potix

SIMPLY RICH

ZK™ The Developer's Reference

Version 3.6.3

June 2009

Potix Corporation

Revision 113

Table of Contents

1. Introduction

Welcome to ZK, the simplest way to make Web applications rich.

The Developer's Reference fully describes properties and methods of components. For concepts, features, refer to **the Developer's Guide**. For installation, refer to **the Quick Start Guide**.

2. The ZK User Interface Markup Language

Implicit Objects

For scripts (aka., zsccript) and EL expressions embedded in a ZUML page, there are a set of implicit objects that enable developers to access components more efficiently.

applicationScope - `java.util.Map`

A map of custom attributes associated with the Web application. It is the same as the `getAttributes` method in the `org.zkoss.zk.ui.WebApp` interface.

A Web application is a WAR, and each Web application has an independent set of custom attributes. These attributes are used mainly to communicate among different desktops and sessions.

If the client is based on HTTP, such as a Web browser, this is the same map of attributes stored in `javax.servlet.ServletContext`. In other words, you could use it communicate with other servlets, such as JSF.

arg - `java.util.Map`

The `arg` argument passed to the `createComponents` method in the `org.zkoss.zk.ui.Executions` class. It might be `null`, depending on how `createComponents` is called.

It is the same as `self.desktop.execution.arg`.

```
params.put("name", "John");
Executions.createComponents("/my.zul", null, params);
```

Then, in my.zul,

```
<window title="${arg.name}">
...
```

Notice that `arg` is available only when creating the components for the included page, say `my.zul`. On the other hand, all events, including `onCreate`, are processed later. Thus, if you want to access `arg` in the `onCreate`'s listener, use the `getArg` method of the `org.zkoss.zk.ui.event.CreateEvent` class.

componentScope - `java.util.Map`

A map of custom attributes associated with the component. It is the same as the `getAttributes` method in the `org.zkoss.zk.ui.Component` interface.

desktop - `org.zkoss.zk.ui.Desktop`

The current desktop. It is the same as `self.desktop`.

```
desktop.getPage("main");
```

desktopScope - `java.util.Map`

A map of custom attributes associated with the desktop. It is the same as the `getAttributes` method in the `org.zkoss.zk.ui.Desktop` interface.

It is mainly used to communicate among pages in the same desktop.

each - `java.lang.Object`

The current item of the collection being iterated, when ZK evaluates an iterative element. An iterative element is an element with the `forEach` attribute.

```
<listbox width="100px">
  <listitem label="${each}" forEach="${contacts}"/>
</listbox>
```

event - `org.zkoss.zk.ui.event.Event` or derived

The current event. Available for the event listener only.

```
<textbox onChanging="react(event.value)"/>
<combobox onChanging="autoComplete()"/>
<zscript>
void react(String value) {
...
}
void autoComplete() {
  String value = event.getValue();
...
}
</zscript>
```

execution – `org.zkoss.zk.ui.Execution`

The current execution.

forEachStatus – `org.zkoss.zk.ui.util.ForEachStatus`

The status of an iteration. ZK exposes the information relative to the iteration taking place when evaluating the iterative element.

```
<zk>
  <zscript>
grades = new String[] {"Best", "Better", "Good"};
```

```
  </zscript>
  <listbox width="100px">
    <listitem label="${forEachStatus.index}: ${each}" forEach="${grades}"/>
  </listbox>
</zk>
```

Note: `forEachStatus.index` is absolute with respect to the underlying collection, array or other type. For example, if `forEachBegin` is `5`, then the first value of `forEachStatus.index` with be `5`.

page - `org.zkoss.zk.ui.Page`

The current page. It is the same as `self.page`.

pageContext – `org.zkoss.web.servlet.xel.PageContext`

The current page context used to retrieve the request, response, variable resolver and so on.

pageScope - `java.util.Map`

A map of custom attributes associated with the current page. It is the same as the `getAttributes` method in the `org.zkoss.zk.ui.Page` interface.

requestScope – `java.util.Map`

A map of custom attributes associated with the current execution. It is the same as `getAttributes` method in the `org.zkoss.zk.ui.Execution` interface.

self - `org.zkoss.zk.ui.Component`

The component itself. In other words, it is the closest component, depicted as follows.

```
<listbox>
  <zscript>self.getItems();</zscript><!-- self is listbox -->
  <listitem value="ab" label="${self.value}"/><!-- self is listitem -->
  <zscript>self.getSelectedIndex();</zscript><!-- self is listbox -->
</listbox>
```

session - `org.zkoss.zk.ui.Session`

The session. It is similar to `javax.servlet.http.HttpSession`[1].

sessionScope - `java.util.Map`

A map of custom attributes associated with the session. It is the same as the `getAttributes` method in the `org.zkoss.zk.ui.Session` interface.

1 ZK session actually encapsulates the HTTP session to make ZK applications independent of HTTP.

If the client is based on HTTP, such as a Web browser, this is the same map of attributes stored in `javax.servlet.http.HttpSession`. In other words, you could use it communicate with other servlets, such as JSF.

spaceOwner - `org.zkoss.zk.ui.IdSpace`

The space owner of this component. It is the same as `self.spaceOwner`.

spaceScope - `java.util.Map`

A map of custom attributes associated with the ID space containing this component.

Processing Instructions

The XML processing instructions describe how to process the ZUML page. They will be processed first before processing XML elements.

The `component` Directive

```
<?component name="myName" macroURI="/mypath/my.zul" [inline="true|false"]
  [apply="composer"] [prop1="value1"] [prop2="value2"]... ?>
```

```
<?component name="myName" [class="myPackage.myClass"]
  [extends="nameOfExistComponent"]
  [moldName="myMoldName"] [moldURI="/myMoldURI"]
  [apply="composer"] [prop1="value1"] [prop2="value2"]... ?>
```

Defines a new component. There are two formats: by-macro and by-class.

The by-macro Format

```
<?component name="myName" macroURI="/mypath/my.zul"
  [apply="composer"] [prop1="value1"] [prop2="value2"]... ?>
```

You could define a new component based on a ZUML page. It is also called the *macro component*. In other words, once an instance of the new component is created, it creates child components based on the specified ZUML page (the `macroURI` attribute).

In addition, you could specify the initial properties (such as `prop1` in the above example), such that they are always passed to the macro component (thru the `arg` variable).

The `inline` attribute specifies whether it is an inline macro (`inlinie="true"`) or a regular macro (default).

An inline macro behaves like *inline-expansion*. ZK doesn't create a macro component if an inline macro is encountered. Rather, it inline-expands the components defined in the macro URI. In other words, it works as if you type the content of the inline macro directly to the target page.

On the other hand, ZK will create a real component (called a macro component) to represent the regular macro. That is, the macro component is created as the parent of the components that are defined in the macro.

The by-class Format

```
<?component name="myName" [class="myPackage.myClass"]
  [extends="nameOfExistComponent"]
  [moldName="myMoldName"] [moldURI="/myMoldURI"]
  [apply="composer"] [prop1="value1"] [prop2="value2"]...?>
```

In addition to defining a component by a ZUML page (aka., a macro component), You could define a new component by implementing a class that implements the `org.zkoss.zk.ui.Component` interface. Then, use the `by-class` format to declare such kind of components for a page.

To define a new component, you have to specify at least the `class` attribute, which is used by ZK to instantiate a new instance of the component.

In addition to defining a new component, you can override properties of existent components by specifying the `extends` element with the component's name to extend from (aka., extendee). In other words, if `extends` is specified, the definition of the extendee is loaded as the default value and then override only properties that are specified in this directive.

If the name of extendee and extender are the same, it means the extender will override the definition of extendee.

For example, assume you want to use `MyWindow` instead of the default window, `org.zkoss.zul.html.Window`, for all windows defined in this ZUML page. Then, you can declare it as follows.

```
<?component name="window" extends="window" class="MyWindow"?>
...
<window>
...
</window>
```

It is equivalent to the following codes.

```
<window use="MyWindow">
...
</window>
```

In addition, you could specify the properties to initialize. For example, you want to use the style class called blue for all buttons used in this page, then you could:

```
<?component name="button" extends="button" sclass="blue"?>
```

Similarly, you could use the following definition to use OK as the default label for all buttons specified in this page.

```
<?component name="button" extends="button" label="OK"?>
```

Notice that the properties won't be applied if a component is created manually (by `zscript` or by Java codes). If you still want them to be applied with the initialial properties, you could invoke the `applyProperties` method as follows.

```
<zscript>
   Button btn = new Button();
   btn.applyProperties(); //apply the initial properties
</zscript>
```

apply

[Optional][since 3.6.0]

The apply condition, which is a list of composer's class names or EL expression.

Notice that the list of composers specified here is always applied even if the component has its own apply condition. For example, both BaseComposer and FooComposer are applied in the following example,

```
<?component name="window" extends="window" apply="BaseComposer"?>
<window apply="FooComposer">
</window>
```

class

[Optional]

Used to specify the class to instantiate an instance of such kind of components. Unlike other directives, the class can be defined with `zscript`.

extends

[Optional]

Specifies the component name to extend from. The existent definition of the specified name will be loaded to initialize the new component definition. In other words, it *extends* the existent definition instead of defining a brand-new one.

macroURI

[Required if the by-macro format is used][EL is *not* allowed]

Used with the by-macro format to specify the URI of the ZUML page, which is used as the template to create components.

moldName

[Optional][Default: `default`]

Used with the by-class format to specify the mold name. If `moldName` is specified, `moldURI` must be specified, too.

moldURI

[Optional][EL is allowed]

```
moldURI="~./zul/in-my-jar.dsp"
moldURI="/WEB-INF/in-my-web.dsp"
moldURI="/jsp-or-other-servlet"
moldURI="class:com.mycompany.myrender"
```

Used with the by-class format to specify the mold URI. If `moldURI` is specified but `moldName` is not specified, the mold name is assumed as `default`.

In addition to DSP, JSP and any Servlet technologies, you can implement the `org.zkoss.zk.util.ComponentRenderer` interface, and then specify it in the `moldURI` attribute by starting with `"class:"`. With this approach, the performance is the best.

name

[Required]

The component name. If an existent component is defined with the same name, the existent component is completely invisible in this page. If the by-class format is used, the attributes of the existent components are used to initialize the new components and then override with what are defined in this processing instruction.

The `evaluator` Directive

```
<?evaluator [name="..."] [class="..."] [import="..."]?>
```

It specifies how to evaluate XEL expressions.

name

[optional][Default: *none*][Case insensitive]

The name of the implementation used to evaluate the XEL expressions. There are two ways to specify the implementation. One is the name attribute. The other is the class attribute.

For example, if you want to use MVEL[2], you can specify the name as follows.

```
<?evaluator name="mvel"?>
<window id="w" title="MVEL Demo">
```

2 MVEL is a powerful expression language. Refer to http://mvel.codehaus.org/ for more information.

```
   ${new org.zkoss.zul.Textbox().setParent(w)}
</window>
```

Here are a list of built-in implementations.

Name	Class / Description
`default`	`org.zkoss.xel.el.ELFactory` The default implementation. It is based on ZK Commons EL (`zcommons-el.jar`), which is a performance enhancement version of Apache Commons EL.
`mvel`	`org.zkoss.zkmax.xel.mvel.MVELFactory` The implementation based on MVEL, http://mvel.codehaus.org. *[available only if zkmax.jar is loaded]*
`ognl`	`org.zkoss.zkmax.xel.ognl.OGNLFactory` The implementation based on OGNL, http://www.ognl.org. *[available only if zkmax.jar is loaded]*
`commons-el`	`org.zkoss.zkmax.xel.el.ApacheELFactory` The implementation that is based on Apache Commons EL, `org.apache.commons.el.ExpressionEvaluatorImpl`. *[available only if zkmax.jar is loaded]*
`japser-el`	`org.zkoss.zkmax.xel.el21.ApacheELFactory` The implementation that is based on Apache JSP 2.1 EL, `org.apache.el.ExpressionFactoryImpl`. *[available only if zkmax.jar is loaded]*

You can provide additional implementations by use of the `class` attribute, as described in the following section. The class must implement the `org.zkoss.xel.ExpressionFactory` interface. Or, you can specify the following content in `metainfo/xel/config.xml`.

```
<config>
   <xel-config>
      <evaluator-name>Super</evaluator-name><!-- case insensitive -->
      <evaluator-class>my.SuperEvaluator</evaluator-class>
   </xel-config>
</config>
```

class

[Optional][Default: *dependind on how xel-config is specified*]

The implementation used to evaluate the XEL expressions. In addition to the name attribute, you can specify the class directly. For example, you can use MVEL by specifying class as follows.

```
<?evaluator class="org.zkoss.zkmax.xel.mvel.MVELFactory"?>
<window id="w" title="MVEL Demo">
   ${new org.zkoss.zul.Textbox().setParent(w)}
</window>
```

import

[Optiona][Default: *what are defined in taglib*]

Specifies a list of classes separated with comma to import for evaluating the expression in this page. For example, with MVEL:

```
<?evaluator class="org.zkoss.zkmax.xel.mvel.MVELFactory"
import="org.zkoss.zul.Datebox,org.zkoss.zul.Combobox"?>
<window id="w" title="MVEL Demo">
   ${new Datebox().setParent(w)}
</window>
```

Notice that not all evaluators support the import of classes. For example, all EL-based the evaluators, including the system default one, don't support it. In other words, the `import` attribute is meaningless to them. Rather, you have to use the `taglib` directive to import functions.

The `forward` Directive

```
<?forward uri="..." [if="..."] [unless="..."]?>
```

It specifies the URI to forward the request to, and the condition to decide whether to forward. If the condition is satisfied or not specified, this page won't be rendered, and the request is, instead, forwarded to the URI specified in the `uri` attribute.

Notes

- Even if the forward is effective (i.e., ZK forwards the request to the specified URI), the initiators specified in the `init` directives will be called.
- The `createComponents` method of the Execution interface ignores the `forward` directives. In other words, the `forward` directives are evaluated only if the ZUML page is loaded directly.

uri

[required][EL expressions allowed]

The URI of the page/servlet to forward to. It may be another ZUML page, a JSP page or any servlet.

If an EL expression is specified and it is evaluated to an empty string, it is considered as no forwarding at all.

if

[Optional][Default: true][EL expressions allowed]

The condition to forward to the specified URI. If both `if` and `unless` are omitted, this page won't be evaluated and ZK always forwards the request to the specified URI.

unless

[Optional][Default: false][EL expressions allowed]

The condition *not* to forward to the specified URI. If both `if` and `unless` are omitted, this page won't be evaluated and ZK always forwards the request to the specified URI.

The `function-mapper` Directive

```
<?function-mapper class="..."
  [arg0="..."] [arg1="..."] [arg2="..."] [arg3="..."]?>
```

Specifies the function mapper that will be used by the EL expressions to resolve unknown function. The specified class must implement the `org.zkoss.xel.FunctionMapper` interface.

You can specify multiple function mappers with multiple `function-mapper` directives. The later declared one has higher priority.

Notice that the `function-mapper` directives are evaluated before the `init` directives.

class

[Optional]

A class name that must implement the `org.zkoss.xel.FunctionMapper` interface. Unlike the `init` directive, the class name cannot be the class that is defined in zscript codes.

arg0, arg1...

[Optional]

You could specify any number of arguments. If not specified, the default constructor is assumed. If specified, it will look for the constructor with the signature in the following order:

1. `Foo(Map args)`
2. `Foo(Object[] args)`

3. Foo()

If the first signature is found, the arguments with the name and value are passed to the constructor as an instance of `Map`. If the second signature is found, the values of arguments are passed to the constructor in an array of objects.

Prior to ZK 3.6.2, only the second signature is checked if one or more argument is specified, and it assumes `arg0` as the first argument, `arg1` as the second, and so on.

On the hand, you, with ZK 3.6.2 or later, can use any readable name for arguments as follows.

```
<?function-mapper class="foo.Foo" whatever="anything"?>
```

The `import` Directive

```
<?import uri="..."?>
<?import uri="..." directives="..."?>
```

It imports the directives, such as component definitions (`<?component?>`) and initiators (`<?init?>`), defined in another ZUML page.

A typical use is that you put a set of component definitions in one ZUML page, and then import it in other ZUML pages, such that they share the same set of component definitions, additional to the system default.

```
<!-- special.zul: Common Definitions -->
<?init zscript="/WEB-INF/macros/special.zs"?>
<?component name="special" macroURI="/WEB-INF/macros/special.zuml" class="Special"?>
<?component name="another" macroURI="/WEB-INF/macros/another.zuml"?>
```

where the `Special` class is assumed to be defined in `/WEB-INF/macros/special.zs`.

Then, other ZUML pages can share the same set of component definitions as follows.

```
<?import uri="special.zul"?>
...
<special/><!-- you can use the component defined in special.zul -->
```

Notes

- Unlike other directives, the import directives must be at the topmost level, i.e., at the the same level as the root element.
- The imported directives in the imported page are also imported. For example, if A imports B and B imports C, then A imports both C and B component definitions. If there is a name conflict, A overrides B, while B overrides C.
- Once the directives are imported, they won't be changed until the importing page is change, no matter the imported page is changed or not.

uri

[Required]

The URI of a ZUML page which the component definitions will be imported from.

directives

[Optional]

If the `directives` attribute is omitted, only the `component` and `init` directives are imported. If you want to import particular directives, you can specify a list of the names of the directives separated by comma. For example,

```
<?import uri="/template/taglibs.zul" directives="taglib, xel-method"?>
```

The directives that can be imported include `component`, `init`, `meta`, `taglib`, `variable-resolver`, and `xel-method`. If you want to import them all, specify * to the `directives` attribute. Notice that `meta` implies both the `meta` and `link` directives.

The `init` Directive

`<?init class="..." [arg0="..."] [arg1="..."] [arg2="..."] [arg3="..."]?>`

`<?init zscript="..."?>`

There are two formats. The first format is to specify a class that is used to do the application-specific initialization. The second format is to specify a `zscript` file to do the application-specific initialization.

Since 3.6.2, you can use any (readable) name instead of arg0 and so on. For example,

```
<?init class="org.zkoss.zkplus.databind.AnnotateDataBinderInit" root="./abc"?>
```

The initialization takes place before the page is evaluated and attached to a desktop. Thus, the `getDesktop`, `getId` and `getTitle` method will return null, when initializing. To retrieve the current desktop, you could use the `org.zkoss.zk.ui.Execution` interface.

You could specify any number of the `init` directive. The specified class must implement the `org.zkoss.zk.ui.util.Initator` interface.

```
<?init class="MyInit1"?>
<?init class="MyInit2"?>
```

class

[Optional]

A class name that must implement the `org.zkoss.zk.ui.util.Initator` interface. Unlike the `init` directive, the class name cannot be the class that is defined in zscript codes.

An instance of the specified class is constructed and its `doInit` method is called in the Page Initial phase (i.e., before the page is evaluated). The `doFinally` method is called after the page has been evaluated. The `doCatch` method is called if an exception occurs during the evaluation.

Thus, you could also use it for cleanup and error handling.

zscript

[Optional]

A `script` file that will be evaluated in the Page Initial phase.

arg0, arg1...

[Optional]

You could specify any number of arguments. It will be passed to the `doInit` method if the first format is used. Since 3.6.2, you can use any name for the arguments, but, in the prior version, the first argument is `arg0`, the second is `arg1` and follows.

The `link`, `meta` and `script` Directives

```
<?link [href="uri"] [name0="value0"] [name1="value1"] [name2="value2"]?>
<?meta [name0="value0"] [name1="value1"] [name2="value2"]?>
<?script type="text/javascript" [src="uri"] [charset="encoding"]
  [content="javascript"]?>
```

These are so-called header elements in HTML. They are generated inside the HEAD element. The meta tags are generated before ZK default JavaScript and CSS files, while the other tags are generated after ZK default JavaScript and CSS files. Currently only HTML-based clients (so-called browsers) support them.

Developers can specify whatever attributes with these header directives. ZK only encodes the URI of the `href` and `src` attribute (by use of the `encodeURL` method of the `Executions` class). ZK generates all other attributes directly to the client.

Notice that these header directives are effective only for the main ZUL page. In other words, they are ignored if a page is included by another pages or Servlets. Also, they are ignored if the page is a `zhtml` file.

```
<?link rel="alternate" type="application/rss+xml" title="RSS feed"
 href="/rssfeed.php"?>
<?link rel="shortcut icon" type="image/x-icon" href="/favicon.ico"?>
<?link  rel="stylesheet" type="text/css" href="~./zul/css/ext.css.dsp"?>
<?script type="text/javascript" src="/js/foo.js"?>
<?script type="text/javascript" content="var foo = true;
if (zk.ie) foo = false;"?>
<window title="My App">
```

```
   My content
</window>
```

The `page` Directive

```
<?page [id="..."] [title="..."] [style="..."] [cacheable="false|true"]
  [language="xul/html"] [zscriptLanguage="Java"]
  [contentType="text/html;charset=UTF-8"]
  [docType="tag PUBLIC &quot;doctype name&quot; &quot;doctype UI&quot;"]
  [xml="version=&quot;1.0&quot; encoding=&quot;UTF-8&quot;"]
  [complete="true|false"]?>
```

It specifies how a page shall be handled. The `id` and `title` arguments are the two most important ones.

cacheable

[Optional][Default: false if Ajax devices, true if XML and MIL devices]

It specifies whether the client can cache the output.

Note: Browsers, such as Firefox and IE, don't handle the cache of DHTML correctly, so it is not safe to specify `cacheable` with true for Ajax devices.

complete

[Optional][Default: false]

It specifies that this page is a complete page. By complete we mean the page has everything that the client expects. For example, if the client is a HTML browser, then a complete page will generate all necessary HTML tags, such as `<html>`, `<head>` and `<body>`.

By default (false), a ZK page is assumed to be complete *if and only if* it is *not* included by other page. In other words, if a ZK page is included by other page, ZK will generate `<div>` (if the client is a HTML browser) to enclose the output of the (incomplete) ZK page.

If you have a ZK page that contains a complete HTML page and is included by other page, you have to specify `true` for this option. For example, the includer is nothing but including another page:

```
//includer.jsp
<jsp:include page="includee.zhtml"/>
```

And, the included page contains a complete HTML page:

```
<?page complete="true"?>
<html xmlns="http://www.zkoss.org/2005/zk/native">
   <head>
      <title>My Title</tile>
   </head>
   <body>
      My Content
   </body>
</html>
```

contentType

[Optional][Default: *depends on the device*]

It specifies the content type. If not specified, it depends on the device. For Ajax devices, it is `text/html;charset=UTF-8`. For XML and MIL devices, it is `text/xml;charset=UTF-8`.

Application developers rarely need to change it, unless for XML devices.

docType

[Optional][Default: *depends on the device*]

It specifies the `DOCTYPE` (the root tag and DTD) that will be generated to the output directly. This directive is mainly used by XML devices. You rarely need to specify the DOCTYPE directive for Ajax or MIL devices. For example,

```
<?DOCTYPE value="svg PUBLIC &quot;-//W3C//DTD SVG 1.1//EN&quot;
&quot;http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd&quot;"?>
```

will cause the output to be generated with the following snippet

```
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/
DTD/svg11.dtd"">
```

Notice that the `<!DOCTYPE...>` specified in a ZUML page is processed by ZK Loader. It is not part of the output.

id

[Optional][Default: *generated automatically*][EL allowed]

Specifies the identifier of the page, such that we can retrieve it back. If an alphabetical identifier is assigned, it will be available to scripts (aka., `zscript`) and EL expressions embedded in ZUML pages.

```
<?page id="${param.id}"?>
```

language

[Optional][Default: *depending on the extension*][Allowed values: `xul/html` | `xhtml`]

Specifies the markup language for this page. The markup language determines the default component set. Currently, it supports `xul/html` and `xhtml`.

Note: You can place the page directive in any location of a XML document, but the `language` attribute is meaningful only if the directive is located at the topmost level.

style

[Optional][Default: `width:100%`][EL allowed]

Specifies the CSS style used to render the page. If not specified, it depends on the mold. The default mold uses `width:100%` as the default value.

```
<?page style="width:100%;height:100%"?>
```

title

[Optional][Default: *none*][EL allowed]

Specifies the page title that will be shown as the title of the browser.

It can be changed dynamically by calling the `setTitle` method in the `org.zkoss.zk.ui.Page` interface.

```
<?page title="${param.title}"?>
```

xml

[Optional][Default: *none*]

Specifies the `xml` processing instruction (i.e., `<?xml?>`) that will be generated to the output. Currently only XML devices support this option.

For example,

```
<?page xml="version=&quot;1.0&quot; encoding=&quot;UTF-8&quot;"?>
```

will generate the following as the first line of the output

```
<?xml version="1.0" encoding="UTF-8"?>
```

zscriptLanguage

[Optional][Default: `Java`][Allowed values: `Java` | `JavaScript` | `Ruby` | `Groovy`]

Specifies the default scripting language, which is assumed if an `zscript` element doesn't specify any scripting language explicitly.

```
<?page zscriptLanguage="JavaScript"?>

<zscript>
   var m = round(box.value); //JavaScript is assumed.
</zscript>
```

If this option is omitted, Java is assumed. Currently ZK supports four different languages: Java, JavaScript, Ruby and Groovy. This option is case insensitive.

Note: Deployers can extend the number of supported scripting languages. Refer to the **How to Support More Scripting Language** section in **the Developer's Guide**.

The `root-attributes` Directive

`<?root-attributes any-name1="any-value2" any-name2="any-value2"?>`

It specifies the additional attributes for the root element of the generated output, which depends on the device types.

Currently, only Ajax devices support this feature and the root element is the `html` tag. In other words, the attributes specified in the `root-attribute` directives will become the attributes of the `html` element of the generated output. For example,

```
<?root-attributes xmlns:v="urn:schemas-microsoft-com:vml"?>
```

will cause the HTML output to be generated with the following snippet

```
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:v="urn:schemas-microsoft-com:vml">
```

Note: `xmlns="http://www.w3.org/1999/xhtml"` is always generated.

Note: If the value is specified with an EL expression and it is evaluated to null, the corresponding attribute won't be generated.

`any-name="any-value"`

Any numbers of names and values are allowed. The value could contain EL expressions.

The `taglib` Directive

<code><?tablib uri="*myURI*" prefix="*my*"?></code>

This directive is used to load a `taglib` file, which defines a set of EL functions. The format of a `taglib` file is the same as that of JSP `taglib` files.

In the following example, we loads functions defined in the built-in TLD files identified as `http://www.zkoss.org/dsp/web/core` and then use one of these function called `l`.

```
<?taglib uri="http://www.zkoss.org/dsp/web/core" prefix="c"?>
<window title="${c:l('my.title')}">
...
</window>
```

Tip: ZK searches all TLD files defined in the `/metainfo/tld/config.xml` file from the classpath. If you want ZK to load your custom TLD files, add them to class path and then specify the following content in the `/metainfo/tld/config.xml` file.

```
	<taglib>
		<taglib-uri>http://your-domain.com/your-path</taglib-uri>
```

```
            <taglib-location>/the/path/of/your/tld/file</taglib-location>
      </taglib>
```

If you to load a TLD file from your Web application, you can specify the path as follows.

```
<?taglib uri="/WEB-INF/tld/my.tld" prefix="my"?>
```

uri

[Required][EL is *not* allowed]

A URL of the `taglib` file. Unlike other URL and URI, it doesn't interpret ~ or * specially. And, the page and the `taglib` files it references must be in the same Web application.

prefix

[Required]

A prefix used to identify functions defined in this `taglib` file. The prefix could be any non-empty string.

The `variable-resolver` Directive

```
<?variable-resolver class="..."
  [arg0="..."] [arg1="..."] [arg2="..."] [arg3="..."]?>
```

Specifies the variable resolver that will be used by the `zscript` interpreter to resolve unknown variables. The specified class must implement the `org.zkoss.xel.VariableResolver` interface.

You can specify multiple variable resolvers with multiple `variable-resolver` directives. The later declared one has higher priority.

Notice that the `variable-resolver` directives are evaluated before the `init` directives, so the `zscript` codes referenced by the `init` directives are affected by the variable resolver.

The following is an example when using ZK with the Spring framework. It resolves Java Beans declared in the Spring framework, such that you access them directly.

```
<?variable-resolver class="org.zkoss.zkplus.spring.DelegatingVariableResolver"?>
```

class

[Optional]

A class name that must implement the `org.zkoss.xel.VariableResolver` interface. Unlike the `init` directive, the class name cannot be the class that is defined in zscript codes.

arg0, arg1...

[Optional]

You could specify any number of arguments. If not specified, the default constructor is assumed. If specified, it will look for the constructor with the signature in the following order:

4. `Foo(Map args)`
5. `Foo(Object[] args)`
6. `Foo()`

If the first signature is found, the arguments with the name and value are passed to the constructor as an instance of `Map`. If the second signature is found, the values of arguments are passed to the constructor in an array of objects.

Prior to ZK 3.6.2, only the second signature is checked if one or more argument is specified, and it assumes `arg0` as the first argument, `arg1` as the second, and so on.

On the hand, you, with ZK 3.6.2 or later, can use any readable name for arguments as follows.

```
<?variable-resolver class="foo.Foo" whatever="anything"?>
```

The `xel-method` Directive

```
<?xel-method prefix="..." name="..." class="..."
signature="..."?>
```

Specifies a method that shall be imported by the EL evaluator. For example,

```
<?xel-method prefix="c" name="forName"
   class="java.lang.Class"
   signature="java.lang.Class forName(java.lang.String)"?>
<textbox value="${c:forName('java.util.List')}"/>
```

prefix

[Required]

Specifies the prefix used to identify this method.

name

[Required]

Specifies the name used to identify this method. The full name is "prefix:name".

class

[Required]

Specifies the class that the method is defined in.

signature

[Required]

The signature of the method. Note: the method must be public static.

ZK Elements

ZK elements are special XML elements that are used to control ZUML pages other than creating components.

The XML Namespace

If there is name conflicts, you could specify the XML name space:

`http://www.zkoss.org/2005/zk`

```
<zk:attribute xmlns:zk="http://www.zkoss.org/2005/zk">
...
```

The `attribute` Element

`<attribute name="`*`myName`*`" [trim="true|`**`false`**`"]>`*`myValue`*`</attribute>`

It defines a XML attribute of the enclosing element. The content of the element is the attribute value, while the `name` attribute specifies the attribute name. It is useful if the value of an attribute is sophisticated, or the attribute is conditional.

```
<button label="Hi">
  <attribute name="onClick">alert("Hi")</attribute>
</button>
```

It is equivalent to

```
<button label="Hi" onClick="alert(&quot;Hi&quot;)"/>
```

Another example:

```
<button>
  <attribute name="label" if="${param.happy}">Hello World!</attribute>
</button>
```

In addition, you can specify a XML fragment as the value of the attribute. The XML fragment is so-called the native content.

```
<html>
   <attribute name="content">
      <ol>
```

```
         <li forEach="${values}">${each}</li>
      </ol>
   </attribute>
</html>
```

where `ol` and `li` are part of the native content. They are not ZK components. They will be eventually converted to a String instance and assigned to the specified attribute. If values has three elements, the above example is equivalent to the following:

```
<html>
   <attribute name="content"><![CDATA[
      <ol>
         <li>${values[0]}</li>
         <li>${values[1]}</li>
         <li>${values[2]}</li>
      </ol>
   ]]></attribute>
</html>
```

name

[Required]

Specifies the attribute name.

trim

[Optional][Default: false]

Specifies whether to omit the leading and trailing whitespaces of the attribute value.

if

[Optional][Default: `true`]

Specifies the condition to evaluate this element. This element is ignored if the value specified to this attribute is evaluated to false.

unless

[Optional][Default: `false`]

Specifies the condition *not* to evaluate this element. This element is ignored if the value specified to this attribute is evaluated to true.

The `custom-attributes` Element

```
<custom-attributes
    [scope="component|space|page|desktop|session|application]
    attr1="value1" [attr2="value2"...]/>
```

It defines a set of custom attributes of the specified scope. You could specify as many as attributes you want. These attributes can be retrieved by the `getAttribute` method of the `Component` interface with the specified scope.

```
<custom-attributes cd="${param.cd}" a.b="ab"/>
```

scope

[optional][Default: `component`]

Specifies the scope to which the custom attributes are associated. If not specified, the component enclosing this element is the default scope to use.

composite

[Optional][Default: `none`]

Specifies the format of the value. It could be `none`, `list` or `map`.

By default, the value is assigned to the attribute directly after evaluating EL expressions, if any. For example, "apple, ${more}" is evaluated to "apple, orange", if more is "orange", and assigned to the attribute.

If you want to specify a list of values, you can specify the `composite` attribute with `list` as follows.

```
<custom-attributes simple="apple, ${more}" composite="list"/>
```

Then, it is converted to a list with two elements. The first element is "apple" and the second "orange".

If you want to specify a map of values, you can specify the `composite` attribute with `map` as follows.

```
<custom-attributes simple="juice=apple, flavor=${more}" composite="map"/>
```

Then, it is converted to a map with two entries. The first entry is ("juice", "apple") and the second ("flavor", "orange").

if

[Optional][Default: `true`]

Specifies the condition to evaluate this element. This element is ignored if the value specified to this attribute is evaluated to false.

unless

[Optional][Default: `false`]

Specifies the condition *not* to evaluate this element. This element is ignored if the value specified to this attribute is evaluated to true.

The `variables` Element

`<variables [local="`**`false`**`|true] `*`var1`*`="`*`value1`*`" [`*`var2`*`="`*`value2`*`"...]/>`

It defines a set of variables for the ID space it belongs. It is equivalent to the `setVariable` method of `Component`, if it has a parent component, and `Page`, if it is declared at the page level.

You could specify as many as variables you want. These variables are stored to the namespace of the ID space it belongs. Thus, they can be accessible by the interpreters and EL expressions.

```
<variables cd="${param.cd}" less="more"/>
```

local

[optional][Default: `false`]

Specifies whether to store the variable always at the current ID space. By default, it is false. It means ZK will check the existence of any variable with the same name by looking up the current ID space, the parent ID space, and parent's parent, and so on. If found, the variable's value is replaced with the value specified here. If not, a local variable is created. If true is specified, it doesn't look up any parent ID space.

composite

[Optional][Default: `none`]

Specifies the format of the value. It could be `none`, `list` or `map`.

By default, the value is assigned to the variable directly after evaluating EL expressions, if any. For example, "apple, ${more}" is evaluated to "apple, orange", if more is "orange", and assigned to the variable.

If you want to specify a list of values, you can specify the `composite` attribute with `list` as follows.

```
<variables simple="apple, ${more}" composite="list"/>
```

Then, it is converted to a list with two elements. The first element is "apple" and the second "orange".

If you want to specify a map of values, you can specify the `composite` attribute with `map` as follows.

```
<variables simple="juice=apple, flavor=${more}" composite="map"/>
```

Then, it is converted to a map with two entries. The first entry is ("juice", "apple") and the second ("flavor", "orange").

if

[Optional][Default: `true`]

Specifies the condition to evaluate this element. This element is ignored if the value specified to this attribute is evaluated to false.

unless

[Optional][Default: `false`]

Specifies the condition *not* to evaluate this element. This element is ignored if the value specified to this attribute is evaluated to true.

The `zk` Element

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

It is a special element used to aggregate other components. Unlike a real component (say, `hbox` or `div`), it is not part of the component tree being created. In other words, it doesn't represent any component. For example,

```
<window>
   <zk>
      <textbox/>
      <textbox/>
   </zk>
</window>
```

is equivalent to

```
<window>
   <textbox/>
   <textbox/>
</window>
```

The main use is to represent multiple root elements in XML format.

```
<?page title="Multiple Root"?>
<zk>
   <window title="First">
   ...
   </window>
   <window title="Second" if="${param.secondRequired}">
   ...
   </window>
</zk>
```

The other use is to iterate over versatile components.

```
<window>
   <zk forEach="${mycols}">
      <textbox if="${each.useText}"/>
      <datebox if="${each.useDate}"/>
      <combobox if="${each.useCombo}"/>
   </zk>
</window>
```

if

[Optional][Default: `true`]

Specifies the condition to evaluate this element. This element is ignored if the value specified to this attribute is evaluated to false.

unless

[Optional][Default: `false`]

Specifies the condition *not* to evaluate this element. This element is ignored if the value specified to this attribute is evaluated to true.

forEach

[Optional][Default: *ignored*]

It specifies a collection of objects, such that the `zk` element will be evaluated repeatedly against each object in the collection. If not specified or empty, this attribute is ignored. If non-collection object is specified, it is evaluated only once as if a single-element collection is specified.

forEachBegin

[Optional][Default: 0]

It is used with the `forEach` attribute to specify the starting offset when iterating a collection of objects. If not specified, it iterates from the first element, i.e., 0 is assumed.

forEachBegin

[Optional][Default: 0]

It is used with the `forEach` attribute to specify the index (starting from 0) that the iteration shall begin at. If not specified, the iteration begins at the first element, i.e., 0 is assumed.

If `forEachBegin` is greater than or equals to the number of elements, no iteration is performed.

forEachEnd

[Optional][Default: *the last element*]

It is used with the `forEach` attribute to specify the index (starting from 0) the iteration shall ends at (inclusive). If not specified, the iterations ends at the last element.

If forEachEnd is greater than or equals to the number of elements, the iteration ends at the last element.

switch

[Optional][Default: *none*]

Provide the context for mutually exclusive evaluation. The value specified in this attribute is called the switch condition.

```
<zk swtich="${condition}"/>
```

The only allowed children are the `zk` elements.

case

[Optional][Default: *none*]

Provides an alternative within the switch evaluation.

```
<zk case="apple"/>
```

If the value is a string starting and ending with slash, such as `/a[p]*/`, it is considered as a regular expression, which is used to match the switch condition.

```
<zk case="/a[a-z]*/"/>
```

You can specify multiple cases by separating them with comma.

```
<zk case="apple, ${special}"/>
```

choose

[Optional][Default: *none*]

Provide the context for mutually exclusive evaluation.

```
<zk choose="">
```

The only allowed children are the `zk` elements.

when

[Optional][Default: *none*]

Provides an alternative within the choose evaluation.

```
<zk when="${fruit == 'apple'}">
```

It is evaluated if the condition matches.

The `zscript` Element

`<zscript [language="Java|JavaScript|Ruby|Groovy"]>`*`Scripting codes`*`</zscript>`
`<zscript src="`*`uri`*`" [language="Java|JavaScript|Ruby|Groovy"]/>`

It defines a piece of scripting codes that will be interpreted when the page is evaluated. The language of the scripting codes is, by default, Java. You can select a different language by use the `language` attribute[3].

The `zscript` element has two formats as shown above. The first format is used to embed the scripting codes directly in the page. The second format is used to reference an external file that contains the scripting codes.

```
<zscript>
alert("Hi");
</zscript>
<zscript src="/codes/my.bs"/>
```

Like other ZK elements, it is not a component but a special XML element.

src

[Optional][Default: *none*]

Specifies the URI of the file containing the scripting codes. If specified, the scripting codes will be loaded as if they are embedded directly.

Note: the file shall contain the source codes in the selected scripting language. The encoding must be UTF-8. Don't specify a class file (aka. byte codes).

Like other URL and URI, it has several characteristics as follows.

1. It is relative to the servlet context path (aka., the `getContextPath` method from the `javax.servlet.http.HttpServletRequest` interface). In other words, ZK will prefix it with the servlet context automatically.
2. It resolves "~" to other Web application (aka., different ServletContext). Notice that Web server administrator might disable Web applications from peeking other's content[4].
3. It accepts "*" for loading browser and Locale dependent style sheet.

The algorithm to resolve "*" is as follows.

- If there is one "*" is specified in an URL or URI such as `/my*.css`, then "*" will be replaced with a proper Locale depending on the preferences of user's browser.
  For example, user's preferences is `de_DE`, then ZK searches `/my_de_DE.css`, `/my_de.css`, and `/my.css` one-by-one from your Web site, until any of them is found. If none of them is found, `/my.css` is still used.

3 Furthermore, you can use the page directive to change the default scripting language other than Java.
4 Refer to the `getContext` meth from the `javax.servlet.ServletContext` interface.

- If two or more "*" are specified in an URL or URI such as "/my*/lang*.css", then the first "*" will be replaced with "ie" for Internet Explorer and "moz" for other browsers[5].
  If the last "*" will be replaced with a proper Locale as described above.
- All other "*" are ignored.

language

[Optional][Default: the page's default scripting language]
[Allowed Values: Java | JavaScript | Ruby | Groovy]

It specifies the scripting language which the scripting codes are written in.

deferred

[Optional][Default: `false`]

Specifies whether to defer the evaluation of this element until the first non-deferred `zscript` codes of the same language has to be evaluated. It is used to defer the loading of the interpreter and then speed up the loading of a ZUML page. For example, if all `zscript` elements are deferred, they are evaluated only when the first event listened by a handler implemented in `zscript` is received.

Refer to the **How to Defer the Evaluation** section in the **Developer's Guide**.

if

[Optional][Default: `true`]

Specifies the condition to evaluate this element. This element is ignored if the value specified to this attribute is evaluated to false.

unless

[Optional][Default: `false`]

Specifies the condition *not* to evaluate this element. This element is ignored if the value specified to this attribute is evaluated to true.

ZK Attributes

ZK attributes are used to control the associated element, other than initializing the data member.

5 In the future editions, we will use different codes for browsers other than IE and FF.

The `apply` Attribute

```
apply="a-class-name"
apply="class1, class2,..."
apply="${EL_returns_a_class_or_a_collection_of_classes}"
apply="${EL_returns_an_instance_or_a_collection_of_Composer_instances}"
```

It specifies a class, a collection of classes that are used to initialize the component. The class must implement the `org.zkoss.zk.util.Composer` interface. And then, you can do the initialization in the `doAfterCompose` method, since it is called after the component and all its children are instantiated.

```
<window apply="MyComposer"/>
```

In addition, you specify a `Composer` instance, or a collection of `Composer` instances by use of EL expressions.

Note: the EL expressions are, if specified, evaluated before the component is instantiated. So you cannot reference to the component. Moreover, the `self` variable references to the parent component, if any, or the current page, if it is the root component, in the EL expressions specified in this attribute.

