

SAP HTMLB Guidelines

[Print Version \(PDF\)](#)

Read First

1. Introduction

- [What is HTMLB?](#)
- [About the Reference](#)

2. General

- [Customer Branding and Style Editor](#)
- [Forms - Using Checkboxes](#)
- [Forms - Using Radio Buttons](#)
- [Forms - Using Different List Types](#)
- [Table View Functions](#)
- [Positioning Buttons](#)
- [Error Handling](#)
- [Accessibility of HTMLB controls - General Information](#)

3. Layout

- [General Layout Strategy](#)
- [Layout Hierarchy](#)
- [Spacing Between Grouped Controls](#)
- [Spacing Between Single Controls](#)

4. Layout Controls

- [Content](#)
 - [Control API](#)
- [Document](#)
 - [Control API](#)
- [Page](#)
 - [Control API](#)
- [Form](#)
 - [Control API](#)
- [Flow Layout](#)
 - [Usage and Types](#)
 - [More Info](#)
 - [Control API](#)

- Form Layout
 - [Usage and Types](#)
 - [More Info](#)
 - [Control API](#)
- Grid Layout
 - [Usage and Types](#)
 - [More Info](#)
 - [Control API](#)

5. Visible Controls

- Breadcrumb
 - [Usage and Types](#)
 - [More Info](#)
 - [Control API](#)
- Button
 - [Usage and Types](#)
 - [More Info](#)
 - [Control API](#)
- Chart
 - [Usage and Types](#)
 - [More Info](#)
 - [Control API](#)
- Checkbox
 - [Usage and Types](#)
 - [More Info](#)
 - [Control API](#)
 - [Control API \(Group\)](#)
- Date Navigator
 - [Usage and Types](#)
 - [More Info](#)
 - [Control API](#)
- Dropdown List Box
 - [Usage and Types](#)
 - [More Info](#)
 - [Control API](#)
- File Upload
 - [Usage and Types](#)
 - [More Info](#)
 - [Control API](#)
- Group
 - [Usage and Types](#)
 - [More Info](#)
 - [Control API](#)
- Image
 - [Usage and Types](#)
 - [More Info](#)
 - [Control API](#)
- Input Field
 - [Usage and Types](#)

- [More Info](#)
- [Control API](#)
- **Item List**
 - [Usage and Types](#)
 - [More Info](#)
 - [Control API](#)
- **Label**
 - [Usage and Types](#)
 - [More Info](#)
 - [Control API](#)
- **Link**
 - [Usage and Types](#)
 - [More Info](#)
 - [Control API](#)
- **List Box**
 - [Usage and Types](#)
 - [More Info](#)
 - [Control API](#)
- **Radio Button**
 - [Usage and Types](#)
 - [More Info](#)
 - [Control API](#)
 - [Control API \(Group\)](#)
- **Table View**
 - [Usage and Types](#)
 - [More Info](#)
 - [Control API](#)
- **Tabstrip**
 - [Usage and Types](#)
 - [More Info](#)
 - [Control API](#)
- **Text Edit**
 - [Usage and Types](#)
 - [More Info](#)
 - [Control API](#)
- **Text View**
 - [Usage and Types](#)
 - [More Info](#)
 - [Control API](#)
- **Tray**
 - [Control API](#)
- **Tree View**
 - [Usage and Types](#)
 - [More Info](#)
 - [Control API](#)



Read First

[What Is the Purpose of these Guidelines?](#) | [What Are the Major Building Blocks of the Guidelines?](#) | [Who Is the Target Audience?](#) | [Status](#)

What Is the Purpose of these Guidelines?

These guidelines describe how to develop usable and accessible Web applications and iViews using HTMLB and the SAP Portal framework. These applications include applications that are part of the Portal framework, such as administration and personalization applications.

What Are the Major Building Blocks of the Guidelines?

The three major building blocks are:

- General rules for the design and accessibility of Web applications
- Rules for the overall layout and for special layout controls
- A reference section - the main part of these guidelines - describing the look, behavior, usage and programming interface for each HTMLB control

Who Is the Target Audience?

The guidelines address developers of Web applications and iViews using HTMLB and the SAP Portal framework.

They are also intended as a reference for user interface designers who consult application designers or are directly involved in development projects.

Status

Version 1.0.4, 13 February 2003

Note: This version of the guidelines is a "frozen" version that corresponds to mySAP Enterprise Portal 5.0. It contains design and usability information as well as descriptions of the API for the HTMLB controls. Links to examples and code samples, however, have been removed because these may be subject to change.

If you are interested in more developer-oriented information, please visit the iView Studio Website at www.iviewstudio.com and go to the the "Dev Zone." There you will find more resources and the option to download the Portal Development Kit (PDK). Some code examples may also require that you have a Tomcat server running.

This guideline can be found in *Resources* on the SAP Design Guild Website (www.sapdesignguild.org).

What is HTMLB?

[Basic Idea](#) | [How it Works](#) | [Form](#) | [Controls](#) | [Container](#) | [Events](#) | [Mobile Features](#)

HTMLB (HTML-Business for Java) provides a full set of easy-to-use Web controls. These guidelines describe the HTMLB controls, their types, usage, attributes, and how to set the attributes with the JSP-taglib and the classlib.

For each control there is a general page that describes its usage, types, and design-relevant attributes. This description is on the interaction design level. A further page describes more technical issues, such as browser compatibility, editability in the Style Editor, and accessibility issues. Thirdly, the Control API page provides a development-oriented view of the control with detailed descriptions of attributes and parameters.

In addition, there are pages that describe general interaction design aspects, such as page layout, correct usage of certain often-used controls, as well as hints on finding the right control for a given purpose.

Knowledge of Java, JSP (information can be found at <http://java.sun.com>) and HTML (information can be found at <http://www.w3.org>) is helpful when reading this document.

Basic Idea

HTMLB allows a design-oriented page layout. It is designed to overcome typical servlet problems, such as:

- Visualization and business logic are not separate.
- Content management consumes a lot of qualified manpower. Skills in HTML, CSS, JavaScript etc. are essential.
- Development has to take care of different web clients and -versions.
- Maintaining the corporate identity through out the whole application is hard to achieve.
- Namespace conflicts with form elements

HTMLB provides the technological infrastructure for easy customer branding. See [Customer Branding](#).

How it Works

HTML-Business for Java provides the user with a efficient set of controls - similar to Swing/AWT. The controls are based on servlets and JSP pages. The developer uses bean-like components or JSP tags. Renderer classes finally translate the components into HTML-commands.

To demonstrate the similarity from HTMLB to Swing/AWT some synonyms.

HTMLB	Swing/AWT
Form	ContentPane, JFrame, JDialog
ControlComponent	JComponent
Container	ContainerContainer
Event	AWTEvent, InputEvent ...

Form

It is basically the wrapping paper of your page and essential for the data transfer from the web client to the web browser and for the event handling. Controls in the form must have unique control names. The control names are generated by the HTML-Business for Java renderer - therefore you cannot use e.g. JavaScript to manipulate the controls.

Controls

GUI elements that are used to build an application. The controls are placed in a form. Every control has different attributes that define the "look" of the control. Controls are checkboxes, radio buttons and grids to name a few.

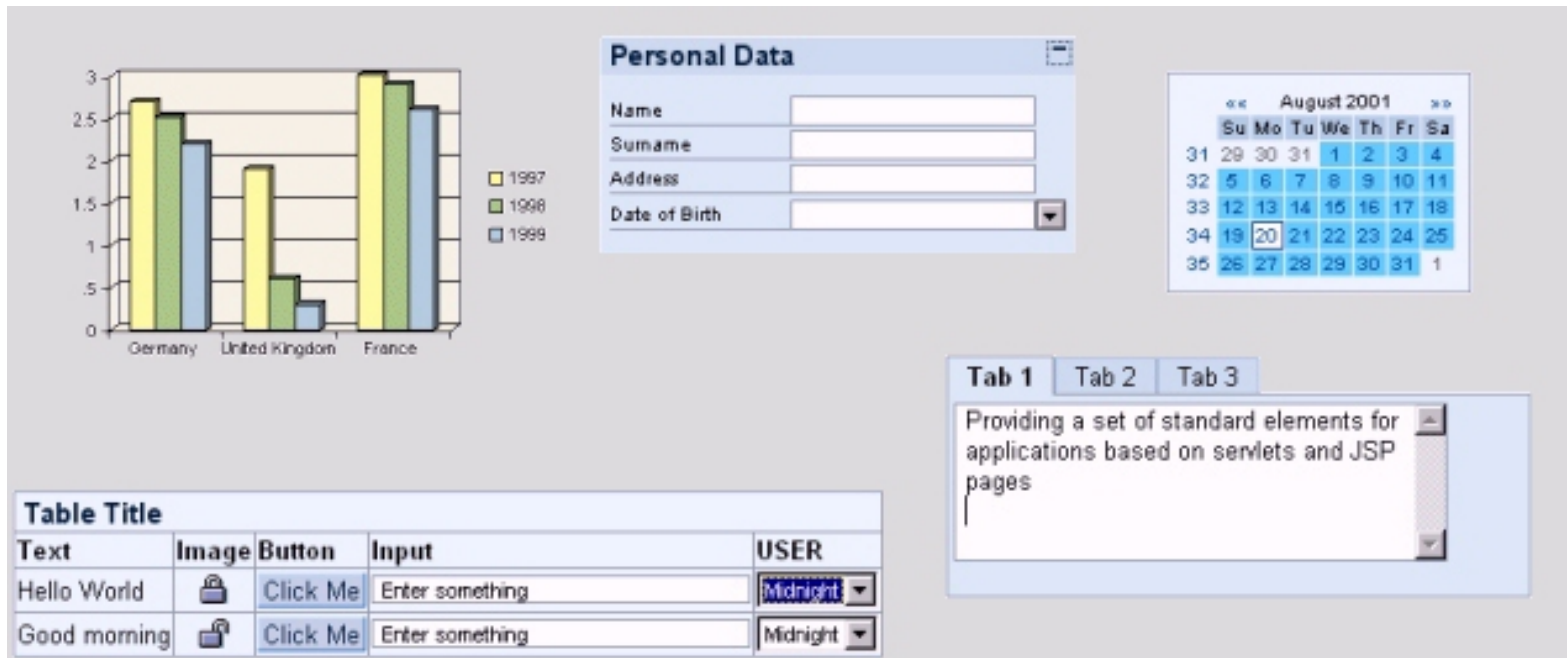


Figure 1: Some HTMLB controls

Container

Container contain controls. Containers can contain containers - nesting. A simple container would be a 'tray' containing a 'gridLayout'. The gridLayout contains 'textView' and 'inputField'.

Events

Components can respond to user action. The response is called an event. An event usually causes a submit (sending the form from the web client to the web server). With the control that can create an event you specify the name of the event handling routine. The web server receives the form, analyzes it and calls the event handling routine which does the further processing.

Mobile Features

The mobile features enhance the functionality of HTML-Business for Java for mobile devices such as Pocket PCs, WAP-enabled mobile phones and other mobile device/browser combinations. The mobile features support a device-independent development of components for mobile devices by providing special renderer classes. These renderer classes consider the special features of different mobile devices and the browsers used.



[Top](#)

About the Reference

[Structure of the Description](#) | [Controls](#)

This page describes the structure of the the Control API pages for the HTMLB controls.

Structure of the Description

The description of the controls is structured in:

- General description - what is it
- Attributes of the control
- Overview of the attributes with possible values, defaults and the manipulation with the JSP-taglib and classlib.
- Example

The **values** column in the overview table specifies which type of parameter the attribute expects. This can be

- String
An ASCII string. Usually event handling routines, names, titles etc.
- Units
An integer value specified in web client units. According to the HTML standard units can be specified in
 - Pixel (px)
Pixels are the smallest addressable unit on the web client. A web client has a maximum resolution, that is the number of horizontal times vertical pixel (e.g. 800x600, 1024x768 etc.) When you specify units in pixel you can make sure that your control is displayed on every web client in the same size.

Pixel is the default unit.

Example
Both expressions set the width of a control to 500 pixel.
`width="500"`
`width="500px"`
 - Percent (%)
The percentage specified is calculated from the visible space of the web client. If e.g. a width of a control is specified with 50% the control uses half of the of the web client width. The control changes its width according to the web client dynamically (e.g. if the web client window gets scaled).

Example
The width of a control is set to 30% of the web client.

`width="30%"`
- Numeric
A numeric expression.
- Others
If an attribute requires specific values. Booleans require "TRUE" or "FALSE" or text size can only be "LARGE", "MEDIUM" and "SMALL".



[Top](#)

Controls

General

To use the controls you have to know about the syntax and the attributes of the controls. Every control has different attributes. In the description we describe the attributes and gather the information in a table which shows the usage with the taglib and the classlib.

Syntax

Programming with the JSP taglib follows the XML syntax. Each control is "wrapped" in tags. To identify the tags as XML the prefix

- hbj: (stands for: HTML-Business for Java)

is used. Some controls (e.g. tray, group) also need a tag body. The tag body specifies the controls that are placed "inside" the tag. The syntax would be like:

Tag

```

<hbj:mycontrol          comment: begin of tag for HTMLB control
  attributes            comment: setting of attributes of HTMLB control
</hbj:mycontrol>      comment: end of tag for HTMLB control

```

Tag with "quick" end of tag (only possible when the tag has no body)

```

<hbj:mycontrol          comment: begin of tag for HTMLB controls
  attributes            comment: setting of attributes of HTMLB control
/>                    comment: end of tag for HTMLB controls

```

Tag with body

```

<hbj:mycontrol_withbody comment: begin of tag for HTMLB control
  attributes            comment: setting of attributes of HTMLB control
  <                    comment: end of tag for HTMLB control
  <hbj:a_control_in_the_body
    attributes
  />
  <hbj:next_control_in_the_body
    attributes
  />
  more controls
</hbj:mycontrol_withbody> comment: end of tag for HTMLB controls with body

```

Scriptlet

A scriptlet can contain any number of language statements, variable or method declarations, or expressions that are valid in the page scripting language.

Within a scriptlet, you can do any of the following:

- Declare variables or methods to use later in the file.
- Write expressions valid in the page scripting language.
- Use any of the implicit objects or any object declared with a `<jsp:useBean>` element.
- Write any other statement valid in the scripting language used in the JSP page (if you use the Java programming language, the statements must conform to the Java Language Specification).

Any text, HTML tags, or JSP elements must be written outside the scriptlet.

Scriptlets are executed at request time, when the JSP container processes the client request. If the scriptlet produces output, the output is stored in the out object.

Certain attributes (if the column "JSP taglib" in the attribute table to each control has no entry) can only be assigned using scriptlets. The scriptlet has to be placed in the tag body of the HTMLB control. The scriptlet starts with `<%` and ends with `%>`. The following example uses the [button](#) control and sets some attributes with a scriptlet.

```

<hbj:button
  id="OrderConfirm"
  width="100px"
  tooltip="Click here to confirm order"
  onClick="ProcessConfirm"
  disabled="false"
  design="STANDARD"
  >
  <%
    OrderConfirm.setText("Confirm"); comment: set the text for the button
    OrderConfirm.setWidth("125px"); comment: set "width" - this overrides
                                   "w
  idth" set in attribute section
  %>
  comment: end of scriptlet
</hbj:button>

```

Result

Confirm

Enumeration Values

In the classlib column some values have to be set as enumeration values. In the classlib column you find the class name and the enum (separated by a dot).

- **Example**
`breadcrumb.setSize(BreadCrumbSize.MEDIUM)`

For an executable program you have to add the location of the enum. That is:

- `com.sapportals.htmlb.enum.`

So according to the example above you have to specify:

- Your program:
`breadcrumb.setSize(com.sapportals.htmlb.enum.BreadCrumbSize.MEDIUM)`

To save some typing when you enumeration values more often the package can be imported:

- `<%@ page import="com.sapportals.htmlb.enum.BreadCrumbSize, " %>`

Boolean Values

Taglib:
Boolean values are specified as string and can be lowercase and/or uppercase.

Classlib:
Boolean values are specified as **boolean** and have to be specified only in lowercase characters.

 [Top](#)

Customer Branding and Style Editor

[Style Editor](#) | [HTMLB Controls and Style Editor](#)

Portal software must reflect the customer's corporate identity and branding guidelines. For this reason, we provide a technological infrastructure and tools to support customers in this goal. The current portal release offers a certain amount of design flexibility that allows our customers to fulfill their branding needs.

This flexibility is achieved by:

- Placing all design information into cascading style sheets (CSS) instead of writing it directly into the code. As far as possible, images are defined with the CSS attribute background-image.
- Using only central CSS in all HTMLB controls.
- Shipping predefined design variants (color templates) of the portal among which our customers can choose.
- Providing the Style Editor tool for supporting branding activities at the customers' sites.

Below you see the portal with the *Mango and Polarwind* standard design and the same portal with a customized design.

