

```
<ckeditor customConfigurationsPath="/config.js"/>
```

Custom Save Button

You can implement a custom plugin to enable the save button and fire the `onChange` event to the server to save the editor's content. Please refer to on the [github](#)^[8].

resizable attribute and sizing

`default: true`

The ckeditor container can be set resizable with the `resizable` attribute since 4.16.1.1.

```
<ckeditor resizable="true" ...>
```

If `resizable="true"` (default) is set, and the editor has not received a set height (from `height=""` or `flex=""` attributes), the whole component height will be modified on user resizing. In this case the editor outer div dimensions change to accommodate the new size set by the user.

If the component is resizable and has received a set height, it will display a scrollbar in order to maintain its declared size, but allow the user to modify the height of the editing space. In this case, the editor outer div dimensions do not change.

Plugin Installation

1. Download a plugin^[9] according to CKEditor version

ZK Ckeditor version aligns the bundled CKEditor version.

2. put plugin folder into zkckeditor's plugins folder

In ZK, you have to copy the plugin folder into the folder below: (assuming a Maven project)

/resources/web/js/ckeze/ext/CKeditor/plugins/

3. setup in a custom config js

Then provide a **config.js** mentioned at Custom Configuration.

(i) Notice: Since ZK Ckeditor is a Java wrapper of js CKEditor, the installed plugins just work at the client-side and cannot be controlled in Java by default.

Example

1. Download Line Height plugin^[10]

2. its js files under

/resources/web/js/ckeze/ext/CKeditor/plugins/lineheight

3. setup in your custom config js

```
CKEDITOR.editorConfig = function(config) {  
    config.extraPlugins = 'lineheight';  
}
```

Work with ZK6 MVVM

(i) Notice: Since Ckeditor 3.6.0.1, we have added data binding annotation into the lang-addon.xml file, so you no more need to add the settings below.

For work with ZK6 MVVM, it is required to create an addon XML and add the server annotation as follows:

WEB-INF/ckeze-bind-addon.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
<language-addon>  
    <!-- The name of this addon. It must be unique -->  
    <addon-name>ckezebind</addon-name>  
    <!-- Specifies what other addon this depends  
    <depends></depends>  
    -->  
    <!-- Which language this addon will be added to -->  
    <language-name>xul/html</language-name>  
  
    <component>  
        <component-name>ckeditor</component-name>  
        <extends>ckeditor</extends>  
        <annotation>  
            <annotation-name>ZKBIND</annotation-name>  
            <property-name>value</property-name>  
            <attribute>
```

```

        <attribute-name>ACCESS</attribute-name>
        <attribute-value>both</attribute-value>
    </attribute>
    <attribute>
        <attribute-name>SAVE_EVENT</attribute-name>
        <attribute-value>onChange</attribute-value>
    </attribute>
    <attribute>
        <attribute-name>LOAD_REPLACEMENT</attribute-name>
        <attribute-value>value</attribute-value>
    </attribute>
    <attribute>
        <attribute-name>LOAD_TYPE</attribute-name>
        <attribute-value>java.lang.String</attribute-value>
    </attribute>
</annotation>
</component>
</language-addon>
```

then add it into WEB-INF/zk.xml

```

<zk>
    <language-config>
        <addon-uri>/WEB-INF/ckeze-bind-addon.xml</addon-uri>
    </language-config>
</zk>
```

Supported Events

Name	Event Type
onChange	InputEvent [4] Description: Denotes the content of an input component has been modified by the user.
onChanging	InputEvent [4] Description: Denotes that user is changing the content of an input component. Notice that the component's content (at the server) won't be changed until onChange is received. Thus, you have to invoke the <code>getValue</code> method in the InputEvent class to retrieve the temporary value.
onSave	InputEvent [4] Description: Denotes the save button of the CKEditor component has been clicked by the user.

Supported Children

*NONE

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

- [1] http://www.zkoss.org/zkdemo/input/wysiwyg_editor
- [2] <https://github.com/zkoss/zkckeditor>
- [3] <http://sourceforge.net/projects/zk1/files/ZK%20CKEditor/ZK%205/>
- [4] <http://code.google.com/p/zkckeditor/>
- [5] <http://mavensync.zkoss.org/maven2/org/zkoss/zkforge/ckeZ/>
- [6] <http://ckeditor.com/>
- [7] http://docs.cksource.com/CKEditor_3.x/Developers_Guide/File_Browser_%28Uploader%29
- [8] <https://github.com/zkoss/zkbooks/blob/master/componentreference/src/main/webapp/input/ckeditor-save.zul>
- [9] <https://ckeditor.com/cke4/addons/plugins/all>
- [10] <https://ckeditor.com/cke4/addon/lineheight>

Colorbox

Colorbox

- Demonstration: Colorbox ^[1]
- Java API: Colorbox ^[2]
- JavaScript API: Colorbox ^[3]
- Style Guide: Colorbox
- Available for ZK:
-   

Employment/Purpose

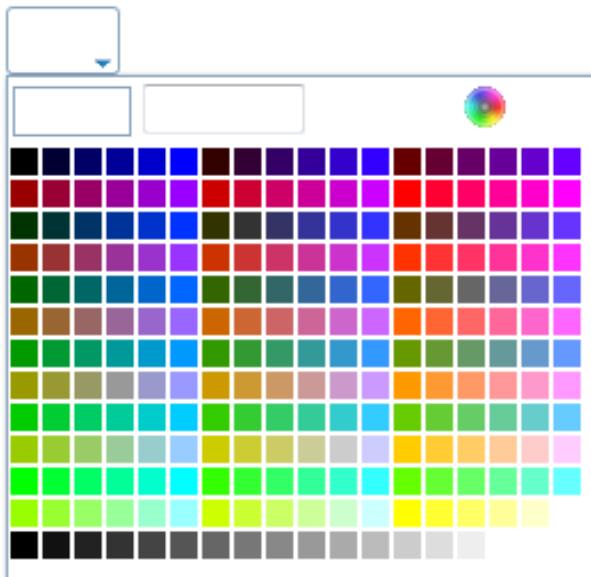
A Colorbox used to retrieve an input that the user can select a color.

Key control

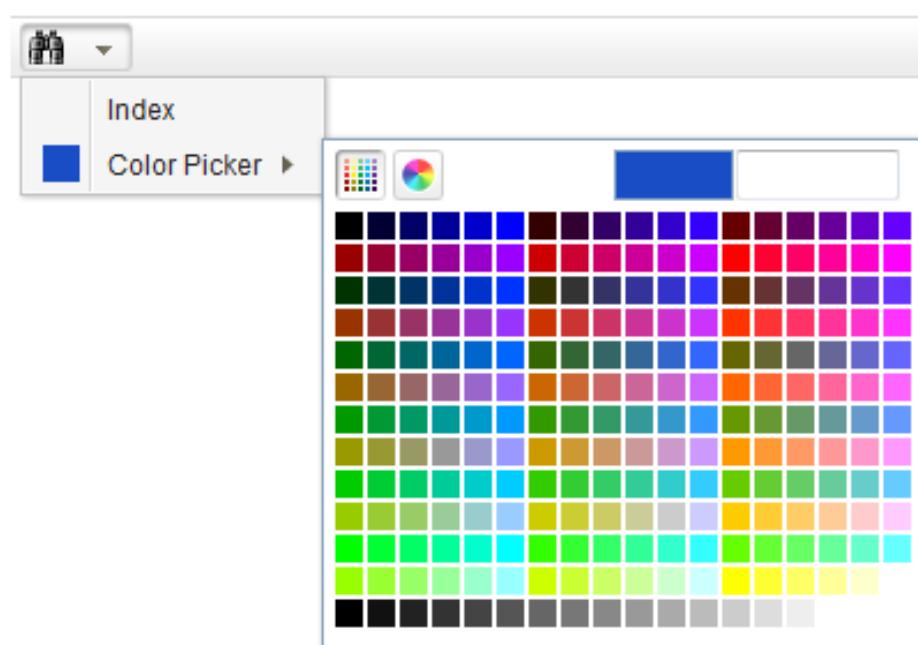
* [since 6.0.0]

- LEFT, RIGHT, UP and DOWN to change the selected color from the calendar.

Example



```
<colorbox color="#FFFFFF" />
```



```

<menubar id="menubar" width="100%">
  <menu image="/img/Centigrade-Widget-Icons/Spyglass-16x16.png">
    <menupopup>
      <menuitem label="Index" onClick="alert(self.label)" />
      <menu label="Color Picker" content="#color=#184dc6"/>
    </menupopup>
  </menu>
</menubar>

```

Supported Events

Name	Event Type
onChange	Event: InputEvent ^[4] Notifies the application with the onChange event if its content is changed

- Inherited Supported Events: XulElement

Supported Children

*None

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content

References

- [1] http://www.zkoss.org/zkdemo/input/color_picker
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkex/zul/Colorbox.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zkex/inp/Colorbox.html#>

Combobox

Combobox

- Demonstration: Combobox ^[1]
- Java API: Combobox ^[2]
- JavaScript API: Combobox ^[3]
- Style Guide: Combobox

Employment/Purpose

Components: combobox and comboitem.

A combobox is a special text box that embeds a drop-down list. With comboboxes, users are allowed to select from a drop-down list, in addition to entering the text manually.

Example

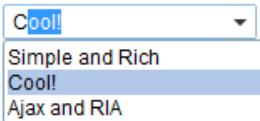
Selection Only

Recommend to use listbox select mold or selectbox.

```
<style>
.z-combobox-readonly>input{
    background-color: initial;
}
</style>

<combobox model="#${model}" readonly="true"/>
```

- a read-only combobox can avoid users from entering non-existed items



```
<combobox>
    <comboitem label="Simple and Rich"/>
    <comboitem label="Cool!" />
    <comboitem label="Ajax and RIA"/>
</combobox>
```

Default Selection

```
Steven ▾  
<zscript><![CDATA[  
    ListModelList model = new ListModelList(Arrays.asList(new  
String[] { "David",  
            "Thomas", "Steven" }));  
    model.addToSelection(model.get(2));  
]]></zscript>  
  
<combobox model="${model}" />
```

- Line 4: When you assign a model object to a component, set the selection through the model object e.g.
`model.addToSelection()`

Mouseless Entry Combobox

- Alt+DOWN to pop up the list.
- Alt+UP or ESC to close the list.
- UP and DOWN to change the selection of the items from the list.
- ENTER to confirm the change of selection.
- ESC to abort the change of selection. It is meaningful if instantSelect is false.

Live Data

Selectable

By specifying the selection, you can invoke the addSelection() to select a default value, For example,

```
<combobox id="combobox" width="100px">  
    <attribute name="onCreate"><![CDATA[  
        List list2 = new ArrayList();  
        list2.add("David");  
        list2.add("Thomas");  
        list2.add("Steven");  
        ListModelList lm2 = new ListModelList(list2);  
        lm2.addToSelection(lm2.get(0));  
        combobox.setModel(lm2);  
    ]]></attribute>  
</combobox>
```

[Since 5.0.4]

Properties

Value and selectedItem

The value stores the selected `comboitem`'s label value and selectedItem stores the selected `comboitem`'s value value:

If you select the 1st item:

```
<combobox value="@bind(vm.inputValue)" selectedItem="@bind(vm.selectedValue)" >
    <comboitem label="label 1" value="value1" />
    <comboitem label="label 2" value="value2" />
</combobox>
```

- vm.inputValue is label 1.
- vm.selectedValue is value1.

Autocomplete

By default, it will autocomplete your input with the first item in the list that has the same starting string in a case-insensitive way.

Autocomplete in a Brute-force Way

The straightforward way to implement the autocomplete feature is to listen to the onChanging event. For example,

```
<combobox>
    <attribute name="onChanging"><! [CDATA[
        self.getChildren().clear(); //remove all children
        for (String value: getMatched(event.getValue()))
            self.appendChild(new Comboitem(value));
    ]]></attribute>
</combobox>
```

Line 4: We assume getMatched() is an application-specific method that returns a collection of matched values.

Autocomplete by ListSubModel

To separate the data from the view (Combobox) better, we can implement ListSubModel^[4] and set it to Combobox.

There are 2 ways:

- Use SimpleListModel^[8].

Unlike ListModelList^[5] and others, SimpleListModel^[8] implements ListSubModel^[4] by default. You can use SimpleListModel^[8] directly but it handles only an array of data.

- Convert ListModel^[7] object with ListModels.toListSubModel(org.zkoss.zul.ListModel)^[6]

The methods convert ListModel to ListSubModel that proxies the original ListModel.

For example,

```
<combobox apply="org.zkoss.reference.component.input.MyAutoCompleteComposer"/>
```

then in MyAutoComplete.java, you could have

```
public class MyAutoCompleteComposer extends SelectorComposer<Component> {
    @Wire("combobox")
```

```

private Combobox combobox;
private List<Locale> items = Arrays.asList(Locale.getAvailableLocales());

@Override
public void doAfterCompose(Component comp) throws Exception {
    super.doAfterCompose(comp);
    combobox.setModel(ListModels.toListSubModel(new
ListModelList(getAllItems())));
}

List getAllItems() {
    return items;
}
}

```

By default, it shows the first 15 items that matches the value entered by the user. If you want to have a different value or a different comparator to find out matched items, you could invoke java.util.Comparator, int) ListModels.toListSubModel(org.zkoss.zul.ListModel, java.util.Comparator, int)^[7] instead.

Note: Passing an instance of ListModelList^[5] directly to a combobox will show up all items in the list model, since it doesn't implement ListSubModel^[4].

Readonly

Once set, the user is not allowed to type characters, but he still can select the items in the Combobox. (Default:false)

```
<combobox readonly="true"/>
```

Autodrop

By default, the drop-down list won't be opened until the user clicks the button, or presses Alt+DOWN. However, you could set the autodrop property to true, meaning as soon as the user types a character the drop-down list will be opened. This is helpful for novice users, but it might be annoying for experienced users.

If you prefer the combobox to drop down the list when the user types a character, you could specify the autodrop attribute as follows.

```
<combobox autodrop="true"/>
```

If you prefer to drop down the list when gaining the focus, you could provide a client-side listener as follows.

```
<combobox w:onFocus="this.open()" xmlns:w="client"/>
```

Description

You are able to add a description to each combo item to make it more descriptive or assign an image to every item.

```

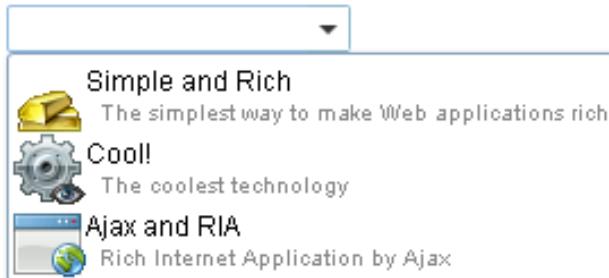
<zk>
    <combobox>
        <comboitem label="Simple and Rich"
image="/img/Centigrade-Widget-Icons/GoldBar-32x32.gif"
description="The simplest way to make Web"

```

```

applications rich" />
    <comboitem label="Cool!" image="/img/Centigrade-Widget-Icons/CogwheelEye-32x32.gif" description="The coolest technology" />
    <comboitem label="Ajax and RIA" image="/img/Centigrade-Widget-Icons/WindowGlobe-32x32.gif" description="Rich Internet Application by Ajax" />
</combobox>
</zk>

```



Akin to other components that support images, you are able to use the `setImageContent` method to assign a dynamically generated image to the `comboitem` component. Please refer to the **Image** section for details.

The onOpen Event

The `onOpen` event is sent to the application when a user opens the drop-down list. To defer the creation of combo items, you can use the `fulfill` attribute as shown below.



```

<zk>
    <combobox fulfill="onOpen">
        <comboitem label="Simple and Rich"/>
        <comboitem label="Cool!"/>
        <comboitem label="Ajax and RIA"/>
    </combobox>
</zk>

```

Alternatively, you can listen to the `onOpen` event and prepare the drop-down list or change it dynamically as demonstrated below.

```

<zk>
    <zscript>
        void prepare()
        {
            if (combo.getItemCount() == 0)
            {
                combo.appendItem("Simple and Rich");
                combo.appendItem("Cool!");
            }
        }
    </zscript>
</zk>

```

```

        combo.appendItem("Ajax and RIA");
    }
}

</zscript>
<combobox id="combo" onOpen="prepare()" />
</zk>
```

The `appendItem` method is equivalent to creating a combo item and then setting the combobox as its parent.

The onChanging Event

Since a combobox is also a text box, you are also able to listen to an `onChanging` event. By listening to this event, you can manipulate the drop-down list as demonstrated by Google Suggests (<http://www.google.com/webhp?complete=1&hl=en>). This feature is sometimes called auto-complete.

As illustrated below, you can populate the drop-down list based on what user is entering.

```

<zk>
<zscript>
void suggest()
{
    combo.getItems().clear();
    if (event.value.startsWith("A")) {
        combo.appendItem("Ace");
        combo.appendItem("Ajax");
        combo.appendItem("Apple");
    } else if (event.value.startsWith("B")) {
        combo.appendItem("Best");
        combo.appendItem("Blog");
    }
}
</zscript>

<combobox id="combo" autodrop="true" onChanging="suggest()" />
</zk>
```

Notice that, when the `onChanging` event is received, the content of the combobox has not changed. Therefore, you cannot use the `value` property of the combobox. Instead, you should use the `value` property of the `InputEvent`^[4].

Constraint

Please see [ZK_Component_Reference/Base_Components/InputElement#Constraint](#).

PopupWidth

[Since 8.0.3]

By specifying this property, the width of the popup will be set and ignore the default behavior.

If percentage is specified to this property, the width of the popup will be calculated with the width of the bandbox.

For example, if it's set to 130%, and the width of the bandbox is 300px, the popup of the bandbox will be 300px *

130% = 390px

If others is specified, it will be set to the width of the popup directly.

InstantSelect

[Since 8.6.1]

By default, any change of selection using the keyboard will trigger `onSelect` and `onChange` events instantly. Once set this property `false`, users need to confirm the change by pressing Enter key or make combobox lose its focus so `onSelect` and `onChange` events will be triggered. And pressing Esc key can abort the change and revert to previous selection.

IconSclass

[Since 8.6.2]

Specify the sclass name of the Combobox button icon.

Inherited Functions

Please refer to Textbox for inherited functions.

Supported Events

Name	Event Type
onSelect	Event: SelectEvent ^[6] Represents an event caused by user's the list selection is changed at the client.
onOpen	Event: OpenEvent ^[3] Denotes that the user has opened or closed a component. Note: unlike <code>onClose</code> , this event is only a notification. The client sends this event after opening or closing the component. It is useful to implement <i>load-on-demand</i> by listening to the <code>onOpen</code> event, and creating components when the first time the component is opened.
onAfterRender	Event: Event ^[7] Notifies one that the model's data has been rendered.

- Inherited Supported Events: Textbox

Supported Molds

Available molds of a component are defined in lang.xml embedded in zul.jar.

Name	Snapshot
default	
rounded	 [Since 5.0.0]

Supported Children

* `Comboitem`

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content
8.6.1	January 2019	ZK-4185 [8]: Combobox: provide option to reduce onSelect/onChange events when using keyboard
5.0.4	August 2010	ListModels [9] was introduced to simplify the implementation of autocomplete.
5.0.4	July 2010	Combobox supported Selectable [10] if it is also implemented with the specified ListModel [7].
5.0.4	July 2010	Supported onAfterRender event

References

- [1] <http://www.zkoss.org/zkdemo/combobox>
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Combobox.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/inp/Combobox.html#>
- [4] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/ListSubModel.html#>
- [5] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/ListModelList.html#>
- [6] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/ListModels.html#toListSubModel\(org.zkoss.zul.ListModel\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/ListModels.html#toListSubModel(org.zkoss.zul.ListModel))
- [7] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/ListModels.html#toListSubModel\(org.zkoss.zul.ListModel,](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/ListModels.html#toListSubModel(org.zkoss.zul.ListModel,)
- [8] <https://tracker.zkoss.org/browse/ZK-4185>
- [9] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/ListModels.html#>
- [10] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/ext>Selectable.html#>

Comboitem

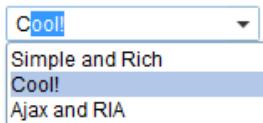
Comboitem

- Demonstration: Combobox ^[1]
- Java API: Comboitem ^[1]
- JavaScript API: Comboitem ^[2]
- Style Guide: Comboitem

Employment/Purpose

An item of a combo box.

Example



```
<combobox>
    <comboitem label="Simple and Rich"/>
    <comboitem label="Cool!" />
    <comboitem label="Ajax and RIA"/>
</combobox>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: LabelImageElement

Supported Children

*NONE

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Comboitem.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/inp/Comboitem.html#>

Datebox

Datebox

- Demonstration: Date and Time ^[1]
- Java API: Datebox ^[2]
- JavaScript API: Datebox ^[3]
- Style Guide: Datebox

Employment/Purpose

An edit box for holding a date. After click on the calender, a calender will pop-up for inputting date.

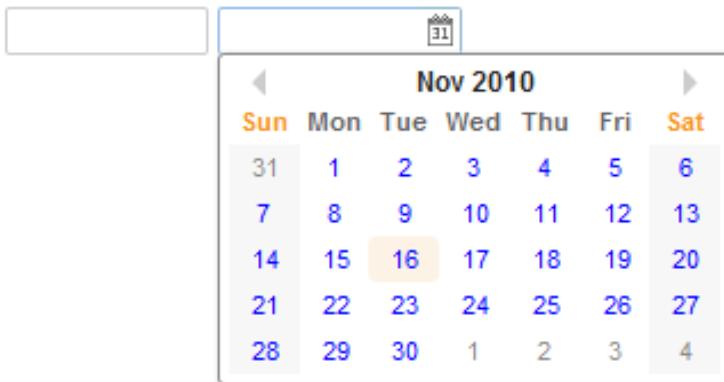
Mouseless Entry datebox

- Alt+DOWN to pop up the calendar.
- LEFT, RIGHT, UP and DOWN to change the selected day from the calendar.
- ENTER to activate the selection by copying the selected day to the datebox control.
- Alt+UP or ESC to give up the selection and close the calendar.

Customization

You can customize the rendering of the calendar at the client by JavaScript code that overrides Renderer ^[4].

Example



```
<datebox lenient="true" buttonVisible="false" />  
<datebox lenient="false" buttonVisible="true" />
```

Properties and Features

Constraint

You can specify the date range to accept by the `constraint` property with one or multiple following values:

- no empty
- no future
- no past
- no today.

It also supports an interval of dates. For example,

```
<datebox constraint="between 20071225 and 20071203"/>  
<datebox constraint="before 20071225"/>  
<datebox constraint="after 20071225"/>
```

Notices

1. The format of the date in the constraint must be `yyyMMdd`. It is independent of the locale.
2. The date specified in the above constraints (`before/after/between`) is *included*. For example, "`before 20071225`" includes December 25, 2007 and any day before it, and "`after 20110305`" includes March 5, 2011 and any day after it.
3. The constraint is actually represented with an instance of the `org.zkoss.zul.SimpleDateConstraint` class. You can retrieve the parsed beginning and ending date with the `getBeginDate` and `getEndDate` methods.

```
((SimpleDateConstraint)datebox.getConstraint()).getBeginDate();
```

Multiple Constraints

To specify two or more constraints, use a comma to separate them as follows:

```
<datebox constraint="no past,no empty"/>
```

Custom Error Message

If you prefer to display a different message from the default one, you can append the error message to the constraint with a colon.

```
<datebox constraint="no empty, no future: now or never"/>
```

Notes:

- The error message, if specified, must be the last element and start with a colon.
- To support multiple languages, you could use the `「1」` function as depicted in the **Internationalization** chapter.

```
<datebox constraint="no empty, no future: ${c:l('err.date.nowornever')}"/>
```

Displayed Time Zones

The image below shows the new Datebox [2] functionality which allows the user to change the time zone to other predefined time zones. Viewing the zul markup provided below the image we can see that the displayedTimeZones is set to "GMT+12,GMT+8". These options are specified by the developer and restrict the user to the available time zones.



Format

You are able to format the field by providing specifying the attribute with a formatting string. The default value is null. When the formatting of the `datebox` is null, it means the date will be outputted using the format `yyyy/MM/dd`.

```
<datebox format="MM/dd/yyyy"/>
```

Like any other properties, you are able to change the format dynamically, as depicted below.

```
<datebox id="db"/>
<button label="set MM-dd-yyyy" onClick='db.setFormat("MM-dd-yyyy")'></button>
```

Length Option

In addition to specifying the format explicitly, you could specify the length option. It supports four different length options mentioned at `java.text.SimpleDateFormat` [4]:

- short
- medium
- long
- full

For example, you could specify the styling rather than the real format as follows.

```
<datebox format="short"/>
<datebox format="long"/>
```

Then the real format of the datebox will be decided at run-time depending on the configuration. For more information, please refer to ZK Developer's Reference: Date and Time Formatting.

In addition, you could specify the format for both date and time by using the syntax:

```
format="option_for_date+option_for_time"
```

For example,

```
<datebox format="medium+full"/>
```

Warning

Because Java default locale provider changes since JDK 9 [5], the built-in formats (e.g. `long`) for some locales might change since JDK 9. We recommend you to use a fixed format pattern.

Validation

If a user's input doesn't match the specified format, a Datebox will show an error message. It's a client-side validation.

Locale

Default: depends on the current locale (i.e., `Locales.getCurrent()` [6]) at run-time

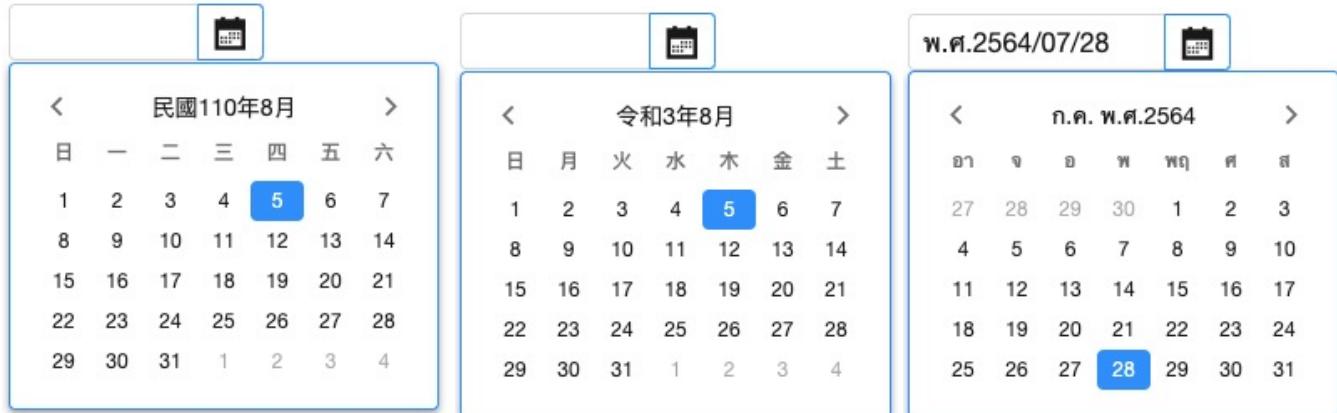
You can enforce the locale for an individual component such as:

```
<datebox locale="de-DE" format="full"/>
<datebox locale="fr" format="full"/>
```

Multiple-Eras Calendar

Datebox can display some multiple-eras calendar systems including:

- ROC(Taiwan): `locale="zh-TW-u-ca-roc"`
- Japan: `locale="ja-JP-u-ca-japanese"`
- Buddhist: `locale="th-TH-u-ca-buddhist"`



See complete locale list ^[7].

Reference: Era ^[8]

Position

By default, the popup position is set to `after_start`, for other possible popup positions please refer to `Popup#Position`.

The First Day of the Week

The first day of the week is decided by the locale (actually the return value of the `getFirstDayOfWeek` method in the `java.util.Calendar`).

Since 5.0.3, you can control the first day of the week by the use of the `session` attribute and the `library` property. Please refer to `The First Day of the Week` for details.

2DigitYearStart

You can control the `2DigitYearStart` by the use of the `library` property, `org.zkoss.web.preferred.2DigitYearStart`.

Show Week Number

- Available for ZK:
-

Datebox supports to show a week number of the year in a calendar.



```
<datebox weekOfYear="true" />
```

Show Link of Today

Datebox supports a link to jump back to the date of today quickly

[Since 8.0.0]



```
<datebox id="db" showTodayLink="true" ></datebox>
```

The format is the same with that specified on format attribute

Show Timebox

By default, there is no Timebox in the popup Calendar. If the specified `format` attribute contains a time format (like below), it will show a Timebox at the bottom of the popup Calendar.

```
<datebox format="yyyy-MM-dd HH:mm"/>
```

Monthly / yearly option

You can specify the `selectLevel` attribute to restrict the date granularity users can select. Available options are `year`, `month` and `day`. Default setting is `day`.

```
<datebox format="yyyy" selectLevel="year"/>
```

Close Popup OnTimezoneChange

Datebox supports switching whether to auto close the datebox popup after changing the timezone.

example:

```
<datebox closePopupOnTimezoneChange="false" displayedTimeZones="GMT+12,GMT+8" timeZone="GMT+8" timeZonesReadOnly="false"/>
```

Inherited Functions

Please refer to FormatInputElement for inherited functions.

Supported Events

Name	Event Type
onTimeZoneChange	Event: Event [7] Denotes the time zone of the component is changed by user.

- Inherited Supported Events: FormatInputElement

Supported Molds

Available molds of a component are defined in lang.xml embedded in zul.jar.

Name	Snapshot
default	
rounded	 [Since 5.0.0]

Supported Children

Name	Event Type
None	None

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content
5.0.3	July, 2010	An application can control the first day of the week by use of the session attribute and the library property. Please refer to The First Day of the Week for details.
5.0.4	August, 2010	Calendar supports moving to next/prev mon by mouse scrolling.
5.0.7	April, 2011	Datebox.setFormat(java.lang.String) ^[9] supported the styling.
5.0.7	April, 2011	Datebox.setLocale(java.util.Locale) ^[10] was introduced.
6.5.0	June, 2012	ZK-1175 ^[11] : Calendar support show week number
9.5.1	October 2020	ZK-3289 ^[12] : Monthly / yearly options for datebox.

References

- [1] http://www.zkoss.org/zkdemo/input/date_and_time_picker
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Datebox.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/db/Datebox.html#>
- [4] <http://docs.oracle.com/javase/7/docs/api/java/text/DateFormat.html>
- [5] <http://openjdk.java.net/jeps/252>
- [6] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/util/Locales.html#getCurrent\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/util/Locales.html#getCurrent())
- [7] <https://www.oracle.com/java/technologies/javase/jdk8-jre8-supported-locales.html>
- [8] <https://docs.oracle.com/javase/8/docs/api/java/time/chrono/Era.html>
- [9] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Datebox.html#setFormat\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Datebox.html#setFormat(java.lang.String))
- [10] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Datebox.html#setLocale\(java.util.Locale\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Datebox.html#setLocale(java.util.Locale))
- [11] <https://tracker.zkoss.org/browse/ZK-1175>
- [12] <https://tracker.zkoss.org/browse/ZK-3289>

Decimalbox

Decimalbox

- Demonstration: Decimalbox ^[1]
- Java API: Decimalbox ^[1]
- JavaScript API: Decimalbox ^[2]
- Style Guide: Decimalbox

Employment/Purpose

An edit box for holding big decimal value .

Example

```
155
```

```
<decimalbox value="155"/>
```

Properties

Format

You are able to format the field by providing specifying the attribute with a formatting string. The default value is null.

```
<decimalbox format="#,##0.##"/>
```

Since 8.5.2

You can provide a locale to format the number by specify the String starts with "locale:"

```
<decimalbox format="locale:zh-TW"/>
```

Constraint

You could specify what value to accept for input controls by the use of the constraint property. It could be a combination of no positive, no negative, no zero, no empty.

To specify two or more constraints, use comma to separate them as follows.

```
<decimalbox constraint="no negative,no empty"/>
```

If you prefer to display different message to the default one, you can append the error message to the constraint with a colon.

```
<decimalbox constraint="no negative: it shall not be negative"/>
```

Notes:

- The error message, if specified, must be the last element and start with colon.
- To support multiple languages, you could use the 「」 function as depicted in the **Internationalization** chapter.

```
<decimalbox constraint="no negative: ${c:l('err.num.negative')}"/>
```

Inherited Functions

Please refer to NumberInputElement for inherited functions.

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: NumberInputElement

Supported Children

*NONE

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Decimalbox.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/inp/Decimalbox.html#>

Doublebox

Doublebox

- Demonstration: Doublebox ^[1]
- Java API: Doublebox ^[1]
- JavaScript API: Doublebox ^[2]
- Style Guide: Doublebox

Employment/Purpose

An edit box for holding a float point value (double).

Example

```
2.3
```

```
<doublebox value="2.3"/>
```

Properties

Format

You are able to format the field by providing specifying the attribute with a formatting string. The default value is null.

```
<doublebox format="#,##0.#"/>
```

Since 8.5.2

You can provide a locale to format the number by specifying the String starts with "locale:"

```
<doublebox format="locale:zh-TW"/>
```

Constraint

You could specify what value to accept for input controls by the use of the constraint property. It could be a combination of no positive, no negative, no zero, no empty.

To specify two or more constraints, use comma to separate them as follows.

```
<doublebox constraint="no negative,no empty"/>
```

If you prefer to display different message to the default one, you can append the error message to the constraint with a colon.

```
<doublebox constraint="no negative: it shall not be negative"/>
```

Notes:

- The error message, if specified, must be the last element and start with colon.
- To support multiple languages, you could use the 「1」 function as depicted in the **Internationalization** chapter.

```
<doublebox constraint="no negative: ${c:l('err.num.negative')}"/>
```

Inherited Functions

Please refer to NumberInputElement for inherited functions.

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: NumberInputElement

Supported Children

*NONE

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Doublebox.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/inp/Doublebox.html#>

Doublespinner

Doublespinner

- Demonstration: Spinner ^[1]
- Java API: Doublespinner ^[1]
- JavaScript API: Doublespinner ^[2]
- Style Guide: Doublespinner

Employment/Purpose

An edit box for holding a constrained double.

Example



```
<doublespinner step="0.5" />
```

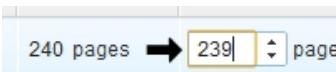
In-place Editing

Fixed Width

```
<doublespinner width="100px" inplace="true" value="30" />
```

Dynamic Width

Because inplace editing function in ZK is pure client side action, so we can use client api to modify the width (server side do not need to know)



```
<zk xmlns:c="client">  
  <doublespinner inplace="true" value="240" width="30px" c:onFocus='this.setWidth("60px")' c:onBlur='this.setWidth("30px")' />  
</zk>
```

Properties

Format

You are able to format the field by providing specifying the attribute with a formatting string. The default value is null.

```
<doublespinner format="#,##0.##"/>
```

Since 8.5.2

You can provide a locale to format the number by specify the String starts with "locale:"

```
<doublespinner format="locale:zh-TW"/>
```

Constraint

You could specify what value to accept for input controls by use of the `constraint` property. It could be a combination of no empty and the minimum and maximum to doublespinner.

To specify two or more constraints, use comma to separate them as follows.

```
<doublespinner step="0.5" constraint="no empty,min -2.5 max 6.5"/>
```

If you prefer to display different message to the default one, you can append the error message to the constraint with a colon.

```
<doublespinner step="0.5" constraint="no empty,min -2.5 max 6.5: between -2.5 to 6.5"/>
```

Notes:

- The error message, if specified, must be the last element and start with colon.
- To support multiple languages, you could use the `「1」` function as depicted in the **Internationalization** chapter.

```
<doublespinner step="0.5" constraint="no empty,min -2.5 max 6.5: ${c:l('err.msg.doublespinner')}"/>
```

Inherited Functions

Please refer to NumberInputElement for inherited functions.

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: NumberInputElement

Supported Molds

Available molds of a component are defined in lang.xml embedded in zul.jar.

Name	Snapshot
default	
rounded	

Supported Children

*None

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content
5.0.6	Dec. 2010	add new component

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Doublespinner.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/inp/Doublespinner.html#>

Intbox

Intbox

- Demonstration: Intbox ^[1]
- Java API: Intbox ^[1]
- JavaScript API: Intbox ^[2]
- Style Guide: Intbox

Employment/Purpose

An `intbox` is used to let users input integer data.

Example



While input invalid data:



```
<window title="Intbox Demo" border="normal" width="200px">
    int box:<intbox/>
</window>
```

Properties

Format

You are able to format the field by providing specifying the attribute with a formatting string. The default value is null.

```
<intbox format="#,##0"/>
```

Since 8.5.2

You can provide a locale to format the number by specify the String starts with "locale:"

```
<intbox format="locale:zh-TW"/>
```

Constraint

You could specify what value to accept for input controls by use of the `constraint` property. It could be a combination of no positive, no negative, no zero, no empty.

To specify two or more constraints, use comma to separate them as follows.

```
<intbox constraint="no negative,no empty"/>
```

If you prefer to display different message to the default one, you can append the error message to the constraint with a colon.

```
<intbox constraint="no negative: it shall not be negative"/>
```

Notes:

- The error message, if specified, must be the last element and start with colon.
- To support multiple languages, you could use the `「1」` function as depicted in the **Internationalization** chapter.

```
<intbox constraint="no negative: ${c:l('err.num.negative')}"/>
```

Inherited Functions

Please refer to `NumberInputElement` for inherited functions.

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: `NumberInputElement`

Supported Children

*NONE

Use Cases

Version	Description	Example Location
3.6	Leading zero in Intbox	[3]
3.6	Constraint Intbox to accept only digits	[4]

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Intbox.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/inp/Intbox.html#>
- [3] <http://www.zkoss.org/forum/listComment/10271>
- [4] <http://www.zkoss.org/forum/listComment/4603>

Longbox

Longbox

- Demonstration: Longbox ^[1]
- Java API: Longbox ^[1]
- JavaScript API: Longbox ^[2]
- Style Guide: Longbox

Employment/Purpose

A longbox is used to let users input long data.

Example

Longbox Demo

long box: 9223372036854775807

```
<window title="Longbox Demo" border="normal" width="400px">
    long box:<longbox width="250px"/>
</window>
```

Properties

Format

You are able to format the field by providing specifying the attribute with a formatting string. The default value is null.

```
<longbox format="#,##0"/>
```

Since 8.5.2

You can provide a locale to format the number by specify the String starts with "locale:"

```
<longbox format="locale:zh-TW"/>
```

Constraint

You could specify what value to accept for input controls by use of the `constraint` property. It could be a combination of no positive, no negative, no zero, no empty.

To specify two or more constraints, use comma to separate them as follows.

```
<longbox constraint="no negative,no empty"/>
```

If you prefer to display different message to the default one, you can append the error message to the constraint with a colon.

```
<intbox constraint="no negative: it shall not be negative"/>
```

Notes:

- The error message, if specified, must be the last element and start with colon.
- To support multiple languages, you could use the `「1」` function as depicted in the **Internationalization** chapter.

```
<longbox constraint="no negative: ${c:l('err.num.negative')}"/>
```

Inherited Functions

Please refer to `NumberInputElement` for inherited functions.

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: `NumberInputElement`

Supported Children

*NONE

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Longbox.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/inp/Longbox.html#>

Multislider

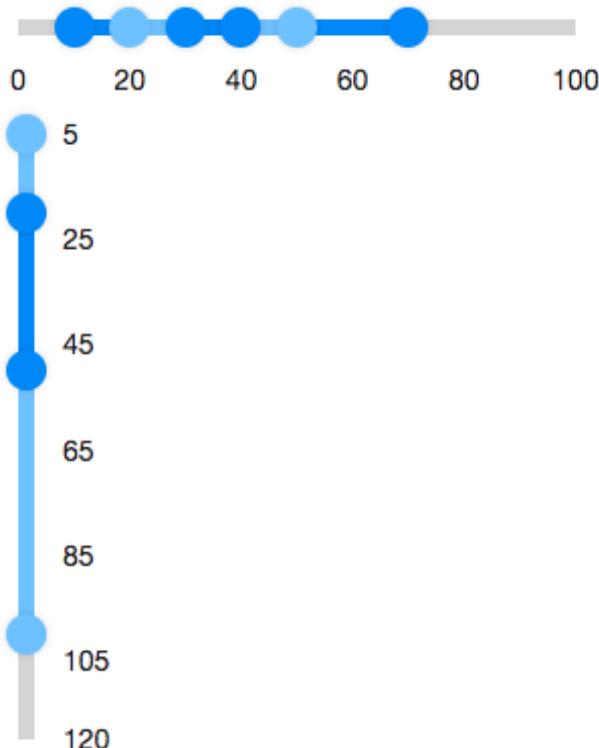
Multislider

- Demonstration:
- Java API: Multislider ^[1]
- JavaScript API: Multislider ^[2]
- Available for ZK:
- CE PE EE

Employment/Purpose

A multislider component represents a slider with multiple ranges. It includes sliderbuttons, which can be used to let user select a start value and an end value. A multislider accepts a range of values starting from 0 to a maximum value you defined. The default maximum value is 100. You can change the maximum value by setting the max property. Notice that the value of max property is always larger than the value of min property.

Example



```
<zk>
  <multislider>
    <sliderbuttons startValue="10" endValue="70"/>
    <sliderbuttons startValue="20" endValue="50"/>
    <sliderbuttons startValue="30" endValue="40"/>
  </multislider>
  <multislider min="5" max="120" orient="vertical">
    <sliderbuttons startValue="5" endValue="100"/>
    <sliderbuttons startValue="20" endValue="50"/>
  </multislider>
</zk>
```

Properties

Disabled

If the multislider is disabled, users can not drag the slider buttons.

Orient

Sets the orientation to either "horizontal" or "vertical" to display the multislider.

Marks

Sets the marks information for displaying value marks.

It supports Map<Integer, String>. The key is represented as the value of multislider, and the value is represented as the displayed mark label. It means that each value mark could be displayed in different text.

MarkScale

Sets the marks information for displaying value marks.(Default: "20")

By default the value marks are displayed every 20 starting from the minimum value. (if min is 0, then it displays "0 20 40 60 ...")

Max

Defines the max value of the multislider. It can be changed by the max property. (Default: 100)

Min

Defines the minimal value of the multislider. It can be changed by the min property. (Default: 0)

Step

By default, the multislider will scroll to the position continuously when a user drags it. If you prefer to scroll a discrete fixed amount at each step, you can set the Step property.

TooltipVisible

The tooltip displays the value of slider buttons in the multislider. If tooltipvisible is true, the tooltips of the slider buttons will always be displayed. (Default: false)

Supported Events

- Inherited Supported Events: XulElement

Supported Children

* Sliderbuttons

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
9.0.0	November, 2019	Multislider [1] was introduced.

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Multislider.html>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/slider/Multislider.html>

Radio

Radio

- Demonstration: Radio ^[1]
- Java API: Radio ^[2]
- JavaScript API: Radio ^[3]
- Style Guide: Radio

Employment/Purpose

A `radio` button is a component that can be turned on and off. Radio buttons are grouped together in a group, called `radiogroup`. Only one radio button with the same group may be selected at a time.

Example

You have selected :
Apple

```
<vlayout>
    <radiogroup onCheck="fruit.value = self.selectedItem.label">
        <radio label="Apple"/>
        <radio label="Orange"/>
        <radio label="Banana"/>
    </radiogroup>
    You have selected :
    <label id="fruit" style="color:red"/>
</vlayout>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: Checkbox

Supported Children

*NONE

Use Cases

Version	Description	Example Location
3.6	Radio buttons with Listitems	[4] [4]
3.6	Radiogroup radio's in separate table/grid rows	[5] [5]

See also: Radiogroup

Version History

Version	Date	Content

References

- [1] http://www.zkoss.org/zkdemo/input/radio_button
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Radio.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Radio.html#>
- [4] <http://www.zkoss.org/forum/listComment/3867>
- [5] <http://www.zkoss.org/forum/listComment/9002>

Radiogroup

Radiogroup

- Demonstration: Radiogroup ^[1]
- Java API: Radiogroup ^[1]
- JavaScript API: Radiogroup ^[2]
- Style Guide: N/A

Employment/Purpose

Used to group multiple radio buttons. In one radiogroup. Only one radio button may be selected at a time.

Example



You have selected :

Apple

```
<window title="Radiobox & Radio Demo" width="200px" border="normal">
  <vbox>
    <radiogroup onCheck="fruit.value = self.selectedItem.label">
      <radio label="Apple" />
      <radio label="Orange" />
```

```

<radio label="Banana" />
</radiogroup>
You have selected :
<label id="fruit" style="color:red" />
</vbox>
</window>

```

Note: To support the versatile layout, a radio group accepts any kind of children , including Radio. On the other hand, the parent of a radio, if any, must be a radio group.

Radiogroup as an Ancestor of Radio

ZK groups radio components into the same radio group if they share the same ancestor, not just direct parent. It allows a more sophisticated layout. For example,

```

<radiogroup>
  <vlayout>
    <hlayout>
      <radio label="radio 1"/>
      <radio label="radio 2"/>
      <radio label="radio 3"/>
    </hlayout>
    <hlayout>
      <radio label="radio 4"/>
      <radio label="radio 5"/>
      <radio label="radio 6"/>
    </hlayout>
  </vlayout>
</radiogroup>

```

A Row of a Grid as a Radio Group

Sometimes it is not possible to make the radiogroup component as an ancestor of all radio components. For example, each row of a grid might be an independent group. To solve this, you have to assign the radiogroup component to the radio component explicitly by the use of Radio.setRadiogroup(java.lang.String) [3] or Radio.setRadiogroup(org.zkoss.zul.Radiogroup) [4].

Question	Option 1	Option 2	Option 3	Comment
Most popular	<input checked="" type="radio"/> Java	<input checked="" type="radio"/> Groovy	<input checked="" type="radio"/> C#	<input type="text"/>
Most fun	<input checked="" type="radio"/> Open Source	<input checked="" type="radio"/> Social Networking	<input checked="" type="radio"/> Searching	<input type="text"/>

```

<zk>
  <radiogroup id="popular"/>
  <radiogroup id="fun"/>
  <grid>
    <columns>
      <column label="Question"/>
      <column label="Option 1"/>

```

```
<column label="Option 2"/>
<column label="Option 3"/>
<column label="Comment"/>
</columns>
<rows>
<row>
    Most popular
    <radio label="Java" radiogroup="popular"/>
    <radio label="Groovy" radiogroup="popular"/>
    <radio label="C#" radiogroup="popular"/>
    <textbox/>
</row>
<row>
    Most fun
    <radio label="Open Source" radiogroup="fun"/>
    <radio label="Social Networking" radiogroup="fun"/>
    <radio label="Searching" radiogroup="fun"/>
    <textbox/>
</row>
</rows>
</grid>
</zk>
```

Live Data

Like a listbox, radiogroup supports ListModel^[5], so that developers are able to separate the data from the view. In other words, developers only need to provide the data by implementing the ListModel^[7] interface, rather than manipulating the radiogroup directly. The benefits are twofold.

- It is easier to show the same set of data in different views.
- The grid sends the data to the client only if it is visible. It saves a lot of network traffic if the amount of data is large.

There are three steps to make use of live data.

1. Store your data object in a ListModel^[7]

ZK provides several implementation implementations of ListModel, just choose one upon your needs.

2. Set the ListModel at the model attribute.

3. (Optional) Implement the <T>.html# RadioRenderer<T>^[6] interface to render each radio and specify it in the radioRenderer attribute

* This is optional. If it is not specified, ZK will render it with the default renderer.

* You can implement different renderers for representing the same data in different views.

In the following example, we prepared a ListModel called strset, assign it to a radigroup using the model attribute. Then, the radigroup will do the rest.

Live Radiogroup

option 0 option 1 option 2 option 3 option 4

```
<zscript><! [CDATA [
    String[] data = new String[5];
    for(int j=0; j < data.length; ++j) {
        data[j] = "option "+j;
    }
    ListModel strset = new SimpleListModel(data);
]]></zscript>
<radiogroup model="${strset}" />
```

Supported Events

Name	Event Type
onCheck	Event: CheckEvent [5] Denotes when a radio under the radiogroup is checked.

- Inherited Supported Events: XulElement

Supported Children

*ALL

Use Cases

Version	Description	Example Location
5.0	Radiogroup and selection	[7]
5.0	Radiogroup, data binding and TypeConverter	[8]
6.0.0	Introduce model to Radiogroup	

Version History

Version	Date	Content
5.0.4	August 2010	Allow a radio component associated with a non-ancestor radiogroup.

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Radiogroup.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Radiogroup.html#>
- [3] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Radio.html#setRadiogroup\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Radio.html#setRadiogroup(java.lang.String))
- [4] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Radio.html#setRadiogroup\(org.zkoss.zul.Radiogroup\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Radio.html#setRadiogroup(org.zkoss.zul.Radiogroup))
- [5] https://www.zkoss.org/wiki/ZK_Developer%27s_Reference/MVC/Model>List_Model
- [6] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/RadioRenderer>

- [7] <http://www.zkoss.org/forum/listComment/12148>
- [8] <http://www.zkoss.org/forum/listComment/7011>

Rangeslider

Rangeslider

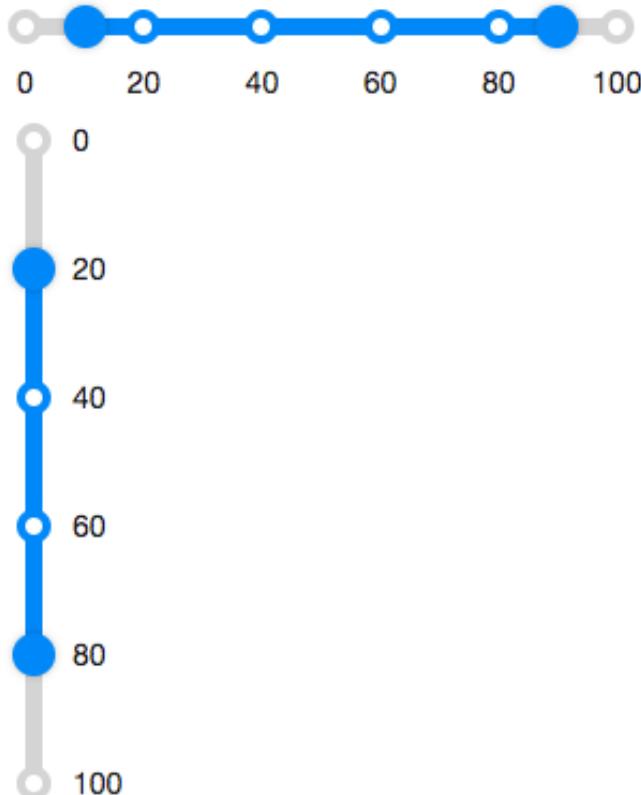
- Demonstration:
- Java API: Rangeslider ^[1]
- JavaScript API: Rangeslider ^[2]
- Available for ZK:
- CE PE EE

since 9.0.0

Employment/Purpose

A rangeslider component represents a slider with a start and an end value. A rangeslider accepts a range of value starting from 0 to a certain maximum value. The default maximum value of rangeslider is 100. You can change the maximum allowed value by setting the max property. Notice that the value of max property is always larger than the value of min property.

Example



```
<zk>
<rangeslider startValue="10" endValue="90"/>
```

```
<separator />
<rangerlider orient="vertical" startValue="20" endValue="80" markScale="20" />
</zk>
```

Properties

Disabled

If the rangeslider is disabled, then users can not drag the slider buttons.

Orient

Sets the orient either "horizontal" or "vertical" to display rangslider.

Marks

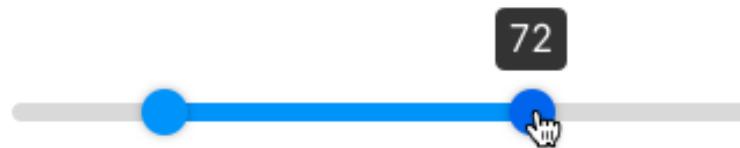
Sets the marks information map for displaying value marks. (Default: null) In this map, the key represents the number value of slider, and the value represents the displayed scale text.

MarkScale

Sets the marks information for displaying value marks (Default: "20").

By default, the value marks will be displayed every 20 starting from the minimum value. (if min is 0, then it displays "0 20 40 ...")

If the MarkScale is 0 and there is no Map information in Marks (see above), the marks will be empty.



Max

Rangeslider supports maximal position, which can be changed by the max property. (Default: 100)

Min

Rangeslider supports minimal position, which can be changed by the min property. (Default: 0)

StartValue, EndValue

Represent the range value of Rangeslider. (Default: 0)

Step

By default, the rangeslider will scroll to the position continuously when a user drags it. If you prefer to scroll a discrete fixed amount at each step, you can set the amount of value using the step property.

TooltipVisible

The tooltip displays the value of a slider button in the rangeslider. If the tooltipvisible is true, the tooltips of the slider buttons will always be visible. (Default: false)

Supported Events

Name	Event Type
onRangeValueChange	Event: RangeValueChangeEvent ^[3] Denotes the range value of a component has been changed by the user.

- Inherited Supported Events: XulElement

Supported Children

*None

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
9.0.0	November, 2019	Rangeslider ^[4] was introduced.

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkex/zul/Rangeslider.html>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkex/slider/Rangeslider.html>
- [3] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkex/zul/event/RangeValueChangeEvent.html#>
- [4] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Rangeslider.html>

Searchbox

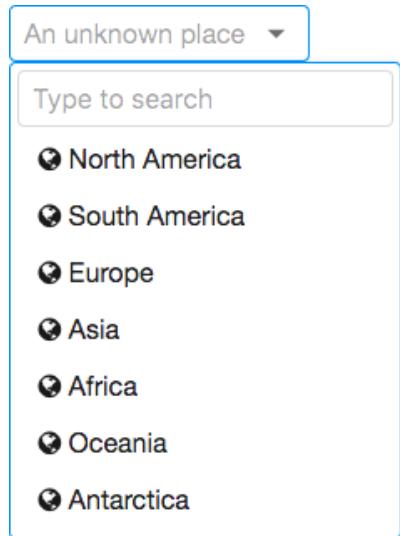
Searchbox

- Demonstration: zkoss-demo/zksearchbox-demo ^[1]
- Java API: Searchbox ^[2]
- JavaScript API: Searchbox ^[3]
- Available for ZK:
-   

Employment/Purpose

A dropdown list that allows users to search and select items.

Example



```
<zscript>
ListModel model = new ListModelArray(new String[] {
    "North America", "South America", "Europe", "Asia", "Africa",
    "Oceania", "Antarctica"
});
</zscript>
<searchbox model="${model}" placeholder="An unknown place" autoclose="true">
    <template name="model">
        <html><! [CDATA[
            <i class="z-icon-globe"></i> ${each}
        ]]></html>
    </template>
</searchbox>
```

Mouseless Entry Searchbox

- UP or DOWN to pop up the list if being focused.
- ESC to close the list.
- UP, DOWN, HOME, END, PAGE UP and PAGE DOWN to change the selection of the item from the list.
- ENTER to confirm the change of selection.
- DELETE or BACKSPACE to clear the selection. (since 9.5.0)

Properties

Autoclose

Sets whether to automatically close the list if a user selected any item. The default value is `false`. It means even if the user selected an item, the list still remains open. You might want to set it as `true` in single selection mode (`multiple=false`).

Disabled

Sets whether it is disabled. A list can still be opened programmatically by calling `open()` even if the component is in the disabled state.

ItemConverter

By implementing your own Converter^[4], you can generate the label that represents the selected items. The default implementation is joining all the `toString()` result of items by commas.

ItemRenderer

See also: ZK_Developer's_Reference/MVC/View/Renderer/Searchbox_Renderer

By implementing your own ItemRenderer^[5], you can generate the HTML fragment for the data model. Normally you would like to use the default implementation and use `<template name="model">` instead.

Model

Since this component doesn't accept any child, you must specify a `ListModel`.

If a `ListSubModel`^[4] is provided, all the items will not be rendered to the client directly. Instead, users must type a keyword to search for a subset of the list. It's suitable for a large model.

Note: If you assign a `ListModel` to a searchbox, you should enable multiple selection with `ListModel`. Please do not set `multiple` on searchbox directly. You should set `multiple` on the model instead.

```
...
List Items = new ArrayList();
for (int i = 0; i < 1000; i++) {
    Items.add("data "+i);
}
ListModelList model = new ListModelList(Items);
model.setMultiple(true);
...
```

```
<searchbox model="${model}" ... />
```

Multiple

Sets whether multiple selections are allowed.

Open

Drops down or closes the list of items.

Placeholder

Sets the placeholder text that is displayed when there's nothing selected.

SearchMessage

Sets the placeholder message of the search text field. The default is "Type to search".

SelectedItem

Returns the selected item, or null if no item is selected. When multiple is true, it returns the first of the selected items.

Don't use MVVM annotations in both `selectedItem` and `selectedItems` at the same time since `@save selectedItem` will deselect all of the currently selected items first.

SelectedItems

Returns all selected items.

Supported Events

Name	Event Type
onAfterRender	Event: Event [7] Notifies one that the model's data has been rendered.
onSelect	Event: SelectEvent [6] Represents an event caused by the user that the list selection is changed at the client.
onOpen	Event: OpenEvent [3] Denotes that the user has opened or closed a component. Note: unlike <code>onClose</code> , this event is only a notification. The client sends this event after opening or closing the component.
onSearching	Event: Event [7] Notifies one that the user is searching by keywords.

- Inherited Supported Events: `HtmlBasedComponent`

Supported Children

* none

Version History

Version	Date	Content
9.0.0	September 2019	ZK-4380 ^[4] : Provide a Searchbox component
9.5.0	August 2020	ZK-4497 ^[5] : searchbox: improve clearing selection, key shortcut / clear icon

References

- [1] <https://github.com/zkoss-demo/zksearchbox-demo>
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Searchbox.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/inp/Searchbox.html#>
- [4] <https://tracker.zkoss.org/browse/ZK-4380>
- [5] <https://tracker.zkoss.org/browse/ZK-4497>

Signature

Signature

- Available for ZK:
-  **EE**
- Java API: Signature ^[1]
- JavaScript API: Signature ^[2]

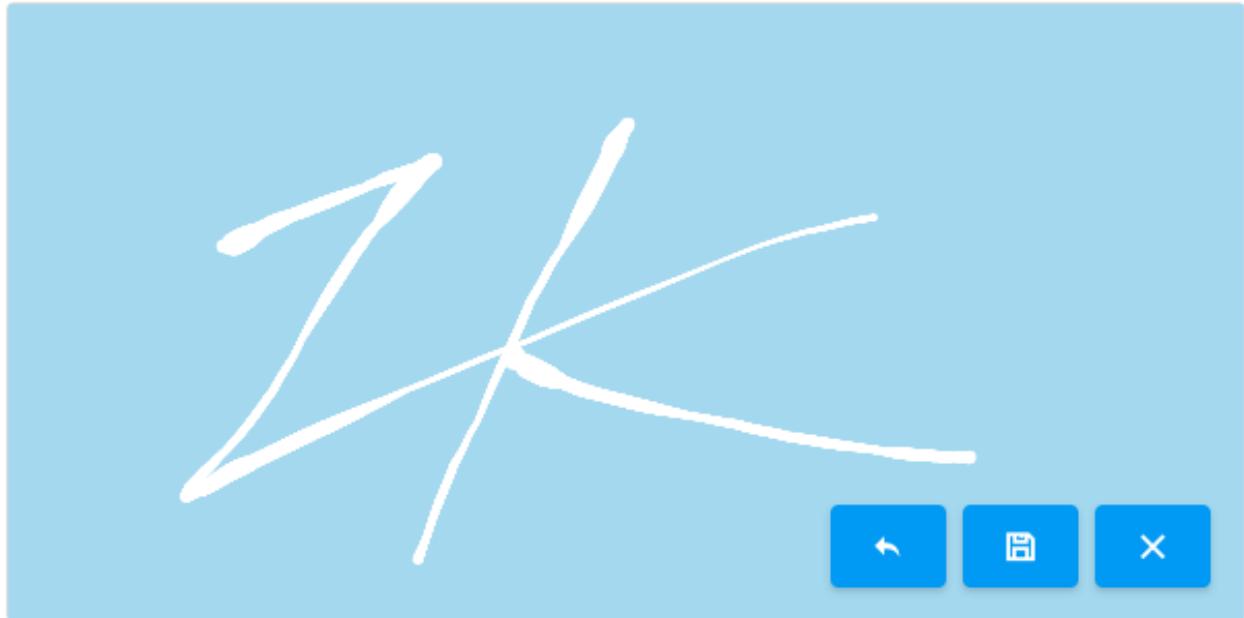
Browser Support

- This component supports IE10+ and modern browsers.

Employment/Purpose

Signature components support signature pad on the desktop and mobile browsers. User can customize pen size, pen color, background color etc., it also provides undo, save and clear methods.

Example



```
<signature width="600px" height="300px" penColor="white" backgroundColor="#AED6F1" penSize="6"/>
```

Buttons

There are 3 buttons when you hover on this component: (from left to right)

1. **undo**: to remove the last step that was drawn on the signature pad.
2. **save**: to save the signature image to the server, a user can get the image by listening onSave event.
3. **clear**: to clear signature pad.

toolbarVisible

The toolbar contain three buttons: undo button, clear button and save button, it is enabled by default. If you don't want to show the toolbar, please use the following setting to hide it.

```
<signature toolbarVisible="false"/>
```

The toolbar button only contains icon by default. If you want to show message after icons on the button, we provide three attributes: undoLabel, clearLabel, saveLabel.

Default:



Customized:

 Undo Save Clear

```
<signature undoLabel="Undo" clearLabel="Clear" saveLabel="Save"/>
```

Style Attributes

There are some attributes to adjust the signature style:

penSize

the width of a line on the signature, the default is 1.

penColor

Can be any color format accepted by context.fillStyle(canvas), defaults to black.

backgroundColor

Can be any color format accepted by context.fillStyle(canvas), defaults to white.

backgroundImage

Can be any image format accepted by context.drawImage(canvas), defaults to null.

BackgroundIncluded

The background color and image will be saved by default. If you don't want to save the background color and image, please use the following setting.

```
<signature backgroundIncluded="false"/>
```

Upload

After calling save method, the signature will be uploaded to server. You can listen onSave to get the uploaded image and integrate with Image^[3].

For example:

```
<signature onSave="image.setContent(event.getMedia())"/>
<image id="image"/>
```

Supported Events

Name	Event Type
onSave	Event: UploadEvent ^[5] When user invoke the save method, the event would be triggered.
onClear	Event: Event ^[7] When user invoke the clear method, the event would be triggered.

Version History

Version	Date	Content
8.6.0	August 2018	

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Signature.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/wgt/Signature.html#>
- [3] https://www.zkoss.org/wiki/ZK_Component_Reference/Essential_Components/Image

Slider

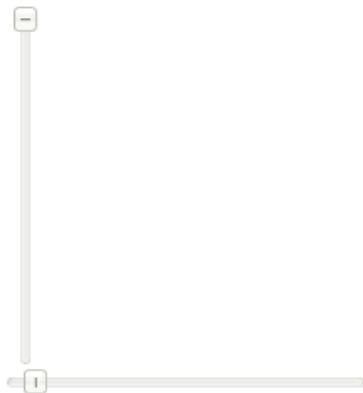
Slider

- Demonstration: Slider ^[1]
- Java API: Slider ^[2]
- JavaScript API: Slider ^[3]
- Style Guide: Slider

Employment/Purpose

A slider component represents a slider with a scale and a knob. It can be used to let user select a value by sliding the knob along the scale. A slider accepts a range of value starting from 0 to certain maximum value. The default maximum value of slider scale is 100. You could change the maximum allowed value by the `maxpos` property. However the default minimum is 0 and cannot be changed.

Example



```
<slider id="slider" orient="vertical"/>
<slider curpos="1" maxpos="20" />
```

Minimal Position

[Since 7.0.1]

Slider supports minimal position, which can be changed by the minpos property as follows.



```
<slider minpos="30"/>
```

Slider also provides `setRange(int, int)` and `setRange(double, double)` methods to help user change the range from minimal position to maximum position.

Decimal Mode

[Since 7.0.1]

Set the mode property to "decimal" will enable decimal slider. So the slider can represent decimal number.



```
<slider mode="decimal" step="0.1"/>
```

Page Increment

By default, the slider will move to the position of the try on which an user clicks. If you prefer to move in a fixed amount (like the scrollbar does), you could specify the amount of value to move by use of Slider.setPageIncrement(int) [4].

```
<slider pageIncrement="10"/>
```

Step

[Since 7.0.1]

By default, the slider will scroll to the position continuously when an user drags it. If you prefer to scroll a discrete fixed amount on each step, you can set the amount of value of the step property. Step property is useful in decimal mode, slider's position value could be rounded to a fixed number by specifying step property. For example, if we want to retrieve the value from decimal slider in the range of 10.0 to 12.0. As the below picture shows, the default decimal show the value contains only one digit in fractional part.



If we want to retrieve the value which contains two digits in fractional part, we can set the step value to 0.01. Then the value will increase as 10.01, 10.02, 10.03 on each step when scrolling the slider. If step is 0.05, the value will increase as 10.05, 10.10, 10.15 on each step, as below.



```
<slider mode="decimal" minpos="10.0" maxpos="12.0" step="0.05"/>
```

Knob Mold

```
[ ZK EE ]  
[ since 8.6.0 ]
```

Set the mold property to "knob" will enable knob slider. So the slider can act as a normal knob. The knob can controlled by wheel, drag, click and enter a value to the input element.



```
<slider mold="knob" minpos="0.0" maxpos="100.0" curpos="80.5" step="0.5" strokeWidth="40"/>
```

AngelArc, StrokeWidth and ScaleInput

AngelArc, strokeWidth and scaleInput are properties only for knob mold. Set the angelArc property with a double for the angle of the knob slider. Set the strokeWidth property with a double for the stroke width of the knob. ScaleInput is the scale ratio of the input size.



```
<slider mold="knob" strokeWidth="40" angelArc="270" scaleInput="1.3" minpos="0.0" maxpos="100.0" curpos="80.5" step="0.5"/>
```

Supported Events

Name	Event Type
onScroll	Event: ScrollEvent [5] Denotes the content of a scrollable component has been scrolled by the user.
onScrolling	Event: ScrollEvent [5] Denotes that the user is scrolling a scrollable component. Notice that the component's content (at the server) won't be changed until onScroll is received. Thus, you have to invoke the <code>getPos</code> method in the ScrollEvent class to retrieve the temporary position.

- Inherited Supported Events: XulElement

Supported Molds

Available molds of a component are defined in lang.xml embedded in zul.jar.

Name	Snapshot
default	
sphere	
scale	
knob	

[Since 7.0.0] [mold knob Since 8.6.0]

the scale mold is deprecated because designs are changed.

Supported Children

*None

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content
5.0.4	August 2010	Slider.setPageIncrement(int) [4] is supported.
5.0.4	August 2010	Slider support for clicking to increment or decrement
7.0.1	January 2014	Slider support minimal position and decimal mode [6]

References

- [1] <http://www.zkoss.org/zkdemo/input/slider>
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Slider.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/inp/Slider.html#>
- [4] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Slider.html#setPageIncrement\(int\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Slider.html#setPageIncrement(int))
- [5] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/ScrollEvent.html#>
- [6] <http://tracker.zkoss.org/browse/ZK-2085>

Sliderbuttons

Sliderbuttons

- Demonstration:
- Java API: Sliderbuttons [1]
- JavaScript API: Sliderbuttons [2]
- Available for ZK:
- CE PE EE

since 9.0.0

Employment/Purpose

A pair of draggable buttons defining the start value and the end value in a Multislider.

Example



```
<zk>
  <multislider>
    <sliderbuttons startValue="10" endValue="70"/>
  </multislider>
</zk>
```

Properties

StartValue, EndValue

Represent the range value. (Default: 0)

Supported Events

Name	Event Type
onRangeValueChange	Event: RangeValueChangeEvent ^[3] Denotes the range value of a component has been changed by the user.

- Inherited Supported Events: XulElement

Supported Children

*None

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content
9.0.0	November, 2019	Sliderbuttons ^[3] was introduced.

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkex/zul/Sliderbuttons.html>

[2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkex/slider/Sliderbuttons.html>

[3] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Sliderbuttons.html>

Spinner

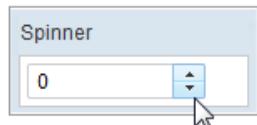
Spinner

- Demonstration: Spinner ^[1]
- Java API: Spinner ^[1]
- JavaScript API: Spinner ^[2]
- Style Guide: Spinner

Employment/Purpose

An edit box for holding a constrained integer.

Example



```
<window title="Spinner" border="normal" width="150px">
    <spinner />
</window>
```

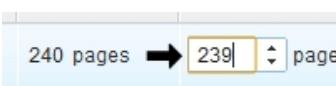
In-place Editing

Fixed Width

```
<spinner width="100px" inplace="true" value="30" />
```

Dynamic Width

Because inplace editing function in ZK is pure client side action, so we can use client api to modify the width (server side do not need to know)



```
<zk xmlns:c="client">
    <spinner inplace="true" value="240" width="30px" c:onFocus='this.setWidth("60px")' c:onBlur='this.setWidth("30px")' />
</zk>
```

Properties

Format

You are able to format the field by providing specifying the attribute with a formatting string. The default value is null.

```
<spinner format="#,##0.##"/>
```

Since 8.5.2

You can provide a locale to format the number by specify the String starts with "locale:"

```
<spinner format="locale:zh-TW"/>
```

Constraint

You could specify what value to accept for input controls by use of the `constraint` property. It could be a combination of no `empty` and the minimum and maximum to spinner.

To specify two or more constraints, use comma to separate them as follows.

```
<spinner constraint="no empty,min -2 max 6"/>
```

If you prefer to display different message to the default one, you can append the error message to the constraint with a colon.

```
<spinner constraint="no empty,min -2 max 6: between -2 to 6"/>
```

Notes:

- The error message, if specified, must be the last element and start with colon.
- To support multiple languages, you could use the `「1」` function as depicted in the **Internationalization** chapter.

```
<spinner constraint="no empty,min -2 max 6: ${c:l('err.msg.spinner')}"/>
```

Inherited Functions

Please refer to `NumberInputElement` for inherited functions.

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: `NumberInputElement`

Supported Molds

Available molds of a component are defined in lang.xml embedded in zul.jar.

Name	Snapshot
default	
rounded	[Since 5.0.0]

Supported Children

*None

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Spinner.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/inp/Spinner.html#>

Tbeditor

Tbeditor

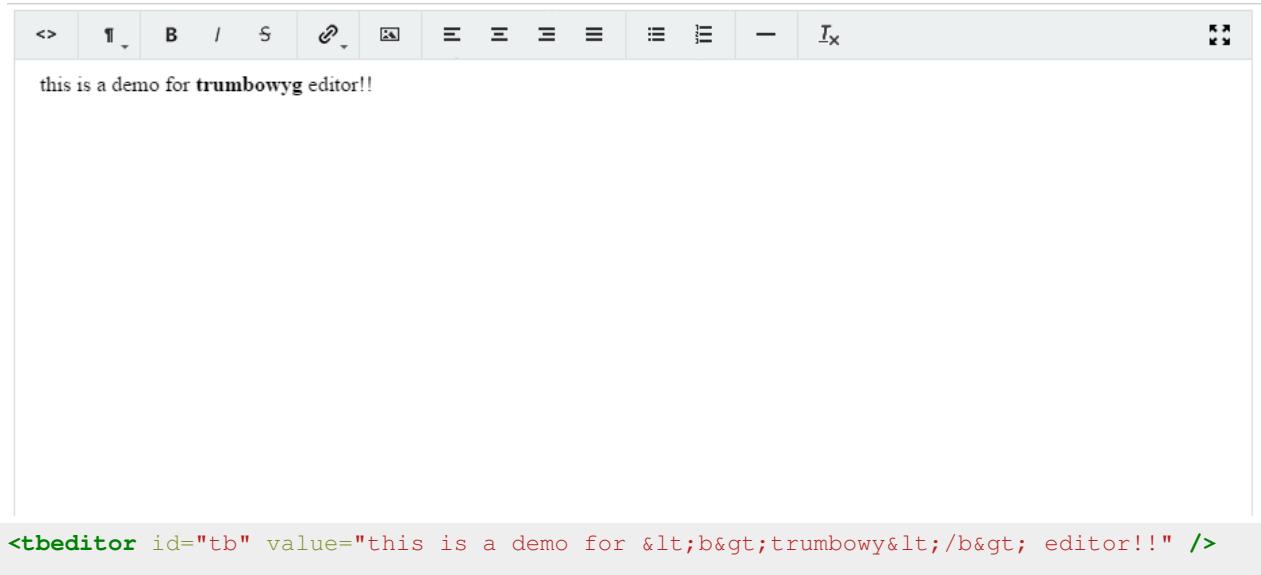
- Java API: N/A
- JavaScript API: N/A

Employment/Purpose

The component used to represent Trumbowyg^[1]

Tbeditor is a rich text editor to be used inside web pages. It's a WYSIWYG editor, which means that the text being edited on it looks as similar as possible to the results users have when publishing it.

Example



Customized Properties

Tbeditor provides a way for users to customize their own properties, check official document^[2] for the detail^[3]. Here shows a simple example how to programmatic change the property.