If you want more control such as handling the exception, you can also implement the `org.zkoss.zk.util.ComposerExt` interface.

The `forEach` Attribute

```
forEach="${an-EL-expr}"
forEach="an-value, ${an-EL-expr}"
```

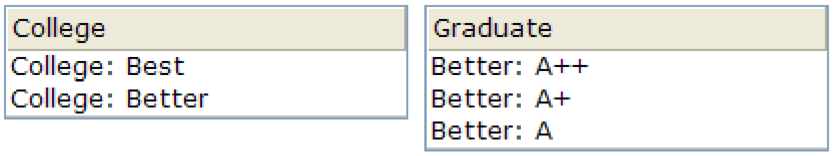
There are two formats. First, you specify a value without comma. The value is usually a collection of objects, such that the associated element will be evaluated repeatedly against each object in the collection. If not specified or empty, this attribute is ignored. If non-collection object is specified, it is evaluated only once as if a single-element collection is specified.

Second, you can specify a list of values by separating them with comma. Then, the associated element will be evaluated repeatedly against each value in the list.

For each iteration, two variables, `each` and `forEachStatus`, are assigned automatically to let developers control how to evaluate the associated element.

```
<hbox>
  <zscript>
classes = new String[] {"College", "Graduate"};
grades = new Object[] {
  new String[] {"Best", "Better"}, new String[] {"A++", "A+", "A"}
};
  </zscript>
  <listbox width="200px" forEach="${classes}">
```

```
    <listhead>
      <listheader label="${each}"/>
    </listhead>
    <listitem label="${forEachStatus.previous.each}: ${each}"
    forEach="${grades[forEachStatus.index]}"/>
  </listbox>
</hbox>
```



College
College: Best
College: Better

Graduate
Better: A++
Better: A+
Better: A

The `forEachBegin` Attribute

`forEachBegin="${`*`an-EL-expr`*`}"`

It is used with the `forEach` attribute to specify the index (starting from 0) that the iteration shall begin at. If not specified, the iteration begins at the first element, i.e., 0 is assumed.

If `forEachBegin` is greater than or equals to the number of elements, no iteration is performed.

Note: `forEachStatus.index` always starts from 0, no matter what `forEachBegin` is.

The `forEachEnd` Attribute

`forEachEnd="${`*`an-EL-expr`*`}"`

It is used with the `forEach` attribute to specify the index (starting from 0) the iteration shall ends at (inclusive). If not specified, the iterations ends at the last element.

If `forEachEnd` is greater than or equals to the number of elements, the iteration ends at the last element.

The `forward` Attribute

```
forward="orginalEvent=targetId1/targetId2.targetEvent"
forward="orginalEvent=targetId1/targetId2.targetEvent(eventData)"
forward="originalEvent=${el-target}.targetEvent(${el-eventdata})"
forward="targetEvent"
```

It is used to forward an event, that is targeting a specific component, to another component in another event name. It is called the forward condition.

The forward event is an instance of the `org.zkoss.zk.ui.event.ForwardEvent` class. you can invoke the `getOrigin` method to retrieve the original event.

The original event is optional. If it is omitted, `onClick` is assumed. Similarly, the target ID is also optional. If omitted, the space owner is assumed.

The data specified in the parenthesis is the application-specific data, which can be retrieved by calling the `getData` method.

```
<button forward="onCancel(abort)"/><!-- "abort" is passed -->
<button forward="onPrint(${inf})"/><!-- the object returned by ${inf} is passed -->
```

If you want to forward several events, you can specify these conditions in the forward attribute by separating them with the comma (,):

```
<textbox forward="onChanging=onUpdating, onChange=some.onUpdate"/>
```

The target component and the event data can be specified in EL expressions, while the event names cannot.

The `fulfill` Attribute

`fulfill="`*`event-name`*`"`
`fulfill="`*`target-id`*`.`*`event-name`*`"`
`fulfill="`*`id1/id2/id3`*`.`*`event-name`*`"`
`fulfill="${`*`el-expr`*`}.`*`event-name`*`"`

It is used to specify when to create the child components. By default (i.e., `fulfill` is not specified), the child components are created right after its parent component, at the time the ZUML page is loaded.

If you want to defer the creation of the child components, you can specify the condition with the `fulfill` attribute. The condition consists of the event name and, optionally, the target component's identifier or path. It means that the child elements won't be processed, until the event is received by, if specified, the target component. If the identifier is omitted, the same component is assumed.

If an EL expression is specified, it must return a component, an identifier or a path.

The `onFulfill` Event

After ZK applies the fulfill condition, i.e., creates all descendant components, it fires the `onFulfill` event with an instance of `org.zkoss.zk.ui.event.FulfillEvent` to notify the component for further processing if any.

For example, if you use the `wireVariables` method of the `org.zkoss.zk.ui.Components` class, you might have to call `wireVariables` again to wire the new components in the `onFulfill` event.

```
<div fulfill="b1.onClick, b2.onOpen"
 onFulfill="Components.wireVariables(self, controller)">
  ...
</div>
```

The `if` Attribute

`if="${`*`an-EL-expr`*`}"`

It specified the condition to evaluate the associated element. In other words, the associated element and all its child elements are ignored, if the condition is evaluated to false.

The `unless` Attribute

`unless="${`*`an-EL-expr`*`}"`

It specified the condition *not* to evaluate the associated element. In other words, the associated element and all its child elements are ignored, if the condition is evaluated to true.

The `use` Attribute

```
use="a-class-name"
use="${EL_returns_a_class_or_a_class_name_or_a_component}"
```

It specifies a class to create a component instead of the default one. In the following example, `MyWindow` is used instead of the default class, `org.zkoss.zul.html.Window`.

```
<window use="MyWindow"/>
```

Notice that, if the expression returns a component, the component shall not belong to any page.

3. EL Expressions

This chapter describes the details about applying EL expressions to ZUML pages.

Overview

EL expressions use the syntax `${expr}`. For example,

```
<element attr1=”${bean.property}”.../>
${map[entry]}
<another-element>${3+counter} is ${empty map}</another-element>
```

When an EL expression is used as an attribute value, it could return any kind of objects as long as the component accepts it. For example, the following expression will be evaluated to a Boolean object.

```
<window if="${some > 10}">
```

Using EL Expressions

EL expressions can be used

- In static text
- In any attribute's value including XML elements and XML processing instructions.

Variables

Implicit Objects

Literals

Operators

Functions

Using Functions

Defining Functions

Standard Implicit Objects that ZK supports

Like using EL expressions in JSP pages, you could use most of standard implicit objects in ZUML pages.

applicationScope - `java.util.Map`

A map of application-scoped attributes (String, Object).

cookie - `java.util.Map`

A map of cookies of the request. (String, Cookie).

header - `java.util.Map`

A map of headers of the request. (String, String).

headerValues - `java.util.Map`

A map of headers of the request. (String, String[]).

pageScope - `java.util.Map`

A map of page-scoped attributes (String, Object).

Notice: the page concept is a bit different from JSP because a ZK page exists across requests.

param - `java.util.Map`

A map of parameters of the request (String, String).

paramValues - `java.util.Map`

A map of parameters of the request. (String, String[]).

requestScope - `java.util.Map`

A map of request-scoped attributes (String, Object).

sessionScope - `java.util.Map`

A map of session-scoped attributes (String, Object).

ZK Implicit Objects

All variables defined in ZK scripts (aka., zscript) are available for the EL expressions. Thus, all implicit objects described in the previous chapter are also the implicit objects for the EL expressions. You are free to use `self`, `event`, `componentScope` and others. Refer to the **Implict Objects** section in the **ZK User Interface Markup Language** chapter.

4. The XUL Components

Overview

- All XUL components are packed in the `org.zkoss.zul.html` package.
- The XML name space is `http://www.zkoss.org/2005/zul`
- The extensions include `xul` and `zul`.
- The component names are case-sensitive. They are all in lower-cases.

AbstractComponent

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

Class Name

`org.zkoss.zk.ui.AbstractComponent`

Properties

Property	Description	Data Type	Default Value
id	Sets the ID.	java.lang.String	UUID (universal unique ID)
mold	Sets the mold for this component.	java.lang.String	default
visible	Sets whether this component is visible.	boolean	true

Methods

Name	Description	Return Data Type
AddAnnotation (java.lang.String annotName, java.util.Map annotAttrs)	Associates an annotation to this component.	void
addAnnotation(java.lang.String propName, java.lang.String annotName, java.util.Map annotAttrs)	Adds an annotation to the specified proeprty of this component.	void
addEventHandler(java.lang.String name, EventHandler evthd)	Adds an event handler.	void
addEventListener(java.lang.String	Adds an event listener to	boolean

Name	Description	Return Data Type
evtnm, EventListener listener)	specified event for this component.	
addSharedAnnotationMap(AnnotationMap annots)	Add a map of annotations which is shared by other components.	void
addSharedEventHandlerMap(EventHandlerMap evthds)	Adds a map of event handlers which is shared by other components.	void
appendChild(Component child)	Appends a child to the end of all children.	boolean
applyProperties()	Initializes the properties (aka. members) and custom-attributes based on what are defined in the component definition.	void
clone()	Clones the component.	java.lang.Object
containsVariable(java.lang.String name, boolean local)	Returns whether the specified variable is defined.	boolean
detach()	Detaches this component such that it won't belong to any page.	void
getAnnotatedProperties()	Returns a read-only list of the name (String) of properties that are associated at least one annotation (never null).	java.util.List
getAnnotatedPropertiesBy(java.lang.String annotName)	Returns a read-only list of the names (String) of the properties that are associated with the specified annotation (never null).	java.util.List
getAnnotation(java.lang.String annotName)	Returns the annotation associated with the component, or null if not available.	org.zkoss.zk.ui.metainfo.Annotation
getAnnotation(java.lang.String propName, java.lang.String annotName)	Returns the annotation associated with the definition of the specified property, or null if not	org.zkoss.zk.ui.metainfo.Annotation

Name	Description	Return Data Type
	available.	
getAnnotations()	Returns a read-only collection of all annotations associated with this component (never null).	java.util.Collection
getAnnotations(java.lang.String propName)	Returns a read-only collection of all annotations associated with the specified property (never null).	java.util.Collection
getAttribute(java.lang.String name)	Returns the custom attribute associated with this component, i.e., Component.COMPONENT_SCOPE.	java.lang.Object
getAttribute(java.lang.String name, int scope)	Returns the value of the specified custom attribute in the specified scope, or null if not defined.	java.lang.Object
getAttributes()	Returns all custom attributes associated with this component, i.e., Component.COMPONENT_SCOPE.	java.util.Map
getAttributes(int scope)	Returns all custom attributes of the specified scope.	java.util.Map
getChildren()	Returns a live list of children.	java.util.List
getDefinition()	Returns the component definition of this component (never null).	org.zkoss.zk.ui.metainfo. ComponentDefinition
getDesktop()	Returns the desktop of this component, or null if this component doesn't belong to any desktop.	org.zkoss.zk.ui.Desktop
getEventHandler(java.lang.String evtnm)	Returns the event handler of the specified name, or null if not found.	org.zkoss.zk.ui.metainfo.ZScript

Name	Description	Return Data Type
getExtraCtrl()	Returns the extra controls that tell ZK how to handle this component specially.	java.lang.Object
getFellow(java.lang.String compId)	Returns a component of the specified ID in the same ID space.	org.zkoss.zk.ui.Component
getFellowIfAny(java.lang.String compId)	Returns a component of the specified ID in the same ID space, or null if not found.	org.zkoss.zk.ui.Component
getListenerIterator(java.lang.String evtnm)	Returns an iterator for iterating listener for the specified event.	java.util.Iterator
getNamespace()	Returns the namespace to store variables and functions belonging to the ID space of this component.	org.zkoss.zk.scripting.Namespace
getPage()	Returns the page that this component belongs to, or null if it doesn't belong to any page.	Page
getParent()	Returns the parent component, or null if this is the root component.	org.zkoss.zk.ui.Component
getPropagatee(java.lang.String evtnm)	Default: null (no propagation at all).	org.zkoss.zk.ui.Component
getRoot()	Returns the root of the specified component.	org.zkoss.zk.ui.Component
getSpaceOwner()	Returns the owner of the ID space that this component belongs to.	org.zkoss.zk.ui.IdSpace
getUuid()	Returns UUID (universal unique ID) which is unquie in the whole session.	java.lang.String
getVariable(java.lang.String name, boolean local)	Returns the value of a variable defined in the namespace, or null if not defined or the value is null.	java.lang.Object
insertBefore(Component newChild, Component refChild)	Inserts a child before the reference child.	boolean

Name	Description	Return Data Type
`invalidate()`	Invalidates this component by setting the dirty flag such that it will be redraw the whole content later.	`void`
`isChildable()`	Default: return true (allows to have children).	`boolean`
`isListenerAvailable(java.lang.String evtnm, boolean asap)`	Returns whether the event listener is available.	`boolean`
`onChildAdded(Component child)`	Default: does nothing.	`void`
`onChildRemoved(Component child)`	Default: does nothing.	`void`
`onDrawNewChild(Component child, java.lang.StringBuffer out)`	Called when a new-created child is drawn.	`void`
`onWrongValue(WrongValueException ex)`	Notifies that an WrongValueException instance is thrown, and WrongValueException.getComponent() is this component.	`org.zkoss.zk.ui.WrongValueException`
`redraw(java.io.Writer out)`	Includes the page returned by getMoldURI() and set the self attribute to be this component.	`void`
`removeAttribute(java.lang.String name)`	Removes the custom attribute associated with this component, i.e., Component.COMPONENT_SCOPE.	`java.lang.Object`
`removeAttribute(java.lang.String name, int scope)`	Removes the specified custom attribute in the specified scope.	`java.lang.Object`
`removeChild(Component child)`	Removes a child.	`boolean`
`removeEventListener(java.lang.String evtnm, EventListener listener)`	Removes an event listener.	`boolean`
`response(java.lang.String key, AuResponse response)`	Causes a response (aka., a command) to be sent to the client.	`void`
`sessionDidActivate(Page page)`	Notification that the session, which owns this component, has just been	`void`

Name	Description	Return Data Type
	activated (aka., deserialized).	
`sessionWillPassivate(Page page)`	Notification that the session, which owns this component, is about to be passivated (aka., serialized).	`void`
`setAttribute(java.lang.String name, java.lang.Object value)`	Sets the custom attribute associated with this component, i.e., Component.COMPONENT_SCOPE.	`java.lang.Object`
`setAttribute(java.lang.String name, java.lang.Object value, int scope)`	Sets the value of the specified custom attribute in the specified scope.	`java.lang.Object`
`setComponentDefinition(ComponentDefinition compdef)`	Sets the component definition.	`void`
`setPage(Page page)`	Sets the page that this component belongs to.	`void`
`setParent(Component parent)`	Sets the parent component.	`void`
`setVariable(java.lang.String name, java.lang.Object val, boolean local)`	Sets a variable to the namespace.	`void`
`smartUpdate(java.lang.String attr, boolean value)`	A special smart-update that update a value in boolean.	`void`
`smartUpdate(java.lang.String attr, int value)`	A special smart-update that update a value in int.	`void`
`smartUpdate(java.lang.String attr, java.lang.String value)`	Smart-updates a property with the specified value.	`void`
`toString()`		`java.lang.String`
`unsetVariable(java.lang.String name, boolean local)`	Unsets a variable defined in the namespace.	`void`

FormatInputElement

A skeletal implementation for an input box with format. .

Class Name

`org.zkoss.zul.impl.FormatInputElement`

Properties

Property	Description	Data Type	Default Values
`Format`	Sets the format	`String`	`<empty string>`

Methods

Name	Description	Return Data Type
`getOuterAttrs()`		`String`

Inherited From

Inherited From
org.zkoss.zul.impl.InputElement
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

HeaderElement

A skeletal implementation for a header.

Class Name

`org.zkoss.zul.impl.HeaderElement`

Properties

Property	Description	Data Type	Default Value
`align`	Sets the horizontal alignment of this column.	`java.lang.String`	`null`
`valign`	Sets the vertical alignment of this grid.	`java.lang.String`	`null`

Methods

Name	Description	Return Data Type
`getColAttrs()`	Returns the attributes used to generate HTML TD tag for each cell of the rows contained in the parent control, e.g., Listcell.	`java.lang.String`
`getOuterAttrs()`		`java.lang.String`
`isChildable()`	Children are not allowed.	`boolean`
`setWidth(java.lang.String width)`		`void`

Inherited From

Inherited From
org.zkoss.zul.impl.LabelElement
org.zkoss.zul.impl.LabelImageElement
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

HeadersElement

A skeletal implementation for headers, the parent of a group of `HeaderElement`.

Class Name

`org.zkoss.zul.impl.HeadersElement`

Properties

Property	Description	Data Type	Default Value
`sizeable`	Sets the horizontal alignment of this column. Sets whether the width of the child column is sizable.	`boolean`	`false`

Methods

Name	Description	Return Data Type
`getOuterAttrs()`		`java.lang.String`

Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

HtmlBasedComponent

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

Class Name

`org.zkoss.zk.ui.HtmlBasedComponent`

Supported Child Components

*ALL

Supported Events

Event Name	Event Type
`onDrop`	org.zkoss.zk.ui.event.DropEvent Description: Represents an event cause by user's dragging and dropping a component.

Properties

Property	Description	Data Type	Default Value
`droppable`	Sets "true" or "false" to denote whether a component is `droppable`, or a list of identifiers of `draggable` types of objects that could be dropped to this component. **Value:**`true` | `false|the identifier of a draggable type of objects`	`java.lang.String`	`<null>`
`droppable`	Sets "true" or "false" to denote whether a component is droppable, or a list of identifiers of draggable types of objects that could be droped to this component. **Value:**`true` | `false|` `the identifier of a draggable type of objects`	`java.lang.String`	`<null>`
`height`	Sets the height.	`java.lang.String`	`<null>`
`left`	`Sets the left position.`	`java.lang.String`	`<null>`
`sclass`	Sets the CSS class.	`java.lang.String`	`<null>`
`style`	Sets the CSS style.	`java.lang.String`	`<null>`
`tooltiptext`	Sets the text as the `tooltip`.	`java.lang.String`	`<null>`
`top`	Sets the top position.	`java.lang.String`	`<null>`
`width`	Sets the width.	`java.lang.String`	`<null>`
`zIndex`	Sets the Z index.	`int`	`0`

Methods

Name	Description	Return Data Type
`addEventListener(java.lang.String, EventListener)`	Adds an event listener to specified event for this component.	`boolean`
`focus()`	Sets focus to this element.	`void`

Name	Description	Return Data Type
`getInnerAttrs()`	Returns the interior attributes for generating the inner HTML tag; never return null. Used only by component developers. Default: empty string. Refer to `getOuterAttrs` for more details.	`java.lang.String`

`getOuterAttrs()`	Returns the exterior attributes for generating the enclosing HTML tag; never return null.
	Used only by component developers.
	Default: Generates the `tooltip text, style, sclass, draggable` and `droppable` attribute if necessary. In other words, the corresponding attribute is generated if `getTooltiptext, getRealStyle, getSclass, getDraggable, getDroppable` are defined.
	
	You have to call both `getOuterAttrs` and `getInnerAttrs` to generate complete attributes.
	For simple components that all attributes are put on the outset HTML element, all you need is as follows.
	`<xx id="${self.uuid}"${self.outerAttrs}${self.innerAttrs}>`
	If you want to put attributes in a nested HTML element, you shall use the following pattern. Notice: if `getInnerAttrs` in a different tag, the tag must be named with "${self.uuid}!real".
	`<xx id="${self.uuid}"$`
		`java.lang.String`

Name	Description	Return Data Type

	`{self.outerAttrs}>`
	
	`<yy id="${self.uuid}!real"${self.innerAttrs}>...`
	
	Note: This class handles non-deferrable event listeners automatically. However, you have to invoke appendAsapAttr for each event the component handles in `getOuterAttrs` as follows.
	
	`appendAsapAttr(sb, Events.ON_OPEN);`
	
	`appendAsapAttr(sb, Events.ON_CHANGE);`
	
	Theoretically, you could put any attributes in either `getInnerAttrs` or `getOuterAttrs`. However, `zkau.js` assumes all attributes are put at the outer one. If you want something different, you have to provide your own `setAttr` (refer to how `checkbox` is implemented).	

`removeEventListener(java.lang.String, EventListener)`	Removes an event listener.	`boolean`

Inherited From

Inherited From
org.zkoss.zk.ui.AbstractComponent

InputElement

`InputElement` is a super class for components which provie user key input, such as Textbox, Intbox, etc.

Some features are implemented in this class, such as `constraint`, `disabled`, `maxlength`, `name`, `readonly`, etc.

You sholuld not deirectly use this class, please use the inherited class.

Class Name

`org.zkoss.zul.InputElement`

Supported Events

Event Name	Event Type
`onOK`	Event: `org.zkoss.zk.ui.event.KeyEvent` Denotes user has pressed the `ENTER` key.

Properties

Property	Description	Data Type	Default Value
`cols`	Sets the columns. **Note:** non-positive means the same as browser's default	`int`	0

Property	Description	Data Type	Default Value
`constraint`	Sets the constraint, must be a default constraint expression. The value,except regular expression , could be a combination String by comma **Values :** no positive | no negative | no zero | no empty | nofuture | no past | no today | a regular expression.	`String`	<empty string>
`disabled`	Sets whether it is disabled. **Values :** `true|false`	`boolean`	`false`
`maxlength`	Sets the max length. **Note :** non-postive means unlimited.	`int`	0
`name`	Sets the name of this component. Don't use this method if your application is purely based on ZK's event-driven model. The name is used only to work with "legacy" Web application that handles user's request by servlets. It works only with HTTP/HTML-based browsers. It doesn't work with other kind of clients.	`String`	`null`
`readonly`	Sets whether it is read only **Values :** `true|false`	`int`	`false`
`tabindex`	Sets the tab order of this component. **Note :** -1 means the same as browser's default	`int`	-1
`text`	Sets the value in the String format. **Note :** default value depends on implementation of sub-class.	`String`	`null`

Methods

*NONE

Inherited From

Inherited From
org.zkoss.zul.imp.XulElement

Inherited From
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

LabelElement

A HTML element with a label.

Class Name

`org.zkoss.zul.impl.LabelElement`

Properties

Property	Description	Data Type	Values
`label`	Sets the label	`String`	`Any text`

Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

LabelImageElement

A HTML element with a label and an image.

Class Name

`org.zkoss.zul.impl.LabelImageElement`

Properties

Property	Description	Data Type	Default Value
`image`	Sets the label	`String`	`null`
`imageContent`	Sets the content directly	`org.zkoss.image.Image`	`null`
`src`	Sets the image URI	`String`	`null`

Methods

Name	Description	Return Data Type
`isImageAssigned()`	Returns whether the image is available. It return true if setImage(java.lang.String) or setImageContent(org.zkoss.image.Image) is called with non-null.	`boolean`
`getImgTag()`	Returns the HTML IMG tag for the image part. Used only for component template, not for application developers. Note: the component template shall use this method to generate the HTML tag, instead of using getImage().	`String`

Inherited From

Inherited From
org.zkoss.zul.impl.LabelElement
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

LayoutRegion

This class represents a region in a layout manager.

Class Name

`org.zkoss.zkex.zul.LayoutRegion`

Supported Child Components

*NONE

Supported Events

Name	Inherited From
`onOpen`	org.zkoss.zk.ui.event.OpenEvent **Description:** When a layout is collapsed or opened by a user, the `onOpen` event is sent to the application.

Properties

Property	Description	Data Type	Default Value
flex	Sets whether to grow and shrink vertical/horizontal to fit their given space, so called flexibility.	java.lang.String	false
size	Sets the size of this region. This method is shortcut for *setHeight(String)* and *setWidth(String).* If this region is *North* or *South*, this method will invoke *setHeight(String)*. If this region is *West* or *East*, this method will invoke *setWidth(String)*. Otherwise it will throw a UnsupportedOperationException.	java.lang.String	null
splittable	Sets whether enable the split functionality.	boolean	false
collapsible	Sets whether set the initial display to collapse.	boolean	false
margins	Sets margins for the element "0,1,2,3" that direction is "top,left,right,bottom".	java.lang.String	0,0,0,0
open	Opens or collapses the splitter. Meaningful only if isCollapsible is not false.	boolean	true
autoscroll	Sets whether enable overflow scrolling.	boolean	false
border	Sets the border (either none or normal).	java.lang.String	normal
maxsize	Sets the maximum size of the resizing element.	int	2000
minsize	Sets the minimum size of the resizing element.	int	0

Methods

Name	Description	Return Data Type
`getOuterAttrs()`		
insertBefore(org.zkoss.zk.ui.Component child, org.zkoss.zk.ui.Component insertBefore)		
onChildRemoved(org.zkoss.zk.ui.Component child)		

Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

NumberInputElement

A skeletal implementation for number-type input box.

Class Name

`org.zkoss.zul.impl.NumberInputElement`

Properties

Property	Description	Data Type	Default Values
`RoundingMode`	the rounding mode.	`int`	`BigDecimal.ROUND_HALF_EVEN.`

Methods

Name	Description	Return Data Type
`setRoundingMode()`		`void`

Inherited From

Inherited From
org.zkoss.zul.impl.FormatInputElement
org.zkoss.zul.impl.InputElement
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

XulElement

The fundamental class for XUL elements.

Class Name

`org.zkoss.zul.impl.XulElement`

Properties

Property	Description	Data Type	Default Value
`action`	Sets the label	`String`	`null`
`context`	Sets the ID of Popup that should appear when the user right-clicks on the element (aka., context menu).	`String`	`null`
`popup`	Sets the ID of Popup that should appear when the user clicks on the element.	`String`	`null`
`tooltip`	Sets the ID of Popup that should be used as a tooltip window when the mouse hovers over the element for a moment.	`String`	`null`

Methods

Name	Description	Return Data Type
`getInnerAttrs()`	Generates the Client-Side-Action attributes to the interior tag. **Reason:** `onfocus` is the main use.	`String`
`getOuterAttrs()`		`String`

Inherited From

Inherited From
org.zkoss.zk.ui.HtmlBasedComponent_

Inherited From
org.zkoss.zk.ui.AbstractComponent

Components

Audio

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.

Depending on the browser and the `audio` plugin, developers might be able to control the play of an audio by the `play`, `stop` and `pause` methods. Currently, Internet Explorer with Media Player is capable of such controls.

```
<audio id="audio" height="20"/>
```

Class Name

`org.zkoss.zul.Audio`

Supported Child Components

*NONE

Supported Events

*NONE

Properties

Property	Description	Data Type	Default Value
align	Sets the alignment: one of top, **Value:** texttop, middle, absmiddle, bottom, absbottom, baseline, left, right and center.	java.lang.String	<null>
autostart	Sets whether to auto start playing the audio.	boolean	false
border	Sets the width of the border.	java.lang.String	<null>
content	Sets the content directly.	org.zkoss.sound.Audio	<null>
src	Sets the src.	java.lang.String	<null>

Methods

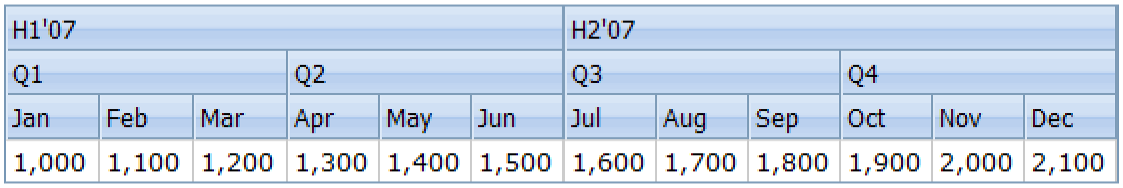
Name	Description	Return Data Type
isChildable()	Determines whether it accepts child components **Value:** false **Note:** No child is allowed.	boolean
pause()	Pauses the audio at the client.	void
play()	Plays the audio at the client.	void
stop()	Stops the audio at the client.	void

Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Auxhead

Used to define a collection of auxiliary headers (`Auxheader`).



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>
```

Class Name

`org.zkoss.zul.Auxhead`

Supported Child Components

Auxheader

Supported Events

*NONE

Properties

*NONE

Methods

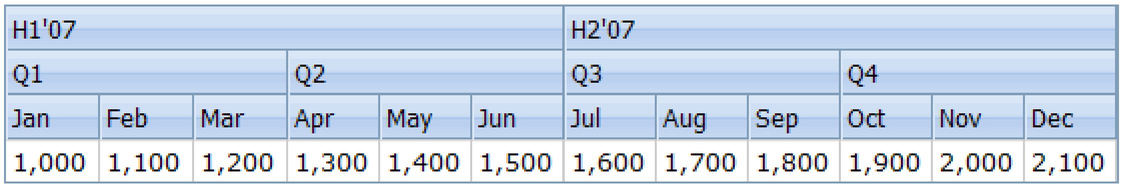
Name	Description	Return Data Type
`insertBefore(org.zkoss.zk.ui.Component child, org.zkoss.zk.ui.Component insertBefore)`		`boolean`

Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Auxheader

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>
```

Class Name

`org.zkoss.zul.Auxheader`

Supported Child Components

*NONE

Supported Events

*NONE

Properties

Property	Description	Data Type
`colspan`	Sets the number of columns to span this header.	`int`
`rowspan`	Sets the number of rows to span this header.	`int`

Methods

Name	Description	Return Data Type
getOuterAttrs()		`java.lang.String`
setParent(org.zkoss.zk.ui.Component parent)		

Inherited From

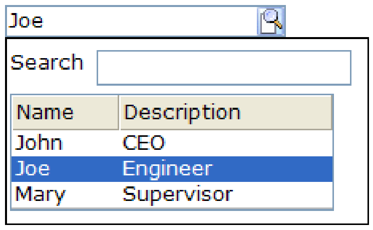
Inherited From
org.zkoss.zul.impl.HeaderElement
org.zkoss.zul.impl.LabelImageElement
org.zkoss.zul.impl.LabelElement
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Bandbox

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 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.

```
<bandbox  id="bd">
   <bandpopup>
   <vbox>
   <hbox>Search <textbox/></hbox>
   <listbox  width="200px"
   onSelect="bd.value=self.selectedItem.label;   bd.closeDropdown();">
      <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>
```


Joe

Search

Name	Description
John	CEO
Joe	Engineer
Mary	Supervisor

Class Name

`org.zkoss.zul.Bandbox`

Supported Child Components

Bandpopup

Supported Events

Name	Event Type
`onOpen`	**Event:** org.zkoss.zk.ui.event.OpenEvent 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.

Properties

Property	Description	Data Type	Default Value
`autodrop`	Sets whether to automatically drop the list if users is changing this text box.	`boolean`	`false`
`buttonVisible`	Sets whether the button (on the right of the textbox) is visible.	`boolean`	`true`
`image`	Sets the image URI that is displayed as the button to open Bandpopup.	`java.lang.String`	`"~./zul/img/bandbtn.gif".`

Methods

Name	Description	Return Data Type
`clone()`		`java.lang.Object`
`closeDropdown()`	Closes the popup (getDropdown()).	`void`
`getDropdown()`	Returns the dropdown window belonging to this band box.	`org.zkoss.zul.Bandpopup`
`getInnerAttrs()`	Generates the Client-Side-Action attributes to the interior tag.	`java.lang.String`
`getOuterAttrs()`	Returns west componennt	`java.lang.String`
`getRealStyleFlags()`	Returns RS_NO_WIDTH|RS_NO_HEIGHT.	

Name	Description	Return Data Type
`insertBefore(org.zkoss.zk.ui.Component newChild, org.zkoss.zk.ui.Component refChild)`		`boolean`

Inherited From

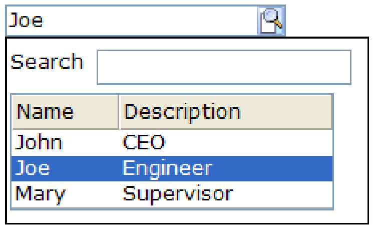
Inherited From
org.zkoss.zul.InputElement
org.zkoss.zul.imp.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Bandpopup

The popup that belongs to a `Bandbox` instance.

Developer usually listen to the `onOpen` event that is sent to `Bandbox` and then creates proper components as children of this component.

```
<bandbox id="bd">
    <bandpopup>
    <vbox>
    <hbox>Search <textbox/></hbox>
    <listbox width="200px"
    onSelect="bd.value=self.selectedItem.label;   bd.closeDropdown();">
        <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>
```


Joe

Search

Name	Description
John	CEO
Joe	Engineer
Mary	Supervisor

Class Name

`org.zkoss.zul.Bandpopup`

Supported Child Components

*NONE

Supported Events

*NONE

Properties

Property	Description	Data Type	Default Value
`visible`	Sets whether this component is visible.	`boolean`	`true`

Methods

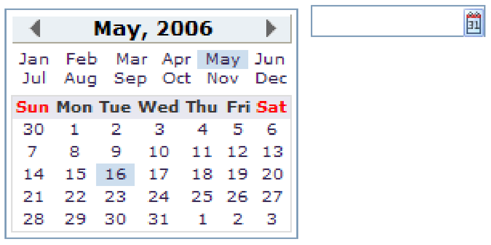
Name	Description	Return Data Type
`setParent(org.zkoss.zk.ui.Component parent)`		`void`

Inherited From

Inherited From
org.zkoss.zul.InputElement
org.zkoss.zul.imp.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Calendar

A calendar displays a 'flat' calendar and allows user to select a day from it.



```
<hbox>
   <calendar id="cal" onChange="in.value = cal.value"/>
   <datebox id="in" onChange="cal.value = in.value"/>
</hbox>
```

Class Name

`org.zkoss.zul.Calendar`

Supported Child Components

*NONE

Supported Events

*NONE

Properties

Property	Description	Data Type	Default Value
`compact`	Sets whether to use a compact layout.	`boolean`	`true`
`name`	Sets the name of this component.	`java.lang.String`	`null`
`timeZone`	Sets the time zone that this date box belongs to, or null if the default time zone is used.	`java.util.TimeZone`	`Determined by TimeZones.getCurrent()`
`value`	Assigns a value to this component.	`java.util.Date`	`today`
`visible`	Sets whether this component is visible.	`boolean`	`true`

Methods

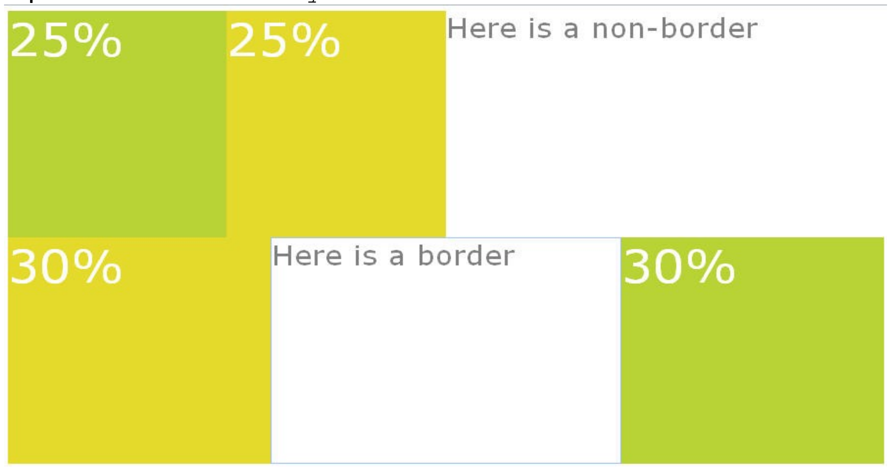
Name	Description	Return Data Type
`getOuterAttrs()`		`java.lang.String`

Inherited From

Inherited From
org.zkoss.zul.imp.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Borderlayout

The layout component is a nested component. The parent component is `borderlayout`, and its children components include `north`, `south`, `center`, `west`, and `east`. The combination of children components of `borderlayout` is free.



```
<borderlayout height="500px">
   <north size="50%" border="0">
      <borderlayout>
         <west size="25%" border="none" flex="true">
            <div style="background:#B8D335">
               <label value="25%"
                  style="color:white;font-size:50px" />
            </div>
         </west>
         <center border="none" flex="true">
            <div style="background:#E6D92C">
               <label value="25%"
                  style="color:white;font-size:50px" />
            </div>
         </center>
         <east size="50%" border="none" flex="true">
            <label value="Here is a non-border"
               style="color:gray;font-size:30px" />
         </east>
      </borderlayout>
   </north>
   <center border="0">
      <borderlayout>
         <west size="30%" flex="true" border="0">
            <div style="background:#E6D92C">
               <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 size="30%" flex="true" border="0">
            <div style="background:#B8D335">
               <label value="30%"
                  style="color:white;font-size:50px" />
            </div>
         </east>
      </borderlayout>
   </center>
</borderlayout>
```

Class Name

`org.zkoss.zkex.zul.Borderlayout`

Supported Child Components

North, East, West, South , Center

Supported Events

*None

Properties

*None

Methods

Name	Description	Return Data Type
`getCenter()`	Returns `center` component	`Center`
`getEast()`	Returns `east` component	`East`
`getNorth()`	Returns `north` component	`North`
`getSouth()`	Returns `south` component	`South`
`getWest()`	Returns `west` componennt	`West`
`insertBefore(org.zkoss.zk.ui.Component child, org.zkoss.zk.ui.Component insertBefore)`		
`resize()`	Re-size the layout component.	

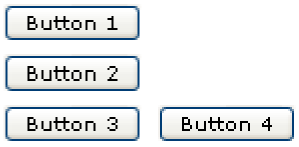
Inherited From

Inherited From
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Box

The box model of XUL is used to divide a portion of the display into a series of boxes. Components inside of 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.



```
<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>
```

Class Name

`org.zkoss.zul.Box`

Supported Child Components

*ALL

Supported Events

*NONE

Properties

Property	Description	Data Type	Default Value
`heights`	Sets the widths/heights, which is a list of numbers separated by comma to denote the width/height of each cell in a box.	`java.lang.String`	`<null>`
`orient`	Sets the orient. **Values:**`horizontal` | `vertical`	`java.lang.String`	`<null>`
`spacing`	Sets the spacing.(such as "`0`", "`5px`", "`3pt`" or "`1em`")	`java.lang.String`	`<null>`
`valign`	Sets the vertical alignment of the adjacent cells of a box. **Value:**`top|middle|bottom`	`java.lang.String`	`top`
`widths`	Sets the widths/heights, which is a list of numbers separated by comma to denote the width/height of each cell in a box.	`java.lang.String`	`<empty>`

Methods

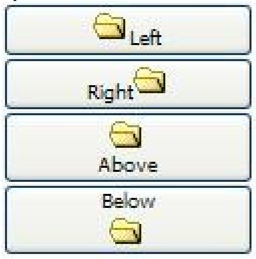
Name	Description	Return Data Type
`getChildInnerAttrs(org.zkoss.zk.ui.Component)`	Returns the inner attributes used to wrap the children (never null).	`java.lang.String`
`getChildOuterAttrs(org.zkoss.zk.ui.Component)`	Returns the outer attributes used to wrap the children (never null).	`java.lang.String`
`onDrawNewChild(org.zkoss.zk.ui.Component, java.lang.StringBuffer)`		`void`

Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Button

You could assign a `label` and an `image` to a button 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.



```
<button label="Left" image="/img/folder.gif" width="125px"/>
<button label="Right" image="/img/folder.gif" dir="reverse" width="125px"/>
<button label="Above" image="/img/folder.gif" orient="vertical" width="125px"/>
<button label="Below" image="/img/folder.gif" orient="vertical" dir="reverse"
width="125px"/>
```

Class Name

org.zkoss.zul.Button

Supported Child Components

*NONE

Supported Events

Name	Event Type
onClick	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has clicked the component.
onRightClick	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has right-clicked the component.
onDoubleClick	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has double-clicked the component.
onFocus	org.zkoss.zk.ui.event.Event

Name	Event Type
	Description: Denotes when a component gets the focus.
onBlur	org.zkoss.zk.ui.even.Event Description: Denotes when a component loses the focus.

Properties

Property	Description	Data Type	Default Value
dir	Sets the direction of button **Value:**normal | reverse	java.lang.String	normal
disabled	Sets whether it is disabled or not	boolean	false
href	Provides a hyper link	java.lang.String	<empty string>
orient	Sets the orientation of button **Value:**horizontal | vertical	java.lang.String	horizontal
target	Sets the target frame or window	java.lang.String	<null>
tabindex	Sets the tab order of this component	int	-1

Methods

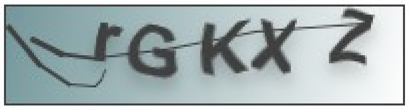
Name	Description	Return Data Type
isChildable()	Determines whether it accepts child components **Value:** false **Note:** No child is allowed.	boolean

Inherited From

Inherited From
org.zkoss.zul.impl.LabelImageElement
org.zkoss.zul.impl.LabelElement
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Captcha

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.



```
<vbox>
   <captcha id="cpa" length="5" width="200px" height="50px"/>
</vbox>
```

Class Name

org.zkoss.zul.Captcha

Supported Child Components

*NONE

Supported Events

Name	Event Type
onClick	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has clicked the component. Use getX(), getY() method get coordinates.