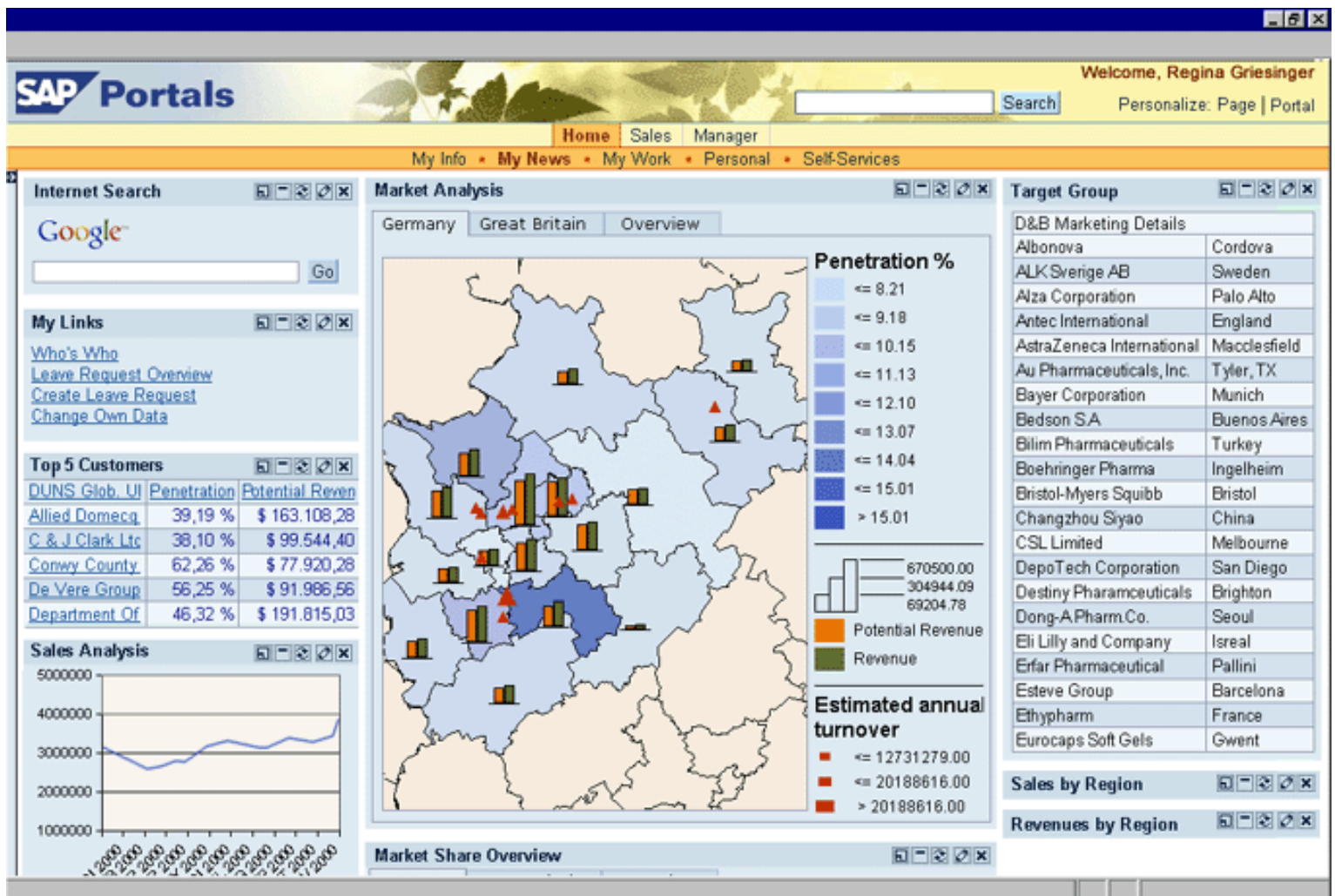


Figure 1: Portal standard design Mango and Polarwind

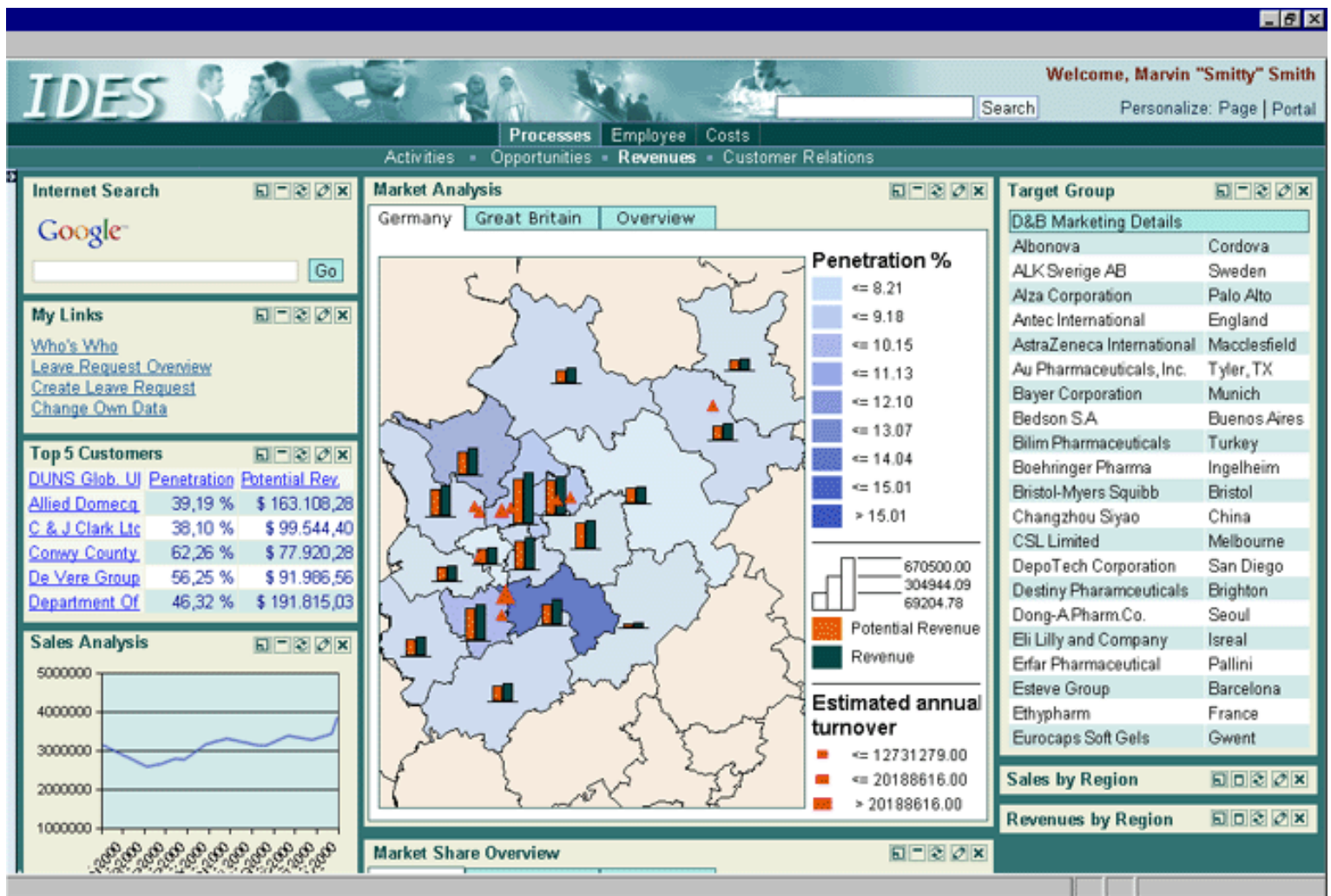


Figure 2: Example of a customized design

[Top](#)

Style Editor

The Style Editor is a Web-based tool, which allows a designer or administrator to copy and then modify any of our predefined color templates to create a new design. With the Style Editor, an authorized user can change the look-and-feel of the portal without having to be an HTML expert. For example, no knowledge about CSS attribute names is required. Below, you see the entry screen of the Style Editor, where users can select between predefined and customized designs, provided the customer created such.

Design Selection

Select the design you want to view, use as template, or want to edit. Editing is possible for your own designs only.

Your Designs: No designs to display

Color Templates:

- Text - Example
Sample text
[Hightide and Rosewood](#)
- Text - Example
Sample text
[Projector-safe](#)
- Text - Example
Sample text
[Mango and Polarwind](#)
- Text - Example
Sample text
[Indigo and Meadow](#)
- Text - Example
Sample text
[Dusk and Mocha](#)
- Text - Example
Sample text
[Highcontrast](#)
- Text - Example
Sample text
[Mint and Sage](#)

Related Tools:

- [ITS Style Editor](#)
- [Style Archive](#)

Figure 3: Design selection screen showing all available color templates

A clearly defined number of styles, such as the background colors, font colors, or images are presented on the user interface. Users can immediately check their changes in the preview area.

The Style Editor creates the CSS files for all platforms and browser versions that SAP Portals supports. Note that style sheets cannot be edited directly. The following example shows the user interface for changing the look of the tabstrip control.

The screenshot shows the SAP Portals Style Editor interface. The top navigation bar includes 'My Pages', 'Home', 'Content Management', 'Portal Admin', 'Content Admin', 'KM Admin', 'Portal Monitoring', and 'System Configuration'. The current design is 'Indigo and Meadow [read-only]'. The left sidebar lists various elements like 'Portal Outer Frame', 'Common Styles Content Area', 'Complex Elements', 'Simple Elements', and 'Special Backgrounds'. The main area shows a preview of a 'Tabstrip' control with two tabs, 'Tab 1' and 'Tab 2', and a 'Standard Text' area. Below the preview is a 'Styles: Tabstrip' section with a table of style properties and their values.

Tabstrip Styles			
Selected Tab Background Color	#E7F2DE	Options	Info
Inactive Tab Background Color	#D6E5CB	Options	Info
Left Border for Inactive Tabs	1px solid #B2D3A4	Options	Info
Right Border for Inactive Tabs	1px solid #B2D3A4	Options	Info
Top Border for Inactive Tabs	1px solid #B2D3A4	Options	Info
Tab Padding	1px 10px 1px 10px	Options	Info
Common Styles Used for Tabstrips (Changes Affect Other Elements)			
Container Border	1px solid #97AE90	Options	Info
Top Border of Container	1px solid #97AE90	Options	Info

At the bottom of the style editor, there are buttons for 'Save', 'Save As', 'Undo', 'Redo', and 'Reset'.

Figure 4: Customizing the tabstrip control

Note: Customer branding with the Style Editor works for central CSS only. Imagine that a customer wants to change the light blue color of the standard design to a light yellow all over the portal content area. If you as a developer defined an area with a light blue background directly in your code, the customer has no chance to change this color. Therefore, use central rendering mechanisms only.



[Top](#)

HTMLB Controls and Style Editor

The look of most visible HTMLB controls can be adapted with the Style Editor. See section *Editability in Style Editor* on the *More Info* page for the respective HTMLB controls.

Controls that use common styles only, such as the standard font color, are not presented in the Style Editor. Examples for these controls are the checkbox, the dropdown list box, and the radio button. The look of these controls can be customized by changing common styles, such as text, links, or cursor definitions.

Browser Platforms

There is a difference with respect to which attributes can be edited or not on different browser platforms. In general, the options for Netscape 4.7 are more restricted than those for other Web browsers.

The following controls cannot be changed for Netscape 4.7 as target platform:

- **Cursor:** Cursor for clickable and non-clickable elements
- **Input Field:** Background color of editable and non-editable input fields
- **Button:** Buttons have the default HTML look (gray and 3D); only the font type and size can be changed via common styles
- **Text Edit:** The default HTML element is used; only the font type and size can be changed via common styles

For details with respect to specific controls, see section *Editability in Style Editor* on the *More Info* pages.

Common Styles

There are so-called common styles, which affect more than one control in the content area. We list these styles here, in order to avoid redundant lists for each control.

While for Internet Explorer 5 and above, the common styles affect the controls **cursor**, **input field**, **link** and **text**, for Netscape 4.7 common styles affect only **link** and **text**.

Control	Style	IE 5 and above	Netscape 4.7
Cursor	Cursor for Clickable Elements	x	
	Cursor for Non-Clickable Elements	x	
Input Field	Background Color for Editable Fields	x	
	Background Color for Non-Editable Fields	x	
Link	Font Color for Unvisited Links	x	x
	Text Decoration for Unvisited Links	x	x
	Font Color for Active Links	x	
	Text Decoration for Active Links	x	
	Font Color for Links on Mouseover (Hover)	x	
	Text Decoration for Links on Mouseover	x	
	Font Color for Visited Links	x	x
	Text Decoration for Visited Links	x	x
Text			
Text Styles	Standard Font Family	x	x
Standard Text	Standard Font Size	x	x
	Standard Font Color	x	x
	Standard Font Style	x	x
	Standard Font Weight	x	x
Non-Standard Text	Font Size for Small Text	x	x
	Font Size for Large Text	x	x
	Font Size for Extra Large Text	x	x
	Font Style for Text Used as a Reference	x	x
	Font Color for Headlines	x	x
	Font Weight for Headlines	x	x
	Font Weight for Emphasized Text	x	x

Table 1: Common styles for controls and different browsers platforms

Forms - Using Checkboxes

[Alignment](#) | [Dependent Fields](#) | [Related Controls](#)

Include the following search sources

SAP R/3 Intranet Lotus Notes
 Google Yahoo HotBot

Checkboxes offer one or multiple choices to the user. The user can select none, one, or as many options as desired in a group of checkboxes.

Figure 1: A checkbox group

This page covers the arrangement of checkboxes, that is, their spatial and dependency relation to fields and other elements in form-like structures. For details on the checkbox control itself, see [Checkbox](#).

Checkbox groups offer users a set of multiple options that may be arranged either horizontally (2-3 checkboxes), vertically (not more than about 12 checkboxes), or in a matrix-like fashion. Note that checkbox groups are appropriate for static and relative small numbers of options only. Use the [table view](#) for larger and dynamic option sets.

Arrangement

For the alignment of checkboxes we distinguish the following cases:

- Checkboxes that refer to adjacent fields
- Checkboxes that do not refer to elements but should be included in field groups
- Checkboxes that can be arranged as an independent block of information

Case 1: Checkboxes that Refer to One or More Fields

Align dependent checkboxes with the left border of other input elements (figure 1a-b). Place the checkbox labels right to the checkbox (done automatically for the checkbox control).

Name
 Password
 Save password
 Phone number
 ← Alignment of Checkboxes

First name *
 Last name *
 Gender
 City
 Street
 Same address as last order
 ← Alignment of Checkboxes

Figure 1a-b: Checkbox that refers to one input field above it (left) or to two (City and Street, right)

Alternatively, a single checkbox can be placed to the right of a reference field (figure 1c) if space permits. If there is more than one reference field place the checkbox right to the bottom reference field.

A light blue rectangular area containing a form. On the left, there are three vertically stacked input fields with labels 'Name', 'Password', and 'Phone number' to their left. To the right of the 'Password' field, there is a checkbox followed by the text 'Save password'.

Figure 1c: Checkbox that refers to an input field left of it (equivalent to figure 1a)

Case 2: Checkboxes that are Included in a Field Group

If checkboxes are included in a field group but do not refer to a certain field, place the checkbox labels to the left and align the checkboxes themselves with the other input fields (figure 2a).

Note: In this case you have to set the checkbox text to an empty text and use the [label](#) control for the label.

Alternatively, you can add a label that is left-aligned with the other labels of the group and use the checkbox text for additional information (figure 2b). In that case, only the first checkbox should have a label that describes the whole group.

A light blue rectangular area containing a form. On the left, there are five vertically stacked labels: 'Name', 'Password', 'Save password', 'Phone number', and 'Connect automatically'. To the right of 'Name', 'Password', and 'Phone number' are input fields. To the right of 'Save password' and 'Connect automatically' are checkboxes.

A light blue rectangular area containing a form. On the left, there are five vertically stacked labels: 'First name', 'Last name', 'Contact', 'City', and 'Street'. To the right of 'First name' and 'Last name' are input fields with an asterisk to their left. To the right of 'Contact' are two checkboxes labeled 'Mail' and 'Fax'. To the right of 'City' and 'Street' are input fields.

Figure 2a-b: Checkbox within a field group, either with label to the left (left), or with two labels

If space permits you can alternatively use a horizontal checkbox group that occupies one line (typically for 2-3 checkboxes, figure 2c)

First name *

Last name *

Contact Mail Fax

City

Street

Name

Password

Connect automatically

Phone number

Wrong alignment because the checkbox does not refer to the password field

Figure 2c-d: Horizontal checkbox group within a field group (left); indented checkbox without group label (right)

Note: Do not use an arrangement without a group label in this case (figure 2d) because it may lead to misinterpretations. Such a layout suggests a dependency from the field above the group to the user. Even though the layout in figure 2d is the same as in 1a, the usage is incorrect because the checkbox does not refer to the password field. Therefore, use it only if such a dependency does exist (case 1).

Case 3: Checkboxes that Form an Independent Information Block

If checkboxes are arranged in a checkbox group, they are left aligned with other labels and arranged in a matrix-like fashion. Such groups have either to be included in a [group](#) control (see figure 3a) or separated from the field group by white space (figure 3b).

Include the following search sources

SAP R/3 Intranet Lotus Notes

Google Yahoo HotBot

Name

Password

Phone number

Save password Connect automatically

Figure 3a-b: Checkbox group that forms an information unit of its own - either included in a group (left) or separated by an empty line (right)

Instead of a matrix, you can use a horizontal arrangement if there are only few checkboxes. In this case, set the **columnCount** attribute of the the checkbox group control to a value that results in one row only.

There are two possible arrangements for horizontal checkbox groups:

- The checkbox row can be introduced by a label to the left - in this case align the checkboxes with other elements and use the label control for the label (figure 4a)
- The checkbox row does not have an introductory label (figure 4b)

Separate the horizontal checkbox group from preceding fields by an empty line.