```
Map config = new HashMap();
config.put("btns", new JavaScriptValue("[ 'bold', 'italic', '|', 'link' ]"));
config.put("closable", true);
tb.setConfig(config);
```

We create a map while key is property name with reasonable value. Note that we have to wrap the value into `JavaScriptValue` object if it's not String.

-
- [1] <http://alex-d.github.io/Trumbowyg/>
 [2] <http://alex-d.github.io/Trumbowyg/documentation.html>
 [3] Not support all properties, for example, localization, custom skin, are not supported.

Supported Events

Name	Event Type
onChange	InputEvent (http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/InputEvent.html#) Description: Denotes the content of an input component has been modified by the user.
onChanging	InputEvent (http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/InputEvent.html#) Description: Denotes that user is changing the content of an input component. Notice that the component's content (at the server) won't be changed until onChange is received. Thus, you have to invoke the <code>getValue</code> method in the InputEvent class to retrieve the temporary value.

Supported Children

*NONE

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content

Textbox

Textbox

- Demonstration: Textbox ^[1]
- Java API: Textbox ^[1]
- JavaScript API: Textbox ^[2]
- Style Guide: Textbox

Employment/Purpose

A **textbox** is used to let users input textual data.

You could assign `value`, `type`, `constraint`, `rows`, `cols` to a textbox using the corresponding properties. When you assign the property `type` to a string value "password" when `multiline` is false (`multiline` will be `true` if you set `rows` larger than 1 or set `multiline` to `true` directly) then any character in this component will replace by '*'.

You could also assign a constraint value with a regular expression string or a default constraint expression (available value is "no empty"). When user change the value of textbox, it will cause a validating process to validate the value. If the validation fails, then a notification will pop up.

Example

The screenshot shows a user interface with three text input fields. The first field contains "text...". The second field contains "*****". The third field has the placeholder "Not mail format". A red validation message box is displayed, containing a warning icon and the text "Please enter an e-mail address". Above the message box, two lines of text are visible: "text line1..." and "text line2...".

```
<textbox value="text..." />
<textbox value="secret" type="password" />
<textbox constraint="/.+@.+\.+[a-z]+/: Please enter an e-mail address" />
<textbox rows="5" cols="40">
    <attribute name="value">
        text line1...
        text line2...
    </attribute>
</textbox>
```

To specify `multilines` value, you can use the `attribute` element or `\n` as shown below

```
<textbox rows="5" cols="40">
    <attribute name="value">
        text line1...
        text line2...
    </attribute>
</textbox>
<textbox value="Line 1\nLine 2" rows="3"/>
```

Properties

Tabbable

By specifying a true, the tabbox can insert a long space or format the content inside textbox conveniently. For example,



```
<textbox tabbable="true"/>
```

SubmitByEnter

since 8.5.2

When you specify **true**, pressing Enter will fire onOK event rather than move to next line, if you want to move to next line, you should press **Shift + Enter**.

When submitByEnter="false", press Enter will move to next line.

The default is **false**.

```
<textbox submitByEnter="true" onOK="" />
```

Type

The `type` attribute can be used with the `textbox` based components and can be given the value `password`. By setting the type as `password` the text that is entered into the box cannot be viewed and is replaced by stars.

```
Username: <textbox/>  
Password: <textbox type="password"/>
```

since 6.5.0

The `type` attribute supports HTML 5 defined types `tel`, `email` and `url`.

```
Phone: <textbox type="tel"/>  
Email: <textbox type="email"/>  
WebSite: <textbox type="url"/>
```

Constraint

Please refer to ZK Component Reference/Base Components/InputElement#Constraint.

Inherited Functions

Please refer to InputElement for inherited functions, such as in-place edition.

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: InputElement

Supported Children

*NONE

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Browser Limitations

Browser	description
IE	<code><textbox value="color" style="color:red !important;" disabled="true"/></code> There is no way to change the text color in a disabled input in IE.

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Textbox.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/inp/Textbox.html#>

Timebox

Timebox

- Demonstration: Date and Time ^[1]
- Java API: Timebox ^[1]
- JavaScript API: Timebox ^[2]
- Style Guide: Timebox

Employment/Purpose

An edit box for holding a time (a java.util.Date Object), but only Hour & Minute are used.

Example

 ^ ▼

```
<timebox cols="11"/>
```

Properties

Constraint

You could specify what value to accept for input controls by use of the `constraint` property. It could be no empty.

If you prefer to display different message to the default one, you can append the error message to the constraint with a colon.

```
<timebox constraint="no empty: cannot be empty"/>
```

Notes:

- The error message, if specified, must be the last element and start with colon.
- To support multiple languages, you could use the `['1']` function as depicted in the **Internationalization** chapter.

```
<timebox constraint="no empty: ${c:l('err.time.required')}"/>
```

Format

Use *a* to signify it is *am* or *pm*. The input string follows the formatting of the `SimpleDateFormat` ^[3].

Below is an example of using *a* within the format.

```
<timebox cols="20" format="a hh:mm:ss"/>
```

24 hours mode:

```
<timebox cols="8" format="HH:mm:ss"/>
```

In addition to specifying the format explicitly, you could specify the styling^[4]. There are four different types of styling: short, medium, long and full (representing the styling of java.text.DateFormat). For example, you could specify the styling rather than the real format as follows.

```
<timebox format="short"/>  
<timebox format="long"/>
```

Then the real format of the timebox will be decided at run time depending the configuration. For more information, please refer to ZK Developer's Reference: Date and Time Formatting.

-
- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Timebox.html#>
 - [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/db/Timebox.html#>
 - [3] <http://java.sun.com/j2se/1.5.0/docs/api/java/text/SimpleDateFormat.html>
 - [4] The styling is available since 5.0.7

Locale

By default, the real format depends on the current locale (i.e., Locales.getCurrent()) ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/util/Locales.html#getCurrent\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/util/Locales.html#getCurrent())). However, you could specify the locale for an individual instance such as:

```
<timebox format="medium" locale="de_DE"/>  
<timebox format="long" locale="fr"/>
```

Text

You should set `text` attribute after `format` attribute or ZK might not convert the specified text well.

```
<timebox format="hh:mm:ss a" locale="en" text="12:00:00 AM"/>
```

- If you set `format` later than setting `text` attribute, ZK might probably fail to convert the text to a `Date` object according to default format and throw `org.zkoss.zk.ui.WrongValueException`.

Inherited Functions

Please refer to `FormatInputElement` for inherited functions.

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: `FormatInputElement`

Supported Molds

Available molds of a component are defined in `lang.xml` embedded in `zul.jar`.

Name	Snapshot
default	00:01 ▾
rounded	00:01 ▾ [Since 5.0.0]

Supported Children

*NONE

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
5.0.7	April, 2011	Timebox.setFormat(java.lang.String) (http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Timebox.html#setFormat(java.lang.String)) supported the styling.
5.0.7	April, 2011	Timebox.setLocale(java.util.Locale) (http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Timebox.html#setLocale(java.util.Locale)) was introduced to specify a locale other than the current locale.

Timepicker

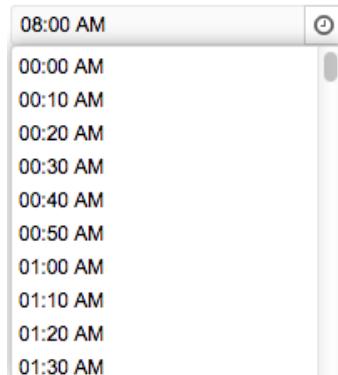
Timepicker

- Java API: Timepicker ^[1]
- JavaScript API: Timepicker ^[2]
- Style Guide: Timepicker

Employment/Purpose

A selection box for holding a time (a java.util.Date Object) , but only Hour, Minute, and Second are used.

Example



```
<timepicker/>
```

Properties

Format

Use *a* to signify it is *am* or *pm*. The input string follows the formatting of the SimpleDateFormat ^[3].

Below is an example of using *a* within the format.

```
<zkb>
    <window title="Test">
        <timepicker format="a hh:mm:ss"/>
    </window>
</zkb>
```

In addition to specifying the format explicitly, you could specify the styling^[3]. There are two different types of styling: short, medium (representing the styling of java.text.DateFormat). For example, you could specify the styling rather than the real format as follows.

```
<timepicker format="short"/>
<timepicker format="medium"/>
```

Then the real format of the timepicker will be decided at run time depending the configuration. For more information, please refer to ZK Developer's Reference: Date and Time Formatting.

-
- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Timepicker.html#>
 - [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/inp/Timepicker.html#>
 - [3] The styling is available since 5.0.7

Minimum Time

By default, the options of timepicker start at 0:00 AM, you could specify another minimum time by using a java.util.Date Object.

```
<zscript>
    import java.util.Date;
    Date min = new Date();
    min.setTime(0);
</zscript>
<timepicker format="HH:mm a" min="${min}" />
```

Maximum Time

By default, the options of timepicker end before 12:00 AM, you could specify another maximum time by using a java.util.Date Object.

```
<zscript>
    import java.util.Date;
    Date max = new Date();
    max.setTime(0);
</zscript>
<timepicker format="HH:mm a" max="${max}" />
```

Interval

By default, the interval of the options in timepicker is one hour, you could specify the interval by using an integer (unit: 1 second).

```
<timepicker format="HH:mm a" interval="600" />
```

Inherited Functions

Please refer to FormatInputElement for inherited functions.

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: FormatInputElement

Supported Children

*NONE

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

Layouts

This section outlines components which enable developers to control the layout of their application.

For introduction, please refer to ZK Developer's Reference: Layouts and Containers

Absolutelayout

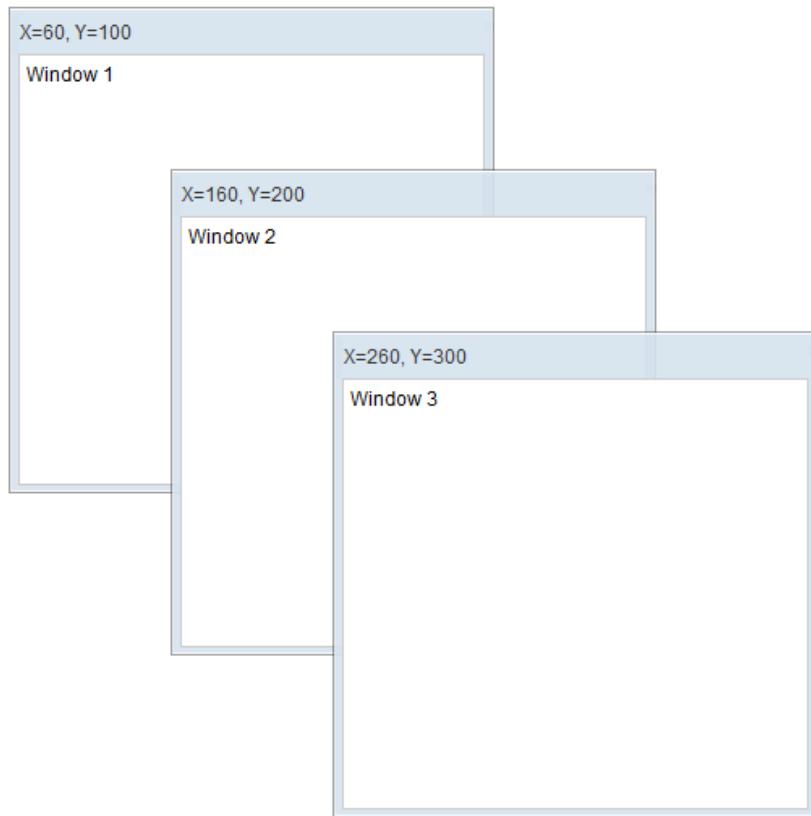
Absolutelayout

- Demonstration: N/A
- Java API: Absolutelayout ^[1]
- JavaScript API: Absolutelayout ^[2]
- Style Guide: N/A

Employment/Purpose

An Absolutelayout component can contain absolute positioned multiple absolutechildren components.

Example



```
<?component name="window" extends="window" border="normal" width="300px" height="300px"?>
<zkc>
    <absolutelayout>
        <absolutechildren id="w1" x="60" y="100">
            <window title="X=60, Y=100">
                Window 1
            </window>
        </absolutechildren>
        <absolutechildren id="w2" x="160" y="200">
            <window title="X=160, Y=200">
                Window 2
            </window>
        </absolutechildren>
        <absolutechildren id="w3" x="260" y="300">
            <window title="X=260, Y=300">
                Window 3
            </window>
        </absolutechildren>
    </absolutelayout>
</zkc>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

*Absolutechildren

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content
6.0.0	October 4, 2011	Add the new Absolutelayout component

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Absolutelayout.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/layout/Absolutelayout.html#>

Absolutechildren

Absolutechildren

- Demonstration: N/A
- Java API: Absolutechildren [1]
- JavaScript API: Absolutechildren [2]
- Style Guide: N/A

Employment/Purpose

A container component that can contain any other ZK component and can only be contained as direct child of Absolutelayout component. It can be absolutely positioned within Absolutelayout component by either setting "x" and "y" attribute or calling setX(int) and setY(int) methods.

Example



```
<?component name="window" extends="window" border="normal" width="300px" height="300px"?>
<zk>
    <absolutelayout>
        <absolutechildren id="w1" x="60" y="100">
            <window title="X=60, Y=100">
                Window 1
            </window>
        </absolutechildren>
```

```

<absolutechildren id="w2" x="160" y="200">
    <window title="X=60, Y=100">
        Window 2
    </window>
</absolutechildren>
<absolutechildren id="w3" x="260" y="300">
    <window title="X=60, Y=100">
        Window 3
    </window>
</absolutechildren>
</absoluteLayout>
</zk>

```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

*All

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content
6.0.0	October 4, 2011	Add the new Absolutechildren component

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Absolutechildren.html#>
[2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/layout/Absolutechildren.html#>

Anchorlayout

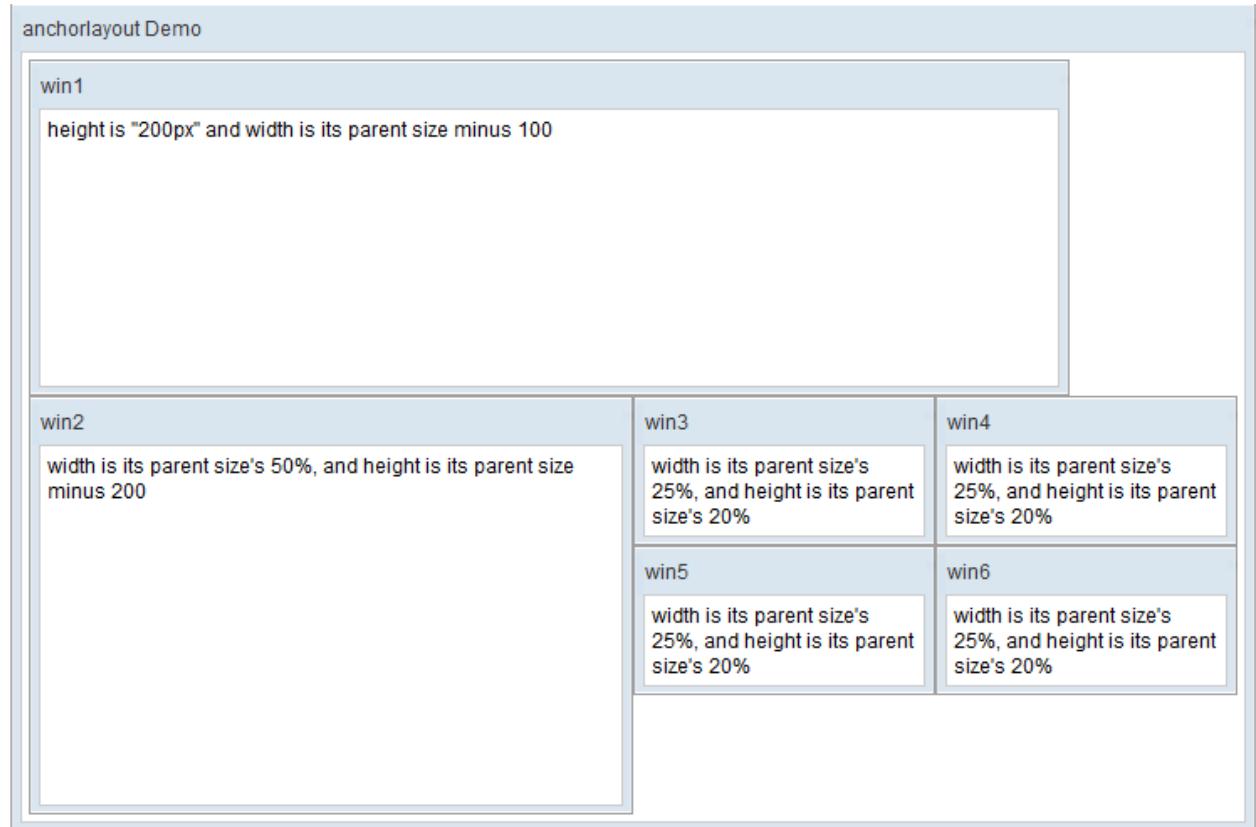
Anchorlayout

- Demonstration: N/A
- Java API: Anchorlayout [1]
- JavaScript API: Anchorlayout [2]
- Style Guide: N/A

Employment/Purpose

An anchorlayout lays out a container which can resize its children base on its width and height

Example



```
<window title="anchorlayout Demo" border="normal" width="100%" height="100%>
    <anchorlayout id="al" width="100%" vflex="1" style="overflow:auto">
        <anchordchildren height="200px" anchor="-100">
            <window title="win1" border="normal" width="100%" height="100%">
                height is "200px" and width is its parent size minus 100
            </window>
        </anchordchildren>
    </anchorlayout>
</window>
```

```
</anchordchildren>
<anchordchildren anchor="50% -200">
    <window title="win2" border="normal" width="100%" height="100%">
        width is its parent size's 50%, and height is
its parent
        size minus 200
    </window>
</anchordchildren>
<anchordchildren anchor="25% 20%">
    <window title="win3" border="normal" width="100%" height="100%">
        width is its parent size's 25%, and height is
its parent
        size's 20%
    </window>
</anchordchildren>
<anchordchildren anchor="25% 20%">
    <window title="win4" border="normal" width="100%" height="100%">
        width is its parent size's 25%, and height is
its parent
        size's 20%
    </window>
</anchordchildren>
<anchordchildren anchor="25% 20%">
    <window title="win5" border="normal" width="100%" height="100%">
        width is its parent size's 25%, and height is
its parent
        size's 20%
    </window>
</anchordchildren>
<anchordchildren anchor="25% 20%">
    <window title="win6" border="normal" width="100%" height="100%">
        width is its parent size's 25%, and height is
its parent
        size's 20%
    </window>
</anchordchildren>
</anchordchildren>
</anchorlayout>
</window>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

*Anchordchildren

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content
6.0.0	October 4, 2011	Add the new Anchorlayout component

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Anchorlayout.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/layout/Anchorlayout.html#>

Anchorchildren

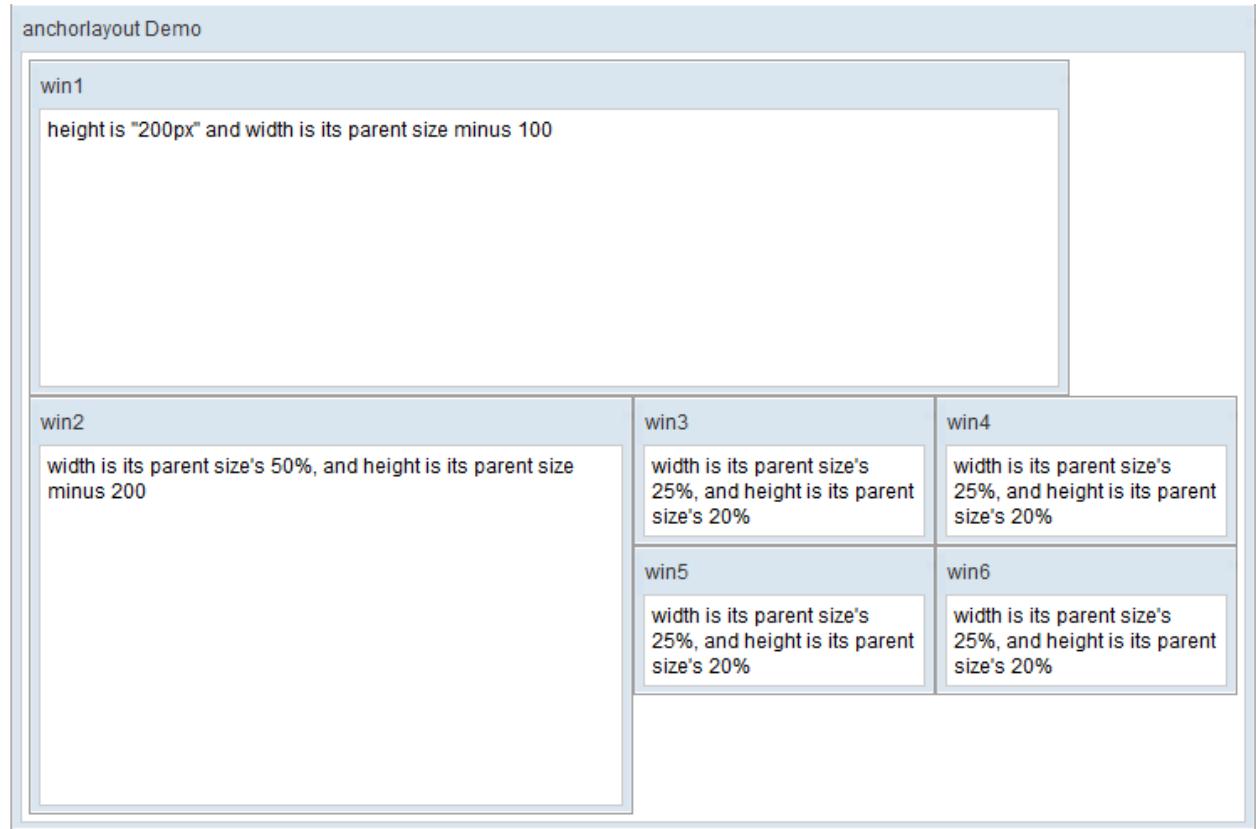
Anchorchildren

- Demonstration: N/A
- Java API: Anchorchildren [1]
- JavaScript API: Anchorchildren [2]
- Style Guide: N/A

Employment/Purpose

The children of Anchorlayout that can anchor to the position that according to the size of the Anchorlayout.

Example



```
<window title="anchorlayout Demo" border="normal" width="100%" height="100%>
    <anchorlayout id="al" width="100%" vflex="1" style="overflow:auto">
        <anchorchildren height="200px" anchor="-100">
            <window title="win1" border="normal" width="100%" height="100%">
                height is "200px" and width is its parent size minus 100
            </window>
        </anchorchildren>
    </anchorlayout>
</window>
```

```
</anchorchildren>
<anchorchildren anchor="50% -200">
    <window title="win2" border="normal" width="100%"
        height="100%">
        width is its parent size's 50%, and height is
its parent
        size minus 200
    </window>
</anchorchildren>
<anchorchildren anchor="25% 20%">
    <window title="win3" border="normal" width="100%"
        height="100%">
        width is its parent size's 25%, and height is
its parent
        size's 20%
    </window>
</anchorchildren>
<anchorchildren anchor="25% 20%">
    <window title="win4" border="normal" width="100%"
        height="100%">
        width is its parent size's 25%, and height is
its parent
        size's 20%
    </window>
</anchorchildren>
<anchorchildren anchor="25% 20%">
    <window title="win5" border="normal" width="100%"
        height="100%">
        width is its parent size's 25%, and height is
its parent
        size's 20%
    </window>
</anchorchildren>
<anchorchildren anchor="25% 20%">
    <window title="win6" border="normal" width="100%"
        height="100%">
        width is its parent size's 25%, and height is
its parent
        size's 20%
    </window>
</anchorchildren>
</anchorlayout>
</window>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

*All

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content
6.0.0	October 4, 2011	Add the new Anchorchildren component

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Anchorchildren.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/layout/Anchorchildren.html#>

Borderlayout

Borderlayout

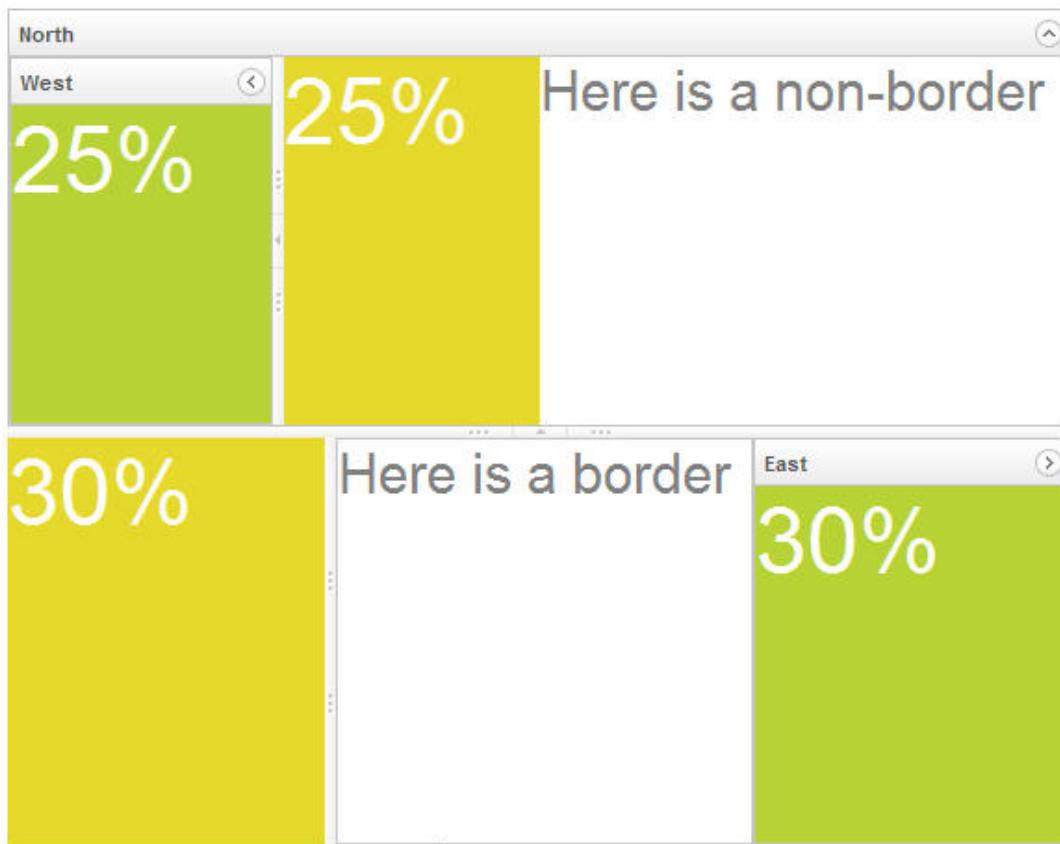
- Demonstration: Borderlayout ^[1]
- Java API: Borderlayout ^[2]
- JavaScript API: Borderlayout ^[3]
- Style Guide: Borderlayout

Employment/Purpose

The layout component is a nested component. The parent component is borderlayout, and its children components include north, south, center, west, and east. All extra space is placed in the center area. The combination of children components of borderlayout is free.

A borderlayout could be nested to another borderlayout (actually, almost all kinds of components) to form a complicated layout.

Example



```
<borderlayout height="450px">
    <north title="North" maxsize="300" size="50%" splittable="true" collapsible="true">
        <borderlayout>
            <west title="West" size="25%" flex="true" maxsize="250" splittable="true" collapsible="true">
                <div style="background:#B8D335">
                    <label value="25%">
```

</div>
</west>
<center border="none">
<div style="background:#E6D92C" vflex="1">
<label value="25%" style="color:white;font-size:50px"
/>
</div>
</center>
<east size="50%" border="none">
<label value="Here is a non-border" style="color:gray;font-size:30px" />
</east>
</borderlayout>
</north>
<center border="0">
<borderlayout>
<west maxsize="600" size="30%" border="0" splittable="true">
<div style="background:#E6D92C" vflex="1">
<label value="30%" style="color:white;font-size:50px"
/>
</div>
</west>
<center>
<label value="Here is a border" style="color:gray;font-size:30px" />
</center>
<east title="East" size="30%" collapsible="true">
<div style="background:#B8D335" vflex="1">
<label value="30%" style="color:white;font-size:50px"
/>
</div>
</east>
</borderlayout>
</center>
</borderlayout>

How to Layout

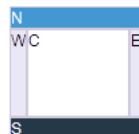
Borderlayout divides its child components into to five areas: North, South, East, West and Center. The heights of North and South are firstly decided, the remainder space is then given to Center as its height. Note that East and West also takes on the height of Center.



```
<borderlayout width="100px" height="100px">
    <north>
        <div style="background:#008db7;color:white;">N</div>
    </north>
    <south>
        <div style="background:#112f37;color:white;">S</div>
    </south>
    <center>
        <div>C</div>
    </center>
    <east>
        <div style="background:#f2f2f2;">E</div>
    </east>
    <west>
        <div style="background:#f2f2f2;">W</div>
    </west>
</borderlayout>
```

flex

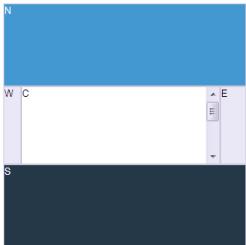
Layout region shares the height of Borderlayout with a distributing sequence of: North, South and Center while the heights of East and West takes on the height of Center. In the previous sample, the div in the layout region does not take up all of layout region's space. In order for the child to occupy the whole area, please set vflex="1" to the child component.



```
<borderlayout width="100px" height="100px">
    <north>
        <div style="background:#008db7;color:white;">N</div>
    </north>
    <south>
        <div style="background:#112f37;color:white;">S</div>
    </south>
    <center>
        <div>C</div>
    </center>
    <east>
        <div vflex="1" style="background:#f2f2f2;">E</div>
    </east>
    <west>
        <div vflex="1" style="background:#f2f2f2;">W</div>
    </west>
</borderlayout>
```

Scrolling

- The height of Center depends on Borderlayout but not on its child, therefore, the height of Center will not be expanded by the growing size of its child components. If Center's height is too short for it's child, Center will cut out the contents of it's child, hence, to avoid this, specify autoscroll="true" to Center in order to assign Center to handle the scrolling.



```
<font size="7.76">
<borderlayout width="300px" height="300px">
    <north>
        <div height="100px" style="background:#008db7;color:white;">N</div>
    </north>
    <south>
        <div height="100px" style="background:#112f37;color:white;">S</div>
    </south>
    <center autoscroll="true">
        <div height="200px">C</div>
    </center>
    <east flex="true">
        <div width="30px" style="background:#f2f2f2;">E</div>
    </east>
    <west flex="true">
        <div width="20px" style="background:#f2f2f2;">W</div>
    </west>
</borderlayout>
</font>
```

[Since 7.0.0]

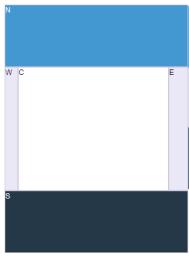
The autoscroll attribute will create floating scrollbar and it is not visible unless user mouse over on the region. To turn off the floating scrollbar and use browser's default scrollbar, please add the following configuration in zk.xml.

```
<library-property>
    <name>org.zkoss.zul.nativebar</name>
    <value>true</value>
</library-property>
```

Note: the value of org.zkoss.zul.nativebar is true by default (since 7.0.2)

Grown by children

- To make Borderlayout dependable on the size of its child components, vflex feature is applied. Specify vflex="min" to each layout region and Borderlayout.



```
<font size="8.19">
<borderlayout width="300px" vflex="min">
    <north vflex="min">
        <div height="100px" style="background:#008db7;color:white;">N</div>
    </north>
    <south vflex="min">
        <div height="100px" style="background:#112f37;color:white;">S</div>
    </south>
    <center vflex="min">
        <div height="200px">C</div>
    </center>
    <east flex="true">
        <div width="30px" style="background:#f2f2f2;">E</div>
    </east>
    <west flex="true">
        <div width="20px" style="background:#f2f2f2;">W</div>
    </west>
</borderlayout>
</font>
```

Borderlayout in a container

- Almost all containers' heights depend on their child components, however, the height of Borderlayout does not expand accordingly to the sizes of its child components, therefore, when placing Borderlayout in a container, users have to specify a fixed height in order for Borderlayout to be visible.

```
<zK>
<window title="win" border="normal">
    <borderlayout height="200px">
        <north>
            <div style="background:blue">N</div>
        </north>
        <south>
            <div style="background:blue">S</div>
        </south>
        <center>
            <div>C</div>
        </center>
        <east>
            <div style="background:yellow">E</div>
        </east>
        <west>
            <div style="background:yellow">W</div>
        </west>
    </borderlayout>
</window>
</zK>
```

- The default height of Borderlayout is dependent on its parent component, therefore, users can also put Borderlayout in a container with a fixed height.

```
<zk>
    <window title="win" border="normal" height="200px">
        <borderlayout>
            <north>
                <div style="background:blue">N</div>
            </north>
            <south>
                <div style="background:blue">S</div>
            </south>
            <center>
                <div>C</div>
            </center>
            <east>
                <div style="background:yellow">E</div>
            </east>
            <west>
                <div style="background:yellow">W</div>
            </west>
        </borderlayout>
    </window>
</zk>
```

Properties

AnimationDisabled

You can specify this property to true to disable the animation effects of this component.

Configure to Disable the Animation Effects as Default

If you prefer to disable the animation effects as default, you could configure ZK by adding the following to /WEB-INF/zk.xml

```
<library-property>
    <name>org.zkoss.zul.BorderLayout.animation.disabled</name>
    <value>true</value>
</library-property>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: HtmlBasedComponent

Supported Children

* North, South, Center, West, East

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
5.0.8	August 11, 2011	Added a way to disable the animation of borderlayout.
6.0.0	Feb 14, 2012	The flex attribute has been deprecated, please set vflex="1" to the child component in order to occupy the whole area.
7.0.2	April 2014	Due to the better user-friendly for the scrollbar layout, we changed the org.zkoss.zul.nativebar of the library property to true by default for Grid, Listbox, Tree and Borderlayout component.

References

- [1] http://www.zkoss.org/zkdemo/layout/border_layout
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Borderlayout.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/layout/Borderlayout.html#>

Center

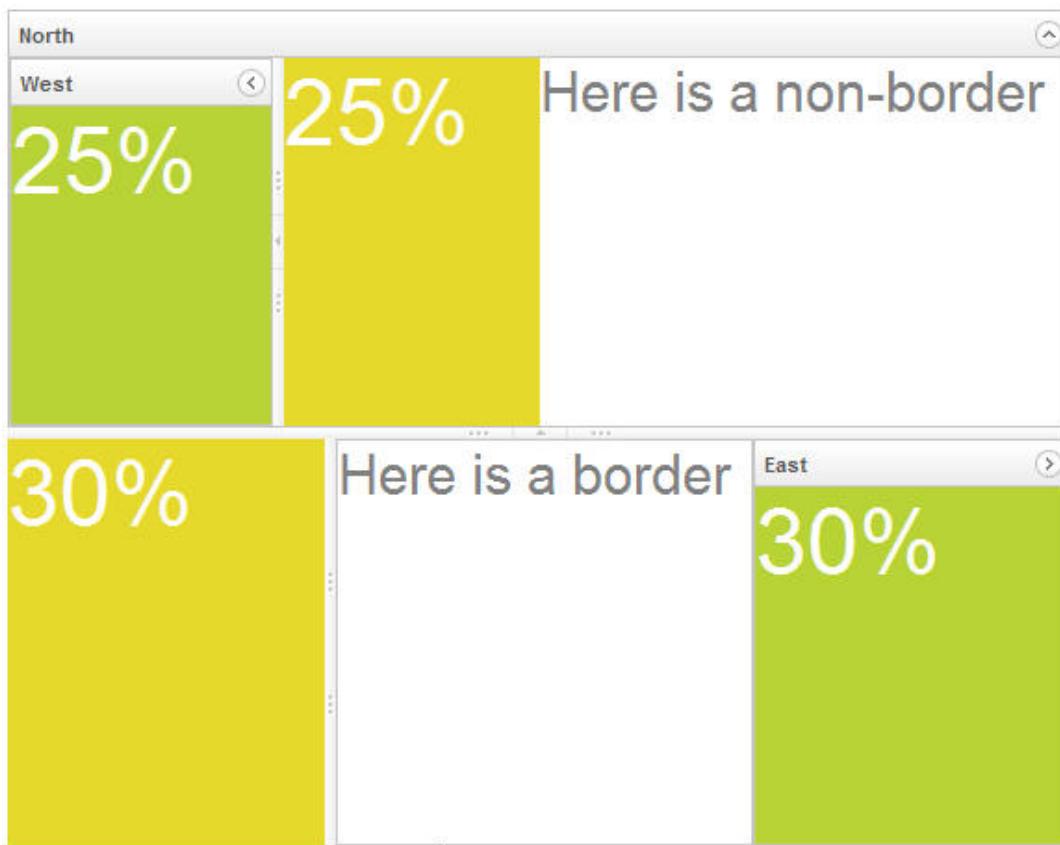
Center

- Demonstration: Borderlayout [1]
- Java API: Center [1]
- JavaScript API: Center [2]
- Style Guide: Center

Employment/Purpose

A center region of a border layout and only allows one component as its child.

Example



```
<borderlayout height="450px">
    <north title="North" maxsize="300" size="50%" splittable="true" collapsible="true">
        <borderlayout>
            <west title="West" size="25%" flex="true" maxsize="250" splittable="true" collapsible="true">
                <div style="background:#B8D335">
                    <label value="25%" style="color:white;font-size:50px">
                </div>
            </west>
        </borderlayout>
    </north>
    <center title="Center" maxsize="500" size="50%">
        <div style="background-color:#FFFF00; width:100%; height:100%; text-align:center; font-size:24px; padding-top:10px">
            Here is a non-border
        </div>
    </center>
    <south title="South" maxsize="300" size="50%">
        <div style="background-color:#FFFF00; width:100%; height:100%; text-align:center; font-size:24px; padding-top:10px">
            Here is a border
        </div>
    </south>
    <east title="East" maxsize="300" size="50%">
        <div style="background:#B8D335; width:100%; height:100%; text-align:center; font-size:24px; padding-top:10px">
            30%
        </div>
    </east>
</borderlayout>
```

```
<center border="none">
  <div style="background:#E6D92C" vflex="1">
    <label value="25%" style="color:white;font-size:50px"
  />
  </div>
</center>
<east size="50%" border="none">
  <label value="Here is a non-border" style="color:gray;font-size:30px" />
</east>
</borderlayout>
</north>
<center border="0">
  <borderlayout>
    <west maxsize="600" size="30%" border="0" splittable="true">
      <div style="background:#E6D92C" vflex="1">
        <label value="30%" style="color:white;font-size:50px"
      />
      </div>
    </west>
    <center>
      <label value="Here is a border" style="color:gray;font-size:30px" />
    </center>
    <east title="East" size="30%" collapsible="true">
      <div style="background:#B8D335" vflex="1">
        <label value="30%" style="color:white;font-size:50px"
      />
      </div>
    </east>
  </borderlayout>
</center>
</borderlayout>
```

Properties and Features

Caption

A layout region might have a caption, which is specified by declaring a child component called caption.

- Available for ZK:
-   

```
<borderlayout>
  <center>
    <caption label="search" image="/img/live.gif">
      <combobox>
        <comboitem label="item 1" />
        <comboitem label="item 2" />
        <comboitem label="item 3" />
        <comboitem label="item 4" />
      </combobox>
    </caption>
    <div>
      Content
    </div>
  </center>
</borderlayout>
```

How to Layout

For more details, please refer to Borderlayout.

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: LayoutRegion

Supported Children

* ALL

Use Cases

Borderlayout

Version History

Version	Date	Content
6.5.0	June 2012	ZK-969 [3]: The LayoutRegion component support caption component as it's title

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Center.html#>

[2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/layout/Center.html#>

[3] <http://tracker.zkoss.org/browse/ZK-969>

East

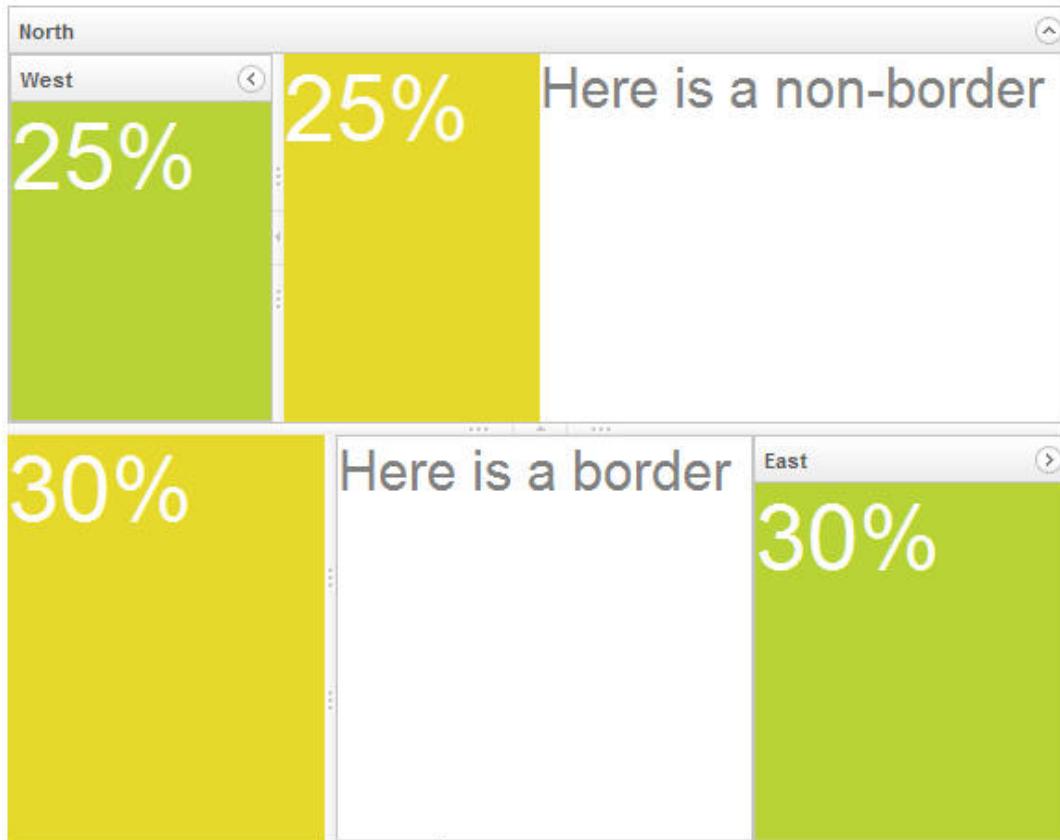
East

- Demonstration: Borderlayout [1]
- Java API: East [1]
- JavaScript API: East [2]
- Style Guide: East

Employment/Purpose

An east region of a border layout and only allows one component as its child.

Example



```
<borderlayout height="450px">

    <north title="North" maxsize="300" size="50%" splittable="true" collapsible="true">
        <borderlayout>
            <west title="West" size="25%" flex="true" maxsize="250" splittable="true" collapsible="true">
                <div style="background:#B8D335">
                    <label value="25%" style="color:white;font-size:50px">
                </div>
            </west>
            <center border="none">
                <div style="background:#E6D92C vflex="1">
                    <label value="25%" style="color:white;font-size:50px">
                </div>
            </center>
            <east size="50%" border="none">
                <label value="Here is a non-border" style="color:gray;font-size:30px" />
            </east>
        </borderlayout>
    </north>
    <center border="0">
```

```
<borderlayout>
    <west maxsize="600" size="30%" border="0" splittable="true">
        <div style="background:#E6D92C" vflex="1">
            <label value="30%" style="color:white;font-size:50px"
        />
            </div>
        </west>
        <center>
            <label value="Here is a border" style="color:gray;font-size:30px" />
        </center>
        <east title="East" size="30%" collapsible="true">
            <div style="background:#B8D335" vflex="1">
                <label value="30%" style="color:white;font-size:50px"
            />
            </div>
        </east>
    </borderlayout>
</center>
</borderlayout>
```

How to Layout

For more details, please refer to Borderlayout.

Properties and Features

Caption

A layout region might have a caption, which is specified by declaring a child component called caption.

[ZK EE]
[Since 6.5.0]

```
<borderlayout>
    <east>
        <caption label="search" image="/img/live.gif">
            <combobox>
                <comboitem label="item 1" />
                <comboitem label="item 2" />
                <comboitem label="item 3" />
                <comboitem label="item 4" />
            </combobox>
        </caption>
        <div>
            Content
        </div>
    </east>
</borderlayout>
```

```
</div>
</east>
</borderlayout>
```

Closable

Whether users can open or close the region. Require `collapsible="true"`. Default: true.

[Since 8.5.2]

Slidable

Whether users can slide (preview) the region when clicked on a collapsed region. Require `collapsible="true"`. Default: true.

[Since 8.5.2]

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: LayoutRegion

Supported Children

*ALL

Use Cases

Borderlayout

Version History

Version	Date	Content
6.5.0	June 2012	ZK-969 [3]: The LayoutRegion component support caption component as it's title
8.5.2	May 2018	ZK-3329 [3]: Collapsible Borderlayout region in slide or open mode only

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/East.html#>

[2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/layout/East.html#>

[3] <http://tracker.zkoss.org/browse/ZK-3329>

North

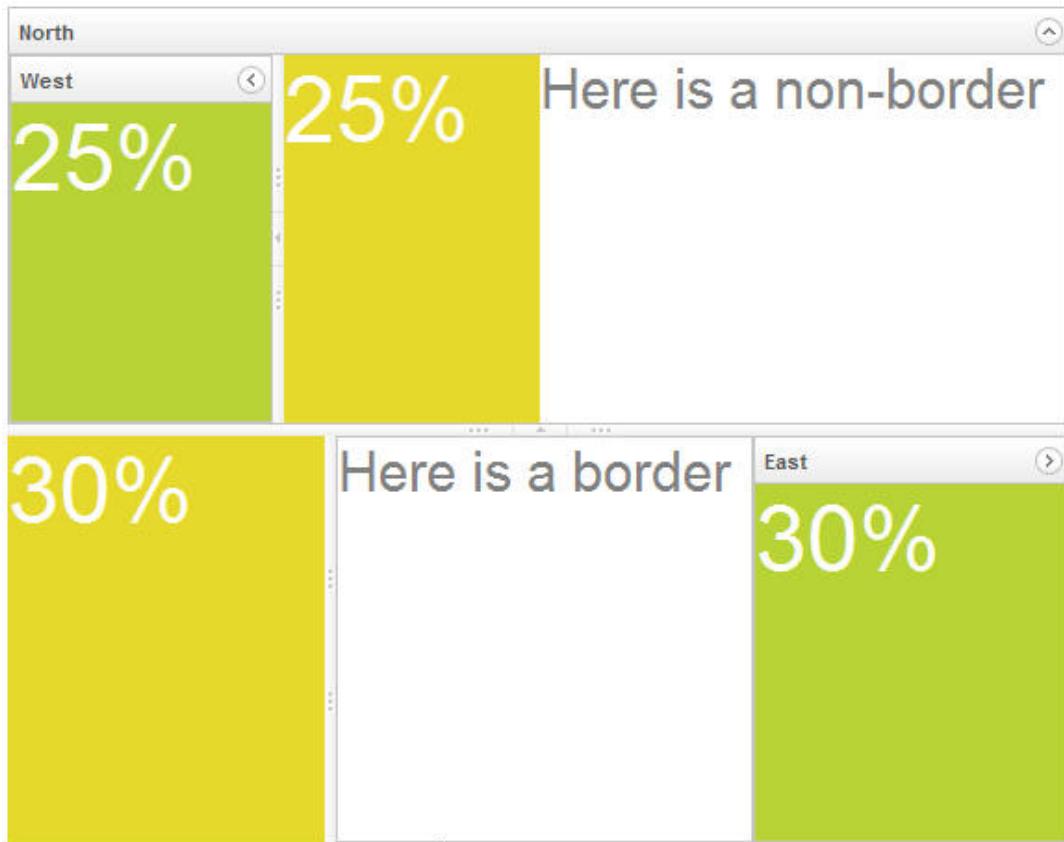
North

- Demonstration: Borderlayout [1]
- Java API: North [1]
- JavaScript API: North [2]
- Style Guide: North

Employment/Purpose

A north region of a border layout and only allows one component as its child.

Example



```
<borderlayout height="450px">
    <north title="North" maxsize="300" size="50%" splittable="true" collapsible="true">
        <borderlayout>
            <west title="West" size="25%" flex="true" maxsize="250" splittable="true" collapsible="true">
                <div style="background:#B8D335">
                    <label value="25%" style="color:white;font-size:50px">
                </div>
            </west>
        </borderlayout>
    </north>
    <center>
        <div>
            <h1>Hello World</h1>
        </div>
    </center>
    <east title="East" size="30%" flex="true" maxsize="300" splittable="true" collapsible="true">
        <div style="background:#B8D335">
            <label value="30%" style="color:white;font-size:50px">
        </div>
    </east>

```

```
<center border="none">
  <div style="background:#E6D92C" vflex="1">
    <label value="25%" style="color:white;font-size:50px"
  />
  </div>
</center>
<east size="50%" border="none">
  <label value="Here is a non-border" style="color:gray;font-size:30px" />
</east>
</borderlayout>
</north>
<center border="0">
  <borderlayout>
    <west maxsize="600" size="30%" border="0" splittable="true">
      <div style="background:#E6D92C" vflex="1">
        <label value="30%" style="color:white;font-size:50px"
      />
      </div>
    </west>
    <center>
      <label value="Here is a border" style="color:gray;font-size:30px" />
    </center>
    <east title="East" size="30%" collapsible="true">
      <div style="background:#B8D335" vflex="1">
        <label value="30%" style="color:white;font-size:50px"
      />
      </div>
    </east>
  </borderlayout>
</center>
</borderlayout>
```

How to Layout

For more details, please refer to Borderlayout.

Properties and Features

Caption

A layout region might have a caption, which is specified by declaring a child component called caption.

[ZK EE]

[Since 6.5.0]

```
<borderlayout>
    <north>
        <caption label="search" image="/img/live.gif">
            <combobox>
                <comboitem label="item 1" />
                <comboitem label="item 2" />
                <comboitem label="item 3" />
                <comboitem label="item 4" />
            </combobox>
        </caption>
        <div>
            Content
        </div>
    </north>
</borderlayout>
```

Closable

Whether users can open or close the region. Require `collapsible="true"`. Default: true.

Slidable

Whether users can slide (preview) the region when clicked on a collapsed region. Require `collapsible="true"`. Default: true.

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: LayoutRegion

Supported Children

* ALL

Use Cases

Borderlayout

Version History

Version	Date	Content
6.5.0	June 2012	ZK-969 ^[3] : The LayoutRegion component support caption component as it's title
8.5.2	May 2018	ZK-3329 ^[3] : Collapsible Borderlayout region in slide or open mode only

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/North.html#>

[2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/layout/North.html#>

South

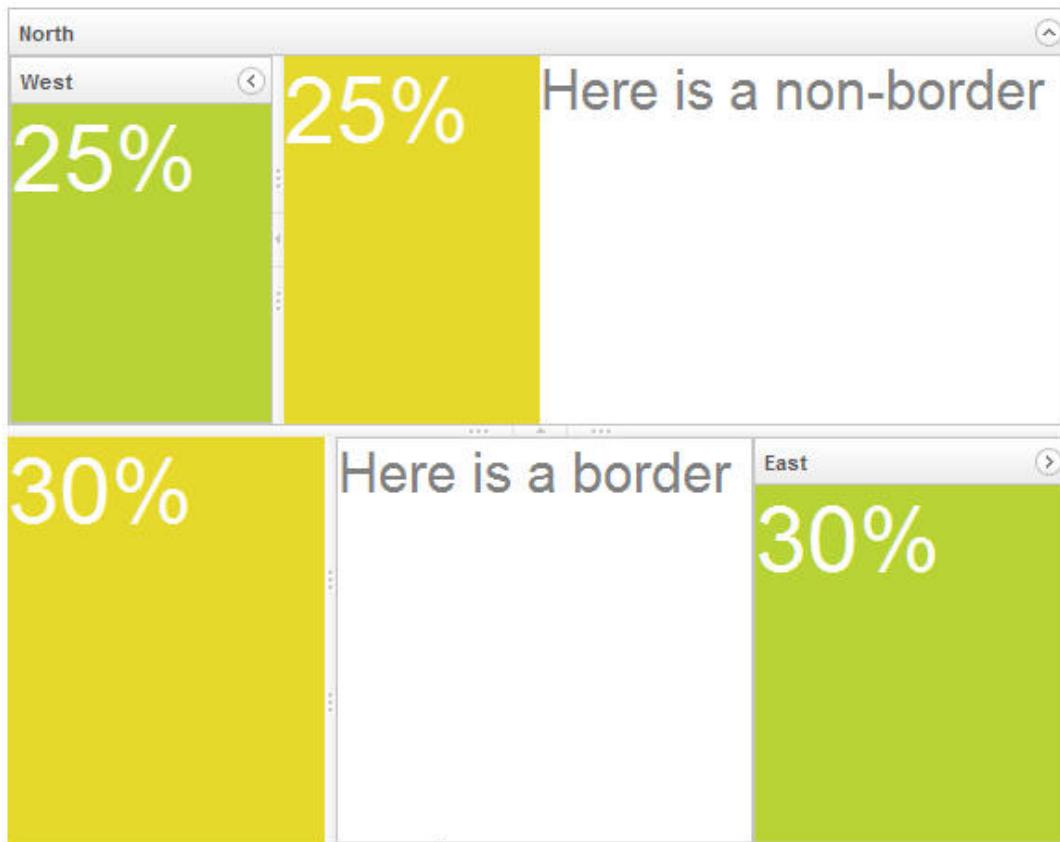
South

- Demonstration: Borderlayout ^[1]
- Java API: South ^[1]
- JavaScript API: South ^[2]
- Style Guide: South

Employment/Purpose

A south region of a border layout and only allows one component as its child.

Example



```
<borderlayout height="450px">

<north title="North" maxsize="300" size="50%" splittable="true" collapsible="true">
    <borderlayout>
        <west title="West" size="25%" flex="true" maxsize="250" splittable="true" collapsible="true">
            <div style="background:#B8D335">
                <label value="25%" style="color:white;font-size:50px">
            </div>
        </west>
        <center border="none">
            <div style="background:#E6D92C" vflex="1">
                <label value="25%" style="color:white;font-size:50px">
            </div>
        </center>
        <east size="50%" border="none">
            <label value="Here is a non-border" style="color:gray;font-size:30px" />
        </east>
    </borderlayout>
</north>
<center border="0">
```

```
<borderlayout>
    <west maxsize="600" size="30%" border="0" splittable="true">
        <div style="background:#E6D92C" vflex="1">
            <label value="30%" style="color:white;font-size:50px"
        />
        </div>
    </west>
    <center>
        <label value="Here is a border" style="color:gray;font-size:30px" />
    </center>
    <east title="East" size="30%" collapsible="true">
        <div style="background:#B8D335" vflex="1">
            <label value="30%" style="color:white;font-size:50px"
        />
        </div>
    </east>
</borderlayout>
</center>
</borderlayout>
```

Properties and Features

Caption

A layout region might have a caption, which is specified by declaring a child component called caption.

[ZK EE]
[Since 6.5.0]

```
<borderlayout>
    <south>
        <caption label="search" image="/img/live.gif">
            <combobox>
                <comboitem label="item 1" />
                <comboitem label="item 2" />
                <comboitem label="item 3" />
                <comboitem label="item 4" />
            </combobox>
        </caption>
        <div>
            Content
        </div>
    </south>
</borderlayout>
```

Closable

Whether users can open or close the region. Require `collapsible="true"`. Default: true.

[Since 8.5.2]

Slidable

Whether users can slide (preview) the region when clicked on a collapsed region. Require `collapsible="true"`. Default: true.

[Since 8.5.2]

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: LayoutRegion

How to Layout

For more details, please refer to Borderlayout.

Supported Children

*ALL

Use Cases

Borderlayout

Version History

Version	Date	Content
6.5.0	June 2012	ZK-969 [3]: The LayoutRegion component support caption component as it's title
8.5.2	May 2018	ZK-3329 [3]: Collapsible Borderlayout region in slide or open mode only

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/South.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/layout/South.html#>

West

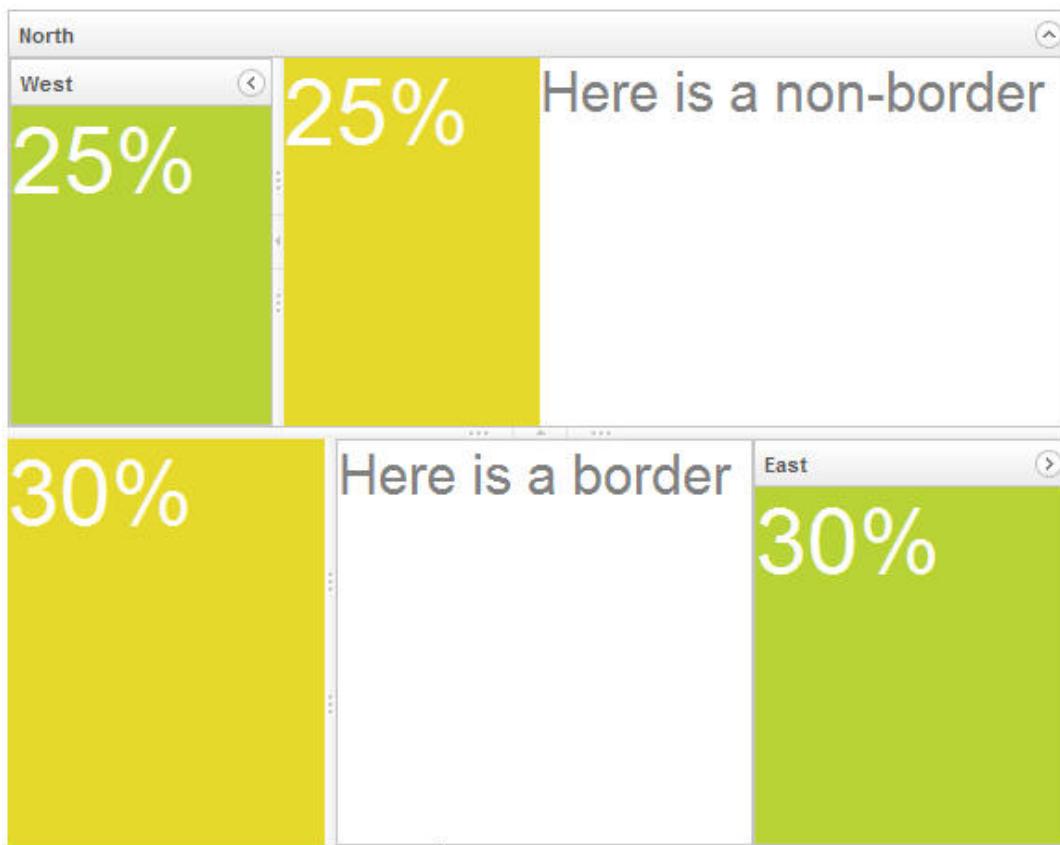
West

- Demonstration: Borderlayout [1]
- Java API: West [1]
- JavaScript API: West [2]
- Style Guide: West

Employment/Purpose

A west region of a border layout and only allows one component as its child.

Example



```
<borderlayout height="450px">
    <north title="North" maxsize="300" size="50%" splittable="true" collapsible="true">
        <borderlayout>
            <west title="West" size="25%" flex="true" maxsize="250" splittable="true" collapsible="true">
                <div style="background:#B8D335">
                    <label value="25%" style="color:white;font-size:50px">
                />
                </div>
            </west>
        </borderlayout>
    </north>
    <center>Here is a non-border</center>
    <center>Here is a border</center>
    <east title="East" maxsize="300" size="30%" splittable="true" collapsible="true">
        <div style="background:#B8D335">
            <label value="30%" style="color:white;font-size:50px">
        />
        </div>
    </east>

```

```
<center border="none">
  <div style="background:#E6D92C" vflex="1">
    <label value="25%" style="color:white;font-size:50px"
  />
  </div>
</center>
<east size="50%" border="none">
  <label value="Here is a non-border" style="color:gray;font-size:30px" />
</east>
</borderlayout>
</north>
<center border="0">
  <borderlayout>
    <west maxsize="600" size="30%" border="0" splittable="true">
      <div style="background:#E6D92C" vflex="1">
        <label value="30%" style="color:white;font-size:50px"
    />
      </div>
    </west>
    <center>
      <label value="Here is a border" style="color:gray;font-size:30px" />
    </center>
    <east title="East" size="30%" collapsible="true">
      <div style="background:#B8D335" vflex="1">
        <label value="30%" style="color:white;font-size:50px"
    />
      </div>
    </east>
  </borderlayout>
</center>
</borderlayout>
```

How to Layout

For more details, please refer to Borderlayout.

Properties and Features

Caption

A layout region might have a caption, which is specified by declaring a child component called caption.

[ZK EE]

[Since 6.5.0]

```
<borderlayout>
    <west>
        <caption label="search" image="/img/live.gif">
            <combobox>
                <comboitem label="item 1" />
                <comboitem label="item 2" />
                <comboitem label="item 3" />
                <comboitem label="item 4" />
            </combobox>
        </caption>
        <div>
            Content
        </div>
    </west>
</borderlayout>
```

Closable

Whether users can open or close the region. Require `collapsible="true"`. Default: true.

[Since 8.5.2]

Slidable

Whether users can slide (preview) the region when clicked on a collapsed region. Require `collapsible="true"`. Default: true.

[Since 8.5.2]

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: LayoutRegion

Supported Children

*ALL

Use Cases

Borderlayout

Version History

Version	Date	Content
6.5.0	June 2012	ZK-969 [3]: The LayoutRegion component support caption component as it's title
8.5.2	May 2018	ZK-3329 [3]: Collapsible Borderlayout region in slide or open mode only

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/West.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/layout/West.html#>

Box

Box

- Demonstration: Box ^[1]
- Java API: Box ^[2]
- JavaScript API: Box ^[3]
- Style Guide: Box

Employment/Purpose

The box model of XUL is used to divide a portion of the display into a series of boxes. Components inside a box will orient themselves horizontally or vertically. By combining a series of boxes and separators, you can control the layout of the visual presentation.

A box can lay out its children in one of two orientations, either horizontally or vertically. A horizontal box lines up its components horizontally and a vertical box orients its components vertically. You can think of a box as one row or one column from an HTML table.

A box is the generic component that can be used for horizontal and vertical layouts. However, it is generally more convenient by the use of hbox and vbox directly.

Notice that hbox and vbox are designed to provide more sophisticated layout, such as splitter, alignment and packing. If you need only the layout feature, it is suggest to use Hlayout and Vlayout instead, since the performance is much better (due to the use of HTML DIV instead of TABLE).

Example



```
<zk>
    <box orient="vertical">
        <button label="Button 1"/>
        <button label="Button 2"/>
    </box>
    <box orient="horizontal">
        <button label="Button 3"/>
        <button label="Button 4"/>
    </box>
</zk>
```

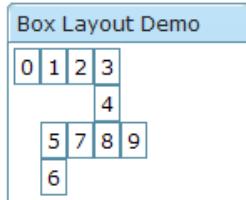
Properties

Spacing

You can control the spacing among children of the `box` control. For example, the following example puts `5em` at both the upper margin and the lower margin. Notice: the total space between two input fields is `10em`.

```
<vbox spacing="5em">
    <textbox/>
    <datebox/>
</vbox>
```

Another example illustrated an interesting layout by the use of zero spacing.