Properties

Property	Description	Data Type	Default Value
`bgColor`	Sets the background color of the chart. It must be `#RRGGBB` format (hexdecimal).	`String`	`"#74979B"`
`exclude`	Sets exclude characters that will not be generated. Note that only digit and character is used in generating text value. If you leave exclude `null`, the default exclude list will be applied; i.e., 0123456789IilOo (only character (no digits) are used except I, i, l, O(big O), o(small o))	`String`	`"0123456789IOilo"`
`fontColor`	Sets font color. It must in `#RRGGBB` format (hexdecimal).	`String`	`#404040`
`height`	Sets height of captcha, it colud be a px, pt or em value and it will transfer to px.	`String`	`50`
`length`	Sets length of the autogenerated text value	`int`	`5`
`noise`	Sets the flag of noise generation	`boolean`	`true`
`value`	Sets the text value to be shown as the distortion captcha	`String`	Random Value
`width`	Set width of captcha, it colud be a px, pt or em value and it will transfer to px.	`String`	`200`

Methods

Name	Description	Return Data Type
`addFont`	Adds fonts into fonts list.	`void`
`getBgRGB`	Gets the background color in int	`int`
`getCaptchaEngine`	Gets the captcha engine.	`CaptchaEngine`
`getDefaultFonts`	Gets the default font list.	`java.awt.Font[]`
`getFont`	Gets n-th Font	`java.awt.Font`
`getFontRGB`	Gets the font color in int.	`int`
`getFonts`	Gets fonts list, default provide two fonts.	`java.util.List`
`getIntHeight`	Gets the captcha int height in pixel;	`int`

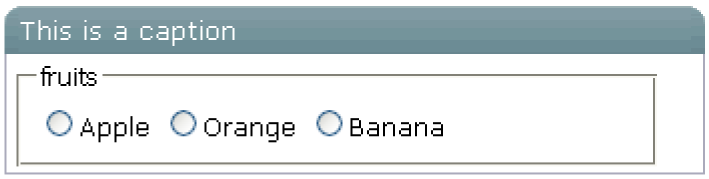
Name	Description	Return Data Type
	to be used by the derived subclass.	
`getIntWidth`	Get the captcha int width in pixel; to be used by the derived subclass.	`int`
`newCaptchaEngine`	Instantiates the default captcha engine.	`CaptchaEngine`
`randomValue`	Regenerates new captcha text value and redraw	`String`
`setEngine`	Set the captcha engine instance or by a class name string.	`void`
`smartDrawCaptcha`	Marks a draw flag to inform that this Chart needs update.	`void`

Inherited From

Inherited From
org.zkoss.zul.Image
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Caption

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



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

Class Name

`org.zkoss.zul.Caption`

Supported Child Components

*ALL

Supported Events

Name	Event Type
`onClick`	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has clicked the component.
`onRightClick`	org.zkoss.zk.ui.event.MouseEvent

Name	Event Type
	Description: Denotes user has right-clicked the component.
onDoubleClick	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has double-clicked the component.

Properties

*NONE

Methods

Name	Description	Return Data Type
getCompoundLabel()	Returns a compound label, which is the catenation of parent's title, if the parent is Window, and LabelElement.getLabel().	java.lang.String
getOuterAttrs()		java.lang.String
getSclass()	Returns the style class.	java.lang.String
invalidate()		void
isClosableVisible()	Returns whether to display the closable button.	boolean
isLegend()	Returns whether the legend mold shall be used.	boolean
setParent(org.zkoss.zk.ui.Component)		void

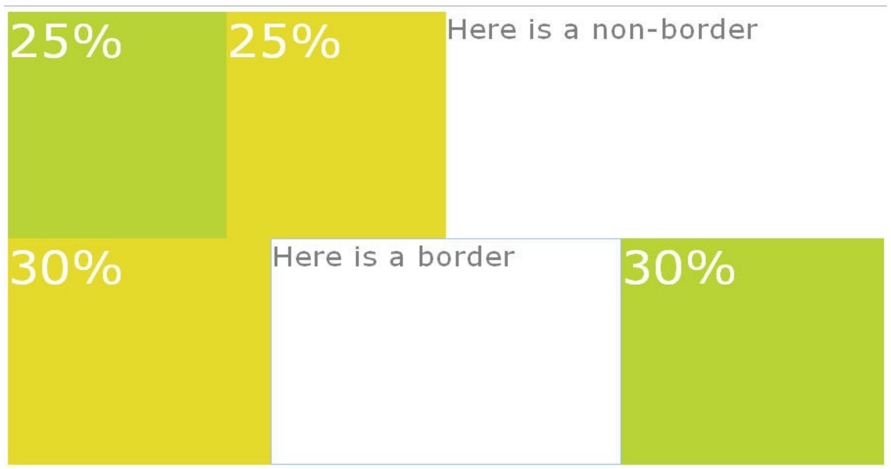
Inherited From

Inherited From
org.zkoss.zul.impl.LabelImageElement
org.zkoss.zul.impl.LabelElement
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent

Inherited From
org.zkoss.zk.ui.AbstractComponent

Center

This component is a center region. The default class of CSS is specified "layout-region-center".



```
<borderlayout height="500px">
   <north size="50%" border="0">
      <borderlayout>
         <west size="25%" border="none" flex="true">
            <div style="background:#B8D335">
               <label value="25%"
                  style="color:white;font-size:50px" />
            </div>
         </west>
         <center border="none" flex="true">
            <div style="background:#E6D92C">
               <label value="25%"
                  style="color:white;font-size:50px" />
            </div>
         </center>
         <east size="50%" border="none" flex="true">
            <label value="Here is a non-border"
               style="color:gray;font-size:30px" />
         </east>
      </borderlayout>
   </north>
   <center border="0">
      <borderlayout>
         <west size="30%" flex="true" border="0">
            <div style="background:#E6D92C">
               <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 size="30%" flex="true" border="0">
            <div style="background:#B8D335">
               <label value="30%"
                  style="color:white;font-size:50px" />
            </div>
         </east>
      </borderlayout>
   </center>
</borderlayout>
```

Class Name

`org.zkoss.zkex.zul.Center`

Supported Child Components

*NONE

Supported Events

Name	Inherited From
`OnOpen`	`org.zkoss.zk.ui.event.OpenEvent` **Description:** When a layout is collapsed or opened by a user, the onOpen event is sent to the application.

Properties

Property	Description	Data Type	Default Value
`size`	Sets the size of this region.	`java.lang.String`	`null`

Methods

Name	Description	Return Data Type
`getPosition()`	Returns `Borderlayout.NORTH.`	`java.lang.String`
setWidth(java.lang.String width)	The width can't be specified in this component because its	void

Name	Description	Return Data Type
	width is determined by other region components (West or East).	

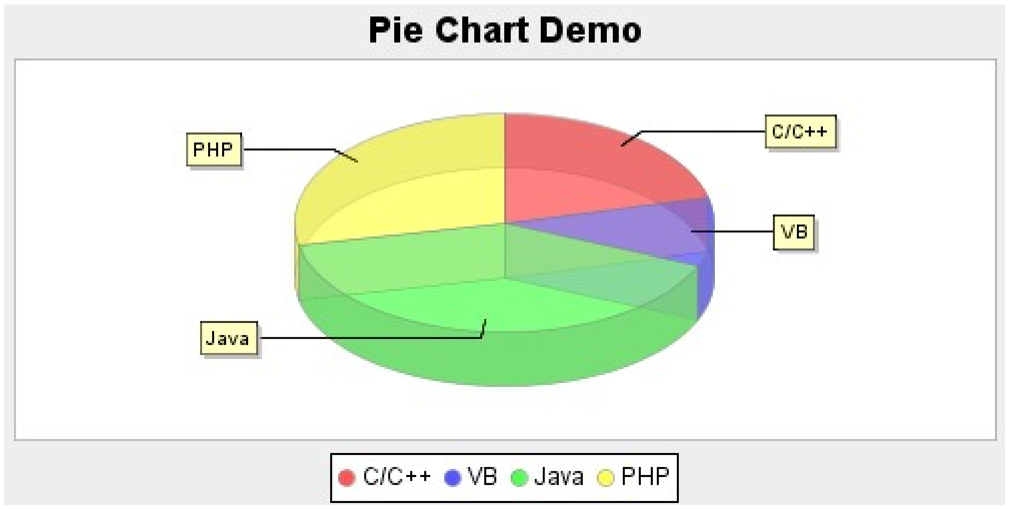
Inherited From

Inherited From
org.zkoss.zkex.zul.LayoutRegion
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Chart

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.

Type	Model	3D
`pie`	`PieModel`	o
`ring`	`PieModel`	x
`bar`	`CategoryModel`	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
`candlestick`	`HiLoModel`	x
`hilow`	`HiLoModel`	x



```
<vbox>
   <chart id="mychart" title="Pie Chart Demo" width="500" height="250" type="pie"
threeD="true" fgAlpha="128"/>
   <zscript>
      PieModel model = new MyPieModel();
      mychart.setModel(model);
   </zscript>
</vbox>
```

Class Name

`org.zkoss.zul.Chart`

Supported Child Components

*NONE

Supported Events

Name	Event Type
onClick	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has clicked the component. Use getX(), getY() method to get coordinates. Use getArea() method to get the area component which user clicks on.

Properties

Property	Description	Data Type	Default Value
bgAlpha	Sets the background alpha (transparency, 0 ~ 255).	int	255
bgColor	Sets the background color of the chart.It must be in `#RRGGBB` format (hexdecimal).	String	null
fgAlpha	Sets the foreground alpha (transparency, 0 ~ 255).	int	255
height	Sets height of chart, it must be a integer string	String	200
orient	Sets the chart orientation. Values: `vertical	horizontal`	String	vertical
paneAlpha	Sets the pane alpha (transparency, 0 ~ 255).	int	255
panelColor	Sets the pane color of the chart.It must be `#RRGGBB` format (hexdecimal).	String	null
period	Sets the period used in Time Series Chart. Values : millisecond | second | minute | | hour | day | week | month | quarter | year	String	null
showLegend	Sets the flag of showing the chart's legend	boolean	true
showTooltiptext	Sets the flag of showing the chart's tool tip text	boolean	true
threeD	Sets true to show three dimensional graph (If a type of chart got no 3d peer, this is ignored).	boolean	false
title	Sets the chart's title.	String	null
type	Set the chart's type Values: `pie	ring	bar	line	area	stacked_bar` | `stacked_`area | `waterfall	polar	scatter	time_series	polar	step_area` | `step	histogram	candlestick	hilow`	String	null
width	Sets width of chart, it must be a integer string	String	400

Property	Description	Data Type	Default Value
xAxis	Sets the label in xAxis.	String	null
yAxis	Sets the label in yAxis.	String	null

Methods

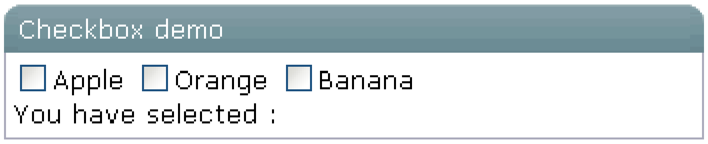
Name	Description	Return Data Type
getAreaListener	Returns the renderer to render each area, or null if the default renderer is used.	ChartAreaListener
getBgRGB	Get the background color in int array (0: red, 1: green, 2:blue).	int[]
getEngine	Returns the implemetation chart engine.	ChartEngine
getIntHeight	Get the chart int width in pixel	int
getIntWidth	Get the chart int width in pixel	int
getModel	Returns the chart model associated with this chart, or null if this chart is not associated with any chart data model	ChartModel
getPaneRGB	Get the pane color in int array (0: red, 1: green, 2:blue).	int[]
getTimeZone	Returns the time zone that this Time Series Chart belongs to, or null if the default time zone is used.	TimeZone
newChartEngine	Instantiates the default chart engine	ChartEngine
setAreaListener	Sets the renderer which is used to render each area	
setEngine	Sets the chart engine	
setModel	Sets the chart model associated with this chart	
setTimeZone	Sets the time zone that this Time Series Chart belongs to, or null if the default time zone is used	
smartDrawChart	mark a draw flag to inform that this Chart needs update	

Inherited From

Inherited From
org.zkoss.zul.Imagemap
org.zkoss.zul.Image
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Checkbox

A checkbox.



```
<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+' ' : &quot;&quot;)
               + (orange.isChecked() ? orange.label+' ' : &quot;&quot;)
               + (banana.isChecked() ? banana.label+' ' : &quot;&quot;);
         }
      </zscript>
</window>
```

Class Name

`org.zkoss.zul.Button`

Supported Child Components

*ALL

Supported Events

Name	Event Type
onRightClick	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has right-clicked the component.
onDoubleClick	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has double-clicked the component.
onFocus	org.zkoss.zk.ui.event.Event **Description:** Denotes when a component gets the focus.

Name	Event Type
onBlur	org.zkoss.zk.ui.even.Event Description: Denotes when a component loses the focus.
onCheck	org.zkoss.zk.ui.event.CheckEvent Description: Denotes when a component loses the focus.

Properties

Property	Description	Data Type	Default Value
checked	Sets whether it is checked.	boolean	false
disabled	Sets whether it is disabled.	boolean	false
name	Sets the name of this component.	java.lang.String	<null>
tabindex	Sets the tab order of this component.	int	-1

Methods

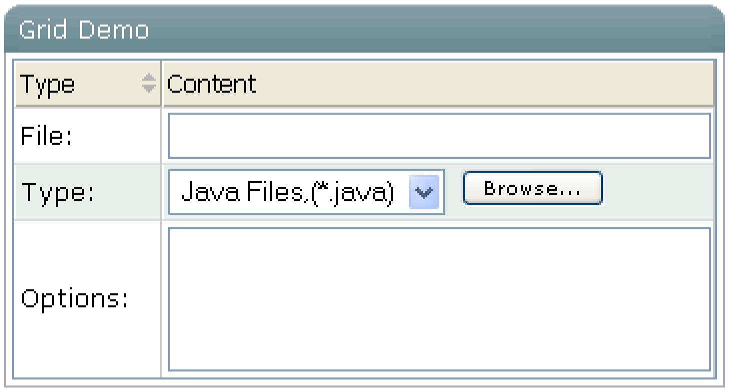
Name	Description	Return Data Type
getInnerAttrs()	Appends interior attributes for generating the HTML checkbox tag (the name, disabled and other attribute).	java.lang.String
getLabelAttrs()	Returns the attributes used by the embedded HTML LABEL tag.	java.lang.String
getOuterAttrs()	Appends exterior attributes for generating the HTML span tag (the event relevant attribute).	java.lang.String

Inherited From

Inherited From
org.zkoss.zul.impl.LabelImageElement
org.zkoss.zul.impl.LabelElement
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Column

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`.



```
<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="&#36;{asc}" sortDescending="&#36;
{dsc}"/>
         <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>
```

Class Name

`org.zkoss.zul.Column`

Supported Child Components

*ALL

Supported Events

Name	Event Type
onClick	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has clicked the component.
onRightClick	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has right-clicked the component.
onDoubleClick	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has double-clicked the component.

Properties

Property	Description	Data Type	Default Value
sortAscending	Sets the ascending sorter, or null for no sorter for the ascending order.	java.util.Comparator	<null>
sortDescending	Sets the descending sorter, or null for no sorter for the descending order.	java.util.Comparator	<null>
sortDirection	Sets the sort direction. **Value:** ascending|descending|natural	java.lang.String	natural

Methods

Name	Description	Return Data Type
getGrid()	Returns the grid that contains this column.	org.zkoss.zul.Grid
getOuterAttrs()		java.lang.String
getSclass()	Returns the style class.	java.lang.String
onSort()	It invokes sort(boolean) to sort list items and maintain getSortDirection().	void
setParent(org.zkoss.zk.ui.Component parent)		void
setSortAscending(java.lang.String)	Sets the ascending sorter with the class name, or null for no sorter for the ascending order.	void
setSortDescending(java.lang.String)	Sets the descending sorter with the class name, or null for no sorter for the descending order.	void
sort(boolean)	Sorts the rows (Row) based on getSortAscending() and getSortDescending(), if getSortDirection() doesn't matches the ascending argument.	boolean
sort(boolean, boolean)	Sorts the rows (Row) based on getSortAscending()	boolean

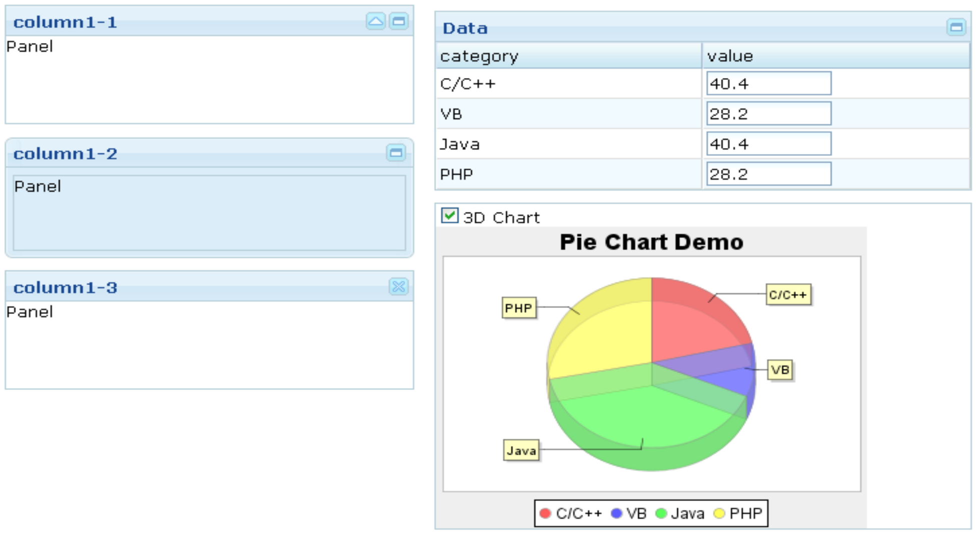
Name	Description	Return Data Type
	and `getSortDescending()`.	

Inherited From

Inherited From
org.zkoss.zul.impl.HeaderElement
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Columnchildren

Child of Columnchildren only can be `Panel`.



category	value
C/C++	40.4
VB	28.2
Java	40.4
PHP	28.2

```
<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="21.2"
```

```
                        constraint="no empty" onChange="update(0)" />
                  </row>
                  <row>
                   <label id="c1" value="VB" />
                   <decimalbox id="v1" value="10.2"
                        constraint="no empty" onChange="update(1)" />
                  </row>
                  <row>
                   <label id="c2" value="Java" />
                   <decimalbox id="v2" value="40.4"
                        constraint="no empty" onChange="update(2)" />
                  </row>
                  <row>
                   <label id="c3" value="PHP" />
                   <decimalbox id="v3" value="28.2"
                        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 (areaid!= 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, newDouble(db.getValue().doubleValue())); }
                   PieModel model = new SimplePieModel();
                   for(int j=0; j &lt; 4; ++j) { update(j); }
                   mychart.setModel(model);
                  </zscript>
            </chart>
         </panelchildren>
      </panel>
   </columnchildren>
</columnlayout>
```

Class Name

org.zkoss.zkex.zul.Columnchildren

Supported Child Components

Panel

Supported Events

*None

Properties

*None

Methods

Name	Description	Return Data Type
`getMoldSclass()`		String
`insertBefore(org.zkoss.zk.ui.Component child, org.zkoss.zk.ui.Component insertBefore)`		boolean
`setParent(org.zkoss.zk.ui.Component parent)`		void

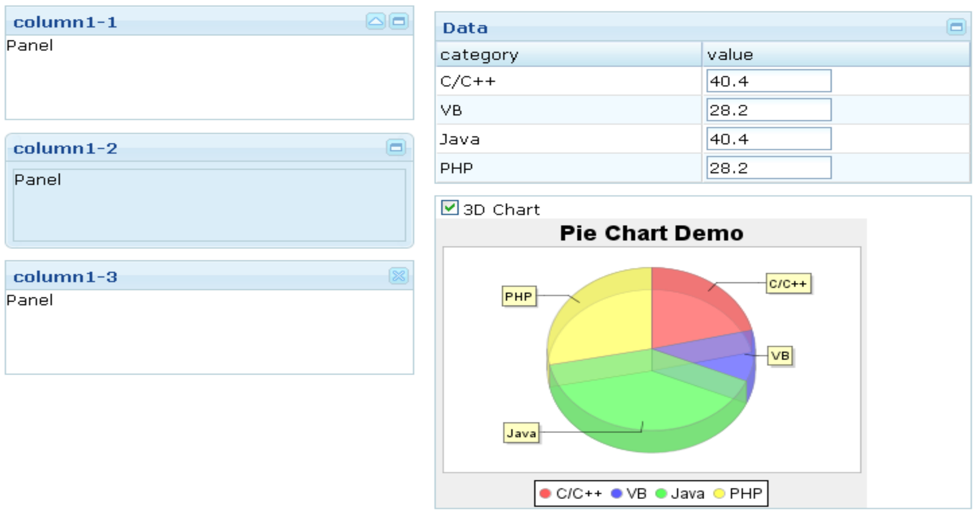
Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Columnlayout

A `columnlayout` lays out a container which can have multiple columns, and each column may contain one or more panel.

Use `Columnlayout` need assign width (either present or pixel) on every `Columnchildren`, or we cannot make sure about layout look.



category	value
C/C++	40.4
VB	28.2
Java	40.4
PHP	28.2

```
<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="21.2"
                          constraint="no empty" onChange="update(0)" />
                   </row>
                   <row>
                    <label id="c1" value="VB" />
                    <decimalbox id="v1" value="10.2"
                          constraint="no empty" onChange="update(1)" />
                   </row>
                   <row>
                    <label id="c2" value="Java" />
                    <decimalbox id="v2" value="40.4"
                          constraint="no empty" onChange="update(2)" />
                   </row>
                   <row>
                    <label id="c3" value="PHP" />
                    <decimalbox id="v3" value="28.2"
                          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 (areaid!= 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, newDouble(db.getValue().doubleValue())); }
                    PieModel model = new SimplePieModel();
                    for(int j=0; j &lt; 4; ++j) { update(j); }
                    mychart.setModel(model);
                   </zscript>
             </chart>
          </panelchildren>
       </panel>
    </columnchildren>
</columnlayout>
```

Class Name

`org.zkoss.zkex.zul.Columnlayout`

Supported Child Components

Columnchildern

Supported Events

*None

Properties

*None

Methods

Name	Description	Return Data Type
`getMoldSclass()`		`String`
`insertBefore(org.zkoss.zk.ui.Component child, org.zkoss.zk.ui.Component insertBefore)`		boolean
`setParent(org.zkoss.zk.ui.Component parent)`		`void`

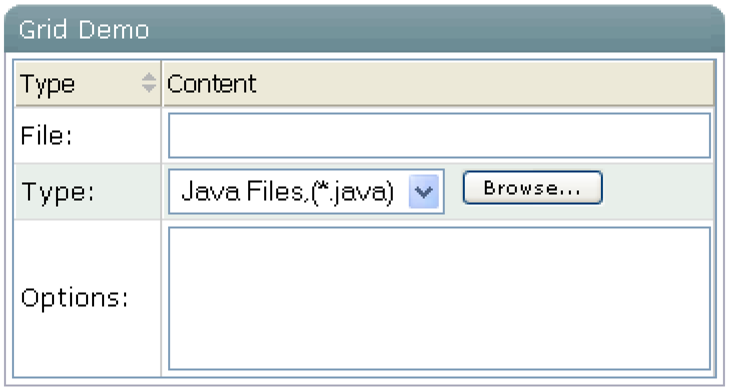
Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Columns

Defines the `columns` of a `grid`.

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



```
<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="&#36;{asc}" sortDescending="&#36;
{dsc}"/>
         <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>
```

Class Name

org.zkoss.zul.Columns

Supported Child Components

*Column

Supported Events

Name	Event Type
onColSize	org.zkoss.zul.event.ColSizeEvent **Description:** Notifies the parent of a group of headers that the widths of two of its children are changed by the user.

Properties

*NONE

Methods

Name	Description	Return Data Type
insertBefore(org.zkoss.zk.ui.Component, org.zkoss.zk.ui.Component)		boolean
removeChild(org.zkoss.zk.ui.Component)		boolean
setParent(org.zkoss.zk.ui.Component)		void

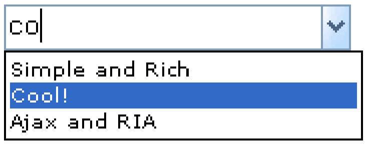
Inherited From

Inherited From
org.zkoss.zul.impl.HeadersElement
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Combobox

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.



```
<combobox>
   <comboitem label="Simple and Rich"/>
   <comboitem label="Cool!"/>
   <comboitem label="Ajax and RIA"/>
</combobox>
```

Class Name

`org.zkoss.zul.Combobox`

Supported Child Components

Comboitem

Supported Events

Name	Event Type
`onChange`	org.zkoss.zk.ui.event.InputEvent **`Description:`** Denotes the content of an input component has been modified by the user.
`onChanging`	org.zkoss.zk.ui.event.InputEvent **`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 `getValue` method in

Name	Event Type
	the `InputEvent` class to retrieve the temporary value.
`onSelection`	org.zkoss.zk.ui.event.SelectionEvent **`Description:`** 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 `getStart` and `getEnd` methods.
`onFocus`	org.zkoss.zk.ui.event.Event **`Description:`** `Denotes when a component gets the focus.`
`onBlur`	org.zkoss.zk.ui.even.Event **`Description:`** `Denotes when a component loses the focus.`
`onOpen`	org.zkoss.zk.ui.event.OpenEvent **`Description:`** 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.
`onSelect`	org.zkoss.zul.event.SelectEvent **`Description:`** `Represents an event cause by user's the list selection is changed at the client.`

Properties

Property	Description	Data Type	Default Value
`autocomplete`	Sets whether to automatically complete this text box by matching the nearest item.	`boolean`	`false`
`autodrop`	Sets whether to automatically drop the list if users is changing this text box.	`boolean`	`false`
`buttonVisible`	Sets whether the button (on the right of the textbox) is visible.	`boolean`	`true`
`image`	Sets the URI of the button image.	`java.lang.String`	`<null>`

Methods

Name	Description	Return Data Type
`isChildable()`	Determines whether it accepts child components **Value:** `true` **Note:** child is allowed.	`boolean`
`appendItem(java.lang.String)`	Appends an item.	`org.zkoss.zul.Comboitem`
`getInnerAttrs()`	Generates the Client-Side-Action attributes to the interior tag.	`java.lang.String`
`getItemAtIndex(int)`	Returns the item at the specified index.	`org.zkoss.zul.Comboitem`
`getItemCount()`	Returns the number of items.	`int`
`getItems()`	Returns a 'live' list of all org.zkoss.zul.Comboitem.	`java.util.List`
`getOuterAttrs()`		`java.lang.String`
`getSelectedItem()`	Returns the selected item, or null if no matched.	`org.zkoss.zul.Comboitem`
`insertBefore(org.zkoss.zk.ui.Component)`		`boolean`
`onChildAdded(org.zkoss.zk.ui.Component)`		`void`
`onChildRemoved(org.`		`void`

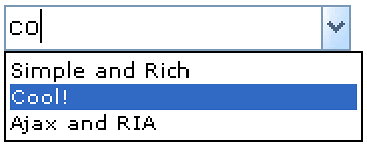
Name	Description	Return Data Type
zkoss.zk.ui.Component)		
removeItemAt(int)	Removes the child item in the list box at the given index.	org.zkoss.zul.Comboitem
setMultiline(boolean)	Sets whether it is multiline. **Note:** Combobox doesn't support multiline.	void
setRows(int)	Sets the rows. **Note:** Combobox doesn't support multiple rows.	void

Inherited From

Inherited From
org.zkoss.zul.Textbox
org.zkoss.zul.impl.InputElement
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Comboitem

An item of a combo box.



```
<combobox>
   <comboitem label="Simple and Rich"/>
   <comboitem label="Cool!"/>
   <comboitem label="Ajax and RIA"/>
</combobox>
```

Class Name

`org.zkoss.zul.Comboitem`

Supported Child Components

*NONE

Supported Events

*NONE

Properties

Property	Description	Data Type	Default Value
`value`	`Associate the value with this combo item.`	`java.lang.Object`	`<null>`
`description`	Sets the description.	java.lang.String	`<empty>`

Methods

Name	Description	Return Data Type
`isChildable()`	Determines whether it accepts child components **Value:** `false` **Note:** No child is allowed.	`boolean`
`setParent(org.zkoss.zk.ui.Component)`		`void`

Inherited From

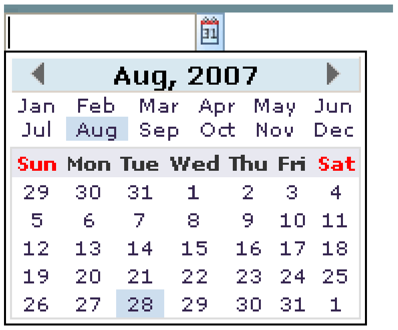
Inherited From
org.zkoss.zul.impl.LabelImageElement
org.zkoss.zul.impl.LabelElement
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Datebox

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`.



```
<datebox lenient="true" image="newButton.jpg" buttonVisible="false" />
<datebox lenient="false" compact="false" buttonVisible="true" />
```

Class Name

`org.zkoss.zul.Datebox`

Supported Child Components

*NONE

Supported Events

Name	Event Type
OnClick	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has clicked the component.
OnSelection	org.zkoss.zk.ui.event.SelectionEvent **Description:** Denotes that user is selecting a portion of the text of an input component. You can retrieve the start and end

Name	Event Type
	position of the selected text by use of the getStart and getEnd methods.
OnFocus	org.zkoss.zk.ui.event.Event **Description:** Denotes when a component gets the focus.
OnBlur	org.zkoss.zk.ui.even.Event Description: Denotes when a component loses the focus.
OnChange	org.zkoss.zk.ui.even.InputEvent **Description:** An input control notifies the application with the onChange event if its content is changed by the user.
OnChanging	org.zkoss.zk.ui.even.InputEvent **Desczription:** An input control also notifies the application with the onChanging event, when user is changing the content.

Attributes

Property	Description	Data Type	Default Value
image	the URI of the button image **Values:** url	String	<empty string>
lenient	whether or not date/time parsing is to be lenient With lenient parsing, the parser may use heuristics to interpret inputs that do not precisely match this object's format. With strict parsing, inputs must match this object's format **Values:**true|false	Boolean	true
compact	whether to use a compact layout **Values:**true|false	Boolean	false
buttonVisible	whether the button (on the right of the textbox) is visible **Values:**true|false	Boolean	true
timezone	the time zone that this date box belongs to, or null if the default time zone is used.	java.util.TimeZone	<null>
Value	the value (in Date)	java.util.Date	<empty string>

Methods

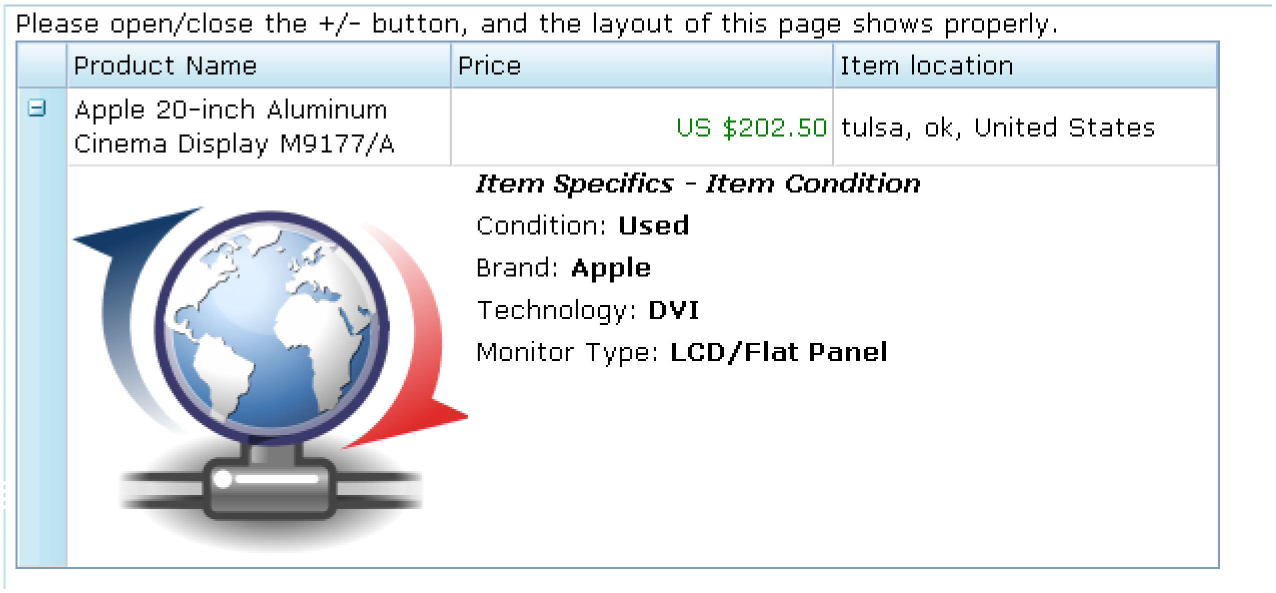
Name	Description	Data Type
getDateFormat()	Returns the date format of the specified format Default: it uses SimpleDateFormat to format the date.	java.text.Dateformat
getDefaultFormat()	Returns the default format, which is used when constructing a datebox.	String
getRealStyleFlags()	Returns RS_NO_WIDTH|RS_NO_HEIGHT	int
GetInnerAttrs()		String
getOuterAttrs()		String

Inherited From

Inherited From
org.zkoss.zul.impl.FormatInputElement
org.zkoss.zul.impl.InputElement
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Detail

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



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

	Product Name	Price	Item location
⊟	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"?>
<zk>
   Please open/close the +/- button, and the layout of this page shows properly.
   <grid fixedLayout="true" width="600px">
      <columns>
         <column width="25px" />
         <column>Product Name</column>
         <column>Price</column>
         <column>Item location</column>
      </columns>
      <rows>
         <row>
            <detail>
                  <hbox>
                  <image width="200px" height="200px" src="/img/icon_update.png"/>
                  <vbox>
                  <label value="Item Specifics - Item Condition  "
                         style="font-weight:bold;font-style: italic;" />
                  <hbox>
                   <label value="Condition:" />
                   <label value="Used"
                         style="font-weight:bold;" />
                  </hbox>
```

```
                   <hbox>
                    <label value="Brand:" />
                    <label value="Apple"
                           style="font-weight:bold;" />
                   </hbox>
                   <hbox>
                    <label value="Technology:" />
                    <label value="DVI"
                           style="font-weight:bold;" />
                   </hbox>
                   <hbox>
                    <label value="Monitor Type:" />
                    <label value="LCD/Flat Panel"
                           style="font-weight:bold;" />
                   </hbox>
                   </vbox>
                </hbox>
             </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>
```

Class Name

org.zkoss.zul.Detail

Supported Child Components

*ALL

Supported Events

Name	Event Type

Properties

Property	Description	Data Type	Default Value
contentStyle	Sets the CSS style for the content block of the window.	jave.lang.String	
image	Sets the URI of the button image.	jave.lang.String	
open	Sets whether the detail is open.	boolean	

Methods

Name	Description	Return Data Type

Inherited From

Inherited From
org.zkoss.zul.impl.HeadersElement
org.zkoss.zul.impl.XulElement

Doublebox

An edit box for holding an float point value (double).

```
<doublebox value="2.3"/>
```

Class Name

`org.zkoss.zul.Doublebox`

Supported Child Components

*NONE

Supported Events

Name	Inherited From
OnClick	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has clicked the component.
OnSelection	org.zkoss.zk.ui.event.SelectionEvent **Description:** 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 getStart and getEnd methods.
OnFocus	org.zkoss.zk.ui.event.Event **Description:** Denotes when a component gets the focus.
OnBlur	org.zkoss.zk.ui.even.Event **Description:** Denotes when a component loses the focus.
OnChange	org.zkoss.zk.ui.even.InputEvent **Description:** An input control notifies the application with the onChange event if its

Name	Inherited From
	content is changed by the user.
OnChanging	org.zkoss.zk.ui.even.InputEvent **Description:** An input control also notifies the application with the onChanging event, when user is changing the content.

Attributes

Property	Description	Data Type	Default Values
value	the value (in Double), might be null unless a constraint stops it.	java.math.BigDecimal	0

Methods

Name	Description	Data Type
intValue()	Returns the value in integer.	int
longValue()	Returns the value in long.	long
doubleValue()	Returns the value in double.	double
shortValue()	Returns the value in short.	short

Inherited From

Inherited From
org.zkoss.zul.impl.NumberInputElement
org.zkoss.zul.impl.FormatInputElement
org.zkoss.zul.impl.InputElement
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Div

The same as `HTML DIV` tag.

An extension. It has the same effect as `<h:div xmlns:h="http://www.w3.org/1999/xhtml">`. Note: a Window without title and caption has the same visual effect as `Div`, but `Div` doesn't implement `IdSpace`. In other words, `Div` won't affect the uniqueness of identifiers.

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

Class Name

`org.zkoss.zul.Div`

Supported Child Components

*All

Supported Events

*NONE

Attributes

Property	Description	Data Type	Values

`align`	The alignment
	Values: `one of left, center, right, ustify.`
		`String`	`<null>`
			`Description: use browser default`

Methods

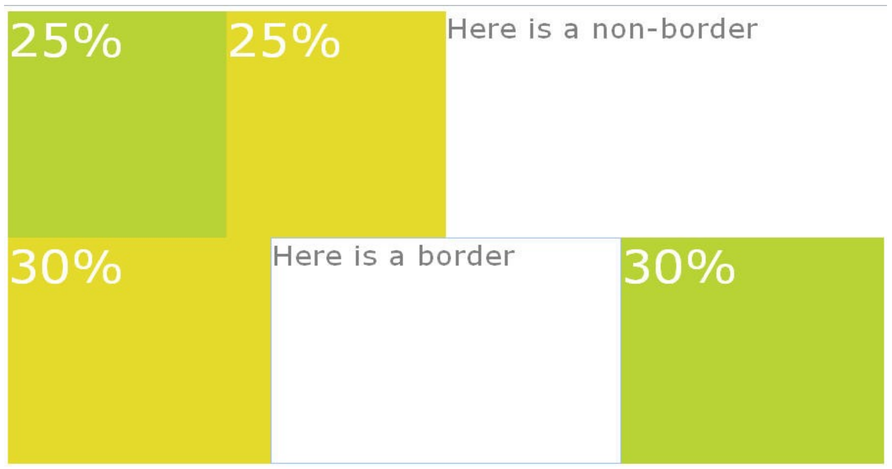
Name	Description	Return Data Type
`getOuterAttrs()`		`String`

Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

East

This component is a east region. The default class of CSS is specified "layout-region-east".



```
<borderlayout height="500px">
   <north size="50%" border="0">
      <borderlayout>
         <west size="25%" border="none" flex="true">
            <div style="background:#B8D335">
               <label value="25%"
                  style="color:white;font-size:50px" />
            </div>
         </west>
         <center border="none" flex="true">
            <div style="background:#E6D92C">
               <label value="25%"
                  style="color:white;font-size:50px" />
            </div>
         </center>
         <east size="50%" border="none" flex="true">
            <label value="Here is a non-border"
               style="color:gray;font-size:30px" />
         </east>
      </borderlayout>
   </north>
   <center border="0">
      <borderlayout>
         <west size="30%" flex="true" border="0">
            <div style="background:#E6D92C">
               <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 size="30%" flex="true" border="0">
            <div style="background:#B8D335">
               <label value="30%"
                  style="color:white;font-size:50px" />
            </div>
         </east>
      </borderlayout>
   </center>
</borderlayout>
```

Class Name

`org.zkoss.zkex.zul.East`

Supported Child Components

*NONE

Supported Events

Name	Inherited From
`OnOpen`	org.zkoss.zk.ui.event.OpenEvent **Description:** When a layout is collapsed or opened by a user, the `onOpen` event is sent to the application.

Properties

Property	Description	Data Type	Default Value
`size`	Sets the size of this region.	`java.lang.String`	`null`

Methods

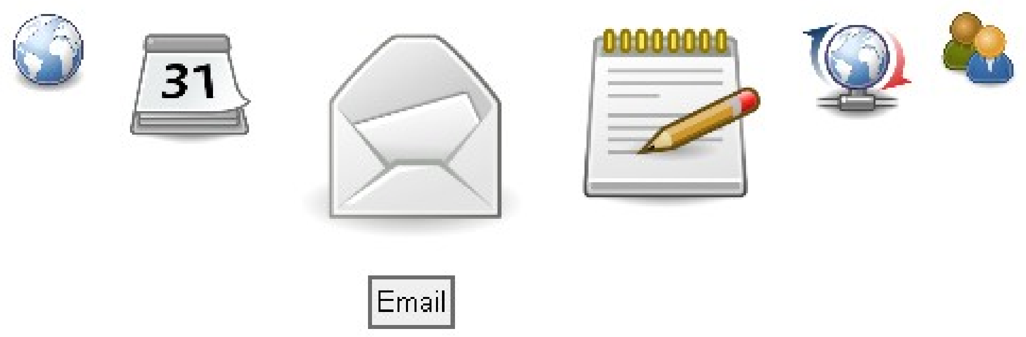
Name	Description	Return Data Type
`getPosition()`	Returns `Borderlayout.NORTH.`	`java.lang.String`
setWidth(java.lang.String width)	The width can't be specified in this component because its width is determined by other region components (West or East).	void

Inherited From

Inherited From
org.zkoss.zkex.zul.LayoutRegion
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Fisheyebar

A fisheye bar is a bar of fisheye that is a menu similar to the fish eye menu on the Mac OS.