Note: An "extreme" case of a horizontal checkbox group is a single checkbox.

First name *

Last name *

City

Street

Contact Mail Fax

Name

Password

Phone number

Save password Connect automatically

Figure 4a-b: Example of a horizontal checkbox group, either with an introductory label to the left (left), or without an introductory label (right).

For more than two to three checkboxes a vertical arrangement with or without label may be more appropriate (see figure 5a and 5b).

First name *

Last name *

City

Street

Contact Mail
 Phone
 Fax
 E-mail

Name

Password

Phone number

Save password
 Connect automatically
 Disconnect of idle for 10 minutes

Figure 5a-b: Example of a vertical checkbox group, either with an introductory label to the left (left), or without (right)



[Top](#)

Dependent Fields

In some cases, the state of input fields, dropdown list boxes, or other controls may depend on the setting of a checkbox. Below we present a simple example where users may enter their contact preferences (figure 6a). An unchecked checkbox describes the default case; if it is set the input field below it is read-only. A checked checkbox describes the less frequent case. If the user checks the checkbox, the input elements are ready for input.

First name *

Last name *

Contact Mail

City/Zip Code

Street

E-mail

Figure 6: Checkboxes that controls the editability of the input field(s) below the checkbox

If there are more dependent elements indent the dependent group so that their labels are left aligned with other input fields (top checkbox). If there is only one dependent field, usually a field label is not needed (bottom checkbox).



Related Controls

[Radio Button](#), [Dropdown List Box](#), [List Box](#), [Label](#), [Grid Layout](#)



Forms - Using Radio Buttons

[Alignment](#) | [Dependent Fields](#) | [Design Alternatives](#) | [Related Controls](#)

Select your payment method

- Invoice
- Credit Card
- Personal Check

Radio buttons provide users with a single choice from a set of alternative options

Figure 1: A radio button group

This page covers the arrangement of radio buttons, that is, their spatial and dependency relation to fields and other elements in form-like structures. For details on the radio button control itself, see [Radio Button](#).

Radio button groups offer users a set of alternative choices that may be arranged either horizontally (2-3 radio buttons), vertically (not more than about 12 radio buttons), or in a matrix-like fashion. Note that radio button groups are appropriate for static and relative small numbers of options only. Use the [table view](#) for larger and dynamic data sets.

Alignment

For the alignment of radio buttons we distinguish the following cases:

- Radio buttons that refer to adjacent fields
- Radio buttons that do not refer to elements but should be included in field groups
- Radio buttons that can be arranged as an independent block of information

In general, the first two cases are not as common as for checkboxes.

Case 1: Radio Buttons that Refer to One or More Fields

Align dependent radio buttons with the left border of other input elements (figure 1). Place the radio button labels right to the radio button (this is done automatically for the radio button control).

Age

Profession

Active

Retired

Nationality

City

Street

Figure 1: A pair of radio buttons that refers to the input field above it (profession)

Note: If there are too many alternatives, consider using a dropdown list box instead of the radio button group.

As noted below, this arrangement should only be used if there is a dependency from other fields above the radio button group.

Case 2: Radio Buttons that are Included in a Field Group

If radio buttons are included in a field group but do not refer to a certain field, do the following (figure 2a-b):

- Place the description of the radio button group to the left of the group and align it with the other field labels
- Align the radio buttons with the other input fields
- Use a label control for the group label

You can use a vertical radio button group, or if space permits a horizontal radio button group that occupies only one line.

The figure shows two side-by-side form panels. The left panel has a vertical radio button group for 'Gender' with 'Female' selected. The right panel has a horizontal radio button group for 'Gender' with 'Female' selected. Both panels include input fields for 'First name', 'Last name', 'City', and 'Street'.

Figure 2a-b: Radio buttons within a field group with group label to the left (left shows vertical, right shows horizontal arrangement)

In general, radio button groups within a field group should have a descriptive label for the group and a label to the right of each radio button.

Rationale: A radio button group is functionally equivalent to a dropdown list box. The group label corresponds to the label for the dropdown list box; the labels to the right of the radio buttons correspond to the list box items.

The figure shows two side-by-side form panels. The left panel has 'Female' and 'Male' labels to the left of the radio buttons. The right panel has 'Female' and 'Male' labels to the right of the radio buttons. Both panels include input fields for 'First name', 'Last name', 'City', and 'Street'.

Wrong level of labels (male/female is are subcategories)

Wrong alignment because the radio buttons do not refer to the name fields

Figure 2c-d: Radio buttons within a field group with two labels to the left of the radio buttons (left) and to the right (right)

Do Not

- A layout without a label for the radio button group and with labels to the left (figure 2c) is harder to understand because the labels are not on the same semantic level as the surrounding field labels.
- Also do not use an arrangement without a group label in this case (figure 2d) because it is equally hard to understand and may lead to misinterpretations. Such a layout suggests a dependency from the field above the group to the user. Even though the layout in figure 2d is the same as in 1a, the usage is incorrect because the radio buttons do not refer to the "Last name" field. Therefore, use it only if such a dependency does exist (case 1).

Case 3: Radio Buttons that Form an Independent Information Block

If radio buttons are arranged in a separate radio button group, arrange them in a matrix-like fashion and left-align them with other elements on the page or in the application. Such groups have either to be included in a [group](#) control (see figure 3a) or separated from the field group by white space (figure 3b).

The image displays two examples of radio button groups. On the left, a light blue box contains the heading "Select your payment method" and three radio button options: "Invoice" (which is selected), "Credit Card", and "Personal Check". On the right, a larger light blue box contains a form with four text input fields: "First name", "Last name", "City", and "Street", each with an asterisk indicating it is required. Below these fields is a "Gender" section with two radio button options: "Female" (selected) and "Male".

Figure 3a-b: Radio button group that forms an information unit of its own - either included in a group (left) or separated by an empty line (right)

Instead of a matrix, you can use a horizontal arrangement if there are only few radio buttons. In this case, set the **columnCount** attribute of the the radio button control to a value that results in one row only.

There are two possible arrangements for horizontal radio button groups:

- The radio button row can be introduced by a label to the left or a header above - in this case align the radio buttons with other elements and use the label control for the label or header (figure 4a-b)
- The radio button row does not have an introductory label (figure 4c-d)

Separate the horizontal radio button group from preceding fields by an empty line.

First name *
 Last name *
 City
 Street
 Gender Female Male

Mrs. Mr. Company
 First name *
 Last name *
 City
 Street

Figure 4a-d: Horizontal radio button groups, either with an introductory label to the left (top left), a heading (bottom left), or without an introductory label (right)

Note: Do not use a single radio button because there are two problems with it: (1) Users cannot deselect a single radio button; after it has been selected, it remains so. (2) Users see the name of one option only; they often can only guess what the alternative option is.



Top

Dependent Fields

In some cases, the state of an input field, dropdown list box, or other control may depend on the setting of a radio button group. Below we present two simple examples, where users may either enter their nationality (figure 5a), or their payment method (figure 5b). The first radio button describes the default case; if it is set, the input fields below it are read-only (see [this state](#)). The second radio button describes the less frequent case; if it is set, the dependent input fields are ready for input. Alternatively, in the first example a dropdown list box could be used instead of the input field if the alternatives are known and their number is not too large.

First name *
 Last name *
 Nationality American
 Other

 City
 Street

First name *
 Last name *
 City
 Street
 Payment method Invoice
 Credit Card
 Card
 Number
 Expired

Figure 5a-b: Vertical radio button group that controls the editability of one (left) or several (right) input fields below the group

Do not use this layout for more than one dependent element. If there are more dependent elements indent the dependent group so that the labels are left aligned with other input fields (figure 5b).



Design Alternatives

Radio buttons are similar in function to [dropdown list boxes](#) and [list boxes](#) with respect to offering users a single choice. Use radio buttons for very small item numbers (2-6) and if the users should immediately see all alternatives.

See [Forms - Using Different List Types](#) for hints on when to choose between those controls.



Related Controls

[Dropdown List Box](#), [Checkbox](#), [List Box](#), [Label](#), [Grid Layout](#)



Forms - Using Different List Types

[Design Options for Lists](#) | [Dropdown List Box vs. List Box](#) | [Item List](#) | [Related Controls](#)

Design Options for Lists

HTMLB offers several controls for displaying and editing data sets and for selecting from data sets.

- **Checkbox:** multiple selection from small data sets (static)
- **Radio Button:** single selection from small data sets (static)
- **List Box:** Single selection from small data sets (static)
- **List Box:** Single selection from small to medium data sets (dynamic or static)
- **Item List:** Display of small to medium data sets in one column - ordered or unordered (static)
- **Table View:** Display and editing of data sets in a variety of display variants - in one or more columns (dynamic), single- or multiple-selection possible

For information on the respective controls themselves, see [Checkbox](#), [Radio Button](#), [List Box](#), [Item List](#), and [Table View](#).

Below we discuss the design options in more detail, especially with respect to overlapping application areas.

Note: The cases of checkbox and radio button groups are not discussed here; see [Forms - Using Checkboxes](#) and [Forms - Using Radio Buttons](#) for details on the layout options for these controls.



[Top](#)

Dropdown List Box vs. List Box

The overview above showed that a list box is similar in function to a dropdown list box - both offer a list of items where users can select one item from, that is, both are single-selection lists. Here you find criteria for choosing between both controls. In addition, we provide hints when a set of radio buttons is the appropriate choice.

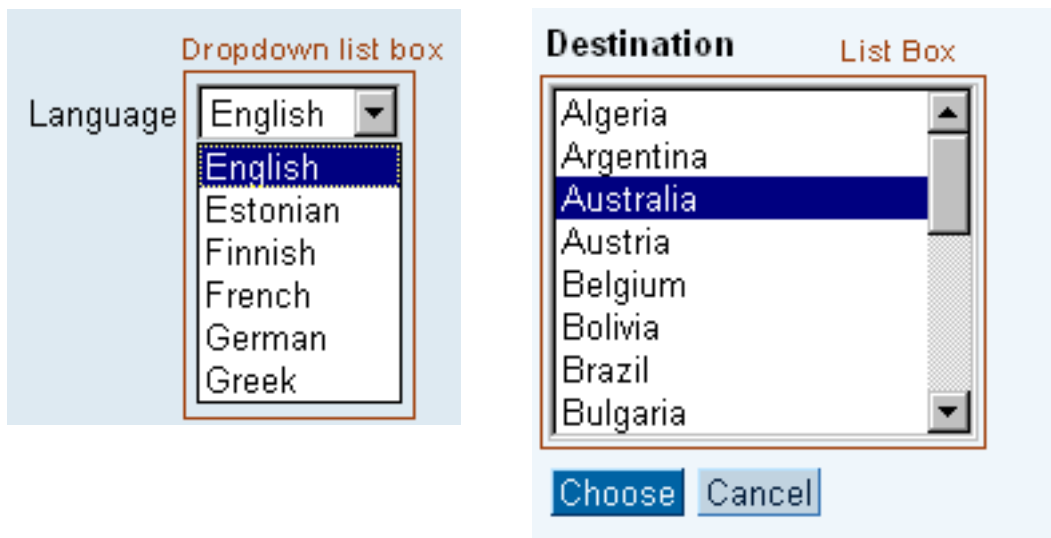


Figure 1: Dropdown list box (left) vs. list box

When Use a List Box?

- If there is more space on the page
- For larger item numbers
- If users need to know the context of the current selection, that is, the item set or at least part of it
- If users need to carefully consider their choice
- For users with low mouse abilities

When Use a Dropdown List Box?

- In table views
- If space is limited - it occupies one line only
- For smaller item numbers (up to 20 items)
- If users only need to know the current selection, not the whole set
- In shufflers

When Use Radio Buttons?

For very small item numbers (2-6) and if the users should immediately see all available alternatives, use [radio buttons](#).

Use larger radio button sets only in special cases, where it is important that all options are visible, where users are untrained, and/or where an application imitates a paper-form, such as a Web-based questionnaire or ordering form.

Note: For multiple choices use checkboxes instead of radio buttons.



[Top](#)

Item List vs. List Box

Both item lists and list boxes can be used for displaying a set of options. While list boxes can also be used for selecting items we focus here on the display aspect. Item lists are static lists with a "paper-like" appearance; they can be ordered or unordered. List boxes, however, have a "form-like" appearance and also may contain more items than are visible.

If you consider to use a list box, you may check whether a table might also be a valid design option. We also provide some hints for this option.



Figure 2: Item list (ordered, left) vs. list box

When Use an Item List?

- In paper-like applications, such as news, articles, etc.
- If it is possible to display all items or if the page or application as a whole can be scrolled
- If the number of items is fixed

When Use a List Box?

- In form-like applications, typically together with other form elements, such as input fields and selection elements
- If a selection is needed
- If space may not suffice to display the whole list
- If the application or page cannot be scrolled in order to display more items
- If bullets or numbers should not appear

When May a Table View Be Used?

In the following, we list scenarios, where a table view may be used instead of a list box. Note that this is an option to consider, not a recommendation. The only exception is multiple-selection, which is available for the table view only.

Typically, you would use the table view in the **transparent** design for this application. Also note that a table view introduces "visual overhead", such as the title, column headers, and scroll buttons.

- In form-like applications where the data form a separate information unit that cannot be mixed with fields
- Where a multiple-column display is needed
- Where scroll buttons are preferred over scrollbars (page-wise scrolling)
- Where multiple-selection is needed (this is currently not possible with list boxes)

Note: Static multiple-selection can also be implemented using a [checkbox group](#).



[Top](#)

Related Controls

[Checkbox](#), [Item List](#), [Dropdown List Box](#), [List Box](#), [Radio Button](#), [Table View](#)



[Top](#)

Table View Functions

[Placement of Buttons for Functions](#) | [Functions Referring to Table View Columns, Sorting](#) | [Functions Referring to Table View Rows](#) | [Filtering, Shufflers](#) | [Related Controls](#)

Table View					
	Course	Course #	Location	Training Facility	Date
<input type="radio"/>	HTML Basics I	50000484	Walldorf	Training Center Walldorf	12/01/2001
<input type="radio"/>	HTML Basics II	50000485	Zürich	Training Center Zürich	12/01/2001
<input type="radio"/>	Web Design Beginners	50000486	Wien	Training Center Wien	12/01/2001
<input type="radio"/>	Web Design Advanced	50000733	Los Angeles	Training Center Los Angeles	12/01/2001
<input checked="" type="radio"/>	Java Basics	50000734	Philadelphia	Training Center Philadelphia	12/01/2001
<input type="radio"/>	Javascript Basics	50000736	Atlanta	Training Center Atlanta	12/01/2001
<input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="↕"/> <input type="button" value="↔"/>					1/3

Figure 1: Example of a table view with different column types and an erroneous input field

Table views not only present data in a tabular fashion, or allow for tabular data entry. They also provide a number of functions, part of which change the presentation of the data, while other functions may change the data themselves.

Table view functions may refer to the table view as a whole, to columns, to rows (= items), or to single cells. These cases are described below; in addition often-used functionalities, such as sorting and filtering are covered.

Note: Here, we do not cover functions that are included in the table view framework, such as scroll functions. See [Table View](#) for these functions.

For details on the table view control itself, see [Table View](#).



[Top](#)

Placement of Buttons for Functions

Location

Place buttons acting on a table view as a whole below the table view and left-aligned with the table. Place the emphasized button to the left if there is one.

Button Groups

Separate button groups by a spacer (non-breaking space).



[Top](#)

Functions Referring to Table View Columns, Sorting

Buttons Referring to Columns

Some functions, for examples *Sort*, refer to certain columns. Place buttons with small icons into the column headers to speed up interaction.

If there are alternative variants of a function (e.g. different sort orders or calculations), consider to display only one icon at a time in order to save space, instead of displaying two or more icons in parallel.

Do not use more than three icons in a column header.

Links in Column Headers

If it is evident what a function does, you can also use a link in the column header text.



[Top](#)

Functions Referring to Table View Rows

Functions referring to table rows typically refer to the item the data of which are displayed in a row. These functions can either be presented as

- Buttons,
- Icons, or
- Links

Use **buttons** for most functions. Use icons and links for the following exception cases:

- **Icons:** Web standards, such as *Delete*, *Info*, *Help*, or *Shopping Basket*
- **Links:** Navigation functions, such as *Details* (place the link in the name or ID column), *additional information*, etc.

As a general rule for selecting the correct control, care for the application context and the respective "Web standards".

Table Cells: Links vs. Buttons

Note that links are not self-explaining. Therefore, use links only where their purpose is evident. Add tooltips to links to support users.

See [Links vs. Buttons](#) for more information.



Filtering, Shufflers

Offer **filters** as much as possible. Filters help to reduce the amount of data displayed in a table view. This helps users and improves performance.

Use **shufflers** for creating filter statements. See the respective section on shufflers in the *iView Guidelines*.

Placement

Place the shuffler statement above the table view and left-align it.

Reason: A left-aligned shuffler is not hidden from view if the window size is altered.

Button

Use a button labeled *Go* to start the filtering process. Use events on dropdown list boxes for simple cases only (one dropdown list box with label).