```
<window title="Box Layout Demo" border="normal">
    <hbox spacing="0">
        <window border="normal">0</window>
        <vbox spacing="0">
            <hbox spacing="0">
                <window border="normal">1</window>
                <window border="normal">2</window>
                <vbox spacing="0">
                    <window border="normal">3</window>
                    <window border="normal">4</window>
                </vbox>
            </hbox>
            <hbox spacing="0">
                <vbox spacing="0">
                    <window border="normal">5</window>
                    <window border="normal">6</window>
                </vbox>
                <window border="normal">7</window>
                <window border="normal">8</window>
                <window border="normal">9</window>
            </hbox>
        </vbox>
    </hbox>
</window>
```

Heights and Widths

(i) Notice: Deprecated. As of release 5.0.0, use Cell instead.

You can control the width for each cell inside a `hbox` with `widths` attribute as follows (don't specify on each cell):

```
<hbox width="100%" height="100px" widths="10%,20%,30%,40%" pack="stretch">
    <label value="10%"/>
    <label value="20%"/>
    <label value="30%"/>
    <label value="40%"/>
</hbox>
<vbox width="100%" height="500px" heights="10%,20%,30%,40%" pack="stretch">
    <label value="10%"/>
    <label value="20%"/>
    <label value="30%"/>
    <label value="40%"/>
</vbox>
```

The value is a comma-separated list of widths. If any value is missed, no width is generated for the corresponding cell and the real width is up to the browser.

Similarly, you can specify the height of each cell inside a `vbox` using the `heights` attribute. These two properties are the same since the orientation of a box can be horizontal or vertical depending on the `orient` property.

Align and Pack

pack / align		stretch	start	center	end
	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
start	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
center	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
end	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3

```
<zk xmlns:n="http://www.zkoss.org/2005/zk/native">

<zscript><! [CDATA[
    Map map = new LinkedHashMap();
    String[] packs = new String[]{"", "start", "center",
"end"};
    String[] aligns = new String[]{"", "stretch", "start",
"center", "end"};

    for (int i = 0; i < aligns.length; i++) {
        String align = aligns[i];
        List list = new ArrayList();
        for (int j = 0; j < packs.length; j++) {
            list.add(packs[j]);
        }
        map.put(align, list);
    }

]]></zscript>


<panel border="normal" height="150px" width="100px"
forEach=' "pack / align", "", "start", "center",
"end" !>
<panelchildren>
<n:h3>${each}</n:h3>
</panelchildren>
</panel>
</tablechildren>

```

[Since 5.0.0]

Cell Component

In ZK5, we have introduced a new component named Cell which can be embedded into a Grid or Box (Hbox and Vbox) to fully control the layout and the style. You can now use the rowspan or the colspan property to layout your Grid, for example a content cell can now cross over multiple rows. The code below demonstrates how to do this:

```
<box>
  <cell sclass="years">
    ...
  </cell>
</box>
```

[Since 5.0.0]

Limitation

Box component is consisted by Table element. Therefore, when put Input element like Textbox, Combobox inside Box component, specify width and height to Box component will be ignored when browser try to render table element.

For example,

```
<hbox height="200px" width="200px" style="border: 1px solid red">
  <textbox hflex="1" value="1" />
  <textbox hflex="1" value="1" />
</hbox>
```

You will see the Box width exceed 200px. Also check the sample ^[4] with pure HTML in jsfiddle.

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Molds

Available molds of a component are defined in lang.xml embedded in zul.jar.

Name	Snapshot
horizontal	
vertical	

Supported Children

*ALL

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content
5.0.4	August, 2010	Add a sizedByContent method for splitter to resize smoothly

References

- [1] <http://www.zkoss.org/zkdemo/layout/box>
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Box.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/box/Box.html#>
- [4] <http://jsfiddle.net/A5g9q/>

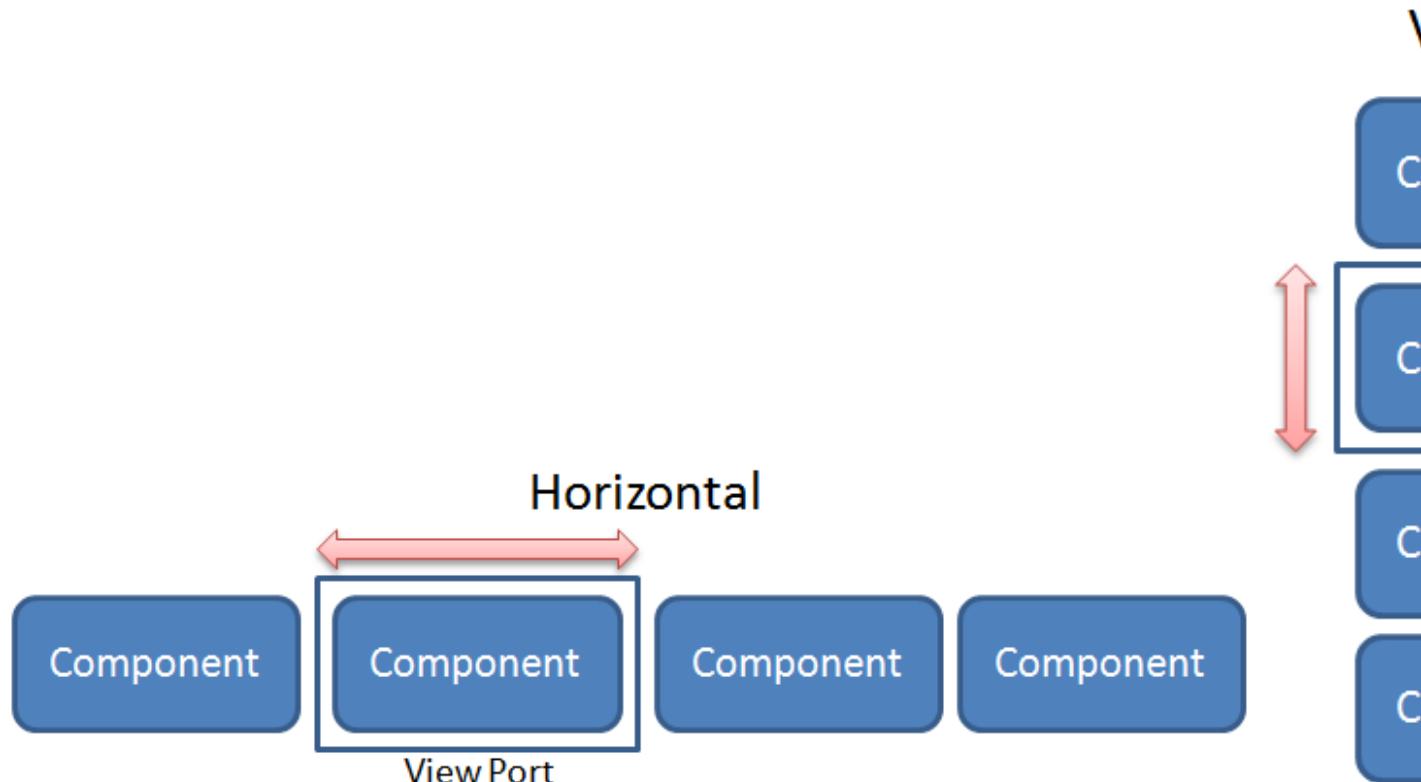
Cardlayout

Cardlayout

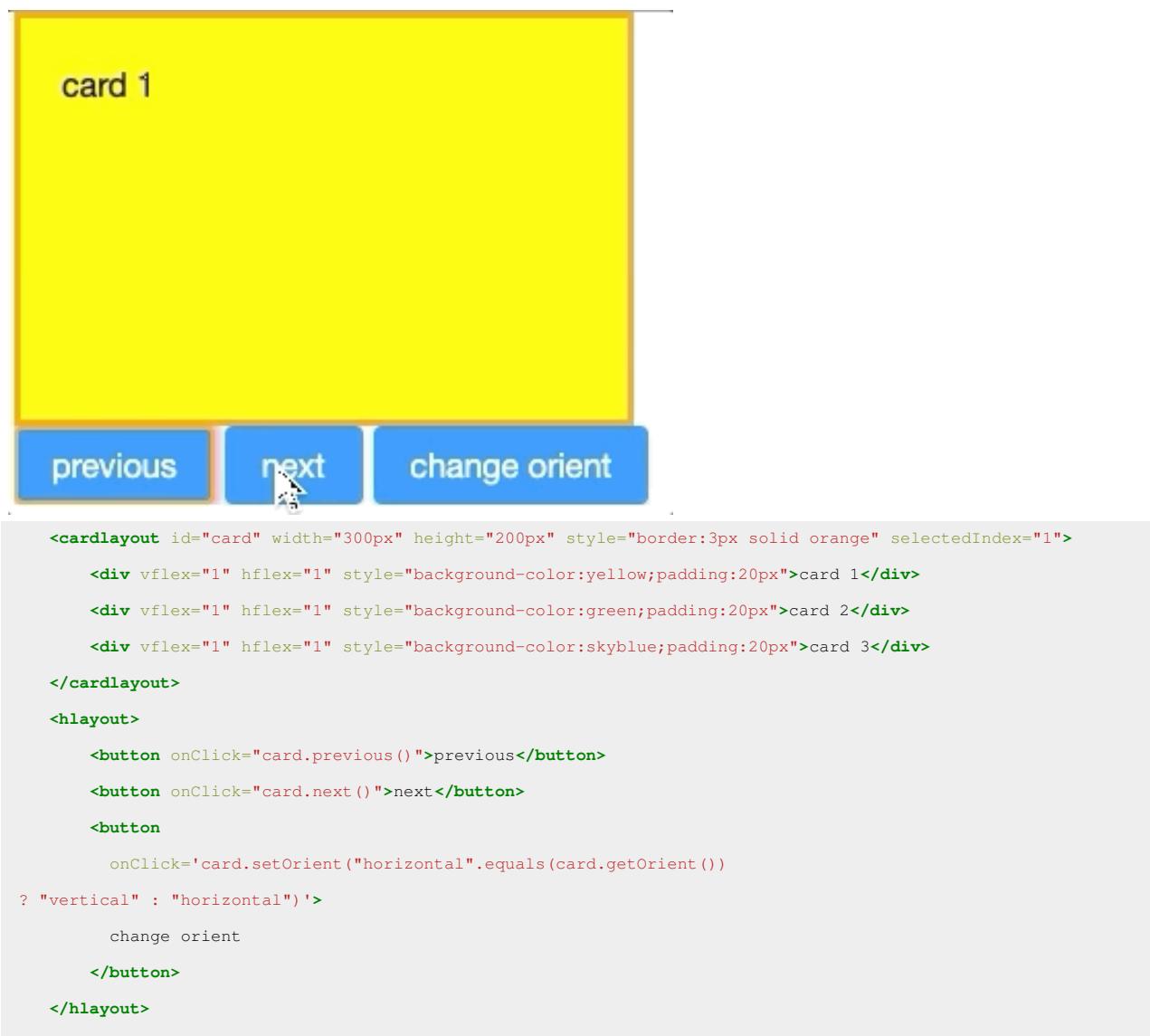
- Demonstration: N/A
- Java API: Cardlayout [1]
- JavaScript API: Cardlayout [2]
- Style Guide: N/A
- Available for ZK:
- CE PE EE

Employment/Purpose

Cardlayout is a layout that allows end-users to change component like changing cards. The `selectedIndex` will decide which component will be shown in the view port. When the value of `selectedIndex` changes or when `next()` or `previous()` is called, transition of components through animation will occur whereas the `orient` attribute decides whether the direction of the animation is horizontal or vertical.



Example



```
<cardlayout id="card" width="300px" height="200px" style="border:3px solid orange" selectedIndex="1">
    <div vflex="1" hflex="1" style="background-color:yellow;padding:20px">card 1</div>
    <div vflex="1" hflex="1" style="background-color:green;padding:20px">card 2</div>
    <div vflex="1" hflex="1" style="background-color:skyblue;padding:20px">card 3</div>
</cardlayout>
<hlayout>
    <button onClick="card.previous()">previous</button>
    <button onClick="card.next()">next</button>
    <button
        onClick='card.setOrient("horizontal".equals(card.getOrient()))
? "vertical" : "horizontal")'>
        change orient
    </button>
</hlayout>
```

Size Issue

If `Cardlayout` of `hflex` is set as `"min"`, its width will be decided by the selected component's size when initializing. On the other hand, if the child component of `Cardlayout` sets `hflex="1"`, its width will equal to `Cardlayout`'s width.

Swipe Distance Issue

On tablet, think for user experience `Cardlayout` will change component if swipe distance bigger than one-third of its width/height. If `Cardlayout`'s width/height is smaller than 90px, the minimum trigger distance will be 30px. Another case is `Image`. If `Cardlayout`'s child component is `Image`, it will use default swipe distance trigger setting.

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

*ALL

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content
6.5.0	August, 2012	Cardlayout [1] was introduced.

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Cardlayout.html#>

[2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/layout/Cardlayout.html#>

Columnlayout

Columnlayout

- Demonstration: Columnlayout ^[1]
- Java API: Columnlayout ^[2]
- JavaScript API: Columnlayout ^[3]
- Style Guide: Columnlayout
- Available for ZK:
- CE PE EE

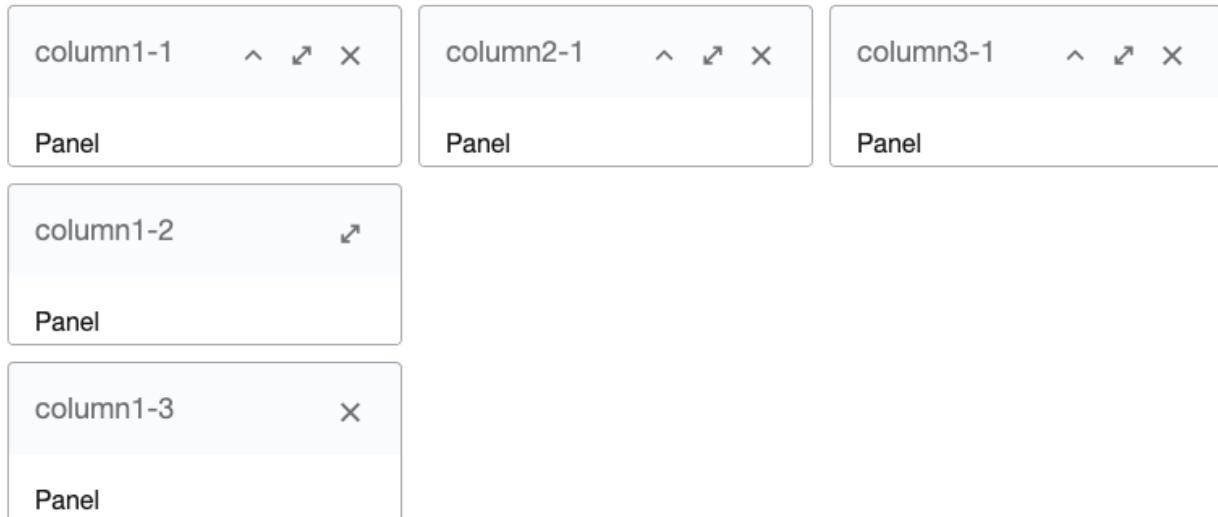
Employment/Purpose

A columnlayout is a layout which can have multiple columns while each column may have any number of panels placed vertically with different heights. When using Columnlayout, you have to assign width (either percent or pixel) on every Columnchildren, otherwise the result may depend on the browser and may not be as expected.

[since 6.0.0]

Each column may have any number of any type of components.

Example



```
<columnlayout>
    <columnchildren width="33%" style="padding: 5px">
        <panel height="100px" title="column1-1" closable="true" collapsible="true"
            border="normal" maximizable="true"
            style="margin-bottom:10px">
            <panelchildren>Panel</panelchildren>
        </panel>
        <panel height="100px" framable="true" title="column1-2"
            border="normal" maximizable="true"
            style="margin-bottom:10px">
```

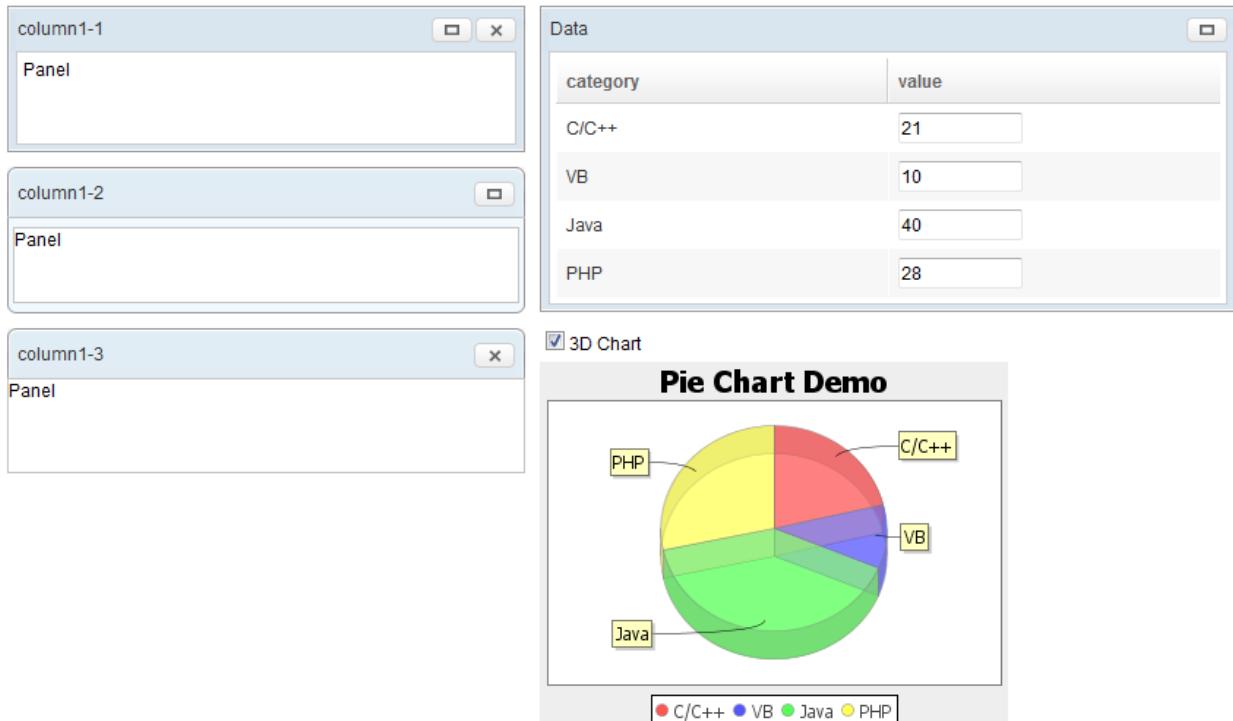
```

<panelchildren>Panel</panelchildren>
</panel>
<panel height="100px" title="column1-3" border="normal"
closable="true">
<panelchildren>Panel</panelchildren>
</panel>
</columnchildren>
<columnchildren width="33%" style="padding: 5px">
<panel height="100px" title="column2-1" closable="true" collapsible="true"
border="normal" maximizable="true"
style="margin-bottom:10px">
<panelchildren>Panel</panelchildren>
</panel>
</columnchildren>
<columnchildren width="33%" style="padding: 5px">
<panel height="100px" title="column3-1" closable="true" collapsible="true"
border="normal" maximizable="true"
style="margin-bottom:10px">
<panelchildren>Panel</panelchildren>
</panel>
</columnchildren>
</columnlayout>

```

[since 6.0.0]

Each column may have any number of any type of components.



```

<columnlayout>
<columnchildren width="30%" style="padding: 5px">
<window height="100px" title="column1-1" closable="true">

```

```
        border="normal" maximizable="true"
style="margin-bottom:10px">
    Panel
</window>
<panel height="100px" framable="true" title="column1-2"
border="normal" maximizable="true"
style="margin-bottom:10px">
    <panelchildren>Panel</panelchildren>
</panel>
<panel height="100px" title="column1-3" border="normal"
closable="true">
    <panelchildren>Panel</panelchildren>
</panel>
</columnchildren>
<columnchildren width="40%" style="padding: 5px">
    <window id="dataWin" title="Data" maximizable="true" border="normal"
style="margin-bottom:10px">
        <grid fixedLayout="true" style="border:0px"
height="100%">
            <columns>
                <column label="category" />
                <column label="value" />
            </columns>
            <rows>
                <row>
                    <label id="c0" value="C/C++" />
                    <decimalbox id="v0"
value="21." constraint="no
empty" onChange="update(0)" />
                </row>
                <row>
                    <label id="c1" value="VB" />
                    <decimalbox id="v1"
value="10." constraint="no
empty" onChange="update(1)" />
                </row>
                <row>
                    <label id="c2" value="Java" />
                    <decimalbox id="v2"
value="40." constraint="no
empty" onChange="update(2)" />
                </row>
                <row>
                    <label id="c3" value="PHP" />
                    <decimalbox id="v3"
value="28." constraint="no
empty" onChange="update(3)" />
                </row>
            </rows>
        </grid>
    </window>
</columnchildren>
```

```
</row>
</rows>
</grid>
</window>
<vbox>
    <checkbox label="3D Chart" checked="true"
        onCheck="mychart.setThreeD(self.isChecked())" />
    <chart id="mychart" title="Pie Chart Demo"
        width="320px" type="pie" threeD="true"
        fgAlpha="128">
        <attribute name="onClick"><![CDATA[
            String areaid = event.getArea();
            if (areaid != null) {
                Area area = self.getFellow(areaid);
                alert("") +
                area.getAttribute("entity") + ":" + area.getTooltiptext());
            }
        ]]></attribute>
        <zscript><![CDATA[
            void update(int rowIndex) {
                Window dataWin =
self.getParent().getParent().getFellow("dataWin");
                Label lb = (Label)
dataWin.getFellow("c" + rowIndex);
                Decimalbox db = (Decimalbox)
dataWin.getFellow("v" + rowIndex);
                model.setValue(lb.value, new
Double(db.getValue().doubleValue()));
            }
            PieModel model = new SimplePieModel();
            for (int j = 0; j < 4; ++j) {
                update(j);
            }
            mychart.setModel(model);
        ]]></zscript>
    </chart>
</vbox>
</columnchildren>
</columnlayout>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

Columnchildren

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

- [1] http://www.zkoss.org/zkdemo/layout/column_layout
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkex/zul/Columnlayout.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zkex/layout/Columnlayout.html#>

Columnchildren

Columnchildren

- Demonstration: Columnlayout ^[1]
- Java API: Columnchildren ^[1]
- JavaScript API: Columnchildren ^[2]
- Style Guide: Columnlayout
- Available for ZK:
- CE PE EE

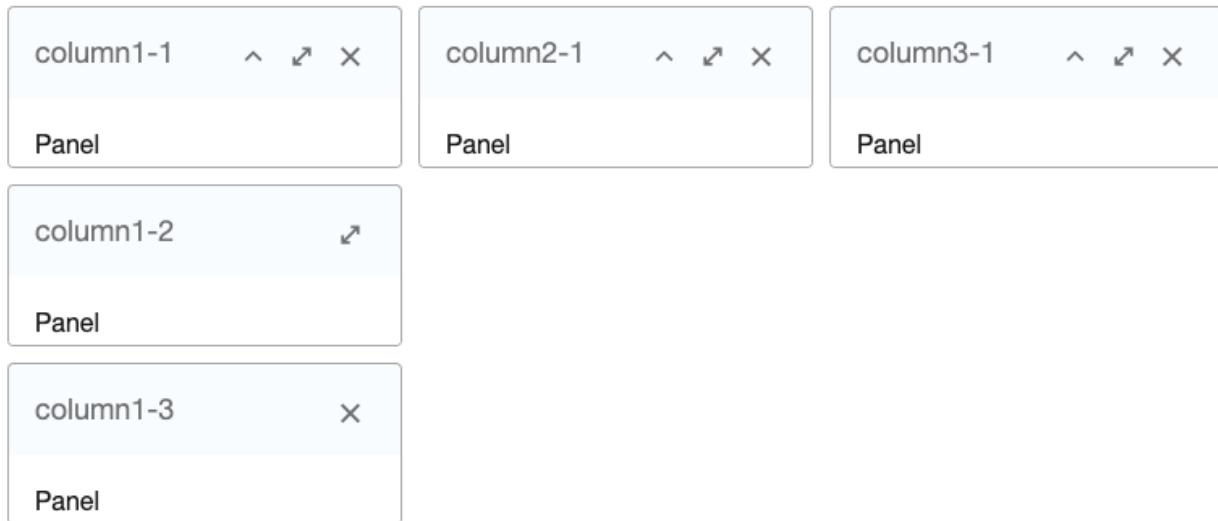
Employment/Purpose

Columnchildren can only allow Panel as its child.

[since 6.0.0]

- Columnchildren can allow any Component as its child.

Example



```
<columnlayout>
    <columnchildren width="30%" style="padding: 5px">
        <panel height="100px" style="margin-bottom:10px" title="column1-1" border="normal" maximizable="true" collapsible="true">
            <panelchildren>Panel</panelchildren>
        </panel>
        <panel height="100px" framable="true" title="column1-2" border="normal" maximizable="true" style="margin-bottom:10px">
            <panelchildren>Panel</panelchildren>
        </panel>
        <panel height="100px" title="column1-3" border="normal" closable="true">
```

```
<panelchildren>Panel</panelchildren>
</panel>
</columnchildren>
<columnchildren width="40%" style="padding: 10px">
<panel title="Data" maximizable="true" border="normal"
style="margin-bottom:10px">
<panelchildren>
<grid fixedLayout="true" style="border:0px" height="100%">
<columns>
<column label="category" />
<column label="value" />
</columns>
<rows>
<row>
<label id="c0" value="C/C++" />
<decimalbox id="v0" value="21." constraint="no empty"
onChange="update(0)" />
</row>
<row>
<label id="c1" value="VB" />
<decimalbox id="v1" value="10." constraint="no empty"
onChange="update(1)" />
</row>
<row>
<label id="c2" value="Java" />
<decimalbox id="v2" value="40." constraint="no empty"
onChange="update(2)" />
</row>
<row>
<label id="c3" value="PHP" />
<decimalbox id="v3" value="28." constraint="no empty"
onChange="update(3)" />
</row>
</rows>
</grid>
</panelchildren>
</panel>
<panel border="normal">
<panelchildren>
<checkbox label="3D Chart" checked="true"
onCheck="mychart.setThreeD(self.isChecked())" />
<chart id="mychart" title="Pie Chart Demo" width="320px" type="pie" threeD="true" fgAlpha="128">
<attribute name="onClick">
String areaid =
event.getArea();
if(araid!= null) {
Area area =

```

```
self.getFellow(areaid);

alert(""+area.getAttribute("entity")+": "+area.getTooltiptext());
}

</attribute>

<zscript> void update(int rowIndex) { Label lb = (Label)
    self.getFellow("c"+rowIndex); Decimalbox db =
    (Decimalbox) self.getFellow("v"+rowIndex);

model.setValue(lb.value,
    new Double(db.getValue().doubleValue())); }

PieModel model = new
SimplePieModel(); for(int j=0; j < 4; ++j) {
update(j); }

mychart.setModel(model);
</zscript>

</chart>

</panelchildren>
</panel>
</columnchildren>
</columnlayout>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

* Panel

[since 6.0.0]

*Any

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkex/zul/Columnchildren.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkex/layout/Columnchildren.html#>

GoldenLayout

GoldenLayout

- Available for ZK:
-   
- Java API: GoldenLayout ^[1]
- JavaScript API: GoldenLayout ^[2]

Browser Support

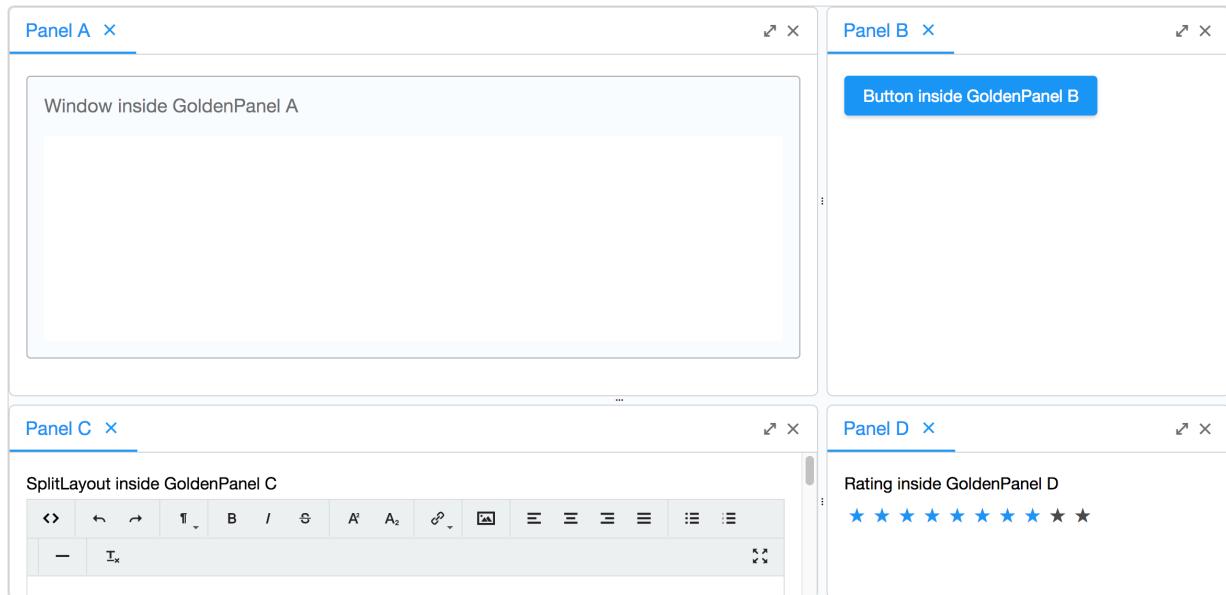
- Due to the limitation of the 3rd party library used in this component, GoldenLayout is not supported in mobile devices.

Employment/Purpose

GoldenLayout is a layout container which layouts panels as docker type and is used to represent GoldenLayout ^[3]. GoldenLayout is the parent component, and its children component can only be GoldenPanel.

A GoldenLayout could be nested to another GoldenLayout (actually, almost all kinds of components) to form a complicated layout.

Example



```
<goldenlayout vflex="1" hflex="1">
  <attribute name="areas">
    A A B
    A A B
    C C D
  </attribute>
  <goldenpanel area="A" title="Panel A">
    <window vflex="1" title="Window inside GoldenPanel A" border="normal"/>
  </goldenpanel>
  <goldenpanel area="B" title="Panel B">
    <button label="Button inside GoldenPanel B"/>
  </goldenpanel>
  <goldenpanel area="C" title="Panel C">
    <vlayout>
      SplitLayout inside GoldenPanel C
      <splitlayout hflex="1" height="500px">
        <tbeditor vflex="1"/>
        <window border="normal" title="Window" vflex="1"/>
      </splitlayout>
    </vlayout>
  </goldenpanel>
  <goldenpanel area="D" title="Panel D">
    <vlayout>
      Rating inside GoldenPanel D
      <rating max="10" rating="8"/>
    </vlayout>
  </goldenpanel>
</goldenlayout>
```

Properties and Features

The goldenLayout is layouted as docker type, and is usually construct with a tree structure, which is hard to layout in zul. ZK change the structure for initial rendering to a one layer design. All the GoldenPanels are in one layer inside GoldenLayout, and the panels layout pattern will be specified by the `areas` attribute of GoldenLayout and `area` attribute of GoldenPanel. The order added to the GoldenLayout only represents the stacking order if they are stacked together in the same area.

Areas

The `areas` attribute is a sugar for initializing the layout. In the example code above, we have the areas in line 2 specified as

A A B

A A B

C C D

The GoldenPanels which `area` attribute is specified as A, will be positioned at the A position of the `areas` attribute.

Note that, A A B will result as `hflex` A is 2 and `hflex` B is 1. Same with `vflex`. But if `hflex` or `vflex` attribute is specified on the GoldenPanel which area is A, it overrides the flex size calculated by the GoldenLayout's `areas` attribute.

Orient

The orient indicates the initial splitting orientation if you layout it by the area attribute. Supported value: (default) "vertical" or "horizontal".

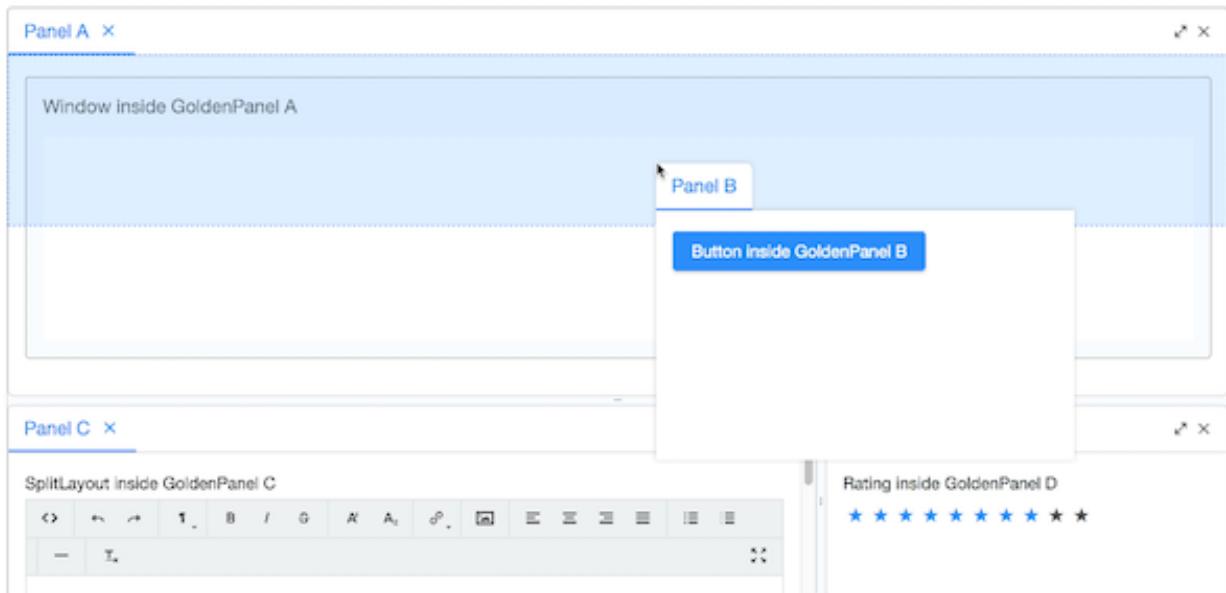
From the example above, it's possible to divide in two ways.

<code>areas</code>	<code>vertical</code>	<code>horizontal</code>																											
<table border="1"> <tr> <td>A</td><td>A</td><td>B</td></tr> <tr> <td>A</td><td>A</td><td>B</td></tr> <tr> <td>C</td><td>C</td><td>D</td></tr> </table>	A	A	B	A	A	B	C	C	D	<table border="1"> <tr> <td>A</td><td>A</td><td>B</td></tr> <tr> <td>A</td><td>A</td><td>B</td></tr> <tr> <td>C</td><td>C</td><td>D</td></tr> </table>	A	A	B	A	A	B	C	C	D	<table border="1"> <tr> <td>A</td><td>A</td><td>B</td></tr> <tr> <td>A</td><td>A</td><td>B</td></tr> <tr> <td>C</td><td>C</td><td>D</td></tr> </table>	A	A	B	A	A	B	C	C	D
A	A	B																											
A	A	B																											
C	C	D																											
A	A	B																											
A	A	B																											
C	C	D																											
A	A	B																											
A	A	B																											
C	C	D																											

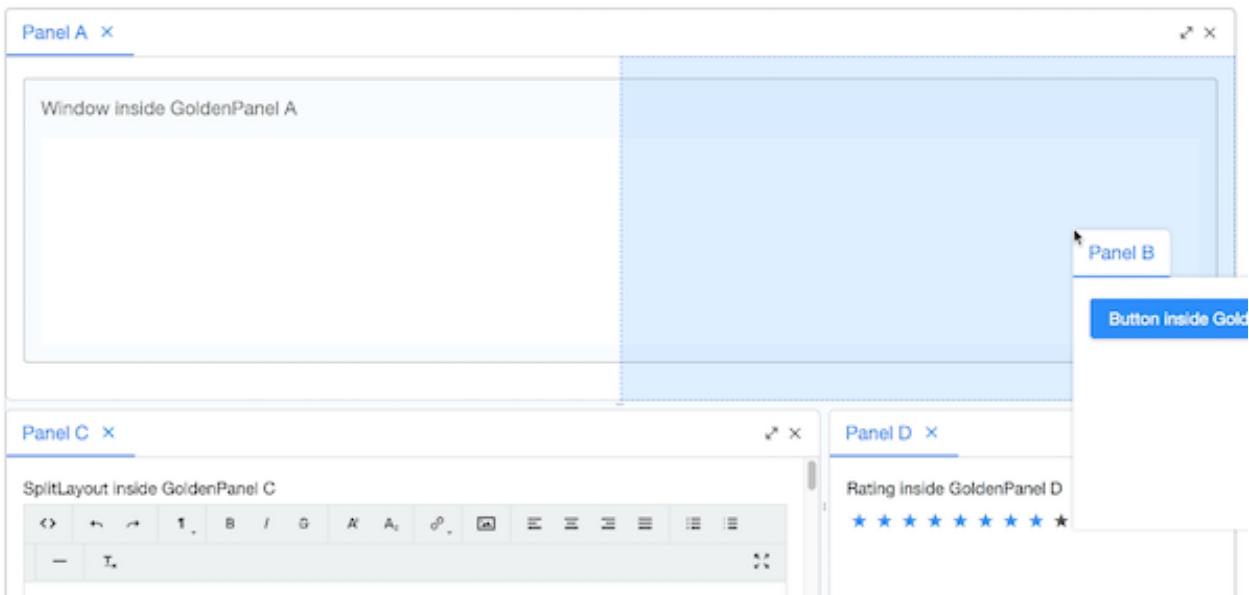
Adding GoldenPanels

By user's drag drop, the GoldenPanels are usually dropped to a region like north, east, south, west and stack of a GoldenPanel. So we provide some GoldenPanel adding APIs to simulate these actions. Please refer to the addPanel APIs like `GoldenPanel addPanel(GoldenPanel, GoldenPanel, String)` [4] in `GoldenLayout` [1].

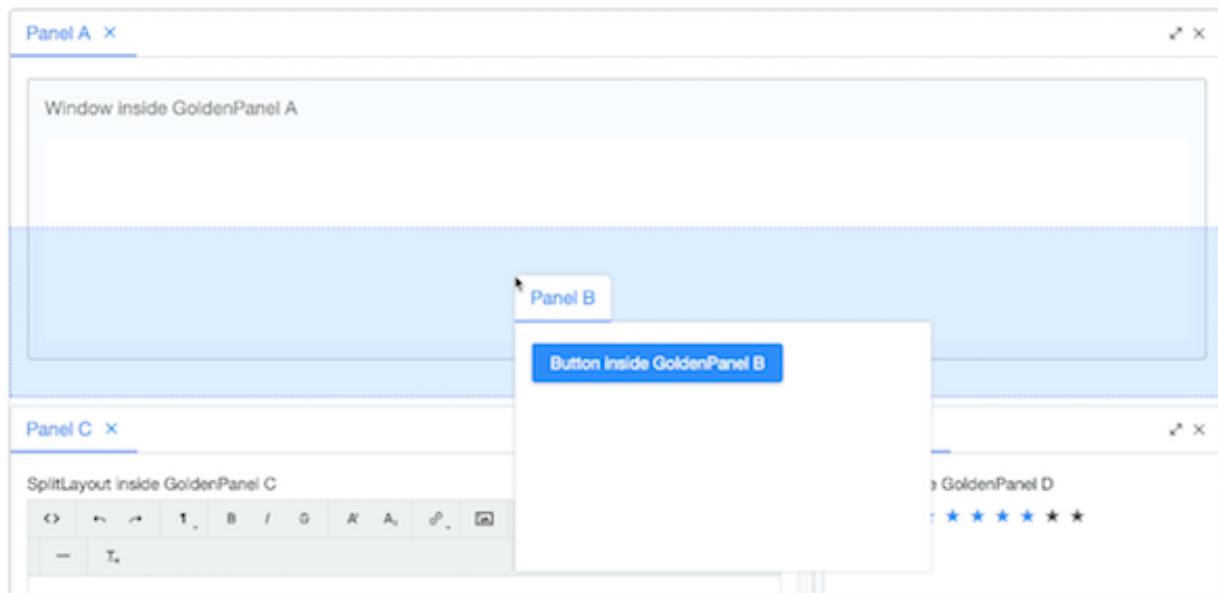
north



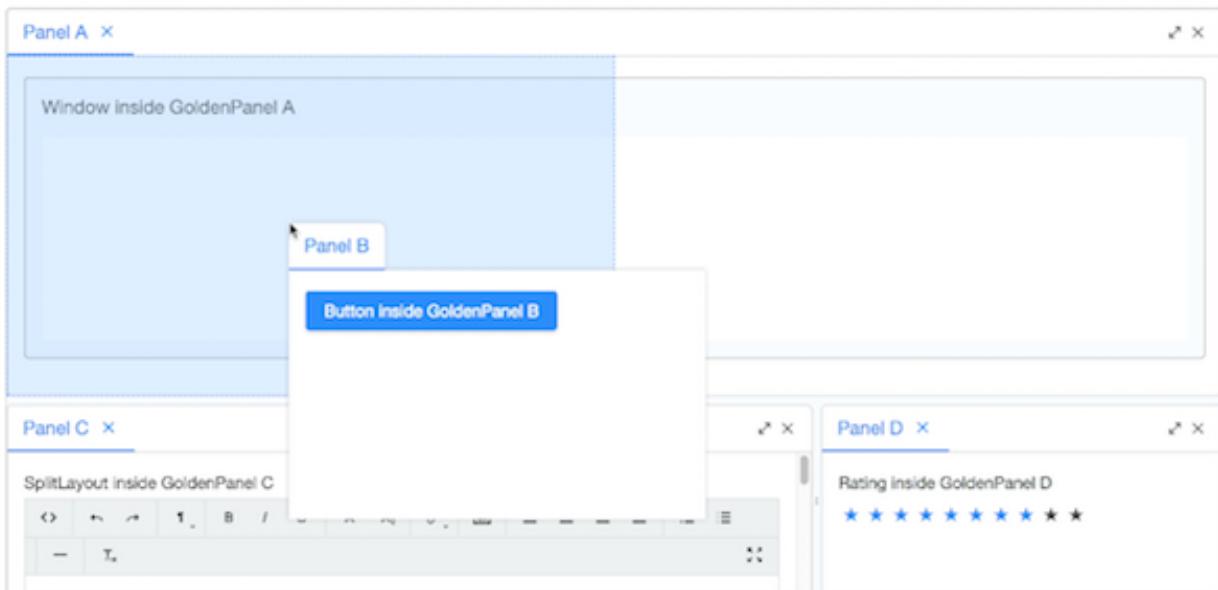
east



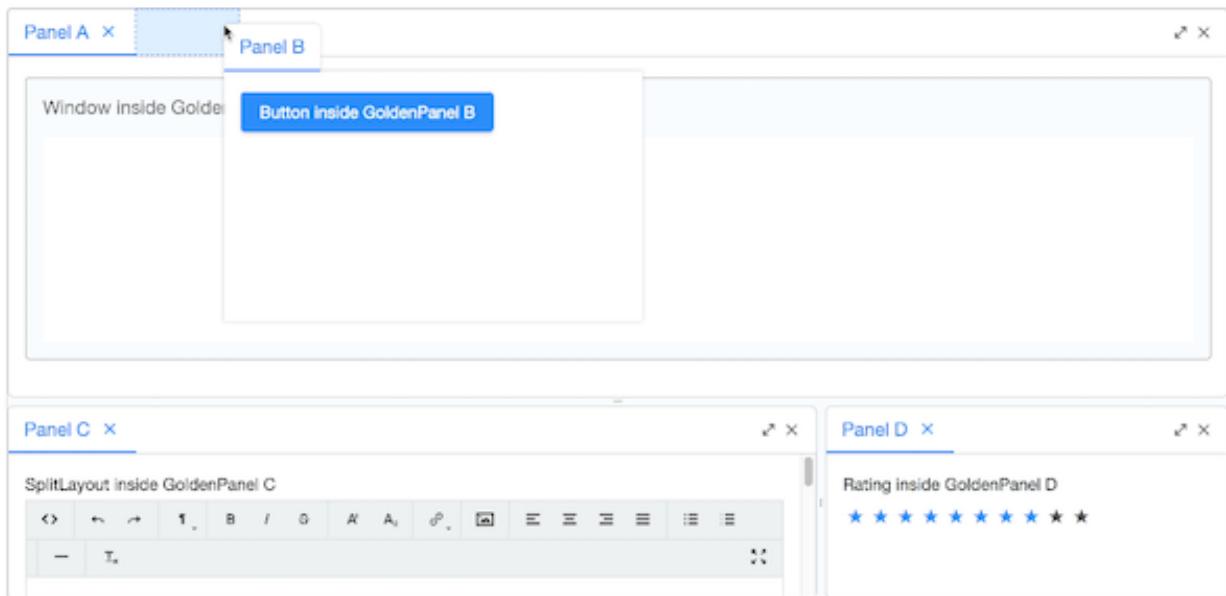
south



west



stack



Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

* GoldenPanel

Version History

Version	Date	Content

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/GoldenLayout.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/layout/GoldenLayout.html#>
- [3] <http://golden-layout.com/>
- [4] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/GoldenLayout.html#addPanel\(GoldenPanel,](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/GoldenLayout.html#addPanel(GoldenPanel,)

GoldenPanel

GoldenPanel

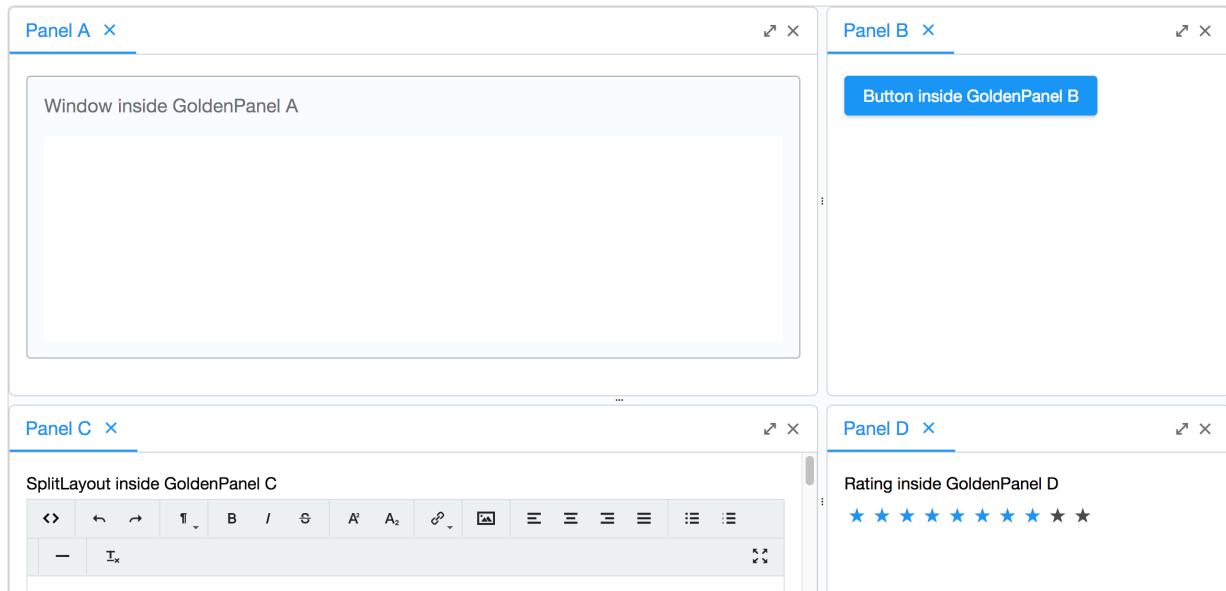
- Java API: GoldenPanel ^[1]
- JavaScript API: GoldenPanel ^[2]
- Available for ZK:
- CE PE EE

[since 8.6.0]

Employment/Purpose

GoldenPanel is the only child type of GoldenLayout. It allows us to rearrange them by dragging the tab.

Example



```
<goldenlayout vflex="1" hflex="1">
    <attribute name="areas">
        A A B
        A A B
        C C D
    </attribute>
    <goldenpanel area="A" title="Panel A">
        <window vflex="1" title="Window inside GoldenPanel A" border="normal"/>
    </goldenpanel>
    <goldenpanel area="B" title="Panel B">
        <button label="Button inside GoldenPanel B"/>
    </goldenpanel>
    <goldenpanel area="C" title="Panel C">
        <vlayout>
```

```
SplitLayout inside GoldenPanel C
<splitlayout hflex="1" height="500px">
    <tbeditor vflex="1"/>
    <window border="normal" title="Window" vflex="1"/>
</splitlayout>
</vlayout>
</goldenpanel>
<goldenpanel area="D" title="Panel D">
    <vlayout>
        Rating inside GoldenPanel D
        <rating max="10" rating="8"/>
    </vlayout>
</goldenpanel>
</goldenlayout>
```

Properties and Features

Area

The `area` attribute is a sugar for initializing the layout. The GoldenPanels with the same `area` will be stacked together with the added order. And will be placed at the position of the area in the `areas` attribute of it's parent `GoldenLayout`.

Title

The `title` attribute is the text on the tab of the `GoldenPanel`. If not specified, tab will still be rendered but with default title `Panel`.

Draggable

If false is specified, This `GoldenPanel` will not be able to dragged.

Droppable

If false is specified, No other `GoldenPanel` is allowed to be dropped on this `GoldenPanel`.

Closable

If false is specified, This `GoldenPanel` will not be able to be close by user action. And also the close icon will not show.

Supported Events

Name	Event Type
onActive	Event: Event ^[7] Denotes user has activated this GoldenPanel.
onPanelDrop	Event: Event ^[7] Denotes user has dropped this GoldenPanel.
onFlexSize	Event: Event ^[7] Denotes user has resized this GoldenPanel.
onClose	Event: Event ^[7] Denotes user has closed this GoldenPanel.
onMaximize	Event: Event ^[7] Denotes user has maximized this GoldenPanel.
onMinimize	Event: Event ^[7] Denotes user has minimized this GoldenPanel.

- Inherited Supported Events: XulElement

Supported Children

* All

Version History

Version	Date	Content

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/GoldenPanel.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/layout/GoldenPanel.html#>

Hbox

Hbox

- Demonstration: Hbox ^[1]
- Java API: Hbox ^[1]
- JavaScript API: Box ^[3]
- Style Guide: Hbox

Employment/Purpose

The hbox component is used to layout its child components horizontally and control those children's horizontal and vertical position..

Comparing to Hlayout

Notice that hbox and vbox are designed to provide more sophisticated layout, such as splitter, alignment and packing. ZK renders these 2 components with HTML table. If you only need the components for layout, we suggest using Hlayout and Vlayout instead, since the performance is much better (due to the use of HTML DIV instead of TABLE).

Example



```
<zk>
    <vbox>
        <button label="Button 1" />
        <button label="Button 2" />
    </vbox>
    <hbox>
        <button label="Button 3" />
        <button label="Button 4" />
    </hbox>
</zk>
```

Properties

- **Inherited Properties:** Box

Align and Pack

pack / align		start	center	end
stretch	1 2 3	1 2 3	1 2 3	1 2 3
start	1 2 3	1 2 3	1 2 3	1 2 3
center	1 2 3	1 2 3	1 2 3	1 2 3
end	1 2 3	1 2 3	1 2 3	1 2 3

```

<zk xmlns:n="native">
  <style content=".box {} class='wikitable' | width='100%'>
    <custom-attributes>
      packs="${['', 'start', 'center', 'end']}"
      aligns="${['', 'stretch', 'start', 'center', 'end']}"
    </custom-attributes>
  </style>

  <vlayout>
    <hlayout height="70px" width="900px">
      <div hflex="1" vflex="1" sclass="box">
        <n:h3>pack / align</n:h3>
      </div>
      <div forEach="${packs}" hflex="1" vflex="1" sclass="box">
        <n:h3>${each}</n:h3>
      </div>
    </hlayout>
    <hlayout forEach="${aligns}" height="100px" width="900px">
      <custom-attributes align="${each}" />
      <div hflex="1" vflex="1" sclass="box">
        <n:h3>${align}</n:h3>
      </div>
    </hlayout>
  </vlayout>
</zk>

```

```
<hbox forEach="${packs}" align="${align}" pack="${each}" hflex="1" vflex="1" sclass="box">
    <button label="1" />
    <button label="2" />
    <button label="3" />
</hbox>
</hlayout>
</vlayout>
</zk>
```

[Since 5.0.0]

Cell Component

In ZK5, we have introduced a new component named Cell which can be embedded into a Grid or Box (Hbox and Vbox) to fully control the layout and the style. You can now use the rowspan or the colspan property to layout your Grid, for example a content cell can now cross over multiple rows. The code below demonstrates how to do this:

```
<hbox>
    <cell sclass="years">
        ...
    </cell>
</hbox>
```

[Since 5.0.0]

Limitation

Box component is consisted by Table element. Therefore, when put Input element like Textbox, Combobox inside Box component, specify width and height to Box component will be ignored when browser try to render table element.

For example,

```
<hbox height="200px" width="200px" style="border: 1px solid red">
    <textbox hflex="1" value="1" />
    <textbox hflex="1" value="1" />
</hbox>
```

You will see the Box width exceed 200px. Also check the sample ^[4] with pure HTML in jsfiddle.

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: Box

Supported Children

*ALL

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Hbox.html#>

Hlayout

Hlayout

- Demonstration: N/A
- Java API: Hlayout ^[1]
- JavaScript API: Hlayout ^[2]
- Style Guide: Hlayout

Employment/Purpose

The hlayout component is a simple horizontal oriented layout. It layouts its child components horizontally in a row.

Notice that hlayout and vlayout do **not** support a splitter inside. If you need it, please use Hbox and Vbox instead.

Example



```
<zk>
    <vlayout>
        <button label="Button 1"/>
        <button label="Button 2"/>
    </vlayout>
    <hlayout>
        <button label="Button 3"/>
        <button label="Button 4"/>
    </hlayout>
</zk>
```

Spacing

The default spacing between two child components is 0.3em. You are allowed to modify it if you like:

```
<vlayout spacing="0">
    <textbox/>
    <button label="Click Me"/>
</vlayout>
```

Vertical Alignment

[since 5.0.5]

By default, the vertical alignment is *middle* (aka., *center*). You can change it to *top* by specifying `sclass="z-valign-top"`, and to *bottom* by `sclass="z-valign-bottom"`. For example,

```
<vlayout>
    <hlayout>
        center: <textbox/>
    </hlayout>
    <hlayout sclass="z-valign-top">
        top: <textbox/>
    </hlayout>
    <hlayout sclass="z-valign-bottom">
        bottom: <textbox/>
    </hlayout>
</vlayout>
```

[since 6.0.0]

The default value of alignment has been changed to *top*. You can change it to *middle* (aka., *center*) by specifying `valign="middle"`, and *bottom* by `valign="bottom"`. For example,

```
<hlayout id="hlOne" height="100px">
    <button id="lbOne" label="align top" />
    <window width="100px" height="100px" title="test window" border="normal" />
</hlayout>
<hlayout id="hlTwo" valign="middle" height="100px">
    <button id="lbTwo" label="align middle" />
    <window width="100px" height="100px" title="test window" border="normal" />
</hlayout>
<hlayout id="hlThree" valign="bottom" height="100px">
    <button id="lbThree" label="align bottom" />
    <window width="100px" height="100px" title="test window" border="normal" />
</hlayout>
```

IE6 Limitation

Notice that, since the vertical alignment is specified in the CSS class (`Component.setSclass(java.lang.String)` [3]), there are some limitations for IE6. First, the vertical alignment is inherited to the inner hlayout. Thus, you have to specify the *middle* alignment explicitly in the inner hlayout if needed. For example,

```
<hlayout sclass="z-valign-bottom">
    bottom: <textbox/>
    <vlayout>
        <hlayout sclass="z-valign-middle">inner: <textbox/></hlayout>
    </vlayout>
</hlayout>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: HtmlBasedComponent

Supported Children

*ALL

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
5.0.4	August, 2010	new added component
5.0.5	October, 2010	Vertical alignment was supported.
6.0.0	February, 2012	The default value of alignment is change to <i>top</i> .

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Hlayout.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/box/Hlayout.html#>
- [3] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/Component.html#setSclass\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/Component.html#setSclass(java.lang.String))

Linelayout

Linelayout

- Java API: Linelayout ^[1]
- JavaScript API: Linelayout ^[2]

- Available for ZK:

-   

[Since 9.0.0]

Employment/Purpose

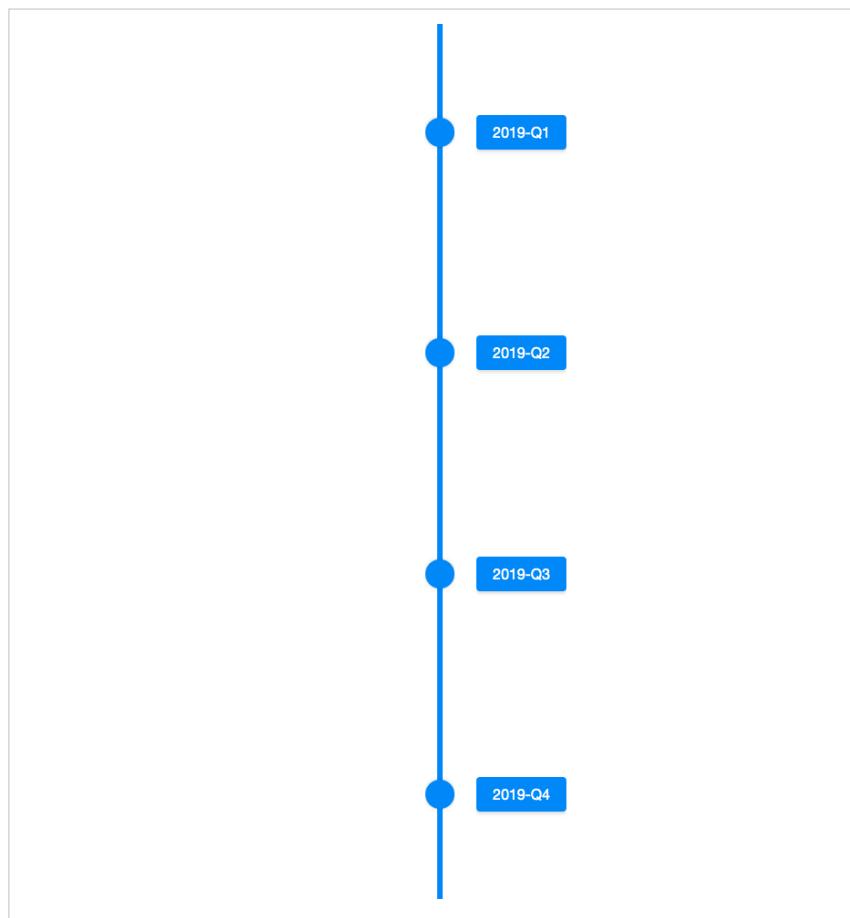
Linelayout is a component for displaying chronological information in a flexible and easy way; in addition to static data it also support ListModel. Linelayout only accept Lineitem as its child.

A Linelayout is composed of three parts: first area, line area and last area. The first and last areas are the containers for the content, and the line area contains the line and the point.

Browser Support

- This component is based on CSS Flexbox ^[3] and is compatible with browsers that support CSS Flexbox such as IE11+, Chrome and Firefox. Please check browser compatibility before using it.

Example

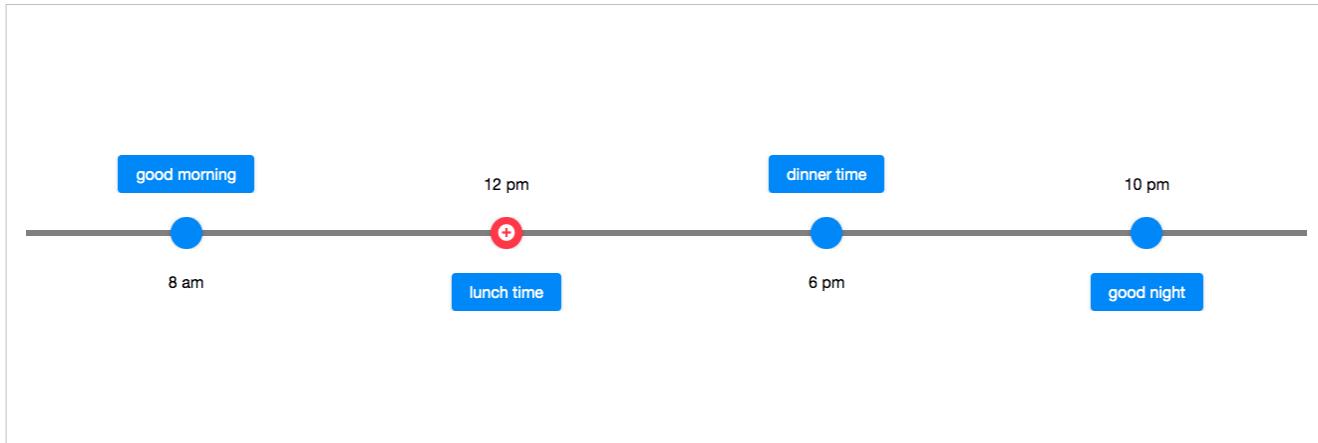


```
<zscript>
ListModel model = new ListModelList(new String[] {
    "2019-Q1",
    "2019-Q2",
    "2019-Q3",
    "2019-Q4"
});
</zscript>
<linelayout model="${model}">
<template name="model">
    <lineitem>
        <button label="${each}"></button>
    </lineitem>
</template>
</linelayout>
```

Properties

Orient

Whether the linelayout displays vertically or horizontally. The default value is "vertical".



```
<linelayout orient="horizontal" lineStyle="background:rgba(0,0,0,0.5)" >
  <lineitem>
    <label>8 am</label>
    <button>good morning</button>
  </lineitem>
  <lineitem opposite="true" pointIconSclass="z-icon-plus-circle" pointStyle="background: #FF4051">
    <label>12 pm</label>
    <button>lunch time</button>
  </lineitem>
  <lineitem>
    <label>6 pm</label>
    <button>dinner time</button>
  </lineitem>
  <lineitem opposite="true">
    <label>10 pm</label>
    <button>good night</button>
  </lineitem>
</linelayout>
```

LineStyle

The CSS inline style for the line.

FirstScale

The scale of space occupied by the first area. The default value is 1, you can change the position of the line by adjusting this property. For example, set firstScale as 1 and lastScale as 3 in a vertical linelayout, the line will be rendered a quarter away from the left boundary since the space ratio becomes 1:3..

LastScale

The scale of space occupied by the last area, the default value is 1.

Please refer to FirstScale.

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

* Lineitem

Version History

Version	Date	Content
9.0.0	Nov 2019	ZK-4377 [4]: Provide a Linelayout component

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Linelayout.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/layout/Linelayout.html#>
- [3] <https://developer.mozilla.org/en-US/docs/Web/CSS/flex>
- [4] <https://tracker.zkoss.org/browse/ZK-4377>

Lineitem

Lineitem

- Java API: Lineitem ^[1]
- JavaScript API: Lineitem ^[2]
- Available for ZK:
-   

[Since 9.0.0]

Employment/Purpose

Lineitem is the child of Linelayout, it can contain up to 2 components inside. The content of Lineitem will be placed separately in different area of linelayout.

Properties

Note: Lineitem does not support inline style, but you can still set height/width and vflex/hflex to customize spacing.

PointVisible

Sets whether the point is visible, the default value is true. This property only affects the point.

PointStyle

The CSS inline style for the point.

PointImageSrc

The source URI of the point background image.

PointImageContent

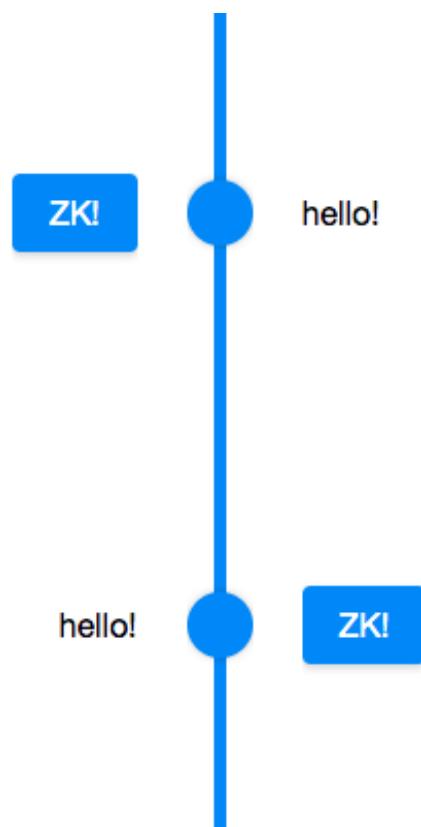
The point background Image content.

PointIconSclass

Specify the sclass name of the point icon.

Opposite

Set whether the first child is displayed in the first area(left when it is vertical / top when it is horizontal). The default value is false. Once the first child is displayed in the first or last area, the second child will be displayed in the other area.



```
<zk>
    <linelayout height="400px">
        <lineitem>
            <label>hello!</label>
            <button label="ZK!"></button>
        </lineitem>
        <lineitem opposite="true">
            <label>hello!</label>
            <button label="ZK!"></button>
        </lineitem>
    </linelayout>
</zk>
```

FrontSpace

Sets additional spacing from the previous lineitem. (such as "5px" or "5em"). If null or empty (""), the default spacing is used (i.e., controlled by CSS alone).

BackSpace

Sets additional spacing to the next lineitem. (such as "5px" or "5em"). If null or empty (""), the default spacing is used (i.e., controlled by CSS alone).

Supported Children

* ALL

Version History

Version	Date	Content
9.0.0	Nov 2019	ZK-4377 ^[4] : Provide a Linelayout component

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Lineitem.html#>

[2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/layout/Lineitem.html#>

Organigram

Organigram

- Java API: Organigram ^[1]
- JavaScript API: Organigram ^[2]
- Available for ZK:
 - CE
 - PE
 - EE

Employment/Purpose

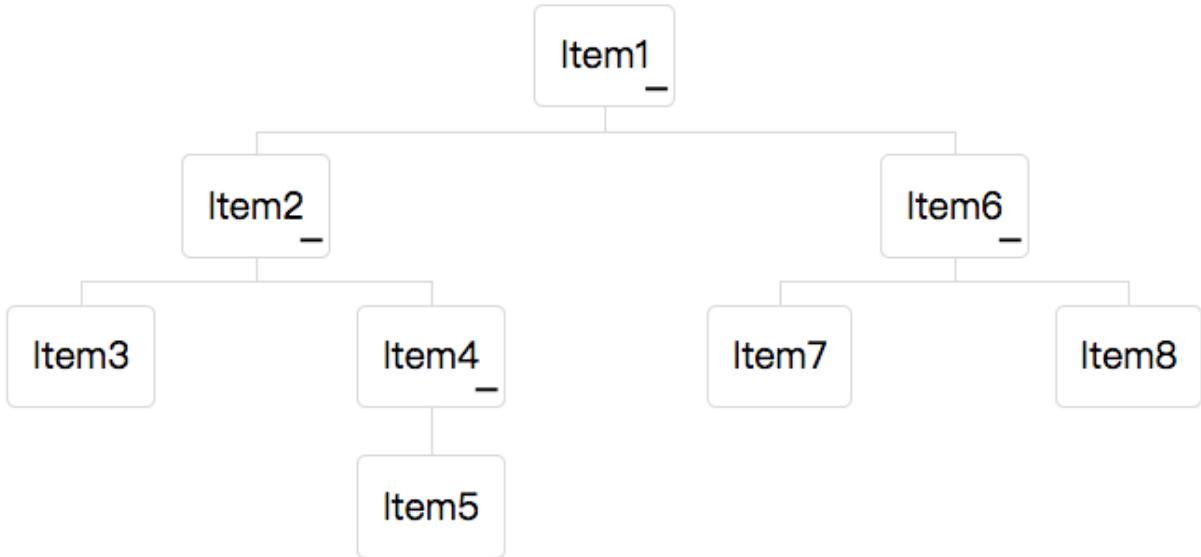
Organigram is a component for showing organizational chart by using tree data structure, it also support TreeModel to hold data, Organigram only accept one Orgchildren as child, developers can put Orgchildren, Orgitem and Orgnode in Organigram to display data.

Organigram supports Client Render on Demand ^[3]

Browser Support

- For IE, this component only supports 11+, it is based on CSS Flexbox ^[3], please check browser compatibility before using it.

Example



```
<organigram width="600px">
  <orgchildren>
    <orgitem label="Item1">
      <orgchildren>
        <orgitem label="Item2">
          <orgchildren>
            <orgitem label="Item3"/>
            <orgitem label="Item4">
              <orgchildren>
                <orgitem label="Item5"/>
              </orgchildren>
            </orgitem>
          </orgchildren>
        </orgitem>
        <orgitem label="Item6">
          <orgchildren>
            <orgitem label="Item7"/>
            <orgitem label="Item8"/>
          </orgchildren>
        </orgitem>
      </orgchildren>
    </orgitem>
  </orgchildren>
</organigram>
```

Model-Driven Display



```

<zscript><![CDATA[
    DefaultTreeNode root = new DefaultTreeNode(null, new
DefaultTreeNode[] {
        new DefaultTreeNode("Item1", new DefaultTreeNode[] {
            new DefaultTreeNode("Item2"), new
DefaultTreeNode("Item3"), new DefaultTreeNode("Item4")
        })
    });
    DefaultTreeModel model = new DefaultTreeModel(root);
    model.addOpenPath(new int[]{0});
]]></zscript>
<organigram width="600px" model="${model}" />
  
```

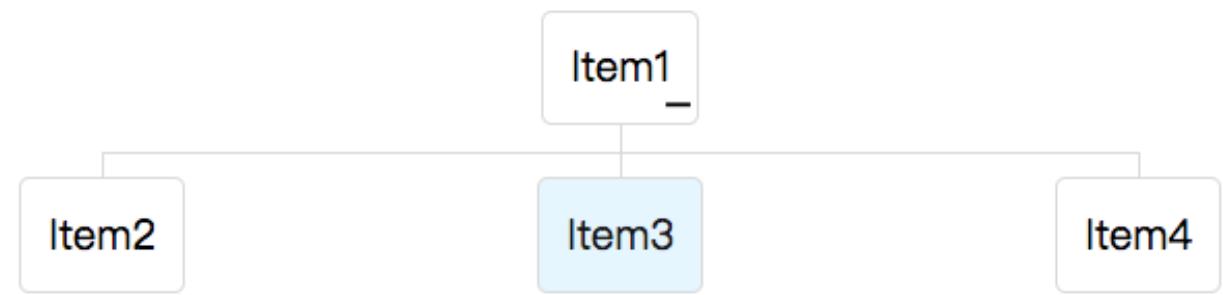
Selection

Organigram supports single selection, Orgitem can be selected by users clicking or by programming:

Organigram.setSelectedItem(org.zkoss.zkmax.zul.Orgitem)^[4], Orgitem.setSelected(java.lang.Boolean)^[5].

When an Orgitem is selected, the onSelect event will sent back to the server to notify the application, you can call Organigram.getSelectedItem()^[6] to get the selected Orgitem.

Example:



```

<organigram width="600px" onSelect="Clients.log(self.getSelectedItem().getLabel())">
    <orgchildren>
        <orgitem label="Item1">
            <orgchildren>
                <orgitem label="Item2"/>
                <orgitem label="Item3" selected="true"/>
                <orgitem label="Item4"/>
            </orgchildren>
        </orgitem>
    </orgchildren>
  
```

```
</orgitem>
</orgchildren>
</organigram>
```

Custom Orgitem

you can also render Organigram with your customized Renderer and Template, please refer to:

Organigram Template ^[7]

Organigram Renderer ^[8]

Supported Events

Name	Event Type
onSelect	Event: SelectEvent ^[6] Notifies one that the user has selected a new item in the organigram.

- Inherited Supported Events: XulElement

Supported Children

* Orgchildren

Version History

Version	Date	Content

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Organigram.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/layout/Organigram.html#>
- [3] https://www.zkoss.org/wiki/ZK_Developer%27s_Reference/Performance_Tips/Client_Render_on_Demand
- [4] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Organigram.html#setSelectedItem\(org.zkoss.zkmax.zul.Orgitem\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Organigram.html#setSelectedItem(org.zkoss.zkmax.zul.Orgitem))
- [5] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Orgitem.html#setSelected\(java.lang.Boolean\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Orgitem.html#setSelected(java.lang.Boolean))
- [6] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Organigram.html#getSelectedItem\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Organigram.html#getSelectedItem())
- [7] https://www.zkoss.org/wiki/ZK_Developer%27s_Reference/MVC/View/Template/Organigram_Template
- [8] https://www.zkoss.org/wiki/ZK_Developer%27s_Reference/MVC/View/Renderer/Organigram_Renderer

Orgchildren

Orgchildren

- Java API: Orgchildren ^[1]
- JavaScript API: Orgchildren ^[2]
- Available for ZK:
- **CE** **PE** **EE**

[Since 8.6.0]

Employment/Purpose

Orgchildren contains a collection of Orgitem components. It is main body of the Organigram and it also the main body of an Orgitem's children.

Example



```
<organigram width="600px">
    <orgchildren>
        <orgitem label="Item1">
            <orgchildren>
                <orgitem label="Item2">
                    <orgchildren>
                        <orgitem label="Item3"/>
                        <orgitem label="Item4"/>
                    </orgchildren>
                </orgitem>
            </orgchildren>
        </orgitem>
    </orgchildren>
</organigram>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

* Orgitem

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Orgchildren.html#>
[2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/layout/Orgchildren.html#>

Orgitem

Orgitem

- Java API: Orgitem ^[1]
- JavaScript API: Orgitem ^[2]
- Available for ZK:
- CE PE EE

[Since 8.6.0]

Employment/Purpose

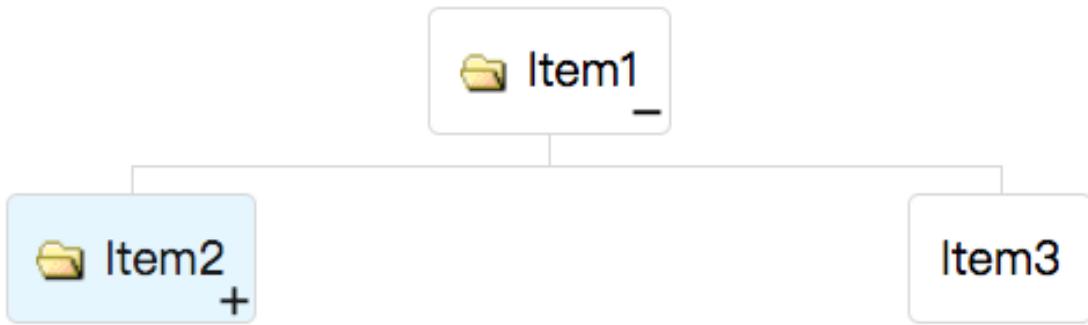
Orgitem contains a node (Orgnode) , and an optional Orgchildren.

If the component doesn't contain an Orgchildren, it is a leaf node that doesn't accept any child items.

If it contains an Orgchildren, it is a branch node that might contain other items.

For a branch node, an +/- button will appear at the bottom right of the node, such that user could open and close the item by clicking on the +/- button.

Example



```
<organigram width="600px">
    <orgchildren>
        <orgitem image="img/folder.gif" label="Item1">
            <orgchildren>
                <orgitem image="img/folder.gif" label="Item2" selected="true" open="false">
                    <orgchildren>
                        <orgitem label="Item4"/>
                    </orgchildren>
                </orgitem>
                <orgitem label="Item3"/>
            </orgchildren>
        </orgitem>
    </orgchildren>
</organigram>
```

Open

Each Orgitem contains the open property which is used to control whether to display its child items. The default value is true. By setting this property to false, you are able to control what part of the Organigram is invisible.

When a user clicks on the +/- button, he opens the Orgitem and makes its children visible. The onOpen event is then sent to the server to notify the application.

You can also open or close the Orgitem by calling Orgitem.setOpen(java.lang.Boolean) ^[3] and get the open state by calling Orgitem.isOpen() ^[4].

Example:

```
<organigram>
    <orgchildren>
        <orgitem label="Item1" open="false" onOpen="createChild()">
            <orgchildren/>
        </orgitem>
    </orgchildren>
    <zscript><![CDATA[
        void createChild() {
            if (event.isOpen())
                new Orgitem("new item").setParent(self.getOrgchildren());
        }
    ]]>
```

```
]]></zscript>
</organigram>
```

Selected

By default, each Orgitem can be selected by users clicking or by programming:

Organigram.setSelectedItem(org.zkoss.zkmax.zul.Orgitem) ^[4] or Orgitem.setSelected(java.lang.Boolean) ^[5]

You can get the selected state by calling Orgitem.isSelected() ^[5]

If you don't allow users to select Orgitem, you can write as following:

```
<orgitem selectable="false"/>
```

or

```
<orgitem disabled="true"/>
```

Disabled has more obvious style to prompt users.

Label and Image

Orgitem provides Orgitem.setImage(java.lang.String) ^[6] and Orgitem.setLabel(java.lang.String) ^[7] to simplify the assignment of image and label to an Orgitem. However, they are actually placed in the node (of the child Orgnode). Furthermore, if the Orgnode is not created, they will be created automatically. For example,

```
<orgitem label="Hello"/>
```

is equivalent to

```
<orgitem>
    <orgnode label="Hello"/>
</orgitem>
```

It also means you cannot attach an Orgnode child to the Orgitem, after setImage or setLabel was invoked. It means, though a bit subtle, the following will cause an exception:

```
<orgitem label="Hello"> <!-- Orgnode is created automatically because of setLabel -->
    <orgnode/> <!-- exception since only one Orgnode is allowed per Orgitem -->
</orgitem>
```

When your Organigram only contains image and text, It is a convenient way to create Organigram without Orgnode tags, if you want to put other components in Orgnode, you could write like following:

```
<zscript><![CDATA[
    Orgchildren orgchildren;
    void newItem(String label) {
        if (orgitem.getOrgchildren() == null) {
            orgchildren = new Orgchildren();
            orgchildren.setParent(orgitem);
        }
        new Orgitem(label).setParent(orgchildren);
    }
]]></zscript>
```

```

<organigram>
  <orgchildren>
    <orgitem id="orgitem">
      <orgnode>
        <textbox onOK="newItem(self.value)"/>
      </orgnode>
    </orgitem>
  </orgchildren>
</organigram>

```

Supported Events

Name	Event Type
onOpen	Event: OpenEvent [3] Denotes user has opened or closed a component. It is useful to implement load-on-demand by listening to the onOpen event, and creating components when the first time the component is opened.

- Inherited Supported Events: XulElement

Supported Children

* Orgnode, Orgchildren

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Orgitem.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/layout/Orgitem.html#>
- [3] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Orgitem.html#setOpen\(java.lang.Boolean\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Orgitem.html#setOpen(java.lang.Boolean))
- [4] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Orgitem.html#isOpen\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Orgitem.html#isOpen())
- [5] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Orgitem.html#isSelected\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Orgitem.html#isSelected())
- [6] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Orgitem.html#setImage\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Orgitem.html#setImage(java.lang.String))
- [7] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Orgitem.html#setLabel\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Orgitem.html#setLabel(java.lang.String))

Orgnode

Orgnode

- Java API: Orgnode ^[1]
- JavaScript API: Orgnode ^[2]
- Available for ZK:
-   

[Since 8.6.0]

Employment/Purpose

Orgnode represents data in an Orgitem. Orgnode can contain any components in it, such as label, image, textbox etc.

Example

```
<organigram width="600px">
    <orgchildren>
        <orgitem>
            <orgnode>
                <button label="Snapshot" onClick="camera.snapshot()"/>
            </orgnode>
            <orgchildren>
                <orgitem>
                    <orgnode width="200px" label="Camera">
                        <camera id="camera" onSnapshotUpload="image.setContent(event.media)"/>
                    </orgnode>
                </orgitem>
                <orgitem>
                    <orgnode width="200px" label="Image">
                        <image id="image"/>
                    </orgnode>
                </orgitem>
            </orgchildren>
        </orgitem>
    </orgchildren>
</organigram>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: LabelImageElement

Supported Children

*ALL

Version History

Version	Date	Content

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Orgnode.html#>
[2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/layout/Orgnode.html#>

Portallayout

Portallayout

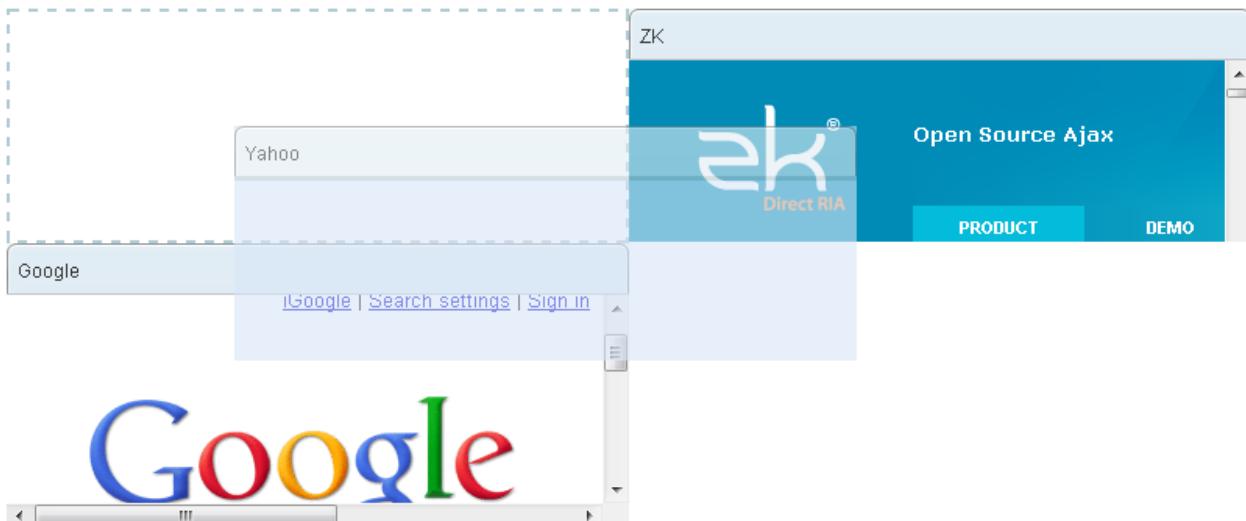
- Demonstration: Portallayout ^[1]
- Java API: Portallayout ^[2]
- JavaScript API: Portallayout ^[3]
- Style Guide: Portallayout
- Available for ZK:
- CE PE EE

Employment/Purpose

A portallayout lays out a container which can have multiple columns, and each column might have any number panels placed vertically with different heights. Portallayout allows users to drag-and-drop a panel to change its location.