```
<window>
   Click "Change orient" button and move cursor over the fisheyebar before it
changes.
   <separator/>
   Fisheyebar will be out of expected.(That is wrong)
   <fisheyebar id="fish" style="position: absolute; top: 50px;
      left:100px;margin:20px;" attachEdge="top">
      <fisheye image="/img/icon_browser.png" label="Web Browser"
         onClick="alert(self.label)" />
      <fisheye image="/img/icon_calendar.png" label="Calendar"
         onClick="alert(self.label)" />
      <fisheye image="/img/icon_email.png" label="Email"
         onClick="alert(self.label)" />
      <fisheye image="/img/icon_texteditor.png" label="Text Editor"
         onClick="alert(self.label)" />
      <fisheye image="/img/icon_update.png" label="Software Update"
         onClick="alert(self.label)" />
      <fisheye image="/img/icon_users.png" label="Users"
         onClick="alert(self.label)" />
   </fisheyebar>
   <button label="Change orient">
      <attribute name="onClick">
         if (fish.orient.equals("horizontal")) {
            fish.orient = "vertical";
         } else {
            fish.orient = "horizontal";
         }
      </attribute>
   </button>
```

```
</window>
```

Class Name

org.zkoss.zkex.zul.Fisheyebar

Supported Child Components

*org.zkoss.zzulkex..fisheye

Supported Events

Name	Event Type

Properties

Property	Description	Data Type	Default Value
orient	Sets the orientation of fisheye. **Value:** horizontal|vertical	jave.lang.String	horizontal
attachEdge	Returns the attach edge. **Value:** center|left|right|top|bottom	jave.lang.String	center
labelEdge	Returns the label edge. **Value:** center|left|right|top|bottom	java.lang.String	bottom

Methods

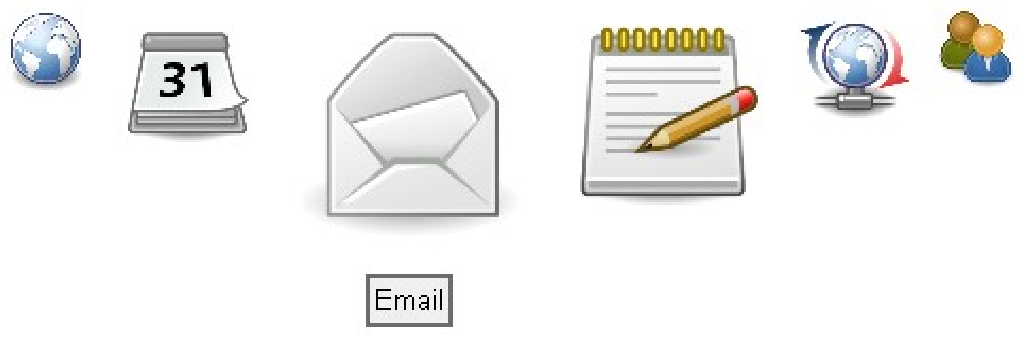
Name	Description	Return Data Type

Inherited From

Inherited From
org.zkoss.zul.impl.HeadersElement
org.zkoss.zul.impl.XulElement

Fisheye

A fisheye item.



```
<window>
   Click "Change orient" button and move cursor over the fisheyebar before it
changes.
   <separator/>
   Fisheyebar will be out of expected.(That is wrong)
   <fisheyebar id="fish" style="position: absolute; top: 50px;
      left:100px;margin:20px;" attachEdge="top">
      <fisheye image="/img/icon_browser.png" label="Web Browser"
         onClick="alert(self.label)" />
      <fisheye image="/img/icon_calendar.png" label="Calendar"
         onClick="alert(self.label)" />
      <fisheye image="/img/icon_email.png" label="Email"
         onClick="alert(self.label)" />
      <fisheye image="/img/icon_texteditor.png" label="Text Editor"
         onClick="alert(self.label)" />
      <fisheye image="/img/icon_update.png" label="Software Update"
         onClick="alert(self.label)" />
      <fisheye image="/img/icon_users.png" label="Users"
         onClick="alert(self.label)" />
   </fisheyebar>
   <button label="Change orient">
      <attribute name="onClick">
         if (fish.orient.equals("horizontal")) {
            fish.orient = "vertical";
         } else {
            fish.orient = "horizontal";
         }
      </attribute>
   </button>
</window>
```

Class Name

`org.zkoss.zkex.zul.Fisheye`

Supported Child Components

*None

Supported Events

Name	Event Type

Properties

Property	Description	Data Type	Default Value
label	Sets the label.	jave.lang.String	
image	Returns the image URI.	jave.lang.String	null

Methods

Name	Description	Return Data Type

Inherited From

Inherited From
org.zkoss.zul.impl.HeadersElement
org.zkoss.zul.impl.XulElement

Flash

A generic flash component.

```
<flash src="SWF/cc.milestones.121503.swf" height="320" width="620"></flash>
```

Class Name

`org.zkoss.zul.Flash`

Supported Child Components

*NONE

Supported Events

*NONE

Properties

Property	Description	Data Type	Default Value
`autoPlay`	Sets wether the song Flash movie playing automatically	`void`	`false`
`bgcolor`	Sets the background color of Flash movie	`void`	`<empty string>`
`loop`	Sets whether the Flash movie plays repeatly	`void`	`true`
`src`	Sets the source path of Flash movie and redraw the component	`java.lang.String`	`<empty string>`
`wode`	Sets the Window Mode property of the Flash movie for transparency, layering, and positioning in the browser. **values:** • `window` - movie plays in its own rectangular window on a web page. • `opaque` - the movie hides everything on the page behind it. • `transparent` - the background of the HTML page shows through all transparent portions of the movie, this may slow animation performance.	`java.lang.String`	`transparent`

Methods

*NONE

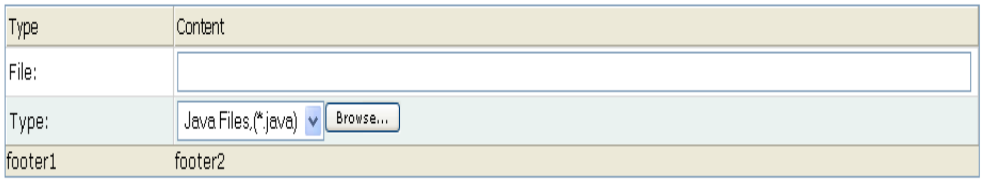
Inherited From

Inherited From
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Footer

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.



```
<grid>
   <columns>
      <column label="Type"/>
      <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>
```

Class Name

`org.zkoss.zul.Footer`

Supported Child Components

*ALL

Supported Events

*NONE

Properties

Property	Description	Data Type	Default Value
span	Sets wether the song Flash movie playing automatically	int	1

Methods

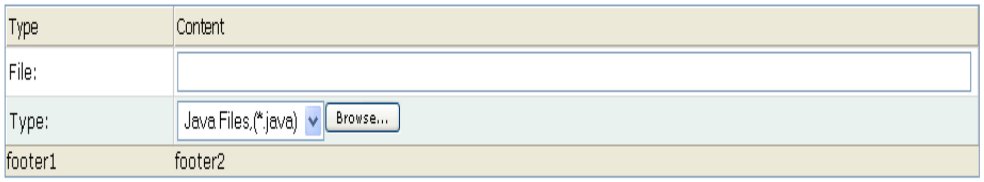
Name	Description	Return Data Type
getColumn()	Returns the column that is in the same column as this footer, or null if not available.	org.zkoss.zul.Column
getColumnIndex()	Returns the column index, starting from 0.	int
getGrid()	Returns the grid that this belongs to.	org.zkoss.zul.Grid
setParent(org.zkoss.zk.ui.Component parent)		void

Inherited From

Inherited From
org.zkoss.zul.impl.LabelElement
org.zkoss.zul.impl.LabelImageElement
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Foot

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



```
<grid>
   <columns>
      <column label="Type"/>
      <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>
```

Class Name

`org.zkoss.zul.Foot`

Supported Child Components

Footer

Supported Events

*NONE

Properties

Property	Description	Data Type
`value`	Sets the current value of the progress meter.	`java.langString`

Methods

Name	Description	Return Data Type
`getGrid()`	Returns the grid that this belongs to.	`org.zkoss.zul.Grid`
`insertBefore(org.zkoss.zk.ui.Component child, org.zkoss.zk.ui.Component insertBefore)`		`boolean`
`setParent(org.zkoss.zk.ui.Component parent)`		`void`

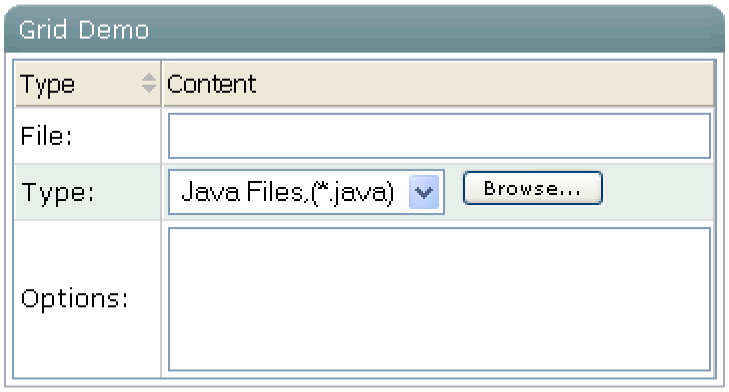
Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Grid

Components: `grid, columns, column, rows and row.`

A `grid` contains components that are aligned in rows like tables. Inside a `grid`, you declare two things, the `columns`, that define the header and `column` attributes, and the `rows`, that provide the content. To declare a set of rows, use the `rows` component, which should be a child element of `grid`. Inside that you should add `row` components, which are used for each `row`. Inside the `row` element, you should place the content that you want inside that `row`. Each child is a `column` of the specific `row`. Similarly, the `columns` are declared with the `columns` component, which should be placed as a child element of the `grid`. Unlike `row` is used to hold the content of each `row`, `column` declares the common attributes of each `column`, such as the width and alignment, and and optional headers, i.e., label and/or image.



```
<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="&#36;{asc}" sortDescending="&#36;
{dsc}"/>
         <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>
```

Class Name

`org.zkoss.zul.Grid`

Supported Child Components

Columns Rows

Supported Events

Name	Event Type
onPaging	org.zkoss.zul.event.PagingEvent **Description:** Notifies one of the pages of a multi-page component is selected by the user.

Properties

Property	Description	Data Type	Default Value
align	Sets the horizontal alignment of the whole grid. **Value:** left|center|right	java.lang.String	<null>
model	Sets the list model associated with this grid.	org.zkoss.zul.ListModel	<null>
pageSize	Sets the page size, aka., the number rows per page. **Note:** Available only the paging mold	int	<null>
paginal		org.zkoss.zul.ext.Paginal	<null>
preloadSize	Sets the number of rows to preload when receiving the rendering request from the client.	int	7
rowrenderer	Sets the renderer which is used to render each row if getModel() is not null.	org.zkoss.zul.RowRenderer	<null>

Methods

Name	Description	Return Data Type
clone()		java.lang.Object
getCell(int, int)	Returns the specified cell, or null if not available.	org.zkoss.zk.ui.Component
getColumns()	Returns the columns.	org.zkoss.zul.Columns
getFoot()	Returns the foot.	org.zkoss.zul.Foot
getOuterAttrs()		java.lang.String
getPaging()	Returns the child paging controller that is created automatically, or null if mold is not "paging", or the controller is specified externally by setPaginal(org.zkoss.zul.ext.Paginal).	org.zkoss.zul.Paging
getRows()	Returns the rows.	org.zkoss.zul.Rows
insertBefore(org.zkoss.zk.ui.Component,		boolean

Name	Description	Return Data Type
org.zkoss.zk.ui.Component)		
onInitRender()	Handles a private event, onInitRender.	void
onPaging()	Called when the onPaging event is received (from getPaginal()).	void
removeChild(org.zkoss.zk.ui.Component)		boolean
renderAll()	Renders all Row if not loaded yet, with getRowRenderer().	void
renderItems(java.util.Set)		void
renderRow(Row)	Renders the specified Row if not loaded yet, with getRowRenderer().	void
renderRows(java.util.Set)	Renders a set of specified rows.	void
setMold(ListModel)		void
setRowRenderer(java.lang.String)	Sets the renderer by use of a class name.	void

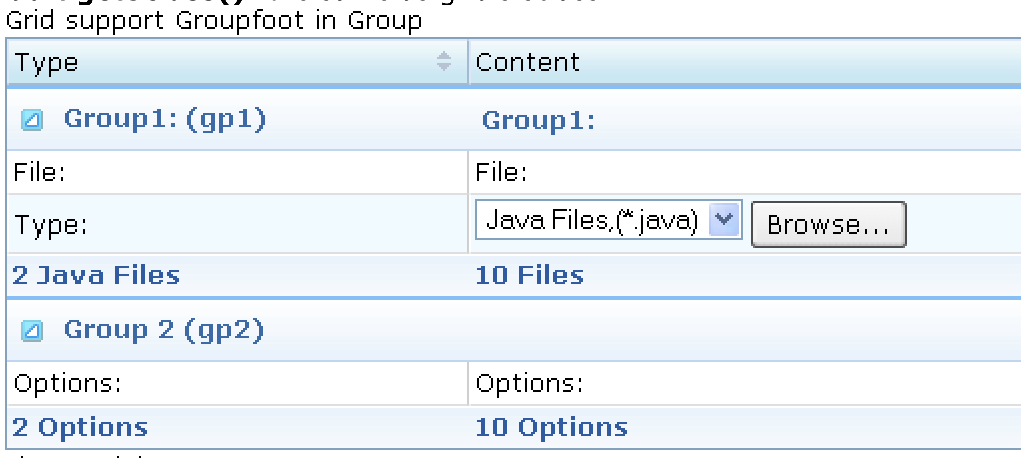
Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Group

Adds the ability for single level grouping to the Grid.

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



Grid support Groupfoot in Group

Type	Content
Group1: (gp1)	Group1:
File:	File:
Type:	Java Files,(*.java) Browse...
2 Java Files	10 Files
Group 2 (gp2)	
Options:	Options:
2 Options	10 Options

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

   <grid id="grid">
      <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>
         <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>
```

Class Name

org.zkoss.zul.Group

Supported Child Components

*ALL

Supported Events

*NONE

Properties

Property	Description	Data Type	Default Value
align	Sets the horizontal alignment of the whole grid. **Value:** left|center|right	java.lang.String	<null>
nowrap	Sets the nowrap.	boolean	false
sclass	Sets the style class.	java.lang.String	<null>
spans	Sets the spans, which is a list of numbers separated by comma.	java.lang.String	<null>
valign	Sets the vertical alignment of the whole row.	java.lang.String	<null>
value	Sets the value.	java.lang.Object	<null>
label	Sets the value of the Label it contains	java.lang.String	<null>

Methods

Name	Description	Return Data Type
`getChildAttrs(int)`	Returns the HTML attributes for the child of the specified index.	`java.lang.String`
`getGrid()`	Returns the grid that contains this row.	`org.zkoss.zul.Grid`
`getOuterAttrs()`		`java.lang.String`
`onDrawNewChild(org.zkoss.zk.ui.Component, java.lang.StringBuffer)`		`void`
`setParent(org.zkoss.zk.ui.Component)`		`void`
`setStyle(java.lang.String)`		`void`

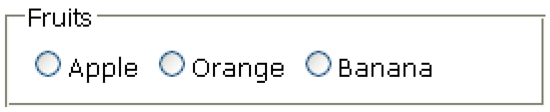
Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Groupbox

Components: `groupbox`.

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 the `caption` component. It works much like the HTML legend element. Unlike windows, a group box is not an owner of the ID space. It cannot be overlapped or popup.



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

Class Name

`org.zkoss.zul.Groupbox`

Supported Child Components

*ALL

Supported Events

Name	Event Type
`onClick`	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has clicked the component.
`onRightClick`	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has right-clicked the component.
`onDoubleClick`	org.zkoss.zk.ui.event.MouseEvent

Name	Event Type
	Description: Denotes user has double-clicked the component.
onOpen	org.zkoss.zk.ui.event.OpenEvent **Description:** 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.

Properties

Property	Description	Data Type	Default Value
closable	Sets whether user can open or close the group box.	boolean	true
contentStyle	Sets the CSS style for the content block of the groupbox.	java.lang.String	<null>
open	Opens or closes this groupbox.	boolean	true

Methods

Name	Description	Return Data Type
getCaption()	Returns the caption of this groupbox.	org.zkoss.zul.Caption
getContentSclass()	Returns the style class used for the content block of the groupbox.	java.lang.String
getOuterAttrs()		java.lang.String
insertBefore(org.zkoss.zk.ui.Component, org.zkoss.zk.ui.Component)		boolean
onChildRemoved(org.zkoss.zk.ui.Component)		void

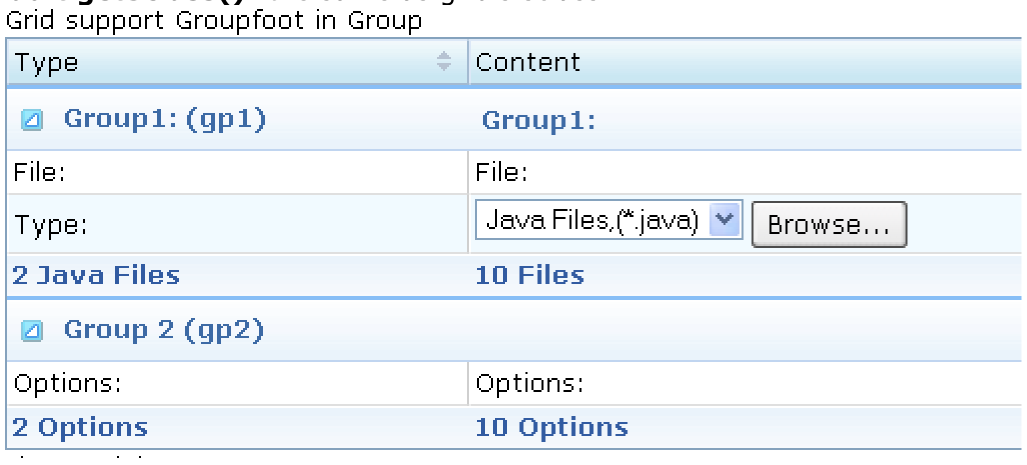
Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Groupfoot

Adds the ability for single level grouping to the Grid.

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



Grid support Groupfoot in Group

Type	Content
Group1: (gp1)	Group1:
File:	File:
Type:	Java Files,(*.java) Browse...
2 Java Files	10 Files
Group 2 (gp2)	
Options:	Options:
2 Options	10 Options

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

   <grid id="grid">
      <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>
         <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>
```

Class Name

org.zkoss.zul.Groupfoot

Supported Child Components

*ALL

Supported Events

*NONE

Properties

Property	Description	Data Type	Default Value
align	Sets the horizontal alignment of the whole grid. **Value:** left|center|right	java.lang.String	<null>
nowrap	Sets the nowrap.	boolean	false
sclass	Sets the style class.	java.lang.String	<null>
spans	Sets the spans, which is a list of numbers separated by comma.	java.lang.String	<null>
valign	Sets the vertical alignment of the whole row.	java.lang.String	<null>
value	Sets the value.	java.lang.Object	<null>
label	Sets the value of the Label it contains	java.lang.String	<null>

Methods

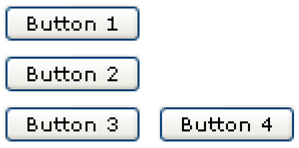
Name	Description	Return Data Type
`getChildAttrs(int)`	Returns the HTML attributes for the child of the specified index.	`java.lang.String`
`getGrid()`	Returns the grid that contains this row.	`org.zkoss.zul.Grid`
`getOuterAttrs()`		`java.lang.String`
`onDrawNewChild(org.zkoss.zk.ui.Component, java.lang.StringBuffer)`		`void`
`setParent(org.zkoss.zk.ui.Component)`		`void`
`setStyle(java.lang.String)`		`void`

Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Hbox

The `hbox` component is used to create a horizontally oriented box. Each component placed in the `hbox` will be placed horizontally in a row.



```
<zk>
   <vbox>
      <button label="Button 1"/>
      <button label="Button 2"/>
   </vbox>
   <hbox>
      <button label="Button 3"/>
      <button label="Button 4"/>
   </hbox>
</zk>
```

Class Name

`org.zkoss.zul.Hbox`

Supported Child Components

*ALL

Supported Events

*NONE

Properties

*NONE

Methods

*NONE

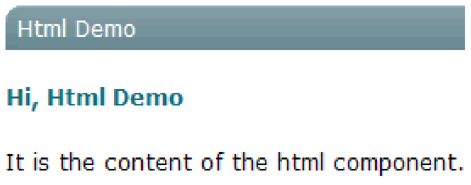
Inherited From

Inherited From
org.zkoss.zul.Box
org.zkoss.zul.impl.XulElement

Inherited From
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Html

The simplest way is to use a 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.



Hi, Html Demo

It is the content of the html component.

```
<window id="win" title="Html Demo">
   <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.

```
<span id=”...”>
   <h4>Hi, Html Demo</h4>
   <p>It is the content of the html component.</p>
</span>
```

Class Name

`org.zkoss.zul.Html`

Supported Child Components

*NONE

Supported Events

*NONE

Properties

Property	Description	Data Type	Default Value
`content`	Returns the embedded content (i.e., HTML tags).	`String`	`empty ("")`

Methods

Name	Description	Data Type	Values
`IsChildable` **(Source Text)**	Determines whether it accepts child components **Note:** No child is allowed.	`Boolean` **(Source Text)**	`false`

Inherited From

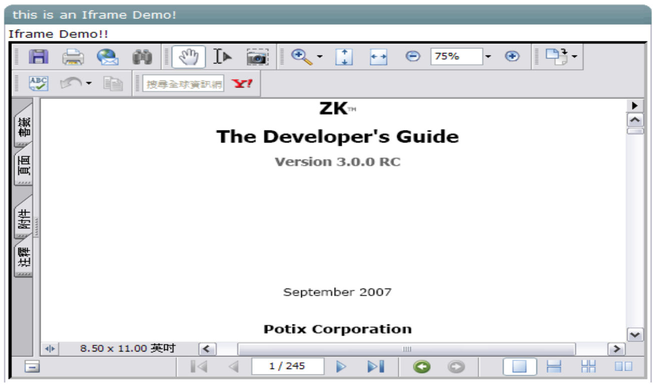
Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Iframe

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.



```
<window id="win" title="This is an Iframe Demo!">
   <iframe style="width:99%; height:400px;border:3px inset;"
      src="/zk-devguide.pdf" />
</window>
```

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[47], 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.

Class Name

`org.zkoss.zul.Iframe`

Supported Child Components

*NONE

Supported Events

Name	Event Type
`onURIChange`	org.zkoss.zk.ui.event.URIEvent **Description:** Denotes the associated URI (`src`) has been changed by user. Use `getURI()` to retrieve the URI being changed to.

Properties

Property	Description	Data Type	Default Value
`content`	`org.zkoss.util.media.Media` any binary content that client side browser accept (i.e., mp3, pdf...).	`Media`	`null`

Methods

Name	Description	Return Data Type
`IsChildable`	Determines whether it accepts child components **Note:** No child is allowed.	`Boolean`

Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Image

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">
```

Like using any other properties that accept an URI, you could specify "*" for identifying a Locale dependent image. For example, if you have different image for different Locales, you could use as follows.

```
<image src="/my*.png">
```

Then, assume 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 not found, it will try `/my_de.png` and finally `/my.png`.

Class Name

`org.zkoss.zul.Image`

Supported Child Components

*NONE

Supported Events

*NONE

Properties

Property	Description	Data Type	Default Value
`content`	`org.zkoss.image.Image` an image object (i.e., jpeg, png...).	`Image`	`null`

Methods

Name	Description	Data Type	Values
`IsChildable`	Determines whether it accepts child components	`Boolean`	`false`

Name	Description	Data Type	Values
	Note: No child is allowed.		

Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Imagemap

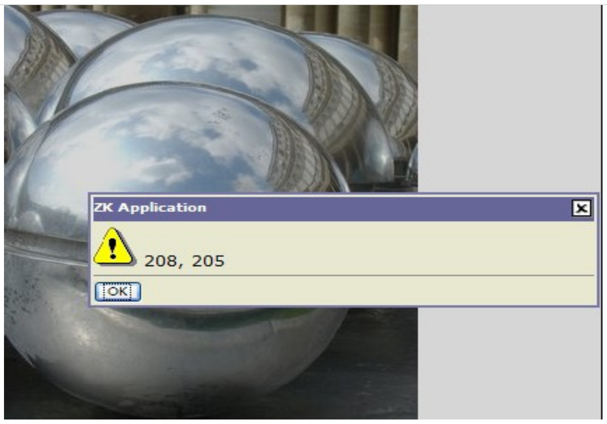
A `imagemap` component is a special image. It accepts whatever properties an `image` component accepts. However, unlike `image`, if a user clicks on the image, an `onClick` event is sent back to the server with the coordinates of the mouse position. In contrast, the `onClick` event sent by `image` doesn't 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 the application receives the `onClick` event, it could examine the coordinates of the mouse position from the `getX` and `getY` methods.

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.

```
<imagemap src="/img/sun.jpg" onClick="alert(event.x + &quot;, &quot; +event.y)"/>
```



Then, the user gets the result as depicted below.

Class Name

`org.zkoss.zul.Imagemap`

Supported Child Components

*NONE

Supported Events

Name	Event Type
`onClick`	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has clicked the component. Use getX(), getY() method get coordinates.

Properties

Property	Description	Data Type	Default Value
`content`	`org.zkoss.image.Image` an image object (i.e., jpeg, png...).	`Image`	`null`

Methods

Name	Description	Return Data Type
`IsChildable`	Determines whether it accepts child components **Note:** No child is allowed.	`Boolean`

Inherited From

Inherited From
org.zkoss.zul.Image
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Include

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/includedHello.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 the run time.

If the included output is another ZUML, developers are allowed 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.

Pass Values to the Included Page

There are two ways to pass values to the included page. First, you can pass them with the query string.

```
<include src="mypage?some=something"/>
```

Then, in the included page, you can access them with the `getParameter` method of the `Execution` interface or the `ServletRequest` interface. In EL expressions (of the included page), you can use the `param` variable to access them. However, you can only pass String-typed values with the query string.

```
${param.some}
```

Alternatively, we can pass any kind of values with the so-called dynamic properties by use of the `setDynamicProperty` method or, in ZUL, a dynamic property as follows:

```
<include 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 the `getAttribute` method of the `Execution` interface or the

ServletRequest interface. In EL expressions (of the included page), you can use the requestScope variable to access them.

```
${requestScope.some}
```

Class Name

org.zkoss.zul.Include

Supported Child Components

*NONE

Supported Events

*NONE

Properties

Property	Description	Data Type	Default Value
src	Sets whether user can open or close the group box.	boolean	true
localized	Sets the CSS style for the content block of the groupbox.	java.lang.String	<null>
open	Opens or closes this groupbox.	boolean	true

Methods

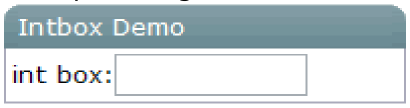
Name	Description	Data Type	Values
IsChildable	Determines whether it accepts child components **Note:** No child is allowed.	Boolean	false

Inherited From

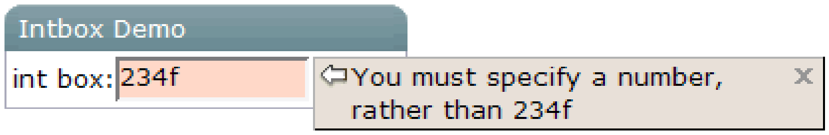
Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Intbox

A `intbox` is used to let users input integer data.



While input invalid data:



```
<window title="Intbox Demo" border="normal" width="200px">
   int box:<intbox/>
</window>
```

Class Name

`org.zkoss.zul.Intbox`

Supported Child Components

*NONE

Supported Events

Event Name	Event Type
`onChange`	org.zkoss.zk.ui.event.InputEvent **Description:** Denotes the content of an input component has been modified by the user.
`onChanging`	org.zkoss.zk.ui.event.InputEvent **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 `getValue` method in the `InputEvent` class to retrieve the temporary value.
`onSelection`	org.zkoss.zk.ui.event.SelectionEvent **Description:** Denotes that user is selecting a portion of the text of an input component. You can retrieve the start and end

Event Name	Event Type
	position of the selected text by use of the `getStart` and `getEnd` methods.
`onFocus`	org.zkoss.zk.ui.event.Event **Description:** 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`	org.zkoss.zk.ui.event.Event **Description:** 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.
`onCreate`	org.zkoss.ui.zk.ui.event.CreateEvent **Description:** Denotes a component is created when rendering a ZUML page.
`onDrop`	org.zkoss.ui.zk.ui.event.DropEvent **Description:** Denotes another component is dropped to the component that receives this event.

Properties

Property	Description	Return Data Type
`value`	Sets the text value.	`Integer`

Methods

*NONE

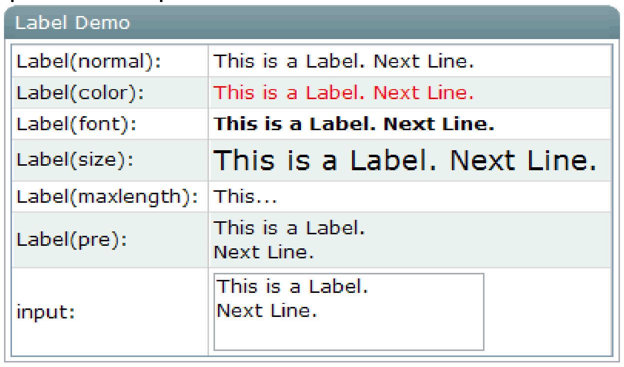
Inherited From

Inherited From
org.zkoss.zul.NumberInputElement
org.zkoss.zul.FormatInputElement
org.zkoss.zul.InputElement
org.zkoss.zul.imp.XulElement

Inherited From
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Label

A label component represents a piece of text.



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.

Class Name

`org.zkoss.zul.Label`

Supported Child Components

*NONE

Supported Events

*NONE

Properties

Property	Description	Data Type
`value`	The `String` value denote this label.	`String`
`pre`	If true, all white spaces, such as new line, space and tab, are preserved.	`boolean`
`maxlength`	Truncated the characters that exceeds the specified	`Positive Integer`

Methods

Name	Description	Return Data Type
`IsChildable`	Determines whether it accepts child components **Note:** No child is allowed.	`Boolean`

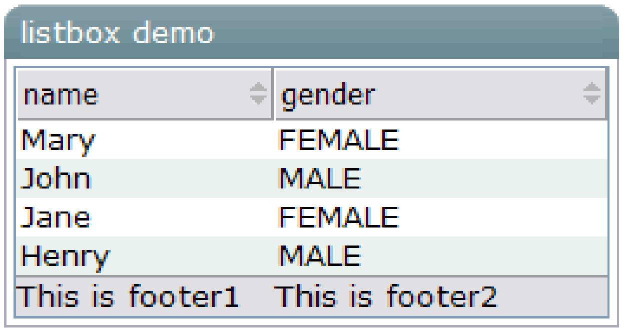
Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Listbox

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.



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

```
<window title="listbox demo" border="normal">
      <listbox id="box" width="250px">
         <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>
```

Listbox has two molds: `default` and `select`. If the select mold is used, the HTML's SELECT tag is generated instead.

Class Name

`org.zkoss.zul.Listbox`

Supported Child Components

Listitem Listhead Listfoot

Supported Events

Name	Event Type
`onPaging`	org.zkoss.zul.event.PagingEvent **Description:** Notifies one of the pages of a multi-page component is selected by the user.

Properties

Property	Description	Data Type	Default Value
align	Sets the horizontal alignment of the whole Listbox. **Value:** left|center|right	java.lang.String	<null>
model	Sets the list model associated with this Listbox.	org.zkoss.zul.ListModel	<null>
pageSize	Sets the page size, aka., the number rows per page. **Note:** Available only the paging mold	int	<null>
paginal		org.zkoss.zul.ext.Paginal	<null>
preloadSize	Sets the number of rows to preload when receiving the rendering request from the client.	int	7
itemRenderer	Sets the renderer which is used to render each `Listitem` if getModel() is not null.	org.zkoss.zul.RowRenderer	<null>
maxlength	the maximal length of each item's label.	int	0(no effect)
multiple	Is multiple selections are allowed.	boolean	false
checkmark	Is the check mark shall be displayed in front of each item.	boolean	false
disable	Is this `Listbox` is disabled.	boolean	false
vflex	To grow and shrink vertical to fit their given space, so called vertial flexibility.	boolean	false

Methods

Name	Description
clone()	
getIndexOfItem(Listitem)	Returns the index of the specified item, or -1 if not found.

Name	Description
`ClearSelection()`	Clears the selection.
`addItemToSelection(Listitem)`	Selects the given item, without deselecting any other items that are already selected..
`appendItem(String,String)`	Appends an item.
`getItemAtIndex(int)`	Returns the item at the specified index.
`getSelectedIndex()`	Returns the index of the selected item (-1 if no one is selected).
`setSelectedIndex(int)`	Deselects all of the currently selected items and selects the item with the given index.
`GetItemCount()`	Returns the number of items.
`GetListHead()`	Returns `Listhead` belonging to this L`istbox`, or null if no list headers at all.
`GetListfoot()`	Returns `Listfoot` belonging to this L`istbox`, or null if no list footers at all.
`getOuterAttrs()`	
`getPaging()`	Returns the child paging controller that is created automatically, or null if mold is not "paging", or the controller is specified externally by setPaginal(org.zkoss.zul.ext.Paginal).
`SelectAll()`	Select all items.
`insertBefore(org.zkoss.zk.ui.Component, org.zkoss.zk.ui.Component)`	
`onInitRender()`	Handles a private event, onInitRender.
`onPaging()`	Called when the onPaging event is received (from getPaginal()).
`removeChild(org.zkoss.zk.ui.Component)`	
`renderAll()`	Renders all `Listitem` if not loaded yet, with `getItemRenderer().`
`renderItems(java.util.Set)`	
`renderItem(Listitem)`	Renders the specified Row if not loaded yet, with getRowRenderer().
`renderItems(java.util.Set)`	Renders a set of specified rows.
`setTabindex(int)`	Sets the tab order of this component.

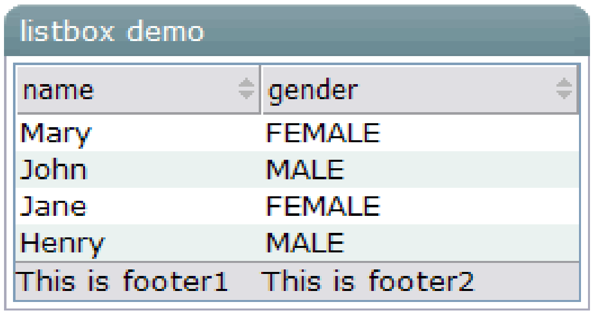
Name	Description
	Currently, only the "select" mold supports this property.
`getTabindex()`	Returns the tab order of this component. Currently, only the "select" mold supports this property. Default: -1 (means the same as browser's default).
`setItemRenderer(java.lang.String)`	Sets the renderer by use of a class name.
`toggleItemSelection(Listitem)`	If the specified item is selected, it is deselected.

Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Listcell

A list cell.



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

```
<window title="listbox demo" border="normal">
      <listbox id="box" width="250px">
         <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>
```

Class Name

`org.zkoss.zul.Listcell`

Supported Child Components

*ALL

Supported Events

*NONE

Properties

Name	Description	Return Data Type
`span`	Number of columns to span this footer.	`int`
`value`	The value this cell stored.	`java.lang.Object`
`width`	the width which the same as `getListheader()`'s width.	`String`

Methods

Name	Description	Return Data Type
`getListbox()`	Returns the `listbox` that contains this column.	`org.zkoss.zul.Listbox`
`getColumnHtmlPostfix()`		`java.lang.String`
`getColumnHtmlPrefix()`		`java.lang.String`
`getOuterAttrs()`		`java.lang.String`
`getColumnIndex()`	`Returns the column index, starting from 0.`	`int`
`setParent(org.zkoss.zk.ui.Component parent)`	Can only be `Listhead`	`void`
`getListheader()`	Returns the list header that is in the same column as this footer, or null if not available.	`org.zkoss.zul.ListHeader`
`Invalidate()`		

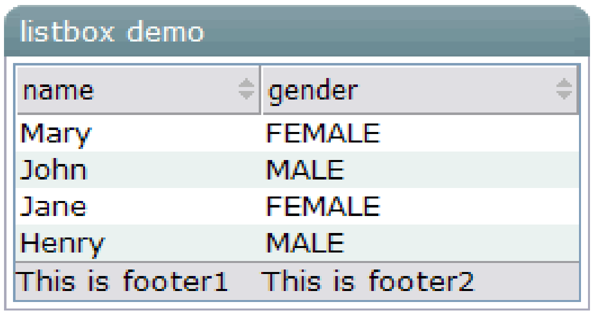
Inherited From

Inherited From
org.zkoss.zul.impl.LabelImageElement
org.zkoss.zul.impl.LabelElement
org.zkoss.zul.impl.XulElement

Inherited From
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Listfoot

Like `Listhead`, each listbox has at most one `Listfoot`.



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

```
<window title="listbox demo" border="normal">
      <listbox id="box" width="250px">
         <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>
```

Class Name

`org.zkoss.zul.Listfoot`

Supported Child Components

Listfooter

Supported Events

*NONE

Properties

*NONE

Methods

Name	Description	Return Data Type
`getListbox()`	Returns the `listbox` that contains this column.	`org.zkoss.zul.Listbox`

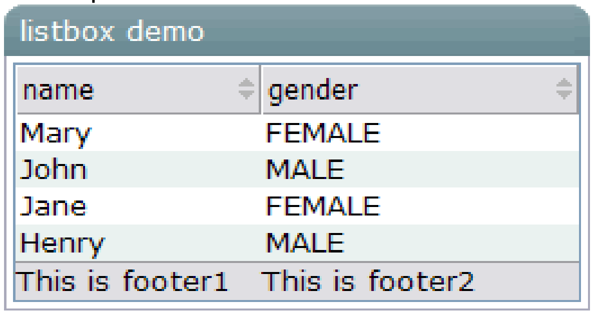
Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Listfooter

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.



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

```
<window title="listbox demo" border="normal">
      <listbox id="box" width="250px">
         <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>
```

Class Name

`org.zkoss.zul.Listfooter`

Supported Child Components

*ALL

Supported Events

*NONE

Properties

Name	Description	Return Data Type
`span`	Number of columns to span this footer.	`int`

Methods

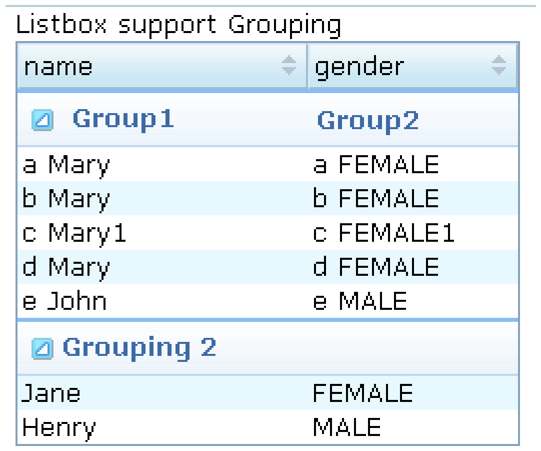
Name	Description	Return Data Type
`getListbox()`	Returns the `listbox` that contains this column.	`org.zkoss.zul.Listbox`
`getOuterAttrs()`		`java.lang.String`
`getColumnIndex()`	`Returns the column index, starting from 0.`	`int`
`setParent(org.zkoss.zk.ui.Component parent)`	Can only be `Listhead`	`void`
`getListfoot()`	Returns the set of footers that this belongs to.	`org.zkoss.zul.Listfoot`
`getListheader()`	Returns the list header that is in the same column as this footer, or null if not available.	`org.zkoss.zul.ListHeader`

Inherited From

Inherited From
org.zkoss.zul.impl.LabelImageElement
org.zkoss.zul.impl.LabelElement
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Listgroup

Adds the ability for single level grouping to the Listbox.


Listbox support Grouping

name	gender
Group1	Group2
a Mary	a FEMALE
b Mary	b FEMALE
c Mary1	c FEMALE1
d Mary	d FEMALE
e John	e MALE
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 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>
```

Class Name

org.zkoss.zul.Listgrroup

Supported Child Components

*org.zkoss.zul.Listcell

Supported Events

Name	Event Type

Properties

Property	Description	Data Type	Default Value
`maxlength`	the maximal length of this item's label.	`int`	
`index`	the index of this item (aka., the order in the listbox).	`int`	
`value`	The value this cell stored.	`Java.lang.Object`	
`label`	the width which the same as `getListheader()`'s width.	`String`	
`src`	the src of the `Listcell` it contains, or null if no such cell.	`String`	
`image`	Returns the image of the `Listcell` it contains.	`String`	
`disable`	Is this `Listitem` is disabled.	`boolean`	`false`
`selected`	Is this `Listitem` is selected.	`boolean`	`false`

Methods

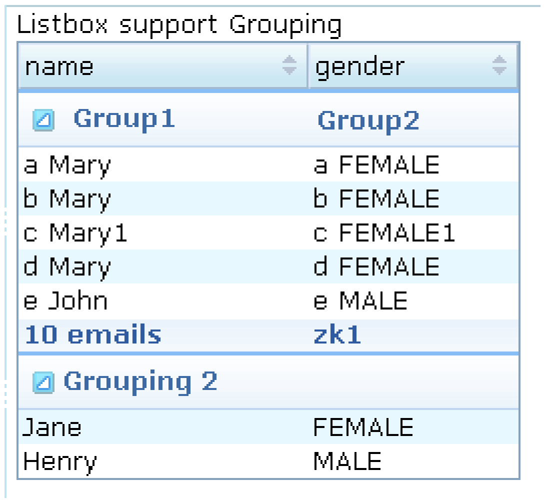
Name	Description	Return Data Type
`getListbox()`	Returns the `listbox` that contains this column.	`org.zkoss.zul.Listbox`
`getOuterAttrs()`		`java.lang.String`
`setParent(org.zkoss.zk.ui.Component parent)`	Can only be `Listbox`	`void`
`Invalidate()`		

Inherited From

Inherited From
org.zkoss.zul.Listitem
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Listgroupfoot

GroupFooter serves as a summary listitem of listgroup.


Listbox support Grouping

name	gender
Group1	Group2
a Mary	a FEMALE
b Mary	b FEMALE
c Mary1	c FEMALE1
d Mary	d FEMALE
e John	e MALE
10 emails	zk1
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 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>
      <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>
```

Class Name

`org.zkoss.zul.Listgroupfoot`

Supported Child Components

*Listcell

Supported Events

Name	Event Type

Properties

Property	Description	Data Type	Default Value
`maxlength`	the maximal length of this item's label.	`int`	
`index`	the index of this item (aka., the order in the listbox).	`int`	
`value`	The value this cell stored.	`Java.lang.Object`	
`label`	the width which the same as `getListheader()`'s width.	`String`	
`src`	the src of the `Listcell` it contains, or null if no such cell.	`String`	
`image`	Returns the image of the `Listcell` it contains.	`String`	
`disable`	Is this `Listitem` is disabled.	`boolean`	`false`
`selected`	Is this `Listitem` is selected.	`boolean`	`false`

Methods

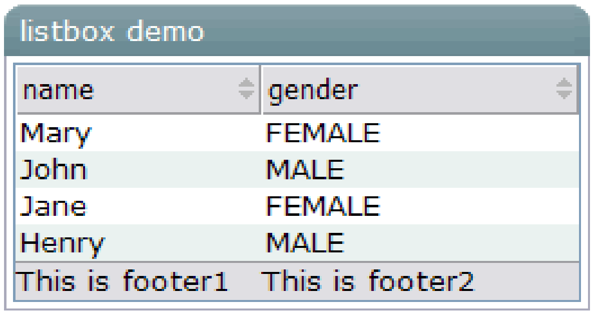
Name	Description	Return Data Type
`getListbox()`	Returns the `listbox` that contains this column.	`org.zkoss.zul.Listbox`
`getOuterAttrs()`		`java.lang.String`
`setParent(org.zkoss.zk.ui.Component parent)`	Can only be `Listbox`	`void`
`Invalidate()`		

Inherited From

Inherited From
org.zkoss.zul.Listitem
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Listhead

A list headers used to define multi-columns and/or headers. Can only surpport `Listheader` as its child.


listbox demo

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

```
<window title="listbox demo" border="normal">
      <listbox id="box" width="250px">
         <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>
```

Class Name

`org.zkoss.zul.Listhead`

Supported Child Components

Listheader

Supported Events

*NONE

Properties

*NONE

Methods

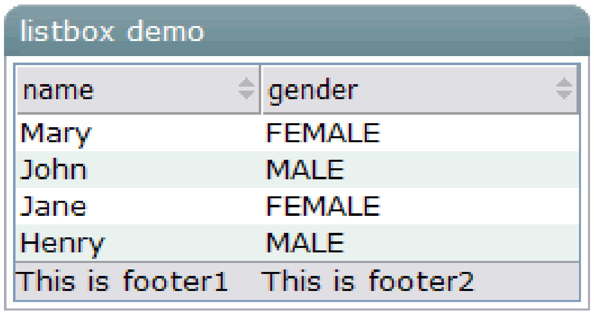
Name	Description	Return Data Type
`getListbox()`	Returns the `listbox` that contains this column.	`org.zkoss.zul.Listbox`

Inherited From

Inherited From
org.zkoss.zul.impl.HeaderElement
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Listheader

The list header which defines the attributes and header of a columen of a list box. Its parent must be `Listhead`.