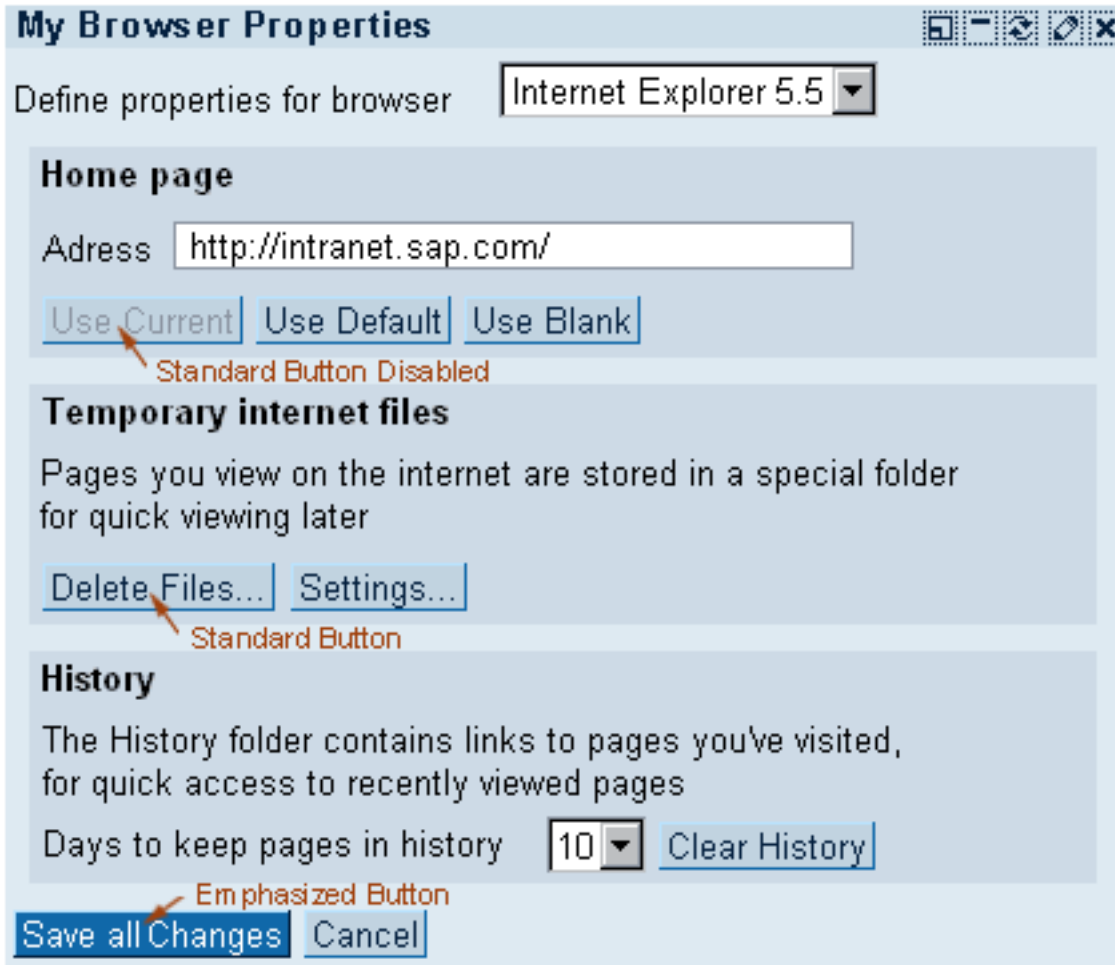
Related Controls

[Tree View](#), [Item List](#), [List Box](#)



Positioning Buttons

[Positioning](#) | [Overview of Positioning Rules](#) | [Related Controls](#)



Buttons are used for explicit functions that refer to a given object or serve for navigational purposes

Figure 1: Example of an iView containing groups with buttons and two buttons belonging to the iView itself

 [Top](#)

Positioning

Place buttons below the object they refer to. If space is scarce, place the buttons to the right of the object (for several objects place them to the right of the bottom object).

How to make clear which object(s) a button refers to:

- Place the buttons inside, or close to the object.

Example: Place buttons inside group boxes, place buttons close to the fields they refer to



Figure 2: Example of a button inside a group referring to a list box

- Left-align button and object.

Example: Left align buttons referring to a field with the field label

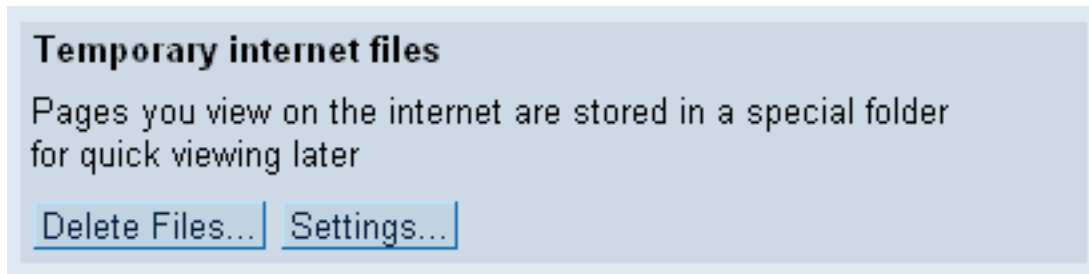


Figure 3: Example of left-aligned buttons inside a group

- Place button(s) on the same level (area, group box) as the objects which are affected by the action.
Example: Place buttons that refer to fields in a group box or area within that group box, or area (figure 2 and 3)
- Show/hide objects: When an object is hidden, buttons are also hidden.
Example: If a table is hidden, the related buttons are also hidden

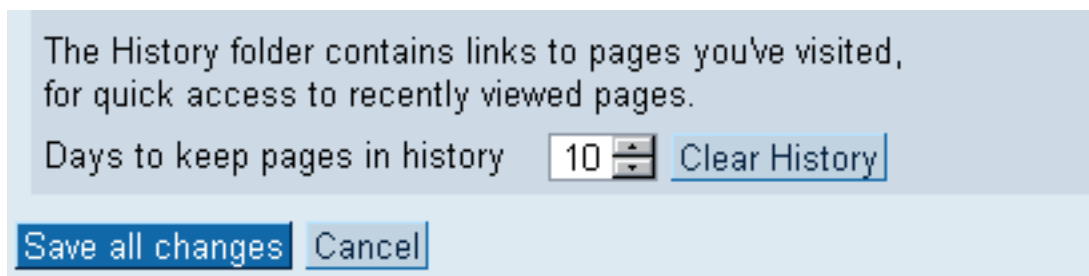


Figure 4: Example of a left-aligned button group containing an emphasized button

- If an emphasized button (see [Types](#)) is a member of a button group, it is the leftmost button in this group.
- Navigational buttons are placed at the bottom left of a screen (or screen area).



Overview of Positioning Rules

The following table summarizes the rules for button placement.


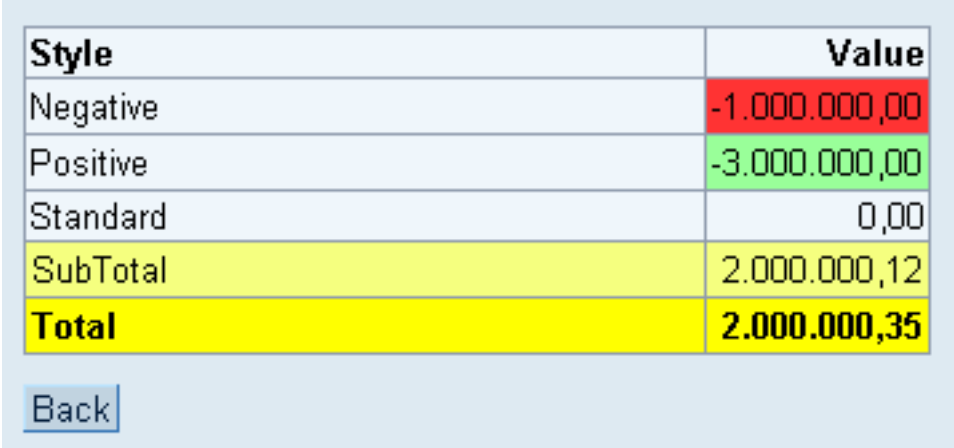
Object	Example	Placement
Single object	Field	Right to the object 
Several objects	Field group	Default case: left-aligned below the bottom object If space is scarce: to the right of the bottom object (see figure 4)

Figure 5: Button next to field

Area, tabstrip, group box	Group box	At the bottom, left-aligned (see figure 3)
---------------------------	-----------	--

Table View (fixed size)	Table based on Table View control or Portal Data Viewer	Below the table, left-aligned with the table
-------------------------	---	--



Style	Value
Negative	-1.000.000,00
Positive	-3.000.000,00
Standard	0,00
SubTotal	2.000.000,12
Total	2.000.000,35

Back

Figure 6: Button below a table

Special case: Long table	Scrolling table	Above and below the table, left-aligned Alternative: Implement a special frame for buttons above a table, which does not scroll.
--------------------------	-----------------	---

Table 1: Rules for button placement

Usage guidelines for the different button types and sizes are presented in [Button](#).



[Top](#)

Related Controls

[Link](#), [Input Field](#), [Group](#), [Table View](#)



[Top](#)

Error Handling

[Error Prevention](#) | [Error Handling for Fields](#) | [Error Handling in Tables](#)

Beside help, **error handling** is an important aspect of user support. Error handling helps users to overcome problem situations and to continue their work.

Typically, error handling is done by indicating the location where the error occurred and by sending an error message that notes the error, explains the reason for it and - ideally - provides hints on how to remedy the error situation.

For details on message texts see chapter *Formulating Messages* in the *SAP Reference Lists* on the *SAP Design Guild*.

This page covers three areas: (1) [error prevention](#), (2) [error handling for fields](#), and (3) [error handling in tables](#).

Error Prevention

Error Prevention Comes First!

Before handling errors, you should first ask how errors can be prevented. Generally, you should design iViews and Web applications so that errors cannot occur. Preventing errors - instead of remedying them - has the following benefits:

- Users cannot come into error situations - many users have problems with recovering from errors.
- The users' work is not interrupted by error messages.
- Users are not confused or puzzled by (often cryptic) error messages.
- There is no need for a screen area that display errors.

If it is not possible to prevent errors, follow the guidelines presented [below](#).

How You can Prevent Errors

Often it needs some rethinking and the giving up "old habits" to find design solutions that prevent errors instead of sending an error message **after** an error has occurred.

In the following we provide some ideas and examples that may stimulate your imagination when looking for ways how errors can be prevented.

Prevent Wrong or Invalid Inputs - General

- Use precise descriptions and instructions - do not be too short (especially for Web applications)
- Indicate required fields (through a red asterisk *) and an explaining text

Prevent Wrong or Invalid Inputs

- Numeric fields: Prevent users from entering letters by parsing the input string.
- Date and time fields: Provide "intelligent" date and time fields that are preformatted, or provide selection controls instead of input fields (dropdown lists, spin buttons, calendar controls).
- Currency fields: Use preformatted fields.

Prevent Incomplete Inputs

- Indicate required fields (through a red asterisk *) and an explaining text

Prevent Invalid Actions

- Disable buttons that cannot be used in the current context.
- Do not offer functionality that is not needed.

Prevent Disastrous Actions

- If actions can have severe consequences for the user, add explaining texts to the respective buttons and inform the users about the consequences
- Send dialogs if users can loose data

Use Controls in the Correct and Intended Ways

- Do not use screen elements where users expect to use them in any order, if there are dependencies or if a certain sequence of steps has to be followed.
Example: Do not use tabstrips for views that depend on each other and cannot be viewed at random. At best, do not force users to perform steps in a fixed sequence.
- In general, do not use controls in other than the intended ways. "Creative" use of controls clashes with the users' expectations and may lead to severe usage problems.
Example: Do not misuse checkboxes as radiobuttons just because you like the look of the checkboxes better.

Make the Page/iView and its Purpose Clear to the User

- Often important information is hidden while unimportant information dominates the page. In other cases users simply have no clue what an application's purpose is. Thus, provide the necessary information and arrange it so that relevant things are recognized first - this way users realize what to do on a screen and how.
- Use precise descriptions and instructions - do not be too short (especially for Web applications)

Error Handling for Fields

Set the field or fields where an error occurred to the error state (see input field) and place an error message as close to the field where the error occurred as possible (if there is more than one field, place the message at the first error field). Place the cursor into the (first) error field.

Avoid Popups!

Popups interrupt the users' work flow and thus annoy them.

Exception: You may use popups for severe errors like aborts that need direct user intervention.

Future Development

After validation of a field, the error message will appear in a line directly below the field. As this change in layout can be performed locally, there will be no major screen flicker.

iViews: In addition, iViews (trays) will have a status bar where a general error message will appear. This status bar may also display warnings and success messages (an icon will indicate the type of the message). The location of the status bar can be either below the title bar or at the bottom of the tray (**open**). The status bar may be hidden by the application.

Error Handling in Tables

Errors can appear in table views for different reasons. For example, a user may enter invalid data, or certain items from a set cannot be posted. These cases have to be handled differently.

Input Errors

If a user enters invalid data, highlight the erroneous fields and scroll the table to the first field where an error occurred.

If an error message is needed, place it below the table view or - if possible - in a table row directly below the row where the error(s) occurred.

Future Development

Table views will have a status bar, where the error message will appear. Place the cursor into the error field and scroll the table to make the field visible in case it is hidden from view.

If there is more than one error field, display the message for the first error field, place the cursor into that field and scroll the table to make it visible if necessary.

If the cursor is placed into a subsequent error field, display the message for the respective field. If an error is corrected move the cursor to the subsequent error field if there is one and display the respective error message.

If the focus is outside the table view, display the first error message again.

iViews: In addition the planned status bar of an iView (tray) may display a general error message.

Posting Errors

Posting errors often do not require to cancel the whole posting process. It is only necessary to correct and re-post those items that were erroneous. Therefore, redisplay the table view with the erroneous items only and provide the user with a possibility to correct the items. Place an error message above the table.

Future Development

Place the error message inside the status bar of the table view.



[Top](#)

Related Controls

[Flow Layout](#), [Grid Layout](#)



[Top](#)

Accessibility of HTMLB Controls

[General Information](#) | [References](#)

General Information

This page offers general information for application developers using HTMLB who want to make their Web applications accessible. For details see section *Accessibility* on the *More Info* page for the respective controls.

Most accessibility features are already provided by the central rendering engine of HTMLB. Therefore, application developers only have to add those features that cannot be provided by default.

As an application developer, keep in mind that you cannot affect page elements on the basis of HTML tags or attributes. The only interface to the HTMLB controls is the HTMLB programming interface -- the HTMLB attributes and methods for the respective controls.

Examples

- Application developers cannot set the *title* attribute of elements in order to extend descriptions, they have to use the *setTooltip* method, instead.
- They also cannot set the *summary* attribute of tables, they have to use the *setSummary* method provided by HTMLB.

Descriptions

The central HTMLB rendering engine already provides general descriptions for HTMLB controls, such as the type, the state, and on-screen text. Therefore, application developers only have to complement descriptions in case that users need more specific descriptions or instructions. The descriptions written by the application developers are added to the default descriptions that are provided by the central rendering mechanism.

Examples

- A button description has to be extended if a button opens a new window.
- In general, a description has to be extended if a button introduces an interaction that cannot be recognized by a blind user.

Accessibility Flag

Also note that the resulting description that is sent to the users depends on the state of the accessibility flag:

- If the accessibility flag is **set**, the default description is extended by the description that the application developer provided.
- If the flag is **not set** only the description that the application developer provided is sent to the user.

Keyboard Accessibility

As application developers cannot set HTML attributes directly, they do not have access to the *tabIndex* attribute of elements. Consequently, application developers cannot add elements to the accessibility hierarchy themselves in order to make them keyboard accessible.

Input Elements and Corresponding Labels

Input elements, such as checkboxes, dropdown listboxes, input fields, radiobuttons, and text edit controls need to be connected to a label, so that screen readers recognize the association of the label with the input element. Use the HTMLB label control for this purpose (use method *setLabelFor* for identifying the corresponding control).

The connection between a label and its corresponding input element also simplifies the interaction with the element when using the keyboard or mouse.



[Top](#)

References

- SAP Portals Accessibility Guidelines
- API Java Docs



[Top](#)

General Layout Strategy

[Structure of the Layout Section in these Guidelines](#) | [General Page Layout Aspects](#)

This page describes a general strategy for layouting Web pages, applications, and iViews. Layouting a page is not just "throwing" controls on a page. Several aspects have to be considered, such as

- Flow of control - how the user progresses through a page when doing his or her work
- Dependencies - how elements on a page affect each other
- Togetherness - which elements on a page belong to each other, there may be closer and farther relations between elements
- Aesthetics and general Gestalt principles - how information can be effectively communicated visually

There are three steps in layouting - these can be done in the following sequence: Determine the ...

1. Sequence of elements (vertical, horizontal)
2. Nesting of elements
3. Spacing between elements at different hierarchy levels.

The **sequence** takes care of the flow of control, dependencies, and information about which elements belong together - the latter in a more linear fashion. The **nesting** also takes care of dependencies and of togetherness -- but in a hierarchical or top-down fashion. The **spacing** takes care for aesthetics and the proper application of Gestalt principles (mostly togetherness).



[Top](#)

Structure of the Layout Section in these Guidelines

This page covers **general** layout aspects, such as the roles of sequence, nesting and spacing. [Layout Hierarchy](#) covers the detailed **nesting**, that is, which objects have to be on the same level and which can be nested. The pages on [Flow Layout](#), [Grid Layout](#), and the pages on **spacing** ([single](#) and [grouped](#) controls) cover the details of spacing.