When using Portallayout, you have to assign the width (either percentage or pixel) to each Portalchildren, or the result might depend on the browser, and not as expected.

Example



```
<portallayout height="100%">
    <portalchildren width="50%">
        <panel height="50%" title="Demo">
            <panelchildren>
                <iframe height="100%" width="100%" src="https://www.zkoss.org/zkdemo"/>
            </panelchildren>
        </panel>
        <panel height="50%" title="Doc">
            <panelchildren>
                <iframe height="100%" width="100%" src="https://www.zkoss.org/documentation"/>
            </panelchildren>
        </panel>
    </portalchildren>
    <portalchildren width="50%">
        <panel height="100%" title="ZK">
            <panelchildren>
                <iframe height="100%" width="100%" src="https://www.zkoss.org/"/>
            </panelchildren>
        </panel>
    </portalchildren>
</portallayout>
```

Row-based layout

Since ZK 7.0.0

If you want the portallayout displayed as row-based layout, you can specify the *orient* property to *horizontal*.

Default is *vertical*.

For example,

```
<portallayout orient="horizontal">
    <portalchildren width="50%">
        <panel height="150px" title="Yahoo">
            <panelchildren>
                <iframe width="100%" src="http://www.yahoo.com/" />
            </panelchildren>
        </panel>
        <panel height="300px" title="Google">
            <panelchildren>
                <iframe width="100%" src="http://www.google.com/" />
            </panelchildren>
        </panel>
    </portalchildren>
    <portalchildren width="50%">
        <panel height="150px" title="ZK">
            <panelchildren>
                <iframe width="100%" src="http://www.zkoss.org/" />
            </panelchildren>
        </panel>
    </portalchildren>
</portallayout>
```

Supported Events

Name	Event Type
onPortalMove	Event: PortalMoveEvent ^[4] Represents an event caused by a portal being moved.
onPortalDrop	Event: PortalDropEvent ^[5] Represents an event after a portal being dropped and before a portal being moved. (Since: 9.5.1)

- Inherited Supported Events: XulElement

Supported Children

* Portalchildren

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
7.0.0	October, 2013	Portallayout supports row based orientation [6]
9.5.1	November, 2020	Kanban missing options to listen to portallayout onPortalMove without affecting the UI [7]

References

- [1] http://www.zkoss.org/zkdemo/layout/portal_layout
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Portallayout.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/layout/Portallayout.html#>
- [4] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/ui/event/PortalMoveEvent.html#>
- [5] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/ui/event/PortalDropEvent.html#>
- [6] <http://tracker.zkoss.org/browse/ZK-1687>
- [7] <https://tracker.zkoss.org/browse/ZK-4423>

Portalchildren

Portalchildren

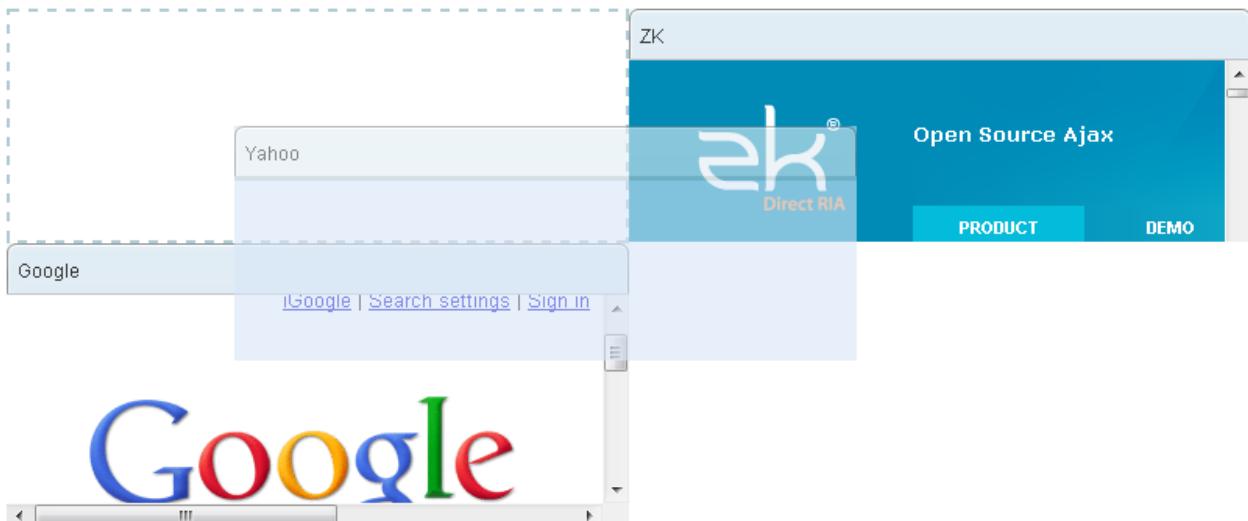
- Demonstration: Portallayout [1]
- Java API: Portalchildren [1]
- JavaScript API: Portalchildren [2]
- Style Guide: Portallayout
- Available for ZK:
-   

Employment/Purpose

The column of Portallayout

Child of Portalchildren can only be Panel

Example



```
<portallayout>
    <portalchildren width="50%">
        <panel height="150px" title="Yahoo">
            <panelchildren>
                <iframe width="100%" src="http://www.yahoo.com/" />
            </panelchildren>
        </panel>
        <panel height="300px" title="Google">
            <panelchildren>
                <iframe width="100%" src="http://www.google.com/" />
            </panelchildren>
        </panel>
    </portalchildren>
    <portalchildren width="50%">
```

```
<panel height="150px" title="ZK">
  <panelchildren>
    <iframe width="100%" src="http://www.zkoss.org/" />
  </panelchildren>
</panel>
</panelchildren>
</portallayout>
```

Properties

Title

[Since 9.0.0]

Sets the title of the portalchildren. If the title is not empty/null, frame design will be applied.

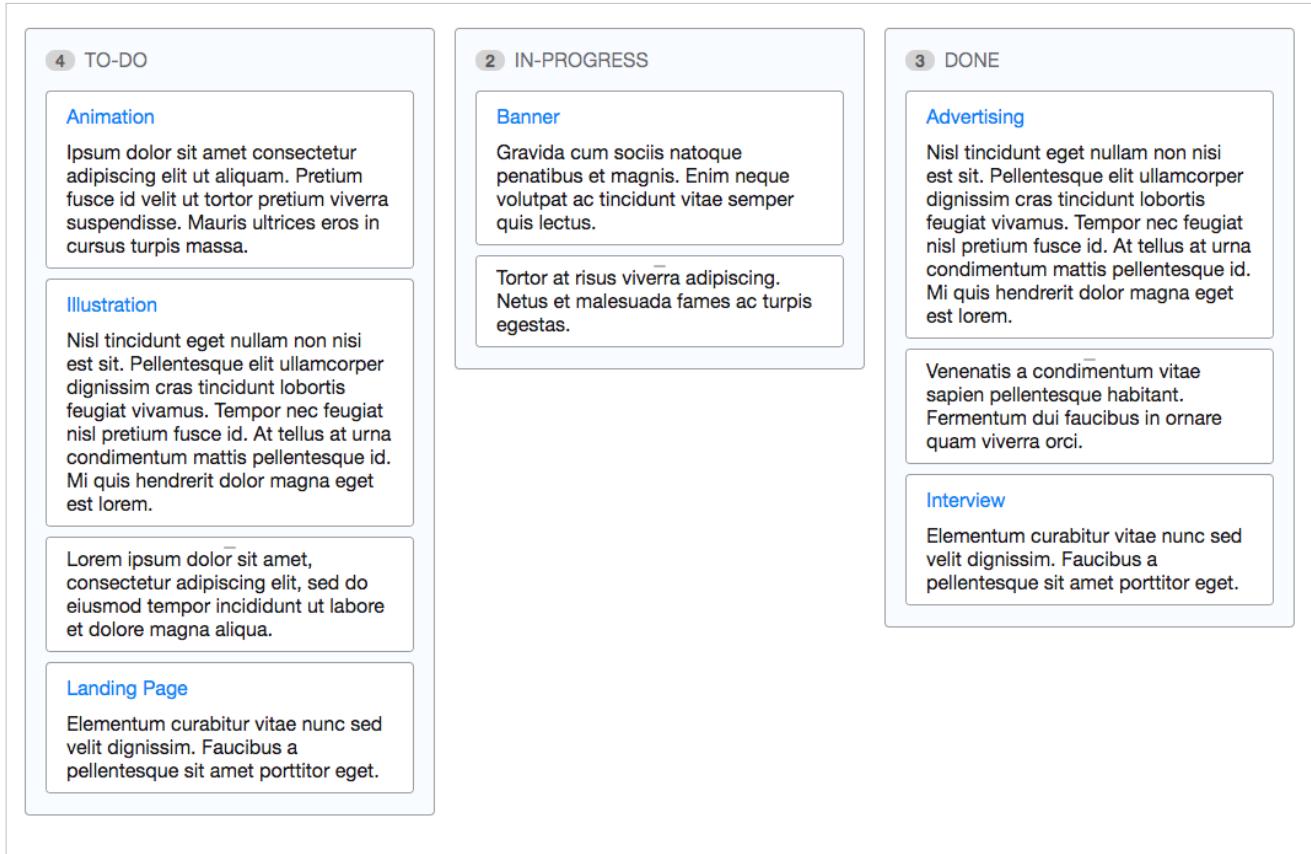
Frame Design

PortalChildren now provides a frame design, making it extremely easy to create a Kanban-like layout for your application. To turn on the PortalChildren frame design, just specify the title attribute on the PortalChildren.

Example

Each PortalChildren with title can be used as a Kanban board column to represent a process stage, and the panels inside each column represent tasks in the said stage. The number next to the PortalChildren title is a counter, indicating the total number of panels inside the said column.

If a panel has a panel title, users can drag and hold the panel title to move it to the appropriate column. If the panel does not have a panel title, you can drag the small dragging button at the top of the panel to move.



```

<zk>
  <style>
    .z-panel {
      width: 300px;
    }
  </style>
  <portallayout>
    <portalchildren title="TO-DO">
      <panel title="Animation" border="normal">
        <panelchildren>.....</panelchildren>
      </panel>
      <panel title="Illustration" border="normal">
        <panelchildren>.....</panelchildren>
      </panel>
      <panel border="normal">
        <panelchildren>.....</panelchildren>
      </panel>
      <panel title="Landing Page" border="normal">
        <panelchildren>.....</panelchildren>
      </panel>
    </portalchildren>
    <portalchildren title="IN-PROGRESS">
      <panel title="Banner" border="normal">
        <panelchildren>.....</panelchildren>
      </panel>
    </portalchildren>
  </portallayout>

```

```

<panel border="normal">
    <panelchildren>.....</panelchildren>
</panel>
</portalchildren>
<portalchildren title="DONE" >
    <panel title="Advertising" border="normal">
        <panelchildren>.....</panelchildren>
    </panel>
    <panel border="normal">
        <panelchildren>.....</panelchildren>
    </panel>
    <panel title="Interview" border="normal">
        <panelchildren>.....</panelchildren>
    </panel>
</portalchildren>
</portallayout>
</zk>

```

CounterVisible

[Since 9.0.0]

Sets whether the counter is visible. Meaningful only if frame design is applied.

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

* Panel

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
9.0.0	Nov 2019	ZK-4398 [3]: Provide PortalChildren title and frame design

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Portalchildren.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/layout/Portalchildren.html#>
- [3] <https://tracker.zkoss.org/browse/ZK-4398>

Rowlayout

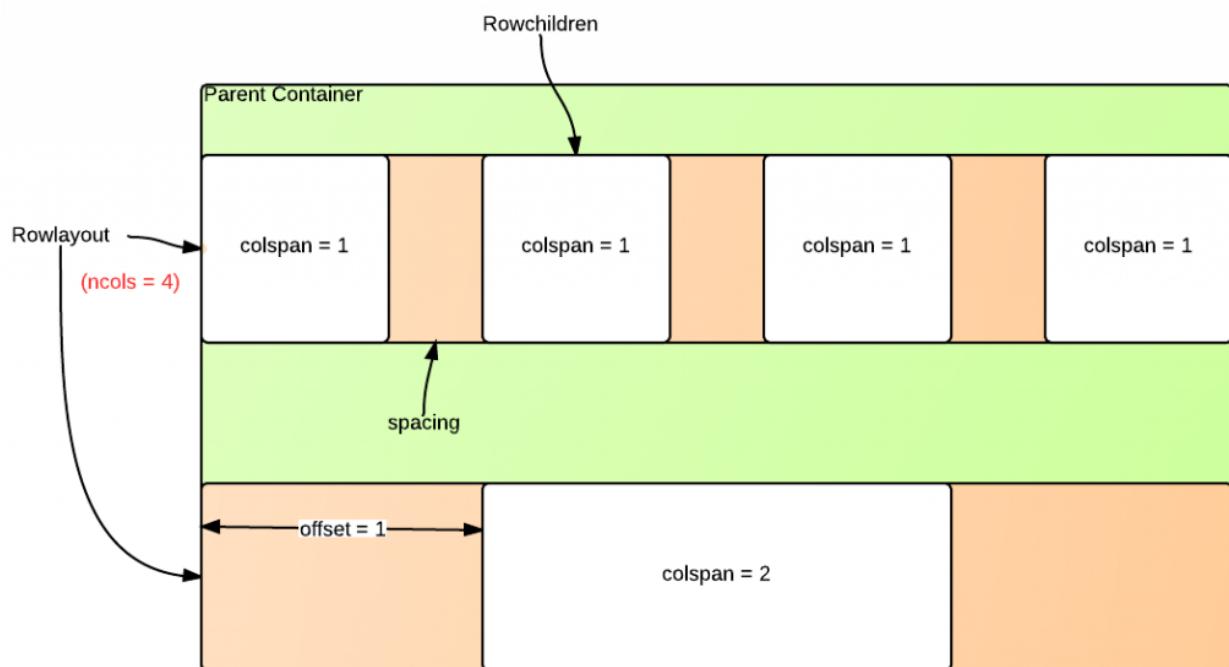
Rowlayout

- Java API: RowLayout ^[1]
- JavaScript API: RowLayout ^[2]
- Style Guide: RowLayout
- Available for ZK:
- CE PE EE

Employment/Purpose

A `rowlayout` lays out a container which can have multiple columns, it offers a 12-column grid out of the box. You can simply choose the number of columns to occupy for each major content area, and may also skip columns for extra space without inserting space-inducing elements.

The following diagram illustrates the `rowlayout`/`rowchildren` components and their various configurable parameters.



Example

Using rowlayout component is simple. First, use rowlayout to divide the horizontal space of its parent container into a number of columns. You can also optionally specify the column/spacing ratio. The default number of columns is 12, and the default column/spacing ratio is 1/3, which means column is 3 times wider than the spacing between columns. Spacing could be given as a ratio, a percentage or a floating-point number.

Next, use rowchildren component to place components into an integral number of these columns. You can also optionally specify how many columns to skip ahead.

Equally Divided

```
<rowlayout ncols="12">
  <forEach begin="1" end="3">
    <rowchildren colspan="4" style="background-color: skyblue">
      1/3
    </rowchildren>
  </forEach>
</rowlayout>
```

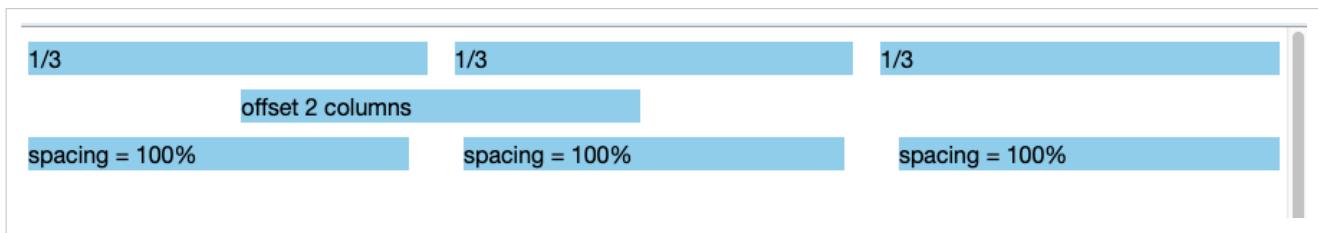
Position Offset

```
<rowlayout ncols="12">
  <rowchildren colspan="4" style="background-color: skyblue" offset="2">
    offset 2 columns
  </rowchildren>
</rowlayout>
```

Column Spacing

```
<rowlayout ncols="12" spacing="100%">
  <forEach begin="1" end="3">
    <rowchildren colspan="4" style="background-color: skyblue">
      spacing = 100%
    </rowchildren>
  </forEach>
</rowlayout>
```

The above examples look like:



Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

* Rowchildren

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content
]

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/RowLayout.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/layout/RowLayout.html#>

Rowchildren

Rowchildren

- Java API: Rowchildren ^[1]
- JavaScript API: Rowchildren ^[2]
- Style Guide: Rowchildren
- Available for ZK:
- CE PE EE

Employment/Purpose

The column of Rowlayout.

Properties

Colspan

By default, the colspan of Rowchildren is 1.

```
<rowlayout>
    <rowchildren colspan="10">
        <window border="normal" title="colspan=10"/>
    </rowchildren>
</rowlayout>
```

Offset

By default, the offset of Rowchildren is 0.

```
<rowlayout>
    <rowchildren offset="10">
        <window border="normal" title="offset=10"/>
    </rowchildren>
</rowlayout>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Version History

Version	Date	Content

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Rowchildren.html#>
[2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/layout/Rowchildren.html#>

Splitlayout

Splitlayout

- Demonstration: Splitlayout ^[1]
- Java API: Splitlayout ^[2]
- JavaScript API: Splitlayout ^[3]
- Available for ZK:
-   

Employment/Purpose

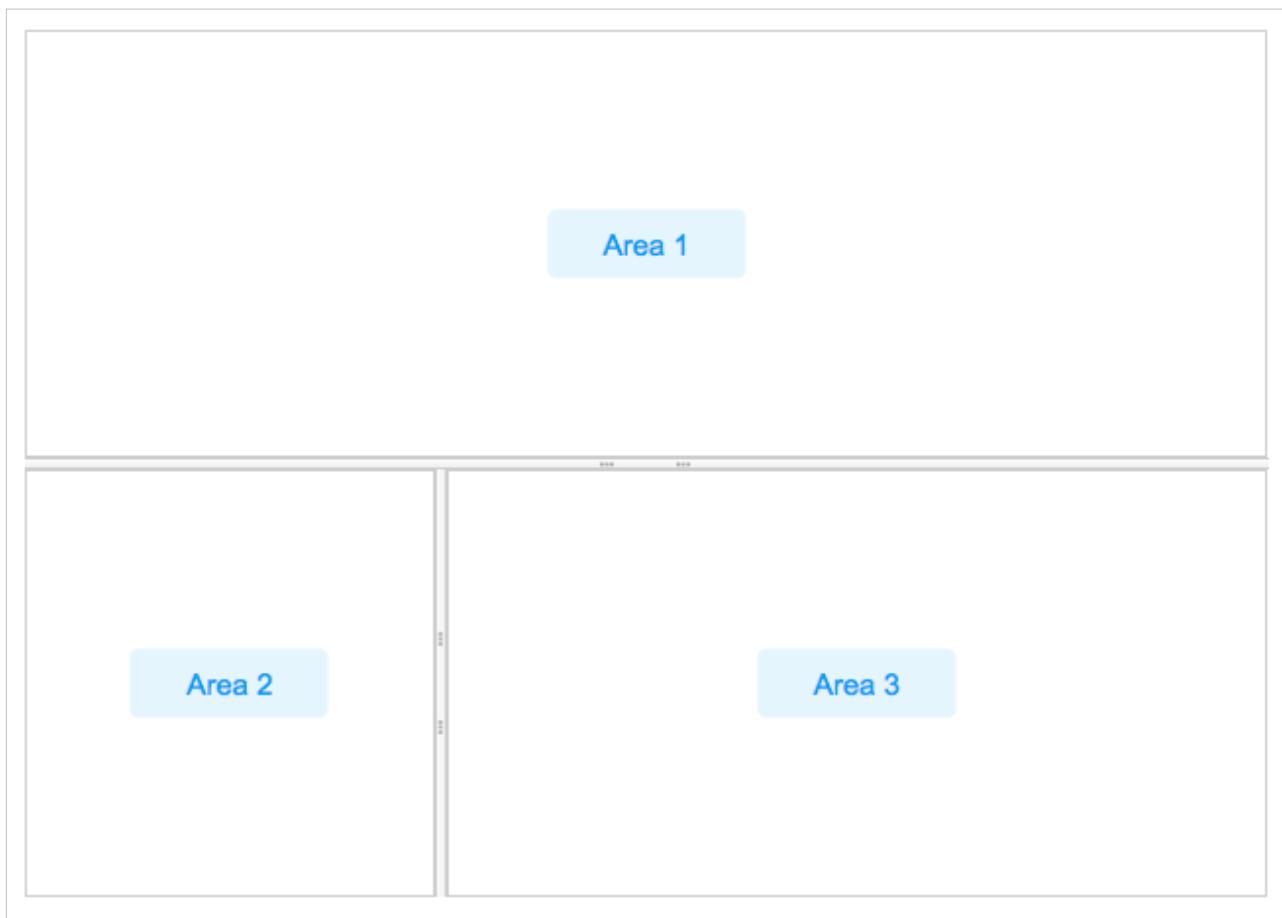
Splitlayout is a layout container, which is used to divide a component into two components.

These two components inside Splitlayout, the splitlayouts, are placed either horizontally or vertically by setting the orientation, and users can easily resize these two viewports by dragging the splitter bar. Also, like other ZK layout components, it supports hflex and vflex, indicating that users can divide the area into three or more spaces by putting the splitlayouts into the outer Splitlayout.

Comparing to Splitter, this component has several advantages:

1. Splitter can only be used inside Hbox/ Vbox. Splitlayout has no such limitation.
2. Hbox/Vbox are both rendered with HTML which is heavy for a browser. Splitlayout is rendered with div which is more light-weighted.

Example



```
<splitlayout vflex="1" hflex="1">
    <div sclass="area" hflex="1" vflex="1">
        <label value="Area 1"/>
    </div>
    <splitlayout vflex="1" hflex="1" orient="horizontal" >
        <div sclass="area" vflex="1" hflex="1">
            <label value="Area 2"/>
        </div>
        <div sclass="area" vflex="1" hflex="2">
            <label value="Area 3"/>
        </div>
    </splitlayout>
</splitlayout>
```

Properties and Features

Orientation

The orientation indicates the how these two children container would display in splitlayout. Supported value: (default) "vertical" or "horizontal".

Collapse

The collapse property (Splitlayout.setCollapse(java.lang.String) [4]) specifies which side of the splitter is collapsed when its grippy (button) is clicked. If this property is not specified, the splitter will not cause a collapse.

Supported value: (default) "none", "before" or "after". "before" means that the splitter in splitlayout would collapse to the left/top, and "after" means splitter in splitlayout would collapse to the right/button.

Open

This method would not be able to work if the "collapse" attribute is not specified.

onOpen Event

When a splitlayout is collapsed or opened by a user, the onOpen event (OpenEvent [3]) is sent to the application.

Widths and Heights

specify widths or heights with a list of numbers(in pixel) separated by a comma to denote the width/height of two areas in splitlayout. Notice that you should use them while using flex in the children component.

Specify Children Size in Proportion

Specify hflex/vflex on 2 children like:

```
<splitlayout vflex="1" >
  <div sclass="area" vflex="2">
    ...
  </div>
  <div sclass="area" vflex="3">
    ...
  </div>
</splitlayout>
```

MinWidths and MinHeights

User setMinWidths(String minWidths) and setMinHeights(String minHeights) to sets the minimum widths/heights in the same format of setWidths/setHeights. When user drag the splitter, the two areas will not be smaller than the minWidths/minHeights.

Supported Events

Name	Event Type
onOpen	Event: OpenEvent [3] When a splitter is collapsed or opened by a user, the <code>onOpen</code> event is sent to the application.

- Inherited Supported Events: XulElement

Supported Children

*ALL

Version History

Version	Date	Content
8.5.0	October, 2017	new added component

References

- [1] https://www.zkoss.org/zkdemo/layout/split_layout
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Splitlayout.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/layout/Splitlayout.html#>
- [4] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Splitlayout.html#setCollapse\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Splitlayout.html#setCollapse(java.lang.String))

Splitter

Splitter

- Demonstration: Splitter ^[1]
- Java API: Splitter ^[2]
- JavaScript API: Splitter ^[3]
- Style Guide: Splitter

Employment/Purpose

An element which should appear before or after an element inside a box (Box, Vbox and Hbox).

When the splitter is dragged, the sibling elements of the splitter are resized. If `getCollapse()` is true, a grippy is placed inside the splitter, and one sibling element of the splitter is collapsed when the grippy is clicked. Using the Splitter in combination with hbox/vbox allows resizing - making it draggable.

Example

Column 1-1: The left-top box. To know whether a splitter is collapsed, you can listen to the `onOpen` event.

Column 2: Whether a splitter allows users to open or collapse depending on the `collapse` attribute.

Column 1-2: You can enforce to open or collapse programming by calling `setOpen` method.

```
<hbox spacing="0" width="100%">
    <vbox height="200px">
        Column 1-1: The left-top box. To know whether a splitter is
        collapsed,
        you can listen to the onOpen event.
        <splitter collapse="after" />
        Column 1-2: You can enforce to open or collapse programming
        by calling
        setOpen method.
    </vbox>
    <splitter collapse="before" />
    Column 2: Whether a splitter allows users to open or collapse
    depending
    on the collapse attribute.
</hbox>
```

Properties and Features

Collapse

The collapse property (`Splitter.setCollapse(java.lang.String)`)^[4] specifies which side of the splitter is collapsed when its grippy (button) is clicked. If this property is not specified, the splitter will not cause a collapse (and the grippy/button won't appear).

Allowed values and their meaning are as follows.

Value	Description
none	No collapsing occurs.
before	When the grippy is clicked, the element immediately before the splitter in the same parent is collapsed so that its width or height is 0.
after	When the grippy is clicked, the element immediately after the splitter in the same parent is collapsed so that its width or height is 0.

Open

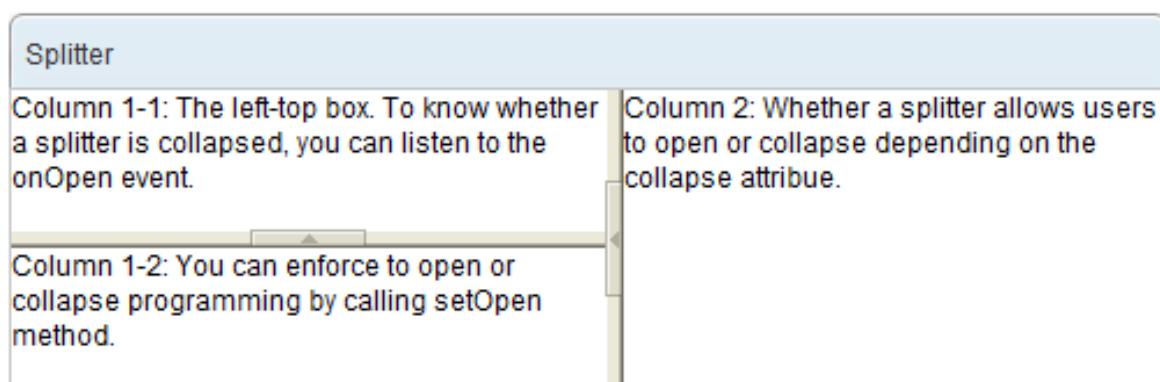
To know whether a splitter is collapsed, you can check the value of the `open` property (`Splitter.isOpen()`)^[5]. To open or collapse dynamically, you are able to set the value of the `open` property (`Splitter.setOpen(boolean)`)^[6].

onOpen Event

When a splitter is collapsed or opened by a user, the `onOpen` event (`OpenEvent`^[3]) is sent to the application.

OS Styling

If you want to change the styling to be more similar to OS's look as follows.



You could specify `HtmlBasedComponent.setZclass(java.lang.String)` [7] with `z-splitter-os-ver` or `z-splitter-os-hor`, depending on the orient is vertical or horizontal.

Here is an example that switches the styling between the default and OS-look:

```
<window>
    <panel title="Splitter" border="normal" width="500px">
        <panelchildren>
            <hbox spacing="0" width="100%" height="100%">
                <vbox spacing="0" width="100%" heights="60px, 60px">
                    Column 1-1: The left-top box. To know
                    whether a splitter
```

```

is collapsed, you can listen to the
onOpen event.

<splitter id="s1" collapse="before"/>
Column 1-2: You can enforce to open or
collapse programming
by calling setOpen method.

</vbox>
<splitter id="s2" collapse="before"/>
Column 2: Whether a splitter allows users to
open or collapse
depending on the collapse attribute.

</hbox>
</panelchildren>
</panel>
<button label="change style">
<attribute name="onClick">
if ("z-splitter-ver".equals(s1.getZclass()))
    s1.setZclass("z-splitter-os-ver");
else
    s1.setZclass("z-splitter-ver");
if ("z-splitter-hor".equals(s2.getZclass()))
    s2.setZclass("z-splitter-os-hor");
else
    s2.setZclass("z-splitter-hor");
</attribute>
</button>
</window>

```

Supported Events

Name	Event Type
onOpen	Event: OpenEvent [3] When a splitter is collapsed or opened by a user, the <code>onOpen</code> event is sent to the application.

- Inherited Supported Events: XulElement

Supported Molds

Available molds of a component are defined in lang.xml embedded in zul.jar.

Name	Snapshot
default	
os	

Supported Children

*NONE

Use Cases

Version	Description	Example Location
5.0	Used to separate contents within hbox/vbox.	[8]

Version History

Version	Date	Content

References

- [1] <http://www.zkoss.org/zkdemo/layout/splitter>
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Splitter.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/box/Splitter.html#>
- [4] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Splitter.html#setCollapse\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Splitter.html#setCollapse(java.lang.String))
- [5] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Splitter.html#isOpen\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Splitter.html#isOpen())
- [6] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Splitter.html#setOpen\(boolean\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Splitter.html#setOpen(boolean))
- [7] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/HtmlBasedComponent.html#setZclass\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/HtmlBasedComponent.html#setZclass(java.lang.String))
- [8] <http://www.zkoss.org/zksandbox/userguide/#113>

Tablelayout

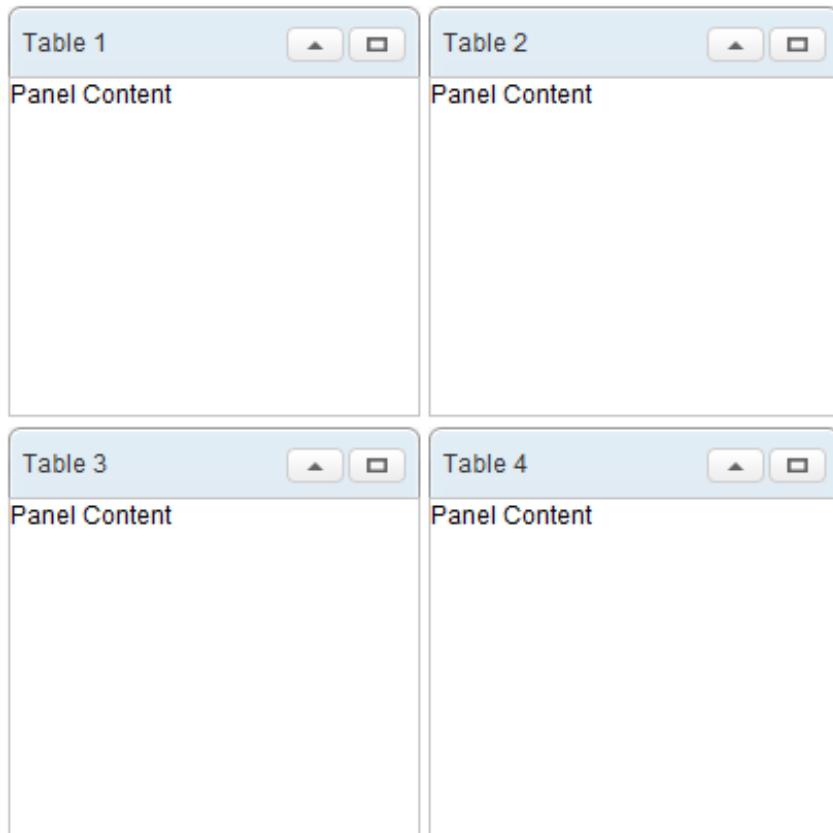
Tablelayout

- Demonstration: Tablelayout ^[1]
- Java API: Tablelayout ^[2]
- JavaScript API: Tablelayout ^[3]
- Style Guide: TableLayout
- Available for ZK:
- CE PE EE

Employment/Purpose

Tablelayout lays out a container as an HTML table in which the columns can be specified, and rowspan and colspan of its child can also be specified to create complex layouts within the table.

Example



```
<tablelayout columns="2">
    <tablechildren>
        <panel title="Table 1" border="normal" maximizable="true"
               collapsible="true" width="200px" height="200px">
            <panelchildren>Panel Content</panelchildren>
        </panel>
    </tablechildren>
```

```
<tablechildren>
    <panel title="Table 2" border="normal" maximizable="true"
        collapsible="true" width="200px" height="200px">
        <panelchildren>Panel Content</panelchildren>
    </panel>
</tablechildren>
<tablechildren>
    <panel title="Table 3" border="normal" maximizable="true"
        collapsible="true" width="200px" height="200px">
        <panelchildren>Panel Content</panelchildren>
    </panel>
</tablechildren>
<tablechildren>
    <panel title="Table 4" border="normal" maximizable="true"
        collapsible="true" width="200px" height="200px">
        <panelchildren>Panel Content</panelchildren>
    </panel>
</tablechildren>
</tablelayout>
```

The child of tablechildren can be any component:



```
<tablelayout columns="2">
    <tablechildren>
        <label value="Table 1" />
    </tablechildren>
    <tablechildren>
        <button label="Table 2" />
    </tablechildren>
    <tablechildren>
        <textbox value="Table 3" />
    </tablechildren>
    <tablechildren>
        <window border="normal">
            Table 4
        </window>
    </tablechildren>
</tablelayout>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

* Tablechildren

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content

References

- [1] http://www.zkoss.org/zkdemo/layout/table_layout
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Tablelayout.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/layout/Tablelayout.html#>

TableChildren

Tablechildren

- Demonstration: Tablelayout ^[1]
- Java API: Tablechildren ^[1]
- JavaScript API: Tablechildren ^[2]
- Style Guide: Tablelayout
- Available in ZK EE only ^[13]

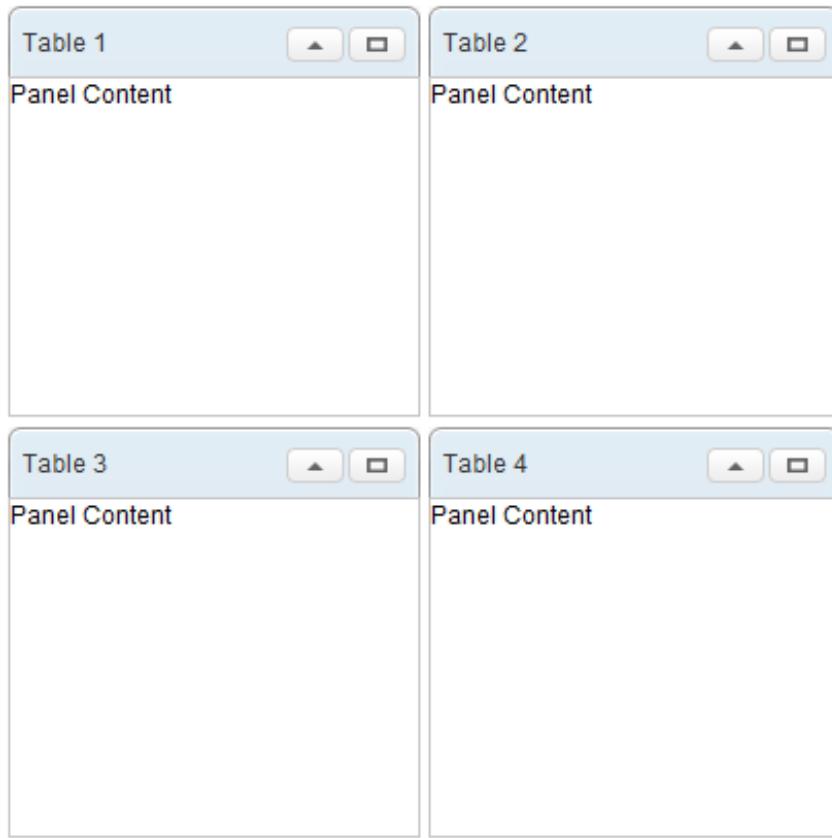
Employment/Purpose

The cell of Tablelayout. The child component of Tablechildren can only be Panel.

* [since 6.0.0]

The child of tablechildren can be any component.

Example



```
<tablelayout columns="2">
    <tablechildren>
        <panel title="Table 1" border="normal" maximizable="true"
            collapsible="true" width="200px" height="200px">
            <panelchildren>Panel Content</panelchildren>
        </panel>
```

```
</tablechildren>
<tablechildren>
    <panel title="Table 2" border="normal" maximizable="true"
        collapsible="true" width="200px" height="200px">
        <panelchildren>Panel Content</panelchildren>
    </panel>
</tablechildren>
<tablechildren>
    <panel title="Table 3" border="normal" maximizable="true"
        collapsible="true" width="200px" height="200px">
        <panelchildren>Panel Content</panelchildren>
    </panel>
</tablechildren>
<tablechildren>
    <panel title="Table 4" border="normal" maximizable="true"
        collapsible="true" width="200px" height="200px">
        <panelchildren>Panel Content</panelchildren>
    </panel>
</tablechildren>
</tablelayout>
```

[since 6.0.0]

The child of tablechildren can be any component.



```
<tablelayout columns="2">
    <tablechildren>
        <label value="Table 1" />
    </tablechildren>
    <tablechildren>
        <button label="Table 2" />
    </tablechildren>
    <tablechildren>
        <textbox value="Table 3" />
    </tablechildren>
    <tablechildren>
        <window border="normal">
            Table 4
        </window>
    </tablechildren>
</tablelayout>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

* Panel

[since 6.0.0]

* Any

Use Cases

Tablelayout

Version History

Version	Date	Content

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Tablechildren.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/layout/Tablechildren.html#>

Vbox

Vbox

- Demonstration: Vbox ^[1]
- Java API: Vbox ^[1]
- JavaScript API: Box ^[3]
- Style Guide: Vbox

Employment/Purpose

The vbox component is used to create a vertically oriented box. Added components will be placed underneath each other in a column.

Notice that hbox and vbox are designed to provide more sophisticated layout, such as splitter, alignment and packing. If you need only the layout feature, it is suggested to use HLayout and VLayout instead, since the performance is much better (due to the use of HTML DIV instead of TABLE).

Example



```
<zk>
    <vbox>
        <button label="Button 1"/>
        <button label="Button 2"/>
    </vbox>
    <hbox>
        <button label="Button 3"/>
        <button label="Button 4"/>
    </hbox>
</zk>
```

Properties

- Inherited Properties: Box

Align and Pack

pack / align		stretch	start	center	end
	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
start	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
center	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
end	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3

```

<zk xmlns:n="native">
  <style content=".box {> class='wikitable' | width='100%'">
    <custom-attributes>
      packs="${['', 'start', 'center', 'end']}"
      aligns="${['', 'stretch', 'start', 'center', 'end']}"
    </custom-attributes>
  </style>

  <vlayout>
    <hlayout height="70px" width="600px">
      <div hflex="1" vflex="1" sclass="box">
        <n:h3>pack / align</n:h3>
      </div>
      <div forEach="${aligns}" hflex="1" vflex="1" sclass="box">
        <n:h3>${each}</n:h3>
      </div>
    </hlayout>
    <hlayout forEach="${packs}" height="150px" width="600px">
      <custom-attributes pack="${each}" />
      <div hflex="1" vflex="1" sclass="box">
        <n:h3>${pack}</n:h3>
      </div>
    </hlayout>
  </vlayout>
</zk>

```

```
<vbox forEach="${aligns}" align="${each}" pack="${pack}" hflex="1" vflex="1" sclass="box">
    <button label="1" />
    <button label="2" />
    <button label="3" />
</vbox>
</hlayout>
</vlayout>
</zk>
```

[Since 5.0.0]

Cell Component

In ZK5, we have introduced a new component named Cell which can be embedded into a Grid or Box (Hbox and Vbox) to fully control the layout and the style. You can now use the rowspan or the colspan property to layout your Grid, for example a content cell can now cross over multiple rows. The code below demonstrates how to do this:

```
<vbox>
    <cell sclass="years">
        ...
    </cell>
</vbox>
```

[Since 5.0.0]

Limitation

Box component is consisted by Table element. Therefore, when put Input element like Textbox, Combobox inside Box component, specify width and height to Box component will be ignored when browser try to render table element.

For example,

```
<hbox height="200px" width="200px" style="border: 1px solid red">
    <textbox hflex="1" value="1" />
    <textbox hflex="1" value="1" />
</hbox>
```

You will see the Box width exceed 200px. Also check the sample ^[4] with pure HTML in jsfiddle.

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: Box

Supported Children

*ALL

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Vbox.html#>

Vlayout

Vlayout

- Demonstration: N/A
- Java API: Vlayout ^[1]
- JavaScript API: Vlayout ^[2]

Employment/Purpose

The vlayout component is a simple vertical oriented layout. Added components will be placed underneath each other in a column.

Notice that hlayout and vlayout do not support splitter, alignment and packing. If you need them, please use ZK Component Reference/Layouts/Hbox and ZK Component Reference/Layouts/Vbox instead.

Example



```
<zk>
    <vlayout>
        <button label="Button 1"/>
        <button label="Button 2"/>
    </vlayout>
    <hlayout>
        <button label="Button 3"/>
        <button label="Button 4"/>
    </hlayout>
</zk>
```

Spacing

The default spacing between two child components is 0.3em. You could modify it if you like:

```
<hlayout spacing="0">
    <textbox/>
    <button label="Click Me"/>
</hlayout>
```

Resize Child Components' Height Dynamically

When a Vlayout's content changes (e.g. adding / removing components or component's visibility changes), it will resize all its child components' height dynamically.

The window's height below (line 6) will grow when we hide the blue-background div. This also works for vflex="min" which doesn't auto resize in a normal case.

```
<vlayout height="400px" style="border: solid 1px">
    <button onClick="div.setVisible(false)" label="hide the blue box below"/>
    <div style="height: 400px; background-color: lightblue" id="div">
        box
    </div>
    <window border="normal" vflex="1">
        0px height at first
    </window>
</vlayout>
```

Supported events

Name	Event Type
None	None

Supported Children

*ALL

Use cases

Version	Description	Example Location

Version History

Version	Date	Content
5.0.4	August, 2010	new added component

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Vlayout.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/layout/Vlayout.html#>

Multimedia and Miscellaneous

This section outlines components which bring further multimedia capabilities to ZK.

Audio

Audio

- Demonstration: Audio ^[1]
- Java API: Audio ^[2]
- JavaScript API: Audio ^[3]

Employment/Purpose

An `audio` component is used to play the audio at the browser. Like `image`, you could use the `src` property to specify an URL of an audio resource, or the `setContent` method to specify a dynamically generated audio. Developers might be able to control the play of an audio by the `play`, `stop` and `pause` methods.

Example



```
<audio src="music.wav" controls="true"></audio>
```

The `audio` supports `controls` property since 7.0.0

Supports HTML5

The `audio` component has now been enhanced to support HTML 5, it includes the properties like `autoplay`, `controls`, `loop`, `muted` and `preload`.

Multiple Sources

Most browsers do not support all the audio formats, so we could specify multiple source files in different formats for different browsers. For examples:

```
<audio src="music.wav, music.mp3, music.ogg" controls="true"></audio>
```

StateChangeEvent

When you call `play()`, `stop()`, `pause()` or the audio is played to the end, an `StateChangeEvent` will be fired. You can check the current state by calling `event.getState()`. There are 4 states: `Audio.PLAY`, `Audio.STOP`, `Audio.PAUSE` and `Audio.END`.

For example:

If you want to do something after the audio starts to play, you can write codes as shown below (MVVM pattern).

```
<audio onStateChange="@command('stateChange', event=event)" />

@Command
public void stateChange(@BindingParam("event") StateChangeEvent event) {
    if (event.getState() == Audio.PLAY) {
        // do something...
    }
}
```

Supported Events

Name	Event Type
onStateChange	Event: StateChangeEvent ^[4] Notifies when invoking play(), stop(), pause() or the audio is played to the end.

- Inherited Supported Events: XulElement

Supported Children

* Track

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
7.0.0	August, 2013	Audio ^[2] now supports HTML 5
9.5.0	September 2020	ZK-4648 ^[5] : Audio supports to add tracks
9.6.0	May 2021	ZK-4779 ^[6] : audio supports to fire an event upon its playing state

References

- [1] <http://www.zkoss.org/zksandbox/userguide/#u5>
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Audio.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/med/Audio.html#>
- [4] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/StateChangeEvent.html#>
- [5] <https://tracker.zkoss.org/browse/ZK-4648>
- [6] <https://tracker.zkoss.org/browse/ZK-4779>

Barcode

Barcode

- Demonstration: Baicode ^[1]
- Java API: Barcode ^[1]
- JavaScript API: Barcode ^[2]
- Available for ZK:
- CE PE EE

Employment/Purpose

A barcode component is used to generate a barcode at the browser, and decode the barcode on the server side. There are many properties to dealing with the barcode component. First, you could use the `type` property to specify the type of barcode, for example: qrcode, code128, code 128A...., after choose the type you want, you could generate the barcode of the code. And every barcode image could response to a certain value, the property `value` is for barcode image which wants to stand for.

Example



```
<barcode type="qr" value="https://www.zkoss.org/" height="100px"/>
```



<https://www.zkoss.org/>

```
<barcode type="code128" value="https://www.zkoss.org/" height="100px"/>
```

type

The Barcode has 18 types for 1D and 1 type for 2D by default. After choosing the type of barcode, and you can render the barcode as the type you choose.

```
<barcode type="qr"/>
```

Note: the type that barcode supported can be referenced as Supported Barcode Type (Default).

value

value is a string attribute that barcode component want to render.

```
<barcode type="qr" value="https://www.zkoss.org"/>
```

Note: Some of the type only support a specific format of value.

height

Height is a string attribute to define the height of barcode component.

```
<barcode type="qr" value="https://www.zkoss.org" height="100px"/>
```

Note:

- (1) height is a string as a format of "number+px"
- (2) In type of qr, the width is as same as the height.

displayValue

The displayValue is the boolean attribute to decide whether the show the value under the barcode or not.

```
<barcode type="code128" displayValue="true"/>
```

Note: This displayValue attribute is only implement for 1D format.

fontSize

The fontSize is a integer attribute that decides the text font size under the 1D barcode.

```
<barcode type="code128" barWidth="2"/>
```

Note: This attribute is only implement for 1D format.

barWidth

The barWidth is a integer attribute that deciding the single bar width of barcode image.

```
<barcode type="code128" barWidth="2"/>
```

Note: This attribute is only implement for 1D format.

registerLibrary

registerLibrary(function () {}, library_name, [array of types]) is a client-side, class-level method to register a custom library into barcode widget. If you register the own custom library into Barcode, every single widget can use the custom library. The way to registering is zkmax.barcode.Barcode.registerLibrary(...). For example: you can register the library as the <script> below.

```
<?script src="mybarcodeLibrary.js"?>
<script>

...
zk.afterLoad('zkmax.barcode', function () {
    zkmax.barcode.Barcode.registerLibrary(function mybarcode(wgt) {
        ...
    });
}, 'library-name', ['type1', 'type2', ...]);
});
```

```
...  
</script>
```

Note:

(1) registerLibrary is the pure-client-side method.

(2) If you want to register the custom library for all the web application, you can add <javascript src="mybarcodeLibrary.js"/>the source file at WEB-INF/lang-addon.xml.

myRegister.js

```
zk.afterLoad('zkmax.barcode', function () {  
    zkmax.barcode.Barcode.registerLibrary(function mybarcode(wgt) {  
        ...  
        };  
        }, 'library-name', ['type1', 'type2', ...]);  
});
```

WEB-INF/lang-addon.xml

```
<javascript src="mybarcodeLibrary.js">  
<javascript src="myRegister.js">
```

Supported Barcode Type (Default)

Name	Barcode Type
1D CODE Family	CODE39, CODE128, CODE128A, CODE128B, CODE128C,
1D EAN Family	EAN13, EAN8, EAN5, EAN2,
1D MSI Family	MSI, MSI10, MSI11, MSI1010, MSI1110,
1D Others	UPC, ITF14, ITF, PHARMACODE, CODABAR
2D	QR

Supported Children

*NONE

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content
8.6.0	Apirl, 2018	Barcode [3]

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Barcode.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/med/Barcode.html#>
- [3] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/barcode/Barcode.html#>

BarcodeScanner

BarcodeScanner

- Demonstration: [BarcodeScanner](#) [1]
- Java API: [BarcodeScanner](#) [1]
- JavaScript API: [BarcodeScanner](#) [2]
- Available for ZK:
- CE PE EE

Browser Support

- IE browsers are not supported
- For iOS Safari only supports 11+

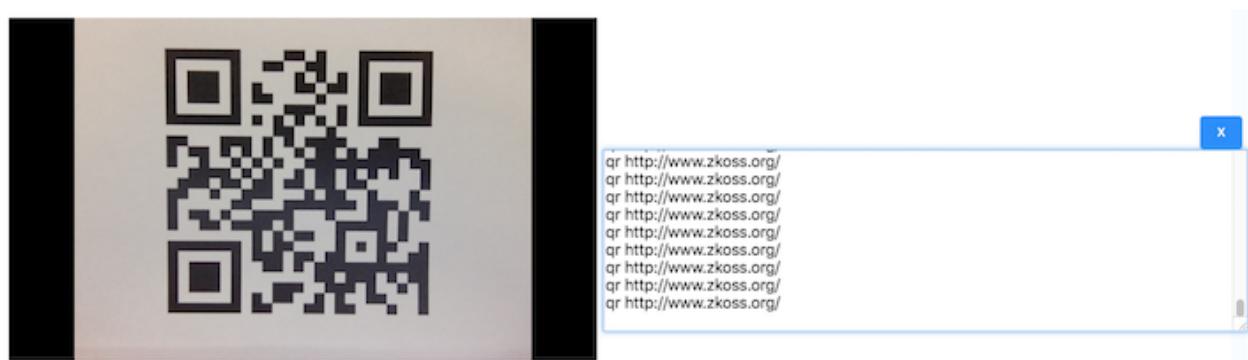
Note: iOS Chrome and other WebView browsers are not supported due to Apple's restriction.

Note: due to Chrome's security policy, starting from Chrome 47, getUserMedia() requests are only allowed from secure origins: HTTPS or localhost.

Employment/Purpose

A BarcodeScanner component is used to scan and decode the barcode on the client side. There are many properties to dealing with the barScan component. First, you could use the type property to specify the type of barcode, for example: qr, code128, ..., after choosing the type you want, you could scan the barcode by the scanner. You can choose the continue scanning by setting continuous="true", and setting the scan rate by setting interval="1000", the unit of interval is millisecond. You can turn on the scanner switch by setting enable="true", or close it by enable="false".

Example



```
<barcodeScanner type="qr,code128" continuous="true" interval="500" height="100px"  
onDetect='Clients.log(event.getType() + " " + event.getResult())' />
```

Scan the barcode by the Barcodescanner.

type

This component supports 9 types for 1D and 1 type for 2D barcode by default. After choosing the type of barcode, the component can scan the barcode of the type you choose. You also can scan multiple types at one time separated by a comma, for example

```
<barcodescanner type="qr,code128"/>
```

The supported types

- 1D CODE Family
 - CODE128
 - CODE39
 - CODE39VIN,
- in the 1D EAN Family
 - EAN (which includes EAN13)
 - EAN8,
- in the 1D Others Families
 - CODABAR
 - UPC
 - UPC_E
 - I2OF5
- 2D Family
 - QR

Note:

- (1) The type that Barcodescanner supported can be referenced as Supported Barcode Type (Default: CODE128).
- (2) Warn: if too many types are set for the widget, it may reduce the detecting accuracy.
- (3) The barcode will restart every time you change the type.

continuous

Continuous is a boolean attribute to let the Barcodescanner can interval scan or not.

```
<barcodescanner type="qr" continuous="true"/>
```

interval

The interval is a subsidiary, integer attribute for continuous scan. The interval="500" means the scanner will scan once every 500 millisecond.

```
<barcodescanner type="code128" continuous="true" interval="500"/>
```

enable

The enable is boolean attribute to switch the BarcodeScanner. You can use the zk mvvm mechanism to switch the BarcodeScanner.

```
<window viewModel="@id('vm')@init('xxxVM')">
    <barcodescanner type="code128" enable= "@bind('vm.enable')"/>
</window>
```

registerLibrary

registerLibrary(constructor, library_name, [array of types]) is a class-level javascript function to register a custom library into barcode widget. The mechanism is like the registerLibrary() in Barcode. The constructor is a json-format parameter to define the required setting for the library.

```
constructor =
{
    create: the function to create a reader prototype,
    name: the name of the reader, and we will mount the reader on the
    widget by wgt._'name'+'Reader',
    init: a function to initialize the reader properties,
    open: a function to enable the detecting function,
    decodeOnce: a reader decode method, and zk will implement the
    contScan attribute for you,
    setType: a function to connect the reader and widget types
function,
    other: a json properties to let you to attend the reader
    method. (future feature)
}
```

Like barcode, if you register your own custom library into Barcodescanner, every single widget should use the custom library object. The way to register is zkmax.barscanner.Barcodescanner.registerLibrary(...). Here is an example that how we inject the quagga library object into the widget:

```
<?script src="myBarcodescannerLibrary.js"?>
<script>
    ...
    zk.afterLoad('zkmax.barcode', function () {
        zkmax.barscanner.Barcodescanner.registerLibrary({
            create: jq.extend(true, {}, Quagga), //the reader
            name: "QUAGGA",
            init: zkmax.barscanner.Barcodescanner._quaggaInit,
//@param(wgt, video, canvas)
            open:
        zkmax.barscanner.Barcodescanner._quaggaOpen, //@param(wgt, video,
        canvas)
            decode:
        zkmax.barscanner.Barcodescanner._quaggaDecode, //@param(wgt, reader)
            setTypes: zkmax.barscanner.Barcodescanner._quaggaSetType,
//@param()
    
```

```
        other: null
    }, 'library-name', ['type1', 'type2', ...]);
};

...
</script>
```

And, you need to implement some methods to build the reader.

```
<?script src="implementation.js"?>
<script>

...
zkmax.barscanner.Barcodescanner._quaggaInit: function (wgt, video,
canvas) {
    /***the reader init method***/
    reader.onDetected() {
        reader.processing = false;
        .....
    }
}
zkmax.barscanner.Barcodescanner._quaggaOpen: function (wgt, video,
canvas) {
    /***the open method for quagga reader init method***/
}
zkmax.barscanner.Barcodescanner._quaggaDecode: function (wgt,
reader) {
    reader.processing = true;
    /***the decode method for quagga reader***/
}
zkmax.barscanner.Barcodescanner._quaggaSetType: function (wgt,
video, canvas) {
    /***the set type method for quagga reader***/
}
...
</script>
```

For most barcode readers, they would have some common points: init, open, decode, on detect... When injecting a reader into the widget, we use those common points to connect the reader and the widget. Here you have to implement some functions for this. As above, you need to implement the init, open, decode, and some other required methods. Here is a little reminder that, because different libraries have different designs for the decode and onDetect in their library, you may need to add a `reader.processing = true` at the beginning of decoding function, and `reader.processing = false` at the detected callback function by yourself. After you inject the library object into the widget, we will generate the JavaScript object mount on the widget. We generate zinit, zexecute, zopen, method onto the reader and we will use it internally. You can get the reader by `wgt._'your_library_name'Reader`.

Note:

As a barcode, If you want to register the custom library for all the web applications, you can add `<javascript src="myBarcodescannerLibrary.js"/>` at `WEB-INF/lang-addon.xml`.

`myRegister.js`

```

zk.afterLoad('zkmax.barcode', function () {
    zkmax.barscanner.Barcodescanner.registerLibrary(function
mybarcode(wgt) {
    ...
    };
    , 'library-name', ['type1', 'type2', ...]);
} );

```

Consistency Buffer (1D barcode only)

The accuracy of 1D barcode scan is not as good as 2D QR code. The scanned result might be wrong if the image quality is not good (such as blurred or dark). To prevent false positive results, we can set `consistencyBufferSize` and `consistencyThreshold`.

- Consistency buffer size: the number of past events to buffer (first in, first out)
- Consistency buffer threshold : the number of buffered events with the same value necessary to fire an event to the server and clear the buffer.

Defaults values

- Consistency buffer size: 5
- Consistency buffer threshold: 3

It means it requires at least 3 out of 5 scan events to be consistent before firing.

If you want to turn off this feature, simply set `consistencyBufferSize` and `consistencyThreshold` both 1.

Supported Barcode Type (Default)

Name	Barcodescanner Type
1D CODE Family	CODE128, CODE39, CODE39VIN,
1D EAN Family	EAN, EAN8,
1D Others	CODABAR, UPC, UPC_E, I2OF5
2D	QR

Supported Events

Name	Event Type
onDetect	Event: DetectEvent ^[3] Notifies if the barcode scanner detect a barcode message.

- Inherited Supported Events: XulElement

Supported Children

*NONE

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
8.6.0	May, 2018	ZK-3923: Provide a Barcode Scanner [4]
8.6.0	Oct 2018	ZK-4095: Add a false positive check threshold on the barcode scanner [5]

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/BarcodeScanner.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/barscanner/BarcodeScanner.html#>
- [3] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/event/DetectEvent.html#>
- [4] <http://tracker.zkoss.org/browse/ZK-3923>
- [5] <http://tracker.zkoss.org/browse/ZK-4095>

Cropper

Cropper

- Java API: Cropper ^[1]
- JavaScript API: Cropper ^[2]
- Available for ZK:
- CE PE EE

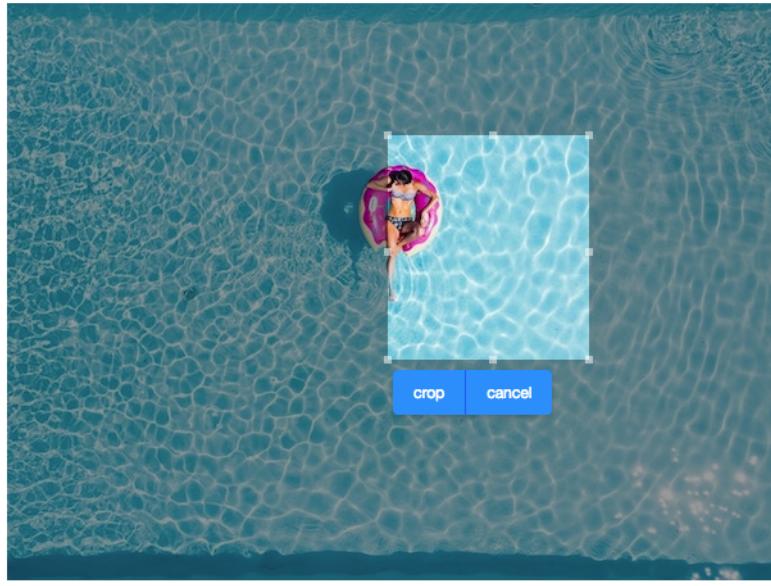
Browser Support

- This component supports IE10+ and modern browsers.

Employment/Purpose

This component allows users to crop a selected range of image.

Example



```
<cropper x="50" y="100" w="100" h="100" onCrop="img.setContent(event.getMedia())" width="800px"
toolbarVisible="true" src="swimming-pool.jpg"/>
<image id="img"/>
```

Properties and Features

Src

The src of the image.

Content

The content image.

AspectRatio

The width and height of the selected range will be fixed to the specified ratio.

MinWidth

The minimum width of the selected range.

MinHeight

The minimum height of the selected range.

MaxWidth

The maximum width of the selected range.

MaxHeight

The maximum height of the selected range

X

The left offset of the selected range.

Y

The top offset of the selected range.

W

The width of the selected range.

H

The height of the selected range.

ToolbarVisible

We provide a built in toolbar with Crop and Cancel feature.

CroppedFormat

Image formats like image/jpeg or image/png is allowed, Default is set to image/png

Supported Events

Name	Event Type
onChange	Event: Event [7] Denotes user has resized the selected range.
onChanging	Event: Event [7] Denotes user is resizing the selected range.
onCrop	Event: UploadEvent [5] Denotes user has cropped the image.

- Inherited Supported Events: HtmlBasedComponent

Supported Children

* Image

Version History

Version	Date	Content

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Cropper.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/med/Cropper.html#>

Camera

Camera

- Java API: Camera ^[1]
- JavaScript API: Camera ^[2]

[ZK EE]
[since 8.6.0]

Employment/Purpose

The Camera component is used to record videos and take snapshots in the browser. Developers can control the camera by start, stop, pause, resume and snapshot.

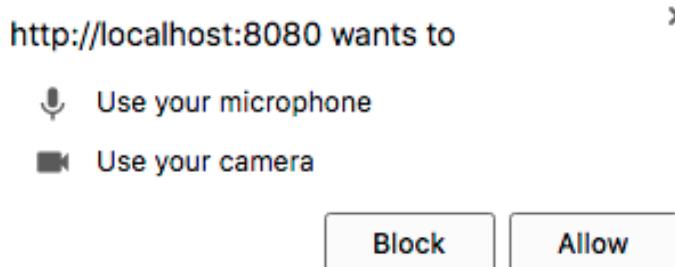
Browser Support

The Camera component currently supports Chrome and Firefox. This component is based on Navigator.getUserMedia() ^[3] and MediaRecorder ^[4], please check your browser compatibility ^[5] before using it.

Example

To record or take a snapshot, users should first enable their camera and microphone (as shown below.) If users reject the requests, they cannot use the features of the Camera component.

Note that some browsers such as Google Chrome will only accept webcam and microphone access from a trusted (https) source.



The Camera component also provides a method `requestCamera()` to request user's media devices before starting recording or taking snapshots.

```
<camera id="camera" />
<button onClick="camera.requestCamera()" />
```

recording

setRecording(true) is the same as invoking start() or resume(), depending on camera's current state.

setRecording(false) is the same as invoking stop(), if you want to pause recording, please invoke pause().

isRecording() can check whether the camera is recording or not.

```
<camera id="camera" />
<button onClick="camera.setRecording(true)" />
```

audio

You can decide whether to record audio while recording the video by specifying the value of audio; the valid value is boolean.

Default: true

```
<camera audio="false" />
```

previewRecord

The Camera component provides a preview screen to preview the recorded content.

When you turn on the preview screen, you can set the screen size by specifying width and height in 'px'.

If you wish to take a snapshot, you must turn on the preview screen first, or nothing will happen.

Default: true

```
<camera width="600px" previewRecord="true" />
```

maxsize

You can set the maximum size for uploading recorded video and snapshot. The unit is "KB". Negative value means unlimited, e,g,. maxsize="-1."

When the size of the recorded video or snapshot is bigger than the configured max size, nothing will be uploaded and it will trigger an event. To handle this, you can listen to onMaxsizeExceed and get the upload size from event.getData().

Default: please refer to max-upload-size^[6]

```
<camera maxsize="1024" onMaxsizeExceed="event.getData()" />
```

lengthLimit

You can set the maximum recording length in "seconds".

When the recorded time exceeds the max length, it will stop recording and trigger an event. To handle this, you can listen to onLengthLimitExceed.

Default: 60

```
<camera lengthLimit="120" onLengthLimitExceed="doSomething()" />
```

Customize recording hint

There are three classes that you can customize your recording hint, z-camera-recording, z-camera-stop and z-camera-pause.

Inspect DOM on the browser and you will see an html tag *<i>* before *<video>*; they are both in the same container *<div>*, these class names will apply to tag *<i>* depending on different states of the camera.

We have default CSS for different status, but you can also override these CSS easily.

```
<div id="xP5Q1" style="width:600px;" class="z-camera z-camera-preview">
  <i id="xP5Q1-recordingHint" class="z-camera-pause"></i> == $0
  <video id="xP5Q1-real" class="z-camera-real" autoplay data-video="0"></video>
</div>
```

For example:

If you want to notify users when the camera is paused, you can write css code like the following:



```
<style>
  .z-camera-pause {
    width: 30px;
    height: 30px;
    top: 30px;
    left: 30px;
    position: absolute;
    border-left: 10px solid red;
    border-right: 10px solid red;
  }
</style>
```

ConstraintsString

```
<camera constraintsString='{"video": { "facingMode": { "exact": "user" } } }'>
```

Please refer to <https://developer.mozilla.org/en-US/docs/Web/API/MediaTrackConstraints>

UploadEvent

There are two types of UploadEvent to listen to data upload: one is `onVideoUpload` and the other is `onSnapshotUpload`. Both ways can receive uploaded data by calling `event.getMedia()`.

`onVideoUpload` will be notified after calling `stop()`, or when the recorded time exceeds the maximum length.

`onSnapshotUpload` will be notified after calling `snapshot()`.

You can easily integrate the Camera component with Video ^[7] and Image ^[3].

For example:

To integrate these components, you can write codes as shown below, and after the video or snapshot is uploaded, you can see the results immediately.

```
<camera onVideoUpload='video.setContent(event.getMedia())'  
       onSnapshotUpload='image.setContent(event.getMedia())' />  
<video id="video" />  
<image id="image" />
```

StateChangeEvent

When you call `start()`, `stop()`, `pause()` or `resume()`, the Camera component will trigger `StateChangeEvent`. You can check the current state by calling `event.getState()`. The Camera component has four states, and you can access them by using `Camera.START`, `Camera.STOP`, `Camera.PAUSE` and `Camera.RESUME`.

For example:

If you want to do something after the recording starts, you can write codes as shown below (MVVM style).

```
<camera onStateChange="@command('stateChange', event=event)" />  
  
@Command  
public void stateChange(@BindingParam("event") StateChangeEvent event) {  
    if (event.getState() == Camera.START) {  
        // do something...  
    }  
}
```

The Camera component also provides `isRecording()`, `isPaused()` and `isStopped()` methods to check camera state.

Supported Events

Name	Event Type
onVideoUpload	Event: UploadEvent ^[5] Notifies after the video has been uploaded.
onSnapshotUpload	Event: UploadEvent ^[5] Notifies after the snapshot has been uploaded.
onMaxsizeExceed	Event: Event ^[7] Notifies if the recorded size is bigger than the max size.
onLengthLimitExceed	Event: Event ^[7] Notifies if the recorded length exceeds the max length.
onStateChange	Event: StateChangeEvent ^[8] Notifies when invoking start(), stop(), pause() or resume().
onCameraUnavailable	Event: DOMExceptionEvent ^[9] Notifies if camera is unavailable after requesting media devices from user.

- Inherited Supported Events: XulElement

Supported Children

NONE

References

- [1] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/ Camera.html#](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/	Camera.html#)
- [2] [http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/med/ Camera.html#](http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/med/	Camera.html#)
- [3] <https://developer.mozilla.org/en-US/docs/Web/API/Navigator/getUserMedia>
- [4] <https://developer.mozilla.org/en-US/docs/Web/API/MediaRecorder>
- [5] <https://caniuse.com/#feat=mediarecorder>
- [6] https://www.zkoss.org/wiki/ZK_Configuration_Reference/zk.xml/The_system-config_Element/The_max-upload-size_Element
- [7] https://www.zkoss.org/wiki/ZK_Component_Reference/Multimedia_and_Miscellaneous/Video
- [8] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/event/StateChangedEvent.html#>
- [9] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/DOMExceptionEvent.html#>

Flash

Flash

- Demonstration: Flash [1]
- Java API: Flash [2]
- JavaScript API: Flash [3]

Employment/Purpose

A generic Flash component.

Example



```
<flash src="http://www.zkoss.org/Steps/learn.swf" height="300" width="800" />
```

Note: The appearance depends on the content of your Flash.

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: HtmlBasedComponent

Supported Children

*NONE

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/zkdemo/multimedia/flash>
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Flash.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/med/Flash.html#>

Pdfviewer

Pdfviewer

- Demonstration: Embed PDF Documents in Your ZK Application ^[1]
- Java API: Pdfviewer ^[2]
- JavaScript API: Pdfviewer ^[3]
- Available for ZK:
-  **CE** **PE** **EE**

Employment/Purpose

The Pdfviewer component is based on Mozilla's great work -- PDF.js^[4], that renders PDF documents in a browser.

Example

Component Based UI 4

which would render a window containing two buttons as shown in the image below:

ZK Essentials

Hello Good-bye

The markup in ZUL is equivalent to the following POJO declarations in Java:

```
Window win = new Window();
    win.setTitle("ZK Essentials");
    win.setBorder("normal");
    win.setWidth("250px");

    Button helloBtn = new Button();
        helloBtn.setLabel("Hello");
        helloBtn.setParent(win);

    Button byeBtn = new Button();
        byeBtn.setLabel("Good-bye");
        byeBtn.setParent(win);
```

C ⌂ | << < > >> 7 / 62 - + 100% ▾ ↗

<pdfviewer src="/pdf/sample.pdf" />

Supported Browsers

It is compatible with HTML5-supported browsers, like IE 11, Edge, Firefox, Opera, Chrome and Safari.

Customize the Toolbar

Pdfviewer accepts only one child: <toolbar>. You can customize the toolbar by adding your own toolbar. By default, the position of the toolbar is at the top. You can use CSS to do more tweaks.

```
<pdfviewer id="pv2">
    <toolbar>
        <toolbarbutton iconScss="z-icon-fw z-icon-fast-backward"
            onClick="pv2.firstPage()"/>
        <toolbarbutton iconScss="z-icon-fw z-icon-chevron-left"
            onClick="pv2.previousPage()"/>
        <toolbarbutton iconScss="z-icon-fw z-icon-chevron-right"
            onClick="pv2.nextPage()"/>
        <toolbarbutton iconScss="z-icon-fw z-icon-fast-forward"
            onClick="pv2.lastPage()"/>
    </toolbar>
</pdfviewer>
```

Zoom to Fit Page Width / Fit Page Height

You can call `setZoom("fit-page-width")` or `setZoom("fit-page-height")` now.

Or just specify it as an initial zoom level:

```
<pdfviewer id="pv" src="/pdf/sample.pdf" zoom="fit-page-width" />
```

Cross-Origin Resource Sharing (CORS)

If you want to load a remote PDF document, please make sure the response headers contain the necessary CORS headers^[5] or it won't be allowed to be loaded.

Enable Fullscreen

Due to a specification limitation, this method must be called while responding to a user interaction (i.g. event handlers). Therefore the component only provides a client-side method `toggleFullscreen()` to enable the full screen mode.