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

```
<window title="listbox demo" border="normal">
      <listbox id="box" width="250px">
         <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>
```

Class Name

`org.zkoss.zul.Listheader`

Supported Child Components

*NONE

Supported Events

Name	Event Type
onClick	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has clicked the component.
onRightClick	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has right-clicked the component.
onDoubleClick	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has double-clicked the component.

Properties

Property	Description	Data Type
sortAscending	Sets the ascending sorter, or null for no sorter for the ascending order.	java.util.Comparator
sortDescending	Sets the descending sorter, or null for no sorter for the descending order.	java.util.Comparator
sortDirection	Sets the sort direction. **Value:** ascending|descending|natural	java.lang.String
maxlength	the maximal length of each item's label.	Positive integer

Methods

Name	Description	Return Data Type
getListbox()	Returns the listbox that contains this column.	org.zkoss.zul.Listbox
getOuterAttrs()		java.lang.String

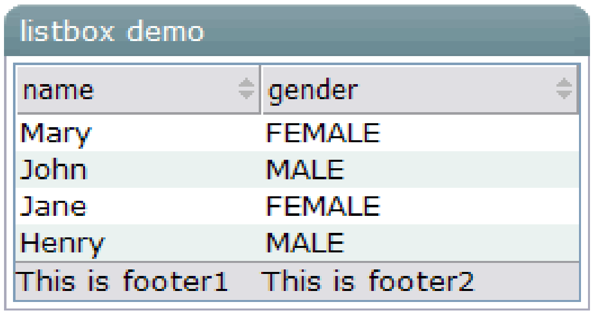
Name	Description	Return Data Type
`getSclass()`	Returns the style class.	`java.lang.String`
`onSort()`	It invokes `sort(boolean)` to sort list items and maintain `getSortDirection()`.	`void`
`getColumnIndex()`	`Returns the column index, starting from 0.`	`int`
`setParent(org.zkoss.zk.ui.Component parent)`	Can only be `Listhead`	`void`
`setSortAscending(java.lang.String)`	Sets the ascending sorter with the class name, or null for no sorter for the ascending order.	`void`
`setSortDescending(java.lang.String)`	Sets the descending sorter with the class name, or null for no sorter for the descending order.	`void`
`sort(boolean)`	Sorts the rows (`Row`) based on `getSortAscending()` and `getSortDescending()`, if `getSortDirection()` doesn't matches the ascending argument.	`boolean`
`sort(boolean, boolean)`	Sorts the rows (`Row`) based on `getSortAscending()` and `getSortDescending()`.	`boolean`

Inherited From

Inherited From
org.zkoss.zul.impl.HeaderElement
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Listitem

A list item.



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

```
<window title="listbox demo" border="normal">
      <listbox id="box" width="250px">
         <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>
```

Class Name

`org.zkoss.zul.Listitem`

Supported Child Components

Listcell

Supported Events

Name	Event Type
`onSelect`	**org.zkoss.zul.event.SelectEvent** `**Description:** Represents an event cause by user's the list selection is changed at the client.`

Properties

Name	Description	Data Type	Default Value
`maxlength`	the maximal length of this item's label.	`int`	
`index`	the index of this item (aka., the order in the listbox).	`int`	
`value`	The value this cell stored.	`java.lang.Object`	
`label`	the width which the same as `getListheader()`'s width.	`String`	
`src`	the src of the `Listcell` it contains, or null if no such cell.	`String`	
`image`	Returns the image of the `Listcell` it contains.	`String`	
`disable`	Is this `Listitem` is disabled.	`boolean`	`false`
`selected`	Is this `Listitem` is selected.	`boolean`	`false`

Methods

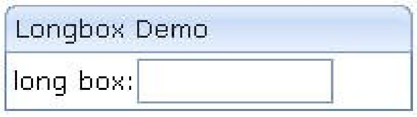
Name	Description	Return Data Type
`getListbox()`	Returns the `listbox` that contains this column.	`org.zkoss.zul.Listbox`
`getOuterAttrs()`		`java.lang.String`
`setParent(org.zkoss.zk.ui.Component parent)`	Can only be `Listbox`	`void`
`Invalidate()`		

Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Longbox

A long`box` is used to let users input long data.



```
<window title="Longbox Demo" border="normal" width="200px">
   long box:<longbox/>
</window>
```

Class Name

`org.zkoss.zul.Longbox`

Supported Child Components

*NONE

Supported Events

Event Name	Event Type
`onChange`	org.zkoss.zk.ui.event.InputEvent **Description:** Denotes the content of an input component has been modified by the user.
`onChanging`	org.zkoss.zk.ui.event.InputEvent **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 `getValue` method in the `InputEvent` class to retrieve the temporary value.
`onSelection`	org.zkoss.zk.ui.event.SelectionEvent **Description:** 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 `getStart` and `getEnd` methods.
`onFocus`	org.zkoss.zk.ui.event.Event **Description:**

Event Name	Event Type
	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`	org.zkoss.zk.ui.event.Event **Description:** 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.
`onCreate`	org.zkoss.ui.zk.ui.event.CreateEvent **Description:** Denotes a component is created when rendering a ZUML page.
`onDrop`	org.zkoss.ui.zk.ui.event.DropEvent **Description:** Denotes another component is dropped to the component that receives this event.

Properties

Property	Description	Return Data Type
`value`	Sets the value.	`Long`

Methods

Name	Description	Data Type	Values
`intValue()`	Returns the value in int.	`int`	
`longValue()`	`Returns the value in long.`	`long`	

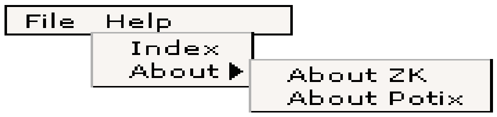
Inherited From

Inherited From
org.zkoss.zul.NumberInputElement
org.zkoss.zul.FormatInputElement
org.zkoss.zul.InputElement

Inherited From
org.zkoss.zul.imp.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Menu

An element, much like a button, that 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`.



```
<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>
```

Class Name

`org.zkoss.zul.Menu`

Supported Child Components

Menupopup

Supported Events

*NONE

Properties

*NONE

Methods

Name	Description	Data Type	Values
`getMenupopup()`	Returns the `Menupopup` it owns, or null if not available.	`Object`	`<null>`
`isTopmost()`	Returns whether this is an top-level menu, i.e., not owning by another `Menupopup`.	`Boolean`	`True`

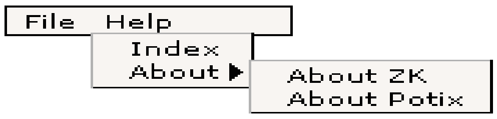
Name	Description	Data Type	Values
	Values: `true | false`		
`getOutAttrs()`			`String`
`insertBefore(org.zkoss.zk.ui.Component child, org.zkoss.zk.ui.Component insertBefore)`			`boolean`

Inherited From

Inherited From
org.zkoss.zul.impl.LabelImageElement
org.zkoss.zul.impl.LabelElement
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Menubar

A container that usually contains menu elements.



```
<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>
```

Class Name

`org.zkoss.zul.Menubar`

Supported Child Components

*NONE

Supported Events

*NONE

Properties

Property	Description	Data Type	Default Values
`orient`	The orient **Values:** `horizontal | vertical`	`String`	`vertical`
`autodrop`	Returns whether to automatically drop down menus if user moves mouse over it. **Values:** `true | false`	`Boolean`	`false`

Methods

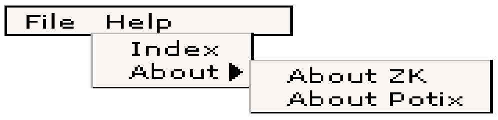
Name	Description	Data Type
`onDrawNewChild(org.zkoss.zk.ui.Component child, java.lang.StringBuffer out)`		`void`
`getOuterAttrs()`		`String`
`insertBefore()`		`Boolean`

Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Menuitem

A single choice in a `Menupopup` element. It acts much like a button but it is rendered on a menu.



```
<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>
```

Class Name

`org.zkoss.zul.Menuitem`

Supported Child Components

`*NONE`

Supported Events

Event Name	Event Type
`onClick`	org.zkoss.zk.ui.event.MouseEvent **Description:** A menu command is associated with a menu item. There are two ways to associate a command to it: the onClick event and the href property. If a event listener is added for a menu item for the onClick event, the listener is invoked when the item is clicked.

Properties

Property	Description	Data Type	Default Value
`value`	The value	`String`	`<empty string>`
`href`	The target frame or window.	`String`	`<null>`
`Target`	The href	`String`	`<null>`
`autocheck`	Whether the menuitem check mark will update each time the menu item is selected **Values**: `true | false`	`Boolean`	`false`
`checked`	Whether it is checked. **Values**: `true | false`	`Boolean`	`false`

Methods

Name	Description	Data Type
`isTopmost()`	Returns whether this is an top-level menu, i.e., not owning by another `Menupopup`. **Values**: true | false	`boolean`
`getOuterAttrs()`		`String`

Inherited From

Inherited From
org.zkoss.zul.impl.LabelImageElement
org.zkoss.zul.impl.LabelElement
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Menupopup

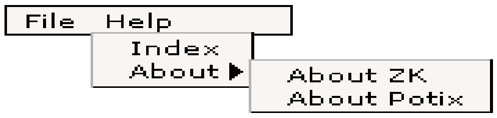
A container used to display menus. It should be placed inside a Menu.

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`.



```
<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>
```

Class Name

`org.zkoss.zul.Menupopup`

Supported Child Components

`Menu`, `Menuitem`, `Menuseparator`

Supported Events

Event Name	Event Type
`onOpen`	org.zkoss.zk.ui.event.OpenEvent **Description:** Denotes user has opened or closed a component.

Properties

*NONE

Methods

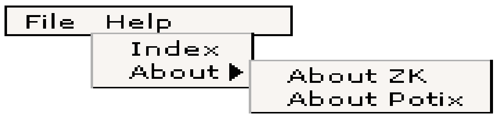
Name	Description	Return Data Type
`getOuterAttrs()`		`String`
`insertBefore(org.zkoss.zk.ui.Component child, org.zkoss.zk.ui.Component insertBefore)`		`boolean`

Inherited From

Inherited From
org.zkoss.zul.impl.LabelImageElement
org.zkoss.zul.impl.LabelElement
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Menuseparator

Used to create a separator between menu items..



```
<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>
```

Class Name

`org.zkoss.zul.Menuseparator`

Supported Child Components

*NONE

Supported Events

*NONE

Properties

*NONE

Methods

Name	Description	Data Type
`isChildable()`	Not childable. **Default**: `false`	`boolean`

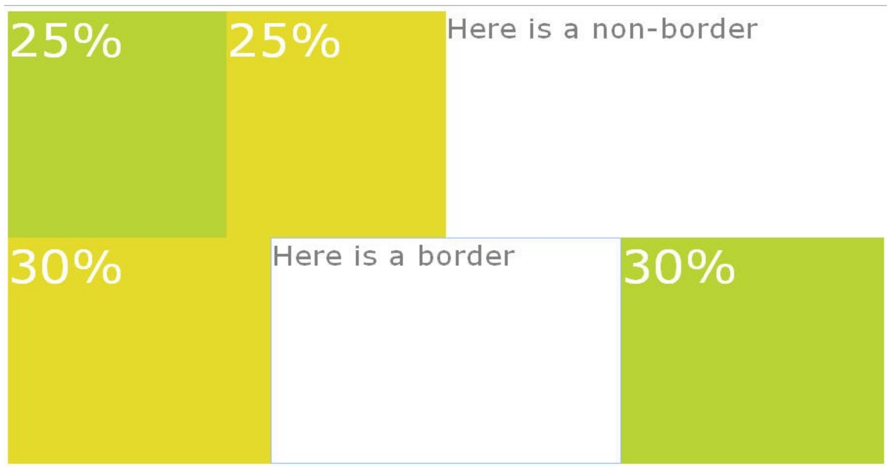
Inherited From

Inherited From
org.zkoss.zul.impl.XulElement

Inherited From
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

North

This component is a north region. The default class of CSS is specified "layout-region-north".



```
<borderlayout height="500px">
   <north size="50%" border="0">
      <borderlayout>
         <west size="25%" border="none" flex="true">
            <div style="background:#B8D335">
               <label value="25%"
                  style="color:white;font-size:50px" />
            </div>
         </west>
         <center border="none" flex="true">
            <div style="background:#E6D92C">
               <label value="25%"
                  style="color:white;font-size:50px" />
            </div>
         </center>
         <east size="50%" border="none" flex="true">
            <label value="Here is a non-border"
               style="color:gray;font-size:30px" />
         </east>
      </borderlayout>
   </north>
   <center border="0">
      <borderlayout>
         <west size="30%" flex="true" border="0">
            <div style="background:#E6D92C">
               <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 size="30%" flex="true" border="0">
            <div style="background:#B8D335">
               <label value="30%"
                  style="color:white;font-size:50px" />
            </div>
         </east>
      </borderlayout>
   </center>
</borderlayout
```

Class Name

`org.zkoss.zkex.zul.North`

Supported Child Components

*NONE

Supported Events

Name	Inherited From
`OnOpen`	org.zkoss.zk.ui.event.OpenEvent **Description:** When a layout is collapsed or opened by a user, the `onOpen` event is sent to the application.

Properties

Property	Description	Data Type	Default Value
`size`	Sets the size of this region.	`java.lang.String`	`null`

Methods

Name	Description	Return Data Type
`getPosition()`	Returns `Borderlayout.NORTH.`	`java.lang.String`
setWidth(java.lang.String width)	The width can't be specified in this component because its width is determined by other region components (West or East).	void

Inherited From

Inherited From
org.zkoss.zkex.zul.LayoutRegion
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Paging

A `paging` component is used to separate long content into multiple pages. For example, assume that you have 100 items and prefer to show 20 items at a time, then you can use the paging components as follows.

Then, when a user clicks on the hyperlinks, the `onPaging` event is sent with an instance of `org.zkoss.zul.event.PagingEvent` to the paging component. To decide which portion of your 100 items are visible, you shall add a listener to the paging component.

```
<vbox>
   <paging totalSize="100" pageSize="20"/>
</vbox>
```

The `listbox` and `grid` component 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 with one paging component.

Class Name

`org.zkoss.zul.Paging`

Supported Child Components

*NONE

Supported Events

Name	Event Type
`onPaging`	org.zkoss.zk.ui.event.PagingEvent **Description:** Notifies one of the pages of a multi-page component is selected by the user.

Properties

Property	Description	Data Type	Default Value
activePage	Sets the active page (starting from 0).	int	0
autohide	Sets whether to automatically hide this component if there is only one page available.	boolean	false
detailed	Sets whether to show the detailed info, such as Paginal.getTotalSize().	boolean	false
pageIncrement	Sets the number of page anchors shall appear at the client.	int	10
pageSize	Sets the number of items per page.	int	20
totalSize	Sets the total number of items.	int	0

Methods

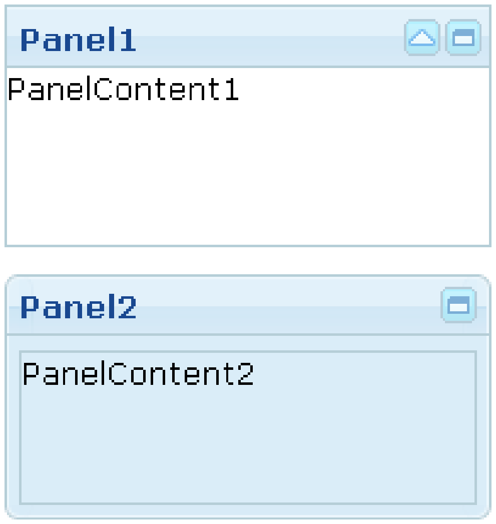
Name	Description	Return Data Type
getPageCount	Returns the number of pages	int
getInnerTags	Returns the inner HTML tags of this component.	String

Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Panel

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.



```
<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>
```

Class Name

`org.zkoss.zul.Panel`

Supported Child Components

Panelchildren

Supported Events

Name	Event Type
`onMove`	**Event:** org.zkoss.zk.ui.event.Event Denotes the close button is pressed by a user, and the component shall detach itself.
`onOpen`	**Event:** org.zkoss.zk.ui.event.OpenEvent 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.
`onMaximize`	**Event:** org.zkoss.zk.ui.event.Maximize Denotes user has maximize a component.
`onMinimize`	**Event:** org.zkoss.zk.ui.event.MinimizeEvent Denotes user has minimize a component.
`onClose`	**Event:** org.zkoss.zk.ui.event.Event Denotes the close button is pressed by a user, and the component shall detach itself.

Properties

Name	Description	Return Data Type
`border`	Sets the border **Values:** none | normal	`java.lang.String`
`closable`	Sets whether to show a close button on the title bar.	`boolean`

Name	Description	Return Data Type
Collapsible	Sets whether to show a toggle button on the title bar.	boolean
Floatable	Sets whether to float the panel to display it inline where it is rendered.	boolean
Framable	Sets whether to render the panel with custom rounded borders.	boolean
maximizable	Sets whether to display the maximizing button and allow the user to maximize the panel, when a panel is maximized, the button will automatically change to a restore button with the appropriate behavior already built-in that will restore the panel to its previous size.	boolean
maximized	Sets whether the panel is maximized, and then the size of the panel will depend on it to show a appropriate size.	boolean
minimized	Sets whether to display the minimizing button and allow the user to minimize the pane	boolean
minimizable	Sets whether the panel is minimized.	boolean
Movable	Sets whether to move the panel to display it inline where it is rendered.	boolean
open	Opens or closes this Panel.	boolean
title	Sets the title.	String
Visible		boolean

Methods

Name	Description	Return Data Type
addMoved(org.zkoss.zk.ui.Component oldparent, org.zkoss.zk.ui.Page oldpg,		void

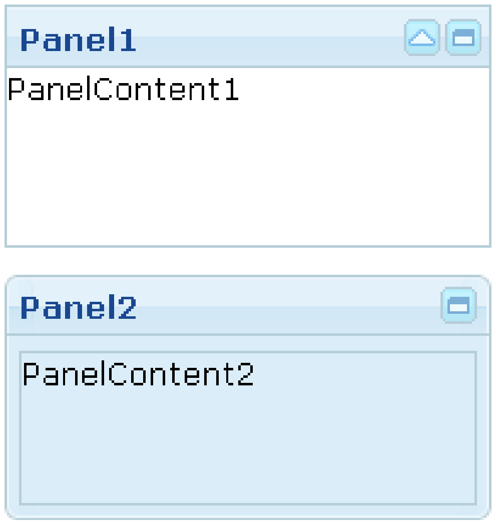
Name	Description	Return Data Type
org.zkoss.zk.ui.Page newpg)		
addToolbar(java.lang.String name, Toolbar toolbar)	Adds the toolbar of the panel by these names, "tbar", "bbar", and "fbar".	boolean
clone()		java.lang.Object
getBottomToolbar()	Returns the bottom toolbar of this panel.	org.zkoss.zul.Toolbar
getCaption()	Returns the caption of this panel.	org.zkoss.zul.Caption
getFootToolbar()		org.zkoss.zul.Toolbar
getMoldSclass()		String
getOuterAttrs()		String
getPanelchildren()	Returns the panelchildren of this panel.	org.zkoss.zul.Panelchildren
getRealSclass()		String
insertBefore(org.zkoss.zk.ui.Component child, org.zkoss.zk.ui.Component insertBefore)		boolean
onChildRemoved(org.zkoss.zk.ui.Component child)		void
onClose()	Process the onClose event sent when the close button is pressed.	void
setParent(org.zkoss.zk.ui.Component parent)		void

Inherited From

Name
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Panelchildren

`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.



```
<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>
```

Class Name

`org.zkoss.zul.Panelchildren`

Supported Child Components

`*ALL`

Supported Events

`*None`

Properties

`*None`

Methods

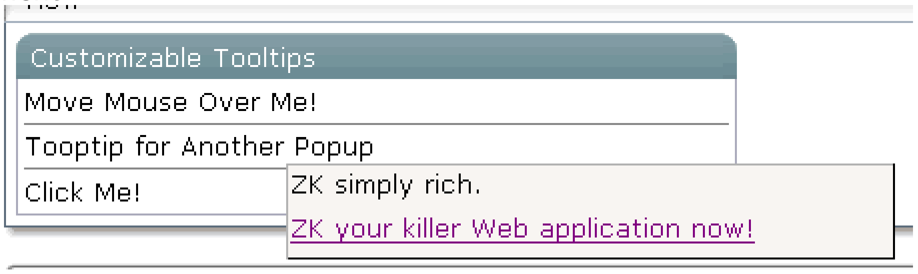
Name	Description	Return Data Type
`getMoldSclass()`		String
`getRealSclass()`	Returns the real style class used for the content block of the panel.	String
`setHeight(java.lang.String height)`	This method is unsupported.	void
`setWidth(java.lang.String width)`	This method is unsupported.	void
`setParent(org.zkoss.zk.ui.Component parent)`		void

Inherited From

Name
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Popup

A container that is displayed as a popup. The popup window does not have any special frame. Popups can be displayed when an element is clicked by assigning the id of the popup to either the XulElement.setPopup(java.lang.String), XulElement.setContext(java.lang.String) or XulElement.setTooltip(java.lang.String) attribute of the element.



```
<label value="Move Mouse Over Me!" tooltip="editPopup"/>
   <separator bar="true"/>
   <label value="Tooptip for Another Popup" tooltip="any"/>
   <separator bar="true"/>
   ...
   <popup id="any" width="300px">
      <vbox>
         ZK simply rich.
         <toolbarbutton label="ZK your killer Web application now!"
href="http://www.zkoss.org"/>
      </vbox>
   </popup>
```

Class Name

org.zkoss.zul.popup

Supported Child Components

*ALL

Supported Events

*NONE

Attributes

*NONE

Methods

Name	Description	Return Data Type
`getOutAttrs()`		`String`
setVisible()	Not allowed	boolean

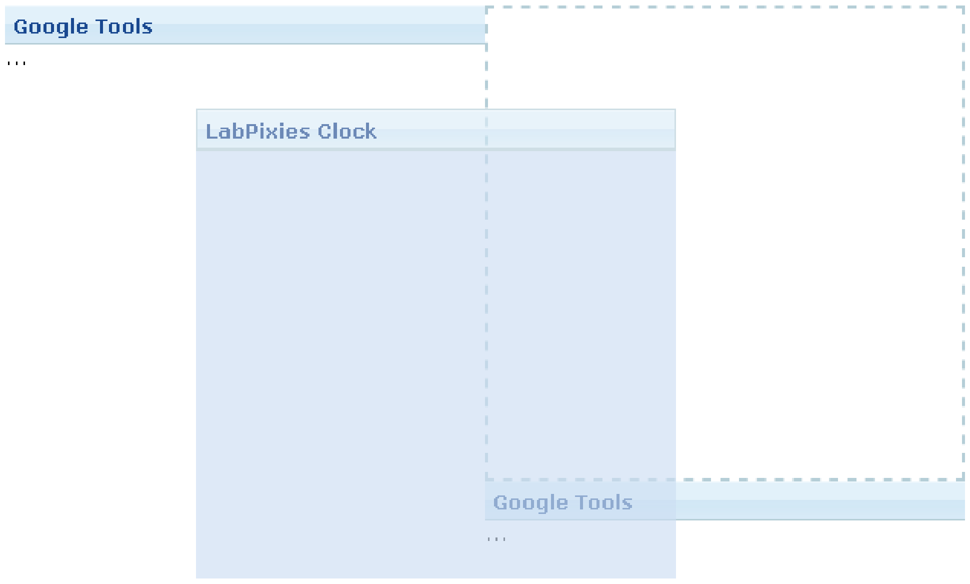
Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Portallayout

A portallayout lays out a container which can have multiple columns, and each column may contain one or more panel. Portallayout provides a way to drag-and-drop panel into other portalchildren from the same portallayout.

Use Portallayout need assign width (either present or pixel) on every Portalchildren, or we cannot make sure about layout look.



```
<portallayout>
   <portalchildren width="30%">
      <panel height="150px" title="Google Tools">
         <panelchildren>...</panelchildren>
      </panel>
      <panel height="300px" title="LabPixies Clock">
         <panelchildren>...</panelchildren>
      </panel>
   </portalchildren>
   <portalchildren width="30%">
      <panel height="150px" title="Google Tools">
         <panelchildren>...</panelchildren>
      </panel>
   </portalchildren>
</portallayout>
```

Class Name

org.zkoss.zkmax.zul.Portallayout

Supported Child Components

Portalchildren

Supported Events

Name	Event Type
onPortalMove	**Event:** `org.zkoss.zkmax.event.PortalMoveEvent` Represents an event caused by a portal being moved.

Properties

*None

Methods

Name	Description	Return Data Type
`getMoldSclass()`		`String`
`getPanel(int col, int row)`	Returns the specified pnael, or null if not available.	`org.zkoss.zul.Panel`
`getPosition(org.zkoss.zul.Panel panel)`	Returns an int array[col, row] that indicates the specified panel located within this portal layout.	`int[]`
`insertBefore(org.zkoss.zk.ui.Component child, org.zkoss.zk.ui.Component insertBefore)`		boolean
`onChildAdded(org.zkoss.zk.ui.Component child)`		`void`
`onChildRemoved(org.zkoss.zk.ui.Component child)`		`void`
`setPanel(org.zkoss.zul.Panel panel, int col, int row)`	Sets the specified panel via the position(col and row).	boolean

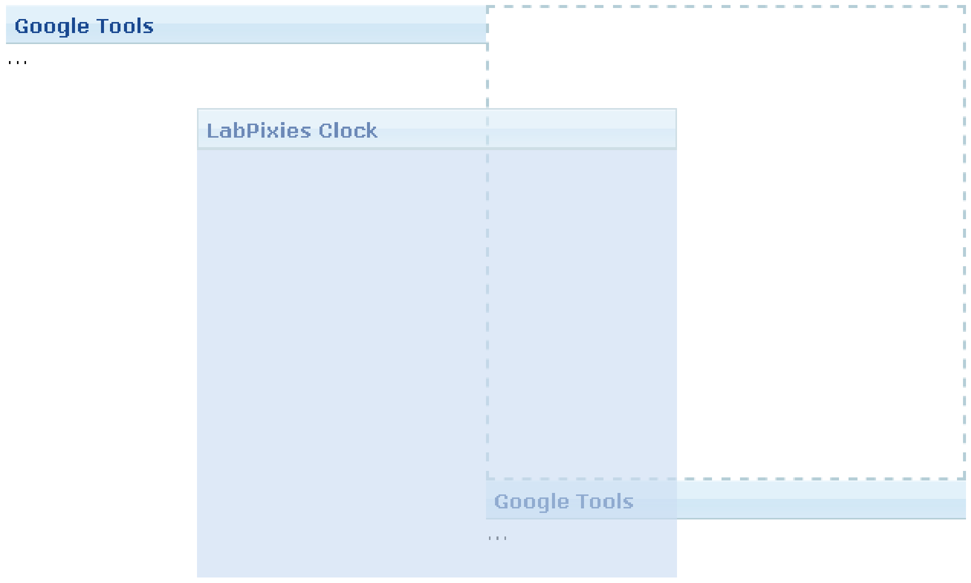
Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Portalchildren

The column of `Portallayout`.

Child of `Portalchildren` can only be `Panel`.



```
<portallayout>
   <portalchildren width="30%">
      <panel height="150px" title="Google Tools">
         <panelchildren>...</panelchildren>
      </panel>
      <panel height="300px" title="LabPixies Clock">
         <panelchildren>...</panelchildren>
      </panel>
   </portalchildren>
   <portalchildren width="30%">
      <panel height="150px" title="Google Tools">
         <panelchildren>...</panelchildren>
      </panel>
   </portalchildren>
</portallayout>
```

Class Name

`org.zkoss.zkmax.zul.Portalchildren`

Supported Child Components

Panel

Supported Events

*None

Properties

*None

Methods

Name	Description	Return Data Type
getMoldSclass()		String
insertBefore(org.zkoss.zk.ui.Component child, org.zkoss.zk.ui.Component insertBefore)		boolean
onChildAdded(org.zkoss.zk.ui.Component child)		void
onChildRemoved(org.zkoss.zk.ui.Component child)		void
setParent(org.zkoss.zk.ui.Component parent)		void

Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Progressmeter

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.

```
<progressmeter value="10"/>
```

Class Name

`org.zkoss.zul.Progressmeter`

Supported Child Components

*NONE

Supported Events

*NONE

Properties

Property	Description	Data Type
`value`	Sets the current value of the progress meter.	`java.lang.String`

Methods

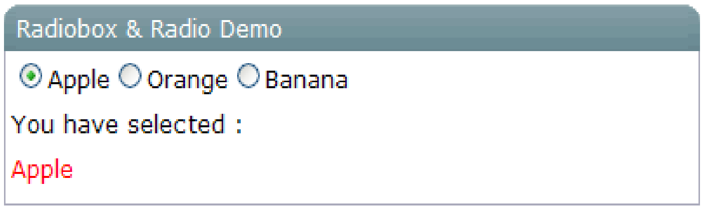
Name	Description	Return Data Type
`getOuterAttrs()`	Returns the grid that this belongs to.	`java.lang.String`
`isChildable()`		`boolean`

Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Radio

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.



You have selected :
Apple

```
<window title="Radiobox &amp; Radio Demo" >
   <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>
```

Class Name

`org.zkoss.zul.Radio`

Supported Child Components

*NONE

Supported Events

Name	Event Type
`onFocus`	org.zkoss.zk.ui.event.Event **Description:** Denotes when a component gets the focus.
`onBlur`	org.zkoss.zk.ui.even.Event Description: Denotes when a component loses the focus.
`onCheck`	org.zkoss.zk.ui.event.CheckEvent Description: Denotes when a component loses the focus.

Properties

Property	Description	Data Type
`value`	The `String` value denote this radio.	`String`
`selected`	The state of this radio.	`boolean`

Methods

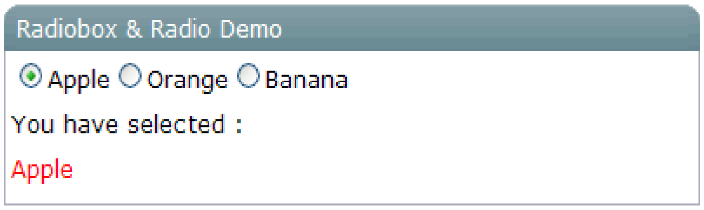
Name	Description	Return Data Type
`IsChildable`	Determines whether it accepts child components **Note:** No child is allowed.	`Boolean`
`getInnerAttrs`	Appends interior attributes for generating the HTML `checkbox` tag (the name, disabled and other attribute).	`java.lang.String`
`getRadiogroup`	Returns `Radiogroup` that this radio button belongs to.	`Radiogroup`
`getName`	Returns the name of this radio button.	`java.lang.String`

Inherited From

Inherited From
org.zkoss.zul.CheckBox
org.zkoss.zul.impl.LabelImageElement
org.zkoss.zul.impl.LabelElement
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Radiogroup

Used to group multiple `radio` buttons. In one `radiogroup`. Only one radio button may be selected at a time.



You have selected :

Apple

```
<window title="Radiobox &amp; Radio Demo" >
   <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.

Class Name

`org.zkoss.zul.Radiogroup`

Supported Child Components

*ALL

Supported Events

*NONE

Properties

Property	Description	Data Type
`value`	The `String` value denote this radio.	`String`
`selectedIndex`	the index of the selected radio button (-1 if no one is selected).	`int`
`selectedItem`	the selected `radio` button.	`org.zkoss.zul.Radio`
`name`	the name of this group of radio buttons.	`String`

Methods

Name	Description	Return Data Type
`getItemCount`	Returns the number of radio buttons in this group.	`int`
`appendItem`	Appends a radio button.	`org.zkoss.zul.Radio`
`removeItemAt`	Removes the child `radio` button in the list box at the given index.	`org.zkoss.zul.Radio`

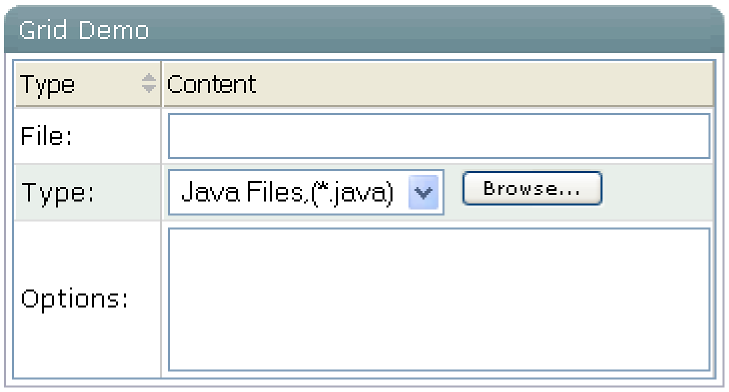
Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Row

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



```
<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="&#36;{asc}" sortDescending="&#36;
{dsc}"/>
         <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>
```

Class Name

org.zkoss.zul.Row

Supported Child Components

*ALL

Supported Events

*NONE

Properties

Property	Description	Data Type	Default Value
align	Sets the horizontal alignment of the whole grid. **Value:** left|center|right	java.lang.String	<null>
nowrap	Sets the nowrap.	boolean	false
sclass	Sets the style class.	java.lang.String	<null>
spans	Sets the spans, which is a list of numbers separated by comma.	java.lang.String	<null>
valign	Sets the vertical alignment of the whole row.	java.lang.String	<null>
value	Sets the value.	java.lang.Object	<null>

Methods

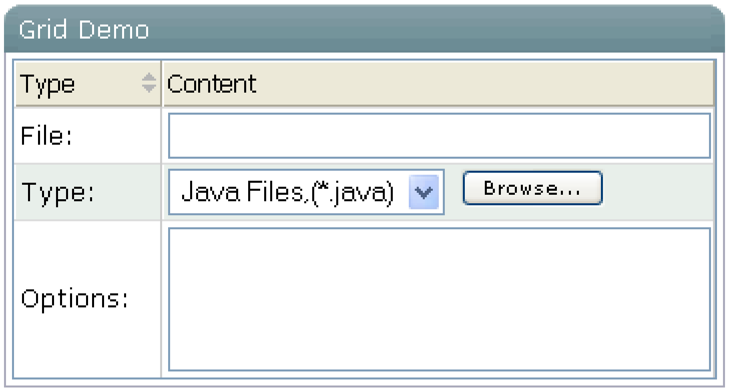
Name	Description	Return Data Type
`getChildAttrs(int)`	Returns the HTML attributes for the child of the specified index.	`java.lang.String`
`getGrid()`	Returns the grid that contains this row.	`org.zkoss.zul.Grid`
`getOuterAttrs()`		`java.lang.String`
`onDrawNewChild(org.zkoss.zk.ui.Component, java.lang.StringBuffer)`		`void`
`setParent(org.zkoss.zk.ui.Component)`		`void`
`setStyle(java.lang.String)`		`void`

Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Rows

Defines the `rows` of a grid. Each child of a `rows` element should be a `org.zkoss.zul.Row` element.



```
<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="&#36;{asc}" sortDescending="&#36;
{dsc}"/>
         <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>
```

Class Name

org.zkoss.zul.Rows

Supported Child Components

Row

Supported Events

*NONE

Properties

*NONE

Methods

Name	Description	Return Data Type
getGrid()	Returns the grid that contains this rows.	org.zkoss.zul.Grid
getVisibleBegin()	Returns the index of the first visible child.	int
getVisibleEnd()	Returns the index of the last visible child.	int
insertBefore(org.zkoss.zk.ui.Component, org.zkoss.zk.ui.Component)		boolean
onChildAdded(org.zkoss.zk.ui.Component)		void
onChildRemoved(org.zkoss.zk.ui.Component)		void
setParent(org.zkoss.zk.ui.Component)		void

Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Script

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.

```
<zk>
<window id="win" >
   <button label="change color" onClick='Clients.evalJavaScript("myfunc()")'/>
</window>
<script type="text/javascript">
function myfunc() {
   $e("${win.uuid}").style.backgroundColor = "blue";
}
</script>
</zk>
```

Class Name

`org.zkoss.zul.Script`

Supported Child Components

*NONE

Supported Events

*NONE

Properties

Property	Description	Data Type
`charset`	Sets whether the button (on the right of the textbox) is visible.	`java.lang.String`
`content`	Sets the content of the script element.	`java.lang.String`
`defer`	Sets whether to defer the execution of the script codes.	`boolean`
`src`	Sets the URI of the source that contains the script codes.	`java.lang.String`
`type`	Sets the type of this client script.	`java.lang.String`

Methods

Name	Description	Return Data Type
isChildable()	Not childable. **Value:** false	boolean
redraw(java.io.Writer out)		

Inherited From

Inherited From
org.zkoss.zk.ui.AbstractComponent

Slider

A slider with slid and knob

A slider is used to let user specifying a value by scrolling.

A slider accepts a range of value starting from 0 to 100. You could change the maximal allowed value by the `maxpos` property.

```
<slider id="slider" />
<slider curpos="1" maxpos="20" />
```

Class Name

`org.zkoss.zul.Slider`

Supported Child Components

*NONE

Supported Events

Name	Inherited From
`OnScroll`	org.zkoss.zk.ui.event.ScrollEvent **Description:** Denotes the content of a scrollable component has been scrolled by the user.
`OnScrolling`	org.zkoss.zk.ui.event.ScrollEvent **Description:** Denotes that 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 `getPos` method in the `ScrollEvent` class to retrieve the temporary position.

Attributes

Property	Description	Data Type	Default Values
`curpos`	the current position of the slider **Values:** 0 to maxpos	`int`	`0`
`maxpos`	the maximum position of the slider.	`int`	`100`
`PageIncrement`	the amount that the value of curpos changes by when the tray of the scroll bar is clicked	`int`	`10`

Methods

Name	Description	Return Data Type
`GetOuterAttrs()`		`String`

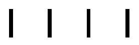
Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Space

Space is a `Separator` with the orient default to "horizontal".

In other words, <space> is equivalent to `<separator orient="horizontal">`



```
<zk>
   <space bar="true"/>
   <space bar="true"/>
   <space bar="true"/>
   <space bar="true"/>
</zk>
```

Class Name

`org.zkoss.zul.Space`

Supported Child Components

*NONE

Supported Events

*NONE

Properties

*NONE

Methods

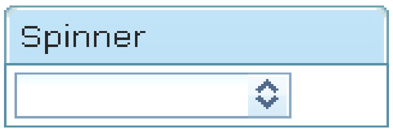
*NONE

Inherited From

Inherited From
org.zkoss.zul.Separator
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Spinner

An edit box for holding a constrained integer.



```
<window title="Spinner" border="normal" width="150px">
   <spinner />
</window>
```

Class Name

`org.zkoss.zul.Spinner`

Supported Child Components

*NONE

Supported Events

*NONE

Properties

Property	Description	Return Data Type
`step`	Set the step of spinner	`integer`
`value`	Sets the text value.	`integer`
`ButtonVisible`	Sets whether the button (on the right of the textbox) is visible.	`boolean`

Methods

Name	Description	Return Data Type
`coerceFromString(java.lang.String value)`	Coerces the value passed to InputElement.setText(java.lang.String).	`java.lang.Object`
`coerceToString(java`	Coerces the value passed to InputElement.setText(java.lang.Stri	`java.lang.Object`

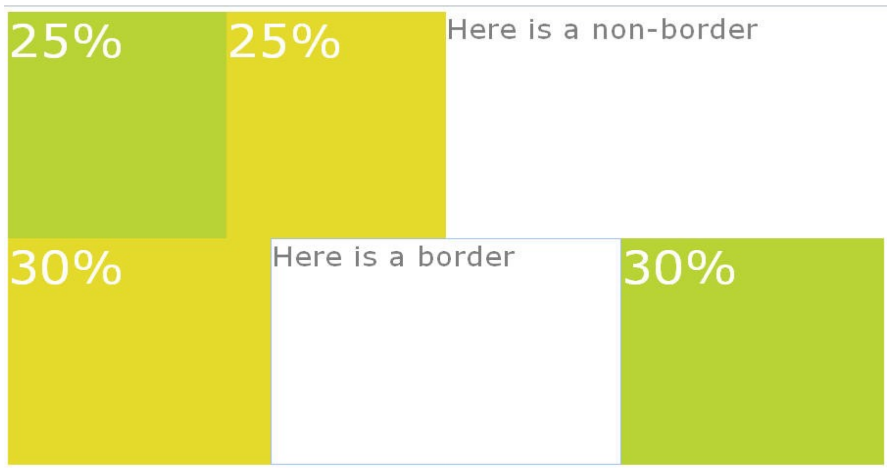
Name	Description	Return Data Type
.lang.Object value)	ng).	
getMoldSclass()		String
setConstraint(java.lang.String constr)		void

Inherited From

Inherited From
org.zkoss.zul.NumberInputElement
org.zkoss.zul.FormatInputElement
org.zkoss.zul.InputElement
org.zkoss.zul.imp.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

South

This component is a south region. The default class of CSS is specified "layout-region-south".