[Top](#)

General Page Layout Aspects

The Role of Sequence

The sequence of elements should typically be determined by the **flow of control**, that is, the way how users perform their tasks. Often, however, a task may not be linear or users have to step back because of errors. Here, the page designer has to find a "natural" sequence that fits most users and scenarios.

In addition, conventions, such as the reading direction, play an important role for the arrangement of elements. For Western cultures, the typical arrangement of elements is from left to right and from top to bottom, just like the reading direction. **Dependencies** are also typically communicated this way. "First things first" is also a motto, which expresses that there is a "natural progression" in most things we do. For example, when entering a customer's address we start with the name, which is the main information that determines the remainder of the information - we do not start with the street and house number, even though one

might infer the customer's name from that information.

Such a rule may be natural to everybody and most designers follow it without even thinking about it. Problems occur, however, if this rule is **broken**, and the flow or dependencies go into the opposite direction. Such reversals often present severe obstacles for users.

Arranging elements on a page is the first step in page design. This can also be done in a prototypical fashion and tested with users (for example with paper prototypes) without worrying for the details of the page design.

The Role of Nesting

There are two basic ways to visually indicate the relation between elements - closeness and nesting. **Closeness** means that objects, which are located closely together, are perceived as more closely related than objects that are farther apart from each other. Closeness of elements is typically combined with direction to indicate flow of control or dependencies. For example, first you enter a value into a search field (left) and then you click the related Go button next to it (right).

Nesting is used to indicate more complex hierarchical relations and dependencies between objects. Nesting is also a way to hide details from users because users can first deal with the high-level objects and then decide, which one they want to inspect more closely.

Nesting can make pages much more complex than simple sequencing of elements because nesting requires the introduction of borders or other visual separators that may clutter pages visually. Therefore, nesting rules have been established that aim to prevent the creation of overly complex pages (see [Layout Hierarchy](#) and the respective controls). Spacing can help to reduce the cluttering effect but often requires more space than is available.

Nesting can also be explored in a prototypical fashion (paper prototypes, HTML prototypes); here, the prototype may already be more detailed than in the initial phase.

The Role of Spacing

Spacing is very important in communicating which elements belong together; it also affects readability and the ability of users to recognize information on a page.

In general, application developers should not need to bother with the details of spacing, that is, with how many pixels they have to insert between, for example, a button and the border of a group. There are two HTMLB controls, the [grid layout](#) and the [flow layout](#), which take care for the exact spacing. In addition, containers, such as the tray and the group, also care for the outer spacing.

Note: Currently, the spacing controls do not work as intended. Therefore, developers should consult the pages on the [grid layout](#) and on the [flow layout](#) for the limitations of these controls.

Only high-level prototypes that intend to offer a realistic preview of a final page need to bother with detailed spacing.



[Top](#)

Related Controls

[Flow Layout](#), [Grid Layout](#)

Layout Hierarchy

[From Containers to the Layout Hierarchy](#) | [Layout Hierarchy for iViews and Web Applications](#) | [Table Overview of the Layout Hierarchy](#) | [General Nesting Rules](#) | [Related Controls](#)

This page describes the layout hierarchy of Web pages, which defines the options for **nesting** page elements. In short, this page tells designers, which page element can be placed into which container element - including placing containers inside containers.

The layout hierarchy is the basis for establishing textual layout rules for pages and page sections. The main goal of such rules is to prevent overly complex and visually cluttered pages caused by excessive nesting.

Note: These rules do not comprise the **spacing** between and within elements.



[Top](#)

From Containers to the Layout Hierarchy

Page elements can either be containers or non-containers. Containers can contain other elements, non-containers not. The layout hierarchy described below basically deals with container elements, that is, with elements that can contain other elements including other containers. This is critical because too much nesting can let a page appear visually overloaded.

Application Containers

At the root of the layout hierarchy there is a "root" container that contains the application. In the Web or portal environment, there are two cases to consider:

- The application container is a simple **background**, such as a frame or window. This is the case for the so-called Web applications, including the portal administration applications
- The application container is a **tray** or tile. The container which may have elements and controls on its own; the application that resides inside this container may use the services of the container. From the application's point of view the container is all it knows about -- at least from a design perspective.

Container Controls Inside Applications

Inside an application, container controls define the layout hierarchy of an application. Such containers are:

- Areas (Web applications only)
- Tabstrips
- Groups
- Subgroups (group of simple elements with or without heading - not included in a group control)

A Matter of Interpretation - Linear Sequences vs. Sequences of Containers

From a technical point of view, not all of these containers are "real" containers. Areas are subdivisions of an application. That is, areas form a linear sequence within an application. Subgroups are groups of simple page elements that may be introduced by a heading. Typically, they are separated from the remainder of the page by whitespace or separator lines.

From a layout point of view, however, it is easier to view areas and subgroups as "real" containers - for the layout process this does not make any difference. The advantage of regarding these elements as real containers is that a layout can be expressed in a hierarchical or treelike fashion, which makes it easy to gain an overview of the page or application structure.

Non-Containers

While containers create the "skeleton" of a page, non-containers are the "flesh" of a page. These elements are fields, buttons, selection elements, text units, and tables. As these elements differ in complexity, nesting rules ensure that a page cannot become too complex. For example, table views are similar in complexity to groups and tabstrips. Therefore, they are placed on the same level in the layout hierarchy as these containers. A respective rule that takes this aspect into account would state that table views may not be placed into groups or tabstrips if they are the only control that is inside the container.

Separators

Separators, such as line or whitespace "separate" elements or containers. Therefore, they are difficult to integrate into a hierarchical model of a page layout. They can be viewed as "concluding elements" or "borders" of containers (they are easier to integrate into a "linear" model of the layout).

Note that separators are different. While you would separate containers or elements that are on the same hierarchy level with whitespace, you would use lines because that would introduce unnecessary framing.

Creating the Layout Hierarchy

The layout hierarchy is created by placing containers and simple elements on a page. The rules presented below govern how page elements can be combined, either by sequencing them vertically or horizontally, or by nesting.

Containers may contain containers (nodes), simple elements (leaves), or both. In addition, non-containers may reside on the same hierarchy level as containers. But they are "end nodes" and do not continue the layout hierarchy.

Example: A table view may reside on the same level as a group or a tabstrip

The layout rules presented below specify:

- Which containers may contain which other container(s) - including itself
- The specific conditions for the nesting, for example, alone or together with other elements
- How many levels deep the nesting may be
- Which simple elements may be placed into which container - and the specific conditions for this
- Which containers and which simple elements are on the same hierarchy level



[Top](#)

Layout Hierarchy for iViews and Web Applications

Depending on the container elements used, different application types can have different layout hierarchies. In the case of the portal environment, there are Web applications and iViews. Both application types use different containers, serve different purposes, and therefore differ in complexity with respect to the layout.

iView

- Tray = iView container
 - Tabstrip - may contain:
 - Group (if it is not the only element)
 - Subgroup
 - Table View (if it is not the only element)
 - Simple Elements
 - Separators
 - Group - may contain:
 - Group (if it is not the only element, different group types only)
 - Subgroup
 - Table View (if it is not the only element)
 - Simple Elements
 - Separators
 - Subgroup - may contain:
 - Simple Elements
 - Separators
 - Table View
 - Simple Elements
 - Separators

Generally, there should not be more than one level of nesting within trays/iViews. Also note that tabstrips may not be nested.

Simple elements are: input fields, selection elements, text, buttons, ...

Note: A similar tree can be created for real iViews based on the elements used.

Web Application

- Application Background = Window/frame background = application container
 - Area - may contain:
 - Tabstrip - may contain:
 - Group (if it is not the only element)
 - Subgroup
 - Table View (if it is not the only element)
 - Simple Elements
 - Separators
 - Group - may contain:
 - Tabstrip (if it is not the only element)
 - Group (if it is not the only element, different group types only)
 - Subgroup
 - Table View (if it is not the only element)
 - Simple Elements
 - Separators
 - Subgroup - may contain:
 - Simple Elements
 - Separators
 - Table View
 - Simple Elements
 - Separators
 - Single elements??? - **Open**

Generally, there should not be more than one level of nesting within Web applications. Also note that tabstrips may not be nested.

Simple elements are: input fields, selection elements, text, buttons, ...

Note: A similar tree can be created for real Web applications, based on the elements used.

Note: The critical question for Web applications is, whether single elements and containers other than areas can be placed on the application background. Currently, the application background may not be used for non-container elements (see the *IAC Guidelines* in the *SAP Design Guild*). In R/3 applications, header data may be placed on the application background; there is no such a container concept in R/3 applications as areas.



[Top](#)

Table Overview of the Layout Hierarchy

The following table overviews present a more detailed description of the layout hierarchy for iViews and Web applications. Red cells explicitly prohibit certain nestings. Yellow cells indicate cases where elements can be placed into other elements with certain restrictions only (see also the [reasons](#) for these rules).

iView

Element below can be placed within Container to the right	Container			
	iView (Tray)	Tabstrip	Group	Subgroup
Tabstrip	yes: together with other elements no: as single element	no	no *	no
Group	yes: together with other elements no: as single element	yes	possible - but use with care! one level at maximum - use different types for the nesting	no
Subgroup	yes	yes	yes	no
Table View	yes: together with other elements no: as single element	yes	no *	no
Text Area, Graphic	yes	yes	yes	no
Separator (White Space, Line)	yes	yes	yes	no
Heading	yes: for subgroup, text area, graphic	yes: subgroup, text area, graphic	yes: subgroup, text area, graphic	yes (as heading for the subgroup)

Field, Selection Element, Icon, Button	yes	yes	yes	yes
---	-----	-----	-----	-----

Legend

- *) As iViews are simple applications, tabstrips and table views should not be placed into group controls.
- Red cells: Nesting forbidden
- Yellow cells: Nesting allowed under certain conditions only
- The **bold no's** indicate common errors, such as nested tabstrips.

Web Application

Element below can be placed within Container to the right	Container				
	Application (Background)	Area	Tabstrip	Group	Subgroup
Tabstrip	???	yes (can be a single element with area header as title)	no	yes: together with other elements no: as single element	no
Group	???	yes: together with other elements no: as single element	yes	possible - but use with care! one level at maximum - use different types for the nesting	no
Subgroup	???	yes	yes	yes	no
Table View	???	yes	yes	yes: together with other elements no: as single element	no
Text Area, Graphic	???	yes	yes	yes	no
Separator (White Space, Line)	???	yes	yes	yes	no
Heading	???	yes: group, text area, graphic	yes: subgroup, text area, graphic	yes: subgroup, text area, graphic	yes
Button	???	yes	yes	yes	yes
Field, Selection Element, Icon	???	yes: Header data, group ??? no: other data ???	yes	yes	yes

Legend

- Red cells: Nesting forbidden
- Yellow cells: Nesting allowed under certain conditions only
- The **bold no's** indicate common errors, such as nested tabstrips.
- ??? **Open** (placement of elements on the application background)



[Top](#)

General Nesting Rules

The following nesting rules are derived from the table overviews above and arranged according to design rationales, such as avoiding too much framing and avoiding redundant headers.

Avoid Redundant Headers

The nesting rules defined for the layout hierarchy strive for **avoiding redundant headings**. Thus, do not place:

- Singular group boxes within areas (Web applications only), or tabstrips
- Singular tables with a table heading within group controls

Avoid Too much Framing (Visual Complexity)

Too many frames and borders make screens visually complex and waste screen space. Thus, do not place:

- Singular tables with a table header within group controls - use a table heading instead
- Singular tabstrips within group controls - Web applications: place them in areas instead; use header texts, or the area header as a title for the tabstrip
- Group controls within group controls - use groups with text headers and separators instead (not forbidden but should be used with care - try to use different types for the nesting)
- Tabstrips within tabstrips - nesting tabstrips is a perfect way of information hiding
- Separator lines between containers or container-like elements



[Top](#)

Related Controls

[Flow Layout](#), [Grid Layout](#)



[Top](#)

Note:

The values you find in here for spacing and layouting can not be used with the grid layout control currently in usage, and are not meant to be used with it. The grid layout control as the current SAP layouting tool does only support very simple design possibilities. For information how to use the grid layout control see: [Grid Layout / Usage and Types](#).

A new form layout control is being developed and will be available latest with the 6.0 version of the portal. The pages about spacing you find under the first point "1. General" have been written to support the development of the form layout tool, and to meet the necessities of future design needs in advance.

Spacing Between Grouped Controls

[Spacing in a Tray](#) | [What's a Correct Spacing Good for](#) | [Spacing between Groups](#) | [Spacing between Group Controls with Header and Border](#) | [Spacing of Elements in Groups](#) | [Arranging Groups](#) | [Spacing Soft Groups](#)

This page describes the detailed spacing between grouped controls. For the spacing between single controls, see [Spacing Between Single Controls](#).

The following issues are covered here:

- [Spacing in a Tray](#) - the offset between a tray's border and its content
- [What's a Correct Spacing Good for](#) - the reasoning behind tray offsets and caesuras
- [Spacing between Groups](#) - the spacing between primary and secondary groups, that is, between nested groups
- [Spacing between Group Controls with Header and Border](#) - this comprises more complex controls and the groups
- [Spacing of Elements in Groups](#) - the offset within groups, such as the offset between the group border and its content and the group header and its content
- [Arranging Groups](#) - Alignment of groups and offsets between groups within trays
- [Spacing Soft Groups](#) - Spacing rules for groupings that do not use a group control as container

The spacing rules in short:

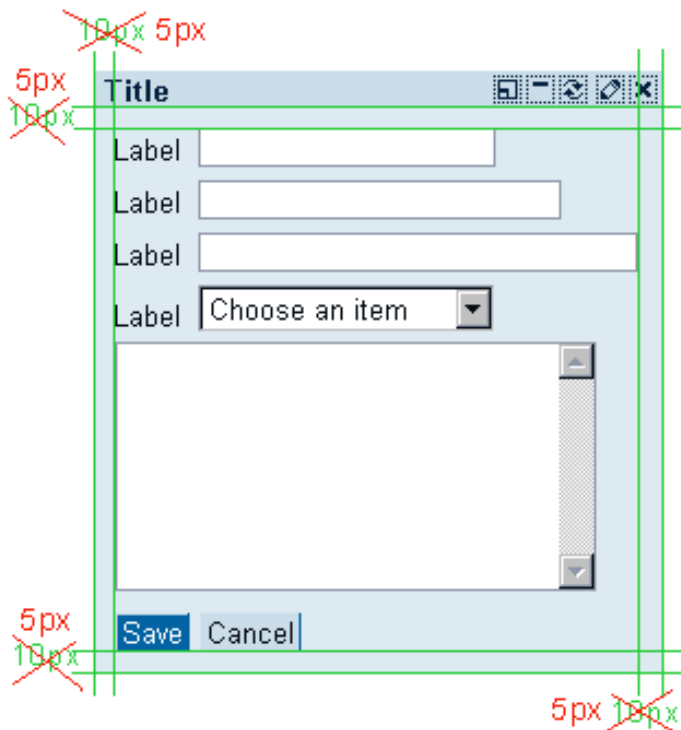
- Offset between tray border/header and content: 5 pixels
- Spacing between primary and secondary groups: 10 pixels
- Spacing between group controls with header and border: 10 pixels
- Offset within groups, i.e. between group border and content: 5 pixels
- Spacing within groups: 10 pixels between title and content, 10 pixels between content and buttons
- Spacing between soft groups: 15 pixels horizontally, 30 pixels vertically

Below you find positive and some negative examples for these cases.



[Top](#)

Spacing in a Tray



This part of the HTMLB guidelines has been updated. The former specifications for the tray offset are not valid any longer. The old specifications are now ~~striked through and replaced by new ones.~~ The reason for this is that due to technical reasons in EP50 the tray did not deliver the 5 pixel offset it was supposed to, though the grid layout control did so. Thus an offset of only five pixels was possible. With EP60 the offset for the tray content will be delivered by the tray itself. See the further specifications.

Figure 1a: Offset around a tray in EP50. The tray offset was only delivered by the grid layout control and thus is only 5 pixels and not 10 as assumed.

 [Top](#)

By EP6.0 the whole content offset of a tray will be delivered by the tray itself. With the new design for EP60 new design specifications have been made. The new tray offset is specified in the picture on the left.