```
<pdfviewer id="pv" src="/pdf/sample.pdf" />
<button xmlns:w="client" w:onClick="zk.$('$pv').toggleFullscreen()" />
```

Supported Events

Name	Event Type
onPaging	Event: PagingEvent ^[6] Notifies one of the pages is selected by the user.
onRender	Event: Event ^[7] Denotes the loading pdf file is finished rendering.
onZoom	Event: ZoomEvent ^[7] Denotes user has changed the zoom level.
onRotate	Event: RotationEvent ^[8] Denotes user has changed the rotation angle.

- Inherited Supported Events: XulElement

Supported Children

*Toolbar

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content
9.0.0	October 2019	ZK-4395 [9]: Provide a pdfviewer component
9.6.0	June 2021	ZK-4917 [10]: set zoom level with fit-page-width or fit-page-height in Java

References

- [1] <https://blog.zkoss.org/2019/10/02/zk-9-preview-embed-pdf-documents-in-your-zk-application/>
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkex/zul/Pdfviewer.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zkex/pdfviewer/Pdfviewer.html#>
- [4] <https://github.com/mozilla/pdf.js>
- [5] <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>
- [6] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/event/PagingEvent.html#>
- [7] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkex/ui/event/ZoomEvent.html#>
- [8] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkex/ui/event/RotateEvent.html#>
- [9] <https://tracker.zkoss.org/browse/ZK-4395>
- [10] <https://tracker.zkoss.org/browse/ZK-4917>

Video

Video

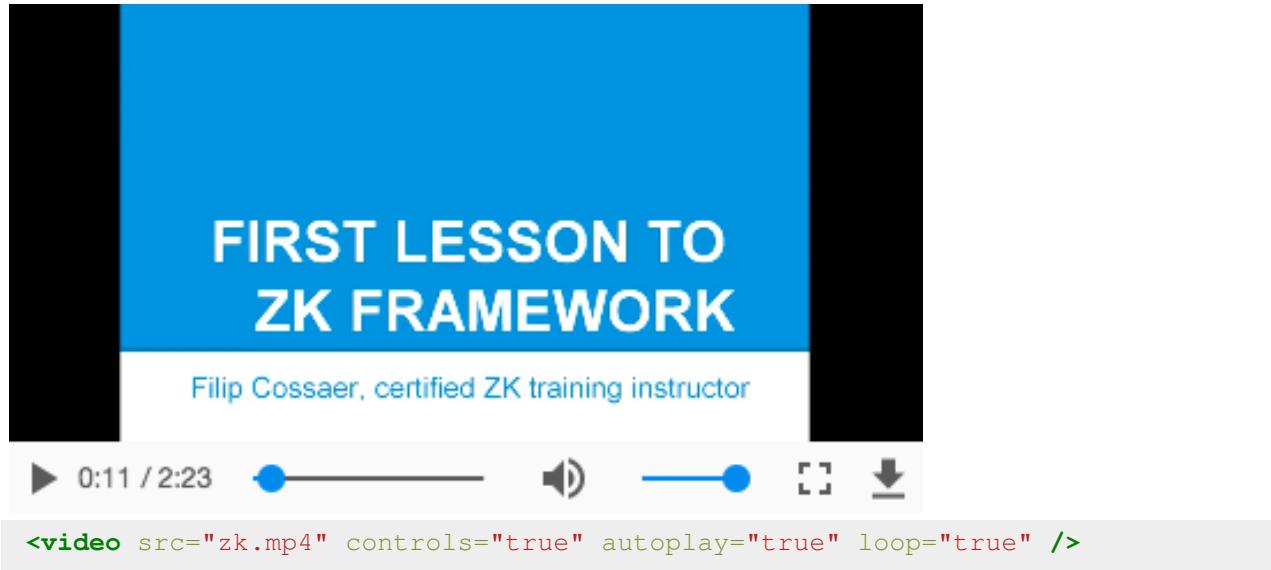
- Java API: Video ^[1]
- JavaScript API: Video ^[2]
- Available for ZK:
- EE

since 8.6.0

Employment/Purpose

A `Video` component is used to play the video in the browser. Like `audio`, you can either use the `src` property to specify an URL of the video resource, or use the `setContent` method to specify a dynamically generated video. Users can control the video by `play`, `stop` and `pause`.

Example



```
<video src="zk.mp4" controls="true" autoplay="true" loop="true" />
```

Supports HTML5

The `Video` component is based on HTML 5's `<video>` tag, and supports the following properties: `src`, `autoplay`, `controls`, `loop`, `playbackRate`, `dimBackground`, `preload`, `clipToFit`, `poster`, `playsinline` and `crossorigin`.

Supported Formats

mp4, WebM, ogg^[3]

Autoplay policy is different between browser

Please refer to [autoplay-policy-changes](#)^[4] and [auto-play-policy-changes-for-macos](#)^[5]

Multiple Sources

Most browsers do not support all the video formats, so you can specify multiple source files in different formats for different browsers. If the first format is not supported by the browser, it will fallback to the 2nd format. For example:

```
<video src="zk.mp4, zk.webm, zk.ogg" />
```

enableFullScreen

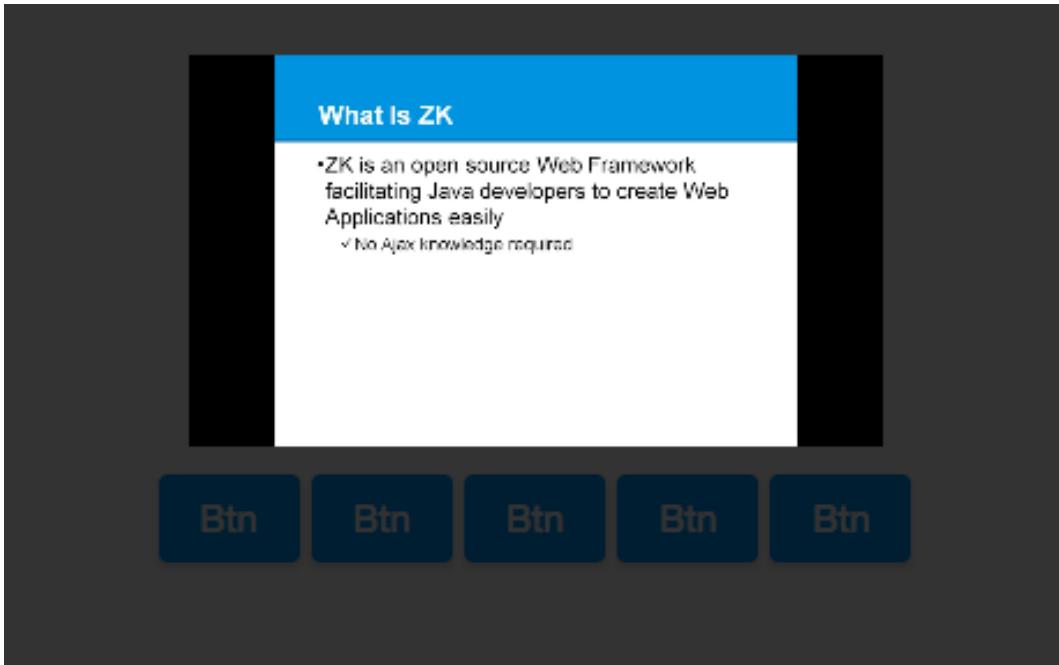
For security reasons, `fullScreen` API can only be initiated by an user gesture. Therefore the `Video` component only provides a client-side method `enableFullScreen()` to enable the full screen mode.

```
<video id="player" src="zk.mp4" controls="true"/>
<button xmlns:w="client" w:onClick="zk.$('player').enableFullScreen()" />
```

dimBackground

The Video component provides a theater mode, If dimBackground="true", the whole page will be covered by translucent black by default except the Video.

When the theater mode is enabled, user can click anywhere on the page outside the Video to disable theater mode and return to the normal view.



```
<video src="zk.mp4" dimBackground="true" />
```

By default, css of dimBackground has two properties as shown in the following css code.

You can also customize the background in your preference by simply overriding .z-video-dim-background in css.

```
<style>
.z-video-dim-background {
    background: black;
    opacity: 0.8;
}
</style>
```

playbackRate

The Video component provides `setPlaybackRate(double)` to control the video playing speed. The valid value depends on the displayed browser.

Default: 1.0

```
<video src="zk.mp4" playbackRate="0.5" />
```

currentTime

The Video component provides `setCurrentTime(double)` to jump to the specified time-point (in seconds) of the playback video.

```
<video src="zk.mp4" currentTime="60" />
```

playing

The Video component provides `setPlaying(boolean)` to play or pause the video.

`playing="true"` is same as invoking `play()`; `playing="false"` is same as invoking `pause()`.

```
<video src="zk.mp4" playing="false" />
```

volume

The Video component provides `setVolume(double)` to change the volume. The value should range between 0.0 and 1.0.

Default: 1.0

```
<video src="zk.mp4" volume="0.5" />
```

muted

The Video component provides `setMuted(boolean)` to mute the video.

Default: false

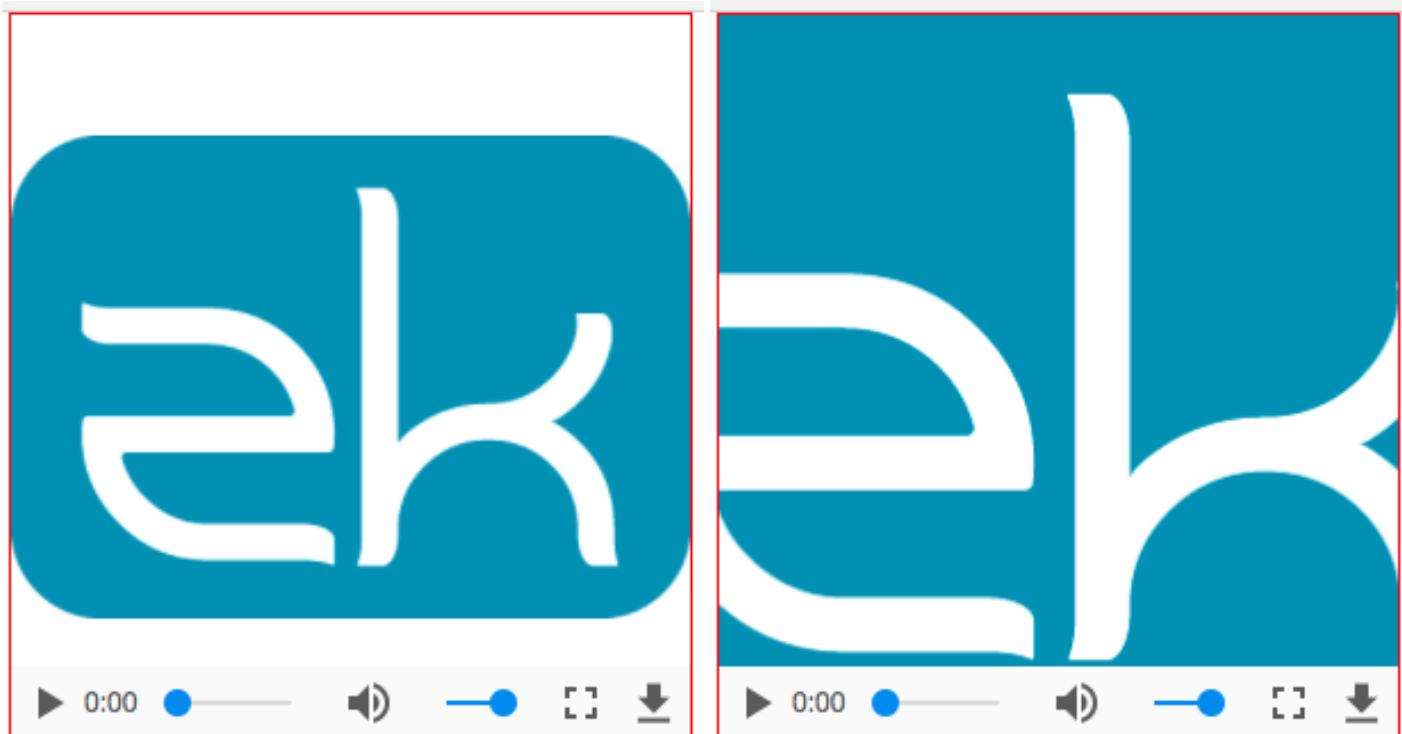
```
<video src="zk.mp4" muted="true" />
```

clipToFit

The Video component provides `setClipToFit(boolean)` to clip the video when the source size doesn't fit the size specified in the Video tag.

For example:

The source image used in the sample below is 450 * 320. When you set `width="300px", height="320px"`, by default, blank space will be inserted above and below the video to preserve the aspect ratio (left image); when you set `clipToFit="true"`, it will cut off the sides and fill up the space (right image).



```
<video width="300px" height="320px" src="zk.mp4" style="border: 1px solid red;" />  
<video width="300px" height="320px" src="zk.mp4" style="border: 1px solid red;" clipToFit="true" />
```

StateChangeEvent

When you call `play()`, `stop()`, `pause()` `StateChangeEvent` will be triggered. You can check the current state by calling `event.getState()`. Video has three states, and you can access them by using `Video.PLAY`, `Video.STOP` and `Video.PAUSE`.

For example:

If you want to do something after the video starts to play, you can write codes as shown below (MVVM style).

```
<video onStateChange="@command('stateChange', event=event)" />  
  
 @Command  
 public void stateChange(@BindingParam("event") StateChangeEvent event) {  
     if (event.getState() == Video.PLAY) {  
         // do something...  
     }  
 }
```

Video component also provides `isPlaying()`, `isPaused()` and `isStopped()` methods to check the video state.

since 9.6.0

Since ZK 9.6.0, a state - `Video.END` is added. When the video is played to the end, the `StateChangeEvent` will be triggered.

Supported Events

Name	Event Type
onStateChange	Event: StateChangeEvent [8] Notifies when invoking play(), stop() or pause().

- Inherited Supported Events: XulElement

Supported Children

* Track

Version History

Version	Date	Content
8.6.0	May 2018	ZK-3845 [6]: Provide a video component
9.5.0	September 2020	ZK-4649 [7]: Video supports to add tracks

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Video.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/med/Video.html#>
- [3] https://developer.mozilla.org/en-US/docs/Web/HTML/Supported_media_formats#File_formats
- [4] <https://developers.google.com/web/updates/2017/09/autoplay-policy-changes>
- [5] <https://webkit.org/blog/7734/auto-play-policy-changes-for-macos/>
- [6] <https://tracker.zkoss.org/browse/ZK-3845>
- [7] <https://tracker.zkoss.org/browse/ZK-4649>

Track

Track

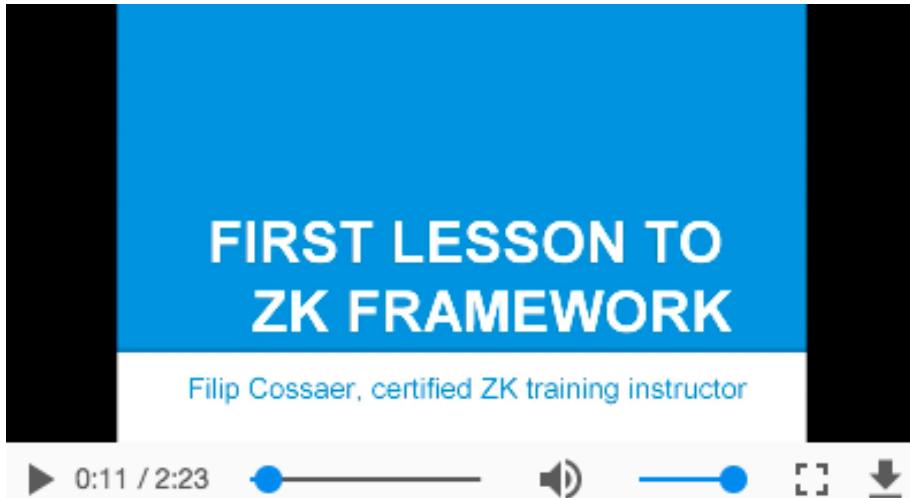
- Java API: Track ^[1]
- JavaScript API: Track ^[2]

[Since 9.5.0]

Employment/Purpose

It lets you specify some timed text tracks like captions or subtitles for media components such as Audio or Video.

Example



```
<video src="course.mp4" controls="true">
  <track kind="captions" src="transcript.vtt" srclang="en" default="true"/>
  <track kind="subtitles" src="transcript_fr.vtt" srclang="fr"/>
  <track kind="subtitles" src="transcript_de.vtt" srclang="de"/>
  <track kind="subtitles" src="transcript_zh.vtt" srclang="zh"/>
</video>
```



```
<audio src="music.wav" controls="true">
  <track kind="captions" src="music_lyric.vtt" srclang="en" default="true"/>
  <track kind="subtitles" src="music_lyric_fr.vtt" srclang="fr"/>
  <track kind="subtitles" src="music_lyric_de.vtt" srclang="de"/>
  <track kind="subtitles" src="music_lyric_zh.vtt" srclang="zh"/>
</audio>
```

Supported Browsers

It is compatible with HTML5-supported browsers, like IE 10, Edge, Firefox, Opera, Chrome, and Safari.

Tracks don't be supported in Internet Explorer 9.

Properties

Default

Specify if the track should be used by default. It must be used on one track only.

Kind

Specify what kind of track it is. Allowed values are:

Name	Description
subtitles	Closed subtitles.
captions	Closed captions.
descriptions	A textual description about what this video/audio is.
chapters	Chapter titles for users to navigate.
metadata	A track used by scripts.

Label

Specify a user-readable title. Used by browsers to show on the list of available tracks.

Src

The URL of the source file. The file must be in WebVTT format. This attribute is required.

Srclang

Specify what language this track is. It's required if the kind is `subtitles`. It must be a valid BCP 47 language tag. For instance, `fr-FR` and `en-US` are valid.

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

*NONE

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
9.5.0	September 2020	ZK-4648 [5]: Audio supports to add tracks ZK-4649 [7]: Video supports to add tracks

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Track.html#>

[2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/med/Track.html#>

Supplementary

This section outlines the components which supplement other ZK components.

Auxhead

Auxhead

- Demonstration: Grid (Merged Header) ^[1]
- Java API: Auxhead ^[2]
- JavaScript API: Auxhead ^[3]
- Style Guide: Auxhead

Employment/Purpose

Used to define a collection of auxiliary headers (Auxheader).

Example

H1'07						H2'07					
Q1			Q2			Q3			Q4		
Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1,000	1,100	1,200	1,300	1,400	1,500	1,600	1,700	1,800	1,900	2,000	2,100

```

<grid>
  <auxhead>
    <auxheader label="H1'07" colspan="6" />
    <auxheader label="H2'07" colspan="6" />
  </auxhead>
  <auxhead>
    <auxheader label="Q1" colspan="3" />
    <auxheader label="Q2" colspan="3" />
    <auxheader label="Q3" colspan="3" />
    <auxheader label="Q4" colspan="3" />
  </auxhead>
  <columns>
    <column label="Jan" />
    <column label="Feb" />
    <column label="Mar" />
    <column label="Apr" />
    <column label="May" />
    <column label="Jun" />
    <column label="Jul" />
    <column label="Aug" />
    <column label="Sep" />
    <column label="Oct" />
    <column label="Nov" />
    <column label="Dec" />
  </columns>
  <rows>
    <row>
      <label value="1,000" />
      <label value="1,100" />
      <label value="1,200" />
      <label value="1,300" />
      <label value="1,400" />
      <label value="1,500" />
      <label value="1,600" />
      <label value="1,700" />
      <label value="1,800" />
      <label value="1,900" />
      <label value="2,000" />
      <label value="2,100" />
    </row>
  </rows>

```

```
</rows>  
</grid>
```

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

* Auxheader

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

- [1] http://www.zkoss.org/zkdemo/grid/merged_header
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Auxhead.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/mesh/Auxhead.html#>

Auxheader

Auxheader

- Demonstration: Grid (Merged Header) ^[1]
- Java API: Auxheader ^[1]
- JavaScript API: Auxheader ^[2]
- Style Guide: Auxhead

Employment/Purpose

The auxiliary headers support the colspan and rowspan properties which allows itself to be spanned across several columns/rows. Auxiliary headers should be accompanied with columns/listhead/treecols when used with grid/listbox/tree.

Example

An auxiliary header.

H1'07						H2'07					
Q1			Q2			Q3			Q4		
Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1,000	1,100	1,200	1,300	1,400	1,500	1,600	1,700	1,800	1,900	2,000	2,100

```
<grid>
    <auxhead>
        <auxheader label="H1'07" colspan="6" />
        <auxheader label="H2'07" colspan="6" />
    </auxhead>
    <auxhead>
        <auxheader label="Q1" colspan="3" />
        <auxheader label="Q2" colspan="3" />
        <auxheader label="Q3" colspan="3" />
        <auxheader label="Q4" colspan="3" />
    </auxhead>
    <columns>
        <column label="Jan" />
        <column label="Feb" />
        <column label="Mar" />
        <column label="Apr" />
        <column label="May" />
        <column label="Jun" />
        <column label="Jul" />
        <column label="Aug" />
        <column label="Sep" />
        <column label="Oct" />
        <column label="Nov" />
```

```
<column label="Dec" />
</columns>
<rows>
  <row>
    <label value="1,000" />
    <label value="1,100" />
    <label value="1,200" />
    <label value="1,300" />
    <label value="1,400" />
    <label value="1,500" />
    <label value="1,600" />
    <label value="1,700" />
    <label value="1,800" />
    <label value="1,900" />
    <label value="2,000" />
    <label value="2,100" />
  </row>
</rows>
</grid>
```

The Limitation of rowspan

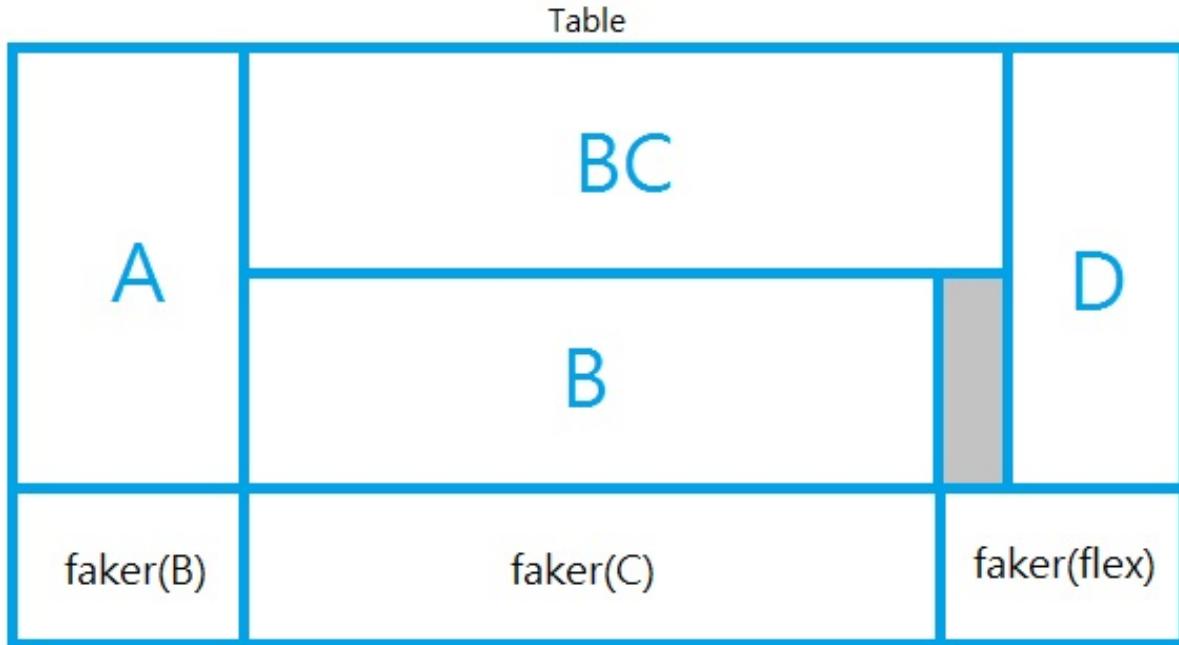
For better performance, every instance of Column will create an invisible HTML TH element called *faker*. However, with some complex combination of rowspan and colspan, Grid might not be able to generate the correct number of *faker* to represent each column.

For example, it is wrong if the number of the column components are not the same as the number of columns in each row as shown below:

```
<grid width="200px">
  <auxhead>
    <auxheader label="A" rowspan="2" />
    <auxheader label="BC" colspan="2" />
    <auxheader label="D" rowspan="2" />
  </auxhead>
  <columns><!-- this is wrong since the number of column components is smaller -->
    <column label="B"/>
    <column label="C"/>
  </columns>
  <rows>
    <row>
      <label forEach="E,F,G,H" value="${each}" /><!-- four columns -->
    </row>
  </rows>
</grid>
```

A	BC	D
B		
E	F	H

As shown above, the column with label C will be invisible, because the fakers are not created correctly. Here is the result but wrong DOM structure:



There is a simple workaround: specify all columns. If you don't want to show all columns, you could use Auxheader instead of Column, and then add an empty Columns. For example, the code in the previous example can be fixed as follows:

```

<grid width="200px">
  <auxhead>
    <auxheader label="A" rowspan="2" />
    <auxheader label="BC" colspan="2" />
    <auxheader label="D" rowspan="2" />
  </auxhead>
  <auxhead>
    <auxheader label="B"/>
    <auxheader label="C"/>
  </auxhead>
  <columns/> <!-- use an empty columns to make fakers created correctly --&gt;
  &lt;rows&gt;
    &lt;row&gt;
      &lt;label forEach="E,F,G,H" value="${each}" /&gt;
    &lt;/row&gt;
  &lt;/rows&gt;
&lt;/grid&gt;
</pre>

```

A	BC		D
	B	C	
E	F	G	H

The other limitation is that the width of the Auxheader component depend on the Column component. Thus, if you'd like to specify the width in the Column component, it means it will take some space even when there are no label in all Column components. The workaround is simple: make the empty Columns component invisible. For example,

```
<grid width="350px">
  <auxhead>
    <auxheader label="A" rowspan="2" />
    <auxheader label="BC" colspan="2" />
    <auxheader label="D" rowspan="2" />
  </auxhead>
  <auxhead>
    <auxheader label="B"/>
    <auxheader label="C"/>
  </auxhead>
  <columns visible="false"><!-- make it invisible -->
    <column width="100px"/><!-- specify width here -->
    <column width="150px"/>
    <column width="50px"/>
    <column width="50px"/>
  </columns>
  <rows>
    <row>
      <label forEach="E,F,G,H" value="${each}" />
    </row>
  </rows>
</grid>
```

A	BC		D
	B	C	
E	F	G	H

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

*ALL

Restrictions

[Since ZK 8.0.0]

Noticed that it's forbidden to set width/height/hflex/vflex in Auxheader.

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Auxheader.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/mesh/Auxheader.html#>

Cell

Cell

- Demonstration: Grid (Spreadsheet Functionalities) ^[1]
- Java API: Cell ^[2]
- JavaScript API: Cell ^[3]
- Style Guide: Cell

Employment/Purpose

We design this generic cell component to be embedded into Row or Hbox or Vbox in order to fully control the alignment and row/column span.

Example

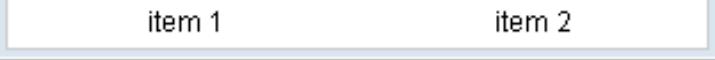
A	B	C	D
item 1	item 2	item 3	item 4
	item 5	item 6	item 7
	item 8	item 9	item 10

```
<zk>
    <grid>
        <columns>
            <column label="A" />
            <column label="B" />
            <column label="C" />
            <column label="D" />
        </columns>
        <rows>
            <row>
                <cell rowspan="4" align="center" valign="bottom">
                    <label value="item 1" />
                </cell>
                <cell colspan="3">
                    <label value="item 2" />
                </cell>
            </row>
            <row>
                <cell colspan="2" align="center">
                    <label value="item 3" />
                </cell>
                <label value="item 4" />
            </row>
            <row>
                <label value="item 5" />
            </row>
        </rows>
    </grid>
```

```

        <label value="item 6" />
        <label value="item 7" />
    </row>
    <row>
        <label value="item 8" />
        <label value="item 9" />
        <label value="item 10" />
    </row>
</rows>
</grid>
</zk>
```

hbox



item 1 item 2

```

<zk>
<window title="hbox" border="normal" width="320px">
    <hbox width="300px" pack="center">
        <cell hflex="1" align="center">
            <label value="item 1" />
        </cell>
        <cell hflex="1" align="center">
            <label value="item 2" />
        </cell>
    </hbox>
</window>
</zk>
```

Properties

The Rowspan Property

It specifies the number of rows this cell should occupy. It has the same effect as HTML TR tag's rowspan attribute does.

Miscellaneous

Comparison to default (no Cell) scenario

The Cell component is meant to provide full control of the DOM structure, so the user needs to expect to handle some lower level styling.

For example, consider the following scenario:

```

<grid>
    <columns>
        <column label="A" />
        <column label="B" />
```

```

</columns>
<rows>
  <row>
    <label>A</label>
    <cell>
      <label>B</label>
    </cell>
  </row>
</rows>
</grid>

```

Although they look alike, the DOM structures generated for the two table cells are slightly different:

```

<tbody id="eU1Y4" class="z-rows">
  <tr id="eU1Y5" class="z-row">
    <td id="eU1Y6-chdextr" class="z-row-inner">
      <div id="eU1Y6-cell" class="z-row-cnt z-overflow-hidden">
        <span id="eU1Y6" class="z-label">A</span>
      </div>
    </td>
    <td id="eU1Y7" class="z-cell">
      <span id="eU1Y8" class="z-label">B</span>
    </td>
  </tr>
</tbody>

```

With a Cell component given, there is no inner `<div>` element generated, which grants you a more flexible control to the DOM structure, but as a result you may need to provide more style handling. It is also recommended to use Cell for handling row span, column span, and alignment. In neither of the above cases, it is recommended not to use Cell. To put more than 1 Component in a grid cell, you can also use a Div to wrap them.

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

*ALL

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

- [1] http://www.zkoss.org/zkdemo/grid/spreadsheet_functionalities
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Cell.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/wgt/Cell.html#>

Coachmark

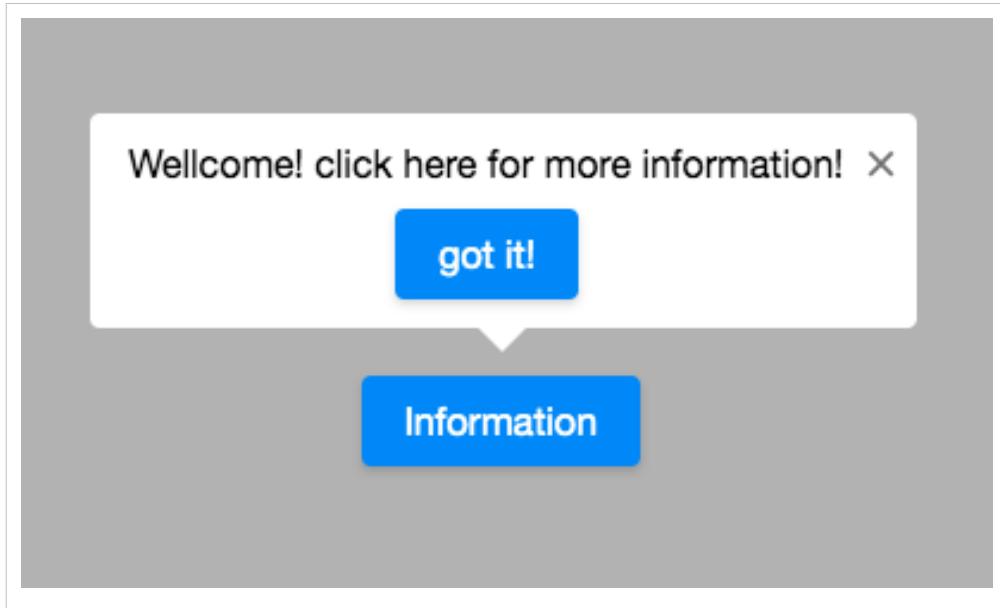
Coachmark

- Java API: Coachmark ^[1]
- JavaScript API: Coachmark ^[2]
- Available for ZK:
- 

Employment/Purpose

Coachmark is used to attract users' attention to the target component and display a dialog. Once a coachmark is opened, the background mask will be displayed and the target component will be highlighted. The content of coachmark should be as relevant as possible to the context. You can use more than one coachmarks to guide users to perform a series of operations in the desired order.

Example



```
<zk>
    <button id="button" label="Information"></button>
    <coachmark target="button" position="before_center">
        <label>Wellcome! click here for more information!</label>
        <button style="display: block; margin: 10px auto 0" label="got it!"></button>
    </coachmark>
</zk>
```

Supported Browsers

This component uses CSS keyframes. Browsers that support CSS keyframes (IE10+, Edge, Chrome, Firefox, Safari) are compatible with this feature.

Open / Close the Coachmark

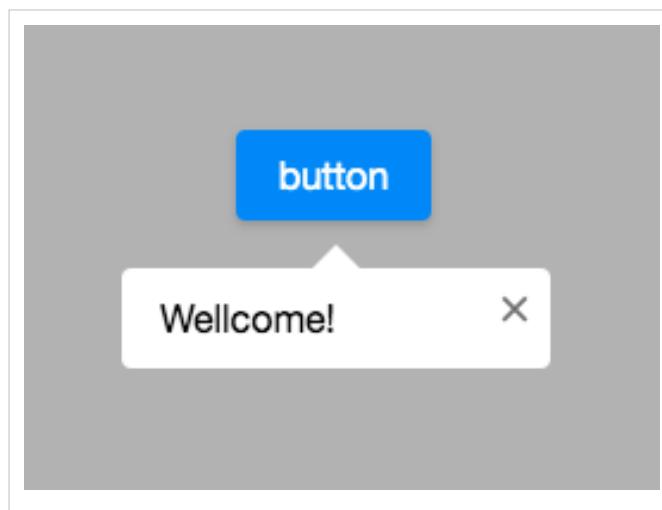
Both `visible` attribute and `open/close` methods allow you to open or close the Coachmark.

Properties

Target

The target component that the Coachmark will point itself to.

Example



```
<zk>
    <button id="button" label="button"></button>
    <coachmark target="#button">
        <label>Wellcome!</label>
    </coachmark>
</zk>
```

Position

The positions of a coachmark(default: after_center). Here are the available options:

	start/before	center	end/after
top	before_start	before_center	before_end
bottom	after_start	after_center	after_end
left	start_before	start_center	start_after
right	end_before	end_center	end_after

Next

The next coachmark which will be opened when the onTargetClick event or the next() method is called.

Note: if you call next(Coachmark coachmark), it will use the specified coachmark instead of the predefined "next" coachmark.

Methods

public void next() : Closes the current coachmark and Opens the next one.

public void next(Coachmark coachmark) : Close the current coachmark and Opens the one you passed.(ignore the next coachmark you already set)

Supported Events

Name	Event Type
onOpen	Event: OpenEvent ^[3] Denotes that the user has opened or closed a component. Note: unlike onClose, this event is only a notification. The client sends this event after the component is opened or closed.
onTargetClick	Event: MouseEvent ^[4] Represents an event caused by a user's click on a highlighted target component.

- Inherited Supported Events: XulElement

Supported Children

*ALL

Version History

Version	Date	Content
9.0.0	Nov, 2019	ZK-4382 ^[3] : Provide a Coachmark component

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Coachmark.html#>

[2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/nav/Coachmark.html#>

[3] <https://tracker.zkoss.org/browse/ZK-4382>

Frozen

Frozen

- Demonstration: Spreadsheet Functionalities [1]
- Java API: Frozen [1]
- JavaScript API: Frozen [2]
- Style Guide: Frozen

Employment/Purpose

A frozen component to represent frozen "columns" in a Grid, like MS Excel. Specify the `start` attribute to define the horizontal scroll starting position.

Example

ID	Priority	Summary	Detail	Group
0001	1	Fix login issue	Login does not work at all	Account
0002	3	Button style broken	Check main.css	Styling
0003	2	Client search result	Search service returns incomplete result	Service

```
<grid width="600px">
    <frozen columns="2" start="1"/>
    <columns>
        <column width="50px">ID</column>
        <column width="50px">Priority</column>
        <column width="50px">Status</column>
        <column width="150px">Summary</column>
        <column width="250px">Detail</column>
        <column width="100px">Group</column>
        <column width="50px">Assign</column>
    </columns>
    <rows>
        <row>
            <cell>0001</cell>
            <cell>1</cell>
            <cell>closed</cell>
            <cell>Fix login issue</cell>
            <cell>Login does not work at all</cell>
            <cell>Account</cell>
            <cell>Bob</cell>
        </row>
    </rows>
</grid>
```

```

<row>
  <cell>0002</cell>
  <cell>3</cell>
  <cell>open</cell>
  <cell>Button style broken</cell>
  <cell>Check main.css</cell>
  <cell>Styling</cell>
  <cell>Alice</cell>
</row>
<row>
  <cell>0003</cell>
  <cell>2</cell>
  <cell>open</cell>
  <cell>Client search result</cell>
  <cell>Search service returns incomplete result</cell>
  <cell>Service</cell>
  <cell>Bob</cell>
</row>
</rows>
</grid>

```

Scroll to Hide Columns

By default, Grid will render extra space (larger width) after the last column. So that you can drag to hide all columns except the last one.

ID	Priority	Assign
0001	1	Bob
0002	3	Alice
0003	2	Bob

Supported Events

Name	Event Type
None	None

- Inherited Supported Events: XulElement

Supported Children

*ALL

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Frozen.html#>
- [2] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/mesh/Frozen.html#>

Paging

Paging

- Demonstration: Paging (with Grid) ^[1], Paging (with Listbox) ^[2], Paging (with Tree) ^[3]
- Java API: Paging ^[4]
- JavaScript API: Paging ^[5]
- Style Guide: Paging

Employment/Purpose

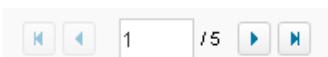
A paging component is used with another component to separate long content into multiple pages. If a component has long content to display, you could separate them into pages, and then use a paging component as a controller to allow the user decide which page to display.

The listbox, grid and tree components support the paging intrinsically, so you don't need to specify a paging component explicitly. In other words, they will instantiate and manage a paging component automatically if the paging mold is specified. Of course, you could specify an external paging component, if you want to have different visual layout, or to control multiple listboxes, grids and/or trees with one single paging component.

Example

For example, suppose you have 100 items and prefer to show 20 items at a time, then you can use the paging components as follows.

```
<vbox>
    <paging totalSize="100" pageSize="20"/>
</vbox>
```



When a user clicks on the hyperlinks, the `onPaging` event is sent with an instance of `PagingEvent` ^[6] to the paging component. To decide which portion of your 100 items are visible, you should add a listener to the paging component. Please note that the code below is pseudo code. For real examples, please refer to User Cases below.

```
<zk>
    <div id="content"/> <!-- the long content is displayed here -->
    <paging id="paging" />

    <zscript>
List result = new SearchEngine().find("ZK");
//assume SearchEngine.find() will return a list of items.

paging.setTotalSize(result.size());
paging.addEventListener("onPaging", new
org.zkoss.zk.ui.event.EventListener() {
    public void onEvent(Event event) {
        int pgno = event.getPaginal().getActivePage();
        int ofs = pgno * event.getPaginal().getPageSize();

        new Viewer().redraw(content,
            result, ofs, ofs +
event.getPaginal().getPageSize() - 1);
        //assume redraw(Div content, List result, int b, int
e) will display
        //the result to the content component from the b-th
item to the e-th item
    }
}
);
</zscript>
</zk>
```

Properties

Disabled

Paging.setDisabled(boolean)^[6] is used to disable the paging component. It can block the user from navigating through the pagination. For example,

```
<paging pageSize="2" disabled="true"/>
```

Limitation

Paging can not apply stubonly at the same time. For example,

```
<listbox mold="paging" pageSize="1" >
    <listitem >
        <listcell stubonly="true"/>
    </listitem>
    <listitem>
        <listcell />
```

```
</listitem>
</listbox>
```

Although paging will invalidate `listbox` and its children, `stubonly` needs the referred widget in client side which is detached during paging and throws mounting error.

Supported Events

Name	Event Type
onPaging	Event: PagingEvent [6] Notifies one of the pages of a multi-page component is selected by the user.

- Inherited Supported Events: XulElement

Supported Molds

Available molds of a component are defined in lang.xml embedded in zul.jar.

Name	Snapshot
default	
os	

Supported Children

*NONE

Use Cases

Version	Description	Example Location
3.6	Small talks	<ul style="list-style-type: none"> • Handling huge data using ZK • Paging Sorting with a filter object • Use Load-On-Demand to Handle Huge Data

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/zkdemo/grid/paging>
- [2] <http://www.zkoss.org/zkdemo/listbox/paging>
- [3] <http://www.zkoss.org/zkdemo/tree/paging>
- [4] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Paging.html#>
- [5] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/mesh/Paging.html#>
- [6] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Paging.html#setDisabled\(boolean\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/Paging.html#setDisabled(boolean))

Stepbar

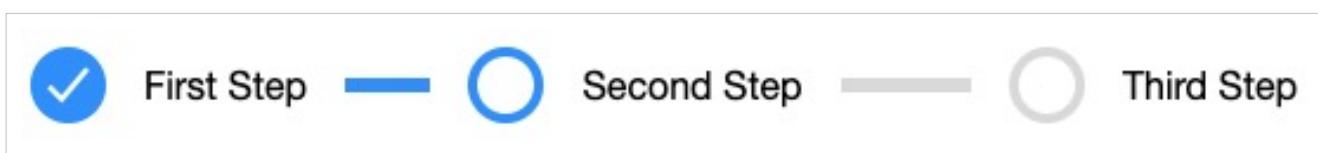
Stepbar

- Demonstration ^[1]
- Java API: Stepbar ^[2]
- JavaScript API: Stepbar ^[3]
- Available for ZK:
- CE PE EE

Employment/Purpose

Stepbar is a navigation component suitable for displaying the progress of a multi-step task.

Example



```
<zk>
    <stepbar activeIndex="1" width="600px">
        <step title="First Step" />
        <step title="Second Step" />
        <step title="Third Step"/>
    </stepbar>
</zk>
```

Supported Browsers

It is compatible with browsers that fully support CSS flexbox, like IE 11, Edge, Firefox, Opera, Chrome, and Safari. IE10 is not supported as it only partially supports flexbox. Check flexbox browser support ^[4].

Properties

ActiveIndex

The index of the active step. (Default: 0)

ActiveStep

The active step object. (Default: first step)

Linear

Set whether the steps in this stepbar are displayed in order.

Non-linear means users can toggle the active steps easily by clicking on any step even if the step is not the next one in the sequence. In linear mode, they can only activate in order.



Model

The step model associated with this stepbar. (Learn Model-Driven Rendering)

You can use DefaultStepModel ^[5], wrap an existing ListModelList ^[5] by DefaultStepModel's constructor, or implement StepModel ^[6].

StepRenderer

The renderer used to render each step.(Learn Model-Driven Rendering)

You can specify your own StepRenderer ^[7] at this attribute to renders a Step object with your data.

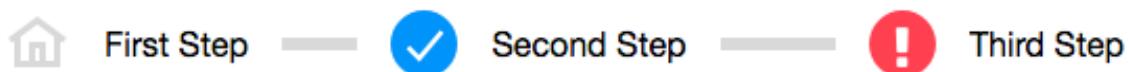
WrappedLabels

Set whether the labels in children steps are wrapped. (Default: false)

true:



false:



Supported Events

Name	Event Type
onChange	Event: Event ^[7] Represents an event caused by a user's selection changed at the client.

- Inherited Supported Events: XulElement

Supported Children

* Step

Use Cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
9.0.0	November, 2019	ZK-4375 ^[8] : Provide a stepbar component

References

- [1] <https://www.zkoss.org/zkdemo/menu/stepbar>
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Stepbar.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/wgt/Stepbar.html#>
- [4] <https://caniuse.com/flexbox>
- [5] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/DefaultStepModel.html#>
- [6] <https://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/StepModel.html>
- [7] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/StepRenderer.html#>
- [8] <https://tracker.zkoss.org/browse/ZK-4375>

Step

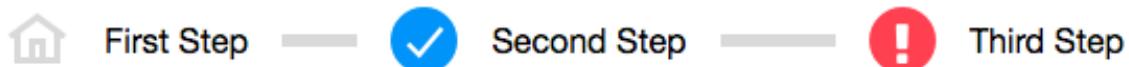
Step

- Demonstration: Step ^[1]
- Java API: Step ^[2]
- JavaScript API: Step ^[3]
- Available for ZK:
- CE PE EE

Employment/Purpose

A step is used for displaying user navigation, it should be placed inside a Stepbar and shouldn't be used without a Stepbar.

Example



```
<zk>
    <stepbar linear="false" activeIndex="2" width="600px">
        <step title="First Step" iconSclass="z-icon-home"/>
        <step title="Second Step" complete="true" />
        <step title="Third Step" error="true" />
    </stepbar>
</zk>
```

Properties

Complete

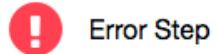
Whether this step is completed. (Default: false)



The default visual style:

Error

Whether this step is in error. (Default: false)



The default visual style:

The priority of `error` is higher than `complete`. If both properties are set, the result will be an error state visually.

IconSclass

Set the icon CSS class to apply a custom icon.

If you set this property, it will override the complete and error icons accordingly.



Custom Step icon



Custom Error



Custom Complete

```
<stepbar width="800px">
    <step title="Custom Step icon" iconSclass="z-icon-star-o"/>
    <step title="Custom Error" error="true" iconSclass="z-icon-bug"/>
    <step title="Custom Complete" iconSclass="z-icon-home"/>
</stepbar>
```

Please read [ZK_Component_Reference/Base_Components/LabelImageElement#IconSclass](#) to know more available build-in icons.

Title

Set the title (label) of each step. (Default: empty)

Supported Events

Name	Event Type
none	none

- Inherited Supported Events: XulElement

Supported Children

* None

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content
9.0.0	November, 2019	ZK-4375 [8]: Provide a stepbar component

References

- [1] <https://www.zkoss.org/zkdemo/menu/stepbar/>
- [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Step.html#>
- [3] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/wgt/Step.html#>

Events

This section outline events which are used in components.

AfterSizeEvent

AfterSizeEvent

- Demonstration: N/A
- Java API: AfterSizeEvent^[1]
- JavaScript API: N/A

Employment/Purpose

Represents an event that resizes and provides the new size of a component.

Example

Resize the window component to show different sized images accordingly.

```
<window title="AfterSizeEvent" border="normal" width="250px" height="200px" maximizable="true" sizable="true">
    <custom-attributes org.zkoss.zul.image.preload="true" />
    <attribute name="onAfterSize"><![CDATA[
        int width = event.getWidth();
        if (width >= 600)
            image.setSrc("/image/large.jpg");
        else if (width >= 400 && width < 600)
            image.setSrc("/image/medium.jpg");
        else
            image.setSrc("/image/small.jpg");
    ]]></attribute>
    <image id="image" src="/image/small.jpg" />
</window>
```

Supported events

Name	Event Type
None	None

Supported Children

*NONE

Use cases

Version	Description	Example Location

Version History

Version	Date	Content
6.5.2	March 2013	Add onAfterSize event to get component size ^[2]

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/AfterSizeEvent.html#>

[2] <http://tracker.zkoss.org/browse/ZK-1672>

BandScrollEvent

BandScrollEvent

- Demonstration: N/A
- Java API: N/A
- JavaScript API: N/A

Employment/Purpose

Represents an event caused by that user is scrolling or has scrolled at the client.

BandScrollEvent will be sent with name as "onBandScrollEvent" after the bandinfo component has been moved by the user.

The components which are supported this event are: Bandinfo .

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*NONE

Use cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

CheckEvent

CheckEvent

- Demonstration: N/A
- Java API: CheckEvent ^[5]
- JavaScript API: N/A

Employment/Purpose

Represents an event cause by the user's check a state at the client.

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*NONE

Use cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

ColSizeEvent

ColSizeEvent

- Demonstration: N/A
- Java API: ColSizeEvent ^[2]
- JavaScript API: N/A

Employment/Purpose

Used to notify that the width of a column is changed.

When an user drags the border of a sizable column, only the width of the column itself is changed, other columns are not affected.

The event is sent to the parent (e.g., Columns and Treecols).

Example

N/A

Supported Events

Name	Event Type
None	None

Supported Children

*None

Use cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
5.0.4	2010/08/03	Updated Employment/Purpose that reflects a change of Javadoc

CreateEvent

CreateEvent

- Demonstration: N/A
- Java API: CreateEvent ^[1]
- JavaScript API: N/A

Employment/Purpose

Used to notify a window that all its children are created and initialized. `UiEngine` post this event to components that declares the `onCreate` handler (either as a method or as in instance definition).

Example

N/A

Notes

Use with data binding

When the data binder processed a collection of data in, say, a grid or a listbox, it will detach the original one, and then clone it to represent each item of the data. For example,

```
<listbox model="@{person.interests}">
    <listitem self="@{each=obj}" value="@{obj}" onCreate="foo()"/>
</listbox>
```

where the execution sequence is as follows.

1. ZK Loader creates a listbox and a listitem, and posts `onCreate` to the listitem (since it has a listener).
2. The data binder processes all annotations, after all the components are created.
 1. When handling `each`, the data binder detaches the listitem, invokes `Component.clone()` ^[2] to make a clone for each item (`person.interests`), and attach the clone to the listbox.
 3. The listitem created by ZK Loader receives `onCreate`.

The detail behavior of step 3 is a bit different since 5.0.4. We will discuss it more detailed in the following sections.

5.0.3 and earlier

With 5.0.3 and earlier, only the original listitem (the listitem used as template to be cloned) will receive `onCreate`. Thus, whatever change the listener made won't affect the cloned listitems.

In summary, when using data binding with 5.0.3 or earlier, don't use `onCreate`.

5.0.4

Since 5.0.4, the data binder will fire `onCreate` to each cloned component, so it is safe to use `onCreate` with the data binder.

However, there is one more thing to be noticed: how the event listener is cloned when a component is cloned. By default, the new component will share the same listener with the original component. Sometimes, it might not be

correct (for example, the listener might be an inner class that assumes `this` to be the original component), the event listener shall implement `ComponentCloneListener`^[3] to clone the listener by itself. For example,

```
public FooCreateListener implements EventListener,
ComponentCloneListener {
    private Listitem _item;
    public FooListener(Listitem item) {
        _item = item;
    }
    public Object willClone(Component comp) {
        return new FooListener((Listitem)comp);
    }
    public void onEvent(Event evt) {
        //handle _item
    }
}
```

Instead of implementing `ComponentCloneListener`^[3], using `Event.getTarget()`^[4] could be easier to make a listener safe to clone.

```
public FooCreateListener implements EventListener,
ComponentCloneListener {
    public void onEvent(Event evt) {
        Listitem item = (Listitem)evt.getTarget();
        //then, handle item
    }
}
```

Supported events

Name	Event Type
None	None

Supported Children

*NONE

Use cases

Version	Description	Example Location

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/CreateEvent.html#>
- [2] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/Component.html#clone\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/Component.html#clone())
- [3] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/util/ComponentCloneListener.html#>
- [4] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/Event.html#getTarget\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/Event.html#getTarget())

DropEvent

DropEvent

- Demonstration: DropEvent ^[1]
- Java API: DropEvent ^[3]
- JavaScript API: N/A

Employment/Purpose

Represents an event cause by user's dragging and dropping a component.

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*NONE

Use cases

Version	Description	Example Location
5.0	Fadeout on Drop Event	[2]

Version History

Version	Date	Content

References

- [1] <http://www.zkoss.org/zkdemo/userguide/#a2>
- [2] <http://www.zkoss.org/forum/listComment/11377>

ErrorEvent

ErrorEvent

- Demonstration: N/A
- Java API: ErrorEvent ^[1]
- JavaScript API: N/A

Employment/Purpose

Represents an event caused by a user's error when entering wrong data or clearing the last wrong data. ErrorEvent is sent when the client detects users entered a wrong value.

Note: if the client doesn't detect the error, the value is sent back to the server with regular event, such as InputEvent

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*NONE

Use cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/ErrorEvent.html#>

Event

Event

- Demonstration: N/A
- Java API: Event ^[7]
- JavaScript API: N/A

Employment/Purpose

An event sent to the event handler of a component.

Supported events

Name	Event Type
None	None

Supported Children

*NONE

Use cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

HistoryPopStateEvent

HistoryPopStateEvent

- Demonstration: N/A
- Java API: HistoryPopStateEvent^[1]
- JavaScript API: N/A

Employment/Purpose

The history pop state event used with `onHistoryPopState` to notify that user pressed BACK, FORWARD or others that causes the history changed (but still in the same desktop).

All root components of all pages of the desktop will receives this event.

Example

The example shows how to push a history state and handle a `HistoryPopStateEvent` object.

```
<zk>
<tabbox id="tb" height="300px">
<attribute name="onSelect"><! [CDATA[
    Tab selected = event.getReference();
    int selectedIndex = selected.getIndex();
    desktop.pushHistoryState(Collections.singletonMap("tabIndex",
selectedIndex), "", "/" + selectedIndex);
]]>
</attribute>
<attribute name="onHistoryPopState"><! [CDATA[
    Map state = event.getState();
    if (state != null)
        tb.setSelectedIndex(state.get("tabIndex"));
]]>
</attribute>
<tabs id="tabs">
    <tab id="A" label="Tab A" />
    <tab id="B" label="Tab B" />
    <tab id="C" label="Tab C" />
    <tab id="D" label="Tab D" />
    <tab id="E" label="Tab E" />
</tabs>
<tabpanel>This is panel A</tabpanel>
<tabpanel>This is panel B</tabpanel>
<tabpanel>This is panel C</tabpanel>
<tabpanel>This is panel D</tabpanel>
<tabpanel>This is panel E</tabpanel>
</tabpanel>
```

```
</tabbox>
</zk>
```

- Line 6: Use Desktop.pushHistoryState(java.lang.Object,java.lang.String,java.lang.String) [2] to push a history state.
- Line 9: Listen `onHistoryPopState` on any root component to handle `HistoryPopStateEvent` object. You can handle events in an MVC fashion.

```
public class TestComposer extends SelectorComposer<Tabbox> {
    @Wire
    private Tabbox tb;

    @Listen("onHistoryPopState = #tb")
    public void handleHistoryPopState(HistoryPopStateEvent event) {
        Map state = (Map) event.getState();
        if (state != null) {
            tb.setSelectedIndex((int) state.get("tabIndex"));
        }
    }
}
```

- Line 5: Listen the `onHistoryPopState` event of the root component `#tb`.

Or you can use a special annotation `HistoryPopState` [3] if you prefer MVVM.

```
public class TestVM {
    private int selectedIndex = 0;

    public int getSelectedIndex() {
        return selectedIndex;
    }

    public void setSelectedIndex(int index) {
        selectedIndex = index;
        Desktop desktop = Executions.getCurrent().getDesktop();
        desktop.pushHistoryState(Collections.singletonMap("tabIndex",
        selectedIndex), "", "/" + selectedIndex);
    }

    @HistoryPopState
    @SmartNotifyChange("selectedIndex")
    public void
    handleHistoryPopState(@ContextParam(ContextType.TRIGGER_EVENT)
HistoryPopStateEvent event) {
        Map state = (Map) event.getState();
        if (state != null) {
            selectedIndex = ((int) state.get("tabIndex"));
        }
    }
}
```

- Line 14: a method annotated with `HistoryPopState` can handle `HistoryPopStateEvent`.

Supported events

Name	Event Type
None	None

Supported Children

*NONE

Use cases

Version	Description	Example Location

Version History

Version	Date	Content
8.5.0	Oct 2017	ZK-3711 [4]: Support HTML5 history API

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/HistoryPopStateEvent.html#>
- [2] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/Desktop.html#pushHistoryState\(java.lang.Object,java.lang.String,java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/Desktop.html#pushHistoryState(java.lang.Object,java.lang.String,java.lang.String))
- [3] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/bind/annotation/HistoryPopState.html#>
- [4] <http://tracker.zkoss.org/browse/ZK-3711>

InfoChangeEvent

InfoChangeEvent

- Demonstration: N/A
- Java API: N/A
- JavaScript API: N/A

Employment/Purpose

Represents a Google Maps related event which is triggered whenever the currently opened `ginfo` or `gmarker` is changed.

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*N/A

Use cases

Version	Description	Example Location

Version History

Version	Date	Content

InputEvent

InputEvent

- Demonstration: N/A
- Java API: InputEvent [1]
- JavaScript API: N/A

Employment/Purpose

Represents an event cause by user's input something at the client.

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*N/A

Use cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/InputEvent.html#>

KeyEvent

KeyEvent

- Demonstration: N/A
- Java API: KeyEvent ^[6]
- JavaScript API: N/A

Employment/Purpose

Represents a key pressed by the user.

For more information, please refer to ZK Developer's Reference: Keystroke Handling.

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*N/A

Use cases

Version	Description	Example Location
3.6	Smalltalk: Keylistener Component	Keylistener Component

Version History

Version	Date	Content

MapDropEvent

MapDropEvent

- Demonstration: N/A
- Java API: N/A
- JavaScript API: N/A

Employment/Purpose

Represents a Google Maps related event which is triggered whenever a component is dragged and dropped on a gmaps or gmarker. This event was sent to the dropped component with the latitude and longitude information of the dropping point.

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*N/A

Use cases

Version	Description	Example Location

Version History

Version	Date	Content

MapMouseEvent

MapMouseEvent

- Demonstration: N/A
- Java API: N/A
- JavaScript API: N/A

Employment/Purpose

Represents a Google Maps related event which is triggered whenever a gmaps is mouse clicked, right clicked, or double clicked. The event includes the latitude and longitude information of the mouse clicking point.

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*N/A

Use cases

Version	Description	Example Location

Version History

Version	Date	Content

MapMoveEvent

MapMoveEvent

- Demonstration: N/A
- Java API: N/A
- JavaScript API: N/A

Employment/Purpose

Represents a Google Maps related event which is triggered whenever the view center of the `gmaps` is moved or the zoom level of the `gmaps` is changed.

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*N/A

Use cases

Version	Description	Example Location

Version History

Version	Date	Content

MapTypeChangeEvent

MapTypeChangeEvent

- Demonstration: N/A
- Java API: N/A
- JavaScript API: N/A

Employment/Purpose

Represents a Google Maps related event which is triggered whenever the current type(normal, satellite, hybrid, physical) of the gmaps is changed.

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*N/A

Use cases

Version	Description	Example Location

Version History

Version	Date	Content

MapZoomEvent

MapZoomEvent

- Demonstration: N/A
- Java API: N/A
- JavaScript API: N/A

Employment/Purpose

Represents a Google Maps related event which is triggered whenever the zoom level of a gmaps is changed.

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*N/A

Use cases

Version	Description	Example Location

Version History

Version	Date	Content

MouseEvent

MouseEvent

- Demonstration: N/A
- Java API: MouseEvent^[1]
- JavaScript API: N/A

Employment/Purpose

Represents an event caused by the mouse's activity. There are two possible ways to identify a mouse event. One is by coordination (`getX()` and `getY()`). The other is by a logical name, called area (`getArea()`).

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*N/A

Use cases

Version	Description	Example Location

Version History

Version	Date	Content
5.0.4	August 2010	MouseEvent.getAreaComponent() ^[6] is introduced to simplify the retrieval of the associated component (usually Area ^[7]).

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/MouseEvent.html#>

MoveEvent

MoveEvent

- Demonstration: N/A
- Java API: MoveEvent [5]
- JavaScript API: N/A

Employment/Purpose

Represents an event caused by a component being moved.

Component Implementation Note:

A movable component must implement Movable for the returned object of ComponentCtrl.getExtraCtrl().

Supported events

Name	Event Type
None	None

Supported Children

*NONE

Use cases

Version	Description	Example Location

Version History

Version	Date	Content

OccurEventSelectEvent

OccurEventSelectEvent

- Demonstration: N/A
- Java API: N/A
- JavaScript API: N/A

Employment/Purpose

Represents an event caused by that user click the Occur Event in the bandinfo component

OccurEventSelectEvent will be sent with name as "onOccurEventSelectEvent " after the Occur Event in the bandinfo component has been clicked by the user.

The components which are supported this event are: Bandinfo .

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*NONE

Use cases

Version	Description	Example Location

Version History

Version	Date	Content

OpenEvent

OpenEvent

- Demonstration: OpenEvent ^[1]
- Java API: OpenEvent ^[3]
- JavaScript API: N/A

Employment/Purpose

Represents an event cause by user's opening or closing something at the client.

Note: it is a bit confusing but `Events.ON_CLOSE` is sent when user clicks a close button. It is a request to ask the server to close a window, a tab or others. If the server ignores the event, nothing will happen at the client. By default, the component is detached when receiving this event.

On the other hand, `Events.ON_OPEN` (with OpenEvent) is a notification. It is sent to notify the server that the client has opened or closed something. And, the server can not prevent the client from opening or closing.

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*NONE

Use cases

Version	Description	Example Location

Version History

Version	Date	Content

References

[1] <http://www.zkoss.org/zkdemo/userguide/#e9>

OverPlotEvent

OverPlotEvent

- Demonstration: N/A
- Java API: N/A
- JavaScript API: N/A

Employment/Purpose

Represents an event caused by that user mouse-over the plotinfo component

OverPlotEvent will be sent with name as "onOverPlotEvent".

The components which are supported this event are: Plotinfo .

Example

N/A

Supported Events

Name	Event Type
None	None

Supported Children

*NONE

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content

PageSizeEvent

PageSizeEvent

- Java API: PageSizeEvent^[1]

Employment/Purpose

Used to notify that the page size is changed (by the user), or by paginal (such as Paging).

Supported events

Name	Event Type
None	None

Supported Children

*NONE

Use cases

Version	Description	Example Location

Version History

Version	Date	Content

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/event/PageSizeEvent.html#>

PagingEvent

PagingEvent

- Demonstration: N/A
- Java API: PagingEvent^[1]
- JavaScript API: N/A

Employment/Purpose

Used to notify that a new page is selected by the user, or by Paginal (such as Paging). It is used for paging long content.

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*ALL

Use cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul//event/PagingEvent.html#>

PortalMoveEvent

PortalMoveEvent

- Demonstration: N/A
- Java API: PortalMoveEvent^[4]
- JavaScript API: N/A

Employment/Purpose

Represents an event caused by a portal being moved.

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*NONE

Use cases

Version	Description	Example Location

Version History

Version	Date	Content

ScrollEvent

ScrollEvent

- Demonstration: N/A
- Java API: ScrollEvent ^[5]
- JavaScript API: N/A

Employment/Purpose

Represents an event caused by that user is scrolling or has scrolled at the client.

ScrollEvent will be sent with name as "onScroll" after setCurposByClient (int) is called to notify application developers that it is called by user (rather than by codes).

For components that might also support ScrollEvent with "onScrolling". It is used to notified the server that user is changing its content (changing is on progress and not finished).

The components which are supported this event are: Slider ^[2].

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*NONE

Use cases

Version	Description	Example Location

Version History

Version	Date	Content

SelectEvent

SelectEvent

- Demonstration: N/A
- Java API: SelectEvent [6]
- JavaScript API: N/A

Employment/Purpose

Represents an event cause by user's the list selection is changed at the client.

Example

Get Keys

It would be very helpful to add the `getKeys()` method of a `MouseEvent` to `SelectEvents`. For example, if you're using a listbox and want to detect if the CTRL key is being held when one of the listitems is selected.

keys: 258

Population	%
a	20%
b	c

```
<zk>
    <label id="i"/>
    <listbox onSelect='i.value = "keys: "+event.getKeys()'>
        <listhead>
            <listheader label="Population"/>
            <listheader label="%"/>
        </listhead>
        <listitem value="A">
            <listcell>a</listcell>
            <listcell label="20%"/>
        </listitem>
        <listitem value="B">
            <listcell>b</listcell>
            <listcell>c</listcell>
        </listitem>
    </listbox>
</zk>
```

Get the Previous Selected Items

If user want to know which of the selected items are new added or removed, they can use the `getPreviousSelectedItems` method to filter out. (if model exists and in paging mold, it should return null)

Get the Previous Selected Objects

If user want to know which of the selected objects are new added or removed, they can use the `getPreviousSelectedObjects` method to filter out. The information is available only when the target component has a model.

* [since 7.0.1]

Get the Unselected Items

If user want to know which of the selected items are deselected, they can use the `getUnselectedItems` method to filter out. (if model exists and in paging mold, it should return null)

* [since 7.0.1]

Get the Unselected Objects

If user want to know which of the selected objects are deselected, they can use the `getUnselectedObjects` method to filter out. The information is available only when the target component has a model.

* [since 7.0.1]

Supported events

Name	Event Type
None	None

Supported Children

*NONE

Use cases

Version	Description	Example Location

Version History

Version	Date	Content
7.0.0	October, 2013.	Multiple selection component(listbox, chosenbox) support extra information for find out added item/removed item [1]
7.0.1	January 2014.	SelectEvent support get UnselectedItems [2]

References

[1] <http://tracker.zkoss.org/browse/ZK-1992>

[2] <http://tracker.zkoss.org/browse/ZK-2089>

SelectionEvent

SelectionEvent

- Demonstration: N/A
- Java API: SelectionEvent [6]
- JavaScript API: N/A

Employment/Purpose

Represents an event cause by user's the list selection is changed at the client.

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*NONE

Use cases

Version	Description	Example Location

Version History

Version	Date	Content
---------	------	---------

SizeEvent

SizeEvent

- Demonstration: N/A
- Java API: SizeEvent ^[4]
- JavaScript API: N/A

Employment/Purpose

Represents an event caused by a component being re-sized.

Component Implementation Note: A sizable component must implement Sizable for the returned object of ComponentCtrl.getExtraCtrl().

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*NONE

Use cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

VisibilityChangeEvent

VisibilityChangeEvent

- Demonstration: N/A
- Java API: VisibilityChangeEvent ^[1]
- JavaScript API: N/A

Employment/Purpose

This event is fired when users change a browser page visibility e.g. switch to another tab or switch back. You should listen to this event on the root component of a page.

ZK implements it based on W3C page visibility ^[2].

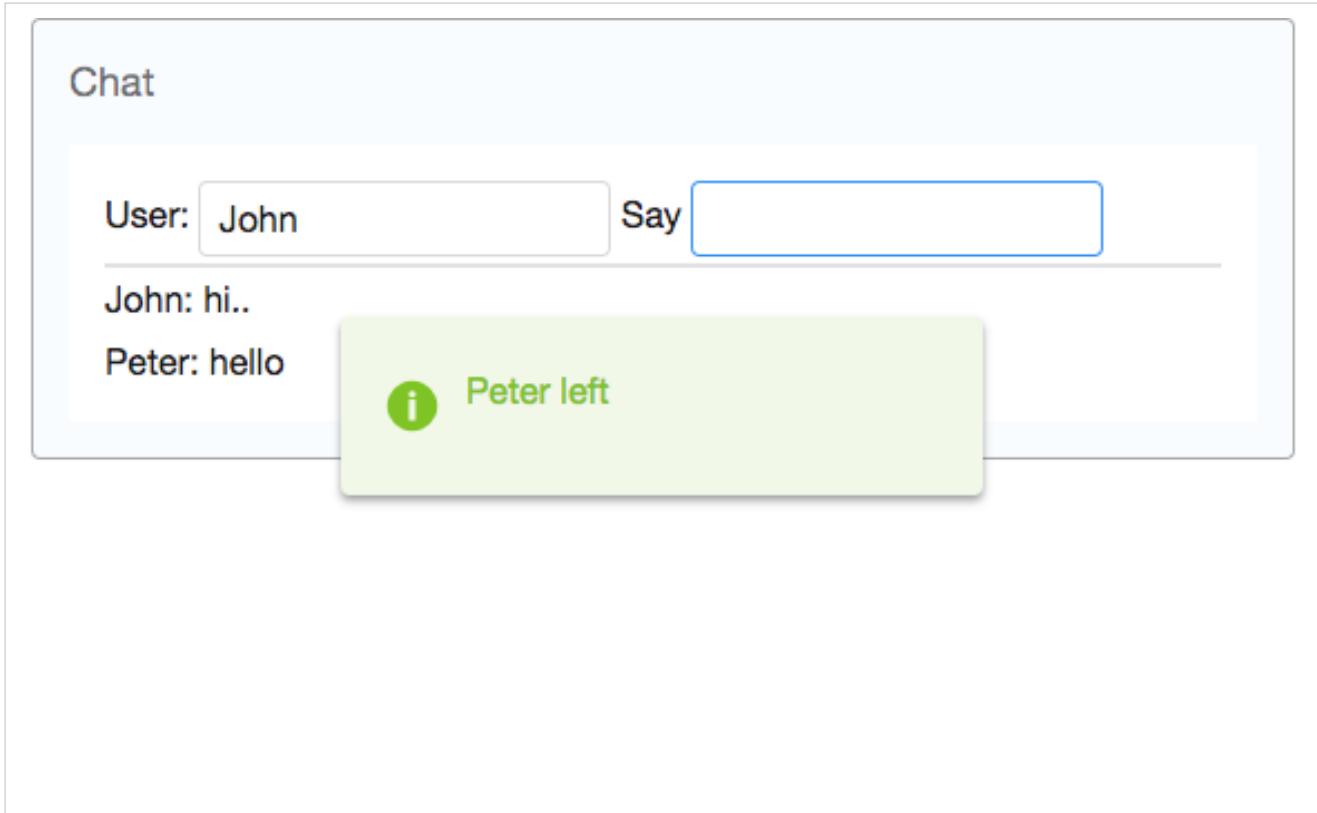
Example

Basic

```
<window title="window" border="normal">
    <attribute name="onVisibilityChange">
        if (!event.isHidden())
            lbl.setValue("Welcome back");
    </attribute>
    <label id="lbl"></label>
</window>
```

Chatroom

In a chatroom application, you detect a user who switches to another tab, then notify other users. Please check the complete source ^[3].



Version History

Version	Date	Content
6.5.1	December 2, 2012	introduced in Control page visibility with HTML5 API in ZK ^[4]

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/VisibilityChangeEvent.html#>
- [2] <http://www.w3.org/TR/page-visibility/>
- [3] <https://github.com/zkoss/zkbooks/blob/master/componentreference/src/main/webapp/events/chatroom.zul>
- [4] <http://blog.zkoss.org/2012/12/02/control-page-visibility-with-html5-api-in-zk/>

UploadEvent

UploadEvent

- Demonstration: UploadEvent ^[1]
- Java API: UploadEvent ^[5]
- JavaScript API: N/A

Employment/Purpose

Represents that the user has uploaded one or several files from the client to the server.

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*NONE

Use cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

[1] <http://www.zkoss.org/zkdemo/userguide/#u1>

ZIndexEvent

ZIndexEvent

- Demonstration: N/A
- Java API: ZIndexEvent^[9]
- JavaScript API: N/A

Employment/Purpose

Represents an event caused by a component whose z-index is modified by the client. A z-indexed component must send ZindexEvent once the z-index of component is modifiable by the client.

The components which are supported this event are: Window.

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*NONE

Use cases

Version	Description	Example Location

Version History

Version	Date	Content

Supporting Classes

This section outlines classes which support ZK's components.

AbstractListModel

AbstractListModel

- Demonstration: N/A
- Java API: AbstractListModel^[4]
- JavaScript API: N/A

Employment/Purpose

A skeletal implementation for ListModel.

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*None

Use cases

Version	Description	Example Location

Version History

Version	Date	Content

Constraint

Constraint

- Demonstration: N/A
- Java API: Constraint^[3]
- JavaScript API: N/A

Employment/Purpose

An interface that defines validate method to verify whether the value is acceptable

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*None

Use cases

Version	Description	Example Location

Version History

Version	Date	Content

Constrained

Constrained

- Demonstration: N/A
- Java API: constrained^[1]
- JavaScript API: N/A

Employment/Purpose

Decorates a component that its value is constrained by Constraint.

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*NONE

Use cases

Version	Description	Example Location

Version History

Version	Date	Content

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/ext/constrained.html#>

ListitemRenderer

ListitemRenderer

- Demonstration: N/A
- Java API: ListitemRenderer^[11]
- JavaScript API: N/A

Employment/Purpose

Identifies components that can be used as "rubber stamps" to paint the cells in a Listbox.

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

None

Use cases

Version	Description	Example Location

Version History

Version	Date	Content

ListModel

ListModel

- Demonstration: Listbox (Live Data) ^[1]
- Java API: ListModel ^[7]
- JavaScript API: N/A

Employment/Purpose

This interface defines the methods that components like Listbox and Grid use to get the content of items.

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*NONE

Use cases

Version	Description	Example Location

Version History

Version	Date	Content

References

[1] <http://www.zkoss.org/zkdemo/userguide/#g8>

Messagebox

Messagebox

- Demonstration: Messagebox ^[1]
- Java API: Messagebox ^[2]
- JavaScript API: N/A

Employment/Purpose

It provides a set of utilities to show a message and have a user to confirm a situation.

It is typically used to alert users when an error occurs, or to prompt users for an decision.

Example

The simplest use of a message box is to inform the user something is done. For example,

```
Messagebox.show("The backup has been done.");
Messagebox.show("Failed to access the information", null, 0,
Messagebox.ERROR);
```

There are a lot of utilities that allow you to show a message in different look, such as the buttons, icon and title. Please refer to Messagebox ^[2] for more information.

Take Actions Depending On Which Button Is Clicked

If you'd like to know which button is clicked, you have to implement an event listener^[3]. For example,

```
Messagebox.show("Something is changed. Are you sure?",
    "Question", Messagebox.OK | Messagebox.CANCEL,
    Messagebox.QUESTION,
    new org.zkoss.zk.ui.event.EventListener() {
        public void onEvent(Event e) {
            if (Messagebox.ON_OK.equals(e.getName())) {
                //OK is clicked
            } else
            if (Messagebox.ON_CANCEL.equals(e.getName())) {
                //Cancel is clicked
            }
        }
    );
}
```

The invocation of `java.lang.String, int, java.lang.String)` `Messagebox.show(java.lang.String, java.lang.String, int, java.lang.String)` ^[4] will return immediately after the invocation^[5]. Then, if the user clicks a button, the event listener will be invoked. You could examine the event name to know which button is clicked. If the user clicked the Close button on the right-top corner, the `onClose` event is fired.

-
- [1] <http://www.zkoss.org/zkdemo/userguide/#18>
 - [2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/MessageBox.html#>
 - [3] If you want to make it running under clustering environment, you should implement Messagebox (<http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/MessageBox.html#>). For more information, please refer to ZK Developer's Reference: Clustering.
 - [4] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/MessageBox.html#show\(java.lang.String,java.awt.event.ActionListener\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/MessageBox.html#show(java.lang.String,java.awt.event.ActionListener))
 - [5] Here we assume the event thread is disabled (default). If the event thread is enabled, the show method will suspend until the user clicks a button. Thus, you could know which button is clicked by simply examining the returned value.

Listen ClickEvent

[since 6.0.0]

Since ZK 6, the event listener will be invoked with an instance of ClickEvent (<http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/ClickEvent.html#>), and it is easy to retrieve the button being clicked from it. For example,

```
Messagebox.show("Something is changed. Are you sure?",  
    "Question", Messagebox.OK | Messagebox.CANCEL,  
    Messagebox.QUESTION,  
    new org.zkoss.zk.ui.event.EventListener<ClickEvent>() {  
        public void onEvent(ClickEvent e) {  
            switch (e.getButton()) {  
                case Messagebox.Button.OK: //OK is clicked  
                case Messagebox.Button.CANCEL: //Cancel is  
                    clicked  
                default: //if the Close button is clicked,  
                    e.getButton() returns null  
            }  
        }  
    };  
);
```

Customization

Assign the Order of Buttons

[since 6.0.0]

If you'd like to assign the order, you could use org.zkoss.zul.MessageBox.Button[([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/MessageBox.html#show\(java.lang.String,org.zkoss.zk.ui.event.EventListener listener\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/MessageBox.html#show(java.lang.String,org.zkoss.zk.ui.event.EventListener listener))) [Messagebox.show\(java.lang.String, org.zkoss.zul.MessageBox.Button\[\], org.zkoss.zk.ui.event.EventListener listener\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/MessageBox.html#show(java.lang.String,org.zkoss.zul.MessageBox.Button[],org.zkoss.zk.ui.event.EventListener listener))] as follows.