```
<borderlayout height="500px">
   <north size="50%" border="0">
      <borderlayout>
         <west size="25%" border="none" flex="true">
            <div style="background:#B8D335">
               <label value="25%"
                  style="color:white;font-size:50px" />
            </div>
         </west>
         <center border="none" flex="true">
            <div style="background:#E6D92C">
               <label value="25%"
                  style="color:white;font-size:50px" />
            </div>
         </center>
         <east size="50%" border="none" flex="true">
            <label value="Here is a non-border"
               style="color:gray;font-size:30px" />
         </east>
      </borderlayout>
   </north>
   <center border="0">
      <borderlayout>
         <west size="30%" flex="true" border="0">
            <div style="background:#E6D92C">
               <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 size="30%" flex="true" border="0">
            <div style="background:#B8D335">
               <label value="30%"
                  style="color:white;font-size:50px" />
            </div>
         </east>
      </borderlayout>
   </center>
</borderlayout>
```

Class Name

`org.zkoss.zkex.zul.south`

Supported Child Components

*NONE

Supported Events

Name	Inherited From
`OnOpen`	org.zkoss.zk.ui.event.OpenEvent **Description:** When a layout is collapsed or opened by a user, the `onOpen` event is sent to the application.

Properties

Property	Description	Data Type	Default Value
`size`	Sets the size of this region.	`java.lang.String`	`null`

Methods

Name	Description	Return Data Type
`getPosition()`	Returns `Borderlayout.NORTH.`	`java.lang.String`
setWidth(java.lang.String width)	The width can't be specified in this component because its width is determined by other region components (West or East).	void

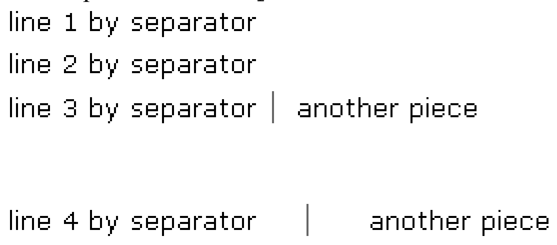
Inherited From

Inherited From
org.zkoss.zkex.zul.LayoutRegion
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Separator

Space is a `Separator` with the orient default to "`horizontal`".

In other words, `<space>` is equivalent to `<separator orient="horizontal">`


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
```

Class Name

`org.zkoss.zul.Space`

Supported Child Components

*NONE

Supported Events

*NONE

Attributes

Property	Description	Data Type	Default Values
spacing	the spacing. **Values**: the spacing (such as "0", "5px", "3pt" or "1em")	String	<null>
orient	the orient. **Values**:horizontal|vertical	String	horizontal
bar	whether to display a visual bar as the separator. **Values**:true/false	boolean	false

Methods

*NONE

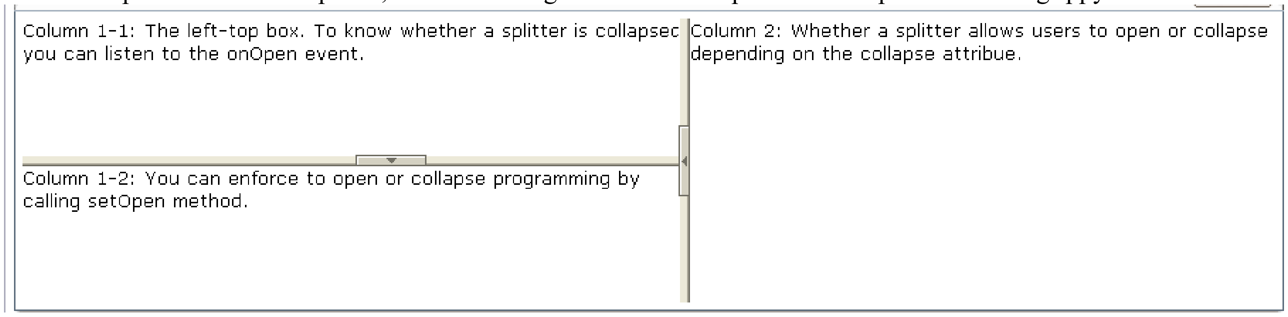
Inherited From

Inherited From
org.zkoss.zul.Separator
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Splitter

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 in placed inside the splitter, and one sibling element of the splitter is collapsed when the grippy is clicked.



Column 1-1: The left-top box. To know whether a splitter is collapsec you can listen to the onOpen event.

Column 1-2: You can enforce to open or collapse programming by calling setOpen method.

Column 2: Whether a splitter allows users to open or collapse depending on the collapse attribue.

```
<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 attribue.
</hbox>
```

Class Name

`org.zkoss.zul.Splitter`

Supported Child Components

*NONE

Supported Events

Name	Inherited From
`OnOpen`	org.zkoss.zk.ui.event.OpenEvent **Description:** When a splitter is collapsed or opened by a user, the `onOpen` event is sent to the application.

Attributes

Property	Description	Data Type	Default Values
`collapse`	Returns which side of the splitter is collapsed when its grippy is clicked. If this attribute is not specified, the splitter will not cause a collapse. **Values:** `None,before,after`	`String`	`none`
`orient`	the maximum position of the slider. **Values:** `horizontal | vertical`	`String`	`vertical`
`open`	the amount that the value of curpos changes by when the tray of the scroll bar is clicked **Values:** `true | false`	`boolean`	`true`

Methods

*NONE

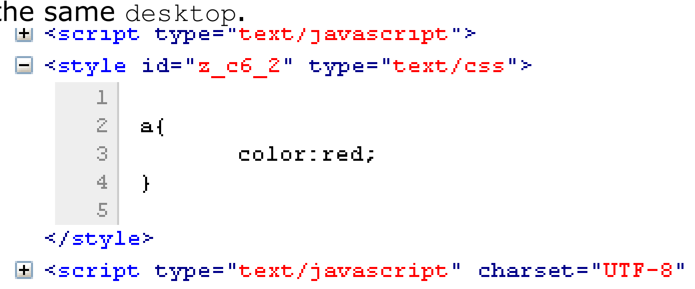
Inherited From

Inherited From
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Style

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.



```
<script type="text/javascript">
<style id="z_c6_2" type="text/css">

a{
        color:red;
}

</style>
<script type="text/javascript" charset="UTF-8"
```

```
<style>
   a{
      color:red;
   }
</style>
```

Class Name

org.zkoss.zul.Style

Supported Child Components

*NONE

Supported Events

*NONE

Attributes

Property	Description	Data Type	Default Values
src	the URI of an external style sheet.	String	<empty string>

Methods

Name	Description	Return Data Type
onChildRemoved(org.zkoss.zk.ui.Component child)		void
redraw(java.io.Writer out)		void
insertBefore(org.zkoss.zk.ui.Co	Only Label children	boolean

Name	Description	Return Data Type
`mponent child, org.zkoss.zk.ui.Component insertBefore)`	are allowed.	

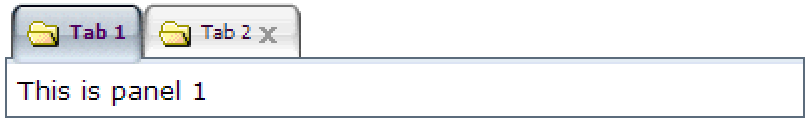
Inherited From

Inherited From
org.zkoss.zk.ui.AbstractComponent

Tab

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.

By setting the `closable` property to true, a close button is shown for the tab, such that user could close the tab and the corresponding tab panel by clicking the button. Once 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.



```
<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>
```

Class Name

`org.zkoss.zul.Tab`

Supported Child Components

*NONE

Supported Events

Event Name	Event Type
`onClick`	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has clicked the component.
`onRightClick`	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has right-clicked the component.

Event Name	Event Type
onDoubleClick	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has double-clicked the component.
onSelect	org.zkoss.zk.ui.event.SelectEvent **Description:** Denotes user has selected a tab. onSelect is sent to both tab and tabbox.
onClose	org.zkoss.ui.zk.ui.event.Event **Description:** Denotes the close button is pressed by a user, and the component shall detach itself.

Properties

Property	Description	Data Type	Default Value
closable	Sets whether this tab is closable. If closable, a button is displayed and the `onClose` event is sent if an user clicks the button. Values : `true|false`	boolean	false
selected	Sets whether this tab is selected. Values : `true|false`	boolean	

Methods

Name	Description	Data Type	Values
getIndex	Returns the index of this panel, or -1 if it doesn't belong to any tabs.	int	
getLinkedPanel	Returns the panel associated with this tab.	Tabpanel	
getTabbox	Returns the tabbox owns this component.	Tabbox	
isChildable	No child is allowed.	boolean	false

Inherited From

Inherited From
org.zkoss.zul.impl.LabelImageElement
org.zkoss.zul.impl.LabelElement
org.zkoss.zul.imp.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

TableChildren

The cell of Tablelayout. The child component of Tablechildren only can be Panel.

table1
Panel
table2
Panel

```
<?xml version="1.0" encoding="UTF-8"?>

<?component name="panel" extends="panel" width="200px" height="200px" ?>

<zk>
<label value="3 columns, 1 colspan, 1 rowspan"/>

<tablelayout id="tbl" columns="3">
      <tablechildren id="tc1" colspan="2">
         <panel title="table1" border="normal"
            maximizable="true" collapsible="true" >
            <panelchildren>Panel</panelchildren>
          </panel>
      </tablechildren>
      <tablechildren>
         <panel title="table2" border="normal"
            maximizable="true" collapsible="true">
            <panelchildren>Panel</panelchildren>
          </panel>
      </tablechildren>
</tablelayout>
</zk>
```

Class Name

`org.zkoss.zkex.zul.Tablelayout`

Supported Child Components

*None

Supported Events

*None

Properties

*None

Methods

Name	Description	Return Data Type

Inherited From

Inherited From
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

TableLayout

Tablelayout lay outs a container as an HTML table whose columns can be specified, and rowspan and colspan of its child can also be specified to create complex layouts within the table.

table1
Panel
table2
Panel

```
<?xml version="1.0" encoding="UTF-8"?>

<?component name="panel" extends="panel" width="200px" height="200px" ?>

<zk>
<label value="3 columns, 1 colspan, 1 rowspan"/>

<tablelayout id="tbl" columns="3">
      <tablechildren id="tc1" colspan="2">
         <panel title="table1" border="normal"
            maximizable="true" collapsible="true" >
            <panelchildren>Panel</panelchildren>
          </panel>
      </tablechildren>
      <tablechildren>
         <panel title="table2" border="normal"
            maximizable="true" collapsible="true">
            <panelchildren>Panel</panelchildren>
          </panel>
      </tablechildren>
</tablelayout>
</zk>
```

Class Name

`org.zkoss.zkex.zul.Tablelayout`

Supported Child Components

Tablechildren

Supported Events

*None

Properties

*None

Methods

Name	Description	Return Data Type
`getColumns()`	Returns number of rows to span this header.	`int`
`insertBefore(org.zkoss.zk.ui.Component child, org.zkoss.zk.ui.Component insertBefore)`		

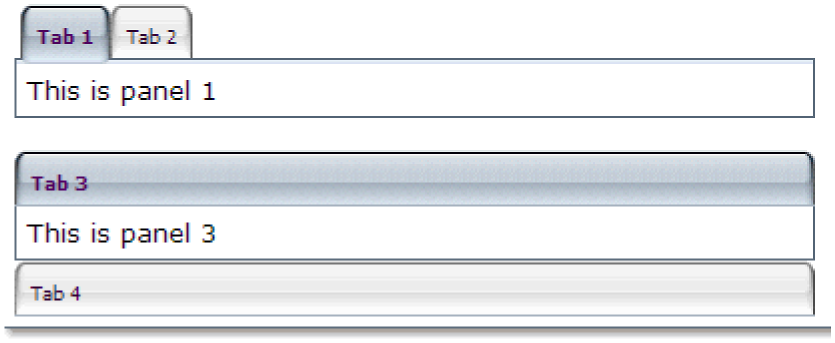
Inherited From

Inherited From
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Tabbox

A `tabbox` that contains the `tabs` and `tabpanels` allows developers to separate a large number of components into several groups(each group contains a `tab` and a `tabpanel)`, and show one group each time, such that the user interface won't be too complicate to read. There is only one group (aka., a `tabpanel)` is visible at the same time. Once the `tab` of an invisible group is clicked, it becomes visible and the previous visible group becomes invisible.

The currently selected tab component is given an additional `selected` property which is set to true. This is used to give the currently selected tab a different appearance so that it will look selected. Only one tab will have a true value for this property at a time. If none of tabs are selected, the first one is selected automatically.



```
<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>
```

Class Name

`org.zkoss.zul.Tabbox`

Supported Child Components

Tabs, Tabpanels

Supported Events

Event Name	Event Type
`onRightClick`	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has right-clicked the component.
`onSelect`	org.zkoss.zk.ui.event.SelectEvent **Description:** Denotes user has selected a tab. onSelect is sent to both tab and tabbox.

Properties

Property	Description	Data Type	Default Value
`orient`	Sets the orient. Values : `horizontal|vertical`	`String`	`horizontal`
`panelSpacing`	Sets the spacing between Tabpanel. This is used by certain molds, such as accordion.	`String`	`<null>`
`selectedIndex`	`Sets the selected index.`	`int`	`0`
`selectedPanel`	`Sets the selected tabpanel`	`Tabpanel`	`<null>`
`selectedTab`	`Sets the selected tab`	`Tab`	`<null>`

Methods

Name	Description	Data Type	Values
`getTabLook`	Returns the look of the `Tab` and `Tabbox.`	`String`	
`getTabpanels`	Returns the tabpanels that this	`Tabpanels`	

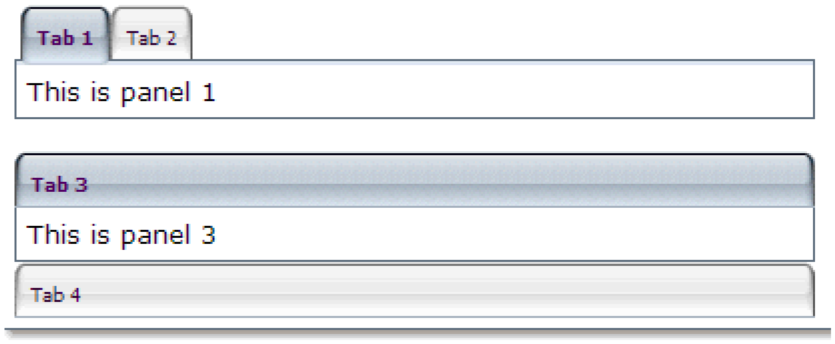
Name	Description	Data Type	Values
	tabbox owns.		
`getTabs`	Returns the tabs that this tabbox owns.	`Tabs`	

Inherited From

Inherited From
org.zkoss.zul.imp.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Tabpanel

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.



```
<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>
```

Class Name

`org.zkoss.zul.Tabpanel`

Supported Child Components

*ALL

Supported Events

Event Name	Event Type
`onClick`	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has clicked the component.
`onRightClick`	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has right-clicked the component.
`onDoubleClick`	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has double-clicked the component.

Properties

*NONE

Methods

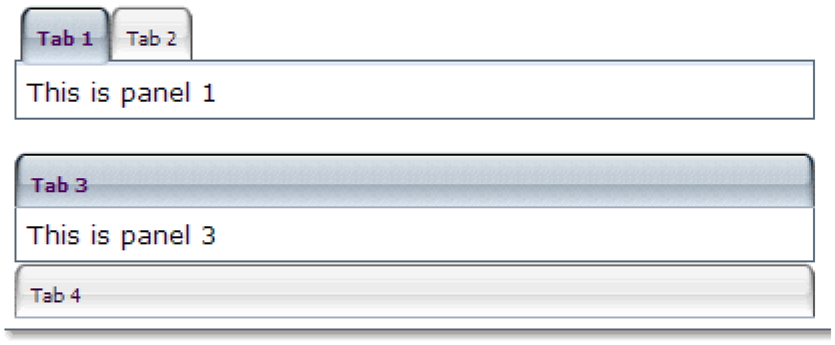
Name	Description	Data Type	Values
`getIndex`	Returns the index of this panel, or -1 if it doesn't belong to any tabpanels.	`int`	
`getLinkedTab`	Returns the tab associated with this tab panel.	`Tab`	
`getTabbox`	Returns the tabbox owns this component.	`Tabbox`	
`isSelected`	Returns whether this tab panel is selected.	`boolean`	

Inherited From

Inherited From
org.zkoss.zul.imp.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Tabpanels

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



```
<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>
```

Class Name

`org.zkoss.zul.Treepanels`

Supported Child Components

Tabpanel

Supported Events

*NONE

Properties

*None

Methods

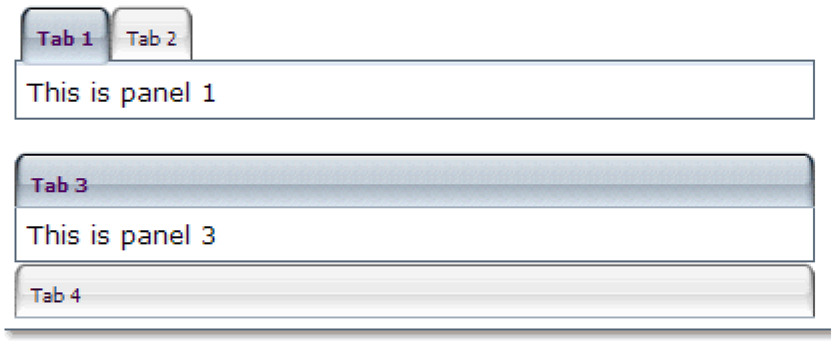
Name	Description	Data Type	Values
`getTabbox`	Returns the `tabbox` owns this component.	`Tabbox`	

Inherited From

Inherited From
org.zkoss.zul.imp.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Tabs

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



```
<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>
```

Class Name

`org.zkoss.zul.Tabs`

Supported Child Components

Tab

Supported Events

*NONE

Properties

*None

Methods

Name	Description	Data Type	Values
`getTabbox`	Returns the `tabbox` owns this component.	`Tabbox`	

Inherited From

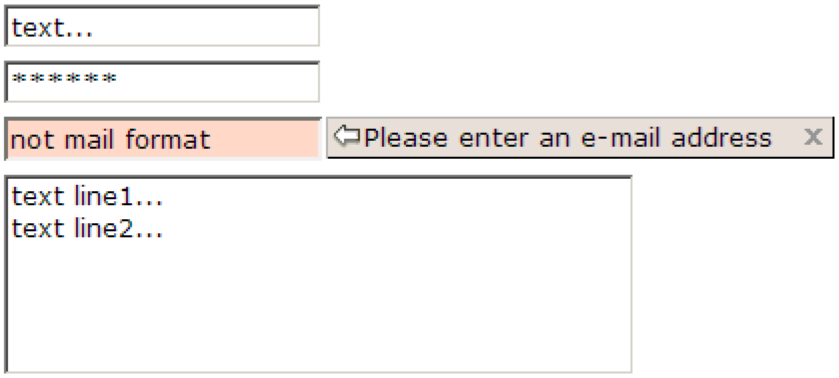
Inherited From
org.zkoss.zul.imp.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Textbox

A textbox is used to let users input text data.

You colud assign value, type, constraint, rows, cols to a textbox by the corresponding properties. When you assigns the property type to a string value "password" when multiline is false(multiline will be true if You set rows large then 1 or sets multiline to true directly) then any character in this component will replace by '*'.

You colud 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, will cause a validating preocess to valdate the value. If valdation fail, then a notification will poped up.



```
<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>
```

Class Name

org.zkoss.zul.Textbox

Supported Child Components

*NONE

Supported Events

Event Name	Event Type

Event Name	Event Type
`onChanging`	org.zkoss.zk.ui.event.InputEvent **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 `getValue` method in the `InputEvent` class to retrieve the temporary value.
`onSelection`	org.zkoss.zk.ui.event.SelectionEvent **Description:** 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 `getStart` and `getEnd` methods.
`onFocus`	org.zkoss.zk.ui.event.Event **Description:** 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`	org.zkoss.zk.ui.event.Event **Description:** 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.
`onCreate`	org.zkoss.ui.zk.ui.event.CreateEvent **Description:** Denotes a component is created when rendering a ZUML page.
`onDrop`	org.zkoss.ui.zk.ui.event.DropEvent **Description:** Denotes another component is dropped to the component that receives this event.

Properties

Property	Description	Data Type	Default Value
`multiline`	Sets whether it is multiline. Values: `true | false` Note: If `rows > 1`, multiline will always return `true`	`boolean`	`false`
rows	Sets the rows.	`int`	1
`type`	Sets the type. Values : `text | password`	`String`	text
`value`	Sets the text value.	`String`	<empty string>

Methods

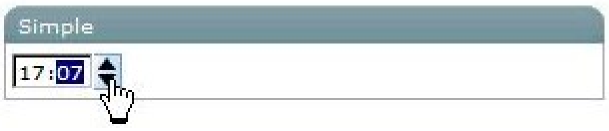
*NONE

Inherited From

Inherited From
org.zkoss.zul.InputElement
org.zkoss.zul.imp.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Timebox

An edit box for holding a time (a java.util.Date Object , but only Hour & Minute are used.



```
<window title="Simple" width="300px" border="normal">
   <timebox id="tb0"/>
</window>
```

Class Name

`org.zkoss.zul.Timebox`

Supported Child Components

*NONE

Supported Events

*NONE

Properties

Property	Description	Data Type
`buttonVisible`	Sets whether the button (on the right of the textbox) is visible.	`boolean`
`image`	Sets the URI of the button image.	`java.lang.String`
`timeZone`	Sets the time zone that this time box belongs to, or null if the default time zone is used.	`java.util.TimeZone`
`value`	Sets the value (in `Date`).	`java.util.Date`

Methods

Name	Description	Return Data Type
getInnerAttrs()	Generates the Client-Side-Action	`java.lang.String`

Name	Description	Return Data Type
	attributes to the interior tag.	

Inherited From

Inherited From
org.zkoss.zul.impl.InputElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Timer

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.

```
<label id="now"/>
<timer id="timer" delay="1000" repeats="true"
   onTimer="now.setValue(new Date().toString())"/>
```

Class Name

`org.zkoss.zul.Timer`

Supported Child Components

*NONE

Supported Events

Event Name	Event Type
`onTimer`	org.zkoss.zk.ui.event.Event **Description:** Denotes the timer you specified has triggered an event. To know which timer, invoke the `getTarget` method in the `Event` class.

Properties

Property	Description	Data Type	Default Value
`delay`	Sets the delay, the number of milliseconds between successive action events. Note : 0 means immediately	`int`	0
`repeats`	Sets whether the timer shall send Event repeatedly. Values : `true|false`	`boolean`	`false`
`running`	Start or stops the timer. Values : `true|false`	`boolean`	`true`

Methods

Name	Description	Data Type	Values
`start`	Starts the timer.	`void`	
`stop`	Stops the timer.	`void`	

Inherited From

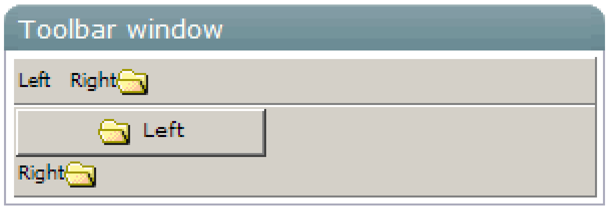
Inherited From
org.zkoss.zul.imp.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Toolbar

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, 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 : org.zkoss.zul.Button, org.zkoss.zul.Toolbarbutton



```
<window title="Toolbar window" border="normal" width="300px">
   <toolbar>
      <toolbarbutton label="Left" /><space/>
      <toolbarbutton label="Right" image="/img/folder.gif" dir="reverse"/>
   </toolbar>
   <toolbar orient="vertical">
      <button label="Left" image="/img/folder.gif" width="125px"/>
      <toolbarbutton label="Right" image="/img/folder.gif" dir="reverse"/>
   </toolbar>
</window>
```

Class Name

`org.zkoss.zul.Toolbar`

Supported Child Components

*ALL

Supported Events

Event Name	Event Type
`onClick`	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has clicked the component.
`onRightClick`	org.zkoss.zk.ui.event.MouseEvent **Description:**

Event Name	Event Type
	Denotes user has right-clicked the component.
onDoubleClick	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has double-clicked the component.

Properties

Property	Description	Data Type	Default Value
orient	Sets the orient. Values : horizontal|vertical	String	horizontal

Methods

*NONE

Inherited From

Inherited From
org.zkoss.zul.imp.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

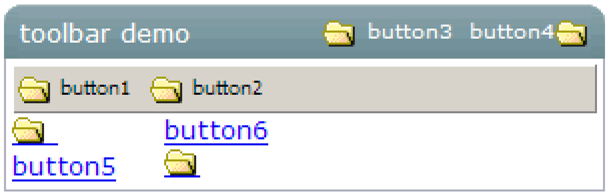
Toolbarbutton

The behave of `Toolbarbutton` is similar to the `button` except the appearance is different. The `button` component uses HTML BUTTON tag, while `thetoolbarbutton` component uses HTML A tag.

A `toolbarbutton` could be placed outside a `toolbar`, However `toolbarbutton`s 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.

See also : org.zkoss.zul.Button, org.zkoss.zul.Toolbar



```
<window title="toolbar demo" border="normal" width="300px">
   <caption>
      <toolbarbutton label="button3" image="/img/folder.gif"/><space/>
      <toolbarbutton label="button4" image="/img/folder.gif" dir="reverse" />
   </caption>
   <toolbar>
      <toolbarbutton label="button1" image="/img/folder.gif" /><space/>
      <toolbarbutton label="button2" image="/img/folder.gif" />
   </toolbar>
   <hbox>
   <toolbarbutton label="button5" image="/img/folder.gif" orient="vertical"/><space/
>
   <toolbarbutton label="button6" image="/img/folder.gif" orient="vertical"
dir="reverse"/>
   </hbox>
</window>
```

Class Name

`org.zkoss.zul.Toolbarbutton`

Supported Child Components

*NONE

Supported Events

Event Name	Event Type
`onClick`	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has clicked the component.
`onRightClick`	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has right-clicked the component.

Properties

Property	Description	Data Type	Default Value
`dir`	Sets the direction of image, if normal then text first, otherwise image first. Check Button to know more. **Values** : `normal|reverse`	`String`	`normal`
`href`	Sets the href, If `null`, the button has no function unless you specify the `onClick` handler. If has value, button will render as a HTML A tag.	`String`	`<null>`
`orient`	Sets the orient. Check Button to know more. **Values** : `horizontal|vertical`	`String`	`horizontal`
`target`	Sets the target frame or window, this attribute works when `href` not null	`String`	`<null>`

Methods

Name	Description	Data Type	Values
`isChildable`	Check Is this component allow children.	`boolean`	`false`

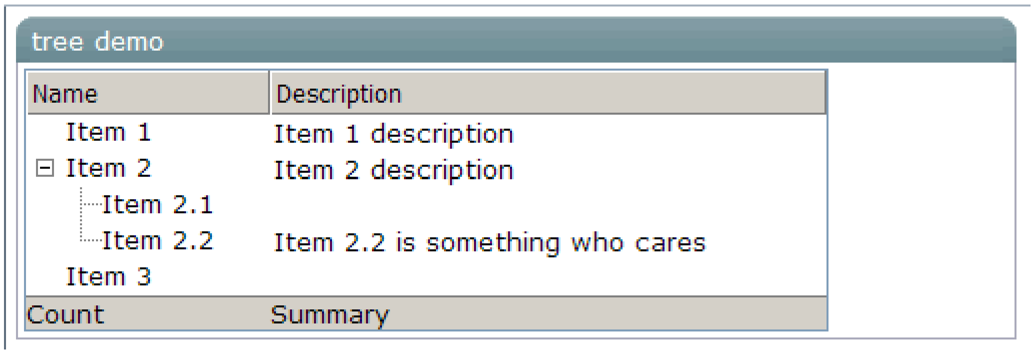
Inherited From

Inherited From
org.zkoss.zul.impl.LabelImageElement
org.zkoss.zul.impl.LabelElement

Inherited From
org.zkoss.zul.imp.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Tree

A tree consists of tree 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.



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">
   <tree id="tree" width="400px" rows="5">
      <treecols sizable="true">
         <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>
               </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>
        <treefoot>
           <treefooter label="Count"/>
           <treefooter label="Summary"/>
        </treefoot>
     </tree>
</window>
```

Class Name

`org.zkoss.zul.Tree`

Supported Child Components

Treecols, Treechildren, Treefoot

Supported Events

Event Name	Event Type
`onSelect`	org.zkoss.zk.ui.event.SelectEvent **Description:** Denotes user has selected one or multiple child components(a set of treeitem).

Properties

Property	Description	Data Type	Default Value
`checkmark`	Sets whether the check mark shall be displayed in front of each item. The check mark is a `checkbox` if `isMultiple()` returns true. It is a `radio` button if `isMultiple()` returns false. Values : `true|false`	`boolean`	`false`

Property	Description	Data Type	Default Value
model	Sets the tree model associated with this tree.	org.zkoss.zul.TreeModel	<null>
multiple	Sets whether multiple selections are allowed. Values : true|false	boolean	false
name	Sets the name of this component. The name is used only to work with "legacy" Web application that handles user's request by servlets. It works only with HTTP/HTML-based browsers. It doesn't work with other kind of clients. Don't use this method if your application is purely based on ZK's event-driven model.	String	<null>
pageSize	Sets the page size that is used by all Treechildren to display a portion of their child Treeitem, or -1 if no limitation.	int	10
rows	Sets the rows. Zero means no limitation.	int	0
selectedItem	Deselects all of the currently selected items and selects the given item.	org.zkoss.zul.Treeitem	<null>
seltype	Sets the seltype. Currently, only "single" is supported. Values : single	String	<single>
treeitemRender	Sets the renderer which is used to render each item if getModel() is not null. Note: changing a render will not cause the tree to re-render. If you want it to re-render, you could assign the same model again (i.e., setModel(getModel())), or fire an TreeDataEvent event.	org.zkoss.zul.TreeitemRenderer	<null>
vflex	Sets whether to grow and shrink vertical to fit their given space, so	boolean	false

Property	Description	Data Type	Default Value
	called vertial flexibility. Note: this attribute is ignored if `setRows(int)` is specified Values : `true|false`		

Methods

Name	Description	Data Type	Values
`addItemToSelection`	Selects the given item, without deselecting any other items that are already selected.	`Treeitem`	
`clear`	Clears all child tree items Note: after clear, `getTreechildren()` won't be null, but it has no child		
`clearSelection`	Clears the selection.		
`getItemCount`	Returns the number of child `Treeitem`	`int`	
`getItems`	Returns a readonly list of all descending `Treeitem` (children's children and so on) ,	`Collection`	
`getSelectedCount`	Returns the number of items being selected.	`int`	
`getSelectedItems`	Returns all selected items.	`Set`	
`getTreechildren`	Returns the treechildren that this tree owns (might null).	`Treechildren`	
`getTreecols`	Returns the treecols that this tree owns (might null).	`Treecols`	
`getTreefoot`	Returns the treefoot that this tree owns (might null).	`Treefoot`	
`removeItemFromSelection`	Deselects the given item without deselecting other items.	`Treeitem`	
`selectAll`	Selects all items.		
`selectItem`	Deselects all of the currently selected items and selects the given item.	`Treeitem`	
`toggleItemSelection`	If the specified item is selected, it is	`Treeitem`	

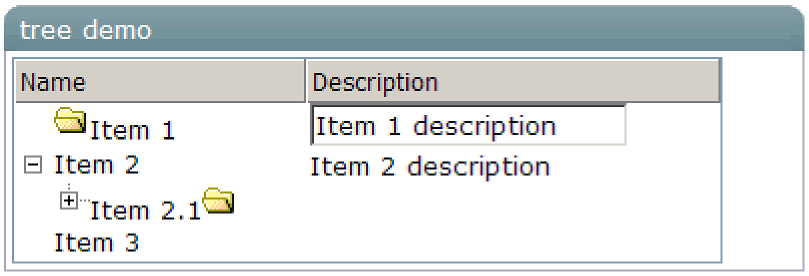
Name	Description	Data Type	Values
	deselected.		

Inherited From

Inherited From
org.zkoss.zul.imp.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Treecell

`Treecell` represent one column in a `treerow` by sequencial. `Treecell` can contains any component in it, such as label, image, textbox etc.



```
<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>
```

Class Name

`org.zkoss.zul.Treecell`

Supported Child Components

*ALL

Supported Events

Event Name	Event Type
`onClick`	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has clicked the component.
`onRightClick`	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has right-clicked the component.
`onDoubleClick`	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has double-clicked the component.

Properties

Property	Description	Data Type	Default Value
`span`	Sets the number of columns to span this cell. It is the same as the colspan attribute of HTML TD tag.	`int`	`1`
`width`	Do not set this property, use `Treecol.width` instead.	`int`	

Methods

Name	Description	Data Type	Values
`getColumnIndex`	Returns the column index of this cell, starting from 0.	`int`	
`getLevel`	Returns the level this cell is.	`int`	
`getMaxlength`	Returns the maximal length for this cell, which is decided by the corresponding `getTreecol`()'s `Treecol.getMaxlength`().	`int`	
`getTree`	Return the tree that owns this cell.	`Tree`	
`getTreecol`	Returns the tree col associated with this cell, or null if not available	`Treecol`	

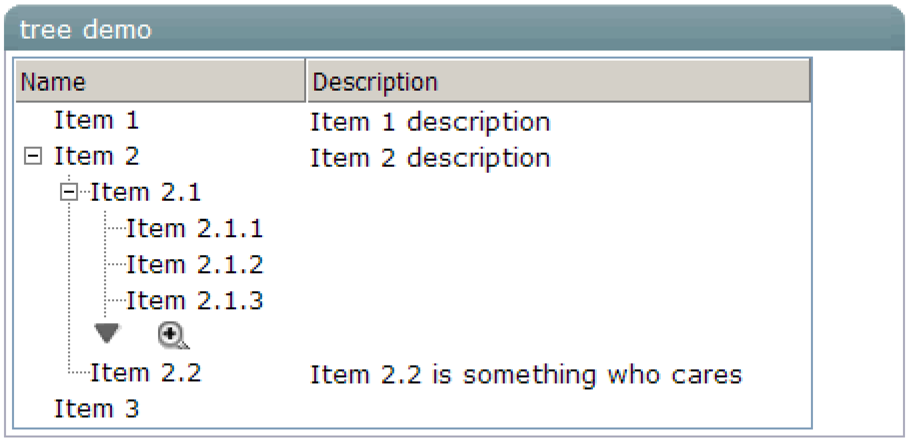
Inherited From

Inherited From
org.zkoss.zul.impl.LabelImageElement
org.zkoss.zul.impl.LabelElement
org.zkoss.zul.imp.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Treechildren

`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



Name	Description
Item 1	Item 1 description
Item 2	Item 2 description
Item 2.1	
Item 2.1.1	
Item 2.1.2	
Item 2.1.3	
Item 2.2	Item 2.2 is something who cares
Item 3	

```
<window title="tree demo" border="normal" width="450px">
   <tree id="tree" width="90%" >
      <treecols sizable="true">
         <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 pageSize="3">
                   <treeitem>
```

```
                          <treerow>
                                <treecell label="Item 2.1.1"/>
                          </treerow>
                    </treeitem>
                    <treeitem>
                          <treerow>
                                <treecell label="Item 2.1.2"/>
                          </treerow>
                    </treeitem>
                    <treeitem>
                          <treerow>
                                <treecell label="Item 2.1.3"/>
                          </treerow>
                    </treeitem>
                    <treeitem>
                          <treerow>
                                <treecell label="Item 2.1.4"/>
                          </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>
</window>
```

Class Name

`org.zkoss.zul.Treechildren`

Supported Child Components

Treeitem

Supported Events

Event Name	Event Type
`onPaging`	org.zkoss.zul.event.PagingEvent **Description:** Notifies one of the pages of a multi-page component is selected by the user.
onPageSize	org.zkoss.zul.event.PageSizeEvent

Event Name	Event Type
	Description: Used to notify that the page size is changed (by the user)

Properties

Property	Description	Data Type	Default Value
`pageSize`	Sets the page size which controls the number of visible child Treeitem. -1 means no limitation. The default value gets by `getTree().getPageSize()`	`int`	`10`
`activePage`	Sets the active page (starting from 0).	`int`	0

Methods

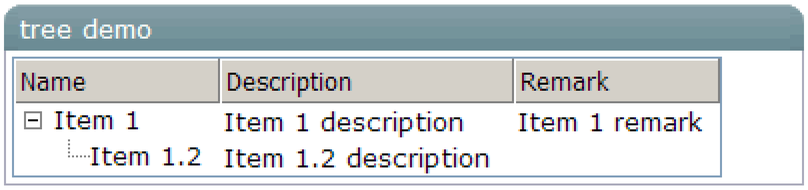
Name	Description	Data Type	Values
`getActivePage`	Returns the active page (starting from 0).	`int`	
`getItemCount`	Returns the number of child Treeitem including all descendants.	`int`	
`getItems`	Returns a readonly list of all descending Treeitem (children's children and so on).	`Collection`	
`getLinkedTreerow`	Returns the `Treerow` that is associated with this treechildren, or null if no such treerow.	`Treerow`	
`getPageCount`	Returns the number of pages (at least one).	`int`	
`getTree`	Returns the Tree instance containing this element.	`tree`	
`getVisibleBegin`	Returns the index of the first visible child.	`int`	
`getVisibleEnd`	Returns the index of the last visible child.	`int`	

Inherited From

Inherited From
org.zkoss.zul.imp.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Treecol

A `treecol` is a top column of tree, Its parent must be `Treecols`.



Name	Description	Remark
⊟ Item 1	Item 1 description	Item 1 remark
Item 1.2	Item 1.2 description	

```
<window title="tree demo" border="normal" width="400px">
   <tree id="tree" width="90%" rows="2">
      <treecols sizable="true">
         <treecol label="Name"/>
         <treecol label="Description"/>
         <treecol label="Remark"/>
      </treecols>
      <treechildren>
         <treeitem>
            <treerow>
               <treecell label="Item 1"/>
               <treecell label="Item 1 description"/>
               <treecell label="Item 1 remark"/>
            </treerow>
         <treechildren>
            <treeitem>
               <treerow>
                  <treecell label="Item 1.2"/>
                  <treecell label="Item 1.2 description"/>
               </treerow>
            </treeitem>
         </treechildren>
         </treeitem>
      </treechildren>
   </tree>
</window>
```

Class Name

`org.zkoss.zul.Treecol`

Supported Child Components

*NONE

Supported Events

*NONE

Properties

Property	Description	Data Type	Default Value
`maxlength`	Sets the maximal length of each item's label.	`int`	`0`

Methods

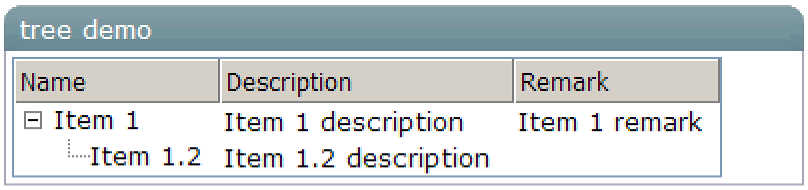
Name	Description	Data Type	Values
`getColumnIndex`	Returns the column index, starting from 0.	`int`	
`getTree`	Returns the tree that it belongs to.	`Tree`	

Inherited From

Inherited From
org.zkoss.zul.impl.HeaderElement
org.zkoss.zul.impl.LabelImageElement
org.zkoss.zul.impl.LabelElement
org.zkoss.zul.imp.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Treecols

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.



Name	Description	Remark
⊟ Item 1	Item 1 description	Item 1 remark
Item 1.2	Item 1.2 description	

```
<window title="tree demo" border="normal" width="400px">
   <tree id="tree" width="90%" rows="2">
      <treecols sizable="true">
         <treecol label="Name"/>
         <treecol label="Description"/>
         <treecol label="Remark"/>
      </treecols>
      <treechildren>
         <treeitem>
            <treerow>
               <treecell label="Item 1"/>
               <treecell label="Item 1 description"/>
               <treecell label="Item 1 remark"/>
            </treerow>
         <treechildren>
            <treeitem>
               <treerow>
                  <treecell label="Item 1.2"/>
                  <treecell label="Item 1.2 description"/>
               </treerow>
            </treeitem>
         </treechildren>
         </treeitem>
      </treechildren>
   </tree>
</window>
```

Class Name

`org.zkoss.zul.Treecols`

Supported Child Components

Treecol

Supported Events

Event Name	Event Type
onColSize	org.zkoss.zul.event.ColSizeEvent **Description:** Notifies the parent of a group of headers that the widths of two of its children are changed by the user.

Properties

*NONE

Methods

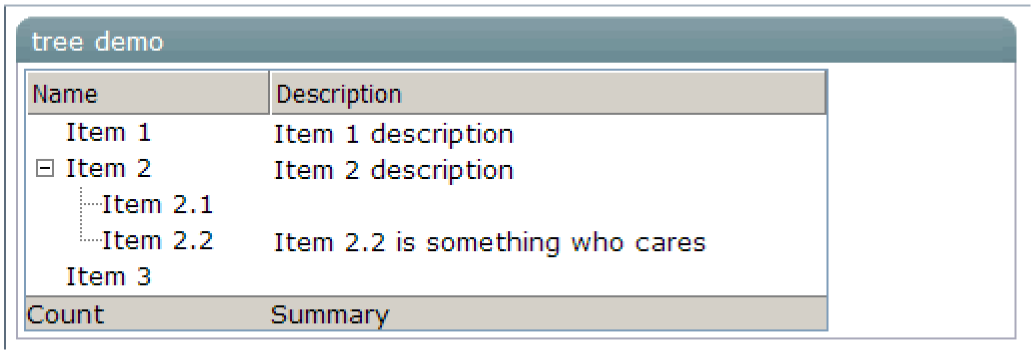
Name	Description	Data Type	Values
getTree	Returns the tree that it belongs to.	org.zkoss.zul.Tree	

Inherited From

Inherited From
org.zkoss.zul.impl.HeadersElement
org.zkoss.zul.imp.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Treefoot

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.



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">
   <tree id="tree" width="400px" rows="5">
      <treecols sizable="true">
         <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>
               </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>
      <treefoot>
         <treefooter label="Count"/>
         <treefooter label="Summary"/>
      </treefoot>
   </tree>
</window>
```

Class Name

`org.zkoss.zul.Treefoot`

Supported Child Components

Treefooter

Supported Events

*None

Properties

*None

Methods

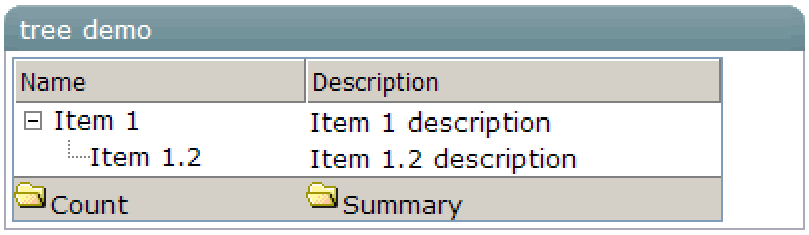
Name	Description	Data Type	Values
getTree	Returns the tree that it belongs to.	Tree	

Inherited From

Inherited From
org.zkoss.zul.imp.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Treefooter

A `treefooter` is a bottom column of tree, Its parent must be `Treefoot`. You colud place any child in a tree footer.



```
<window title="tree demo" border="normal" width="400px">
   <tree id="tree" width="90%" rows="2">
      <treecols sizable="true">
         <treecol label="Name"/>
         <treecol label="Description"/>
      </treecols>
      <treechildren>
         <treeitem>
            <treerow>
               <treecell label="Item 1"/>
               <treecell label="Item 1 description"/>
            </treerow>
         <treechildren>
            <treeitem>
               <treerow>
                  <treecell label="Item 1.2"/>
                  <treecell label="Item 1.2 description"/>
               </treerow>
            </treeitem>
         </treechildren>
         </treeitem>
      </treechildren>
      <treefoot >
         <treefooter><image src="/img/folder.gif"/>Count</treefooter>
         <treefooter><image src="/img/folder.gif"/>Summary</treefooter>
      </treefoot>
   </tree>
</window>
```

Class Name

`org.zkoss.zul.Treefooter`

Supported Child Components

*ALL

Supported Events

*NONE

Properties

Property	Description	Data Type	Default Value
`span`	Sets the number of columns to span this footer.	`int`	`1`

Methods

Name	Description	Data Type	Values
`getColumnIndex`	Returns the column index, starting from 0.	`int`	
`getTree`	Returns the tree that this belongs to.	`Tree`	
`getTreecol`	Returns the tree header that is in the same column as this footer, or null if not available.	`Treecol`	

Inherited From

Inherited From
org.zkoss.zul.impl.LabelImageElement
org.zkoss.zul.impl.LabelElement
org.zkoss.zul.imp.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

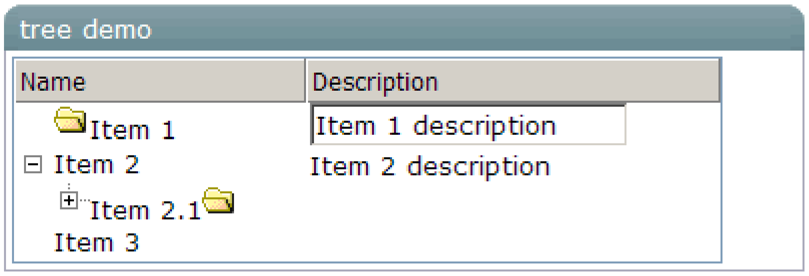
Treeitem

`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.



```
<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>
```

Class Name

`org.zkoss.zul.Treeitem`

Supported Child Components

Treerow, Treechildren

Supported Events

Event Name	Event Type
`onRightClick`	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has right-clicked the component.
`onDoubleClick`	org.zkoss.zk.ui.event.MouseEvent **Description:** Denotes user has double-clicked the component.
`onOpen`	org.zkoss.zk.ui.event.OpenEvent **Description:** 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.