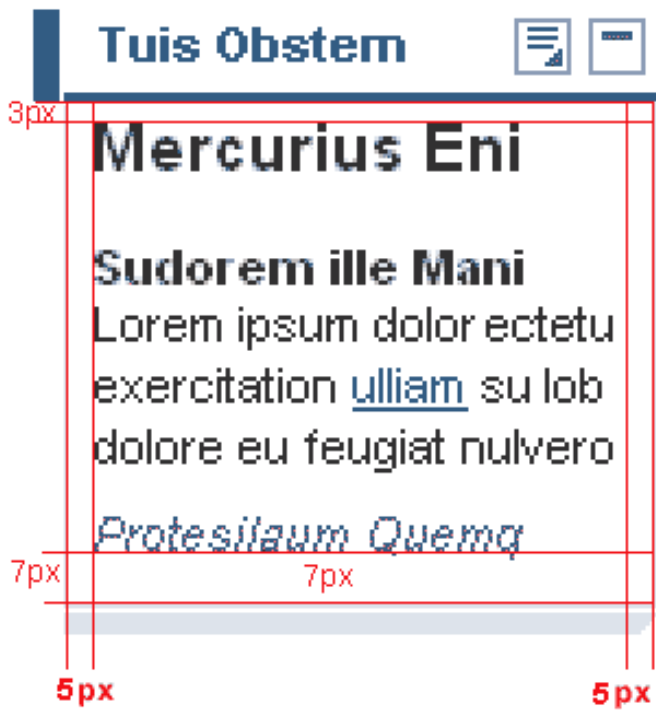


Figure 1b: Offset around a tray in EP60

[Top](#)

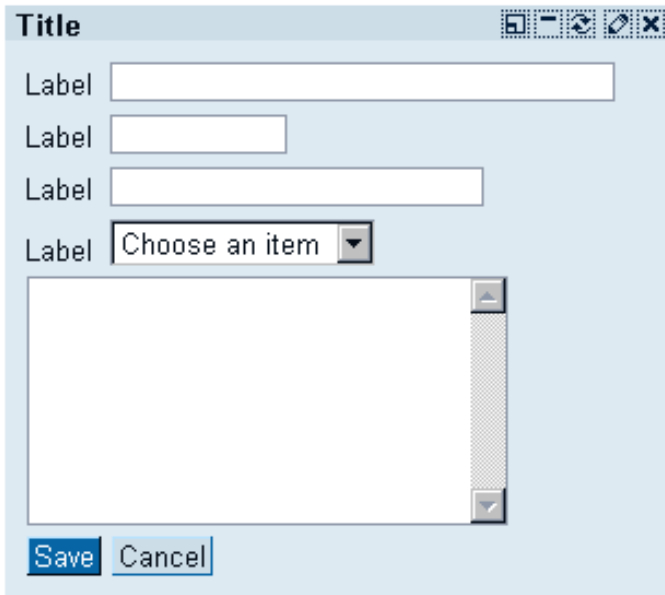


Figure 2: Example of an offset around a tray

[Top](#)

Although you do not always need an offset on the right side, you must always give one to the tray. Whether the offset is needed, depends to the tray's current size which is dependent to the current layout of the portal.

Avoid this. No offset at all gives the impression of elements falling out of the tray.

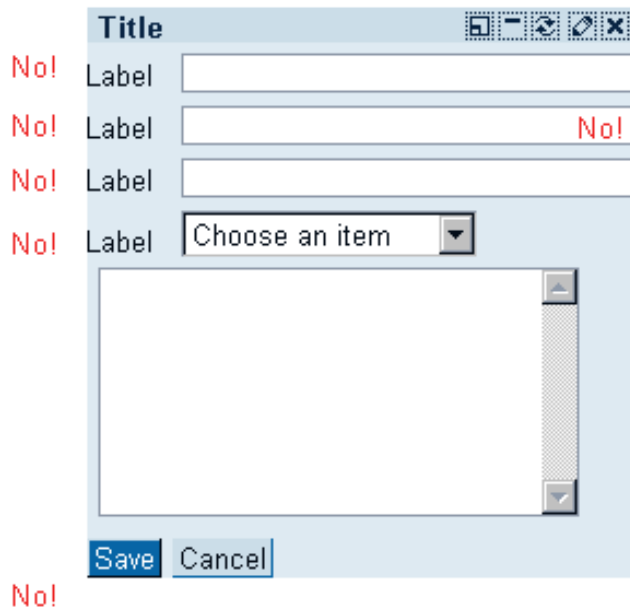


Figure 3: Example of a wrong offset around a tray



[Top](#)

What's a Correct Spacing Good for

We use offsets for both groupings and caesuras. In the above examples a 5 pixel offset around the tray's content area ensures that a tray's content is realized as being in the tray.

Caesuras separate areas from each other. They stress the individual character of the single area, for instance the group.



[Top](#)

Spacing between Primary and Secondary Groups

The spacing around grouping controls of the type "primary group" and "secondary group", i.e., groups without a border and without a header area, should be 10 pixels.

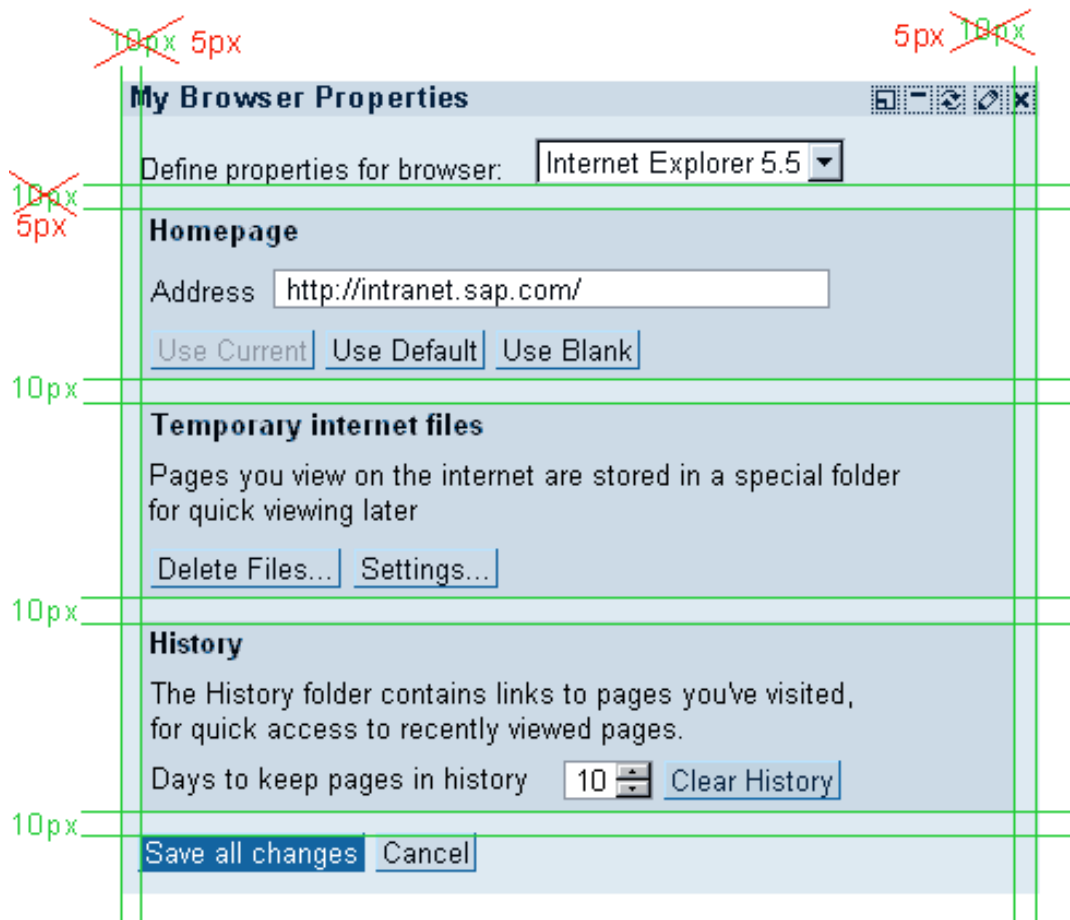


Figure 4: Spacing around secondary groups

[Top](#)

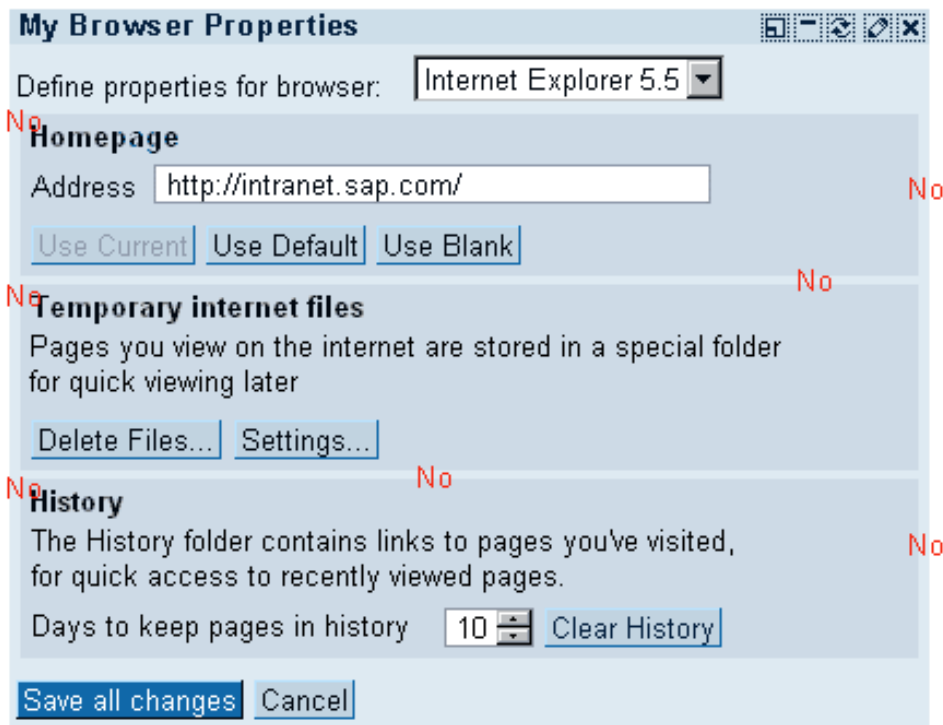


The caesuras clearly stress three areas, realized as three groups.

Figure 5a: Example of spacing around secondary groups



Top



A smaller offset blurs the contrast between secondary group control and the tray's background.

Figure 5c: Example of wrong spacing around secondary groups



Top

Spacing between Group Controls with Header and Border

Group controls with header and border should also be surrounded by a 10 pixel offset to clearly separate the single groups from each other.

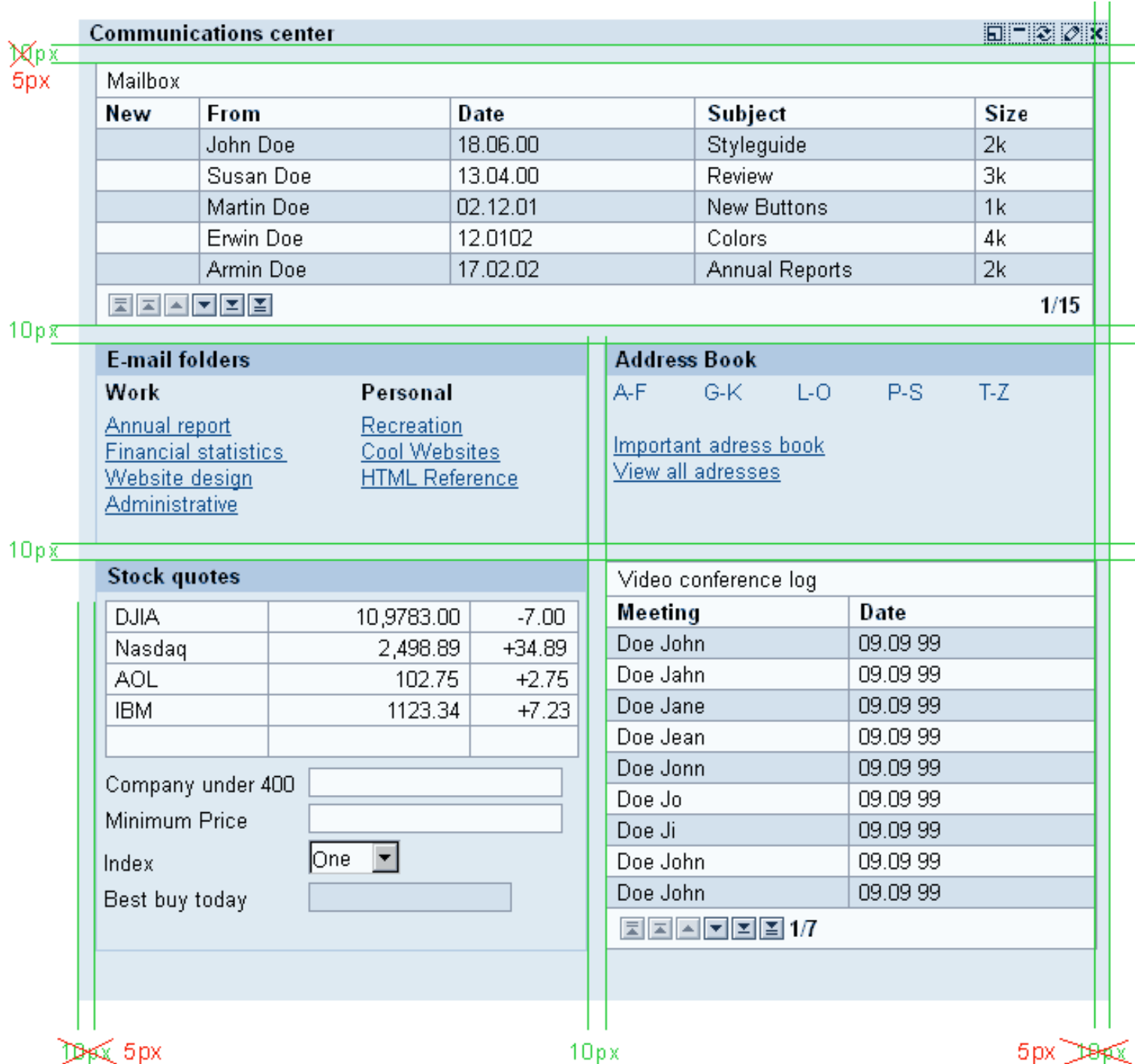



Figure 7a: Spacing group controls with header and border




Top

A smaller offset makes the area around the borders noisy and disquiet. When more than two borders come together in a very small space it is very hard to figure out which border belongs to which group. It becomes even harder, when the borders have the same color.

Communications center 

Mailbox

New	From	Date	Subject	Size
	John Doe	18.06.00	Styleguide	2k
	Susan Doe	13.04.00	Review	3k
	Martin Doe	02.12.01	New Buttons	1k
	Erwin Doe	12.0102	Colors	4k
	Armin Doe	17.02.02	Annual Reports	2k

 No 1/15


E-mail folders		Address Book	
Work	Personal	A-F	G-K
Annual report	Recreation	L-O	P-S
Financial statistics	Cool Websites		T-Z
Website design	HTML Reference	Important address book	
Administrative		View all addresses	

No No

Stock quotes		
DJIA	10,9783.00	-7.00
Nasdaq	2,498.89	+34.89
AOL	102.75	+2.75
IBM	1123.34	+7.23

Company under 400

Minimum Price

Index 

Best buy today

No

Video conference log

Meeting	Date
Doe John	09.09 99
Doe Jahn	09.09 99
Doe Jane	09.09 99
Doe Jean	09.09 99
Doe Jonn	09.09 99
Doe Jo	09.09 99
Doe Ji	09.09 99
Doe John	09.09 99
Doe John	09.09 99


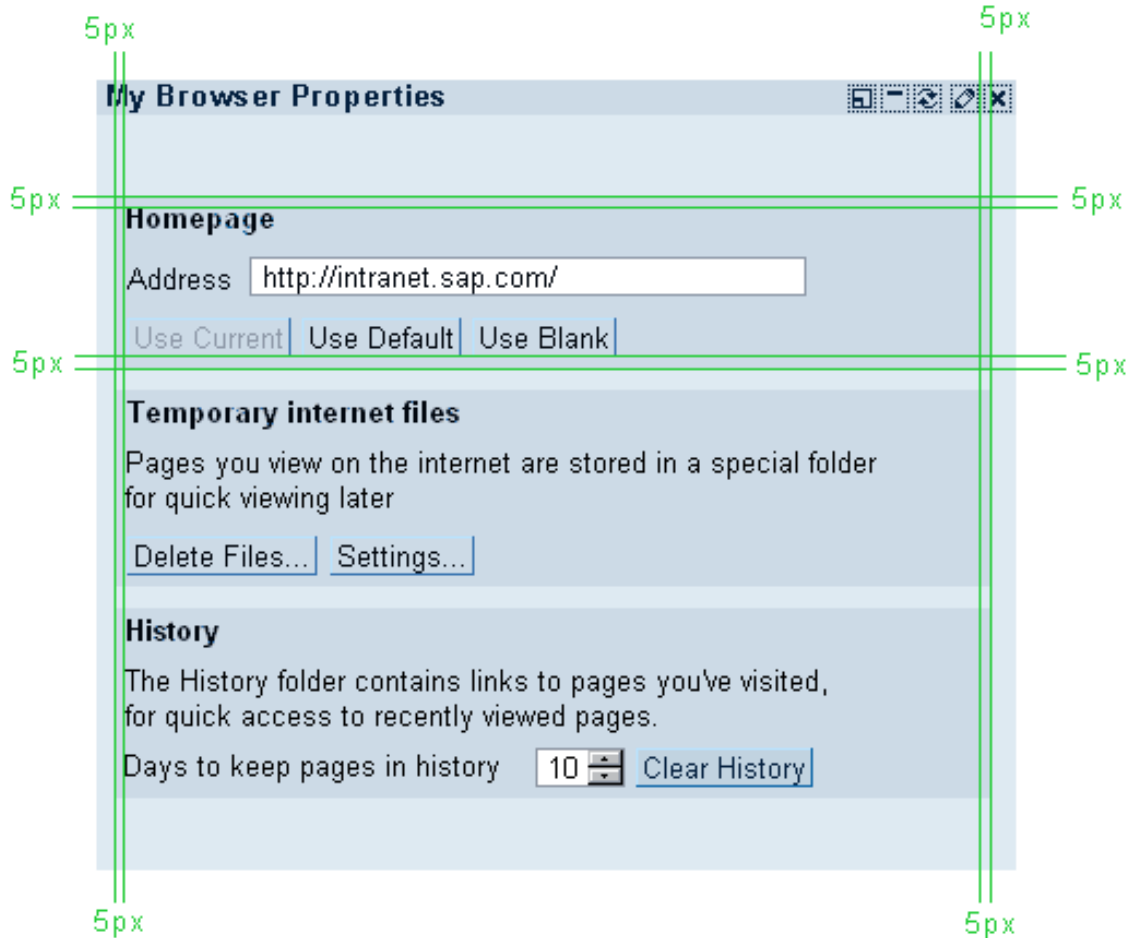
 No 1/7

Figure 7b: Example of noisy interface



Top

Spacing of Elements in Groups



A group's content should be surrounded by an offset of five pixels.