```
Messagebox.show("Cancel the operation?",  
    new MessageBox.Button[] {Messagebox.Button.NO,  
    Messagebox.Button.YES},  
    new EventListener<MessageBox.ClickEvent>() { //optional  
        public void onEvent(MessageBox.ClickEvent event) {  
            //...  
        }  
    } );
```

The buttons will be displayed in the same order as the array specified in the `buttons` argument.

If you don't care the order, you could use a combination of constants, such as `Messagebox.OK` (<http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/MessageBox.html#OK>). For example,

```
Messagebox.show("Cancel the operation?", null,  
Messagebox.YES+Messagebox.NO, null);
```

Assign the Labels of Buttons

[since 6.0.0]

By default, the label of a button is loaded from the message file based on the current locale. However, you could assign any label you'd like.

```
onClick='Messagebox.show("Yes and No", "Custom Labels",  
    new Messagebox.Button[] {Messagebox.BUTTON.YES,  
Messagebox.BUTTON.NO},  
    new String[] {"Yes, it is correct"},  
    Messagebox.INFORMATION, null, null)'>
```

The `btnLabels` argument is an array of labels you'd like to use. If it is null or the length is shorter than the array specified the `buttons` argument, the default label will be used.

The Default Title

If the title is not specified in the application's name (returned by `WebApp getAppName()` ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/WebApp.html#getAppName\(\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/WebApp.html#getAppName()))). You could change it by invoking `WebApp.setAppName(java.lang.String)` ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/WebApp.html#setAppName\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/WebApp.html#setAppName(java.lang.String))).

Since 5.0.6, you could specify the application's name with a library property called `org.zkoss.zk.ui.WebApp.name`. For example, you could specify the following in `WEB-INF/zk.xml`:

```
<library-property>  
    <name>org.zkoss.zk.ui.WebApp.name</name>  
    <value>My Killer Application</value>  
</library-property>
```

The Template

The UI of a message box is based on a ZUL file, so you could customize it by replacing it with your own implementation. It can be done easily by invoking `Messagebox.setTemplate(java.lang.String)` ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/MessageBox.html#setTemplate\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/MessageBox.html#setTemplate(java.lang.String))). Notice that it affects all message boxes used in an application. It is typically called when the application starts (i.e., in `WebAppInit.init(org.zkoss.zk.ui.WebApp)` ([http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/util/WebAppInit.html#init\(org.zkoss.zk.ui.WebApp\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/util/WebAppInit.html#init(org.zkoss.zk.ui.WebApp))) -- for more information, please refer to ZK Developer's Reference: Init and Cleanup).

To implement a custom template, please take a look at the default template (<https://github.com/zkoss/zk/blob/master/zul/src/archive/web/zul/html/messagebox.zul>).

The Width and Parameters

[since 6.0.0]

The `params` argument in `java.lang.String, org.zkoss.zul.Messagebox.Button[(http://www.zkoss.org/javadoc/latest/ zk/ org/ zkoss/ zul/ Messagebox. html#show(java. lang. String,), java.lang.String[], java.lang.String, org.zkoss.zul.Messagebox.Button, org.zkoss.zk.ui.event.EventListener, java.util.Map)` `Messagebox.show(java.lang.String, java.lang.String, org.zkoss.zul.Messagebox.Button[], java.lang.String[], java.lang.String, org.zkoss.zul.Messagebox.Button, org.zkoss.zk.ui.event.EventListener, java.util.Map)]` allows you to customize a message dialog further. For example, you could make the dialog wider with the parameter called `width` as shown below.

```
Map params = new HashMap();
params.put("width", 500);
Messagebox.show("This is a very long statement and meaningless to see
if it looks ok with the given width.",
    null, null, null, Messagebox.INFORMATION, null,
    new EventListener() {
        public void onEvent(Event event) {
            //...
        }
    }, params);
```

The parameters will be passed to the dialog template (described in the previous section), so you could pass whatever you'd like as long as the template recognize them. In additions, the priority of the `params` argument is higher, i.e., it could override the default values, though it is rarely required.

[since 7.0.1]

User also can customize the style of message dialog with the parameter called `sclass` as below.

```
Map params = new HashMap();
params.put("sclass", "myMessagebox");
Messagebox.show("It's a customized style message box.",
    null, null, null, Messagebox.INFORMATION, null,
    new EventListener() {
        public void onEvent(Event event) {
            //...
        }
    }, params);
```

Without Buttons' Dialog

[since 6.5.1]

If you'd like to show a non-buttons dialog, you could use `org.zkoss.zul.Messagebox.Button[(http://www.zkoss.org/ javadoc/ latest/ zk/ org/ zkoss/ zul/ Messagebox. html#show(java. lang. String,), org.zkoss.zk.ui.event.EventListener listener) Messagebox.show(java.lang.String, org.zkoss.zul.Messagebox.Button[], org.zkoss.zk.ui.event.EventListener listener)]` with an empty array as follows.

```
Messagebox.show("Cancel the operation?",
    new Messagebox.Button[0], null);
```

This messagebox will show without any buttons.

Supported events

Name	Event Type
None	None

Supported Children

*NONE

Use cases

Version	Description	Example Location

Version History

Version	Date	Content
6.0.0	October 2011	The order and labels of the buttons were assignable.
6.0.0	October 2011	ClickEvent (http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/MessageBox/ClickEvent.html#) was introduced to simplify the identification of a button.
6.5.1	September 2012	MessageBox with no button (http://tracker.zkoss.org/browse/ZK-1351)
7.0.1	January 2014	Add sclass to messagebox (http://tracker.zkoss.org/browse/ZK-2087)

RendererCtrl

RendererCtrl

- Demonstration: N/A
- Java API: RendererCtrl^[1]
- JavaScript API: N/A

Employment/Purpose

This interface defines the methods components like Listbox used to notify the renderer for several circumstance.

Though `ListitemRenderer.render(org.zkoss.zul.Listitem, java.lang.Object)` is called one item at a time, a request might have several items to render. And, if the renderer implements this interface, `doTry()` will be called before any rendering, and `doFinally()` will be called after all rendering. If any exception occurs, `doCatch(java.lang.Throwable)` will be called.

A typical use is to start a transaction and use it for rendering all items from the same request.

Supported events

Name	Event Type
None	None

Supported Children

*N/A

Use cases

Version	Description	Example Location

Version History

Version	Date	Content

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zul/RendererCtrl.html#>

SimpleConstraint

SimpleConstraint

- Demonstration: N/A
- Java API: SimpleConstraint ^[5]
- JavaScript API: SimpleConstraint ^[1]

Employment/Purpose

A simple constraint that you could build based the predefined constants.

Supported events

Name	Event Type
None	None

Supported Children

*N/A

Use cases

Version	Description	Example Location
---------	-------------	------------------

Version History

Version	Date	Content
---------	------	---------

References

[1] <http://www.zkoss.org/javadoc/latest/jsdoc/zul/inp/SimpleConstraint.html#>

SimpleListModel

SimpleListModel

- Demonstration: N/A
- Java API: SimpleListModel^[8]
- JavaScript API: N/A

Employment/Purpose

A simple implementation of [%s %s].

Example

N/A

Supported events

Name	Event Type
None	None

Supported Children

*ALL

Use cases

Version	Description	Example Location

Version History

Version	Date	Content

XHTML Components

This section describes how to use the XHTML component set (i.e., XHTML components). XHTML components are in a different component set than ZUL components. You have to specify XML namespace to distinguish them if you want to use them in the same ZUML document. For example,

```
<window xmlns:h="xhtml">
  <h:ul>
    <h:li>Click <button/></h:li>
  </h:ul>
</window>
```

Notice that the HTML component set is one of the approaches to use HTML tags directly in a ZUML document. In most cases, it is not the best approach. For information please refer to ZK Developer's Reference: HTML tags.

Encoding URLs

A XHTML component generates attributes directly to native HTML tags. It means, unlike XUL, it doesn't prefix the servlet context path to attributes for specifying URL. For example, the following codes will not work (unless the servlet context is "").

```
<img href="/my/good.png"/>
```

Instead, you should use the `<mp>encodeURL</mp>` function in EL expressions as follows.

```
<?taglib uri="http://www.zkoss.org/dsp/web/core" prefix="p"?>
...
<img href="${p:encodeURL('/my/good.png')}"/>
```

In Java, you should use the method, `Execution.encodeURL(java.lang.String)` [1].

```
<img id="another"/>
<zscript>
  another.setDynamicAttribute("href",
    Executions.getCurrent().encodeURL("/my/good.png"));
</zscript>
```

Notice that XUL components and all ZK features that accept a URL will invoke the `<mp>encodeURL</mp>` method automatically. The reason why we do not handle XHTML components is that we do not know which attribute requires a URL.

References

[1] [http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/Execution.html#encodeURL\(java.lang.String\)](http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/Execution.html#encodeURL(java.lang.String))

In Pure Java

It is also possible to create XHTML components in Java. The XHTML components are mapped to classes by ZK which means you can implement something like this:

```
Td myTd = new Td();
```

This enables you to use XHTML components from ZUL or within Java just like anything ZK related. If you need to output an XHTML component which is not present in ZK you can use the Raw object.

Raw

A special component, Raw^[1] is used to represent any component that is not declared in the following section (i.e., not in lang.xml). In other words, if any unrecognized component name is found, an instance of Raw^[1] is created and a proper HTML tag will be generated accordingly. In other words, any component name is legal as long as the targeted browser supports.

```
<marquee align="top">...</marquee>
```

is equivalent to

```
new Raw("marquee").setDynamicProperty("align", "top");
```

Next let's investigate the differences between XUL and XHTML components.

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/xhtml/Raw.html#>

The Difference Between XUL and XHTML Components

All XHTML components are derived from AbstractTag [1].

An XHTML component is a thin wrapper that encapsulates a native HTML tag. It is different to a XUL component or other none-native component in several ways.

- By implementing the RawId [2] interface, the universal identifier, <mp>getUuid</mp>, is the same as the identifier <mp>getId</mp>
- By implementing the DynamicAttributes [3] interface, all XHTML components support arbitrary attributes. In other words, any attribute name is legal (as long as the targeted browser supports)

Notice that the HTML component set is one of the approaches to use HTML tags directly in a ZUML document. In most cases, it is not the best approach. For information please refer to ZK Developer's Reference: HTML tags.

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zhtml/impl/AbstractTag.html#>

[2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/ext/RawId.html#>

[3] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/ext/DynamicAttributes.html#>

XML Components

This section describes how to use the XML component set. The XML component set is a special component set used in a XML device. A XML device is a client that accepts XML output. You can *not* use it with ZUL or XHTML.

For introduction please refer to ZK Developer's Reference.

Most of XML elements with the XML namespace are mapped to a general XML component (XmlNativeComponent [1]) that will generate the element and all its attributes to the client directly. However, the XML component set also provide some components for different functionality. We discuss them one-by-one in the following sections.

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zml/XmlNativeComponent.html#>

Transformer

Transformer

- Java API: Transformer [1]

Employment/Purpose

Transformer [1] is used to translate a XML document to another with a XSTL [2] template.

```
<?page contentType="text/html; charset=UTF-8"?>
<x:transformer xsl="book.xsl" xmlns:x="xml">
    <book>
        <title>ZK - Ajax without the JavaScript Framework</title>
        <for-who>Web application designers and programmers who wish to
implement
            rich Ajax web applications in the simplest way.</for-who>
        <author>Henri Chen and Robbie Cheng</author>
    </book>
</x:transformer>
```

where transformer is a component of the XML component set, so we have to specify the namespace: xml. Otherwise, the native namespace is assumed, and the element is generated directly.

Then, let us assume the content of book.xsl is as follows.

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
        <html>
            <head>
                <title>Book Info</title>
            </head>
            <body>
                <h1>Book Info</h1>
                <xsl:apply-templates select="book"/>
            </body>
        </html>
    </xsl:template>
    <xsl:template match="book">
        <dl>
            <dt>Title:</dt>
            <dd><xsl:value-of select="title"/></dd>
            <dt>Who is this book for:</dt>
            <dd><xsl:value-of select="for-who"/></dd>
            <dt>Authors</dt>
            <dd><xsl:value-of select="author"/></dd>
        </dl>
    </xsl:template>
```

```
</xsl:stylesheet>
```

Then, the generated XML output will be XHTML as follows.

```
<html>
  <head>
    <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Book Info</title>
  </head>
  <body>
    <h1>Book Info</h1>
    <dl>
      <dt>Title:</dt>
      <dd> ZK - Ajax without the JavaScript Framework</dd>
      <dt>Who is this book for:</dt>
      <dd>Web application designers and programmers who wish to
implement
          rich Ajax web applications in the simplest way.</dd>
      <dt>Authors</dt>
      <dd> Henri Chen and Robbie Cheng</dd>
    </dl>
  </body>
</html>
```

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/xml/Transformer.html#>
- [2] <http://en.wikipedia.org/wiki/XSLT>

Annotation

This section describes how to add annotations on a component definition. When you annotate a component definition, all its instances will have the annotations. To annotate a component definition, you have to specify the annotations in a language definition.

Data Binding

Data binding synchronizes data between View and ViewModel according to component definition's annotation. The annotation specifies when to save (or load) which attribute, how to convert, validate and render the data. It can be found in `metainfo\zk\lang-addon.xml` of `zkbind.jar`. Please refer to ZK Client-side Reference/Language Definition about how to configure language definition and its addon. If you want data binding can works on your newly-created component, you should define its own annotations.

Annotation Attributes

Here is the attribute list used in `lang-addon.xml`:

Attribute Name	Description
ACCESS	Access privilege. The value can be "both", "save", or "load"(default value); default value is used if not specify.
CONVERTER	System converter for special properties. (optional) e.g. SelectedItem in listbox. see <code>ListboxSelectedItemConverter</code> [1]
VALIDATOR	System validator for special properties. (optional)
SAVE_EVENT	Save trigger event. It takes effect only when ACCESS attribute is "both" or "save".
LOAD_EVENT	Load trigger event; It takes effect only when ACCESS attribute is "both" or "load".
LOAD_REPLACEMENT	The replacement attribute for loading. It's used when there is a issue to load to original attribute.; e.g. value of textbox, it loads to "rawValue".
LOAD_TYPE	Type of attribute for loading; e.g. rawValue of textbox is <code>java.lang.String</code> .
SAVE_REPLACEMENT	The replacement attribute for saving. It's used when there is a issue to save to original attribute.; e.g. <code>selectedItem</code> of selectbox, it save the value <code>selectIndex</code> via converter to the bean. (<code>selectedItem</code> is not existed in selectbox).
RENDERER	A special renderer for binding

Example

Let's take a look at some examples.

Textbox's data binding annotation

```
<component>
  <component-name>textbox</component-name>
  <extends>textbox</extends>
  <annotation>
    <annotation-name>ZKBIND</annotation-name>
    <property-name>value</property-name>
    <attribute>
      <attribute-name>ACCESS</attribute-name>
```

```

        <attribute-value>both</attribute-value>
    </attribute>
    <attribute>
        <attribute-name>SAVE_EVENT</attribute-name>
        <attribute-value>onChange</attribute-value>
    </attribute>
    <attribute>
        <attribute-name>LOAD_REPLACEMENT</attribute-name>
        <attribute-value>rawValue</attribute-value>
    </attribute>
    <attribute>
        <attribute-name>LOAD_TYPE</attribute-name>
        <attribute-value>java.lang.String</attribute-value>
    </attribute>
</annotation>
</component>

```

- We use `<extends>` to append data binding annotations based on original component definition. If you write these annotation attributes in a component definition file (`metainfo\zk\lang.xml`), you don't have to use `<extends>`.
- **ZKBIND** is zkbind system's annotation name. It means we use annotation of data binding 2.
- The `<property-name>` is the target property we want data binding to synchronize.
- Because Textbox is a component for input, its access privilege is "both".
- As the `SAVE_EVENT` is `onChange`, you can find when your cursor focus on a Textbox is blurred, Textbox's value is saved.
- Here we use `LOAD_REPLACEMENT` for a "loading empty value" issue related to "constraint" attribute. It is common that when we load textbox first time, its value is empty. If we also define constraint as "not empty", then the first time loading triggers error message to display. This error misleads users, so we use another replacement attribute to load the value for not triggering error message.

Selectbox's data binding annotation

```

<component>
    <component-name>selectbox</component-name>
    <extends>selectbox</extends>
    <annotation>
        <annotation-name>ZKBIND</annotation-name>
        <attribute>
            <attribute-name>RENDERER</attribute-name>
            <attribute-value>itemRenderer=org.zkoss.bind.impl.BindSelectboxRenderer</attribute-value>
        </attribute>
    </annotation>
    <annotation>
        <annotation-name>ZKBIND</annotation-name>
        <property-name>selectedItem</property-name>
        <attribute>
            <attribute-name>SAVE_EVENT</attribute-name>
            <attribute-value>onSelect</attribute-value>
        </attribute>
    </annotation>
</component>

```

```
</attribute>

<attribute>
    <attribute-name>LOAD_EVENT</attribute-name>
    <attribute-value>onAfterRender</attribute-value>
</attribute>

<attribute>
    <attribute-name>ACCESS</attribute-name>
    <attribute-value>both</attribute-value>
</attribute>

<attribute>
    <attribute-name>SAVE_REPLACEMENT</attribute-name>
    <attribute-value>selectedIndex</attribute-value>
</attribute>

<attribute>
    <attribute-name>CONVERTER</attribute-name>

    <attribute-value>org.zkoss.bind.converter.sys.SelectboxSelectedItemConverter</attribute-value>
</attribute>
</annotation>
```

- We set RENDERER to render selectbox's item when model data comes from data binding. (line 7,8)
- Normally users should specify when to reload properties. But if you think the property requires reloading after a specific event, you can set that event's name as value of LOAD_EVENT. (line 19,20)
- The reason we use SAVE_REPLACEMENT is that Selectbox has no "selectedItem" attribute, so we save to another replacement attribute "selectedIndex". (line 27,28) Because those 2 attributes have different type, we have to use a converter, SelectboxSelectedItemConverter^[2] to convert selectedItem. (line 31,32)

Version History

Version	Date	Content
---------	------	---------

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/bind/converter/sys/ListboxSelectedItemConverter.html#>
[2] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/bind/converter/sys>SelectboxSelectedItemConverter.html#>

Tablet Devices

This section describes the feature and the enhancement for tablet devices.

[Since ZK 6.5.0]

Configuration

This section outline configurations which are used for tablet devices.

Viewport

Viewport Content

The default viewport content generated in ZK when using tablet device is:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no" />
```

Customize

User can specify viewport setting by library properties or XML processing instruction.

Library Properties

since 6.5.0

The default value of org.zkoss.zul.tablet.meta.viewport.disabled property is `false`. If it is set to `true`, ZK won't render viewport content for all pages. Thus, user can specify viewport on each ZUL page by meta instruction.

XML processing instruction

since 6.5.0

User can keep the default setting and assign custom viewport on specific pages by overwriting viewport content in page instruction as follows:

```
<?page viewport="width=device-width, initial-scale=1.0"?>
```

Version History

Version	Date	Content
6.5.0	September, 2012	new added configuration

Components

This section outline components which are used for tablet devices.

ScrollView

ScrollView

- Demonstration: N/A
- Java API: ScrollView ^[1]
- JavaScript API: ScrollView ^[2]
- Style Guide: N/A
- Available in ZK EE only ^[13]

Employment/Purpose

since 6.5

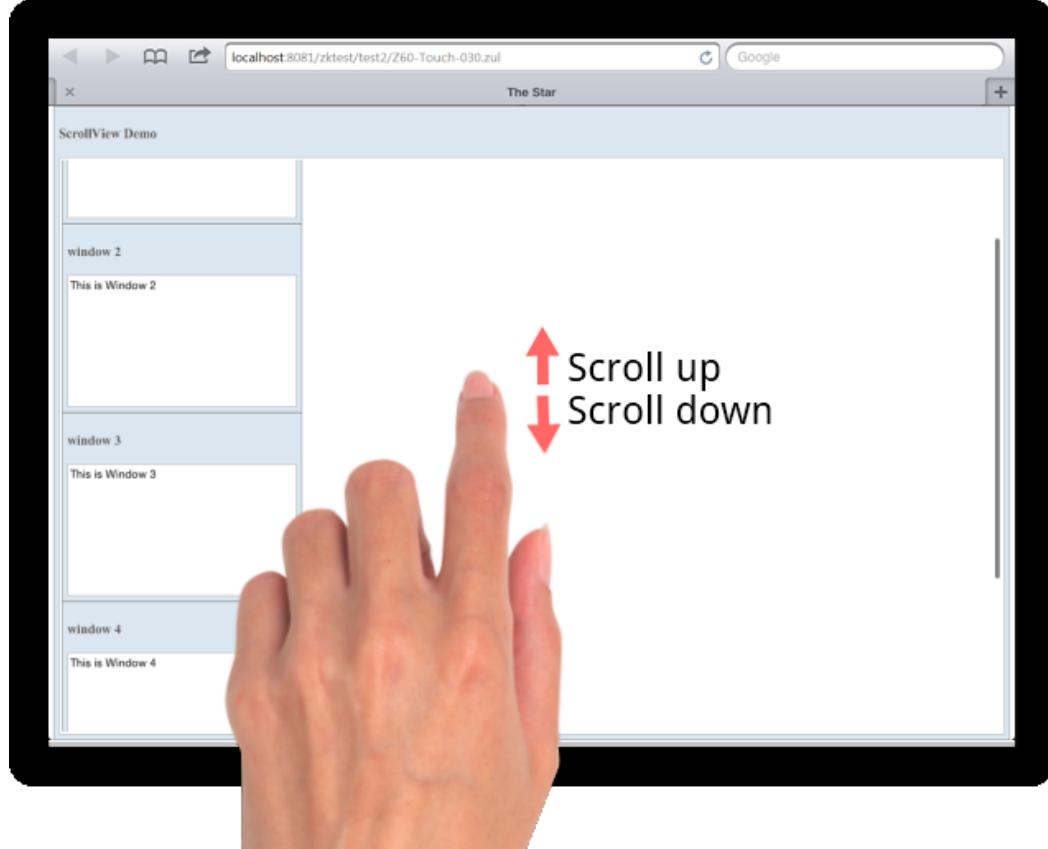
Applications exceeding a browser's viewport relies on desktop browsers to automatically generates scrollbars by using CSS **overflow** attribute so users can view the entire page but this feature, unfortunately, is not available on tablet or mobile browsers.

ScrollView is a container that makes its content scrollable on tablet/mobile device with a **rubber band** effect you see on iPhone/iPad. When you attempt to scroll past the end of the content area, the area still moves slowly with your fingers. Since there is no more content, then the area will bounce back to the correct position after your fingers leave the screen.

When end users access your application via a tablet or any mobile devices, ScrollView detects and triggers this feature. Oppositely, if the application is accessed via a desktop browser, ScrollView remains un-triggered while desktop browser will automatically generate scrollbars when needed just like before.

Example

You can scroll up/down to see other window components with the following sample ZUL.



```
<scrollview vflex="1" hflex="1">
    <zk forEach="#1,2,3,4,5">
        <window title="window ${| class='wikitable' | width='100%'">
            This is Window ${each}
        </window>
    </zk>
</scrollview>
```

Properties

Orient

The default orient of child components inside Scrollview is **vertical**. You can also change it to **horizontal**.

```
<scrollview vflex="1" hflex="1" orient="horizontal" />
```

Supported Events

Name	Event Type
onScroll	Event: ScrollEvent [5] In mobile devices: Denotes that the content of a scrollable component has been scrolled by the user. Notice that you can check if it is scrolled outside/inside boundaries by invoking <code>getOutOfBound</code> method in the ScrollEvent. In desktop: This event will be triggered when users scroll all the way to the top or to the end of the page.
onScrolling	Event: ScrollEvent [5] Denotes that the user is scrolling a scrollable component.

- Inherited Supported Events: XulElement

Supported Children

*ALL

Use Cases

Version	Description	Example Location

Version History

Version	Date	Content
6.5.0	August, 2012	new added component

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zkmax/zul/Scrollview.html#>
[2] <http://www.zkoss.org/javadoc/latest/jsdoc/zkmax/layout/Scrollview.html#>

Events

This section outline events which are used in components for tablet devices.

SwipeEvent

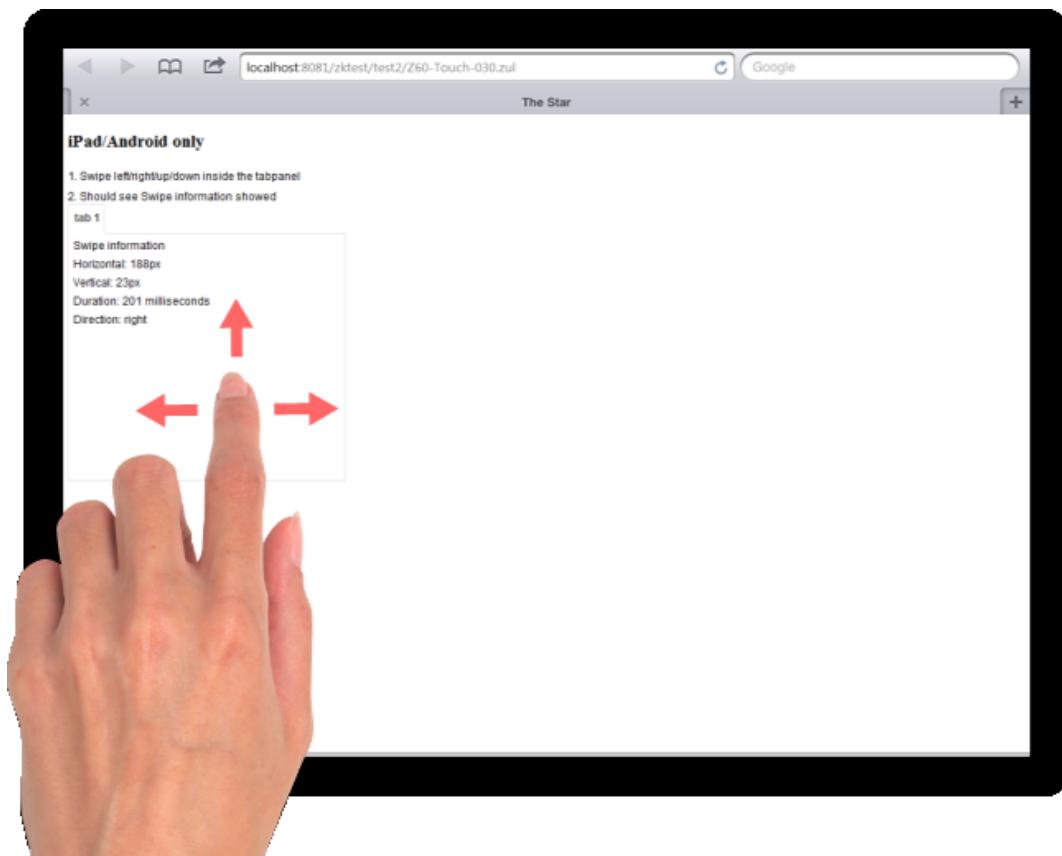
SwipeEvent

- Demonstration: N/A
- Java API: SwipeEvent^[1]
- JavaScript API: N/A

Employment/Purpose

Represents an event that indicates swipe on a component and provides information about the swipe displacement, duration, and direction.

Example



The swipe event can be registered in any component which can be swiped by user on tablet devices.

```
<tabbox height="300px" width="300px">
    <tabs>
        <tab label="tab 1" />
    </tabs>
```

```

<tabpanels>
    <tabpanel>
        <attribute name="onSwipe"><![CDATA[
            SwipeEvent se = (SwipeEvent) event;
            lbl1.setValue("Horizontal: " + se.getSwipeX() +
            "px");
            lbl2.setValue("Vertical: " + se.getSwipeY() +
            "px");
            lbl3.setValue("Duration: " +
            se.getSwipeDuration() + " milliseconds");
            lbl4.setValue("Direction: " +
            se.getSwipeDirection());
        ]]></attribute>
        <vlayout>
            Swipe information
            <label id="lbl1"/>
            <label id="lbl2"/>
            <label id="lbl3"/>
            <label id="lbl4"/>
        </vlayout>
    </tabpanel>
</tabpanels>
</tabbox>

```

Supported events

Name	Event Type
None	None

Supported Children

*NONE

Use cases

Version	Description	Example Location

Version History

Version	Date	Content
6.5.0	July, 2012	ZK Client Widget support swipe event for tablet/mobile device ^[2]

References

[1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/SwipeEvent.html#>

[2] <http://tracker.zkoss.org/browse/ZK-1241>

ClientInfoEvent

ClientInfoEvent

- Demonstration: N/A
- Java API: ClientInfoEvent ^[1]
- JavaScript API: N/A

Employment/Purpose

The onClientInfo event is used to notify the client's information, such as orientation change, time zone and screen resolutions.

Note: All root components of all pages of the desktop will receives this event.

Example

To register this event can resize the layout to fit the whole screen for user.

```
<tabbox id="tbx" height="400px" width="600px">
    <attribute name="onClientInfo"><![CDATA[
        ClientInfoEvent oe = (ClientInfoEvent) event;
        String orient = oe.getOrientation();
        lbl.setValue(orient);
        if (orient.equals("portrait")) {
            tbx.setHeight("600px");
            tbx.setWidth("400px");
        } else {
            tbx.setHeight("400px");
            tbx.setWidth("600px");
        }
    ]]></attribute>
    <tabs>
        <tab label="tab 1" />
    </tabs>
    <tabpanel>
```

```
Current Orientation: <label id="lbl"/>
</tabpanel>
</tabpanel>
</tabbox>
```

Supported events

Name	Event Type
None	None

Supported Children

*NONE

Use cases

Version	Description	Example Location

Version History

Version	Date	Content
6.5.0	July, 2012	Add a way to listen onOrientationChange for tablet device [2]

References

- [1] <http://www.zkoss.org/javadoc/latest/zk/org/zkoss/zk/ui/event/ClientInfoEvent.html#>
- [2] <http://tracker.zkoss.org/browse/ZK-1273>

UI Enhancements

This section outline UI enhancements, new features and the better user-experiences, in components for tablet devices.

Note: the following subsections are only applied for ZK EE version only.

Borderlayout

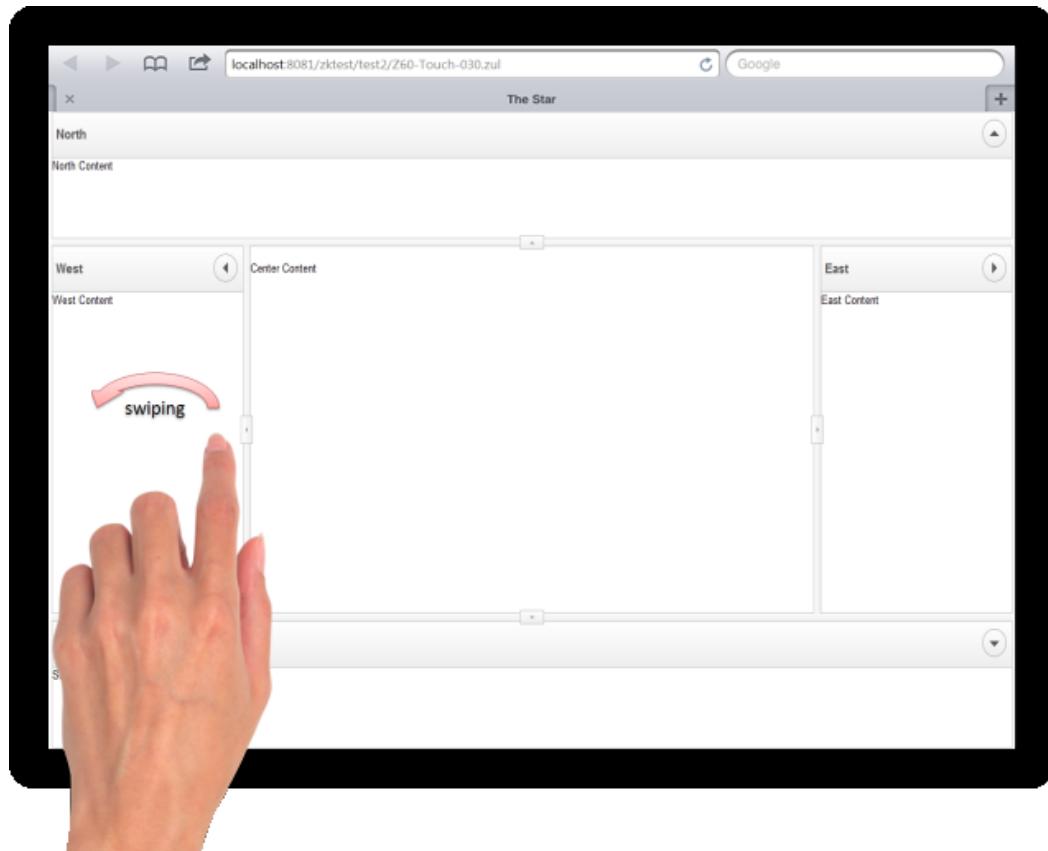
Borderlayout

- Component Reference: Borderlayout
- Available in ZK EE only ^[13]

SwipeEvent Support

Each layout region can support to close and open the region area by user's swipe on the edge of the region with client/attribute.

```
<borderlayout xmlns:ca="client/attribute" ca:data-swipeable="true">
<!-- omitted -->
</borderlayout>
```

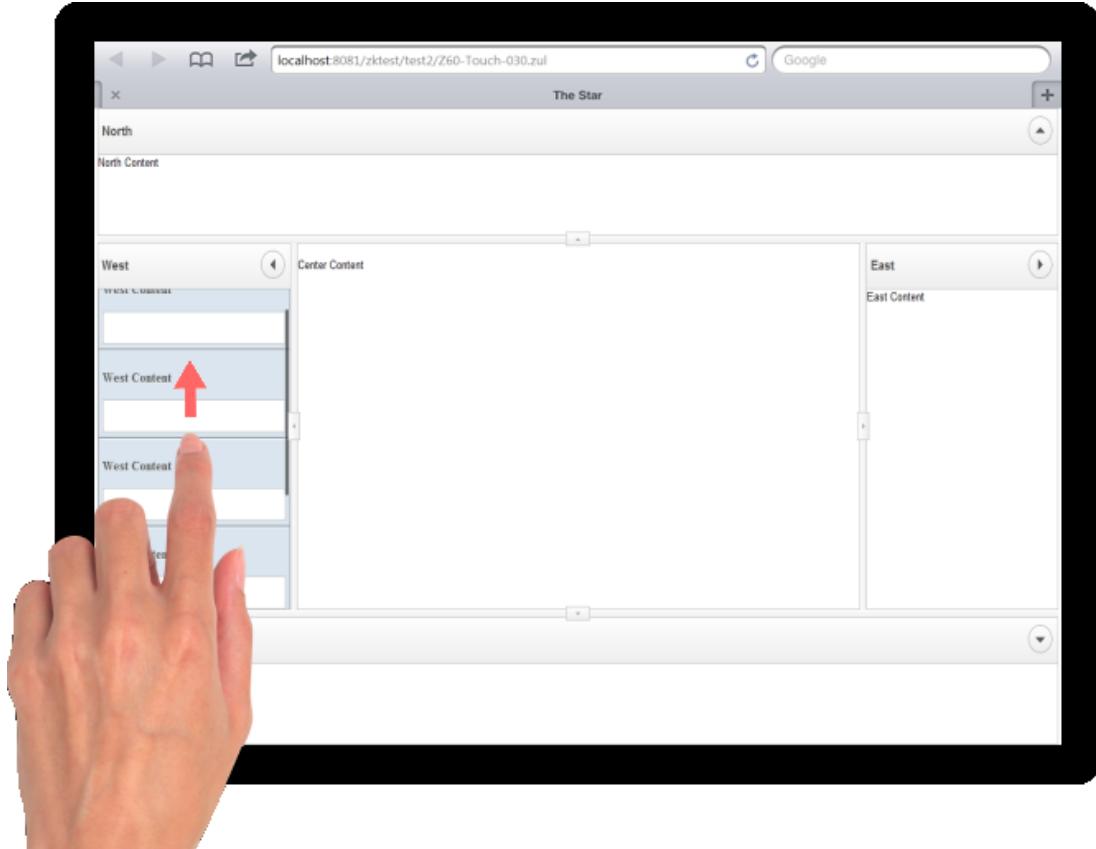


Friendly Scrolling Support

When user swipe on the content of Layout Region from Borderlayout, the friendly scrollbar will appear. To enable the friendly scrollbar, please specify autoscroll to true.

For example,

```
<west title="West" size="20%" autoscroll="true" />
```



Note: to disable the friendly scrollbar, please use the following setting.

```
<west autoscroll="false"/>
```

Version History

Version	Date	Content
6.5.0	July, 2012	Borderlayout support touch's swipe event to close/open the layout region ^[1]

References

[1] <http://tracker.zkoss.org/browse/ZK-1245>

Calendar

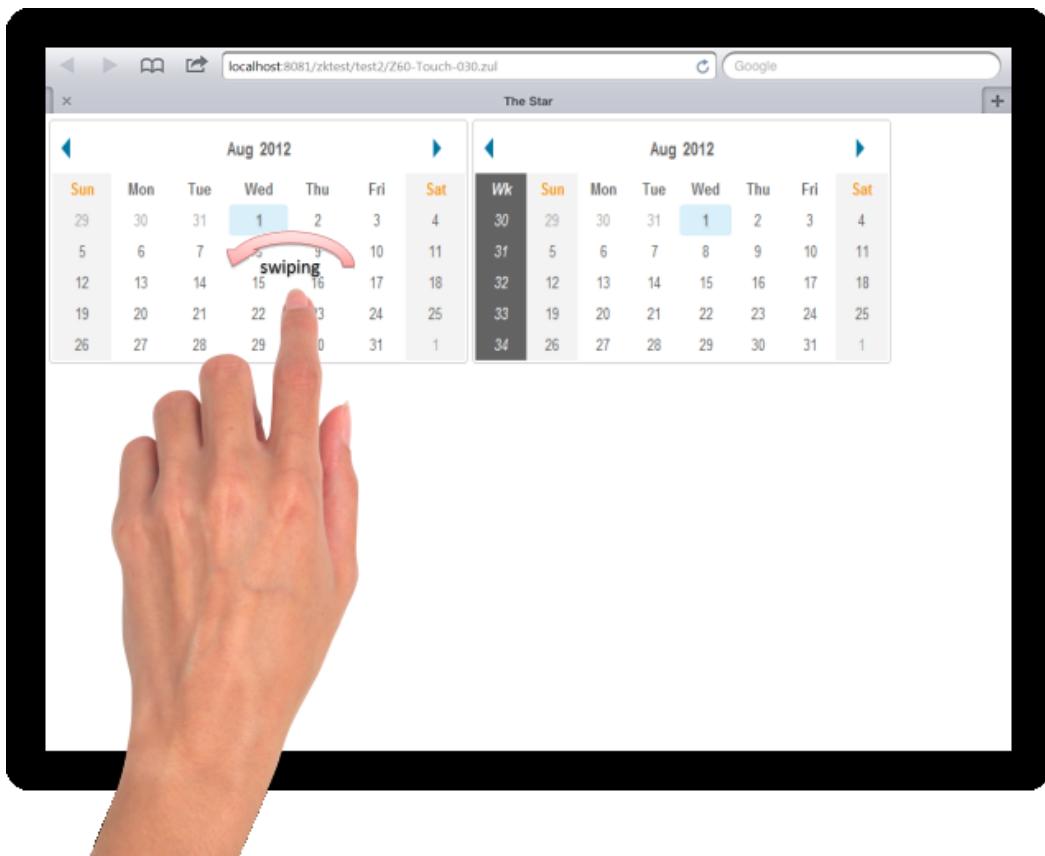
Calendar

- Component Reference: Calendar
- Available in ZK EE only^[13]

Friendly Scrolling Support

Calendar can support to switch the view by user swipe on the content with client/attribute.

```
<calendar xmlns:ca="client/attribute" ca:data-swipeable="true" />
```



Version History

Version	Date	Content
6.5.0	July, 2012	Calendar support touch's swipe event to switch the date view ^[1]

References

[1] <http://tracker.zkoss.org/browse/ZK-1246>

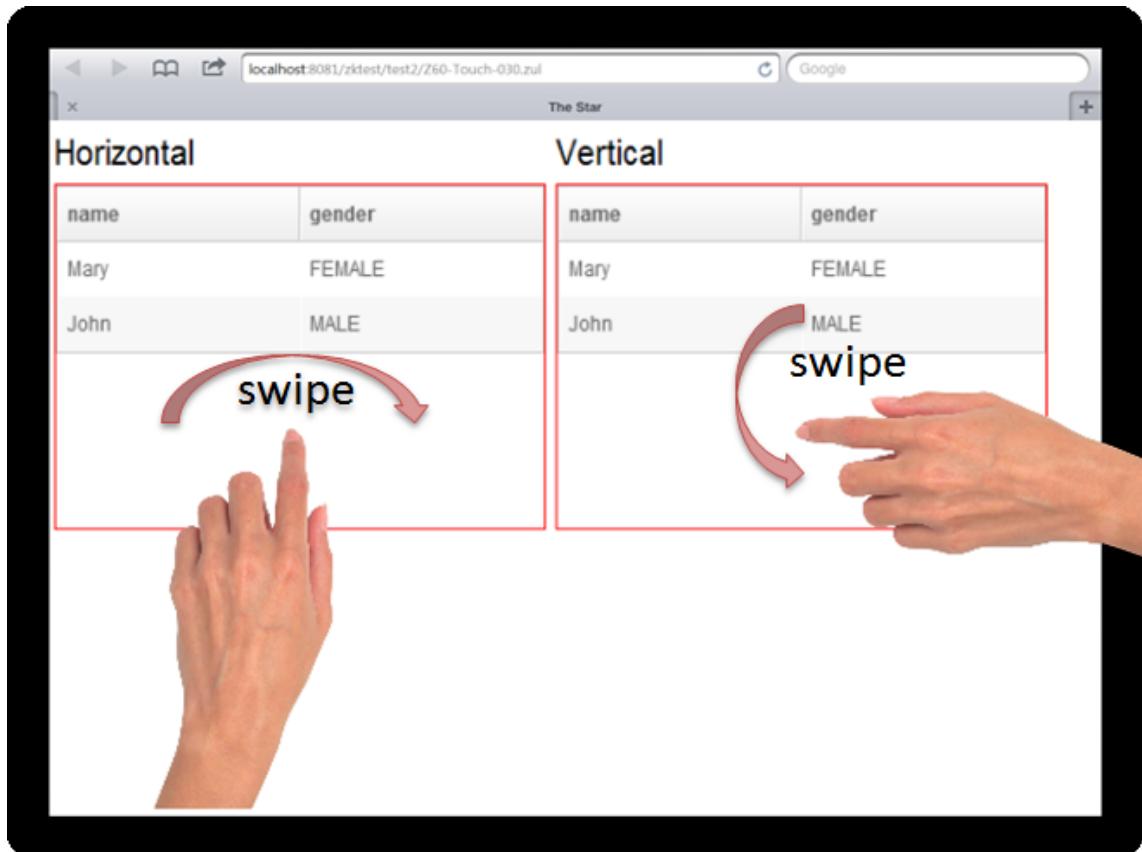
Cardlayout

Cardlayout

- Component Reference: Cardlayout
- Available in ZK EE only^[13]

Swipe to change view

Cardlayout supports the navigation previous or next component by simply swiping on the tablet.



Version History

Version	Date	Content
6.5.0	August, 2012	

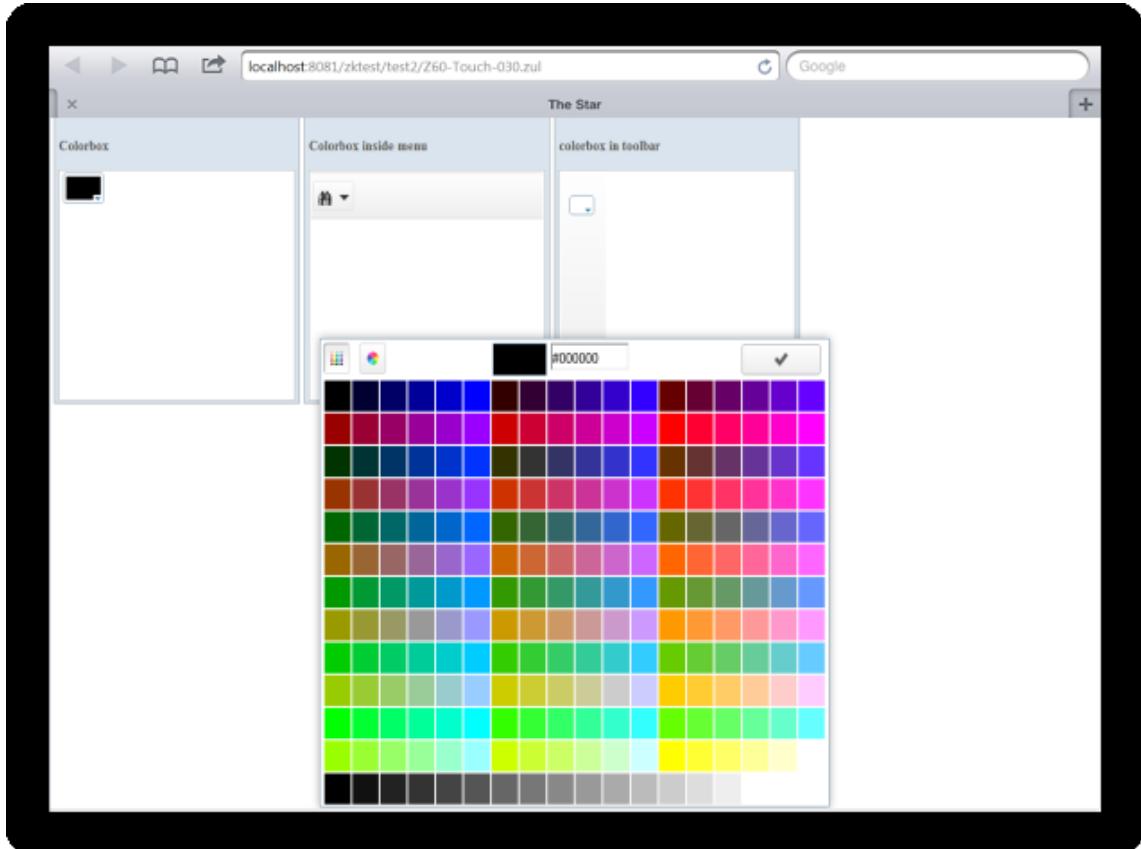
Colorbox

Colorbox

- Component Reference: Colorbox
- Available in ZK EE only^[13]

Layout Enhancement

Big icons for tablet devices



Version History

Version	Date	Content
6.5.0	July, 2012	Add a new theme for tablet/mobile device with big icons ^[1]

References

[1] <http://tracker.zkoss.org/browse/ZK-1247>

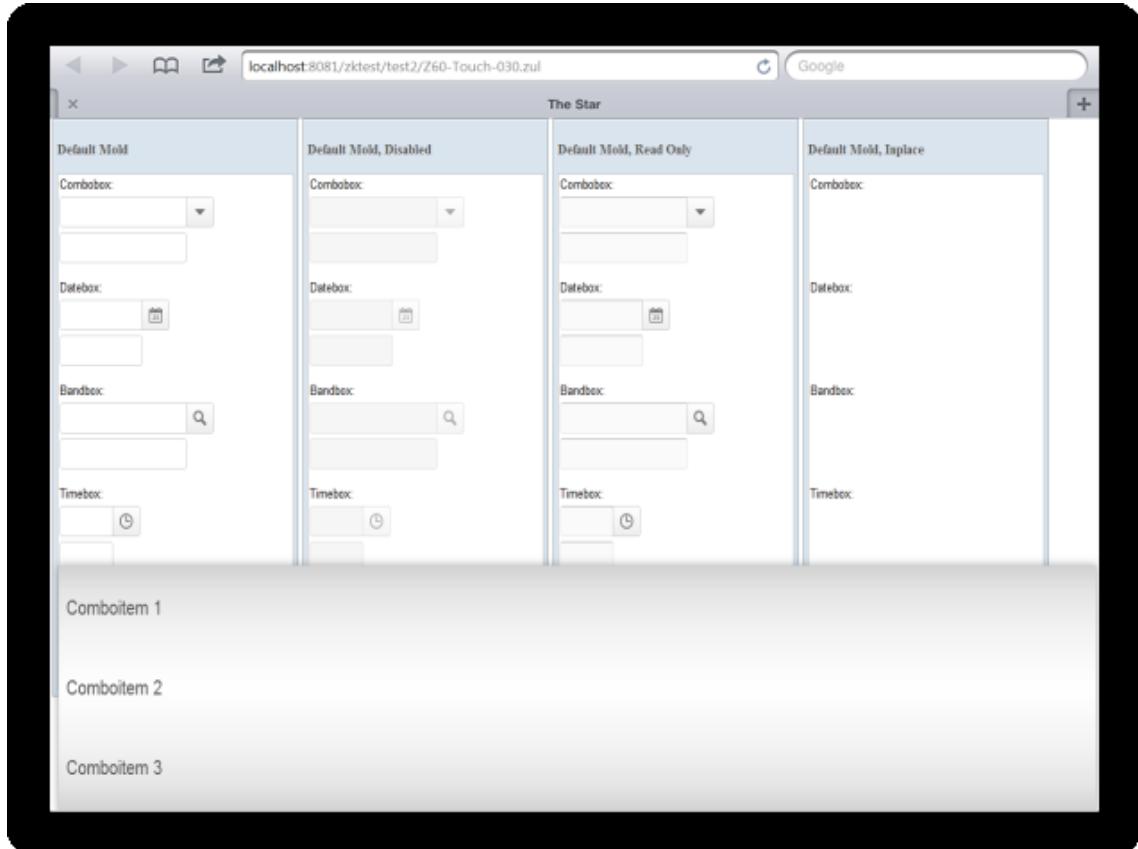
Combobox

Combobox

- Component Reference: Combobox
- Available in ZK EE only^[13]

Layout Enhancement

Big icons for tablet devices



Version History

Version	Date	Content
6.5.0	July, 2012	Add a new theme for tablet/mobile device with big icons ^[1]

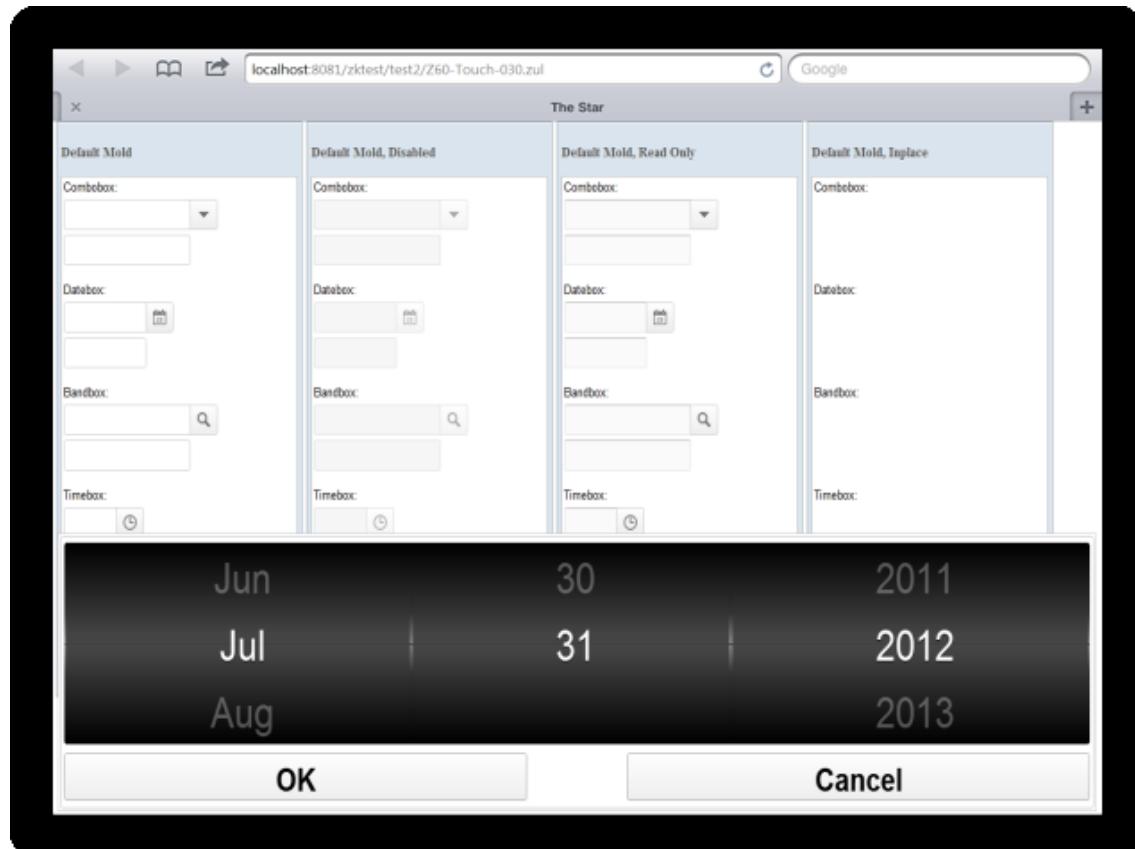
Datebox

Datebox

- Component Reference: Datebox
- Available in ZK EE only^[13]

Layout Enhancement

Big icons for tablet devices



Version History

Version	Date	Content
6.5.0	July, 2012	Add a new theme for tablet/mobile device with big icons ^[1]

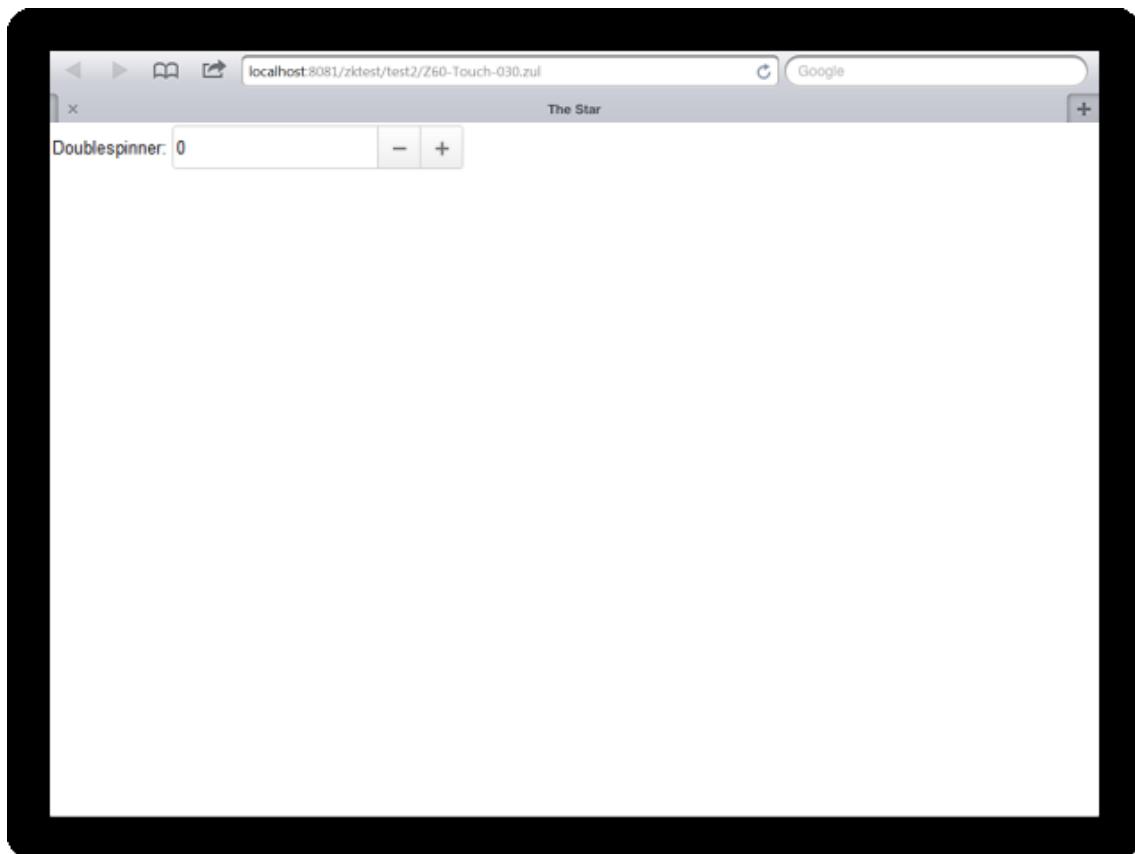
Doublespinner

Doublespinner

- Component Reference: Doublespinner
- Available in ZK EE only ^[13]

Layout Enhancement

Big icons for tablet devices



Unsupported API

Format

In Tablet device, the format of the input element with number type has no support.

Locale

In Tablet device, the locale of the input element with number type has no support.

Version History

Version	Date	Content
---------	------	---------

6.5.0	July, 2012	Add a new theme for tablet/mobile device with big icons [1]
-------	------------	---

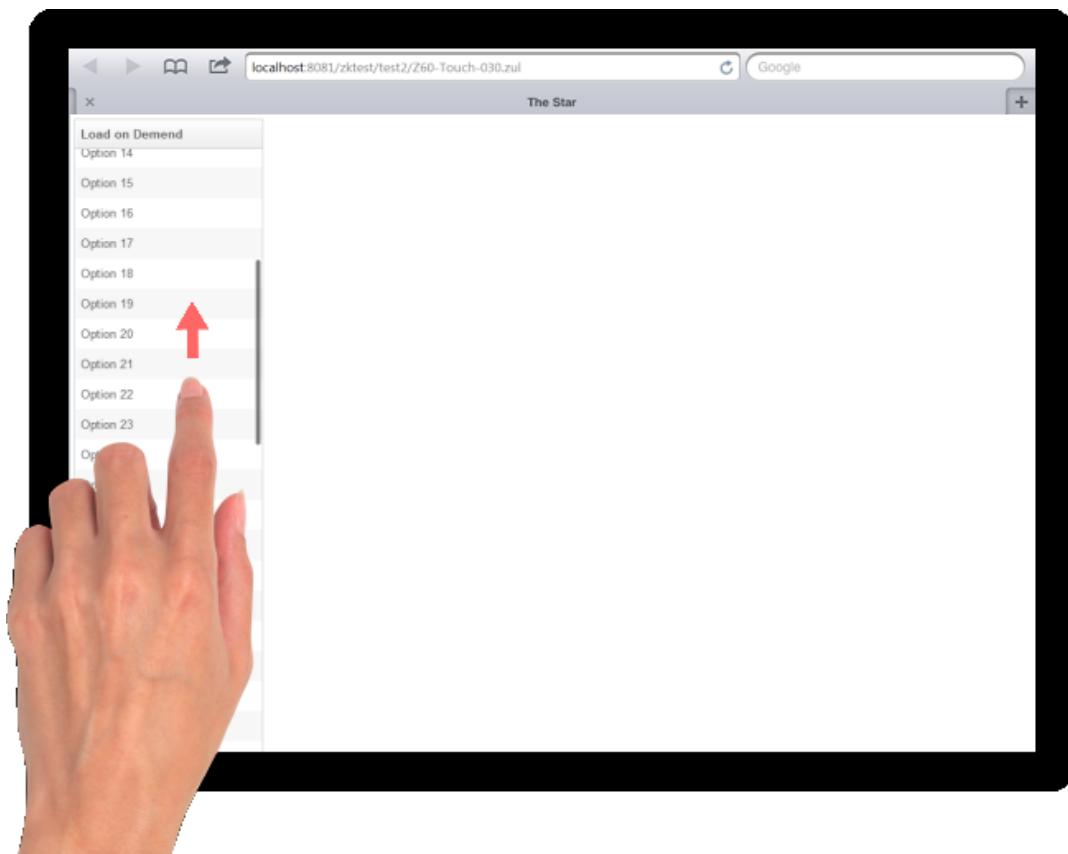
Grid

Grid

- Component Reference: Grid
- Available in ZK EE only [13]

Friendly Scrolling Support

When user swipe on the content of Grid, the friendly scrollbar will appear.



Note 1: to disable the friendly scrollbar, please use the following setting.

```
<grid xmlns:a="client/attribute" a:data-scrollable="false"/>
```

Note 2: to make sure the friendly scrollbar works correctly when containing images inside cell, please add image preload attribute as follows.

```
<grid>
  <custom-attributes org.zkoss.zul.image.preload="true"/>
</grid>
```

Version History

Version	Date	Content
6.5.0	July, 2012	Grid, Listbox, Tree can support to scroll the content by finger on tablet/mobile device ^[1]

References

[1] <http://tracker.zkoss.org/browse/ZK-1239>

Groupbox

Groupbox

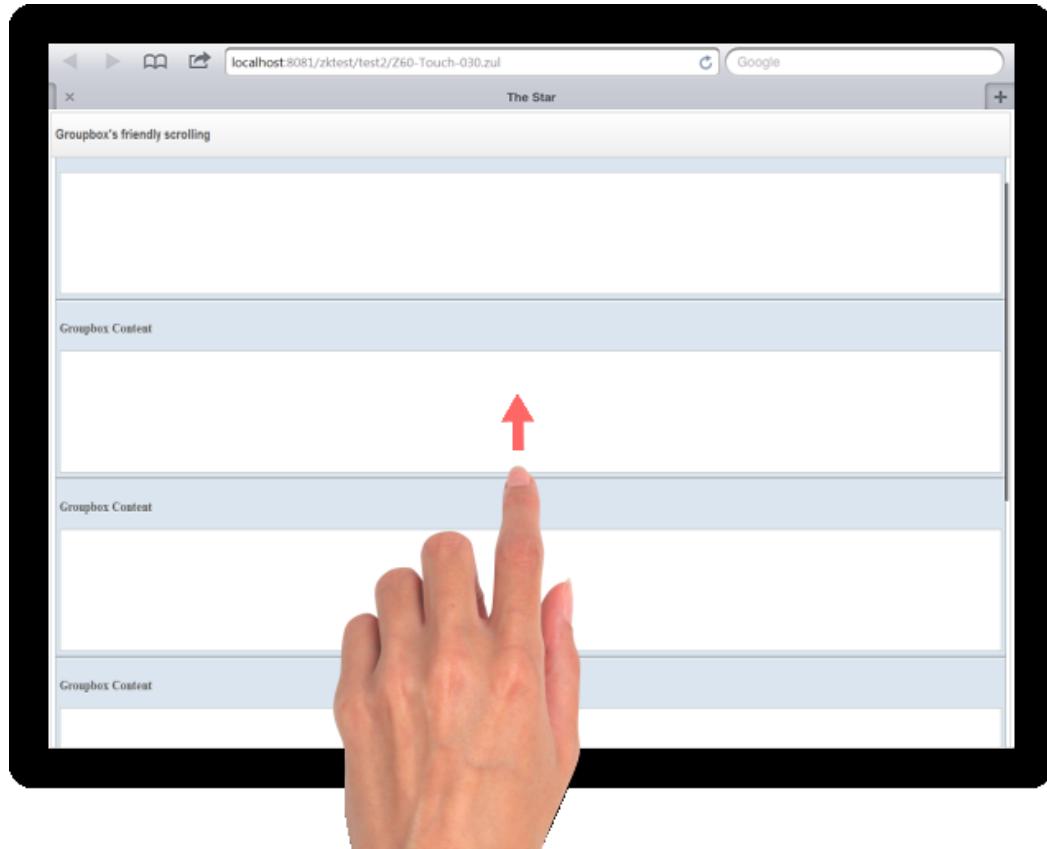
- Component Reference: Groupbox
- Available in ZK EE only ^[13]

Friendly Scrolling Support

When user swipe on the content of Groupbox, the friendly scrollbar will appear. To enable this, please specify the contentStyle with "overflow:auto"

For example,

```
<groupbox contentStyle="overflow:auto">
```



Version History

Version	Date	Content
6.5.0	July, 2012	

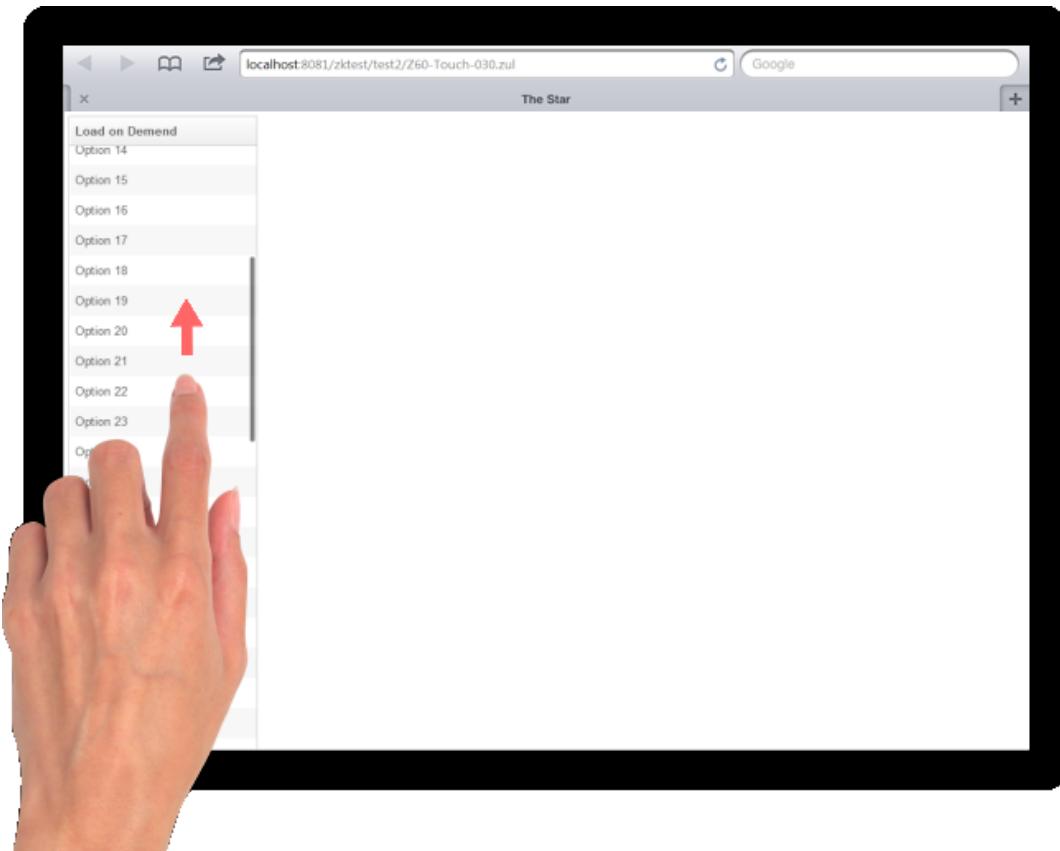
Listbox

Listbox

- Component Reference: Listbox
- Available in ZK EE only ^[13]

Friendly Scrolling Support

When user swipe on the content of Listbox, the friendly scrollbar will appear.



Note 1: to disable the friendly scrollbar, please use the following setting.

```
<listbox xmlns:a="client/attribute" a:data-scrollable="false"/>
```

Note 2: to make sure the friendly scrollbar works correctly when containing images inside listcell, please add image preload attribute as follows.

```
<listbox>
    <custom-attributes org.zkoss.zul.image.preload="true"/>
</listbox>
```

Version History

Version	Date	Content
6.5.0	July, 2012	Grid, Listbox, Tree can support to scroll the content by finger on tablet/mobile device ^[1]

Paging

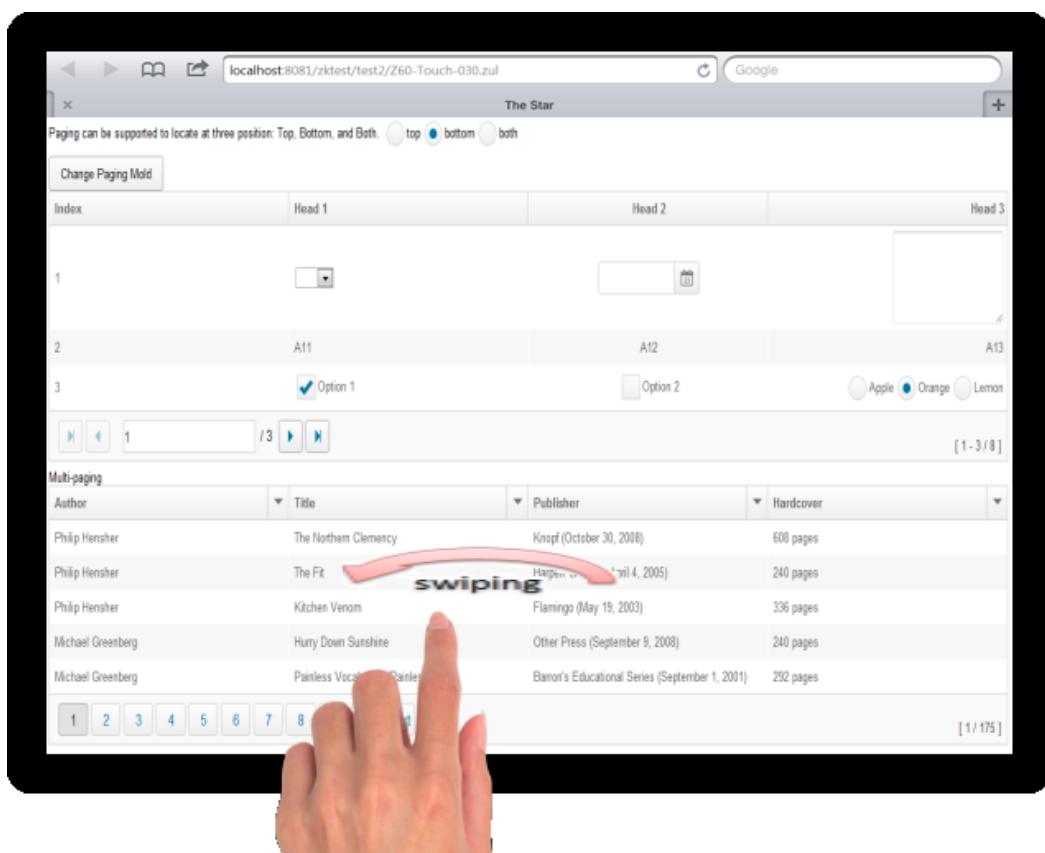
Paging

- Component Reference: Paging
- Available in ZK EE only ^[13]

SwipeEvent Support

Within Tree, Grid, and Listbox, Paging can support to navigate previous page or next page by user swipe on the edge of content with client attribute.