Properties

Property	Description	Data Type	Default Value
`image`	Sets the image of the `Treecell` it contains. If it is not created, we automatically create it. Same as `setSrc`	`String`	
`label`	Sets the label of the Treecell it contains. If it is not created, we automatically create it.	`String`	
`open`	Sets whether this container is open. Values : `true|false`	`boolean`	`true`
`selected`	Sets whether this item is selected. Values : `true|false`	`boolean`	`false`
`src`	Sets the src of the `Treecell` it contains. If it is not created, we automatically create it. The same as `setImage`.	`String`	
`value`	Sets the value. Note: the value is not sent to the browser, so it is OK to be anything.	`Object`	null

Methods

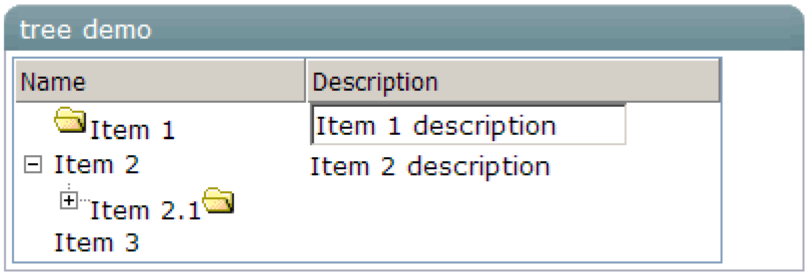
Name	Description	Data Type	Values
`getTreechildren`	Returns the treechildren that this tree item owns, or null if doesn't have any child.	`Treechildren`	
`getTreerow`	Returns the treerow that this tree item owns (might null).	`Treerow`	
`indexOf`	return the index of this container	`int`	
`isContainer`	Returns whether the element is to act as a container which can have child elements.	`boolean`	
`isEmpty`	Returns whether this element contains no child elements.	`boolean`	
`isLoaded`	Return true whether this container is loaded	`boolean`	

Inherited From

Inherited From
org.zkoss.zul.imp.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Treerow

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.



```
<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>
```

Class Name

`org.zkoss.zul.Treerow`

Supported Child Components

Treecell

Supported Events

*None

Properties

Property	Description	Data Type	Default Value
context	Don't use this property of `Treerow`, use `Treeitem.context` instead.	String	Always throws UnsupportedOperationException
popup	Don't use this property of `Treerow`, use `Treeitem.popup` instead.	String	Always throws UnsupportedOperationException
tooltip	Don't use this property of `Treerow`, use `Treeitem.tooltip` instead.	String	Always throws UnsupportedOperationException

Methods

Name	Description	Data Type	Values
getLevel	Returns the level this cell is.	int	
getLinkedTreechildren	Returns the Treechildren associated with this Treerow.	Treechildren	
getSclass	Returns the style class. Note: 1) if not set (or setSclass(null), "item" is	String	

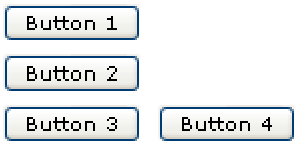
Name	Description	Data Type	Values
	assumed; 2) if selected, it appends " seld" to super's getSclass().		
`getTree`	Returns the Tree instance containing this element.	`Tree`	

Inherited From

Inherited From
org.zkoss.zul.imp.XulElement
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Vbox

The vbox component is used to create a vertically oriented box. Added components will be placed underneath each other in a column.



```
<zk>
   <vbox>
      <button label="Button 1"/>
      <button label="Button 2"/>
   </vbox>
   <hbox>
      <button label="Button 3"/>
      <button label="Button 4"/>
   </hbox>
</zk>
```

Class Name

`org.zkoss.zul.Vbox`

Supported Child Components

*ALL

Supported Events

*NONE

Properties

*NONE

Methods

*NONE

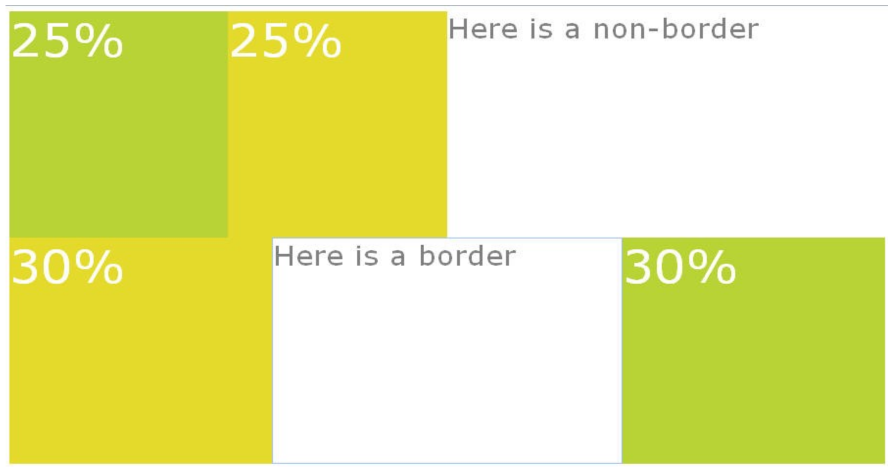
Inherited From

Inherited From
org.zkoss.zul.Box
org.zkoss.zul.impl.XulElement

Inherited From
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

West

This component is a west region. The default class of CSS is specified "layout-region-west".



```
<borderlayout height="500px">
   <north size="50%" border="0">
      <borderlayout>
         <west size="25%" border="none" flex="true">
            <div style="background:#B8D335">
               <label value="25%"
                  style="color:white;font-size:50px" />
            </div>
         </west>
         <center border="none" flex="true">
            <div style="background:#E6D92C">
               <label value="25%"
                  style="color:white;font-size:50px" />
            </div>
         </center>
         <east size="50%" border="none" flex="true">
            <label value="Here is a non-border"
               style="color:gray;font-size:30px" />
         </east>
      </borderlayout>
   </north>
   <center border="0">
      <borderlayout>
         <west size="30%" flex="true" border="0">
            <div style="background:#E6D92C">
               <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 size="30%" flex="true" border="0">
            <div style="background:#B8D335">
               <label value="30%"
                  style="color:white;font-size:50px" />
            </div>
         </east>
      </borderlayout>
   </center>
</borderlayout
```

Class Name

org.zkoss.zkex.zul.West

Supported Child Components

*NONE

Supported Events

Name	Inherited From
OnOpen	org.zkoss.zk.ui.event.OpenEvent **Description:** When a layout is collapsed or opened by a user, the onOpen event is sent to the application.

Properties

Property	Description	Data Type	Default Value
size	Sets the size of this region.	java.lang.String	null

Methods

Name	Description	Return Data Type
getPosition()	Returns Borderlayout.NORTH.	java.lang.String
setWidth(java.lang.String width)	The width can't be specified in this component because its width is determined by other region components (West or East).	void

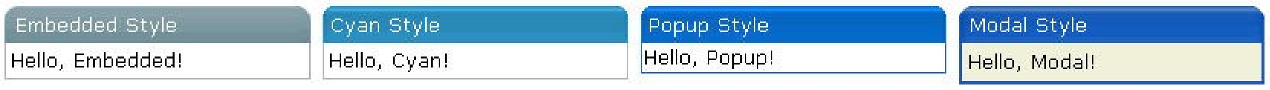
Inherited From

Inherited From
org.zkoss.zkex.zul.LayoutRegion
org.zkoss.zk.ui.HtmlBasedComponent
org.zkoss.zk.ui.AbstractComponent

Window

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 the getFellow method, if it is assigned with an identifier.
- A window could be overlapped, popup, and embedded.
- A window could be a modal dialog.



```
<hbox>
   <window title="Embedded Style" border="normal" width="200px">
      Hello, Embedded!
   </window>
   <window title="Cyan Style" sclass="wndcyan" border="normal" width="200px">
      Hello, Cyan!
   </window>
   <window title="Popup Style" sclass="popup" border="normal" width="200px">
      Hello, Popup!
   </window>
   <window title="Modal Style" sclass="modal" border="normal" width="200px">
      Hello, Modal!
   </window>
</hbox>
```

Class Name

`org.zkoss.zul.Window`

Supported Child Components

*ALL

Supported Events

Name	Event Type
onMove	**Event:** org.zkoss.ui.zk.ui.event.Event Denotes the close button is pressed by a user, and the component shall detach itself.

Name	Event Type
`onOpen`	**Event:** org.zkoss.zk.ui.event.OpenEvent 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.
`onClose`	**Event:** org.zkoss.ui.zk.ui.event.Event Denotes the close button is pressed by a user, and the component shall detach itself.
`onOK`	**Event:** org.zkoss.zk.ui.event.KeyEvent Denotes user has pressed the ENTER key.
`onCacnel`	**Event:** org.zkoss.zk.ui.event.KeyEvent `Denotes user has pressed the ESC key.`
onCtrlKey	**Event:** org.zkoss.zk.ui.event.KeyEvent 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.

Properties

Property	Description	Data Type	Default Value
border	Sets the border **Values:** none | normal	java.lang.String	none
closable	Sets whether to show a close button on the title bar.	boolean	false
contentStyle	Sets the CSS style for the content block of the window.	java.lang.String	<empty string>
ctrlKeys	Sets what keystrokes to intercept.	java.lang.String	<null>
draggable		java.lang.String	<null>
mode	Sets the mode of window. **Values:** overlapped | popup | modal | embedded | highlighted.	int	0
position	Sets how to position the window at the client screen.	java.lang.String	<null>
sizable	Sets whether the window is sizable.	boolean	false
title	Sets the title.	java.lang.String	<empty string>
visible	Changes the visibility of the window.	boolean	false

Methods

Name	Description	Return Data Type
clone()		java.lang.Object
doEmbedded()	Makes this window as embeded with other components	void
doHighlighted()	Makes this window as highlited.	void
doModal()	Makes this window as a modal dialog.	void
doOverlapped()	Makes this window as overlapped with other components.	void
doPopup()	Makes this window as popup, which is overlapped with other component and auto-hiden when user clicks outside of the window.	void

Name	Description	Return Data Type
getContentSclass()	Returns the style class used for the content block.	java.lang.String
getOuterAttrs()		java.lang.String
getTitleSclass()	Returns the style class used for the title.	java.lang.String
inEmbedded()	Returns whether this is embedded with other components	boolean
inHighlighted()	Returns whether this is a highlighted window.	boolean
inModal()	Returns whether this is a modal dialog.	boolean
inOverlapped()	Returns whether this is a overlapped window.	boolean
inPopup()	Returns whether this is a popup window.	boolean
insertBefore(org.zkoss.zk.ui.Component child, org.zkoss.zk.ui.Component insertBefore)		boolean
onChildRemoved(org.zkoss.zk.ui.Component child)		void
onClose()	Process the onClose event sent when the close button is pressed.	void
onModal()	Process the onModal event by making itself a modal window.	void
setPage(org.zkoss.zk.ui.Page page)		void
setParent(org.zkoss.zk.ui.Component parent)		void

Inherited From

Name
org.zkoss.zul.impl.XulElement
org.zkoss.zk.ui.HtmlBasedComponent

Name
org.zkoss.zk.ui.AbstractComponent

Events

CheckEvent

Represents an event cause by user's check a state at the client.

Class Name

`org.zkoss.zk.ui.event.CheckEvent`

Methods

Name	Description	Data Type
`isChecked()`	Returns whether the state is checked.	`boolea`

Inherited From

Inherited From
org.zkoss.zk.ui.event.Event

ColSizeEvent

Used to notify that the widths of two adjacent column are changed.

When an user drags the border of sizable columns, the width of the adjacent columns are changed accordingly - one is enlarged, the other is shrunk and the total width is not changed.

The event is sent to the parent (e.g., Columns and Treecols).

Class Name

`org.zkoss.zul.event.ColSizeEvent`

Methods

Name	Description	Data Type
`getColIndex()`	Return the column index of the first column whose width is changed.	`int`

Name	Description	Data Type
`getColumn1()`	Returns the first column whose width is changed.	org.zkoss.zk.ui.Component
`getColumn2()`	Returns the second column whose width is changed.	org.zkoss.zk.ui.Component
`getKeys()`	Returns what keys were pressed when the column is resized, or 0 if none of them was pressed.	int

Inherited From

Inherited From
org.zkoss.zk.ui.event.Event

CreateEvent

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).

Class Name

`org.zkoss.zk.ui.event.CreateEvent`

Methods

Name	Description	Data Type
`getArg()`	Returns arguments `(org.zkoss.zk.ui.Execution.getArg())` when the component is created.	`java.lang.String`

Inherited From

Inherited From
org.zkoss.zk.ui.event.Event

DropEvent

Represents an event cause by user's dragging and dropping a component.

The component being dragged can be retrieved by `getDragged()`. The component that received the dragged component is `Event.getTarget()`.

Class Name

`org.zkoss.zk.ui.event.DropEvent`

Methods

Name	Description	Data Type
`getDragged`	Returns the component being dragged and drop to `Event.getTarget()`.	`Component`
`getArea`	Not applicable to `DropEvent`. It always returns null if you drag and drop a component to components that partition itself into several areas, such as `imagemap`	String

Inherited From

Inherited From
org.zkoss.zk.ui.event.Event

ErrorEvent

Represents an event cause by user's entering a 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`

Class Name

`org.zkoss.zk.ui.event.ErrorEvent`

Methods

Name	Description	Data Type	Values
`getMessage`	Returns the error message if this event is caused by a wrong data, or `null` if it is to clear messsage.	`String`	

Inherited From

Inherited From
org.zkoss.zk.ui.event.InputEvent
org.zkoss.zk.ui.event.Event

Event

An event sent to the event handler of a component.

Class Name

`org.zkoss.zk.ui.event.Event`

Methods

Name	Description	Data Type
`getData()`	Returns the data accompanies with this event, or null if not available.	`java.util.Set`
`getName()`	Returns the event time.	`java.lang.String`
`getPage()`	Returns the page owning this event, or null if broadcast	`org.zkoss.zk.ui.Page`
`getTarget()`	Returns the target component that receives this event, or null if broadcast.	`org.zkoss.zk.ui.Component`
`isPropagatable()`	Returns whether this is propagatable	`boolean`
`storePropagation()`	Stops the propagation for this event.	`void`
`toString()`		`String`

Inherited From

*NONE

InputEvent

Represents an event cause by user's input something at the client.

Class Name

`org.zkoss.zk.ui.event.InputEvent`

Methods

Name	Description	Data Type
`getValue()`	Returns the value that user input.	`java.lang.String`
`isChangingBySelectBlock()`	Returns whether this event is `onChanging`, and caused by user's selecting a list of items.	`boolean`

Inherited From

Inherited From
org.zkoss.zk.ui.event.Event

KeyEvent

Represents a key pressed by the user.

Class Name

`org.zkoss.zk.ui.event.KeyEvent`

Methods

Name	Description	Data Type
`getKeyCode()`	Returns the key code.	`int`
`isAltKey()`	Returns whether ALT is pressed.	`boolean`
`isCtrlKey()`	Returns whether CTRL is pressed.	`boolean`
`isShiftKey()`	Returns whether SHIFT is pressed.	`boolean`

Inherited From

Inherited From
org.zkoss.zk.ui.event.Event

MouseEvent

Represents an event cause by mouse activitly. There are two possible way to identify a mouse event. One is by coordination (`getX()` and `getY()`. The other is by a logical name, called area (`getArea()`).

Class Name

`org.zkoss.zk.ui.event.MouseEvent`

Methods

Name	Description	Data Type
`getArea()`	Returns the logical name of the area that the click occurs, or null if not available.	`java.lang.String`
`getKeys()`	Returns what keys were pressed when the mouse is clicked, or 0 if none of them was pressed.	int

Name	Description	Data Type
getX()	Returns the x coordination of the mouse pointer relevant to the component.	int
getY()	Returns the y coordination of the mouse pointer relevant to the component.	int

Inherited From

Inherited From
org.zkoss.zk.ui.event.Event

MoveEvent

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()`.

Class Name

`org.zkoss.zk.ui.event.MoveEvent`

Methods

Name	Description	Data Type
getKey	Returns what keys were pressed when the component is moved, or 0 if none of them was pressed. It is a combination of CTRL_KEY, SHIFT_KEY and ALT_KEY.	int
getLeft	Returns the left of the component after moved.	String
getTop	Returns the top of the component after moved.	String

Inherited From

Inherited From
org.zkoss.zk.ui.event.Event

OpenEvent

Represents an event cause by user's openning 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.

Class Name

`org.zkoss.zk.ui.event.OpenEvent`

Methods

Name	Description	Data Type
`getReference`	Returns the reference that is the component causing `Event.getTarget()` to be opened. It is null, if the open event is not caused by opening a context menu, a `tooltip` or a `popup`. Note: the `onOpen` event is also sent when closing the context menu (`tooltip` and `popup`), and this method returns null in this case. Thus, it is better to test `isOpen()` or `getReference()` before accessing the returned value. if (`event.isOpen()`) doSome(`event.getReference()`);	`Component`
`isOpen`	Returns whether it causes open..	boolean

Inherited From

Inherited From
org.zkoss.zk.ui.event.Event

PageSizeEvent

Used to notify that the page size is changed (by the user), or by paginal (such as `Paging`).

Class Name

`org.zkoss.zk.event.PageSizeEvent`

Methods

Name	Description	Data Type
`getPageable`	Returns the pageable controller.	`org.zkoss.zul.ext.Pageable`
getPageSize()	Returns the page size.	`int`

Inherited From

Inherited From
org.zkoss.zk.ui.event.Event

PagingEvent

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.

Class Name

`org.zkoss.zk.ui.event.PagingEvent`

Methods

Name	Description	Data Type
`getPageable`	Returns the `pageable controller`.	`Pageable`
`getActivePage`	Returns the active page (starting from 0).	`int`

Inherited From

Inherited From
org.zkoss.zk.ui.event.Event

PortalMoveEvent

Represents an event caused by a `portal` being moved.

Class Name

`org.zkoss.zkmax.event.PortalMoveEvent`

Methods

Name	Description	Data Type
getDragged	Returns the panel being dragged.	org.zkoss.zul.Panel
getFrom	Returns the portalchildren from the dragged panel.	org.zkoss.zkmax.zul.Portalchildren
getTo	Returns the portalchildren where the dragged panel drops to.	org.zkoss.zkmax.zul.Portalchildren

Inherited From

Inherited From
org.zkoss.zk.ui.event.Event

ScrollEvent

Represents an event caused by that user is scrolling or has scrolled at the client.

`ScrollEvent` will be sent wih 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: org.zkoss.zul.Slider.

Class Name

`org.zkoss.zk.ui.event.ScrollEvent`

Methods

Name	Description	Data Type	Values
`getPos`	Returns the position.	`int`	

Inherited From

Inherited From
org.zkoss.zk.ui.event.Event

SelectEvent

Represents an event cause by user's the list selection is changed at the client.

Class Name

`org.zkoss.zk.ui.event.SelectEvent`

Methods

Name	Description	Data Type
`getSelectedItems()`	Returns the selected items.(never null)	`java.util.Set`

Inherited From

Inherited From
org.zkoss.zk.ui.event.Event

SelectionEvent

Represents an event cause by user's the list selection is changed at the client.

Class Name

`org.zkoss.zk.ui.event.SelectionEvent`

Methods

Name	Description	Data Type
`getSelectedText()`	Returns the selected text contained in this text.	`java.util.Set`
`getSart()`	Returns the selected text's start position.	`int`
`getEnd()`	Returns the selected text's end position.	`int`

Inherited From

Inherited From
org.zkoss.zk.ui.event.Event

SizeEvent

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()`.

Class Name

`org.zkoss.zk.ui.event.SizeEvent`

Methods

Name	Description	Data Type
`getKeyCode()`	Returns the height of the component after re-sized.	`java.lang.String`
`isAltKey()`	Returns what keys were pressed when the component is resized, or 0 if none of them was pressed.	`int`
`isCtrlKey()`	Returns the width of the component after re-sized.	`int`

Inherited From

Inherited From
org.zkoss.zk.ui.event.Event

UploadEvent

Represents that user has uploaded one or several files from the client to the server.

Class Name

`org.zkoss.zk.ui.event.UploadEvent`

Methods

Name	Description	Data Type
`getMedia()`	Returns the first media being uploaded, or null if no file is uploaded.	`org.zkoss.util.media.Media`
`getMedias()`	Returns the array of media being uploaded, or null if the user uploaded no file at all.	`org.zkoss.util.media.Media[]`

Inherited From

Inherited From
org.zkoss.zk.ui.event.Event

ZIndexEvent

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: org.zkoss.zul.Window.

Class Name

`org.zkoss.zk.ui.event.ZIndexEvent`

Methods

Name	Description	Data Type	Values
`getZIndex`	Returns the z-index of the component after moved.	`int`	

Inherited From

Inherited From
org.zkoss.zk.ui.event.Event

Supplemental Classes

AbstractListModel

A skeletal implementation for ListModel.

Class Name

`org.zkoss.zul.AbstractListModel`

Methods

Name	Description	Return Data Type
`AddListDataListener (ListDataListener l)`	Adds a listener to the list that's notified each time a change to the data model occurs.	`void`
`RemoveListDataListener (ListDataListener l)`	Removes a listener from the list that's notified each time a change to the data model occurs.	`void`

Inherited From

Inherited From
org.zkoss.zul.ListModel

Constraint

A constraint.

Interface Name

`org.zkoss.zul.Constraint`

Methods

Name	Description	Return Data Type
`validate(org.zkoss.zk.ui.Component comp, java.lang.Object value)`	Verifies whether the value is acceptable.	`void`

Constrained

Decorates a component that its value is constrained by Constraint.

Interface Name

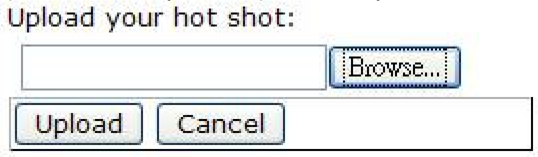
`org.zkoss.zul.Constrainted`

Methods

Name	Description	Return Data Type
`getConstraint()`	Returns the constraint, or null if no constraint at all.	`org.zkoss.zul.Constraint`
`SetConstraint (Constraint constr)`	Sets the constraint.	`void`

Fileupload

A fileupload dialog used to let user upload a file.The `fileupload` component is not a modal dialog. Rather, it is a component, so it is placed inline with other components.



```
<image id="img"/>
Upload your hot shot:
<fileupload onUpload="img.setContent(event.media)"/>
```

Class Name

`org.zkoss.zul.Fileupload`

Properties

Property	Description	Data Type	Default Value
`number`	Sets the maximal allowed number of files to upload.	`int`	`1`
`template`	Sets the template used to create the upload modal dialog. **Template:** ~./zul/html/fileuploaddlg.zul **Note:** the template has no effect, if you use Fileupload as a component (and embed it to a page).	`String`	`~./zul/html/fileuploaddlg.zul`

Methods

Name	Description	Return Data Type
`get()`	Opens a modal dialog with the default message and title, and let user upload a file.	`org.zkoss.util.media.Media`
`get(int max)`	Opens a modal dialog to upload multiple files with the default message and title.	`org.zkoss.util.media.Media[]`
`get(java.lang.String message, java.lang.String title)`	Opens a modal dialog with the specified message and title, and let user upload a file.	`org.zkoss.util.media.Media`
`get(java.lang.String message, java.lang.String title, int max)`	Opens a modal dialog to upload multiple files with the specified message and title.	`org.zkoss.util.media.Media[]`
isChildable()	Determines whether it accepts child components **Value:** `false` **Note:** No child is allowed.	`boolean`
onClose()	Handles the `onClose` event which is sent when file(s) is uploaded or when the cancel button is pressed.	`void`

Inherited From

Inherited From
org.zkoss.zk.ui.HtmlBasedComponent_
org.zkoss.zk.ui.AbstractComponent

ListitemRenderer

Identifies components that can be used as "rubber stamps" to paint the cells in a Listbox.

Interface Name

`org.zkoss.zul.ListitemRenderer`

Methods

Name	Description	Return Data Type
`render(Listitem item, java.lang.Object data)`	Renders the data to the specified list item.	`void`

ListModel

This interface defines the methods that components like Listbox and Grid use to get the content of items.

Interface Name

`org.zkoss.zul.ListModel`

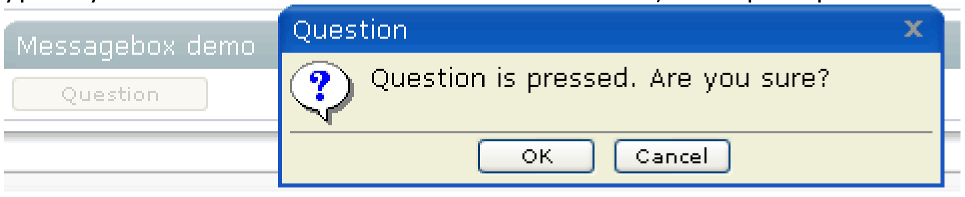
Methods

Name	Description	Return Data Type
`AddListDataListener (ListDataListener l)`	Adds a listener to the list that's notified each time a change to the data model occurs.	`void`
`getElementAt(int index)`	Returns the value at the specified index.	`java.lang.Object`
`getSize()`	Returns the length of the list.	`int`
`removeListDataListener(ListDataListener l)`	Removes a listener from the list that's notified each time a change to the data model occurs.	`void`

Messagebox

It provides a set of utilities to show message boxes.

It is typically used to alert users when an error occurs, or to prompt user for an decision.



```
<window title="Messagebox demo" border="normal">
   <button label="Question" width="100px">
      <attribute name="onClick">{
         Messagebox.show("Question is pressed. Are you sure?", "Question",
         Messagebox.OK | Messagebox.CANCEL, Messagebox.QUESTION);
      }</attribute>
   </button>
</window>
```

Class Name

`org.zkoss.zul.Messagebox`

Properties

Property	Description	Data Type	Default Value
`template`	Sets the template used to create the message dialog.	`String`	`~./zul/html/messagebox.zul`

Methods

Name	Description	Return Data Type
`Show(int messageCode, int titleCode, int button, java.lang.String icon)`	Shows a message box by specifying a message code, and returns what button is pressed.	`int`
`show(int messageCode, java.lang.Object[] args, int titleCode, int button, java.lang.String icon)`	Shows a message box by specifying a message code, and returns what button is pressed.	`int`
`show(int messageCode, java.lang.Object arg, int titleCode, int button, java.lang.String icon)`	Shows a message box by specifying a message code, and returns what button is pressed.	`int`
`show(java.lang.String message)`	Shows a message box and returns what button is pressed.	`int`
`show(java.lang.String message, java.lang.String title, int buttons, java.lang.String icon)`	Shows a message box and returns what button is pressed.	`int`

RendererCtrl

This interface defines the methods components like Listbox use to notify the renderer for several circumstance.

Though `ListitemRenderer.render(org.zkoss.zul.Listitem, java.lang.Object)` is called one item a timer, 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.

Interface Name

`org.zkoss.zul.RendererCtrl`

Methods

Name	Description	Return Data Type
`doCatch(java.lang.Throwable ex)`	Called if any exception occurs when rendering items.	`void`
`doFinally()`	Invoked after all rendering are done successfully or an exception occurs.	`void`
`doTry()`	Called before rendering any item.	`void`

SimpleConstraint

A simple constraint that you could build based the predefined constants.

Interface Name

`org.zkoss.zul.SimpleConstraint`

Methods

Name	Description	Return Data Type
`getClientValidation()`	Returns the function name in JavaScript or a Javascript code snippet used to validate the value at the client.	`String`

Name	Description	Return Data Type
`getErrorMessage(org.zkoss.zk.ui.Component comp)`	Returns the error message when the client detects an error	`String`
`getInstance(java.lang.String flags)`	Parses flags from a string to an integer representing a combination of NO_POSITIVE and other NO_xxx flags.	`org.zkoss.zul. SimpleConstraint`
`isClientComplete()`	Returns whether the client's validation is complete.	`boolean`
`validate(org.zkoss.zk.ui.Component comp, java.lang.Object value)`	Verifies whether the value is acceptable.	`void`

SimpleListModel

A simple implementation of ListModel.

Class Name

`org.zkoss.zul.SimpleListModel`

Methods

Name	Description	Return Data Type
`getElementAt(int index)`	Returns the value at the specified index.	`java.lang.Object`
`getSize()`	Returns the length of the list.	`int`
`sort(java.util.Comparator cmpr, boolean ascending)`	Sorts the data.	`void`

Inherited From

Inherited From
org.zkoss.zul.AbstractListModel

5. The XHTML Components

Overview

- All XHTML components are packed in the `org.zkoss.zhtml` package.
- The XML name space is `http://www.w3.org/1999/xhtml`
- The extensions include `htm`, `html`, `xhtml` and `zhtml`.
- The component names are case-insensitive. Developers could use any combination of lower or upper cases.

URL and encodeURL

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 don't work (unless the servlet context is "").

```
<img href="/my/good.png"/>
```

Rather, you shall use the `encodeURL` function in EL expressions as follows.

```
<?taglib uri="http://www.zkoss.org/dsp/web/core.dsp.tld" prefix="p"?>
...
<img href="${p:encodeURL('/my/good.png')}"/>
```

In Java, you shall use the `encodeURL` method from `org.zkoss.zk.ui.Execution`.

```
<img id="another"/>
<zscript>
  another.setDynamicAttribute("href",
    Executions.getCurrent().encodeURL("/my/good.png"));
</zscript>
```

Notice that XUL components and all ZK features that accept an URL will invoke the `encodeURL` method automatically[6].

AbstractTag

All XHTML components are derived from the `org.zkoss.zhtml.impl.AbstractTag` class.

A XHTML component is a thin wrapper that encapsulates a native HTML tag. It is different from a XUL component or other none-native component in several ways.

- By implementing the `org.zkoss.zk.ui.ext.RawId` interface, the universal identifier (`getUuid`) is the same as the identifier (getId).

6 The reason not to handle XHTML compoents is that we don't know which attribute requires URL.

- By implementing the `org.zkoss.zk.ui.ext.DynamicAttributes` interface, all XHTML components support arbitrary attributes. In other words, any attribute name is legal (as long as the targeted browser supports).

Raw

A special component, `org.zkoss.zhtml.Raw`, 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` is created, such that a proper HTML tag will be generated correspondingly. In other words, any component name is legal (as long as the targeted browser supports).

```
<marquee align="top">...</marquee>
```

It is equivalent to

```
new Raw().setDynamicAttribute("align", "top");
```

Components

A

Abbr

Acronym

Address

Area

B

Base

Big

Blockquote

Body

Br

Button

Caption

Cite

Code

Collection

Colgroup

Dd

Del

Dfn

Dir

Div

Dl

Dt

Em

Embed

Fieldset

Font

Form

H1

H2

H3

H4

Head

Hr

Html

I

Iframe

Img

Input

Ins

Isindex

Kbd

Label

Legend

Li

Link

Map

Menu

Meta

Nobr

Object

Ol

Optgroup

Option

P

Pre

Q

S

Sam

Script

Select

Small

Span

Strong

Style

Sub

Sup

Table

Tbody

Td

Text

Textarea

Tfoot

Th

Thead

Title

Tr

Tt

Ul

Var

Supplement Classes

Fileupload

Messagebox

Appendix A. WEB-INF/web.xml

To add ZK a Web application, you have to add servlets, listeners and a optional filter to web.xml.

ZK Loader

[Required] Class: `org.zkoss.zk.ui.http.DHtmlLayoutServlet`

`DHtmlLayoutServlet` is a servlet used to load ZUML pages when the Web server receives URL requests sent by users.

Notice that you must specify `load-on-startup` since many other servlets depend on the ZK loader.

```
<load-on-startup>1</load-on-startup>
```

It is suggested to map this servlet to the `zul` and `zhtml` extensions as shown in the **Sample** section below. It is OK if you want to map `xul` and `html`, too.

The Initial Parameters

init-param	Descriptions
`update-uri`	[Required] It specifies the URI which the ZK AU engine is mapped to. For example, if the ZK AU engine is mapped to `/zkau/*`, by use of `servlet-mapping`, then specify `/zkau` for this parameter. Note: if the servlet container is used with other Web server, like Apache, you have to map this update URI to the servlet container (in additions to `zul` and `zhtml` files).
`compress`	[Optional][Default: `true`] It specifies whether to compress the output if the browser supports the compression (`Accept-Encoding`) and this Servlet is not included by other Servlets.
`log-level`	[Optional] It specifies the default log level for `org.zkoss`. If not specified, the system default (usually `INFO`) is used. Possible values: `OFF`, `ERROR`, `WARNING`, `INFO`, `DEBUG` and `FINER`. Refer to the **Beyond ZK** chapter in **the Developer's Guide**.

ZK AU Engine

[Required] Class: `org.zkoss.zk.au.http.DHtmlUpdateServlet`

`DHtmlUpdateServlet` is a servlet that handles AJAX requests asynchronously and automatically.

Notice that the URL pattern mapped to this engine must be consistent with the `update-uri` parameter of the ZK Loader.

The Initial Parameters

init-param	Descriptions

`processor0`
`processor1`
`processor2`
`...`
	[Optional]
	
	It specifies an AU processor. The first processor must be specified with the name called `processor0`, second `processor1` and so on.
	
	The syntax of the value is
	
	`/prefix=class`
	
	For example,
	
	`<init-param>`
	`   <param-name>processor0</param-name>`
	`   <param-value>/upload=com.super.MyUploader</param-value>`
	`</init-param>`
	`<init-param>`
	`   <param-name>processor1</param-name>`
	`   <param-value>/extra=com.super.MyExtra</param-value>`
	`</init-param>`
	
	The class must implement the `org.zkoss.zk.au.http.AuProcessor` interface.

ZK Session Cleaner

[Required] Class: `org.zkoss.zk.ui.http.HttpSessionListener`

`HttpSessionListener` is a listener used to clean up memory when a HTTP session is destroyed.

ZK Filter

[Optional] Class: `org.zkoss.zk.ui.http.DHtmlLayoutFilter`

`DHtmlLayoutFilter` is a filter to post-process the output generated by other servlets, such as JSP pages. Its role is similar to the ZK Loader. Unlike the ZK Loader, which loads static ZUML pages from Web applications directly, the ZK filter is designed to process dynamic pages generated by other servlets, say JSP or JSF. It enables developers to add rich user interfaces to existent servlets written in any technology.

Note: the output must be in XHTML (or ZUML) syntax. If you encounter any problem, you can save the generated output into a ZHTML page and then browse the URL whether the ZHTML page is stored.

The Initial Parameters

init-param	Descriptions
`extension`	[Optional][Default: `html`] It specifies how to process the response generated by other servlets. If `html` or `zhtml`, XHTML is assumed to be the default namespace. If `xul` or `zul`, XUL is assumed to be the default namespace.
`charset`	[Optional][Default: `UTF-8`] It specifies the default charset for the output of this filter. If an empty string is specified as follows, the container's default is used. In other words, the `setCharacterEncoding` method of `javax.servlet.ServletResponse` is not called. `<param-value></param-value>`
`compress`	[Optional][Default: `true`] It specifies whether to compress the output if the browser supports the compression (`Accept-Encoding`) and this filter is not included by other Servlets.

How to Specify in web.xml

```
<filter>
   <filter-name>zkFilter</filter-name>
   <filter-class>org.zkoss.zk.ui.http.DHtmlLayoutFilter</filter-class>
</filter>
```

DSP Loader

[Optional] Class: `org.zkoss.web.servlet.dsp.InterpreterServlet`

`InterpreterServlet` is a servlet used to process the DSP files. DSP is a JSP-like template technology. It takes the same syntax as that of JSP. Unlike JSP, DSP is interpreted at the run time, so it is easy to deploy DSP pages. No Java compiler is required in your run-time environment. In addition, you could distribute DSP pages in jar files. This is the way ZK is distributed.

However, you cannot embed Java codes in DSP pages. Actions of DSP, though extensible through TLD files, are different from JSP tags.

The Initial Parameters

init-param	Descriptions
`charset`	[Optional][Default: `UTF-8`] It specifies the default charset for the output of the DSP interpreter. If an empty string is specified as follows, the container's default is used. In other words, the `setCharacterEncoding` method of `javax.servlet.ServletResponse` is not called.
`class-resource`	[Optional][Default: false] Specifies whether to load resources, such as TLD files, from the class loader, in addition to the servlet context.
`compress`	[Optional][Default: `true`] It specifies whether to compress the output if the browser supports the compression (`Accept-Encoding`) and this Servlet is not included by other Servlets.