Figure 8a: A group's inner offset - borders



Top

Leave an offset of 10 pixels beneath titles and above buttons.

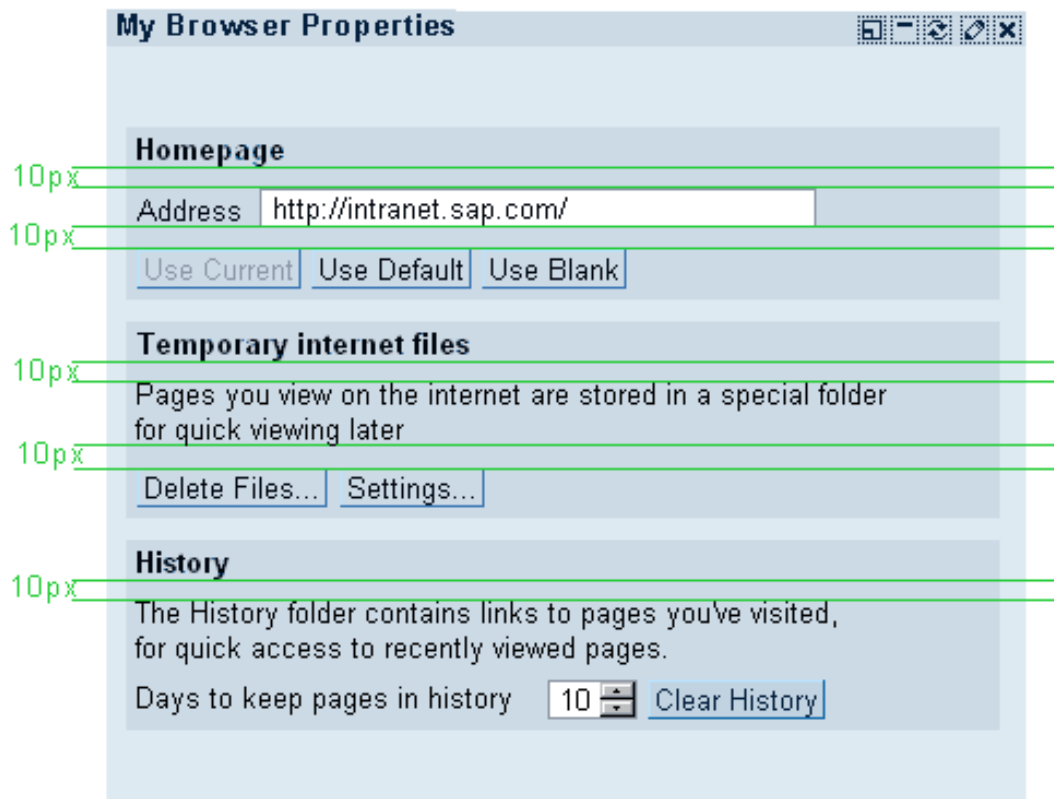


Figure 8b: A group's inner offset - inner spacing

 [Top](#)

Arranging Groups

Groups must have a vertical alignment which is achieved by giving them a width of 50%. A horizontal alignment is nice to have but not necessary. However, a scenario like the following must be avoided by any means.

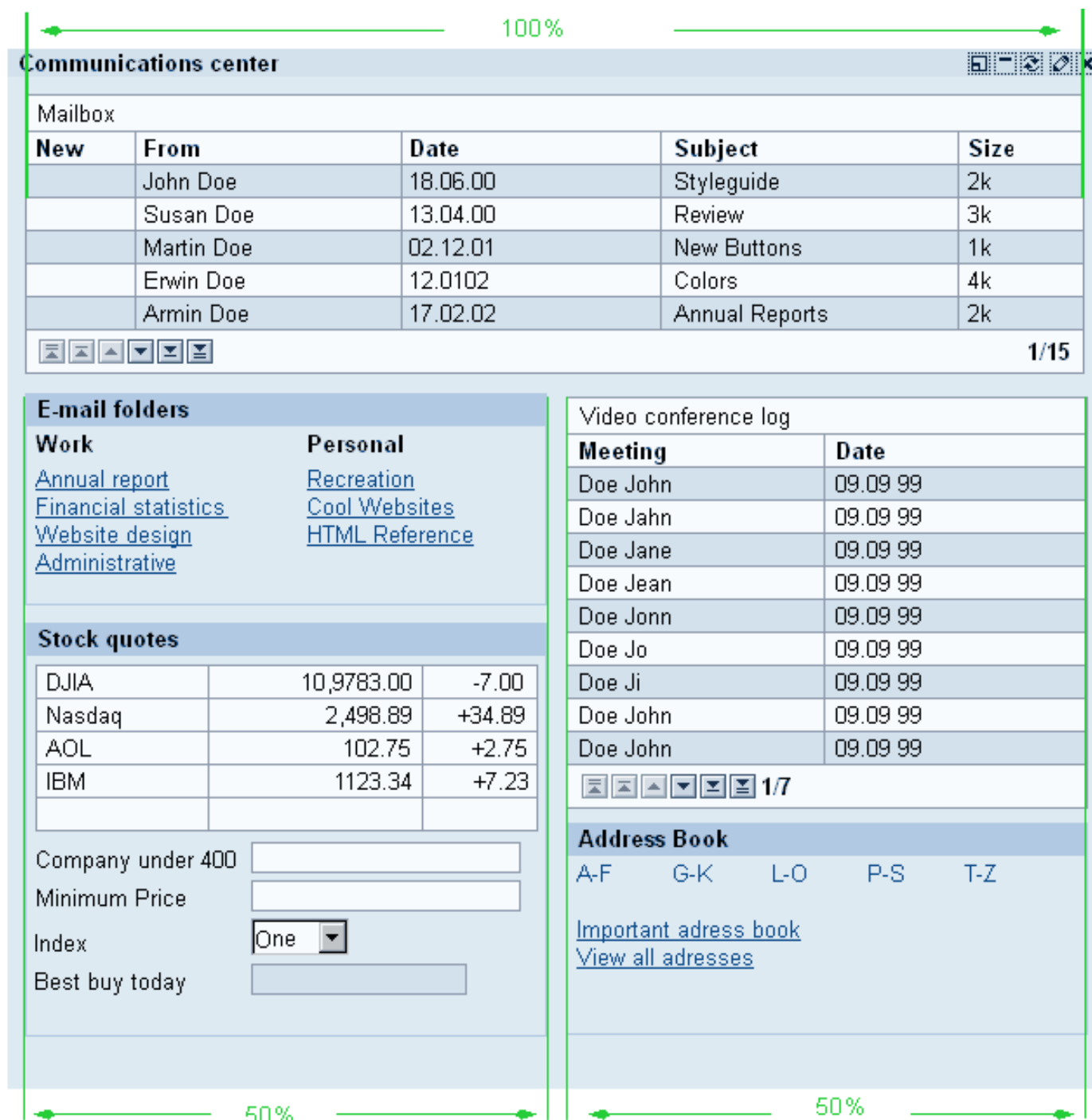


Figure 9a: Arranged groups

[Top](#)

Communications center

E-mail folders

Work	Personal
Annual report	Recreation
Financial statistics	Cool Websites
Website design	HTML Reference
Administrative	

No

Video conference log

Meeting	Date
Doe John	09.09 99
Doe Jahn	09.09 99
Doe Jane	09.09 99
Doe Jean	09.09 99
Doe Jonn	09.09 99
Doe Jo	09.09 99
Doe Ji	09.09 99
Doe John	09.09 99
Doe John	09.09 99

No

Stock quotes

DJIA	10,9783.00	-7.00
Nasdaq	2,498.89	+34.89
AOL	102.75	+2.75
IBM	1123.34	+7.23

Company under 400

Minimum Price

Index

Best buy today

Address Book

A-F G-K L-O P-S T-Z

[Important address book](#)

[View all addresses](#)

No

Avoid scenarios of this kind!

Figure 9b: Example of unaligned groups



Top

Spacing Soft Groups

All those groupings which are not held together by a second or third control that, on the visual side, is rendered as a box, are called soft groups. Soft groups are formatted text or elements that are both, gathered under a header and separated by caesuras. We separate soft groups from each other by using blank space. The advantage of doing so is, we need less code as we do not use an additional control. Second, we have a quite interface as we do not use group boxes with borders or HTML elements like horizontal rulers. The disadvantage of this method is, we have to waste a lot of space to clearly separate single groups from each other.

Use a caesura of 15 pixels to separate soft groups from each other.

Hi-tech [Icons]

Stocks Retreat in Afternoon as Bonds Slip

NEW YORK (Reuters) - Wall Street stocks retraced most of their gains in late trading tracking the bond market lower after a lackluster auction of U.S. Treasury bonds.

DJIA	10,9783.00	-7.00
Nasdaq	2,498.89	+34.89
AOL	102.75	+2.75

Internet Shares Move Higher Once Again

NEW YORK (Reuters) - Internet shares moved higher for the second straight day higher interest rates lessened. The American Stock Exchange 48-share Internet

Internet Shares Mover Higher Once Again

NEW YORK (Reuters) - The American Stock Exchange 48 - share Internet index points, or 2.54 percent, at 269.39 in early afternoon trading, extending.

To send this article to a friend enter the e-mail address

High Tech Companies Listed Alpabetically

[A-C](#) [D-F](#) [G-K](#) [L-O](#) [P-S](#) [T-V](#) [W-Z](#)

[Computers & business equipment](#)

[Computers](#)

[Computer storage devices](#)

[Computer peripherals](#)

[Office machines consumer](#)

[Electronics](#)

[Household audio & video equipment](#)

[Electronic games and toys](#)

[Semiconductors and other](#)

[Resistors](#)

[Coils, transformers and inductors](#)

[Process and control instruments](#)

[Electronic games and toys](#)

[Semiconductors and other](#)

[Resistors](#)

[Coils, transformers and inductors](#)

[Process and control instruments](#)

[Laboratory analytical instruments](#)

Industrial Trends

- [e-business](#)
- [Supply Chain Mgt](#)
- [IT](#)
- [Knowledge Management](#)
- 15px [Globalization](#)
- [Channels of Distribution](#)
- [Emerging Technologies](#)
- [Java](#)
- 15px [RosettaNet](#)
- [XML](#)

Industrial Processes

 - [Enterprise Management](#)
 - [Customer Service](#)
 - [Product Design](#)
 - 15px [Supply Chain Planning](#)
 - [Procurement](#)
 - [Manufacturing](#)

Industries

 - [Computer](#)
 - [Elektronics](#)
 - [Graphics](#)
 - [Hardware](#)
 - [Microcomputers](#)
 - [Motors](#)
 - [Networks](#)
 - [News](#)
 - [Nutrition](#)
 - [Plastic](#)
 - [Strategic Development](#)

Figure 10a: Soft groups



Top

Title and Text can be assigned clearly.

Hi-tech

Stocks Retreat in Afternoon as Bonds Slip
 NEW YORK (Reuters) - Wall Street stocks retraced most of their gains in late trading tracking the bond market lower after a lackluster auction of U.S. Treasury bonds.

DJIA	10,9783.00	-7.00
Nasdaq	2,498.89	+34.89
AOL	102.75	+2.75

Internet Shares Move Higher Once Again
 NEW YORK (Reuters) - Internet shares moved higher for the second straight day higher interest rates lessened. The American Stock Exchange 48-share Internet

Internet Shares Mover Higher Once Again
 NEW YORK (Reuters) - The American Stock Exchange 48 - share Internet index points, or 2.54 percent, at 269.39 in early afternoon trading, extending.

To send this article to a friend enter the e-mail address

High Tech Companies Listed Alpbabetically

[A-C](#)
[D-F](#)
[G-K](#)
[L-O](#)
[P-S](#)
[T-V](#)
[W-Z](#)

[Computers & business equipment](#)
[Computers](#)
[Computer storage devices](#)
[Computer peripherals](#)
[Office machines consumer](#)
[Electronics](#)
[Household audio & video equipment](#)
[Electronic games and toys](#)
[Semiconductors and other](#)
[Resistors](#)
[Coils, transformers and inductors](#)
[Process and control instruments](#)

[Electronic games and toys](#)
[Semiconductors and other](#)
[Resistors](#)
[Coils, transformers and inductors](#)
[Process and control instruments](#)
[Laboratory analytical instruments](#)

Industrial Trends

- [e-business](#)
- [Supply Chain Mgt](#)
- [IT](#)
- [Knowledge Management](#)
- [Globalization](#)
- [Channels of Distribution](#)
- [Emerging Technologies](#)
- [Java](#)
- [RosettaNet](#)
- [XML](#)

Industrial Processes

- [Enterprise Management](#)
- [Customer Service](#)
- [Product Design](#)
- [Supply Chain Planning](#)
- [Procurement](#)
- [Manufacturing](#)

Industries

- [Computer](#)
- [Elektronics](#)
- [Graphics](#)
- [Hardware](#)
- [Microcomputers](#)
- [Motors](#)
- [Networks](#)
- [News](#)
- [Nutrition](#)
- [Plastic](#)
- [Strategic Development](#)

Figure 10b: Example of soft groups



[Top](#)

It is possible to use a two column layout like in this example. If doing so, a caesura of 30 pixels should be used.

The screenshot shows a web page layout with a 30px caesura between two columns. The left column contains news articles and a list of high-tech companies. The right column contains a list of industrial trends and processes. A vertical green line with '30px' labels indicates the spacing between the columns.

Hi-tech

Stocks Retreat in Afternoon as Bonds Slip
NEW YORK (Reuters) - Wall Street stocks retraced most of their gains in late trading tracking the bond market lower after a lackluster auction of U.S. Treasury bonds.

DJIA	10,9783.00	-7.00
Nasdaq	2,498.89	+34.89
AOL	102.75	+2.75

Internet Shares Move Higher Once Again
NEW YORK (Reuters) - Internet shares moved higher for the second straight day higher interest rates lessened. The American Stock Exchange 48-share Internet

Internet Shares Mover Higher Once Again
NEW YORK (Reuters) - The American Stock Exchange 48 - share Internet index points, or 2.54 percent, at 269.39 in early afternoon trading, extending.

To send this article to a friend enter the e-mail address

High Tech Companies Listed Alpbabetically

[A-C](#) [D-F](#) [G-K](#) [L-O](#) [P-S](#) [T-V](#) [W-Z](#)

[Computers & business equipment](#)
[Computers](#)
[Computer storage devices](#)
[Computer peripherals](#)
[Office machines consumer](#)
[Electronics](#)
[Household audio & video equipment](#)
[Electronic games and toys](#)
[Semiconductors and other](#)
[Resistors](#)
[Coils, transformers and inductors](#)
[Process and control instruments](#)

[Electronic games and toys](#)
[Semiconductors and other](#)
[Resistors](#)
[Coils, transformers and inductors](#)
[Process and control instruments](#)
[Laboratory analytical instruments](#)

Industrial Trends

- [e-business](#)
- [Supply Chain Mgt](#)
- [IT](#)
- [Knowledge Management](#)
- [Globalization](#)
- [Channels of Distribution](#)
- [Emerging Technologies](#)
- [Java](#)
- [RosettaNet](#)
- [XML](#)

Industrial Processes

- [Enterprise Management](#)
- [Customer Service](#)
- [Product Design](#)
- [Supply Chain Planning](#)
- [Procurement](#)
- [Manufacturing](#)

Industries

- [Computer](#)
- [Elektronics](#)
- [Graphics](#)
- [Hardware](#)
- [Microcomputers](#)
- [Motors](#)
- [Networks](#)
- [News](#)
- [Nutrition](#)
- [Plastic](#)
- [Strategic Development](#)

Figure 10c: Caesura between a two column layout



[Top](#)

Whenever we decide to use a layout of this kind, we do not use more than two columns. When using two columns we can decide between dividing the available space in proportions of:

1/2 and 1/2
or: 2/3 and 1/3
or: 1/3 and 2/3.

Hi-tech

Stocks Retreat in Afternoon as Bonds Slip

NEW YORK (Reuters) - Wall Street stocks retraced most of their gains in late trading tracking the bond market lower after a lackluster auction of U.S. Treasury bonds.