```
<listbox mold="paging" pageSize="5" xmlns:ca="client/attribute" ca:data-swipeable="true"></listbox>
```



Version History

Version	Date	Content
6.5.0	July, 2012	Grid/List/Tree support changing page by swipe event on tablet device ^[1]

References

[1] <http://tracker.zkoss.org/browse/ZK-1283>

Panel

Panel

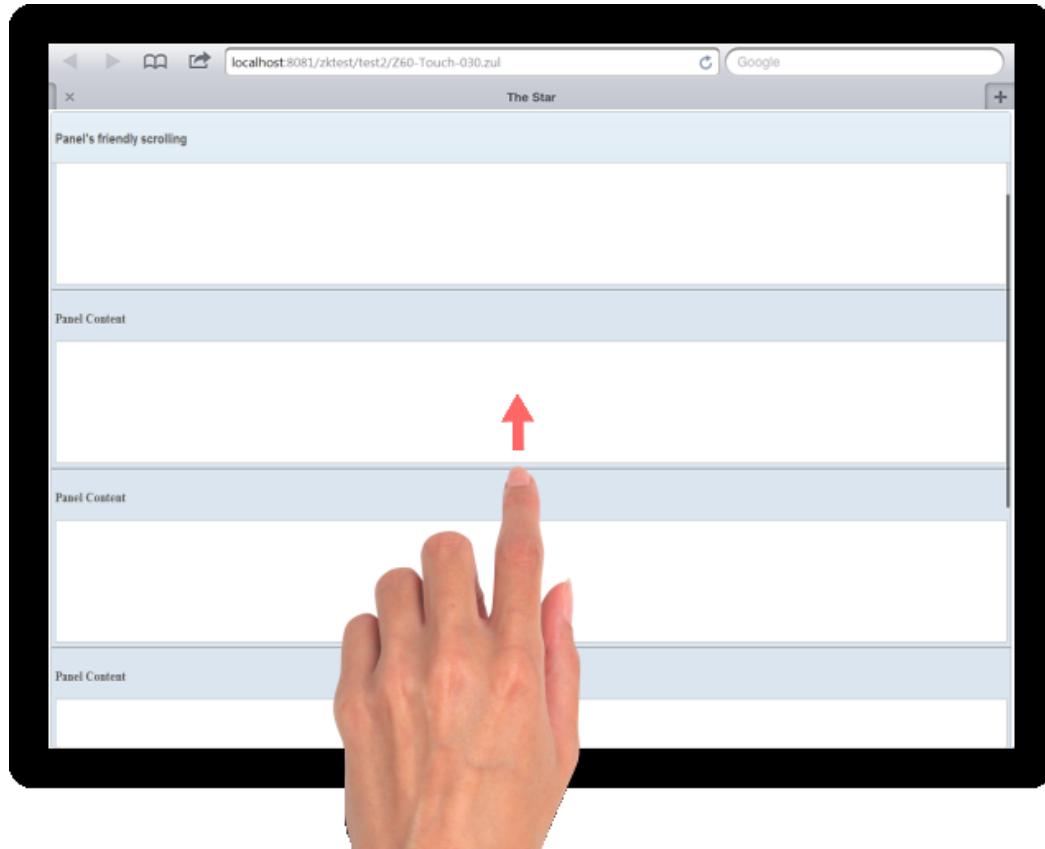
- Component Reference: Panel
- Available in ZK EE only ^[13]

Friendly Scrolling Support

When user swipe on the content of Panel, the friendly scrollbar will appear. To enable this, please specify the style with "overflow:auto"

For example,

```
<panelchildren style="overflow:auto">
```



Version History

Version	Date	Content
6.5.0	July, 2012	

Tabbox

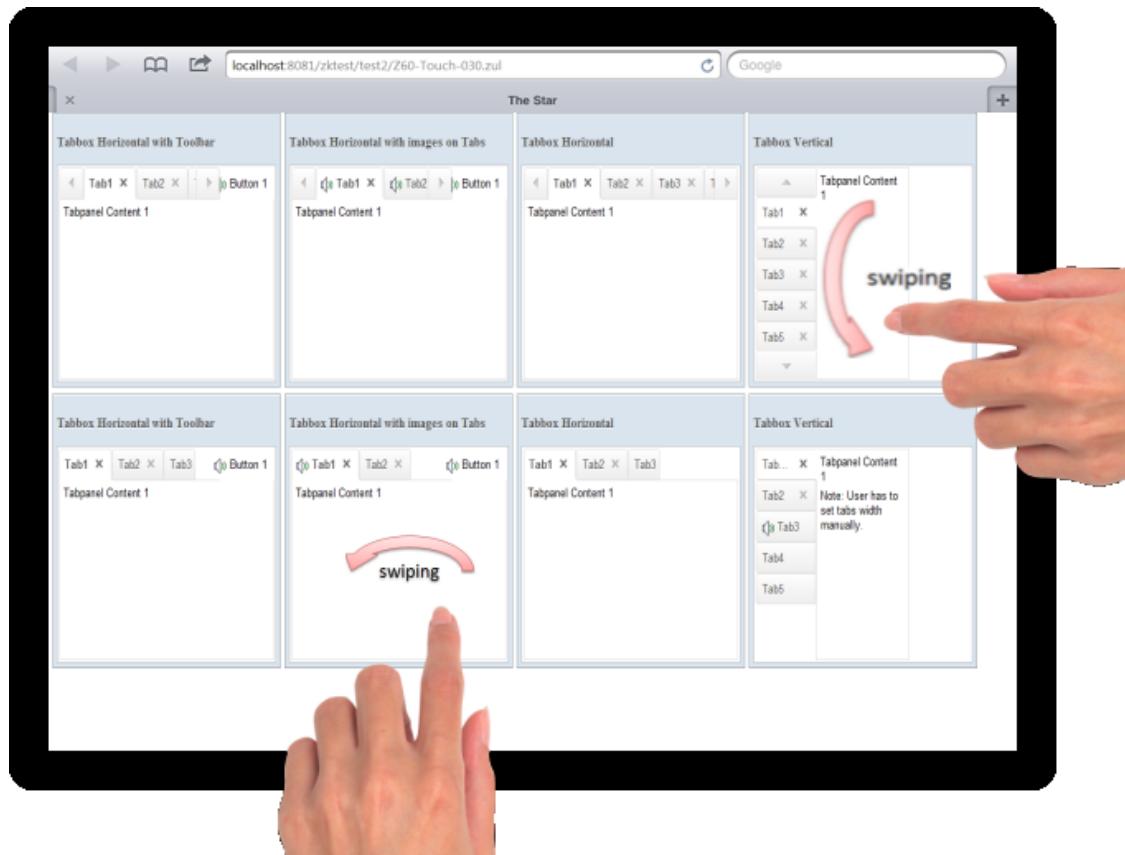
Tabbox

- Component Reference: Tabbox
- Available in ZK EE only ^[13]

SwipeEvent Support

Tabbox support to switch the tab by user swipe on the edge of content with client attribute.

```
<tabbox xmlns:ca="client/attribute" ca:data-swipeable="true">
    <!-- omitted -->
</tabbox>
```

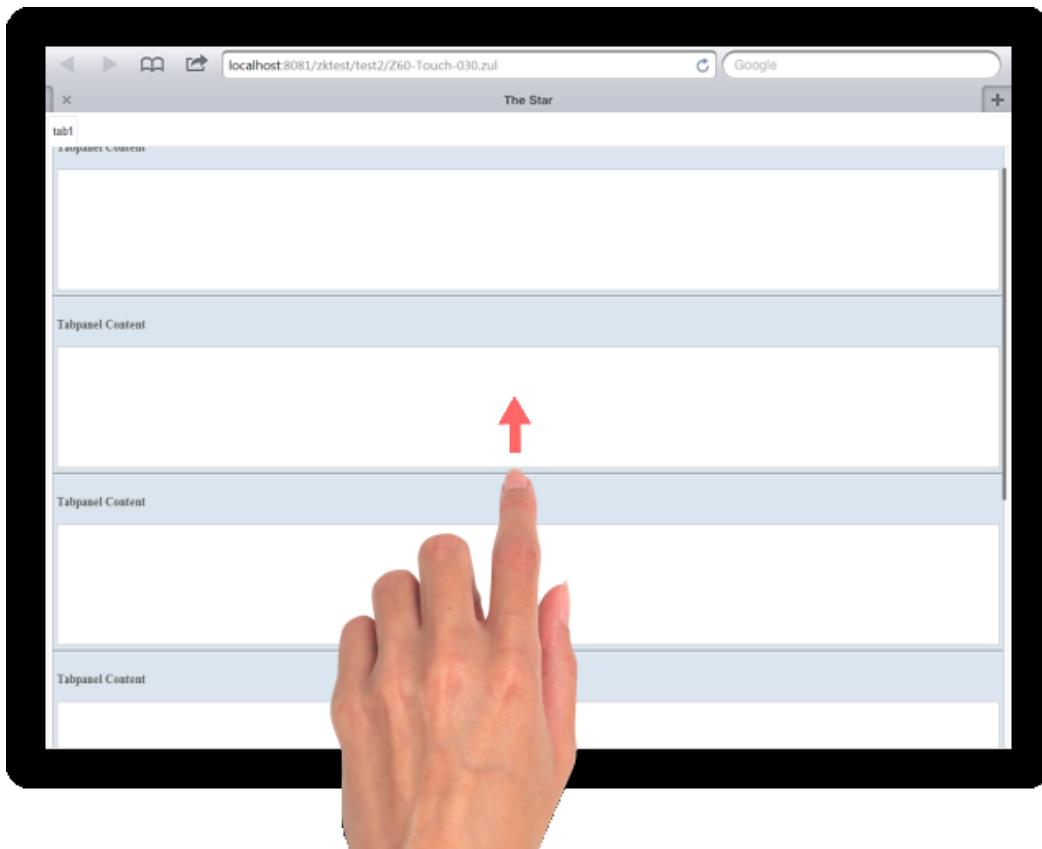


Friendly Scrolling Support

When user swipe on the content of Tabpanel, the friendly scrollbar will appear. To enable the friendly scrollbar, please specify the overflow style to auto.

For example,

```
<tabpanel style="overflow:auto">
```



Version History

Version	Date	Content
6.5.0	July, 2012	Tabbox support touch's swipe event to switch tab selection [1]

References

[1] <http://tracker.zkoss.org/browse/ZK-1244>

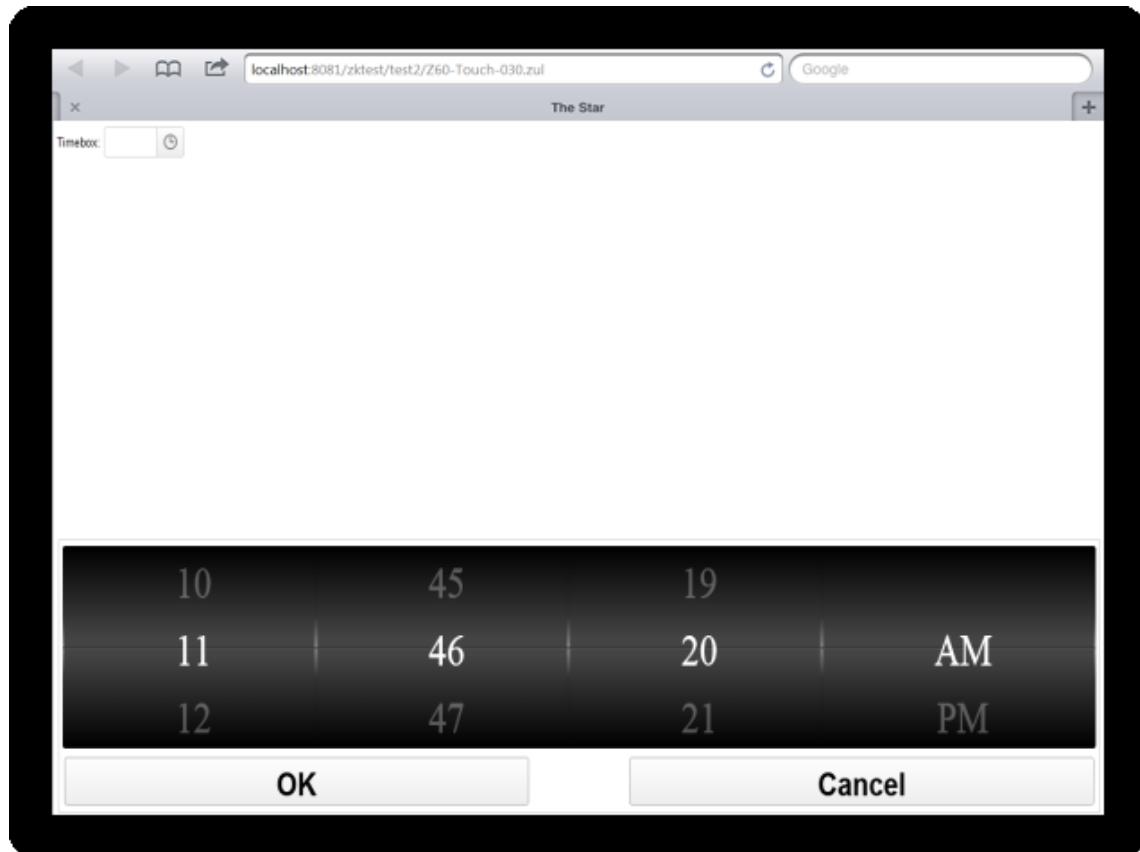
Timebox

Timebox

- Component Reference: Timebox
- Available in ZK EE only ^[13]

Layout Enhancement

Big icons for tablet devices



Version History

Version	Date	Content
6.5.0	July, 2012	Add a new theme for tablet/mobile device with big icons ^[1]

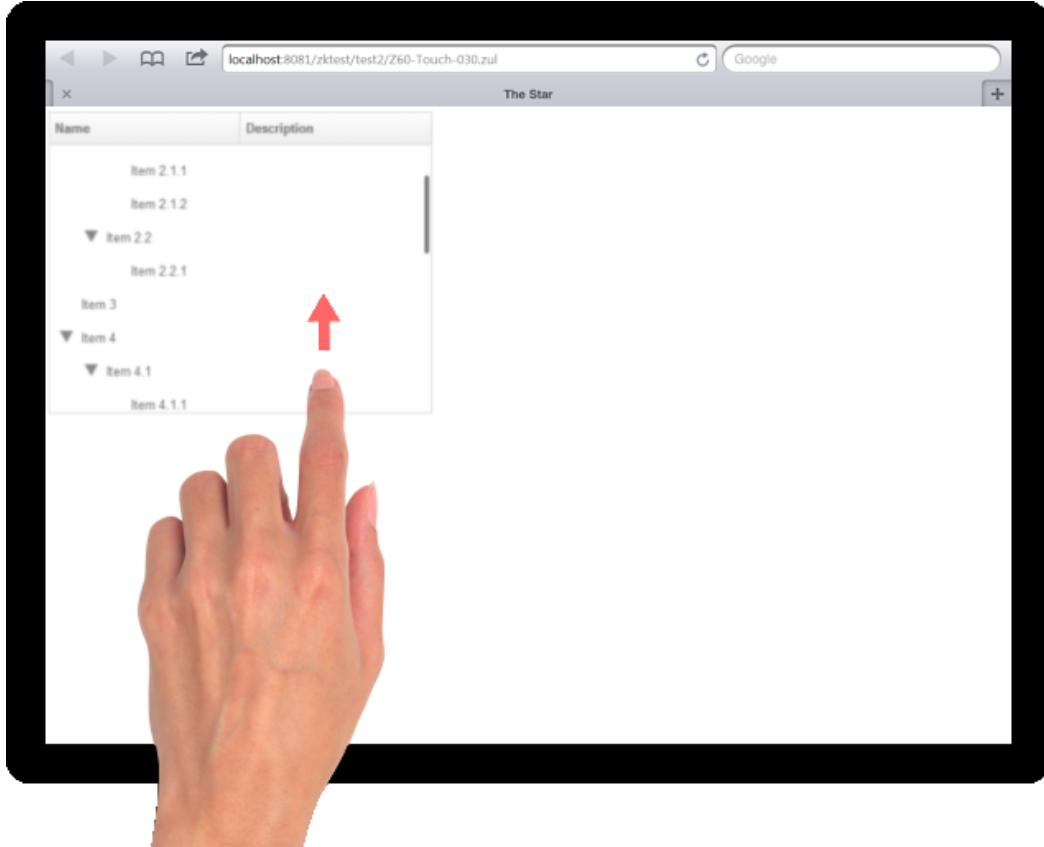
Tree

Tree

- Component Reference: Tree
- Available in ZK EE only ^[13]

Friendly Scrolling Support

When user swipe on the content of Tree, the friendly scrollbar will appear.



Note 1: to disable the friendly scrollbar, please use the following setting.

```
<tree xmlns:a="client/attribute" a:data-scrollable="false"/>
```

Note 2: to make sure the friendly scrollbar works correctly when containing images inside treecell, please add image preload attribute as follows.

```
<tree>
    <custom-attributes org.zkoss.zul.image.preload="true"/>
</tree>
```

Version History

Version	Date	Content
6.5.0	July, 2012	Grid, Listbox, Tree can support to scroll the content by finger on tablet/mobile device ^[1]

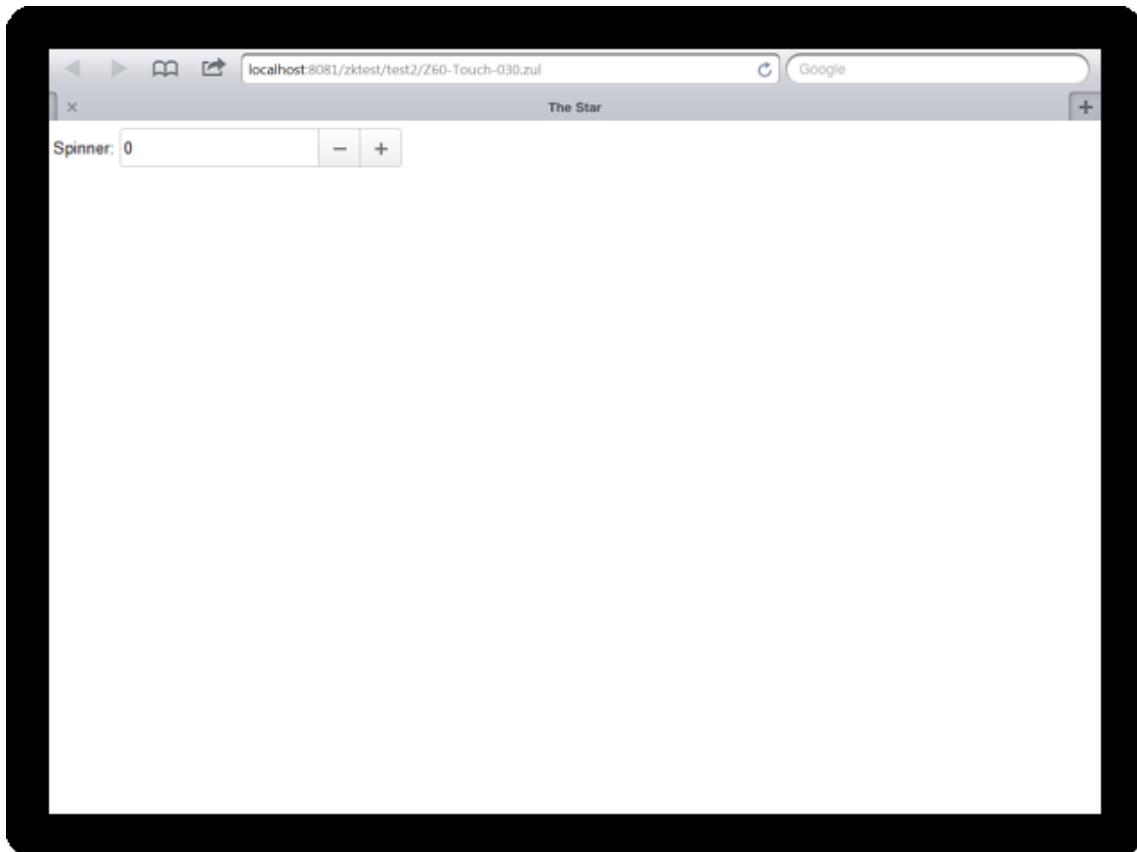
Spinner

Spinner

- Component Reference: Spinner
- Available in ZK EE only ^[13]

Layout Enhancement

Big icons for tablet devices



Unsupported API

Format

In Tablet device, the format of the input element with number type has no support.

Locale

In Tablet device, the locale of the input element with number type has no support.

Version History

Version	Date	Content
6.5.0	July, 2012	Add a new theme for tablet/mobile device with big icons ^[1]

Window

Window

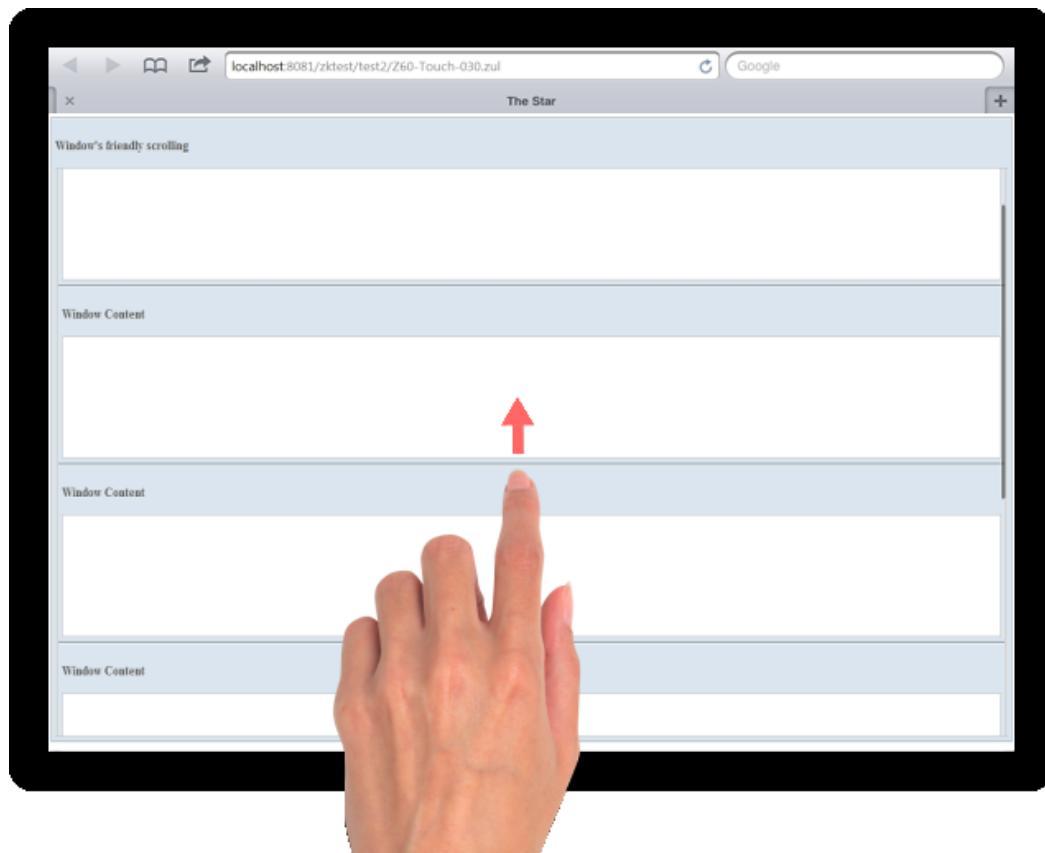
- Component Reference: Window
- Available in ZK EE only ^[13]

Friendly Scrolling Support

When user swipe on the content of Window, the friendly scrollbar will appear. To enable this, please specify the contentStyle with "overflow:auto"

For example,

```
<window contentStyle="overflow:auto">
```



Version History

Version	Date	Content
6.5.0	July, 2012	

Unsupported Molds

The following molds which are not supported in components for tablet devices.

- Button's *os* and *trendy* mold
- Bandbox's *rounded* mold
- Combobox's *rounded* mold
- Datebox's *rounded* mold
- Doublespinner's *rounded* mold
- Spinner's *rounded* mold
- Timebox's *rounded* mold
- Decimalbox's *rounded* mold
- Doublebox's *rounded* mold
- Intbox's *rounded* mold
- Longbox's *rounded* mold
- Textbox's *rounded* mold
- Groupbox's *default* mold
- Slider's *scale* mold
- Splitter's *os* mold
- Tabbox's *accordion-lite* mold

Note: When using the molds above, the component will use default supported mold instead.

Version History

Version	Date	Content
6.5.0	July, 2012	

Limitation

Unsupported Component

In Tablet/mobile device, there are some different design against desktop browser, which make some ZK components are not supported.

Frozen

Frozen component is implemented by cheating the browser to show scrollbar as desired. However, tablet/mobile devices have no scrollbar so this feature is not applicable in tablet/mobile devices.

Unsupported API

In Tablet device, we use some kind of the HTML5 new feature to generate the component output as tablet mold, so there are some limitation as follows.

NumberInputElement

[7.0.3]

Fall back to use Desktop implementation for NumberInputElement with "format" and "locale" attributes in Tablet mold.

[6.5.0]

In Tablet mold, all of the ZK components that extend from NumberInputElement cannot support *format* and *locale* attributes, due to the output of the component which the input type is number.

Version History

Version	Date	Content
6.5.0	July, 2012	
7.0.3	August, 2014	Fall back to use Desktop implementation for NumberInputElement with "format" and "locale" attributes in Tablet mold.

Accessibility

This chapter describes each component's detailed accessibility information including keyboard support and some important ARIA attributes.

For general accessibility information, please refer to ZK Developer's Reference/Accessibility.

- Available for ZK:
-   

Note: the following subsections are only applied for ZK EE version only.

Containers

This section outline Accessibilities in Containers components.

Note: the following subsections are only applied for ZK EE version only.

Drawer

- Available for ZK:
-   

Label a Component

To name a component with ARIA attribute, please refer to ZK Developer's Reference/Accessibility#Specify_ARIA_Attributes

Keyboard Support

Key	Description
Escape	Close the drawer

Limitations

Feature	Description
Autodrop	This feature only supports mouse.

Tabbox

- Available for ZK:
-   

Label a Component

To name a component with ARIA attribute, please refer to [ZK Developer's Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

Key	Description
Horizontal: ArrowLeft / ArrowRight	Navigate tabs.
Vertical: ArrowUp / ArrowDown	
Enter / Spacebar	Select the tab.
Delete	Close the tab. (only if closable is enabled)

Data

This section outline Accessibilities in Data components.

Note: the following subsections are only applied for ZK EE version only.

Biglistbox

- Available for ZK:

CE **PE** **EE**

Label a Component

To name a component with ARIA attribute, please refer to [ZK_Developer's_Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

Key	Description
ArrowUp / ArrowDown	Select items or focus on headers.
ArrowLeft / ArrowRight	Focus on cell or headers.

Grid

- Available for ZK:

CE **PE** **EE**

Label a Component

To name a component with ARIA attribute, please refer to [ZK_Developer's_Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

Key	Description
ArrowUp	Moves focus one cell Up. If focus is on the top cell in the column, focus does not move.
ArrowDown	Moves focus one cell down. If focus is on the bottom cell in the column, focus does not move.
ArrowLeft	Moves focus one cell to the left. If focus is on the left-most cell in the row, focus does not move.
ArrowRight	Moves focus one cell to the right. If focus is on the right-most cell in the row, focus does not move.
Enter / Spacebar	When the Detail/Group is focused, open/close the Detail/Group. When the sortable column is focused, press enter/spacebar to sort it.
Alt + ArrowDown	When the Column with menupopup is focused, open the menupopup.

Listbox

- Available for ZK:
- CE PE EE

Label a Component

To name a component with ARIA attribute, please refer to [ZK_Developer's_Reference/Accessibility#Specify_ARIA_Attributes](#)

You need to label a Listbox first, then ZK will add `aria-labelledby` on the `z-focus-a` button.

Keyboard Support

Key	Description
ArrowUp / ArrowDown	Select Listitems or focus on Listheaders.
ArrowLeft / ArrowRight	Focus on Listcells or Listheaders
Enter / Spacebar	If the checkmark is enabled, pressing Enter/Spacebar will toggle the checkbox.

Tree

- Available for ZK:
- CE PE EE

Label a Component

To name a component with ARIA attribute, please refer to [ZK_Developer's_Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

Key	Description
ArrowUp / ArrowDown	Select Treeitems or focus on Treecols.
ArrowLeft / ArrowRight	If the treeitem is openable, toggle the open status. Focus on Treecell or Treecols.

Essential Components

This section outline Accessibilities in the components which are considered a "must have" by ZK developers.

Note: the following subsections are only applied for ZK EE version only.

Include

- Available for ZK:
- CE PE EE

Label a Component

To name a component with ARIA attribute, please refer to [ZK Developer's Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

- None

Example

To make the included content more semantic, use `enclosingTag` to modify the enclosing tag name.

```
<include src="includes/footer.zul" enclosingTag="footer"/>
```

Menubar

- Available for ZK:
-   

Label a Component

To name a component with ARIA attribute, please refer to [ZK Developer's Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

Key	Description
Horizontal root level: ArrowLeft / ArrowRight	Navigate menu items
Vertical root level: ArrowUp / ArrowDown	
All Menupopup: ArrowUp / ArrowDown	
Horizontal root level: ArrowDown	Open the menu popup
Vertical root level : ArrowRight	
All Menupopup: ArrowLeft / ArrowRight	Open / close the menu popup
Enter	Activate the menu item / Open the menu popup
Spacebar	Activate the menu item / Open the menu popup Toggle the checkbox without closing the menupopup (only apply to autocheck enabled)
Escape	Close the menu popup

Limitations

Feature	Description
Autodrop	This feature only supports mouse.

Navbar

- Available for ZK:
- CE PE EE

Label a Component

To name a component with ARIA attribute, please refer to [ZK_Developer's_Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

Key	Description
ArrowLeft / ArrowRight	Horizontal: Navigate nav items Vertical: Close / open the nav menu popup
ArrowUp / ArrowDown	Horizontal: Close / open the nav menu popup Vertical: Navigate nav items
Enter / Spacebar	Select the nav item / Open the nav menu popup
Escape	Close the nav menu popup

Rating

- Available for ZK:
- CE PE EE

Label a Component

To name a component with ARIA attribute, please refer to [ZK_Developer's_Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

Key	Description
ArrowUp / ArrowRight	Increases the rating value.
ArrowDown / ArrowLeft	Decreases the rating value.
Home	Sets the rating value to 0.
End	Sets the rating to its maximum value.

Toolbar

- Available for ZK:
- CE PE EE

Label a Component

To name a component with ARIA attribute, please refer to [ZK Developer's Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

Key	Description
ArrowUp / ArrowLeft	Moves focus to the previous control. If there is no previous control, focus movement will wrap from the first element to the last element.
ArrowDown / ArrowRight	Moves focus to the next control. If there is no next control, focus movement will wrap from the last element to the first element.
Tab / Shift + Tab	Move focus into and out of the toolbar.

Limitations

Due to this issue ^[1], we suggest not to display any components that need to be controlled with arrow keys in the toolbar.

References

[1] <https://github.com/w3c/aria-practices/issues/1283>

Input

This section outline Accessibilities in Input components.

Note: the following subsections are only applied for ZK EE version only.

Bandbox

- Available for ZK:
- CE PE EE

Label a Component

To name a component with ARIA attribute, please refer to [ZK_Developer's_Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

Key	Description
Alt + ArrowUp	Close the popup.
Alt + ArrowDown	Open the popup.
Tab	If the input is focused and the popup is opened, move focus to the popup.

Calendar

- Available for ZK:
-   

Label a Component

To name a component with ARIA attribute, please refer to [ZK Developer's Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

Key	Description
ArrowUp / ArrowDown / ArrowLeft / ArrowRight	Navigate the date.
Enter / Spacebar	Select the date.

Calendar Day AriaLabel Renderer

This is achieved by overriding the default renderer at the client to customize the aria-label of days on ZK's Calendar.

```
<zk>
    <script><! [CDATA[
        zk.afterLoad('zul.db', function(){
            zul.db.Renderer.cellAriaLabel = function (cal, y, m,
day, monthofs, dayofweek) {
                var localizedSymbols =
cal.getLocalizedSymbols();
                return day + ' ' + localizedSymbols.FMON[m] +
', ' + y; // dd MMMM, yyyy
            };
        });
    ]]></script>
    <calendar/>
</zk>
```

[Since 9.5.0]

Cascader

- Available for ZK:
- CE PE EE

Label a Component

To name a component with ARIA attribute, please refer to [ZK_Developer's_Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

Key	Description
ArrowUp / ArrowLeft / ArrowRight	Navigate options.
ArrowDown	Open the popup or navigate options
Enter / Spacebar	Select the options
Escape	Close the popup
Backspace / Delete	Clear selection

Chosenbox

- Available for ZK:
- CE PE EE

Label a Component

To name a component with ARIA attribute, please refer to [ZK_Developer's_Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

Key	Description
ArrowUp / ArrowDown	Open the popup and navigate options.
ArrowLeft / ArrowRight	Navigate selections
Enter	Select the options
Escape	Close the popup
Delete / Backspace	Remove the focused selection.

Colorbox

- Available for ZK:
- CE** **PE** **EE**

Label a Component

To name a component with ARIA attribute, please refer to [ZK_Developer's_Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

Key	Description
ArrowUp / ArrowDown / ArrowLeft / ArrowRight	Navigate colors.
Enter	Select the color.
Escape	Close the popup

Datebox

- Available for ZK:
- CE PE EE

Label a Component

To name a component with ARIA attribute, please refer to [ZK_Developer's_Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

Key	Description
Alt + ArrowUp / Alt + ArrowDown	Open the date picker popup.
ArrowUp / ArrowDown / ArrowLeft / ArrowRight	(In popup) Navigate the date.
Enter / Spacebar	(In popup) Select the date.
Escape	(In popup) Close the popup.

Multislider

- Available for ZK:
- CE PE EE

Label a Component

To name a component with ARIA attribute, please refer to [ZK_Developer's_Reference/Accessibility#Specify_ARIA_Attributes](#)

Attributes	Description
data-ariaStartLabel	Describe the the slider button (start) on SliderButtons.
data-ariaEndLabel	Describe the the slider button (end) on SliderButtons.
data-largeStep-multiplier (optional)	Describe the moving step of pressing PageUp/PageDown.

Keyboard Support

Key	Description
ArrowUp / ArrowDown	Move the slider button.
ArrowLeft / ArrowRight	Move the slider button.
Home / End	Move the slider button to the minimum/maximum.
PageUp / PageDown	Move the slider button in the large step.

Example

```
<zk xmlns:ca="client/attribute">
  <multislider ca:aria-label="range value">
    <sliderbuttons startValue="0" endValue="100" ca:data-ariaStartLabel="minimal range value"
      ca:data-ariaEndLabel="maximal range value"/>
    <sliderbuttons startValue="10" endValue="50" ca:data-ariaStartLabel="minimal range value"
      ca:data-ariaEndLabel="maximal range value"/>
  </multislider>
</zk>
```

Rangeslider

- Available for ZK:
- CE PE EE

Label a Component

To name a component with ARIA attribute, please refer to [ZK_Developer's_Reference/Accessibility#Specify_ARIA_Attributes](#)

Attributes	Description
data-ariaStartLabel	Describe the the slider button (start).
data-ariaEndLabel	Describe the the slider button (end).
data-largeStep-multiplier (optional)	Describe the moving step of pressing PageUp/PageDown.

Keyboard Support

Key	Description
ArrowUp / ArrowDown	Move the slider button.
ArrowLeft / ArrowRight	Move the slider button.
Home / End	Move the slider button to the minimum/maximum.
PageUp / PageDown	Move the slider button in the large step.

Example

```
<zk xmlns:ca="client/attribute">
  <rangeslider ca:aria-label="range value" ca:data-ariaStartLabel="minimal range value"
    ca:data-ariaEndLabel="maximal range value"/>
</zk>
```

Searchbox

- Available for ZK:
- CE PE EE

Label a Component

To name a component with ARIA attribute, please refer to [ZK Developer's Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

Key	Description
ArrowDown	Open the popup and navigate options.
ArrowUp	Navigate options.
Enter	Select the options
Escape	Close the popup

Slider

- Available for ZK:

CE PE EE

Label a Component

To name a component with ARIA attribute, please refer to [ZK_Developer's_Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

Key	Description
ArrowUp / ArrowRight	Increases the value of the slider by one step.
ArrowDown / ArrowLeft	Decreases the value of the slider by one step.
Home	Sets slider to its minimum value.
End	Sets slider to its maximum value.

Timebox

- Available for ZK:

CE PE EE

Label a Component

To name a component with ARIA attribute, please refer to [ZK_Developer's_Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

Key	Description
ArrowUp / ArrowDown	Increase / decrease the value.
ArrowLeft / ArrowRight	Move the input cursor.

Timepicker

- Available for ZK:
-   

Label a Component

To name a component with ARIA attribute, please refer to [ZK Developer's Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

Key	Description
Alt+Shift+ArrowDown	Open the popup
ArrowUp / ArrowDown	Navigate options
Enter	Select the option
Escape	Close the popup

Layouts

This section outline Accessibilities in Layout components.

Note: the following subsections are only applied for ZK EE version only.

Borderlayout

- Available for ZK:

CE **PE** **EE**

Label a Component

To name a component with ARIA attribute, please refer to [ZK_Developer's_Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

Key	Description
ArrowUp / ArrowDown	When the splitter is focused, move the splitter vertically.
ArrowLeft / ArrowRight	When the splitter is focused, move the splitter horizontally.
Enter / Spacebar	When the splitter is focused, collapses the region. When the header is focused, collapsed/expanded the region.

Cardlayout

- Available for ZK:

CE **PE** **EE**

Label a Component

To name a component with ARIA attribute, please refer to [ZK_Developer's_Reference/Accessibility#Specify_ARIA_Attributes](#)

Cardlayout is often used along with other components to achieve the carousel effect. If you are using cardlayout for carousel, check here for carousel practice ^[1].

Sample

Here is a simple traffic light carousel:

```
<div ca:role="region" ca:aria-roledescription="carousel" ca:aria-label="traffic light" tabindex="0">
  <div>
    <button ca:aria-label="Previous Slide" onClick="card.previous()">previous</button>
    <button ca:aria-label="Next Slide" onClick="card.next()">next</button>
  </div>
  <cardlayout id="card" width="300px" height="200px" style="border:1px solid red" selectedIndex="1" tabindex="0">
    <div ca:aria-label="red" vflex="1" hflex="1" style="background-color:red;padding:20px">red</div>
    <div ca:aria-label="yellow" vflex="1" hflex="1" style="background-color:yellow;padding:20px">yellow</div>
    <div ca:aria-label="green" vflex="1" hflex="1" style="background-color:green;padding:20px">green</div>
  </cardlayout>
```