How to Specify in web.xml

```
<servlet>
   <servlet-name>zkLoader</servlet-name>
   <servlet-class>org.zkoss.web.servlet.dsp.InterpreterServlet</servlet-class>
</servlet>
```

Sample of web.xml

```
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

   <!-- //// -->
   <!-- ZK -->
   <listener>
      <description>ZK listener for cleanup when a session is destroyed</description>
      <listener-class>org.zkoss.zk.ui.http.HttpSessionListener</listener-class>
   </listener>
   <servlet>
      <description>ZK loader for evaluating ZUML pages</description>
      <servlet-name>zkLoader</servlet-name>
      <servlet-class>org.zkoss.zk.ui.http.DHtmlLayoutServlet</servlet-class>

      <!-- Must. Specifies URI of the update engine (DHtmlUpdateServlet).
      It must be the same as <url-pattern> for the update engine.
      -->
      <init-param>
         <param-name>update-uri</param-name>
         <param-value>/zkau</param-value>
      </init-param>
      <load-on-startup>1</load-on-startup><!-- MUST -->
   </servlet>
   <servlet-mapping>
      <servlet-name>zkLoader</servlet-name>
      <url-pattern>*.zul</url-pattern>
   </servlet-mapping>
   <servlet-mapping>
      <servlet-name>zkLoader</servlet-name>
      <url-pattern>*.zhtml</url-pattern>
   </servlet-mapping>
   <servlet>
      <description>The asynchronous update engine for ZK</description>
      <servlet-name>auEngine</servlet-name>
      <servlet-class>org.zkoss.zk.au.http.DHtmlUpdateServlet</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>auEngine</servlet-name>
      <url-pattern>/zkau/*</url-pattern>
   </servlet-mapping>
   <!-- //// -->

   <!-- MIME mapping -->
   <mime-mapping>
      <extension>gif</extension>
      <mime-type>image/gif</mime-type>
   </mime-mapping>
   <mime-mapping>
      <extension>html</extension>
      <mime-type>text/html</mime-type>
   </mime-mapping>
   <mime-mapping>
      <extension>htm</extension>
      <mime-type>text/html</mime-type>
   </mime-mapping>
   <mime-mapping>
      <extension>jad</extension>
      <mime-type>text/vnd.sun.j2me.app-descriptor</mime-type>
   </mime-mapping>
   <mime-mapping>
      <extension>jpeg</extension>
      <mime-type>image/jpeg</mime-type>
   </mime-mapping>
   <mime-mapping>
      <extension>jpg</extension>
      <mime-type>image/jpeg</mime-type>
   </mime-mapping>
   <mime-mapping>
      <extension>js</extension>
      <mime-type>application/x-javascript</mime-type>
   </mime-mapping>
   <mime-mapping>
      <extension>png</extension>
      <mime-type>image/png</mime-type>
   </mime-mapping>
   <mime-mapping>
      <extension>txt</extension>
      <mime-type>text/plain</mime-type>
   </mime-mapping>
   <mime-mapping>
      <extension>xml</extension>
      <mime-type>text/xml</mime-type>
   </mime-mapping>
   <mime-mapping>
      <extension>zhtml</extension>
      <mime-type>text/html</mime-type>
   </mime-mapping>
   <mime-mapping>
      <extension>zul</extension>
      <mime-type>text/html</mime-type>
   </mime-mapping>

   <welcome-file-list>
      <welcome-file>index.zul</welcome-file>
      <welcome-file>index.zhtml</welcome-file>
      <welcome-file>index.html</welcome-file>
      <welcome-file>index.htm</welcome-file>
   </welcome-file-list>
</web-app>
```

Appendix B. WEB-INF/zk.xml

`WEB-INF/zk.xml` is the configuration descriptor of ZK. This file optional. If you need to configure ZK differently from the default, you could provide a file called `zk.xml` under the `WEB-INF` directory.

Overview

The root element must be `<zk>`. Then, you could specify any combination of the following element under the root element.

The `richlet` and `richlet-mapping` elements

To declare a richlet, you have to add the `richlet` element to `zk.xml`. You could specify any number of `richlet` elements. Each of them must have two child elements, `richlet-name` and `richlet-class`, and might have any number of the `init-param` child elements.

The class name specified in the `richlet-class` element must implement the `org.zkoss.zk.ui.Richlet` interface. The name and value specified in the `init-param` element can be retrieved when the `init` method of `org.zkoss.zk.ui.Richlet` is called.

```
<richlet>
   <richlet-name>Test</richlet-name>
   <richlet-class>org.zkoss.zkdemo.TestRichlet</richlet-class>
   <init-param>
      <param-name>any</param-name>
      <param-value>any</param-value>
   </init-param>
</richlet>
```

Once declaring a richlet, you can map it to any number of URL by use of `richlet-mapping` as depicted below.

```
<richlet-mapping>
   <richlet-name>Test</richlet-name>
   <url-pattern>/test</url-pattern>
</richlet-mapping>
<richlet-mapping>
   <richlet-name>Test</richlet-name>
   <url-pattern>/some/more/*</url-pattern>
</richlet-mapping>
```

The URL specified in the `url-pattern` element must start with `/`. If the URI ends with `/*`, then it is matched to all request with the same prefix. To retrieve the real request, you can check the value returned by the `getRequestPath` method of the current page.

```
public void service(Page page) {
   if ("/some/more/hi".equals(page.getRequestPath()) {
      ...
   }
}
```

The `listener` Element

To declare a listener, you have to add the `listener` element to `zk.xml`. You could specify any number of `listener` elements. Each of them could have two child elements, `description` and `listener-class`, where description is optional.

```
<zk>
   <listener>
      <listener-class>my.MyInit</listener-class>
   </listener>
</zk>
```

The type of a listener depends on what interface it implements. For example, if a listener implements the `org.zkoss.zk.ui.event.EventThreadInit` interface, then it is used to listen when an event processing thread is initialized. A listener could implement multiple interfaces and it will be used whenever the corresponding interface is about to call.

The `org.zkoss.zk.ui.event.EventThreadInit` Interface

It is implemented by a listener class that will be used to initialize an event processing thread, before an event is dispatched to it for processing.

If a listener implements this interface, an instance is created, and then the `prepare` method is called in the main thread (aka., the servlet thread), before processing an event. Then, the `init` method is called in the event processing thread.

If a developer wants to prevent an event from being processed, he can throw an exception in the `prepare` method or the `init` method.

A typical use of this feature is to implement auto-authentication. For example, JBoss[7] required you to call `SecurityAssociation.setPrincipal` to grant permissions of a user to the event processing thread, as described in the **Initialization Before Processing Each Event** section, the **Event Listening and Processing** chapter.

The `org.zkoss.zk.ui.event.EventThreadCleanup` interface

It is implemented by a listener class that will be used to cleanup an event processing thread, after it has processed an event.

If a listener implements this interface, an instance is created, and then the `cleanup` method is called in the event processing thread after the thread processes the event. Then, the `complete` method is called in the main thread (aka., the servlet thread), after

7 http://www.jboss.org

the main thread is resumed.

Note: The `complete` method won't be called if the corresponding `cleanup` method threw an exception.

A typical use of this feature is to clean up unclosed transaction.

Once registered, an instance is constructed and the `cleanup` method is called after leaving the event processing thread.

The `org.zkoss.zk.ui.event.EventThreadSuspend` interface

It is implemented by a listener class that will be called before an event processing thread is going to be suspended.

If a listener implements this interface, an instance is created, and then the `beforeSuspend` method, when an event processing thread is going to suspended. It executes in the event processing thread.

A developer can prevent can prevent an event processing thread from being suspended by throwing an exception.

A typical use of this feature is to limit the number of suspended threads.

The `org.zkoss.zk.ui.event.EventThreadResume` interface

It is implemented by a listener class that will be called after an event processing thread is resumed or aborted.

If a listener implements this interface, an instance is created, and then the `beforeResume` method is called in the main thread (aka., the servlet thread), when a suspended event thread is being resumed. Then, the `afterResume` method is called in the event processing thread after the thread is resumed successfully.

If a developer wants to prevent an event from being resumed, he can throw an exception in the `beforeResume` method.

Notice that `beforeResume` executes in the main thread, so it shares the same thread-local storage with the main thread. On the other hand, `afterResume` executes in the event processing thread, so it shares the same thread-local storage with the event thread (and application event listeners).

In additions to resuming normally, a suspended event processing thread might be aborted abnormally. For example, when the desktop is being destroyed, all suspended event threads will be aborted. When the suspended event processing thread is aborted, an instance is created, and the `abortResume` method is called in the main thread.

Note: If a suspended event thread is aborted, none of the `beforeResume` and `afterResume` is called. Moreover, the `cleanup` and `complete` methods of

`EventThreadCleanup` won't be called, either. Thus, you have to handle all necessary cleanups in `abortResume`.

The `org.zkoss.zk.ui.util.EventInterceptor` interface

It is implemented by a listener class that will be used to intercept when an event is sent, posted and processed.

Once registered, an instance is created and shared within the whole application. If you want to intercept events only for a particular desktop, use the `addEventInterceptor` method of the `org.zkoss.zk.ui.Desktop` interface.

The `org.zkoss.zk.ui.util.WebAppInit` interface

It is implemented by a listener class that will be used to initialize a ZK application.

When a ZK application is created, it invokes the `init` method of this interface such that developers could plug the application-specific codes to initialize the application.

The `org.zkoss.zk.ui.util.WebAppCleanup` interface

It is implemented by a listener class that will be used to cleanup a ZK application that is being destroyed.

When a ZK application is going to be destroyed, it invokes the `cleanup` method of this interface such that developers could plug the application-specific codes to cleanup the application.

The `org.zkoss.zk.ui.util.SessionInit` interface

It is implemented by a listener class that will be used to initialize a new session.

When ZK Loader created a new session, it invokes the `init` method of this interface such that developers could plug the application-specific codes to initialize a session.

A developer can prevent a session from being created by throwing an exception in the `init` method.

The `org.zkoss.zk.ui.util.SessionCleanup` interface

It is implemented by a listener class that will be used to cleanup a session that is being destroyed.

When ZK Loader is going to destroy a session, it invokes the `cleanup` method of this interface such that developers could plug the application-specific codes to cleanup a session.

The `org.zkoss.zk.ui.util.DesktopInit` interface

It is implemented by a listener class that will be used to initialize a new desktop.

When ZK Loader created a new desktop, it invokes the `init` method of this interface such that developers could plug the application-specific codes to initialize a desktop.

A developer can prevent a desktop from being created by throwing an exception in the `init` method.

The `org.zkoss.zk.ui.util.DesktopCleanup` interface

It is implemented by a listener class that will be used to cleanup a desktop that is being destroyed.

When ZK Loader is going to destroy a desktop, it invokes the `cleanup` method of this interface such that developers could plug the application-specific codes to cleanup a desktop.

The `org.zkoss.zk.ui.util.ExecutionInit` interface

It is implemented by a listener class that will be used to initialize a new execution.

When ZK Loader and Update Engine created a new execution, it invokes the `init` method of this interface such that developers could plug the application-specific codes to initialize an execution.

Tip: Executions might be stacked. To know whether it is the first execution since a (Servlet) request is processed, you can check whether the `parent` argument is `null`.

A developer can prevent an execution from being created by throwing an exception in the `init` method.

The `org.zkoss.zk.ui.util.ExecutionCleanup` interface

It is implemented by a listener class that will be used to cleanup an execution that is being destroyed.

When ZK Loader is going to destroy an execution, it invokes the `cleanup` method of this interface such that developers could plug the application-specific codes to cleanup an execution.

The `org.zkoss.zk.ui.util.URIInterceptor` interface

It is implemented by a listener class that will be used to intercept the retrieving of ZUML pages with the associated URI. Once registered, an instance of the specified class is created and shared within the whole application. Then, the `request` method is invoked, each time the application wants to retrieve the page definition of a page based on an URI.

A typical use of this interface is to ensure the current user has the authority to access the certain URI.

You can register any number of URI interceptors (`URIInterceptor`).

Note:

1. Unlike `ExecutionInit` and many other listeners, an instance of the registered `URIInterceptor` is created at the time of registration, and then it is shared by the whole application. Thus, you have to make sure it can be accessed concurrently.

The `org.zkoss.zk.ui.util.RequestInterceptor` interface

It is implemented by a listener class that will be used to intercept each request made to ZK Loader and ZK Update Engine. Once registered, an instance of the specified class is created and shared within the whole application. Then, the `request` method is invoked, each time a request is received by ZK Loader or ZK Update Engine.

A typical use of this interface is to determine the locale and/or time zone of the request. Refer to the **Developer's Guide** for more information.

You can register any number of the request interceptors (`RequestInterceptor`).

Note:

1. Unlike `ExecutionInit` and many other listeners, an instance of the registered `RequestInterceptor` is created at the time of registration, and then it is shared by the whole application. Thus, you have to make sure it can be accessed concurrently.
2. The request parameters will be parsed with the proper locale and character encoding, after the `request` method is called. It is not recommended to call the `getParameter` or `getParameterValues` methods (of `javax.servlet.ServletRequest`) in this method.

The `org.zkoss.zk.ui.util.UiLifeCycle` interface

It is implemented by a listener class that will be used to handle something dependin on the life cycle of UI, such as attaching a component to a page, moving a component and so on. Once registered, an instance of the specified class is created and shared within the whole application.

The `org.zkoss.zk.ui.util.PerformanceMeter` interface

It is implemented by a listener that will measure the performance. Unlike other listeners, there is at most one performance meter listener for each Web application. If you like, you can chain them together manually.

The `org.zkoss.zk.ui.util.Monitor` interface

It is implemented by a listener that will be used to monitor the statuses of ZK. Unlike other listener, there is at most one monitor listener for each Web application. If you like, you can chain them together manually.

ZK provides an implementation named `org.zkoss.zk.ui.util.Statistic`, which accumulates the statistic data in the memory. It is a good starting point to understand the load of your ZK application.

The `log` Element

By default, ZK's logger depends on how the Web server is configured. However, you could configure ZK to load and monitor i3-log.conf as described in the **Logger** section of the **Beyond ZK** chapter.

```
<log>
   <log-base>org.zkoss</log-base>
</log>
```

If you want to use the same logging mechanism in your application, you could configure ZK to handle all loggers as follows.

```
<log>
   <log-base></log-base>
</log>
```

where an empty string means all packages, not just `org.zkoss` in the previous example.

The `client-config` Element

It is used to customize the behavior of the ZK Client Engine. You might have multiple `client-config` elements in one `zk.xml`.

```
<client-config>
   <click-filter-delay>0</click-filter-delay>
   <debug-js>false</debug-js>
   <disable-behind-modal>false</disable-behind-modal>
   <error-reload>
      <error-code>301</error-code>
      <reload-uri></reload-uri>
   </error-reload>
   <keep-across-visits>true</keep-across-visits>
   <processing-prompt-delay>900</processing-prompt-delay>
   <tooltip-delay>800</tooltip-delay>
   <resend-delay>9000</resend-delay>
</client-config>
```

The `click-filter-delay` Element

[Default: 0]

It specifies the time, in milliseconds, to filter out consecutive click events. If two click events (including `onOK` and `onCancel`) come too close (within the specified delay), the second one will be removed to avoid the denial-of-service attack.

A non-positive value turns off this feature.

The `debug-js` Element

[Default: `false`]

It specifies whether to debug JavaScript files. By default, it is false and the compressed version of JavaScript files will be loaded. They are hard to read and debug, though the footprint is much smaller.

To debug JavaScript files, you can specify it to true. Then, the original uncompressed JavaScript files will be loaded instead.

The `disable-behind-modal` Element

[Default: `false`]

It specifies whether to disable all elements behind the modal window at the browser. If disabled, the performance is better.

> **Tip**: This option can be considered as obsolete since 3.0.4. The user is almost impossible to change focus to an element behind the modal window, without disabling them.
>
> **Note**: in ZK 3.0.3 and earlier, the option is default to true.

The `error-reload` Element

[Default: *reload if 301, 402 or 403; show an error message, otherwise*]

It specifies what URI to redirect the browser to. For example, if you prefer to redirect to the login page, say, `login.zul`, you can specify the following in zk.xml:

```
<error-reload>
   <error-code>301</error-code>
   <reload-uri>/login.zul</reload-uri>
</error-reload>
<error-reload>
   <error-code>402</error-code>
   <reload-uri>/login.zul</reload-uri>
</error-reload>
<error-reload>
   <error-code>403</error-code>
   <reload-uri>/login.zul</reload-uri>
```

```
</error-reload>
```

If the content of `reload-uri` is empty, the browser simply reloads the same page again.

```
<reload-uri></reload-uri>
```

If you want to show an error message instead, specify `false`.

```
<reload-uri>false</reload-uri>
```

The `keep-across-visits` Element

[Default: `false`]

It specifies whether to keep the desktop when a user reloads an URL or browses away to another URL. Since browsers won't cache HTML pages generated by ZK, ZK removes a desktop as soon as the user reloads the URL or browses to another URL.

However, you have to specify `keep-across-visits` with `true`, if you use the server-side cache for the HTML pages generated by ZK. An example of the server side cache is OpenSymphony CacheFilter[8].

```
<client-config>
   <keep-across-visits>true</keep-across-visits>
</client-config>
```

Note: When working with Opera, ZK always keeps the desktop (until the number of desktops exceed the allowed maximal number), since Opera is smart enough to preserve the most updated content and always reuses the cached page.

The `processing-prompt-delay` Element

[Default: 900]

It specifies the time, in milliseconds, to wait before prompting the user with a dialog indicating that the request is in processing.

The `resend-delay` Element

[Default: 9000 if Enterprise Edition[9], -1 otherwise]

It specifies the time, in milliseconds, to wait before resending the AU requests to the server. There are many reasons an AU request is not received by the server. For example, the server may drop the connection if it is overloaded. To ensure the reliability, ZK will abort the previous request and then resend the request, if the specified delay expires.

Since 3.0.3, you can specify a non-positive number to disable the resend mechanism.

8 http://www.opensymphony.com/oscache/wiki/CacheFilter.html

9 More precisely, 9000 if zkmax.jar is installed.

The `tooltip-delay` Element

[Default: 800]

It specifies the time, in milliseconds, to wait before popping up the tooltip when the user moves the mouse pointer over particular UI components.

The `desktop-config` Element

It is used to customize how ZK handles desktops. You might have multiple `desktop-config` elements in one `zk.xml`.

```
<desktop-config>
   <desktop-timeout>3600</desktop-timeout>
   <disable-theme-uri>~./zul/css/norm*.css.dsp*</disable-theme-uri>
   <extendlet-check-period>10</extendlet-check-period>
   <file-check-period>5</file-check-period>
   <id-to-uuid-prefix>_zid_${page}_</id-to-uuid-prefix>
   <theme-uri>/my/blue**.css</theme-uri>
   <theme-provider-class>my.MyThemeProvider</theme-provider-class>
</desktop-config>
```

The `desktop-timeout` Element

[Default: 3600]

It specifies the time, in seconds, between client requests before a desktop is invalidated. A negative time indicates the desktop should never timeout.

The `disable-theme-uri` Element

[Default: *none*]

It specifies what theme URI to be disabled. The theme URI shall be one of the default theme URI. For example, the following statement disables the generation of the default theme URI for the ZK XUL component set.

```
<desktop-config>
   <disable-theme-uri>~./zul/css/norm*.css.dsp*</disable-theme-uri>
</desktop-config>
```

It is usually used with the `theme-uri` element to replace the default theme. Refer to the **Themes** section of the **Internationalization** chapter in the **Developer's Guide** for details.

The `extendlet-check-period` Element

[Default: -1 (never expired)]

It specifies the time, in seconds, to wait before checking whether a resource loaded by an extendlet is modified. An extendlet is a processor to load the resources usually located in classpath, such as `~./zul/desktop.dsp`.

Resources located in classpath are usually packed as a JAR file, so they are immutable and not need to check if modified. However, in a development environment, you might want to check if they are deployed without reloading the JAR files.

The `file-check-period` Element

[Default: 5]

It specifies the time, in seconds, to wait before checking whether a file is modified.

For better performance, ZK has employed a cache to store parsed ZUML file. The time specified here controls how often ZK checks whether a file is modified. The larger the number the better the performance.

The `id-to-uuid-prefix` Element

[Default: *none*]

It specifies whether to generate UUID based on ID. It is useful for testing purpose, such that the generated UUID is predictable. For example, the following causes the UUID to be generated by prefixing ID with `_zid_`.

```
<desktop-config>
   <id-to-uuid-prefix>_zid_</id-to-uuid-prefix>
</desktop-config>
```

Then, the following component's UUID will be `_zid_foo`.

```
<textbox id="foo"/>
```

Notice that UUID has to be unique in the whole desktop, so the above setting might cause the application unable to run (due to replicated UUID). You can minimize the possibility of ID conflicts by prefixing with page's UUID as follows.

```
<desktop-config>
   <id-to-uuid-prefix>_zid_${page}_</id-to-uuid-prefix>
</desktop-config>
```

Then, `${page}` will be replaced with the page's UUID (if the page is available). However, since a page might still have several ID space, the UUID conflict is still possible.

The `repeat-uuid` Element

[Default: false]

It specifies whether to use the same UUID sequence for desktops for each reboot.

By default, it is turned off so the desktop's UUID is completely different after reboot. It helps to avoid the consistency between the browser and the server. However, it is useful to turn this option on if you want to debug and test the application.

The `theme-uri` Element

[Default: *none*]

It specifies the URI of an addition theme (aka., a style sheet file).

Like other URI, it accepts "*" for loading browser and Locale dependent style sheet. Refer to the **Browser and Locale Dependent URI** section in the **Internationalization** chapter for details.

You can specify any number of `them-uri` as follows.

```
<desktop-config>
   <theme-uri>/my/blue**.css</theme-uri>
   <theme-uri>/my/second.css</theme-uri>
</desktop-config>
```

If you want to replace a default theme, you have to use `theme-uri` with `disable-theme-uri`. Refer to the **Themes** section of the **Internationalization** chapter in the **Developer's Guide** for details.

Notice:

1. All style sheets defined in `lang.xml` and `lang-addon.xml` are loaded, no matter this parameter is defined or not. It is convenient for developers to override certain styles.
2. Each JAR could specify a `lang-addon.xml` file (under the `metainfo/zk` directory), so you could specify style sheets there if you have more than one style sheets.
3. You could specify extra CSS files for individual ZUML pages by use of the `style` component. Refer to the **ZUML with the XUL Component Set** chapter.

The `theme-provider-class` Element

[Default: *none*]

It specifies the class to provide the theme (aka., a style sheet file) URI dynamically. If you want to determine the theme based on the current user, cookie or locale. You can implement a class with the `org.zkoss.zk.ui.util.ThemeProvider` interface, and specify it with the `theme-provider-class` element. Then, an instance of the class will be created, and it is called each time a desktop is rendered to the client to determine the theme URI.

Notice that the theme provider is called with all theme URIs that shall be generated (including what are specified in `theme-uri` and excluding what are specified in `disable-theme-uri`). And, only the return collection of URIs are actually generated. In other words, the theme provider has the highest priority.

The `xel-config` Element

The allowed child elements include `evaluator-class`. At most one `xel-config` element is allowed for each `zk.xml`.

```
<xel-config>
   <evaluator-class>my.MyExpressionFactory</evaluator-class>
</xel-config>
```

The `evaluator-class` Element

[Default: `org.zkoss.xel.el.ELFactory`]

It specifies the class used to evaluate XEL (Extensible Expression Language) expressions. The specified class must implement the `org.zkoss.xel.ExpressionFactory` interface.

If not specified, ZK uses the XEL implementation from ZK Commons EL (`zcommons-el.jar`), which is a performance-enhanced version of Apache Commons EL.

If your Web server uses another implementation, you can do one of the following:

1. If you prefer the implementation based on Apache JSP 2.1 EL, you have to specify the `org.zkoss.xel.el21.ApacheELFactory` class. If the Web server doesn't support Apache JSP 2.1 EL, you have to copy `el-api.jar` (JSP 2.1 API[10]) and `jasper-el.jar` (Apache's implementation) to your Web application.
2. If you prefer the implementation based on Apache Commons EL (JSP 2.0 EL), you have to specify the `or.zkoss.xel.el.ApacheELFactory` class. If the Web server doesn't support Apache Commons EL, you have to copy `commons-el.jar` to your Web application.
3. If you want a different implementation, you can extend from `org.zkoss.xel.el.ELFactory` or `org.zkoss.xel.el21.ApacheELFactory` by simply overriding the `newExpressionEvaluator` method. Of course, if you prefer, you can implement the `org.zkoss.xel.ExpressionFactory` interface directly.

The `language-config` Element

The allowed child elements include `addon-uri`. You might have multiple `language-config` elements in one `zk.xml`.

```
<language-config>
   <addon-uri>/WEB-INF/lang-addon.xml</addon-uri>
   <addon-uri>/WEB-INF/lang-addon2.xml</addon-uri>
</language-config>
```

Note: Unlike most other configurations defined in `WEB-INF/zk.xml`, the definitions defined in language addons are applied to all Web applications sharing the same `zk.jar`.

10 Required only if you are using the Web server that supports only JSP 2.0.

In other words, the definitions in language addons are visible to all Web applications sharing the same zk.jar. Furthermore, it may cause errors in another Web application, if the classes or resources are available only in the Web application defining this.

Thus, if it is an issue, just put zk.jar and relevant ZK libraries under the WEB-INF/lib directory.

The addon-uri Element

[Default: *none*]

It specifies the URI of language add-on definitions. To specify more than one URIs, you have to define them with multiple addon-uri.

A language addon is used to add new components and override the definitions of existent components. Refer to **the Component Development Guide**.

The session-config Element

The allowed child elements include session-timeout and max-desktops-per-session. You might have multiple session-config elements in one zk.xml.

```
<session-config>
   <session-timeout>1800</session-timeout>
   <timer-keep-alive>false</timer-keep-alive>
   <max-desktops-per-session>15</max-desktops-per-session>
   <max-requests-per-session>5</max-requests-per-session>
   <max-pushes-per-session>-1</max-pushes-per-session>
</session-config>
```

The session-timeout Element

[Default: 0 *(depending on the Web server)*]

It specifies the time, in seconds, between client requests before a session is invalidated. A negative time indicates the session should never timeout. The default zero means to use the system default (which is usually specified in web.xml).

The timer-keep-alive Element

[Default: false]

It specifies whether to keep the session alive, when receiving the onTimer event.

A session is considered as timeout (and then invalidated), if it doesn't receive any client request in the specified timeout interval (see the **session-timeout** element above).

By setting this option to true, the onTimer event, just like any other events, will reset the session timeout counter (and then keep the session alive until timeout). Notice that, if this option is false and the timer is shorter than the session timeout, the session won't be expired.

By default, this option is false. It means the `onTimer` event is ignored when handling the session timeout. In other words, the session will expire if no other event is received before timeout.

The `max-desktops-per-session` Element

[Default: 15]

It specifies the maximal allowed number of desktops per session. A desktop represents a HTML page for a browser. In other words, this number controls the number of concurrent browser windows allowed per session.

A negative number means no limitation at all.

Note: If you use `org.zkoss.zk.ui.impl.GlobalDesktopCacheProvider`, then you have to make this number much larger since it means the maximal allowed number of deskoper *per system*.

The `max-pushes-per-session` Element

[Default: -1][since 3.6.1]

It specifies the maximal allowed number of concurrent server-push connections per session.

A negative number means no limitation at all. Zero means no server push is allowed at all.

Notice that a desktop has at most one server-push connection.

The `max-requests-per-session` Element

[Default: 5]

It specifies the maximal allowed number of concurrent requests per session. Each time an user types an URL at the browser, it creates a request and the request ends after the response is sent to the browser. In other words, this number controls how many concurrent requests the same user can send.

A negative number means no limitation at all, but it is not recommended due to the possibility of the denial-of-service (DoS) attacks.

The `system-config` Element

You might have multiple `system-config` elements in one `zk.xml`.

```
<system-config>
   <au-writer-class>my.AuWriter</au-writer-class>
   <cache-provider-class>my.CacheProvider</cache-provider-class>
   <disable-event-thread/>
   <engine-class>my.UiEngine</engine-class>
   <failover-manager-class>my.FailoverManager</failover-manager-class>
   <id-generator-class>my.IdGenerator</id-generator-class>
   <max-spare-threads>100</max-spare-threads>
   <max-suspended-threads>100</max-suspended-threads>
   <max-upload-size>5120</max-upload-size>
   <max-process-time>3000</max-process-time>
   <response-charset>UTF-8</response-charset>
   <session-cache-class>my.SessionCache</session-cache-class>
   <upload-charset>UTF-8</upload-charset>
   <upload-charset-finder-class>my.CharsetFinder</upload-charset-finder-class>
   <ui-factory-class>my.UiFactory</ui-factory-class>
   <url-encoder-class>my.URLEncoder</url-encoder-class>
   <web-app-class>my.WebApp</web-app-class>
</system-config>
```

The `au-writer-class` Element

[Default: `org.zkoss.zk.au.http.HttpAuWriter` for standard and professional editions, or `org.zkoss.zkmax.au.http.SmartAuWriter` for enterprise edition]

It specifies which class used to implement the AU writer. The AU writer is used to generate the output and send it to the client. The class must have a default constructor (without any argument), and implement the `org.zkoss.zk.au.AuWriter` interface.

There are two built-in implementations, `HttpAuWriter` and `SmartAuWriter`. The former one send the output the client after the requests are processed completely. On the other hand, the later one will send a partial output first if the processing is taking too long (half of the value specified in the `resend-delay` element). By sending the partial output, the client will know the server is still alive.

The `cache-provider-class` Element

[Default: `org.zkoss.zk.ui.impl.SessionDesktopCacheProvider`]

It specifies which class used to implement the desktop cache. The class must have a default constructor (without any argument), and implement the `org.zkoss.zk.ui.sys.DesktopCacheProvider` interface.

One instance of the cache provider is created and shared for each Web application, so you have to synchronize the access properly.

Available implementations are as follows.

Class	Description
`org.zkoss.zk.ui.impl.SessionDesktopCacheProvider`	It stores all desktops from the same session in one single cache. It is simple and fast, but not supporting

Class	Description
	clustering.
`org.zkoss.zk.ui.impl.GlobalDesktopCacheProvider`	It stores all desktops from the same Web application in one single cache. In other words, it doesn't count on session at all. It is useful because some Web server, e.g, BEA WebLogic[11], might be configured to use independent sessions for each request.

The `disable-event-thread` Element

[Default: false (enabled)]

It specifies whether to disable the use of the event processing thread. If disabled, no event processing thread will be used at all. In other words, all events are processed in the Servlet thread directly.

The `engine-class` Element

[Default: `org.zkoss.zk.ui.impl.UiEngineImpl`]

It specifies which class used to implement the UI Engine. The class must have a default constructor (without any argument), and implement the `org.zkoss.zk.ui.sys.UiEngine` interface.

One instance of the UI engine is created and shared for each Web application, so you have to synchronize the access properly.

The `failover-manager-class` Element

[Default: *none*]

It specifies which class used to handle the failover. It is called to recover a desktop, when ZK cannot locate a desktop. The class must have a default constructor (without any argument), and implement the `org.zkoss.zk.ui.sys.FailoverManager` interface.

In most cases, you don't need to provide any implementation. Rather, you can let Web servers to handle failover and clustering for you by specifying the `org.zkoss.zk.ui.http.SerializableUiFactory` class in the `ui-factory-class` element as described above.

The `id-generator-class` Element

[Default: *none*]

It specifies which class used to generate UUID of page and components, and ID of

11 http://www.bea.com

desktops. The class must have a default constructor (without any argument), and implement the `org.zkoss.zk.ui.sys.IdGenerator` interface.

One instance of the ID generator is created and shared for each Web application, so you have to synchronize the access properly.

If no ID generator is specified, the default ID generation algorithm will be used.

The `max-spare-threads` Element

[Default: 100]

It specifies the maximal allowed number of the thread pool for queuing the idle event processing threads. ZK will reuse the idle event processing threads by keeping them in a thread pool. The number specified here then controls the maximal size of the pool.

A negative value indicates there is no limit. Zero means no pool at all.

The `max-suspended-threads` Element

[Default: -1 *(no limit)*]

It specifies the maximal allowed number of the suspended event processing threads. A negative value indicates there is no limit at all.

An instance of `org.zkoss.zk.ui.TooManySuspendedException` is thrown, if an event processing thread is going to suspend and the number of suspended threads exceeds the number specified here. You can use the `error-page` element to control how to display this error, or catch the exception and handle it in a different way.

The `max-upload-size` Element

[Default: 5120]

It specifies the maximal allowed size, in kilobytes, to upload a file from the client. A negative value indicates there is no limit.

The `max-process-time` Element

[Default: 3000]

It specifies the maximal allowed time to process events, in milliseconds. It must be positive. ZK will keep processing the requests sent from the client until all requests are processed, or the maximal allowed time expires.

Note: Since 3.0.1, this setting has no obvious effect on Ajax devices. Ajax devices send the requests synchronously.

The `response-charset` Element

[Default: UTF-8]

It specifies the charset for the rendering result of a ZUML page. In other words, it is used to load the ZUML page by the ZK Loader (i.e., DHtmlLayoutServlet).

If you want to use the container's default value, you can specify an empty string as follows.

```
<response-charset></response-charset>
```

The `session-cache-class` Element

[Default: `org.zkoss.zk.ui.http.SimpleSessionCache`]

It specifies the session cache used to store ZK sessions. It must implement the `org.zkoss.zk.ui.sys.SessionCache` interface.

By default, `org.zkoss.zk.ui.http.SimpleSessionCache` is used and it stores the ZK session in an attribute of the native session (i.e., `HttpSession` or `PortletSession`).

The `upload-charset` Element

[Default: UTF-8]

It specifies the charset (aka., encoding) for the uploaded text files if no charset is specified with the content type.

If the uploaded file is binary, there is no encoding issue at all.

Note: the `upload-charset-finder-class` element, see blow, has the higher priority.

The `upload-charset-finder-class` Element

[Default: null]

It specifies the finder that determines charset (aka.., encoding) for the uploaded text files if no charset is specified with the content type.

If the uploaded file is binary, there is no encoding issue at all.

The finder must implement the `org.zkoss.zk.ui.util.CharsetFinder` interface. Then, when a text file is uploaded, the `getCharset` method is called and it can determines the encoding based on the content type and/or the content of the uploaded file.

Note: it has the higher priority than the `upload-charset` element, see above.

The `ui-factory-class` Element

[Default: `org.zkoss.zk.ui.http.SimpleUiFactory`]

It specifies which class used to create desktops and pages, and to convert URL to a page definition. The class must have a default constructor (without any argument), and implement the `org.zkoss.zk.ui.sys.UiFactory` interface.

One instance of the UI factory is created and shared for each Web application, so you have to synchronize the access properly.

A common use is to load page definitions and other UI information from the database, rather than from the resources of the Web application.

In addition, you might use it to implement a controller in a MVC model, such that it creates the correct desktop based on the request URL.

Available implementations are as follows.

Class	Description
`org.zkoss.zk.ui.http.SimpleUiFactory`	The default UI factory. The sessions generated by this factory is *not* serializable
`org.zkoss.zk.ui.http.SerializableUiFactory`	The sessions generated by this factory is serializable. If you want to store sessions when the Web server is shutdown and restore them after it started, you can specify this implementation.

The `url-encoder-class` Element

[Default: *none*]

It specifies the URL encoder to post-process the URL before sending to the client. By default, the URI generated is in the following format. Sometimes[12] it is appended with the session ID.

```
/context-path/related-uri
/zkdemo/zkau/web/img/spacer.gif
```

In a sophisticated environment, you might want to modify a bit. Then, you can implement the `org.zkoss.web.servlet.http.Encodes.URLEncoder` interface, and specify it with the `url-encoder-class` element.

Notice that, unlike most other configuration, the `url-encoder-class` element affects all applications that share the same ZK libraries.

The `web-app-class` Element

[Default: `org.zkoss.zk.ui.http.SimpleWebApp`]

It specifies which class used to implement the Web application. The class must have a default constructor (without any argument), and implement both the

12 For example, the session ID is appended if the browser disabled the cookie.

org.zkoss.zk.ui.WebApp and org.zkoss.zk.ui.sys.WebAppCtrl interfaces. Instead of implementing from scratch, you can extend it from the org.zkoss.zk.ui.impl.AbstractWebApp or org.zkoss.zk.ui.http.SimpleWebApp classes.

The zscript-config Element

It configures the interpreters to interpret the zscript codes. The allowed child element is language-name and interpreter-class. You might have multiple zscript-config elements in one zk.xml.

```
<zscript-config>
   <language-name>Java</language-name><!-- case insensitive --!>
   <interpreter-class>my.MySuperJavaInterpreter</interpreter-class>
</zscript-config>
```

Note: Unlike most other configurations defined in WEB-INF/zk.xml, the definitions defined in zscript-config are applied to all Web applications sharing the same zk.jar.

In other words, the scripting language defined here are visible to all Web applications sharing the same zk.jar. Furthermore, it may cause errors in another Web application, if the classes or resources are available only in the Web application defining this.

Thus, if it is an issue, just put zk.jar and relevant ZK libraries under the WEB-INF/lib directory.

The language-name Element

[Required]

It specifies the language name. It is case insensitive. The previous implementation with the same language name will be replaced if any.

The interpreter-class Element

[Required]

It specifies the implementation class. It must implement the org.zkoss.zk.scripting.Interpreter interface. Instead of implementing it from scratch, you can derive from the org.zkoss.zk.scripting.util.GenericInterpreter class. If you want to support the hierarchical scopes (i.e., one interpreter scope per namespace), it can also implement the org.zkoss.zk.scripting.HierachicalAware interface.

The device-config Element

It specifies a device. A device represents a client. Different clients have different implementation. Currently there are two types: ajax and mil. They represents the Web browsers with Ajax, and the mobile device with Mobile Interactive Language[13]. It is used to create an instance returned by the `getDevice` method of the `Desktop` intereface.

The allowed child element is `device-type` , `device-class`, `timeout-uri`, and `unavailable-message`. You might have multiple `device-config` elements in one `zk.xml`.

```
<device-config>
   <device-type>ajax</device-type>
   <device-class>my.MyAjaxDevice</device-class>
   <timeout-uri>/my-timeout.zul</timeout-uri>
   <server-push-class>my.ServerPush</server-push-class>
   <unavailable-message><![CDATA[
<p style="color:red">Sorry, JavaScript must be enabled in order for you to use
KillApp.</p>
   ]]></unavailable-message>
   <embed><![CDATA[
<script type="text/javascript">
   AU_progressbar = function (id, msg) {
      Boot_progressbox(id, msg, 0, 0, true, true);
   };
</script>
   ]]></embed>
</device-config>
```

Note: Unlike most other configurations defined in `WEB-INF/zk.xml`, the definitions defined in `device-config` are applied to all Web applications sharing the same `zk.jar`. Refer to the `zscript-config` element for more information.

The `device-class` Element

[Optional]

It specifies the implementation class. The class must implement the `org.zkoss.zk.device.Device` interface. Instead of implementing it from scratch, you can derive from the proper implementation, such as `AjaxDevice` and `MilDevice`.

The `device-type` Element

[Required]

It specifies the device type. The previous implementation with the same device type will be replaced if any.

The `embed` Element

[Optional][Multiple]

It specifies the content that shall be added to the output generated and sent to the client when rendering a desktop. The syntax of the content depends on the client. For Ajax

13 MIL is a ZK markup language used to communicate with the mobile devices.

clients, it can be any tags that can be placed inside HTML `HEAD` tag, such as `SCRIPT`, `META` and others.

For example, if you want to show the progress bar in the center of the browser window, you can specify the following in `zk.xml`.

```
<device-config>
   <device-type>ajax</device-type>
   <embed><![CDATA[
<script type="text/javascript">
   AU_progressbar = function (id, msg) {
      Boot_progressbox(id, msg, 0, 0, true, true);
   };
</script>
   ]]></embed>
</device-config>
```

You can specify multiple `embed` elements and their content be concatenated together.

The `server-push-class` Element

[Optional][Default: *depends on device and what edition you use*]

It specifies which class used to implement the server-push feature. The class must have a default constructor (without any argument), and implement the `org.zkoss.zk.ui.sys.ServerPush` interface.

```
<device-config>
   <device-type>ajax</device-type>
   <server-push-class>my.ServerPush</server-push-class>
</device-config>
```

The `timeout-uri` Element

[Optional][Default: *null*]

It specifies the target URI that will be used to redirect users to, when the desktop no longer exists – it is usually caused by session timeout. If this element is omitted, an error message will be shown up at the browser to alert users for what happens.

To reload the same URI again, you can specify an *empty* content as follows.

```
<device-config>
   <device-type>ajax</device-type>
   <timeout-uri></timeout-uri>
</device-config>
```

The `unavailable-message` Element

[Optional][Default: *depends on device*]

It specifies the message that will be displayed if the client doesn't support this device.

The `error-page` Element

```
<error-page>
   <device-type>[ajax|mil]</device-type>
   <exception-type>ClassName</exception-type>
   <location>the error page's URI</location>
</error-page>
```

It specifies an error page used when an un-caught exception is thrown in updating a ZUML page (e.g., in an event listener). Each page is associated with an exception type, aka, a class deriving from `java.lang.Throwable`. You can specify multiple error pages, each with a different exception type. When an error occurs, ZK searches the proper error page by examing the exception type one-by-one. If none is found, it shows, by default, an alert message at the client.

The `device-type` element is optional. If omitted, `ajax` is assumed. If you want to specify an error page for mobile devices, it has to be `mil`.

The `preference` Element

```
<preference>
   <name>any name</name>
   <value>any value</value>
</preference>
```

Specify a preference with the `preference` element depicted above. The name and value are application specific and you can specify whatever value you like. Mupltie To avoid name conflict, it is suggested to prefix the name with your domain name, such as `com.friend.some.another`.

The preferences are application-level. They are shared by a single ZK application (`org.zkoss.zk.ui.WebApp`).

They can then be retrieved back by calling the `getPreference` method of the `org.zkoss.zk.ui.util.Configuration` class. Notice that each Web application has one configuration, which can be found by use of `getConfiguration` method of the `org.zkoss.zk.ui.WebApp` interface.

```
String value = webApp.getConfiguration().getPreference("org.zkoss.name", null);
if (value != null) {
   ...
}
```

The `library-property` Element

```
<library-property>
   <name>any name</name>
   <value>any value</value>
</library-property>
```

Species a library-level property with the `library-property` element. The above example is equivalent to

```
org.zkoss.lang.Library.setProperty("any name", "any value");
```

The library properties are shared by all Java codes that use the same set of ZK libraries. If you installed ZK libraries in WEB-INF/ib of a ZK application, the library properties are shared only within the application.

The library properties are easier to access than the preferences since they are static members (of `org.zkoss.lang.Library`). However, their scope depends on the installation.

The `system-property` Element

```
<system-property>
   <name>any name</name>
   <value>any value</value>
</system-property>
```

Specifies a system properties with the `system-property` element. The above example is equivalent to

```
System.setProperty("any name", "any value");
```