```
</div>
```

References

[1] <https://www.w3.org/TR/wai-aria-practices/#carousel>

Linelayout

- Available for ZK:
-   

Label a Component

To name a component with ARIA attribute, please refer to
ZK_Developer's_Reference/Accessibility#Specify_ARIA_Attributes

Keyboard Support

Key	Description
ArrowUp / ArrowLeft	Moves focus to previous item.
ArrowDown / ArrowRight	Moves focus to next item.

Organigram

- Available for ZK:
- CE PE EE

Label a Component

To name a component with ARIA attribute, please refer to [ZK_Developer's_Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

Key	Description
ArrowUp	Moves focus to upper level item. / If an opened item is focused, close it.
ArrowDown	Moves focus to next level item. / If a closed item is focused, open it.
ArrowLeft	Moves focus to the previous sibling.
ArrowRight	Moves focus to the next sibling.
Enter / Spacebar	When the focus item is not selected, select it. When the focus item is selected, toggle its open/close.

Portallayout

- Available for ZK:
- CE PE EE

Label a Component

To name a component with ARIA attribute, please refer to [ZK_Developer's_Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

To move the Panel from a Portalchildren to another with keyboard:

Key	Description
Spacebar	When the target Panel is focused, press spacebar will open a popup list with optional Portalchildren.
ArrowUp / ArrowDown	When the popup is opened, move the current selection in the popup list.
Enter	When the popup is opened, select the target Portalchildren. The target Panel will move to the target Portalchildren then close the popup list.
Esc	When the popup is opened, close the popup list.

Splitter

- Available for ZK:
- CE PE EE**

Label a Component

To name a component with ARIA attribute, please refer to [ZK_Developer's_Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

Key	Description
ArrowUp / ArrowDown	When the splitter is focused, move the splitter vertically.
ArrowLeft / ArrowRight	When the splitter is focused, move the splitter horizontally.
Enter	If the primary pane is not collapsed, collapse the pane. If the pane is collapsed, restore the splitter to its previous position.

Splitlayout

- Available for ZK:

CE PE EE

Label a Component

To name a component with ARIA attribute, please refer to [ZK Developer's Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

Key	Description
ArrowUp / ArrowDown	When the splitter is focused, move the splitter vertically.
ArrowLeft / ArrowRight	When the splitter is focused, move the splitter horizontally.
Enter	If the primary pane is not collapsed, collapses the pane. If the pane is collapsed, restores the splitter to its previous position.

Multimedia and Miscellaneous

This section outline Accessibilities in Multimedia and Miscellaneous components.

Note: the following subsections are only applied for ZK EE version only.

Barcode

- Available for ZK:
- CE PE EE

Label a Component

To name a component with ARIA attribute, please refer to [ZK_Developer's_Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

None.

BarcodeScanner

- Available for ZK:
- CE PE EE

Label a Component

To name a component with ARIA attribute, please refer to [ZK_Developer's_Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

None.

Pdfviewer

- Available for ZK:

CE PE EE

Label a Component

To name a component with ARIA attribute, please refer to [ZK Developer's Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

Key	Description
ArrowUp / ArrowDown PageUp / PageDown	Scroll the vertical scrollbar if any. Navigate to the previous/next page if the scrollbar is on the topmost/bottommost.
ArrowLeft / ArrowRight	Scroll the horizontal scrollbar if any. Navigate to the previous/next page if the scrollbar is on the leftmost/rightmost.
Home / End	Scroll the vertical scrollbar if any. Navigate to the first/last page if the scrollbar is on the topmost/bottommost.

Supplementary

This section outline Accessibilities in Supplementary components.

Note: the following subsections are only applied for ZK EE version only.

Stepbar

- Available for ZK:
-   

Label a Component

To name a component with ARIA attribute, please refer to [ZK_Developer's_Reference/Accessibility#Specify_ARIA_Attributes](#)

Keyboard Support

(Only available if `linear` is `false`)

Key	Description
ArrowLeft / ArrowUp	Focus on the previous step.
ArrowRight/ ArrowDown	Focus on the next step.
Enter / Spacebar	Select the step.

Unsupported Components

The following components are not WCAG-compliant, mostly due to the underlying 3rd party library; or the nature of the technologies being used. Avoid using these components - when the following components are being used, assistive technology users will be hard to interact with them, making it impossible for the users to navigate the application smoothly.

- Captcha (Please use reCAPTCHA instead)
- Chart (Suggest using ZKCharts [1])
- Cropper (Depends on Jcrop, a third-party library. Drag and drop)
- Dropupload (Drag and drop)
- GoldenLayout (Depends on GoldenLayout, a third-party library)
- Jasperreport (Depends on JasperReports, a third-party library)
- Signature (Depends on Signature Pad, a third-party library)
- Tbeditor (Depends on Trumbowyg, a third-party library)

Version History

Version	Date	Content
9.5.0	August 2020	

References

- [1] <https://www.zkoss.org/product/zkcharts>

Article Sources and Contributors

ZK Component Reference *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference *Contributors:* Alicelin, Hawk, Jeanher, Southerncrossie, Sphota, Tmillsclare, Tomyeh, Wingor

Introduction *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Introduction *Contributors:* Tmillsclare, Tomyeh

Example Source Code *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Introduction/Example_Source_Code *Contributors:* Hawk

Base Components *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Base_Components *Contributors:* Tmillsclare

AbstractComponent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Base_Components/AbstractComponent *Contributors:* Ashishd, Hawk, Tmillsclare, Tomyeh, Zkwikiadmin

FooterElement *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Base_Components/FooterElement *Contributors:* Hawk, Jimmyshiau

FormatInputElement *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Base_Components/FormatInputElement *Contributors:* Ashishd, Hawk, Jumperchen, Tmillsclare, Zkwikiadmin

HeaderElement *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Base_Components/HeaderElement *Contributors:* Ashishd, Hawk, Jimmyshiau, Tmillsclare, Zkwikiadmin

HeadersElement *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Base_Components/HeadersElement *Contributors:* Hawk, Iantsai, Jimmyshiau, Robertwenzel, Tmillsclare, Zkwikiadmin

HtmlBasedComponent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Base_Components/HtmlBasedComponent *Contributors:* Hawk, Iantsai, Jumperchen, Tmillsclare, Tomyeh, Zkwikiadmin

HtmlMacroComponent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Base_Components/HtmlMacroComponent *Contributors:* Alicelin, Ashishd, Hawk, Jumperchen, Tmillsclare, Tomyeh

InputElement *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Base_Components/InputElement *Contributors:* Alicelin, Chunfuchang, Flyworld, Hawk, Henrichen, Iantsai, Jameschu, Jumperchen, Leonlee, Matthewcheng, MontyPan, Neilee, Rudyhuang, SimonPai, Tmillsclare, Tomyeh, Vincent, Zkwikiadmin

LabelElement *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Base_Components/LabelElement *Contributors:* Hawk, Jimmyshiau, Jumperchen, Tmillsclare, Zkwikiadmin

LabelImageElement *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Base_Components/LabelImageElement *Contributors:* Alicelin, Hawk, Jeanher, Jimmyshiau, Jumperchen, Matthieu, Rudyhuang, Tmillsclare, Tomyeh, Vincent, Zkwikiadmin

LayoutRegion *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Base_Components/LayoutRegion *Contributors:* Hawk, Jimmyshiau, Jumperchen, Rudyhuang, Tmillsclare, Zkwikiadmin

NumberInputElement *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Base_Components/NumberInputElement *Contributors:* Benbai, Hawk, Jumperchen, Tmillsclare, Zkwikiadmin

XulElement *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Base_Components/XulElement *Contributors:* Hawk, Jumperchen, Tmillsclare, Tomyeh, Zkwikiadmin

Containers *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Containers *Contributors:* Tmillsclare, Tomyeh

Caption *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Containers/Caption *Contributors:* Benbai, Ejikipaticy, Hawk, Jimmyshiau, Jumperchen, SimonPai, Sphota, Tmillsclare, Tomyeh

Div *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Containers/Div *Contributors:* Hawk, SimonPai, Sphota, Tmillsclare, Tomyeh

Drawer *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Containers/Drawer *Contributors:* Hawk, Jeanher, Rudyhuang

Fragment *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Containers/Fragement *Contributors:* Elenalin, Hawk, Michellechen, Rudyhuang

Groupbox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Containers/Groupbox *Contributors:* Benbai, Hawk, Jimmyshiau, Jumperchen, Peterkuo, Sphota, Tmillsclare, Tomyeh

Idspace *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Containers/Idspace *Contributors:* Benbai, Hawk, Jameschu

Inputgroup *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Containers/Inputgroup *Contributors:* Hawk, Rudyhuang

Nodom *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Containers/Nodom *Contributors:* Hawk, Jameschu

Panel *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Containers/Panel *Contributors:* Alicelin, Hawk, Jumperchen, Peterkuo, Rudyhuang, SimonPai, Tmillsclare, Tomyeh

Panelchildren *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Containers/Panel/Panelchildren *Contributors:* Hawk, Jumperchen, Peterkuo, Tmillsclare, Zkwikiadmin

Span *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Containers/Span *Contributors:* Hawk, Samchuang, SimonPai, Tmillsclare, Tomyeh

Tabbox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Containers/Tabbox *Contributors:* Hawk, Jumperchen, Leonlee, Noahhuang, Peterkuo, Samchuang, Southerncrossie, Tmillsclare, Tomyeh

Tab *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Containers/Tabbox/Tab *Contributors:* Alicelin, Hawk, Jumperchen, SimonPai, Tmillsclare

Tabs *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Containers/Tabbox/Tabs *Contributors:* Hawk, Jumperchen, SimonPai, Tmillsclare

Tabpanel *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Containers/Tabbox/Tabpanel *Contributors:* Flyworld, Hawk, Jumperchen, SimonPai, Tmillsclare

Tabpanels *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Containers/Tabbox/Tabpanels *Contributors:* Hawk, Jumperchen, SimonPai, Tmillsclare

Window *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Containers/Window *Contributors:* Alicelin, Hawk, Jumperchen, Peterkuo, Samchuang, SimonPai, Tmillsclare, Tomyeh, Tonyq

Data *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Data *Contributors:* Tmillsclare, Tomyeh, Zkwikiadmin

Grid *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Grid *Contributors:* Alicelin, Char, Christopherszu, Hawk, Iantsai, Jameschu, Jerrychen, Jimmyshiau, Jumperchen, Kbsimm, Noahhuang, Peterkuo, Robertwenzel, Rudyhuang, Samchuang, SimonPai, Tmillsclare, Tomyeh, Tonyq, Vincent, Wenninghsu

Column *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Grid/Column *Contributors:* Hawk, Jimmyshiau, Jumperchen, SimonPai, Tmillsclare, Tomyeh, Zkwikiadmin

Columns *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Grid/Columns *Contributors:* Hawk, Jimmyshiau, Jumperchen, SimonPai, Tmillsclare, Tomyeh, Zkwikiadmin

Detail *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Grid/Detail *Contributors:* Hawk, Jimmyshiau, Jumperchen, Peterkuo, SimonPai, Tmillsclare, Tomyeh, Zkwikiadmin

Foot Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Grid/Foot_Contributors: Flyworld, Hawk, Jimmyshiau, Jumperchen, SimonPai, Tmillsclare, Tomyeh, Zwikiadmin

Footer Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Grid/Footer_Contributors: Hawk, Jimmyshiau, Jumperchen, Peterkuo, SimonPai, Tmillsclare, Tomyeh, Zwikiadmin

Group Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Grid/Group_Contributors: Hawk, Jimmyshiau, Jumperchen, SimonPai, Tmillsclare, Tomyeh, Zwikiadmin

Groupfoot Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Grid/Groupfoot_Contributors: Hawk, Jimmyshiau, Jumperchen, SimonPai, Sphota, Tmillsclare, Tomyeh, Zwikiadmin

Row Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Grid/Row_Contributors: Hawk, Jimmyshiau, Peterkuo, SimonPai, Tmillsclare, Tomyeh

Rows Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Grid/Rows_Contributors: Hawk, Jimmyshiau, Peterkuo, SimonPai, Tmillsclare, Tomyeh

Listbox Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Listbox_Contributors: Alicelin, Benbai, Charlesqiu, Hawk, Jameschu, Jerrychen, Jimmyshiau, Jumperchen, Matthieu, MontyPan, Neilllee2, Noahuang, Peterkuo, Pipe, Robertwenzel, Rudyhuang, Samchuang, SimonPai, Sphota, Tmillsclare, Tomyeh, Tonyq, Vincent, Wenninghsu

Listcell Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Listbox/Listcell_Contributors: Hawk, Jimmyshiau, Peterkuo, SimonPai, Tmillsclare, Tomyeh, Zwikiadmin

Listfoot Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Listbox/Listfoot_Contributors: Hawk, Jimmyshiau, SimonPai, Sphota, Tmillsclare, Tomyeh

Listfooter Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Listbox/Listfooter_Contributors: Hawk, Jimmyshiau, Peterkuo, SimonPai, Tmillsclare, Tomyeh, Zwikiadmin

Listgroup Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Listbox/Listgroup_Contributors: Hawk, Jimmyshiau, Jumperchen, Peterkuo, SimonPai, Tmillsclare, Tomyeh, Zwikiadmin

Listgroupfoot Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Listbox/Listgroupfoot_Contributors: Hawk, Jimmyshiau, Jumperchen, Samchuang, SimonPai, Tmillsclare, Tomyeh, Zwikiadmin

Listhead Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Listbox/Listhead_Contributors: Hawk, Jameschu, Jimmyshiau, Jumperchen, Samchuang, SimonPai, Tmillsclare, Tomyeh, Zwikiadmin

Listheader Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Listbox/Listheader_Contributors: Hawk, Jimmyshiau, Jumperchen, Samchuang, SimonPai, Tmillsclare, Tomyeh, Zwikiadmin

Listitem Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Listbox/ListItem_Contributors: Hawk, Jimmyshiau, Peterkuo, SimonPai, Tmillsclare, Tomyeh, Zwikiadmin

Tree Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Tree_Contributors: Alicelin, Char, Charlesqiu, Hawk, Jameschu, Jeanher, Jerrychen, Jimmyshiau, Jumperchen, Neilllee2, Noahuang, Peterkuo, Samchuang, SimonPai, Tmillsclare, Tomyeh, Vincent, Wenninghsu

Treecell Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Tree/Treecell_Contributors: Alicelin, Ashishd, Hawk, Jimmyshiau, SimonPai, Tmillsclare, Tomyeh, Zwikiadmin

Treechildren Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Tree/Treechildren_Contributors: Hawk, Iantsai, Jimmyshiau, Peterkuo, SimonPai, Tmillsclare, Tomyeh, Zwikiadmin

Treecol Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Tree/Treecol_Contributors: Hawk, Iantsai, Jimmyshiau, SimonPai, Tmillsclare, Tomyeh, Zwikiadmin

Treecols Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Tree/Treecols_Contributors: Alicelin, Hawk, Iantsai, Jimmyshiau, Tmillsclare, Tomyeh, Zwikiadmin

Treefoot Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Tree/Treefoot_Contributors: Hawk, Jimmyshiau, SimonPai, Tmillsclare, Tomyeh, Zwikiadmin

Treefooter Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Tree/Treefooter_Contributors: Hawk, Jimmyshiau, SimonPai, Tmillsclare, Tomyeh, Zwikiadmin

Treeitem Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Tree/Treeitem_Contributors: Hawk, Iantsai, Jimmyshiau, SimonPai, Tmillsclare, Tomyeh

Treerow Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Tree/Treerow_Contributors: Alicelin, Hawk, Jimmyshiau, SimonPai, Tmillsclare, Tomyeh, Zwikiadmin

Biglistbox Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Data/Biglistbox_Contributors: Hawk, Jumperchen, Samchuang

Diagrams and Reports Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Diagrams_and_Reports_Contributors: Tmillsclare

Chart Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Diagrams_and_Reports/Chart_Contributors: Alicelin, Hawk, Jimmyshiau, Jumperchen, Newacct, SimonPai, Tmillsclare, Tomyeh

Flashchart Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Diagrams_and_Reports/Flashchart_Contributors: Hawk, Jumperchen, Samchuang, SimonPai, Tmillsclare

Fusionchart Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Diagrams_and_Reports/Fusionchart_Contributors: Alicelin, Hawk, Jeanher, Jimmyshiau, Jumperchen, Murasakiwu, Robertwenzel, Samchuang, Tonyq, Vincent

Gmaps Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Diagrams_and_Reports/Gmaps_Contributors: Alicelin, Benbai, Ejikipaticy, Hawk, Henrichen, Jeanher, Jimmyshiau, Matthieu, Robertwenzel, SimonPai, Sphota, Tmillsclare

Gimage Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Diagrams_and_Reports/Gmaps/Gimage_Contributors: Hawk, Henrichen, Jimmyshiau, SimonPai, Tmillsclare

Ginfo Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Diagrams_and_Reports/Gmaps/Ginfo_Contributors: Hawk, Henrichen, Jimmyshiau, SimonPai, Tmillsclare

Gmarker Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Diagrams_and_Reports/Gmaps/Gmarker_Contributors: Hawk, Henrichen, Jimmyshiau, SimonPai, Tmillsclare

Gpolyline Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Diagrams_and_Reports/Gmaps/Gpolyline_Contributors: Hawk, Henrichen, Jimmyshiau, Robertwenzel, SimonPai, Tmillsclare

Gpolygon Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Diagrams_and_Reports/Gmaps/Gpolygon_Contributors: Hawk, Henrichen, Jimmyshiau, Robertwenzel, SimonPai, Tmillsclare

Gscreen Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Diagrams_and_Reports/Gmaps/Gscreen_Contributors: Hawk, Henrichen, Jimmyshiau, SimonPai, Tmillsclare

Gcircle Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Diagrams_and_Reports/Gmaps/Gcircle_Contributors: Hawk, Matthieu

Jasperreport Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Diagrams_and_Reports/Jasperreport_Contributors: Charlesqiu, Hawk, Jimmyshiau, Jumperchen, Robertwenzel, Samchuang, SimonPai, Tmillsclare, Tomyeh, Tonyq

Timeline Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Diagrams_and_Reports/Timeline_Contributors: Char, Hawk, Jameschu, Jimmyshiau, Jumperchen, Tmillsclare

Bandinfo Source: http://10.1.3.180/index.php?title=ZK_Component_Reference/Diagrams_and_Reports/Timeline/Bandinfo_Contributors: Hawk, Jameschu, Jimmyshiau, Jumperchen, Tmillsclare

Hotzone *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Diagrams_and_Reports/Timeline/Hotzone *Contributors:* Hawk, Jameschu, Jimmyshiau, Jumperchen, Tmillsclare

Timeplot *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Diagrams_and_Reports/Timeplot *Contributors:* Char, Hawk, Jimmyshiau, Jumperchen, Tmillsclare

Plotinfo *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Diagrams_and_Reports/Timeplot/Plotinfo *Contributors:* Hawk, Jimmyshiau, Jumperchen, Tmillsclare

Essential Components *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components *Contributors:* Tmillsclare, Zkwikiadmin

A *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/A *Contributors:* Alicelin, Hawk, Jumperchen, Matthieu, Samchuang, SimonPai, Tmillsclare, Tomyeh

Anchornav *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Anchornav *Contributors:* Hawk, Jameschu, Jeanher, Matthieu, Rudyhuang

Applet *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Applet *Contributors:* Alicelin, Char, Hawk, Jumperchen, Samchuang, Tmillsclare, Tomyeh

Button *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Button *Contributors:* Alicelin, Ashishd, Char, Hawk, Jameschu, Jimmyshiau, Jumperchen, Peterkuo, SimonPai, Tmillsclare, Tomyeh, Tonyq

Captcha *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Captcha *Contributors:* Hawk, Iantsai, Jumperchen, Tmillsclare, Tomyeh

Combobutton *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Combobutton *Contributors:* Benbai, Hawk, Jumperchen, Vincent

Dropupload *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Dropupload *Contributors:* Charlesqiu, Christopherszu, Hawk, Jumperchen, MontyPan, Noahhuang, Southerncrosse

Filedownload *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Filedownload *Contributors:* Alicelin, Hawk, Matthieu, Robertwenzel, Tomyeh

Fileupload *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Fileupload *Contributors:* Alicelin, Flyworld, Hawk, Jumperchen, Matthieu, Peterkuo, Samchuang, Tmillsclare, Tomyeh, Zkwikiadmin

Fisheye *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Fisheye *Contributors:* Alicelin, Ashishd, Hawk, Jumperchen, Peterkuo, Tmillsclare

Fisheyebar *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Fisheyebar *Contributors:* Hawk, Jumperchen, Sphota, Tmillsclare

Html *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Html *Contributors:* Alicelin, Char, Hawk, Jumperchen, MontyPan, SimonPai, Tmillsclare, Tomyeh, Vincent

Iframe *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Iframe *Contributors:* Alicelin, Hawk, Jumperchen, Matthewcheng, SimonPai, Tmillsclare, Tomyeh

Include *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Include *Contributors:* Alicelin, Ashishd, Hawk, Jameschu, Jimmyshiau, Jumperchen, Matthieu, SimonPai, Tmillsclare, Tomyeh, Tonyq

Image *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Image *Contributors:* Alicelin, Ashishd, Hawk, Jumperchen, Murasakiwu, Peterkuo, SimonPai, Tmillsclare, Tomyeh, Vincent

Imagemap *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Imagemap *Contributors:* Hawk, Jumperchen, Peterkuo, SimonPai, Tmillsclare, Tonyq

Area *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Imagemap/Area *Contributors:* Hawk, Jumperchen, Murasakiwu, Samchuang, Tmillsclare

Label *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Label *Contributors:* Ashishd, Hawk, Jimmyshiau, Jumperchen, Matthewcheng, Tmillsclare, Tomyeh

Menu *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Menu *Contributors:* Alicelin, Hawk, Iantsai, Jumperchen, Peterkuo, Tmillsclare, Tomyeh

Menubar *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Menu/Menubar *Contributors:* Alicelin, Hawk, Iantsai, Jumperchen, SimonPai, Tmillsclare, Tomyeh, Zkwikiadmin

MenuItem *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Menu/MenuItem *Contributors:* Alicelin, Char, Hawk, Iantsai, Jimmyshiau, Jumperchen, Robertwenzel, Tmillsclare, Tomyeh, Zkwikiadmin

Menupopup *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Menu/Menupopup *Contributors:* Alicelin, Danchen, Hawk, Jimmyshiau, Jumperchen, Rudyhuang, Tmillsclare, Tomyeh

Menuseparator *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Menu/Menuseparator *Contributors:* Hawk, Jimmyshiau, Jumperchen, Tmillsclare, Tomyeh, Zkwikiadmin

Nav *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Nav *Contributors:* Hawk, Raymondchao, Vincent

Navbar *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Nav/Navbar *Contributors:* Hawk, Jeanher, Raymondchao, Robertwenzel, Vincent

NavItem *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Nav/NavItem *Contributors:* Hawk, Leonlee, Raymondchao, Vincent

Navseparator *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Nav/Navseparator *Contributors:* Hawk, Raymondchao, Vincent

Popup *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Popups *Contributors:* Alicelin, Char, Hawk, Jerrychen, Jimmyshiau, Jumperchen, Raymondchao, Robertwenzel, SimonPai, Tmillsclare, Tomyeh

Progressmeter *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Progressmeter *Contributors:* Hawk, Jumperchen, Leonlee, Peterkuo, Tmillsclare

Rating *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Rating *Contributors:* Hawk, Wenninghsu

Selectbox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Selectbox *Contributors:* Benbai, Hawk, Jeanher, Jumperchen

Separator *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Separator *Contributors:* Hawk, Henrichen, Jumperchen, Tmillsclare

Space *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Space *Contributors:* Hawk, Jumperchen, Tmillsclare

Script *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Script *Contributors:* Char, Hawk, Jimmyshiau, Jumperchen, Robertwenzel, Tmillsclare, Tomyeh

Style *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Style *Contributors:* Hawk, Jumperchen, Sphota, Tmillsclare, Tomyeh

Timer *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Timer *Contributors:* Alicelin, Ashishd, Hawk, Jumperchen, Tmillsclare

Toolbar *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Toolbar *Contributors:* Alicelin, Charlesqiu, Hawk, Jumperchen, KatherineLin, Peterkuo, Sphota, Tmillsclare, Tomyeh

Toolbarbutton *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Essential_Components/Toolbarbutton *Contributors:* Alicelin, Char, Hawk, Jumperchen, Sphota, Tmillsclare, Tomyeh, Tonyq

Input *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input *Contributors:* Tmillsclare

Bandbox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Bandbox *Contributors:* Alicelin, Hawk, Jimmyshiau, Jumperchen, Leonlee, Matthieu, Peterkuo, Robertwenzel, Tmillsclare, Tomyeh

Bandpopup *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Bandpopup *Contributors:* Alicelin, Hawk, Jimmyshiau, Jumperchen, Sphota, Tmillsclare, Tomyeh

Calendar *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Calendar *Contributors:* Alicelin, Char, Hawk, Jimmyshiau, Jumperchen, Leonlee, Sphota, Tmillsclare, Tomyeh

Cascader *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Cascader *Contributors:* Hawk, Jameschu, Jeanher, Rudyhuang

Checkbox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Checkbox *Contributors:* Benbai, Charlesqiu, Hawk, Jeanher, Jumperchen, Leonlee, Sphota, Tmillsclare

Chosenbox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Chosenbox *Contributors:* Benbai, Hawk, Jameschu, Noahhuang, Southerncrossie, Vincent

CKEditor *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/CKEditor *Contributors:* Benbai, Hawk, Jeanher, Jimmyshiau, Matthieu, Noahhuang, SimonPai, Tmillsclare

Colorbox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Colorbox *Contributors:* Benbai, Hawk, Jumperchen, Samchuang, Tmillsclare

Combobox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Combobox *Contributors:* Alicelin, Hawk, Henrichen, Jimmyshiau, Jumperchen, Leonlee, Peterkuo, Robertwenzel, Rudyhuang, Tmillsclare, Tomyeh, Tonyq, Vincent, Wenninghsu, Zkwikiadmin

Comboitem *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Combobox/Comboitem *Contributors:* Hawk, Jumperchen, Peterkuo, Tmillsclare, Tomyeh, Vincent

Datebox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Datebox *Contributors:* Alicelin, Char, Chunfuchang, Hawk, Jimmyshiau, Jumperchen, Leonlee, Peterkuo, Robertwenzel, Rudyhuang, Tmillsclare, Tomyeh, Wenninghsu

Decimalbox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Decimalbox *Contributors:* Alicelin, Charlesqiu, Dennischen, Hawk, Jimmyshiau, Jumperchen, Samchuang, Tmillsclare

Doublebox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Doublebox *Contributors:* Alicelin, Charlesqiu, Hawk, Jimmyshiau, Jumperchen, Samchuang, Tmillsclare

Doublespinner *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Doublespinner *Contributors:* Charlesqiu, Hawk, Jumperchen, Tmillsclare

Intbox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Intbox *Contributors:* Alicelin, Charlesqiu, Hawk, Jimmyshiau, Jumperchen, SimonPai, Tmillsclare

Longbox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Longbox *Contributors:* Charlesqiu, Hawk, Jimmyshiau, Jumperchen, SimonPai, Tmillsclare

Multislider *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Multislider *Contributors:* Hawk, Jameschu, Jeanher

Radio *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Radio *Contributors:* Hawk, Jumperchen, SimonPai, Tmillsclare

Radiogroup *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Radiogroup *Contributors:* Alicelin, Ashishd, Benbai, Hawk, Jimmyshiau, Jumperchen, Tmillsclare, Tomyeh, Tonyq

Rangeslider *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Rangeslider *Contributors:* Hawk, Jameschu, Jeanher

Searchbox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Searchbox *Contributors:* Hawk, Jeanher, Rudyhuang

Signature *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Signature *Contributors:* Charlesqiu, Hawk, KatherineLin, Klyvechen, Michellechen, Robertwenzel

Slider *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Slider *Contributors:* Alicelin, Ashishd, Charlesqiu, Hawk, Jimmyshiau, Jumperchen, Klyvechen, Peterkuo, Raymondchao, Tmillsclare, Tomyeh, Wenninghsu

Sliderbuttons *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Sliderbuttons *Contributors:* Hawk, Jameschu, Jeanher

Spinner *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Spinner *Contributors:* Ashishd, Charlesqiu, Flyworld, Hawk, Jimmyshiau, Jumperchen, Peterkuo, Tmillsclare

Tbeditor *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Tbeditor *Contributors:* Chunfuchang, Hawk

Textbox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Textbox *Contributors:* Alicelin, Charlesqiu, Hawk, Iantsai, Jimmyshiau, Jumperchen, Neillee2, Tmillsclare, Tomyeh, Vincent

Timebox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Timebox *Contributors:* Alicelin, Hawk, Iantsai, Jerrychen, Jimmyshiau, Jumperchen, Peterkuo, Tmillsclare, Tomyeh

Timepicker *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Input/Timepicker *Contributors:* Hawk, Jameschu

Layouts *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts *Contributors:* Tmillsclare, Tomyeh, Zkwikiadmin

Absolutelayout *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Absolutelayout *Contributors:* Hawk, Jumperchen

Absolutechildren *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Absolutelayout/Absolutechildren *Contributors:* Hawk, Jumperchen

Anchorlayout *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Anchorlayout *Contributors:* Hawk, Jumperchen

Anchorchildren *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Anchorlayout/Anchorchildren *Contributors:* Hawk, Jumperchen

Borderlayout *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Borderlayout *Contributors:* Charlesqiu, Hawk, Iantsai, Jimmyshiau, Jumperchen, Tmillsclare, Tomyeh, Vincent, Zkwikiadmin

Center *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Borderlayout/Center *Contributors:* Hawk, Jimmyshiau, Jumperchen, Tmillsclare, Tomyeh, Zkwikiadmin

East *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Borderlayout/East *Contributors:* Hawk, Jimmyshiau, Jumperchen, Rudyhuang, Tmillsclare, Tomyeh, Zkwikiadmin

North *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Borderlayout/North *Contributors:* Hawk, Jimmyshiau, Jumperchen, Rudyhuang, Tmillsclare, Tomyeh, Zkwikiadmin

South *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Borderlayout/South *Contributors:* Hawk, Jimmyshiau, Jumperchen, Rudyhuang, Tmillsclare, Tomyeh, Zkwikiadmin

West *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Borderlayout/West *Contributors:* Hawk, Jimmyshiau, Jumperchen, Rudyhuang, Tmillsclare, Tomyeh, Zkwikiadmin

Box *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Box *Contributors:* Alicelin, Hawk, Jumperchen, Peterkuo, Sphota, Tmillsclare, Tomyeh, Vincent

Cardlayout *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Cardlayout *Contributors:* Hawk, MontyPan, Southerncrossie, Vincent

Columnlayout *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Columnlayout *Contributors:* Benbai, Hawk, Jimmyshiau, Jumperchen, Southerncrossie, Tmillsclare, Tomyeh, Zkwikiadmin

Columnchildren *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Columnlayout/Columnchildren *Contributors:* Ashishd, Benbai, Hawk, Jumperchen, Tmillsclare, Tomyeh

GoldenLayout *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/GoldenLayout *Contributors:* Hawk, Jeanher, Wenninghsu

GoldenPanel *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/GoldenLayout/GoldenPanel *Contributors:* Hawk, Wenninghsu

Hbox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Hbox *Contributors:* Alicelin, Hawk, Jeanher, Jimmyshiau, Jumperchen, Peterkuo, Robertwenzel, Tmillsclare, Tomyeh, Vincent

HBoxLayout *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Hlayout *Contributors:* Benbai, Hawk, Jimmyshiau, Jumperchen, Southerncrossie, Tomyeh

Linelayout *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Linelayout *Contributors:* Hawk, Jeanher, Leonlee

Lineitem *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Linelayout/Lineitem *Contributors:* Hawk, Jeanher, Leonlee

Organigram *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Organigram *Contributors:* Charlesqiu, Hawk, Michellechen

Orgchildren *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Organigram/Orgchildren *Contributors:* Charlesqiu, Hawk, Michellechen

Orgitem *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Organigram/Orgitem *Contributors:* Charlesqiu, Hawk, Michellechen

Orgnode *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Organigram/Orgnode *Contributors:* Charlesqiu, Hawk, Michellechen

Portallayout *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Portallayout *Contributors:* Alicelin, Hawk, Jimmyshiau, Jumperchen, Leonlee, Neillee, SimonPai, Sphota, Tmillsclare, Tomyeh, Zkwikiadmin

Portalchildren *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Portallayout/Portalchildren *Contributors:* Hawk, Jeanher, Jimmyshiau, Jumperchen, Leonlee, SimonPai, Sphota, Tmillsclare, Tomyeh

Rowlayout *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Rowlayout *Contributors:* Hawk, Jameschu

Rowchildren *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Rowlayout/Rowchildren *Contributors:* Hawk, Jameschu

Splitlayout *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Splitlayout *Contributors:* Hawk, Jameschu

Splitter *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Splitter *Contributors:* Darius, Hawk, Jimmyshiau, Peterkuo, Robertwenzel, Samchuang, SimonPai, Tmillsclare, Tomyeh

Tablelayout *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Tablelayout *Contributors:* Benbai, Hawk, Jumperchen, SimonPai, Southerncrossie, Sphota, Tmillsclare, Tomyeh, Zkwikiadmin

TableChildren *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Tablelayout/TableChildren *Contributors:* Benbai, Hawk, Iantsai, Jumperchen, SimonPai, Southerncrossie, Tmillsclare

Vbox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Vbox *Contributors:* Ejikipaticy, Flyworld, Hawk, Jameschu, Jimmyshiau, Jumperchen, Peterkuo, Robertwenzel, Sphota, Tmillsclare, Tomyeh, Vincent

Vlayout *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Layouts/Vlayout *Contributors:* Hawk, Jumperchen, Tomyeh

Multimedia and Miscellaneous *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Multimedia_and_Miscellaneous *Contributors:* Tmillsclare, Zkwikiadmin

Audio *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Multimedia_and_Miscellaneous/Audio *Contributors:* Hawk, Leonlee, Raymondchao, Rudyhuang, SimonPai, Sphota, Tmillsclare

Barcode *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Multimedia_and_Miscellaneous/Barcode *Contributors:* Hawk, Klyvechen

BarcodeScanner *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Multimedia_and_Miscellaneous/BarcodeScanner *Contributors:* Hawk, Jeanher, Klyvechen, Leonlee, Matthieu, Rudyhuang

Cropper *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Multimedia_and_Miscellaneous/Cropper *Contributors:* Hawk, Jeanher, Wenninghsu

Camera *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Multimedia_and_Miscellaneous/Camera *Contributors:* Charlesqiu, Elenalin, Hawk, Jeanher, Michellechen

Flash *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Multimedia_and_Miscellaneous/Flash *Contributors:* Hawk, SimonPai, Sphota, Tmillsclare

Pdfviewer *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Multimedia_and_Miscellaneous/Pdfviewer *Contributors:* Hawk, Jeanher, Rudyhuang

Video *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Multimedia_and_Miscellaneous/Video *Contributors:* Charlesqiu, Elenalin, Hawk, Jameschu, Jeanher, Michellechen, Rudyhuang

Track *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Multimedia_and_Miscellaneous/Track *Contributors:* Hawk, Rudyhuang

Supplementary *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Supplementary *Contributors:* Alicelin, Tmillsclare

Auxhead *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Supplementary/Auxhead *Contributors:* Hawk, Iantsai, Jimmyshiau, SimonPai, Tmillsclare, Tomyeh, Zkwikiadmin

Auxheader *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Supplementary/Auxheader *Contributors:* Alicelin, Hawk, Iantsai, Jameschu, Jimmyshiau, SimonPai, Tmillsclare, Tomyeh, Zkwikiadmin

Cell *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Supplementary/Cell *Contributors:* Alicelin, Hawk, Jimmyshiau, Jumperchen, SimonPai, Tmillsclare, Tomyeh

Coachmark *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Supplementary/Coachmark *Contributors:* Hawk, Jeanher, Leonlee

Frozen *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Supplementary/Frozen *Contributors:* Hawk, Jumperchen, Robertwenzel, SimonPai, Tmillsclare

Paging *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Supplementary/Paging *Contributors:* Alicelin, Char, Chunfuchang, Hawk, Jimmyshiau, Peterkuo, SimonPai, Tmillsclare, Tomyeh, Wenninghsu

Stepbar *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Supplementary/Stepbar *Contributors:* Hawk, Jameschu, Jeanher, Rudyhuang

Step *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Supplementary/Stepbar/Step *Contributors:* Hawk, Jameschu, Jeanher, Rudyhuang

Events *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events *Contributors:* Tmillsclare, Zwikiadmin

AfterSizeEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/AfterSizeEvent *Contributors:* Hawk, Southerncrossie, Vincent

BandScrollEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/BandScrollEvent *Contributors:* Hawk, Jimmyshiau, Tmillsclare

CheckEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/CheckEvent *Contributors:* Alicelin, Hawk, Peterkuo, Tmillsclare, Zwikiadmin

ColSizeEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/ColSizeEvent *Contributors:* Hawk, Jimmyshiau, Peterkuo, SimonPai, Tmillsclare, Zwikiadmin

CreateEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/CreateEvent *Contributors:* Alicelin, Hawk, Peterkuo, Tmillsclare, Tomyeh, Zwikiadmin

DropEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/DropEvent *Contributors:* Hawk, Samchuang, Tmillsclare, Zwikiadmin

ErrorEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/ErrorEvent *Contributors:* Alicelin, Hawk, Jimmyshiau, Samchuang, Tmillsclare, Zwikiadmin

Event *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/Event *Contributors:* Hawk, Samchuang, Tmillsclare

HistoryPopStateEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/HistoryPopStateEvent *Contributors:* Hawk, Phoebelin, Rudyhuang

InfoChangeEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/InfoChangeEvent *Contributors:* Hawk, Henrichen, Tmillsclare

InputEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/InputEvent *Contributors:* Hawk, SimonPai, Tmillsclare, Zwikiadmin

KeyEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/KeyEvent *Contributors:* Ashishd, Char, Hawk, SimonPai, Tmillsclare, Tomyeh, Zwikiadmin

MapDropEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/MapDropEvent *Contributors:* Hawk, Henrichen, Tmillsclare

MapMouseEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/MapMouseEvent *Contributors:* Hawk, Henrichen, Tmillsclare

MapMoveEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/MapMoveEvent *Contributors:* Hawk, Henrichen, Tmillsclare

MapTypeChangeEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/MapTypeChangeEvent *Contributors:* Hawk, Henrichen, Tmillsclare

MapZoomEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/MapZoomEvent *Contributors:* Hawk, Henrichen, Tmillsclare

MouseEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/MouseEvent *Contributors:* Alicelin, Hawk, SimonPai, Tmillsclare, Tomyeh, Zwikiadmin

MoveEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/MoveEvent *Contributors:* Ashishd, Hawk, SimonPai, Tmillsclare

OccurEventSelectEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/OccurEventSelectEvent *Contributors:* Hawk, Jimmyshiau, Tmillsclare

OpenEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/OpenEvent *Contributors:* Ashishd, Hawk, Tmillsclare, Zwikiadmin

OverPlotEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/OverPlotEvent *Contributors:* Hawk, Jumperchen

PageSizeEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/PageSizeEvent *Contributors:* Ashishd, Hawk, Tmillsclare, Zwikiadmin

PagingEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/PagingEvent *Contributors:* Hawk, Iantsai, Tmillsclare, Zwikiadmin

PortalMoveEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/PortalMoveEvent *Contributors:* Hawk, Iantsai, Tmillsclare, Zwikiadmin

ScrollEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/ScrollEvent *Contributors:* Hawk, Iantsai, Tmillsclare, Zwikiadmin

SelectEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/SelectEvent *Contributors:* Hawk, Jimmyshiau, Jumperchen, Noahhuang, Tmillsclare, Zwikiadmin

SelectionEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/SelectionEvent *Contributors:* Hawk, Jimmyshiau, Tmillsclare, Zwikiadmin

SizeEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/SizeEvent *Contributors:* Hawk, Jimmyshiau, Tmillsclare, Zwikiadmin

VisibilityChangeEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/VisibilityChangeEvent *Contributors:* Hawk

UploadEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/UploadEvent *Contributors:* Alicelin, Hawk, Jumperchen, Tmillsclare, Zwikiadmin

ZIndexEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Events/ZIndexEvent *Contributors:* Hawk, Jumperchen, Tmillsclare, Zwikiadmin

Supporting Classes *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Supporting_Classes *Contributors:* Tmillsclare

AbstractListModel *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Supporting_Classes/AbstractListModel *Contributors:* Ashishd, Hawk, Tmillsclare, Tomyeh

Constraint *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Supporting_Classes/Constraint *Contributors:* Ashishd, Hawk, Tmillsclare

Constrained *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Supporting_Classes/Constrained *Contributors:* Hawk, Sphota, Tmillsclare

ListitemRenderer *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Supporting_Classes/ListitemRenderer *Contributors:* Hawk, Sphota, Tmillsclare

ListModel *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Supporting_Classes/ListModel *Contributors:* Hawk, Peterkuo, Tmillsclare

Messagebox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Supporting_Classes/Messagebox *Contributors:* Alicelin, Hawk, Jumperchen, Matthieu, MontyPan, Peterkuo, Raymondchao, SimonPai, Tmillsclare, Tomyeh, Tonyq

RendererCtrl *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Supporting_Classes/RendererCtrl *Contributors:* Alicelin, Hawk, SimonPai, Tmillsclare

SimpleConstraint *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Supporting_Classes/SimpleConstraint *Contributors:* Hawk, SimonPai, Tmillsclare

SimpleListModel *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Supporting_Classes/SimpleListModel *Contributors:* Hawk, Iantsai, Tmillsclare

XHTML Components *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/XHTML_Components *Contributors:* Flyworld, Malta, Tmillsclare, Tomyeh

Encoding URLs *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/XHTML_Components/Encoding_URLs *Contributors:* Alicelin, Hawk, Tmillsclare

In Pure Java *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/XHTML_Components/In_Pure_Java *Contributors:* Robertwenzel, Tmillsclare

The Difference Between XUL and XHTML Components *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/XHTML_Components/The_Difference_Between_XUL_and_XHTML_Components *Contributors:* Tmillsclare, Tomyeh

XML Components *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/XML_Components *Contributors:* Alicelin, Tomyeh

Transformer *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/XML_Components/Transformer *Contributors:* Hawk, Tomyeh

Annotation *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Annotation *Contributors:* Hawk

Data Binding *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Annotation/Data_Binding *Contributors:* Hawk

Tablet Devices *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices *Contributors:* Jumperchen

Configuration *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/Configuration *Contributors:* Vincent

Viewport *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/Configuration/Viewport *Contributors:* Hawk, Southerncrossie, Vincent

Components *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/Components *Contributors:* Vincent

ScrollView *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/Components/ScrollView *Contributors:* Chunfuchang, DanielHsu, Hawk, Southerncrossie, Vincent

Events *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/Events *Contributors:* Jumperchen

SwipeEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/Events/SwipeEvent *Contributors:* DanielHsu, Hawk, Jumperchen

ClientInfoEvent *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/Events/ClientInfoEvent *Contributors:* Hawk, Jumperchen

UI Enhancements *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/UI_Enhancements *Contributors:* Jumperchen

Borderlayout *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/UI_Enhancements/Borderlayout *Contributors:* Hawk, Jumperchen, Vincent

Calendar *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/UI_Enhancements/Calendar *Contributors:* Hawk, Jumperchen, Vincent

Cardlayout *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/UI_Enhancements/Cardlayout *Contributors:* Hawk, MontyPan

Colorbox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/UI_Enhancements/Colorbox *Contributors:* Hawk, Jumperchen

Combobox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/UI_Enhancements/Combobox *Contributors:* Hawk, Jumperchen

Datebox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/UI_Enhancements/Datebox *Contributors:* Hawk, Jumperchen

Doublespinner *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/UI_Enhancements/Doublespinner *Contributors:* Hawk, Jumperchen

Grid *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/UI_Enhancements/Grid *Contributors:* Hawk, Jumperchen, Vincent

Groupbox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/UI_Enhancements/Groupbox *Contributors:* Hawk, Jumperchen, Vincent

Listbox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/UI_Enhancements/Listbox *Contributors:* Hawk, Jumperchen, Vincent

Paging *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/UI_Enhancements/Paging *Contributors:* Hawk, Jumperchen, Vincent

Panel *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/UI_Enhancements/Panel *Contributors:* Hawk, Jumperchen, Vincent

Tabbox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/UI_Enhancements/Tabbox *Contributors:* Hawk, Jumperchen, Vincent

Timebox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/UI_Enhancements/Timebox *Contributors:* Hawk, Jumperchen

Tree *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/UI_Enhancements/Tree *Contributors:* Hawk, Jumperchen, Vincent

Spinner *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/UI_Enhancements/Spinner *Contributors:* Hawk, Jumperchen

Window *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/UI_Enhancements/Window *Contributors:* Hawk, Jumperchen, Vincent

Unsupported Molds *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/Unsupported_Molds *Contributors:* Hawk, Jumperchen

Limitation *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Tablet_Devices/Limitation *Contributors:* Hawk, Jeanher, Jumperchen, Vincent

Accessibility *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility *Contributors:* Hawk, Jameschu, Jeanher

Containers *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Containers *Contributors:* Rudyhuang

Drawer *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Containers/Drawer *Contributors:* Hawk, Jameschu, Rudyhuang

Tabbox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Containers/Tabbox *Contributors:* Hawk, Jameschu, Rudyhuang

Data *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Data *Contributors:* Jameschu

Biglistbox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Data/Biglistbox *Contributors:* Hawk, Jameschu

Grid *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Data/Grid *Contributors:* Hawk, Jameschu, Leonlee

Listbox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Data/Listbox *Contributors:* Hawk, Jameschu

Tree *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Data/Tree *Contributors:* Hawk, Jameschu

Essential Components *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Essential_Components *Contributors:* Jameschu

Include *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Essential_Components/Include *Contributors:* Hawk, Jameschu, Rudyhuang

Menubar *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Essential_Components/Menubar *Contributors:* Hawk, Jameschu, Jeanher, Rudyhuang

Navbar *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Essential_Components/Navbar *Contributors:* Hawk, Jameschu, Jeanher, Rudyhuang

Rating *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Essential_Components/Rating *Contributors:* Hawk, Jameschu, Jeanher, Leonlee

Toolbar *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Essential_Components/Toolbar *Contributors:* Hawk, Jameschu, Jeanher, Leonlee

Input *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Input *Contributors:* Jameschu

Bandbox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Input/Bandbox *Contributors:* Hawk, Jameschu, Leonlee

Calendar *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Input/Calendar *Contributors:* Hawk, Jameschu, Rudyhuang

Cascader *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Input/Cascader *Contributors:* Hawk, Jameschu

Chosenbox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Input/Chosenbox *Contributors:* Hawk, Jameschu

Colorbox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Input/Colorbox *Contributors:* Hawk, Jameschu

Datebox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Input/Datebox *Contributors:* Hawk, Jameschu, Jeanher, Rudyhuang

Multislider *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Input/Multislider *Contributors:* Hawk, Jameschu

Rangeslider *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Input/Rangeslider *Contributors:* Hawk, Jameschu

Searchbox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Input/Searchbox *Contributors:* Hawk, Jameschu

Slider *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Input/Slider *Contributors:* Hawk, Jameschu, Jeanher, Leonlee

Timebox *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Input/Timebox *Contributors:* Hawk, Jameschu, Jeanher, Rudyhuang

Timepicker *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Input/Timepicker *Contributors:* Hawk, Jameschu

Layouts *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Layouts *Contributors:* Jameschu

Borderlayout *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Layouts/Borderlayout *Contributors:* Hawk, Jameschu, Leonlee

Cardlayout *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Layouts/Cardlayout *Contributors:* Hawk, Jameschu, Jeanher, Leonlee

Linelayout *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Layouts/Linelayout *Contributors:* Hawk, Jameschu, Leonlee

Organigram *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Layouts/Organigram *Contributors:* Hawk, Jameschu, Leonlee

Portallayout *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Layouts/Portallayout *Contributors:* Hawk, Jameschu, Leonlee

Splitter *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Layouts/Splitter *Contributors:* Hawk, Jameschu

Splitlayout *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Layouts/Splitlayout *Contributors:* Hawk, Jameschu

Multimedia and Miscellaneous *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Multimedia_and_Miscellaneous *Contributors:* Jameschu

Barcode *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Multimedia_and_Miscellaneous/Barcode *Contributors:* Hawk, Jameschu

BarcodeScanner *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Multimedia_and_Miscellaneous/BarcodeScanner *Contributors:* Hawk, Jameschu

Pdfviewer *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Multimedia_and_Miscellaneous/Pdfviewer *Contributors:* Hawk, Jameschu, Rudyhuang

Supplementary *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Supplementary *Contributors:* Rudyhuang

Stepbar *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Supplementary/Stepbar *Contributors:* Hawk, Jameschu, Rudyhuang

Unsupported Components *Source:* http://10.1.3.180/index.php?title=ZK_Component_Reference/Accessibility/Unsupported_Components *Contributors:* Hawk, Jeanher, Rudyhuang

Image Sources, Licenses and Contributors

Image:CrInputs.png Source: http://10.1.3.180/index.php?title=File:CrInputs.png License: unknown Contributors: Tomyeh

Image:CrCustomConstraint.png Source: http://10.1.3.180/index.php?title=File:CrCustomConstraint.png License: unknown Contributors: Tomyeh

File:zk_textbox_placeholder.png Source: http://10.1.3.180/index.php?title=File:Zk_textbox_placeholder.png License: unknown Contributors: Tmillsclare

File:NumberInputElement-Locales.png Source: http://10.1.3.180/index.php?title=File:NumberInputElement-Locales.png License: unknown Contributors: Jumperchen

File:ZKComRef_Caption_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Caption_Example.png License: unknown Contributors: Jimmyshiau, Jumperchen

Image:ZKComRef_Div_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Div_Example.png License: unknown Contributors: SimonPai, Sphota

Image:version_ee.png Source: http://10.1.3.180/index.php?title=File:Version_ee.png License: unknown Contributors: Tmillsclare

File:ZK-Drawer-Example.png Source: http://10.1.3.180/index.php?title=File:ZK-Drawer-Example.png License: unknown Contributors: Jeanher, Rudyhuang

File:ZK-Drawer-Example.gif Source: http://10.1.3.180/index.php?title=File:ZK-Drawer-Example.gif License: unknown Contributors: Hawk

File:ZK-Drawer-Autodrop.gif Source: http://10.1.3.180/index.php?title=File:ZK-Drawer-Autodrop.gif License: unknown Contributors: ProtonChang, Rudyhuang

File:ZKComRef_Fragment_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Fragment_Example.png License: unknown Contributors: Rudyhuang

Image:ZKComRef_Groupbox_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Groupbox_Example.png License: unknown Contributors: Sphota, Tmillsclare

Image:ZKComRef_Groupbox_ContentStyle.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Groupbox_ContentStyle.png License: unknown Contributors: Jumperchen

File:groupbox-3d.jpg Source: http://10.1.3.180/index.php?title=File:Groupbox-3d.jpg License: unknown Contributors: Tomyeh

Image:groupbox_mold_default.png Source: http://10.1.3.180/index.php?title=File:Groupbox_mold_default.png License: unknown Contributors: Peterkuo

Image:groupbox_mold_3d.png Source: http://10.1.3.180/index.php?title=File:Groupbox_mold_3d.png License: unknown Contributors: Peterkuo

Image:ZKComRef_Idspace_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Idspace_Example.png License: unknown Contributors: Benbai

File:Inputgroup_basic.png Source: http://10.1.3.180/index.php?title=File:Inputgroup_basic.png License: unknown Contributors: Rudyhuang

File:Inputgroup_vertical.png Source: http://10.1.3.180/index.php?title=File:Inputgroup_vertical.png License: unknown Contributors: Rudyhuang

Image:ZKComRef_Panel_Simple_Examples.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Panel_Simple_Examples.PNG License: unknown Contributors: Hawk, Jumperchen, Peterkuo

Image:DrPanelBorder.png Source: http://10.1.3.180/index.php?title=File:DrPanelBorder.png License: unknown Contributors: Tomyeh

File:Panel-des.gif Source: http://10.1.3.180/index.php?title=File:Panel-des.gif License: unknown Contributors: Elton776

Image:ZKComRef_Span_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Span_Example.png License: unknown Contributors: SimonPai

Image:ZKComRef_Tabbox_Examples.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Tabbox_Examples.PNG License: unknown Contributors: Jumperchen, Samchuang

Image:ZKComRef_Tabbox_Toolbar_Examples.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Tabbox_Toolbar_Examples.png License: unknown Contributors: Jumperchen

Image:tabbox_maximalHeight_0.png Source: http://10.1.3.180/index.php?title=File:Tabbox_maximalHeight_0.png License: unknown Contributors: Noahhuang

Image:tabbox_maximalHeight_1.png Source: http://10.1.3.180/index.php?title=File:Tabbox_maximalHeight_1.png License: unknown Contributors: Noahhuang

Image:tabbox_maximalHeight_2.png Source: http://10.1.3.180/index.php?title=File:Tabbox_maximalHeight_2.png License: unknown Contributors: Noahhuang

Image:tabbox_mold_default.png Source: http://10.1.3.180/index.php?title=File:Tabbox_mold_default.png License: unknown Contributors: Peterkuo

Image:tabbox_mold_accordion.png Source: http://10.1.3.180/index.php?title=File:Tabbox_mold_accordion.png License: unknown Contributors: Peterkuo

Image:tabbox_mold_accordion-lite.png Source: http://10.1.3.180/index.php?title=File:Tabbox_mold_accordion-lite.png License: unknown Contributors: Peterkuo

Image:tabbox_orient_top.png Source: http://10.1.3.180/index.php?title=File:Tabbox_orient_top.png License: unknown Contributors: Noahhuang

Image:tabbox_orient_vertical.png Source: http://10.1.3.180/index.php?title=File:Tabbox_orient_vertical.png License: unknown Contributors: Noahhuang

Image:tabbox_orient_vertical-right.png Source: http://10.1.3.180/index.php?title=File:Tabbox_orient_vertical-right.png License: unknown Contributors: Noahhuang

Image:tabbox_orient_bottom.png Source: http://10.1.3.180/index.php?title=File:Tabbox_orient_bottom.png License: unknown Contributors: Noahhuang

Image:ZKComRef_Containers_Tab.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Containers_Tab.PNG License: unknown Contributors: Jumperchen, SimonPai

Image:ZKComRef_Containers_Tab_Caption.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Containers_Tab_Caption.PNG License: unknown Contributors: Jumperchen

Image:ZKComRef_Containers_Tabs.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Containers_Tabs.PNG License: unknown Contributors: Jumperchen, SimonPai

Image:ZKComRef_Window_Multiple_Examples.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Window_Multiple_Examples.PNG License: unknown Contributors: Jumperchen, Tmillsclare

File:Modalwindow.png Source: http://10.1.3.180/index.php?title=File:Modalwindow.png License: unknown Contributors: Tomyeh

File:Icon_info.png Source: http://10.1.3.180/index.php?title=File:Icon_info.png License: unknown Contributors: Hawk, Tmillsclare

Image:1000000000000CE000000546D42136E.png Source: http://10.1.3.180/index.php?title=File:1000000000000CE000000546D42136E.png License: unknown Contributors: Maya001122

Image:1000000000000CB000003292CB8174.png Source: http://10.1.3.180/index.php?title=File:1000000000000CB000003292CB8174.png License: unknown Contributors: Maya001122

Image:1000000000009C0000006819656516.png Source: http://10.1.3.180/index.php?title=File:1000000000009C0000006819656516.png License: unknown Contributors: Maya001122

Image:100000000000164000004CEB4969A9.png Source: http://10.1.3.180/index.php?title=File:100000000000164000004CEB4969A9.png License: unknown Contributors: Maya001122

Image:1000000000000CD00000042FABAB4CE.png Source: http://10.1.3.180/index.php?title=File:1000000000000CD00000042FABAB4CE.png License: unknown Contributors: Maya001122

Image:ZKComRef_Grid_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Grid_Example.png License: unknown Contributors: Jimmyshiau

Image:grid_onAfterRender.png Source: http://10.1.3.180/index.php?title=File:Grid_onAfterRender.png License: unknown Contributors: Jimmyshiau

Image:ZKComRef_Grid_Paging.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Grid_Paging.png License: unknown Contributors: Jimmyshiau

Image:ZKComRef_Grid_Paginal.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Grid_Paginal.png License: unknown Contributors: Jimmyshiau

File:Grid_autopageSize.png Source: http://10.1.3.180/index.php?title=File:Grid_autopageSize.png License: unknown Contributors: Elton776, Tmillsclare

Image:ZKComRef_Grid_LiveData.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Grid_LiveData.png License: unknown Contributors: Jimmyshiau

Image:ZKComRef_Grid_ScrollableGrid.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Grid_ScrollableGrid.png License: unknown Contributors: Jimmyshiau

Image:ZKComRef_Grid_AuxiliaryHeaders.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Grid_AuxiliaryHeaders.png License: unknown Contributors: Jimmyshiau

File:Auxheader_rowspan_limitation.jpg Source: http://10.1.3.180/index.php?title=File:Auxheader_rowspan_limitation.jpg License: unknown Contributors: Jimmyshiau

File:Auxheader_rowspan_limitation01.jpg Source: http://10.1.3.180/index.php?title=File:Auxheader_rowspan_limitation01.jpg License: unknown Contributors: Jimmyshiau

File:Auxheader_rowspan_limitation02.jpg Source: http://10.1.3.180/index.php?title=File:Auxheader_rowspan_limitation02.jpg License: unknown Contributors: Jimmyshiau

File:Auxheader_rowspan_limitation03.jpg Source: http://10.1.3.180/index.php?title=File:Auxheader_rowspan_limitation03.jpg License: unknown Contributors: Jimmyshiau

Image:ZKComRef_Grid_Nospan.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Grid_Nospan.png License: unknown Contributors: Samchuang

Image:ZKComRef_Grid_Span.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Grid_Span.png License: unknown Contributors: Samchuang

Image:ZKComRef_Grid_Detail.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Grid_Detail.png License: unknown Contributors: Jumperchen

Image: ZKComRef_Grid_Columns_Menu.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Grid_Columns_Menu.png License: unknown Contributors: Jumperchen

Image: ZKComRef_Grid_Columns_Menu_Ungroup.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Grid_Columns_Menu_Ungroup.PNG License: unknown Contributors: Jumperchen

Image:ZKComRef_Grid_Grouping.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Grid_Grouping.png License: unknown Contributors: Jumperchen
Image:grid_mold_default.png Source: http://10.1.3.180/index.php?title=File:Grid_mold_default.png License: unknown Contributors: Peterkuo
Image:grid_mold.paging.png Source: http://10.1.3.180/index.php?title=File:Grid_mold.paging.png License: unknown Contributors: Peterkuo
Image:version_pe-ee.png Source: http://10.1.3.180/index.php?title=File:Version_pe-ee.png License: unknown Contributors: Timmclare
Image:ZKComRef_Grid_Columns_Menu.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Grid_Columns_Menu.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Grid_Columns_Customized_Menu.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Grid_Columns_Customized_Menu.png License: unknown Contributors: Jumperchen
Image:detail.png Source: http://10.1.3.180/index.php?title=File:Detail.png License: unknown Contributors: Jimmyshiau
Image:ZKComRef_Foot_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Foot_Example.png License: unknown Contributors: Jimmyshiau, Jumperchen
Image:ZKComRef_Group_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Group_Example.png License: unknown Contributors: Jimmyshiau, Jumperchen
Image:ZKComRef_Listbox_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Listbox_Example.png License: unknown Contributors: Jimmyshiau, Sphota
Image:ZKComRef_Listbox_Example_ContainComponents.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Listbox_Example_ContainComponents.png License: unknown Contributors: Jimmyshiau
Image:10000000000008500000343B08C7D1.png Source: http://10.1.3.180/index.php?title=File:10000000000008500000343B08C7D1.png License: unknown Contributors: Hawk, Maya001122
Image:Select-mold-optgroup.png Source: http://10.1.3.180/index.php?title=File:Select-mold-optgroup.png License: unknown Contributors: Rudyhuang
File:listbox-select-separately.png Source: http://10.1.3.180/index.php?title=File>Listbox-select-separately.png License: unknown Contributors: Hawk
File:listbox-select-consecutive.png Source: http://10.1.3.180/index.php?title=File>Listbox-select-consecutive.png License: unknown Contributors: Hawk
Image:ZKComRef_Listbox_Checkmark.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Listbox_Checkmark.png License: unknown Contributors: Jimmyshiau
Image:ZKComRef_Listbox_Checkmark2.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Listbox_Checkmark2.png License: unknown Contributors: Jimmyshiau
Image:ZKComRef_Listbox_Sorting.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Listbox_Sorting.png License: unknown Contributors: Hawk, Jimmyshiau
Image:ZKComRef_Listbox_Auto_Sorting.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Listbox_Auto_Sorting.png License: unknown Contributors: Hawk, Jimmyshiau, Jumperchen
Image:ZKComRef_Listbox_LiveData.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Listbox_LiveData.png License: unknown Contributors: Jimmyshiau
Image:ZKComRef_Listbox_SingleColumn.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Listbox_SingleColumn.png License: unknown Contributors: Jimmyshiau
Image:ZKComRef_Listbox_MultiColumn.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Listbox_MultiColumn.png License: unknown Contributors: Jimmyshiau
Image:ZKComRef_Listbox_ColumnHeaders.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Listbox_ColumnHeaders.png License: unknown Contributors: Jimmyshiau
Image:ZKComRef_Listbox_ColumnFooters.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Listbox_ColumnFooters.png License: unknown Contributors: Jimmyshiau
Image:10000000000004900000488DCF4463.png Source: http://10.1.3.180/index.php?title=File:10000000000004900000488DCF4463.png License: unknown Contributors: Maya001122
Image:ZKComRef_Listbox_Scrollable.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Listbox_Scrollable.png License: unknown Contributors: Jimmyshiau
Image:ZKComRef_Listbox_Nospan.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Listbox_Nospan.png License: unknown Contributors: Samchuang
Image:ZKComRef_Listbox_Span.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Listbox_Span.png License: unknown Contributors: Samchuang
File:listbox-sizedByContent.png Source: http://10.1.3.180/index.php?title=File>Listbox-sizedByContent.png License: unknown Contributors: Hawk
Image:listbox_onAfterRender.png Source: http://10.1.3.180/index.php?title=File>Listbox_onAfterRender.png License: unknown Contributors: Jimmyshiau
Image: ZKComRef_Listbox_Columns_Menu.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Listbox_Columns_Menu.PNG License: unknown Contributors: Jumperchen
Image: ZKComRef_Listbox_Columns_Menu_Ungroup.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Listbox_Columns_Menu_Ungroup.PNG License: unknown Contributors: Jimmyshiau, Jumperchen
Image:ZKComRef_Listbox_Grouping.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Listbox_Grouping.png License: unknown Contributors: Jumperchen
Image:listbox_mold_default.png Source: http://10.1.3.180/index.php?title=File>Listbox_mold_default.png License: unknown Contributors: Peterkuo
Image:listbox_mold_select.png Source: http://10.1.3.180/index.php?title=File>Listbox_mold_select.png License: unknown Contributors: Peterkuo
Image:listbox_mold.paging.png Source: http://10.1.3.180/index.php?title=File>Listbox_mold.paging.png License: unknown Contributors: Peterkuo
Image:ZKComRef_Listgroup_Example.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Listgroup_Example.PNG License: unknown Contributors: Jimmyshiau, Peterkuo
Image:ZKComRef_Listbox_Columns_Customized_Menu.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Listbox_Columns_Customized_Menu.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Listbox_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Listbox_Example.png License: unknown Contributors: Jimmyshiau, Sphota
Image:ZKComRef_Tree_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Tree_Example.png License: unknown Contributors: Jimmyshiau
Image:tree_onAfterRender.png Source: http://10.1.3.180/index.php?title=File:Tree_onAfterRender.png License: unknown Contributors: Jimmyshiau
Image:ZKComRef_Tree_checkmark.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Tree_checkmark.png License: unknown Contributors: Jimmyshiau
File:scrollable_tree.png Source: http://10.1.3.180/index.php?title=File:Scrollable_tree.png License: unknown Contributors: Vincent
Image:tree_mold_default.png Source: http://10.1.3.180/index.php?title=File:Tree_mold_default.png License: unknown Contributors: Peterkuo
Image:tree_mold.paging.png Source: http://10.1.3.180/index.php?title=File:Tree_mold.paging.png License: unknown Contributors: Peterkuo
Image:ZKComRef_Treelistem.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Treelistem.png License: unknown Contributors: Jimmyshiau
Image:ZKComRef_Biglistbox.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Biglistbox.PNG License: unknown Contributors: Jumperchen
File:Capture1.PNG Source: http://10.1.3.180/index.php?title=File:Capture1.PNG License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart.png License: unknown Contributors: Jimmyshiau, SimonPai
Image:ZKComRef_Chart_Pie_3D.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_Pie_3D.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_Ring.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_Ring.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_Bar.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_Bar.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_Bar_3D.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_Bar_3D.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_Stacked_Bar.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_Stacked_Bar.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_Stacked_Bar_3D.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_Stacked_Bar_3D.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_Line.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_Line.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_3D.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_3D.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_Area.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_Area.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_Stacked_Area.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_Stacked_Area.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_Waterfall.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_Waterfall.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_Polar.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_Polar.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_Scatter.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_Scatter.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_Timeseries.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_Timeseries.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_XY_Area.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_XY_Area.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_XY_Line.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_XY_Line.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_Step_Area.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_Step_Area.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_Step.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_Step.png License: unknown Contributors: Jumperchen

Image:ZKComRef_Chart_XY_Stacked_Area.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_XY_Stacked_Area.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_XY_Bar.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_XY_Bar.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_Histogram.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_Histogram.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_Candlestick.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_Candlestick.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_Highlow.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_Highlow.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_Bubble.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_Bubble.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_HBubble.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_HBubble.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_Wafer_Map.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_Wafer_Map.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_Gantt.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_Gantt.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_Wind.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_Wind.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Chart_Dial.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_Dial.png License: unknown Contributors: Jumperchen
File:ZKComRef_Chart_Font_Examples.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Chart_Font_Examples.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Flashchart_Examples.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Flashchart_Examples.PNG License: unknown Contributors: Samchuang, SimonPai
Image:Barchart_2d.png Source: http://10.1.3.180/index.php?title=File:Barchart_2d.png License: unknown Contributors: Murasakiwu
Image:Barchart_2d_h.png Source: http://10.1.3.180/index.php?title=File:Barchart_2d_h.png License: unknown Contributors: Murasakiwu
Image:Barchart_3d.png Source: http://10.1.3.180/index.php?title=File:Barchart_3d.png License: unknown Contributors: Murasakiwu
File:Fusionchart_barchartconfig.png Source: http://10.1.3.180/index.php?title=File:Fusionchart_barchartconfig.png License: unknown Contributors: Tonyq
Image:Linechart.png Source: http://10.1.3.180/index.php?title=File:Linechart.png License: unknown Contributors: Murasakiwu
Image:Piechart_2d.png Source: http://10.1.3.180/index.php?title=File:Piechart_2d.png License: unknown Contributors: Murasakiwu
Image:Piechart_3d.png Source: http://10.1.3.180/index.php?title=File:Piechart_3d.png License: unknown Contributors: Murasakiwu
Image:Funnelchart.png Source: http://10.1.3.180/index.php?title=File:Funnelchart.png License: unknown Contributors: Robertwenzel, Tmillsclare
Image:Combinationchart_2d.png Source: http://10.1.3.180/index.php?title=File:Combinationchart_2d.png License: unknown Contributors: Murasakiwu
Image:Combinationchart_3d.png Source: http://10.1.3.180/index.php?title=File:Combinationchart_3d.png License: unknown Contributors: Murasakiwu
Image:Stackedchart_2d.png Source: http://10.1.3.180/index.php?title=File:Stackedchart_2d.png License: unknown Contributors: Murasakiwu
Image:Stackedchart_3d.png Source: http://10.1.3.180/index.php?title=File:Stackedchart_3d.png License: unknown Contributors: Murasakiwu
Image:Areachart.png Source: http://10.1.3.180/index.php?title=File:Areachart.png License: unknown Contributors: Murasakiwu
Image:Gantt.png Source: http://10.1.3.180/index.php?title=File:Gantt.png License: unknown Contributors: Murasakiwu
Image:ZKComRef_Gmaps_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Gmaps_Example.png License: unknown Contributors: Henrichen, Jimmyshiau, Robertwenzel
Image:ZKComRef_Gimage_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Gimage_Example.png License: unknown Contributors: Henrichen, Jimmyshiau
Image:ZKComRef_Gpolyline_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Gpolyline_Example.png License: unknown Contributors: Henrichen, Jimmyshiau
Image:ZKComRef_Gpolygon_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Gpolygon_Example.png License: unknown Contributors: Henrichen, Jimmyshiau
Image:ZKComRef_Gscreen_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Gscreen_Example.png License: unknown Contributors: Henrichen, Jimmyshiau
Image:ZKComRef_Jasperreport_Examples.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Jasperreport_Examples.PNG License: unknown Contributors: Samchuang
Image:ZKComRef_Timeline.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Timeline.png License: unknown Contributors: Jimmyshiau, Jumperchen
Image:ZKComRef_TimeLine2.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_TimeLine2.png License: unknown Contributors: Jimmyshiau, Jumperchen
Image:ZKComRef_Timeplot.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Timeplot.png License: unknown Contributors: Jimmyshiau
Image:ZKComRef_A_Examples.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_A_Examples.PNG License: unknown Contributors: Samchuang, SimonPai
Image:ZKComRef_Anchornav.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Anchornav.png License: unknown Contributors: Jameschu
Image:ZKComRef_Applet_Examples.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Applet_Examples.PNG License: unknown Contributors: Samchuang
Image:ZKComRef_Button.jpg Source: http://10.1.3.180/index.php?title=File:ZKComRef_Button.jpg License: unknown Contributors: Jumperchen, SimonPai
Image:button_mold_default.png Source: http://10.1.3.180/index.php?title=File:Button_mold_default.png License: unknown Contributors: Peterkuo
Image:button_mold_trendy.png Source: http://10.1.3.180/index.php?title=File:Button_mold_trendy.png License: unknown Contributors: Peterkuo
Image:button_mold_os.png Source: http://10.1.3.180/index.php?title=File:Button_mold_os.png License: unknown Contributors: Peterkuo
Image:captcha.png Source: http://10.1.3.180/index.php?title=File:Captcha.png License: unknown Contributors: Elton776, Tmillsclare
Image:ZKComRef_Combobutton_with_Popup.jpg Source: http://10.1.3.180/index.php?title=File:ZKComRef_Combobutton_with_Popup.jpg License: unknown Contributors: Benbai
Image:ZKComRef_Combobutton_with_Menupopup.jpg Source: http://10.1.3.180/index.php?title=File:ZKComRef_Combobutton_with_Menupopup.jpg License: unknown Contributors: Benbai
Image:ZKComRef_Combobutton_ToolbarbuttonMold.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Combobutton_ToolbarbuttonMold.png License: unknown Contributors: Benbai
File:initial-run.png Source: http://10.1.3.180/index.php?title=File:Initial-run.png License: unknown Contributors: Tmillsclare
File:dragged-over.png Source: http://10.1.3.180/index.php?title=File:Dragged-over.png License: unknown Contributors: Tmillsclare
File:uploaded-image.png Source: http://10.1.3.180/index.php?title=File:Uploaded-image.png License: unknown Contributors: Tmillsclare
File:Dropupload_Anchor.png Source: http://10.1.3.180/index.php?title=File:Dropupload_Anchor.png License: unknown Contributors: Noahhuang
File:Dropupload_Anchor_1.png Source: http://10.1.3.180/index.php?title=File:Dropupload_Anchor_1.png License: unknown Contributors: Noahhuang
Image:DefaultFileUploadVeiwer.JPG Source: http://10.1.3.180/index.php?title=File:DefaultFileUploadVeiwer.JPG License: unknown Contributors: Elton776
File:CustomizedFileUploadVeiwer.JPG Source: http://10.1.3.180/index.php?title=File:CustomizedFileUploadVeiwer.JPG License: unknown Contributors: Elton776
Image:10000000000002AF000001BB582C2DD7.png Source: http://10.1.3.180/index.php?title=File:10000000000002AF000001BB582C2DD7.png License: unknown Contributors: Char
File:IePreventDownload.png Source: http://10.1.3.180/index.php?title=File:IePreventDownload.png License: unknown Contributors: Tomyeh
Image:ZKComRef_fisheyebar.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_fisheyebar.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Html.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Html.png License: unknown Contributors: Jumperchen, SimonPai
Image:ZKComRef_Iframe.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Iframe.png License: unknown Contributors: Jumperchen, SimonPai
Image:ZKComRef_Imagemap.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Imagemap.png License: unknown Contributors: Jumperchen, SimonPai
Image:ZKComRef_Label.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Label.PNG License: unknown Contributors: Jumperchen
Image:ZKComRef_Label_Example2.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Label_Example2.png License: unknown Contributors: Jimmyshiau
Image:ZKComRef_Label_Example3.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Label_Example3.png License: unknown Contributors: Jimmyshiau
Image:ZKComRef_Label_Text_ZK5.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Label_Text_ZK5.png License: unknown Contributors: Jimmyshiau
Image:ZKComRef_Label_Text_ZK3.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Label_Text_ZK3.png License: unknown Contributors: Jimmyshiau
Image:ZKComRef_Menu.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Menu.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Menu_onClick.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Menu_onClick.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Menuubar.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Menuubar.png License: unknown Contributors: Jumperchen

Image:ZKComRef_Menubar_Scrollable.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Menubar_Scrollable.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Menuitem.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Menuitem.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Menuseparator.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Menuseparator.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Nav.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Nav.png License: unknown Contributors: Raymondchao
Image:ZKComRef_Nav_badgeText.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Nav_badgeText.png License: unknown Contributors: Phoebelin, Raymondchao
Image:ZKComRef_Nav_hor.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Nav_hor.png License: unknown Contributors: Raymondchao
Image:ZKComRef_Nav_Hor_Cld.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Nav_Hor_Cld.png License: unknown Contributors: Phoebelin, Raymondchao
Image:ZKComRef_Nav_Hor_No.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Nav_Hor_No.png License: unknown Contributors: Phoebelin, Raymondchao
Image:ZKComRef_Nav_Ver_Cld.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Nav_Ver_Cld.png License: unknown Contributors: Phoebelin, Raymondchao
Image:ZKComRef_Nav_Ver_No.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Nav_Ver_No.png License: unknown Contributors: Phoebelin, Raymondchao
Image:ZKComRef_Navseparator.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Navseparator.png License: unknown Contributors: Phoebelin, Raymondchao
Image:ZKComRef_Popup.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Popup.PNG License: unknown Contributors: Jumperchen
Image:ZKComRef_Popup2.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Popup2.PNG License: unknown Contributors: Jumperchen
Image:ZKComRef_Popup_Position_601.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Popup_Position_601.png License: unknown Contributors: SimonPai
Image:ZKComRef_Popup_Position.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Popup_Position.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Popup_Beforestart.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Popup_Beforestart.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Progressmeter_Example.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Progressmeter_Example.PNG License: unknown Contributors: Jumperchen, Peterkuo
File:rating.gif Source: http://10.1.3.180/index.php?title=File:Rating.gif License: unknown Contributors: Hawk
Image:Selectbox-Example1.png Source: http://10.1.3.180/index.php?title=File:Selectbox-Example1.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Separator_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Separator_Example.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Space_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Space_Example.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Script_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Script_Example.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Style_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Style_Example.png License: unknown Contributors: Sphota
Image:ZKComRef_Toolbar_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Toolbar_Example.png License: unknown Contributors: Jumperchen, Sphota
File:Toolbar-overflowPopup.png Source: http://10.1.3.180/index.php?title=File:Toolbar-overflowPopup.png License: unknown Contributors: Charlesqiu
File:Toolbar-overflowPopupIconSclass.png Source: http://10.1.3.180/index.php?title=File:Toolbar-overflowPopupIconSclass.png License: unknown Contributors: KatherineLin
Image:toolbar_mold_default.png Source: http://10.1.3.180/index.php?title=File:Toolbar_mold_default.png License: unknown Contributors: Peterkuo
Image:toolbar_mold_panel.png Source: http://10.1.3.180/index.php?title=File:Toolbar_mold_panel.png License: unknown Contributors: Peterkuo
Image:ZKComRef_Toolbarbutton_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Toolbarbutton_Example.png License: unknown Contributors: Jumperchen, Sphota
Image:Toolbarbutton_togglemode.png Source: http://10.1.3.180/index.php?title=File:Toolbarbutton_togglemode.png License: unknown Contributors: Tonyq
Image:ZKComRef_Bandbox_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Bandbox_Example.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Bandbox_Autodrop.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Bandbox_Autodrop.PNG License: unknown Contributors: Jimmyshiau
Image:bandbox_mold_default.png Source: http://10.1.3.180/index.php?title=File:Bandbox_mold_default.png License: unknown Contributors: Peterkuo
Image:bandbox_mold_rounded.png Source: http://10.1.3.180/index.php?title=File:Bandbox_mold_rounded.png License: unknown Contributors: Peterkuo
Image:ZKComRef_Calendar_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Calendar_Example.png License: unknown Contributors: Jumperchen, Tmillsclare
File:dateRangeSelector.png Source: http://10.1.3.180/index.php?title=File:DateRangeSelector.png License: unknown Contributors: Hawk
Image:ZKComRef_Calendar_Example2.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Calendar_Example2.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Calendar_Week_Of_Year.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Calendar_Week_Of_Year.PNG License: unknown Contributors: Jumperchen
File:Cascader-example.png Source: http://10.1.3.180/index.php?title=File:Cascader-example.png License: unknown Contributors: Rudyhuang
Image:ZKComRef_Checkbox_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Checkbox_Example.png License: unknown Contributors: Jumperchen, Sphota
File:Switch-off.png Source: http://10.1.3.180/index.php?title=File:Switch-off.png License: unknown Contributors: Charlesqiu
File:Switch-on.png Source: http://10.1.3.180/index.php?title=File:Switch-on.png License: unknown Contributors: Charlesqiu
File:Switch-off-customized.png Source: http://10.1.3.180/index.php?title=File:Switch-off-customized.png License: unknown Contributors: Charlesqiu
File:Switch-on-customized.png Source: http://10.1.3.180/index.php?title=File:Switch-on-customized.png License: unknown Contributors: Charlesqiu
File:Toggle-off.png Source: http://10.1.3.180/index.php?title=File:Toggle-off.png License: unknown Contributors: Charlesqiu
File:Toggle-on.png Source: http://10.1.3.180/index.php?title=File:Toggle-on.png License: unknown Contributors: Charlesqiu
File:Toggle-off-customized.png Source: http://10.1.3.180/index.php?title=File:Toggle-off-customized.png License: unknown Contributors: Charlesqiu
File:Toggle-on-customized.png Source: http://10.1.3.180/index.php?title=File:Toggle-on-customized.png License: unknown Contributors: Charlesqiu
File:Tristate.png Source: http://10.1.3.180/index.php?title=File:Tristate.png License: unknown Contributors: Leonlee
File:Indeterminate.png Source: http://10.1.3.180/index.php?title=File:Indeterminate.png License: unknown Contributors: Charlesqiu
Image:CompREF_Chosenbox_msgEx_01.png Source: http://10.1.3.180/index.php?title=File:CompREF_Chosenbox_msgEx_01.png License: unknown Contributors: Benbai
Image:CompREF_Chosenbox_msgEx_02.png Source: http://10.1.3.180/index.php?title=File:CompREF_Chosenbox_msgEx_02.png License: unknown Contributors: Benbai
Image:CompREF_Chosenbox_msgEx_03.png Source: http://10.1.3.180/index.php?title=File:CompREF_Chosenbox_msgEx_03.png License: unknown Contributors: Benbai
Image:CompREF_Chosenbox_01.png Source: http://10.1.3.180/index.php?title=File:CompREF_Chosenbox_01.png License: unknown Contributors: Benbai
Image:ZKCompRef_CKEditor.png Source: http://10.1.3.180/index.php?title=File:ZKCompRef_CKEditor.png License: unknown Contributors: Jimmyshiau, SimonPai
Image:ZKCompRef_CKEditor2.png Source: http://10.1.3.180/index.php?title=File:ZKCompRef_CKEditor2.png License: unknown Contributors: Jimmyshiau, SimonPai
File:ZKCompRef_CKEditor_filebrowser.png Source: http://10.1.3.180/index.php?title=File:ZKCompRef_CKEditor_filebrowser.png License: unknown Contributors: Jimmyshiau
File:ZKCompRef_CKEditor_filebrowser2.png Source: http://10.1.3.180/index.php?title=File:ZKCompRef_CKEditor_filebrowser2.png License: unknown Contributors: Jimmyshiau
File:ZKCompRef_CKEditor_filebrowser3.png Source: http://10.1.3.180/index.php?title=File:ZKCompRef_CKEditor_filebrowser3.png License: unknown Contributors: Jimmyshiau
File:ZKCompRef_CKEditor_fileupload.png Source: http://10.1.3.180/index.php?title=File:ZKCompRef_CKEditor_fileupload.png License: unknown Contributors: Jimmyshiau
File:ZKCompRef_CKEditor_fileupload2.png Source: http://10.1.3.180/index.php?title=File:ZKCompRef_CKEditor_fileupload2.png License: unknown Contributors: Jimmyshiau
Image:ZKComRef_Carousel_Examples.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Carousel_Examples.PNG License: unknown Contributors: Samchuang
Image:ZKComRef_Carousel_Examples2.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Carousel_Examples2.PNG License: unknown Contributors: Jumperchen, Samchuang
Image:ZKComRef_Combobox_Example.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Combobox_Example.PNG License: unknown Contributors: Jumperchen, Peterkuo
Image:combobox_onAfterRender.png Source: http://10.1.3.180/index.php?title=File:Combobox_onAfterRender.png License: unknown Contributors: Jimmyshiau, Jumperchen
Image:ZKComRef_Combobox_Description.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Combobox_Description.PNG License: unknown Contributors: Jimmyshiau
Image:combobox_mold_default.png Source: http://10.1.3.180/index.php?title=File:Combobox_mold_default.png License: unknown Contributors: Peterkuo
Image:combobox_mold_rounded.png Source: http://10.1.3.180/index.php?title=File:Combobox_mold_rounded.png License: unknown Contributors: Peterkuo
Image:ZKComRef_Datebox_Example.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Datebox_Example.PNG License: unknown Contributors: Jumperchen, Peterkuo

Image:ZKComRef_Datebox_Timezone.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Datebox_Timezone.png License: unknown Contributors: Jumperchen
File:Roc-calendar.jpg Source: http://10.1.3.180/index.php?title=File:Roc-calendar.jpg License: unknown Contributors: Hawk
File:jp-calendar.jpg Source: http://10.1.3.180/index.php?title=File:Jp-calendar.jpg License: unknown Contributors: Hawk
File:buddhist-calendar.jpg Source: http://10.1.3.180/index.php?title=File:Buddhist-calendar.jpg License: unknown Contributors: Hawk
Image:ZKComRef_Datebox_Week_Of_Year.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Datebox_Week_Of_Year.PNG License: unknown Contributors: Jumperchen
Image:ZKComRef_Datebox_Link_Of_Today.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Datebox_Link_Of_Today.PNG License: unknown Contributors: Chunfuchang
Image:datebox_mold_default.png Source: http://10.1.3.180/index.php?title=File:Datebox_mold_default.png License: unknown Contributors: Peterkuo
Image:datebox_mold_rounded.png Source: http://10.1.3.180/index.php?title=File:Datebox_mold_rounded.png License: unknown Contributors: Peterkuo
Image:ZKComRef_Decimalbox_Examples.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Decimalbox_Examples.PNG License: unknown Contributors: Samchuang
Image:ZKComRef_Doublebox_Examples.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Doublebox_Examples.PNG License: unknown Contributors: Samchuang
Image:ZKComRef_Doublespinner.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Doublespinner.png License: unknown Contributors: Jumperchen, Tmillsclare
File:ZK_Component_Reference-Input-Spinner-inplace.jpg Source: http://10.1.3.180/index.php?title=File:ZK_Component_Reference-Input-Spinner-inplace.jpg License: unknown Contributors: Tmillsclare
Image:spinner_mold_default.png Source: http://10.1.3.180/index.php?title=File:Spinner_mold_default.png License: unknown Contributors: Peterkuo
Image:spinner_mold_rounded.png Source: http://10.1.3.180/index.php?title=File:Spinner_mold_rounded.png License: unknown Contributors: Peterkuo
Image:ZKComRef_Inbox.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Inbox.png License: unknown Contributors: Jumperchen, SimonPai
Image:ZKComRef_Inbox2.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Inbox2.png License: unknown Contributors: Jumperchen, SimonPai
Image:ZKComRef_longbox.jpg Source: http://10.1.3.180/index.php?title=File:ZKComRef_longbox.jpg License: unknown Contributors: Jumperchen, SimonPai
Image:ZKComRef_Multislider.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Multislider.png License: unknown Contributors: Jameschu
Image:ZKComRef_radio.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_radio.png License: unknown Contributors: Hawk, Jumperchen, SimonPai
File:ZKComRef_Radiogroup_Grid.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Radiogroup_Grid.png License: unknown Contributors: Jimmyshiau
Image:Radiogroup.png Source: http://10.1.3.180/index.php?title=File:Radiogroup.png License: unknown Contributors: Tonyq
Image:ZKComRef_Rangeslider.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Rangeslider.png License: unknown Contributors: Jameschu
Image:ZKComRef_RangesliderNoMarks.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_RangesliderNoMarks.png License: unknown Contributors: Jameschu
File:Searchbox-example.png Source: http://10.1.3.180/index.php?title=File:Searchbox-example.png License: unknown Contributors: Rudyhuang
File:Signature.png Source: http://10.1.3.180/index.php?title=File:Signature.png License: unknown Contributors: Charlesqiu
File:Signature_toolbar.png Source: http://10.1.3.180/index.php?title=File:Signature_toolbar.png License: unknown Contributors: Charlesqiu
File:Signature_toolbar2.png Source: http://10.1.3.180/index.php?title=File:Signature_toolbar2.png License: unknown Contributors: Charlesqiu
Image:ZKComRef_Slider.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Slider.png License: unknown Contributors: Jumperchen
File:min_slider.png Source: http://10.1.3.180/index.php?title=File:Min_slider.png License: unknown Contributors: Raymondchao
File:dec_slider.png Source: http://10.1.3.180/index.php?title=File:Dec_slider.png License: unknown Contributors: Raymondchao
File:dec_slider_no_step.png Source: http://10.1.3.180/index.php?title=File:Dec_slider_no_step.png License: unknown Contributors: Raymondchao
File:dec_slider_step.png Source: http://10.1.3.180/index.php?title=File:Dec_slider_step.png License: unknown Contributors: Raymondchao
File:knob360.png Source: http://10.1.3.180/index.php?title=File:Knob360.png License: unknown Contributors: Klyvechen, Wenninghsu
File:knob270.png Source: http://10.1.3.180/index.php?title=File:Knob270.png License: unknown Contributors: Klyvechen, Wenninghsu
Image:slider_mold_default.png Source: http://10.1.3.180/index.php?title=File:Slider_mold_default.png License: unknown Contributors: Peterkuo
Image:slider_mold_sphere.png Source: http://10.1.3.180/index.php?title=File:Slider_mold_sphere.png License: unknown Contributors: Peterkuo
Image:slider_mold_scale.png Source: http://10.1.3.180/index.php?title=File:Slider_mold_scale.png License: unknown Contributors: Peterkuo
Image:knob270.png Source: http://10.1.3.180/index.php?title=File:Knob270.png License: unknown Contributors: Klyvechen, Wenninghsu
Image:ZKComRef_Sliderbuttons.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Sliderbuttons.png License: unknown Contributors: Jameschu
Image:ZKComRef_Spinner.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Spinner.png License: unknown Contributors: Jumperchen
Image:Zkcompref_teditor.png Source: http://10.1.3.180/index.php?title=File:Zkcompref_teditor.png License: unknown Contributors: Chunfuchang
Image:ZKComPref_Textbox.png Source: http://10.1.3.180/index.php?title=File:ZKComPref_Textbox.png License: unknown Contributors: Jumperchen, Tmillsclare
Image:ZKComRef_Textbox_Tabbable.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Textbox_Tabbable.png License: unknown Contributors: Jumperchen
Image:ZKCompRef_Timebox.png Source: http://10.1.3.180/index.php?title=File:ZKCompRef_Timebox.png License: unknown Contributors: Hawk, Jumperchen, Tmillsclare
Image:timebox_mold_default.png Source: http://10.1.3.180/index.php?title=File:Timebox_mold_default.png License: unknown Contributors: Peterkuo
Image:timebox_mold_rounded.png Source: http://10.1.3.180/index.php?title=File:Timebox_mold_rounded.png License: unknown Contributors: Peterkuo
Image:ZKCompRef_Timepicker.png Source: http://10.1.3.180/index.php?title=File:ZKCompRef_Timepicker.png License: unknown Contributors: Jameschu
Image:ZKComRef_Absolutelayout_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Absolutelayout_Example.png License: unknown Contributors: Jumperchen
Image:ZKComRef_Anchorlayout_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Anchorlayout_Example.png License: unknown Contributors: Jumperchen
Image:ZKCompRef_Borderlayout.jpg Source: http://10.1.3.180/index.php?title=File:ZKCompRef_Borderlayout.jpg License: unknown Contributors: Jumperchen, Tmillsclare
Image:DrBorderlayout.png Source: http://10.1.3.180/index.php?title=File:DrBorderlayout.png License: unknown Contributors: Alicelin, Jimmyshiau, Tomyeh
Image:DrBorderlayout_flex.png Source: http://10.1.3.180/index.php?title=File:DrBorderlayout_flex.png License: unknown Contributors: Alicelin, Jimmyshiau
Image:DrBorderlayout_Center_scrolling.png Source: http://10.1.3.180/index.php?title=File:DrBorderlayout_Center_scrolling.png License: unknown Contributors: Alicelin, Jimmyshiau
Image:DrBorderlayout_grow.png Source: http://10.1.3.180/index.php?title=File:DrBorderlayout_grow.png License: unknown Contributors: Alicelin, Jimmyshiau
Image:ZKComRef_Box_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Box_Example.png License: unknown Contributors: Jumperchen, Sphota
Image:100000000000009300000077C9A14E08.png Source: http://10.1.3.180/index.php?title=File:100000000000009300000077C9A14E08.png License: unknown Contributors: Maya001122
Image:ZKComRef_Vbox_Simple_Example_align_pack.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Vbox_Simple_Example_align_pack.PNG License: unknown Contributors: Jimmyshiau, Robertwenzel
Image:box_mold_horizontal.png Source: http://10.1.3.180/index.php?title=File:Box_mold_horizontal.png License: unknown Contributors: Peterkuo
Image:box_mold_vertical.png Source: http://10.1.3.180/index.php?title=File:Box_mold_vertical.png License: unknown Contributors: Peterkuo
Image:ZKComRef_Cardlayout_Horizontal.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Cardlayout_Horizontal.png License: unknown Contributors: MontyPan
Image:ZKComRef_Cardlayout_Vertical.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Cardlayout_Vertical.png License: unknown Contributors: MontyPan
Image:cardlayout.gif Source: http://10.1.3.180/index.php?title=File:Cardlayout.gif License: unknown Contributors: Hawk
Image:ZKComRef_Columnlayout_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Columnlayout_Example.png License: unknown Contributors: Hawk, Jumperchen, Tomyeh
Image:ZKComRef_Columnlayout_Example_ZK6.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Columnlayout_Example_ZK6.png License: unknown Contributors: Benbai
File:ZKCompRef_GoldenLayout.png Source: http://10.1.3.180/index.php?title=File:ZKCompRef_GoldenLayout.png License: unknown Contributors: Wenninghsu
File:ZKCompRef_GoldenLayout_areas.png Source: http://10.1.3.180/index.php?title=File:ZKCompRef_GoldenLayout_areas.png License: unknown Contributors: Wenninghsu
File:ZKCompRef_GoldenLayout_vertical.png Source: http://10.1.3.180/index.php?title=File:ZKCompRef_GoldenLayout_vertical.png License: unknown Contributors: Wenninghsu
File:ZKCompRef_GoldenLayout_horizontal.png Source: http://10.1.3.180/index.php?title=File:ZKCompRef_GoldenLayout_horizontal.png License: unknown Contributors: Wenninghsu

File:ZKCompRef_GoldenLayout_region_north.png Source: http://10.1.3.180/index.php?title=File:ZKCompRef_GoldenLayout_region_north.png License: unknown Contributors: Wenninghsu

File:ZKCompRef_GoldenLayout_region_east.png Source: http://10.1.3.180/index.php?title=File:ZKCompRef_GoldenLayout_region_east.png License: unknown Contributors: Wenninghsu

File:ZKCompRef_GoldenLayout_region_south.png Source: http://10.1.3.180/index.php?title=File:ZKCompRef_GoldenLayout_region_south.png License: unknown Contributors: Wenninghsu

File:ZKCompRef_GoldenLayout_region_west.png Source: http://10.1.3.180/index.php?title=File:ZKCompRef_GoldenLayout_region_west.png License: unknown Contributors: Wenninghsu

File:ZKCompRef_GoldenLayout_region_stack.png Source: http://10.1.3.180/index.php?title=File:ZKCompRef_GoldenLayout_region_stack.png License: unknown Contributors: Wenninghsu

Image:ZKComRef_Hbox_Simple_Examples.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Hbox_Simple_Examples.PNG License: unknown Contributors: Peterkuo

Image:ZKComRef_Hbox_Simple_Examples_align_pack.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Hbox_Simple_Examples_align_pack.PNG License: unknown Contributors: Jimmyshiau, Robertwenzel

Image:ZKComRef_Hlayout_Simple_Example.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Hlayout_Simple_Example.PNG License: unknown Contributors: Jumperchen

Image:Linelayout-1.png Source: http://10.1.3.180/index.php?title=File:Linelayout-1.png License: unknown Contributors: Leonlee

Image:Linelayout-3.png Source: http://10.1.3.180/index.php?title=File:Linelayout-3.png License: unknown Contributors: Leonlee

File:Lineitem-opposite.png Source: http://10.1.3.180/index.php?title=File:Lineitem-opposite.png License: unknown Contributors: Leonlee

Image:Organigram_example.png Source: http://10.1.3.180/index.php?title=File:Organigram_example.png License: unknown Contributors: Charlesqiu

Image:Organigram_example2.png Source: http://10.1.3.180/index.php?title=File:Organigram_example2.png License: unknown Contributors: Charlesqiu

Image:Organigram_selection.png Source: http://10.1.3.180/index.php?title=File:Organigram_selection.png License: unknown Contributors: Charlesqiu

Image:Orgchildren_example.png Source: http://10.1.3.180/index.php?title=File:Orgchildren_example.png License: unknown Contributors: Charlesqiu

Image:Orgitem_example.png Source: http://10.1.3.180/index.php?title=File:Orgitem_example.png License: unknown Contributors: Charlesqiu

Image:ZKComRef_Portallayout_Example.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Portallayout_Example.PNG License: unknown Contributors: Jimmyshiau, Sphota

File:Kanban-1.png Source: http://10.1.3.180/index.php?title=File:Kanban-1.png License: unknown Contributors: Leonlee

Image:ZKComRef_Rowlayout.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Rowlayout.PNG License: unknown Contributors: Jameschu

File:rowlayout-examples.png Source: http://10.1.3.180/index.php?title=File:Rowlayout-examples.png License: unknown Contributors: Hawk

Image:ZKComRef_Splitlayout_Examples.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Splitlayout_Examples.PNG License: unknown Contributors: Ronchuang

Image:ZKComRef_Splitter_Examples.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Splitter_Examples.PNG License: unknown Contributors: Jimmyshiau, Samchuang

File:DrSplitterOS.png Source: http://10.1.3.180/index.php?title=File:DrSplitterOS.png License: unknown Contributors: Tomyeh

Image:splitter_mold_default.png Source: http://10.1.3.180/index.php?title=File:Splitter_mold_default.png License: unknown Contributors: Peterkuo

Image:splitter_mold_os.png Source: http://10.1.3.180/index.php?title=File:Splitter_mold_os.png License: unknown Contributors: Peterkuo

Image:ZKComRef_Tablelayout_Example.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Tablelayout_Example.PNG License: unknown Contributors: SimonPai, Sphota

Image:ZKComRef_Tablelayout_Example_ZK6.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Tablelayout_Example_ZK6.PNG License: unknown Contributors: Benbai

Image:ZKComRef_Tablelayout_Example.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Tablelayout_Example.PNG License: unknown Contributors: SimonPai, Sphota

Image:ZKComRef_Vbox_Simple_Example.PNG Source: http://10.1.3.180/index.php?title=File:ZKComRef_Vbox_Simple_Example.PNG License: unknown Contributors: Tmillsclare

Image:ZKComRef_Audio_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Audio_Example.png License: unknown Contributors: Raymondchao, SimonPai, Sphota

Image:qrcode.png Source: http://10.1.3.180/index.php?title=File:Qrcode.png License: unknown Contributors: Klyvechen

Image:code128.png Source: http://10.1.3.180/index.php?title=File:Code128.png License: unknown Contributors: Klyvechen

Image:BQR.png Source: http://10.1.3.180/index.php?title=File:BQR.png License: unknown Contributors: Klyvechen

File:ZKCompRef_Cropper.png Source: http://10.1.3.180/index.php?title=File:ZKCompRef_Cropper.png License: unknown Contributors: Wenninghsu

File:RequestCamera.png Source: http://10.1.3.180/index.php?title=File:RequestCamera.png License: unknown Contributors: Charlesqiu

File:CameraDOM.png Source: http://10.1.3.180/index.php?title=File:CameraDOM.png License: unknown Contributors: Charlesqiu

File:Pause.png Source: http://10.1.3.180/index.php?title=File:Pause.png License: unknown Contributors: Charlesqiu

Image:ZKComRef_Flash_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Flash_Example.png License: unknown Contributors: SimonPai, Sphota

File:ZK-pdfviewer-example.png Source: http://10.1.3.180/index.php?title=File:ZK-pdfviewer-example.png License: unknown Contributors: Rudyhuang

Image:Player-Sample.png Source: http://10.1.3.180/index.php?title=File:Player-Sample.png License: unknown Contributors: Charlesqiu

Image:Player-turnOffLight.png Source: http://10.1.3.180/index.php?title=File:Player-turnOffLight.png License: unknown Contributors: Charlesqiu

file:ClipToFit(false).png Source: http://10.1.3.180/index.php?title=File:ClipToFit(false).png License: unknown Contributors: Charlesqiu

file:ClipToFit(true).png Source: http://10.1.3.180/index.php?title=File:ClipToFit(true).png License: unknown Contributors: Charlesqiu

Image:ZKComRef_Auxheader.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Auxheader.png License: unknown Contributors: Jimmyshiau, Tmillsclare

Image:ZKComRef_Cell_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Cell_Example.png License: unknown Contributors: Jimmyshiau

Image:ZKComRef_Cell_Example_Hbox.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Cell_Example_Hbox.png License: unknown Contributors: Jimmyshiau

Image:ZK5ComRef_Cell_DOM_Comparison.png Source: http://10.1.3.180/index.php?title=File:ZK5ComRef_Cell_DOM_Comparison.png License: unknown Contributors: SimonPai

File:Coachmark-2.png Source: http://10.1.3.180/index.php?title=File:Coachmark-2.png License: unknown Contributors: Leonlee

File:Coachmark-1.png Source: http://10.1.3.180/index.php?title=File:Coachmark-1.png License: unknown Contributors: Leonlee

Image:ZKComRef_Frozen_Example.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_Frozen_Example.png License: unknown Contributors: Robertwenzel, SimonPai

File:hide-columns.png Source: http://10.1.3.180/index.php?title=File:Hide-columns.png License: unknown Contributors: Hawk

Image:paging_mold_default.png Source: http://10.1.3.180/index.php?title=File:Paging_mold_default.png License: unknown Contributors: Peterkuo

Image:paging_mold_os.png Source: http://10.1.3.180/index.php?title=File:Paging_mold_os.png License: unknown Contributors: Peterkuo

File:Stepbar-example.jpg Source: http://10.1.3.180/index.php?title=File:Stepbar-example.jpg License: unknown Contributors: Hawk

File:Stepbar-linear.gif Source: http://10.1.3.180/index.php?title=File:Stepbar-linear.gif License: unknown Contributors: Rudyhuang

File:Stepbar-WrappedLabels.png Source: http://10.1.3.180/index.php?title=File:Stepbar-WrappedLabels.png License: unknown Contributors: Rudyhuang

File:Stepbar-example.png Source: http://10.1.3.180/index.php?title=File:Stepbar-example.png License: unknown Contributors: Rudyhuang

File:Step-complete-default.png Source: http://10.1.3.180/index.php?title=File:Step-complete-default.png License: unknown Contributors: Rudyhuang

File:Step-error-default.png Source: http://10.1.3.180/index.php?title=File:Step-error-default.png License: unknown Contributors: Rudyhuang

File:Step-iconsclass.png Source: http://10.1.3.180/index.php?title=File:Step-iconsclass.png License: unknown Contributors: Hawk, Rudyhuang

Image:ZKComRef_SelectEvent_GetKeys.png Source: http://10.1.3.180/index.php?title=File:ZKComRef_SelectEvent_GetKeys.png License: unknown Contributors: Jumperchen

File:chatroom.png Source: http://10.1.3.180/index.php?title=File:Chatroom.png License: unknown Contributors: Hawk

image:ScrollviewExample_Update.png Source: http://10.1.3.180/index.php?title=File:ScrollviewExample_Update.png License: unknown Contributors: DanielHsu

File:SwipeEventExample_Update.png Source: http://10.1.3.180/index.php?title=File:SwipeEventExample_Update.png License: unknown Contributors: DanielHsu

File:Borderlayout_Tablet_Swipe_Example.png Source: http://10.1.3.180/index.php?title=File:Borderlayout_Tablet_Swipe_Example.png License: unknown Contributors: DanielHsu, Jumperchen

File:Borderlayout_Tablet_Scrolling_Example.png Source: http://10.1.3.180/index.php?title=File:Borderlayout_Tablet_Scrolling_Example.png License: unknown Contributors: DanielHsu, Jumperchen

File:Calendar_Tablet_Example.png *Source:* http://10.1.3.180/index.php?title=File:Calendar_Tablet_Example.png *License:* unknown *Contributors:* DanielHsu, Jumperchen

File:Cardlayout_Tablet_Example.png *Source:* http://10.1.3.180/index.php?title=File:Cardlayout_Tablet_Example.png *License:* unknown *Contributors:* DanielHsu, MontyPan

File:Colorbox_Tablet_Example.png *Source:* http://10.1.3.180/index.php?title=File:Colorbox_Tablet_Example.png *License:* unknown *Contributors:* DanielHsu, Jumperchen

File:Comobox_Tablet_Example.png *Source:* http://10.1.3.180/index.php?title=File:Comobox_Tablet_Example.png *License:* unknown *Contributors:* DanielHsu, Jumperchen

File:Datebox_Tablet_Example.png *Source:* http://10.1.3.180/index.php?title=File:Datebox_Tablet_Example.png *License:* unknown *Contributors:* DanielHsu, Jumperchen

File:Doublespinner_Tablet_Example.png *Source:* http://10.1.3.180/index.php?title=File:Doublespinner_Tablet_Example.png *License:* unknown *Contributors:* DanielHsu, Jumperchen

File:Grid_Tablet_Example.png *Source:* http://10.1.3.180/index.php?title=File:Grid_Tablet_Example.png *License:* unknown *Contributors:* DanielHsu, Jumperchen

File:Groupbox_Tablet_Scrolling_Example.png *Source:* http://10.1.3.180/index.php?title=File:Groupbox_Tablet_Scrolling_Example.png *License:* unknown *Contributors:* DanielHsu, Jumperchen

File>Listbox_Tablet_Example.png *Source:* http://10.1.3.180/index.php?title=File>Listbox_Tablet_Example.png *License:* unknown *Contributors:* DanielHsu, Jumperchen

File:Paging_Tablet_Example.png *Source:* http://10.1.3.180/index.php?title=File:Paging_Tablet_Example.png *License:* unknown *Contributors:* DanielHsu, Jumperchen

File:Panel_Tablet_Scrolling_Example.png *Source:* http://10.1.3.180/index.php?title=File:Panel_Tablet_Scrolling_Example.png *License:* unknown *Contributors:* DanielHsu, Jumperchen

File:Tabbox_Tablet_Example.png *Source:* http://10.1.3.180/index.php?title=File:Tabbox_Tablet_Example.png *License:* unknown *Contributors:* DanielHsu, Jumperchen

File:Tabbox_Tablet_Scrolling_Example.png *Source:* http://10.1.3.180/index.php?title=File:Tabbox_Tablet_Scrolling_Example.png *License:* unknown *Contributors:* DanielHsu, Jumperchen

File:Timebox_Tablet_Example.png *Source:* http://10.1.3.180/index.php?title=File:Timebox_Tablet_Example.png *License:* unknown *Contributors:* DanielHsu, Jumperchen

File:Tree_Tablet_Example.png *Source:* http://10.1.3.180/index.php?title=File:Tree_Tablet_Example.png *License:* unknown *Contributors:* DanielHsu, Jumperchen

File:Spinner_Tablet_Example.png *Source:* http://10.1.3.180/index.php?title=File:Spinner_Tablet_Example.png *License:* unknown *Contributors:* DanielHsu, Jumperchen

File:Window_Tablet_Scrolling_Example.png *Source:* http://10.1.3.180/index.php?title=File:Window_Tablet_Scrolling_Example.png *License:* unknown *Contributors:* DanielHsu, Jumperchen