DJIA	10,9783.00	-7.00
Nasdaq	2,498.89	+34.89
AOL	102.75	+2.75

Internet Shares Move Higher Once Again

NEW YORK (Reuters) - Internet shares moved higher for the second straight day higher interest rates lessened. The American Stock Exchange 48-share Internet

Internet Shares Mover Higher Once Again

NEW YORK (Reuters) - The American Stock Exchange 48 - share Internet index points, or 2.54 percent, at 269.39 in early afternoon trading, extending.

To send this article to a friend enter the e-mail address

 2/3

High Tech Companies Listed Alpbetically

[A-C](#)
[D-F](#)
[G-K](#)
[L-O](#)
[P-S](#)
[T-V](#)
[W-Z](#)

[Computers & business equipment](#)

[Computers](#)

[Computer storage devices](#)

[Computer peripherals](#)

[Office machines consumer](#)

[Electronics](#)

[Household audio & video equipment](#)

[Electronic games and toys](#)

[Semiconductors and other](#)

[Resistors](#)

[Coils, transformers and, inductors](#)

[Process and control instruments](#)

[Electronic games and toys](#)

[Semiconductors and other](#)

[Resistors](#)

[Coils, transformers and, inductors](#)

[Process and control instruments](#)

[Laboratory analytical instruments](#)

Industrial Trends

- [e-business](#)
- [Supply Chain Mgt](#)
- [IT](#)
- [Knowledge Management](#)
- [Globalization](#)
- [Channels of Distribution](#)
- [Emerging Technologies](#)
- [Java](#)
- [RosettaNet](#)
- [XML](#)

Industrial Processes

- [Enterprise Management](#)
- [Customer Service](#)
- [Product Design](#)
- [Supply Chain Planning](#)
- [Procurement](#)
- [Manufacturing](#)

Industries

- [Computer](#)
- [Elektronics](#)
- [Graphics](#)
- [Hardware](#)
- [Microcomputers](#)
- [Motors](#)
- [Networks](#)
- [News](#)
- [Nutrition](#)
- [Plastic](#)
- [Strategic Development](#)

Figure 10d: Multiple column layout



Top

Note:

The values you find in here for spacing and layouting can not be used with the grid layout control currently in usage, and are not meant to be used with it. The grid layout control as the current SAP layouting tool does only support very simple design possibilities. For information how to use the grid layout control see: [Grid Layout / Usage and Types](#).

A new form layout control is being developed and will be available latest with the 6.0 version of the portal. The pages about spacing you find under the first point "1. General" have been written to support the development of the form layout tool, and to meet the necessities of future design needs in advance.

Spacing Between Single Controls

[Groups of Entry Fields](#) | [Check Box Groups](#) | [Radio Button Groups](#) | [Mixed Form Elements in Vertical Succession](#)

This page describes the detailed spacing between single controls. For the spacing between grouped controls, see [Spacing Between Grouped Controls](#).

The following controls are covered here:

- [Groups of Entry Fields](#) - this is the most often needed case for form-line applications
- [Check Box Groups](#) and [Radio Button Groups](#) - these elements are often used in groups for offering choices or options
- [Mixed Form Elements in Vertical Succession](#) - this case covers combinations of different input elements, which are arranged in one vertical column; for several columns refer to the spacing for multi-column checkbox/radio button groups

The spacing rules in short:

- Vertical spacing between single elements: 5 pixels for fields and dropdown list boxes, 8 pixels for checkboxes and radio buttons
- Horizontal spacing between multiple columns: 15 pixels
- Horizontal spacing between label and input element, width of label column: Width of widest label plus an offset of 8 to 22 pixels
- Spacing between selection element and label: 8 pixels for checkboxes and radio buttons

Below you find positive and negative examples for all these cases.



[Top](#)

Groups of Entry Fields

Leave an offset of five pixels between entry fields.

The screenshot shows a 'Create an Account' form with the following fields and their offsets:

Field Label	Offset
First Name	5px
Last Name	5px
Preferred Greeting Name	5px
E-mail Address	5px
Re-enter E-mail Address	5px
Password	5px
Re-enter Your Password	5px
Birth Date	-

A 'Continue' button is located at the bottom left of the form.

Figure 1a: Offset between fields



The screenshot shows a 'Create an Account' form with the following fields:

- First Name
- Last Name
- Greeting Name
- E-mail
- Re-enter E-mail
- Password
- Re-enter Your Password
- Birth Date

A 'Continue' button is located at the bottom left of the form. The right edges of the input fields are ragged.

At SAP it is common style to have fields that are left aligned with ragged edges on the right side.

Figure 1b: Fields have ragged edges



Create an Account

First Name

Last Name

Preferred Greeting Name

E-mail Address

Re-enter E-mail Address

Password

Re-enter Your Password

Birth Date

[Continue](#)

Figure 2: Example of justified fields

Create an Account

First Name

Last Name

Greeting Name

E-mail

Re-enter E-mail

Password

Re-enter Your Password **widest label in this row**

Birth Date

[Continue](#)

Figure 3a: Offset between label and fields

Justified fields are not necessarily wrong. However, it is hard to figure out why one needs a birth date field with more than eight characters.

As one can never predict the length of a field label on the one side, and how many fields will be necessary in one scenario in succession on the other, it is hardly possible to give a standard offset between label and entry field. As a rule of thumb one can say: In one row of entry fields that follow each other in succession consider the offset between the widest label and its entry field. If possible, try to avoid an offset smaller than 8 pixels, which is one character, and wider than 22 pixels, which is three characters. In the scenario on the left the offset between widest label and its corresponding entry field is 8 pixels.

Create an Account

First Name

Last Name

Greeting Name

E-mail

Re-enter E-mail

Password

Re-enter Your Password Here the offset is 22 pixels

Birth Date

Continue

By restricting the space next to the widest label to a maximum size we ensure that the offset between the smallest label and its corresponding entry field is not too large and the user can still adjust label and entry field to each other.

Figure 3b: Offset between label and fields

Create an Account

First Name

Last Name

Greeting Name

E-mail

Re-enter E-mail

Password

Re-enter Your Password

Birth Date

Continue

Avoid this

Here you can still adjust the largest label and its corresponding field but it becomes almost hard work adjusting "E-mail" to its input field.

Figure 3c: Example of too large offset between label and fields

Though all offsets seem to look correct the missing offset between "Reenter Your Password" and its entry field makes the whole interface look ugly.

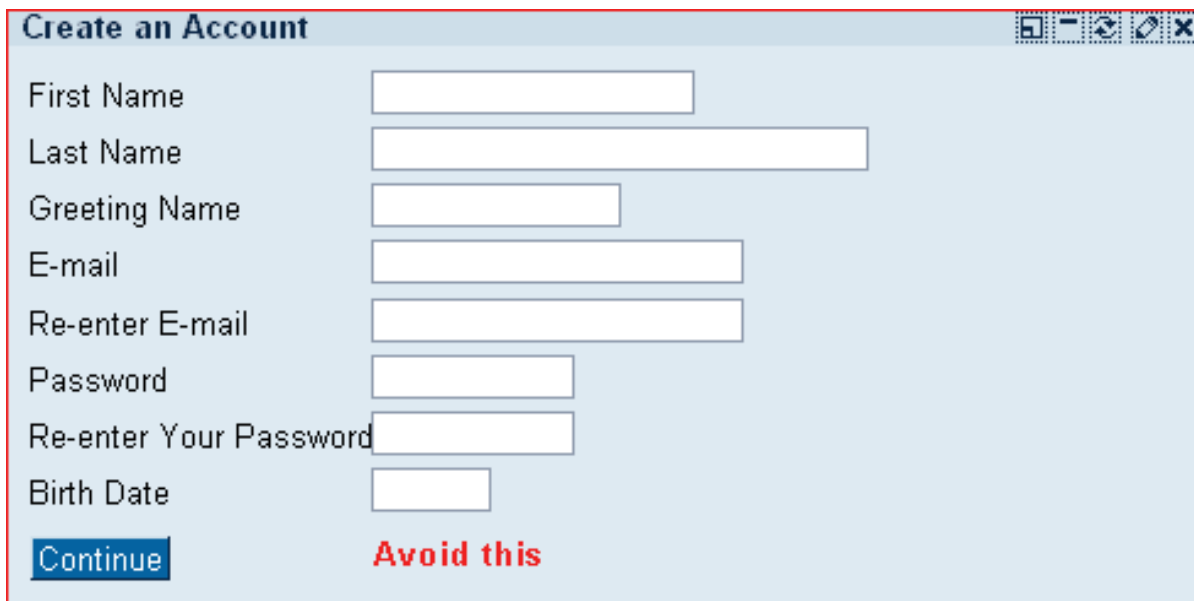


Figure 3d: Example of too small offset between label and fields



[Top](#)

Check Box Groups



Leave an offset of eight pixels between check boxes and their corresponding label.

Figure 4a: Offset between check boxes and their labels



[Top](#)



Leave an offset of eight pixels between rows of check boxes.

Figure 4b: Offset between rows of check boxes.

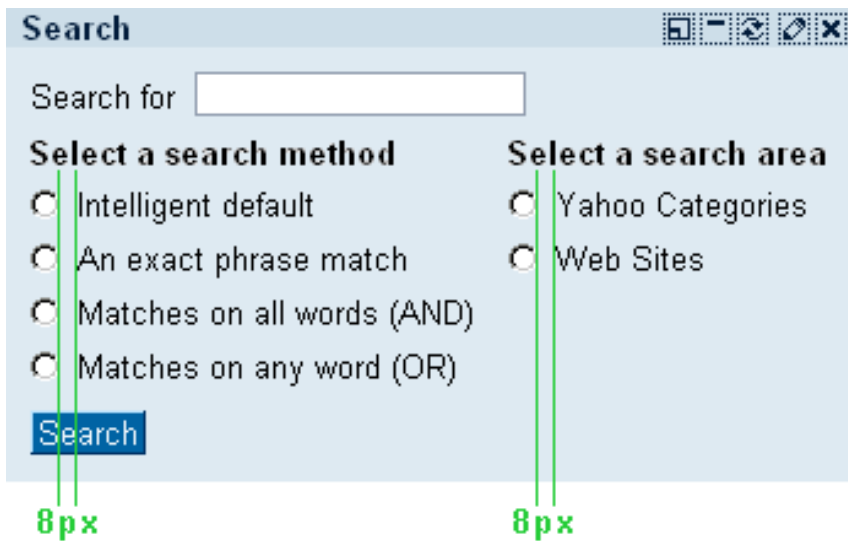


Leave an offset of 15 pixels between columns of check boxes.

Figure 4c: Offset between groups of check boxes

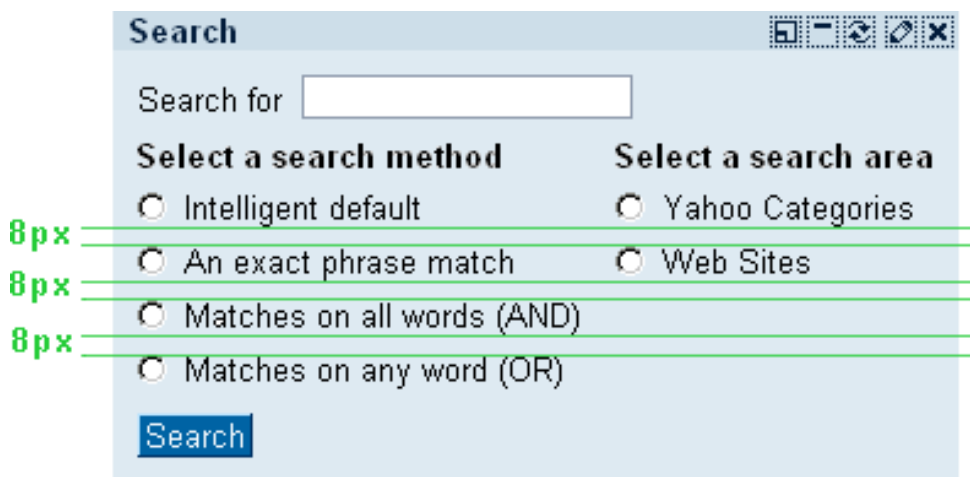


Radio Button Groups



Leave an offset of eight pixels between radio buttons and their corresponding label.

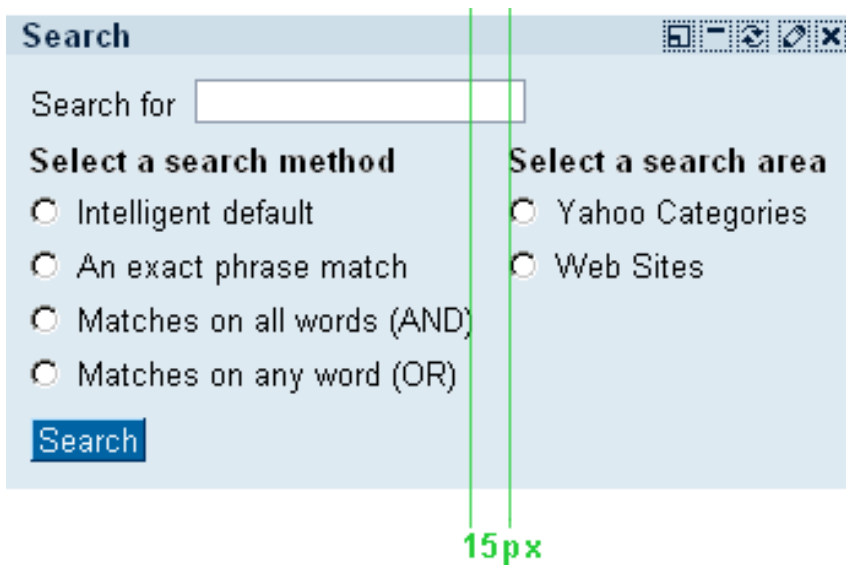
Figure 5a: Offset between radio buttons and their labels



Leave an offset of eight pixels between rows of radio buttons.

Figure 5b: Offset between rows of radio buttons



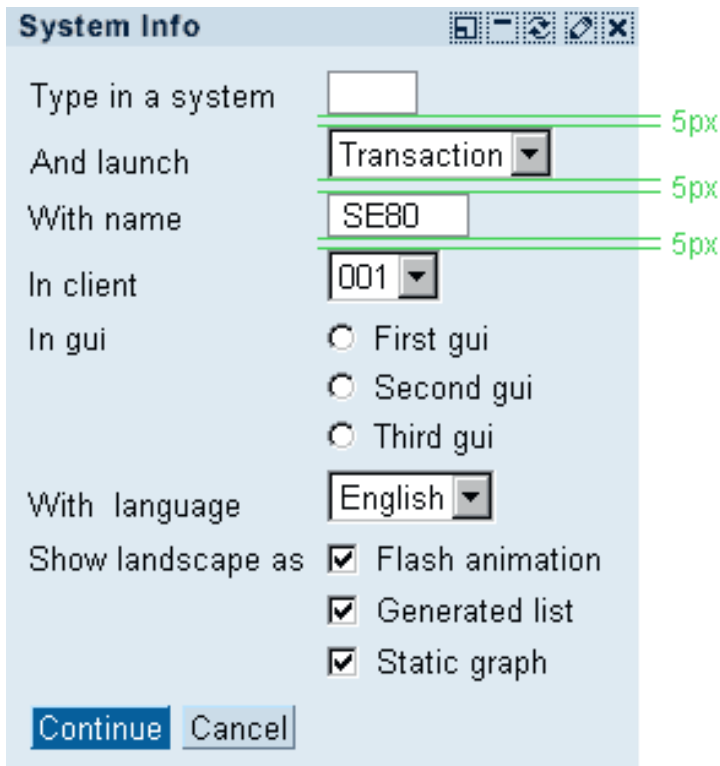


Leave an offset of 15 pixels between columns of radio buttons.

Figure 5c: Offset between radio button groups



Mixed Form Elements in Vertical Succession



Between field like form elements in a vertical succession is always an offset of 5 pixels.

Figure 6a: Offset around mixed form elements in vertical succession



System Info

Type in a system

And launch

With name

In client

In gui

- First gui
- Second gui
- Third gui

With language

Show landscape as

- Flash animation
- Generated list
- Static graph

Above and beneath button like form elements is always an offset of 8 pixels regardless of the following or previous element.

Figure 6b: Offset around mixed form elements in vertical succession



Top

System Info

Type in a system

And launch

With name

In client

In gui

- First gui
- Second gui
- Third gui

With language

Show landscape as

- Flash animation
- Generated list
- Static graph

Leave an offset of 15 pixels between columns of form elements.

Figure 6c: Offset between horizontal groups of mixed form elements in vertical succession



Top

Control API for Content (content)

Creates a scripting variable which provides the rendering context for the following tags. It is a plain HTML tag.

- **id**
Identification name of the content.

attribute	req.	values	default	case sens.	JSP taglib	classlib
id	yes	String	none	yes	id="myContent"	

Example

```
<hbj:content id="myContent">  
  ...  
</hbj:content>
```

Control API for Document (document, documentBody, documentHead)

There are three HTMLB elements for a document, which correspond to HTML elements as follows:

- **document**: Renders the root tag of the document, e.g. <html> or <wml>
- **documentBody**: Renders the <body> section of the document
- **documentHead**: Renders the <head> section of the document

document

Renders the root tag of the document (e.g. <html> or <wml>) depending on the markup language used. It is a plain HTML tag with no attributes.

attribute	req.	values	default	case sens.	JSP taglib	classlib
id	yes		none	yes		setDocumentId("myContent")
title	no	String	none	no		setTitle("SAPPortals")

Example

```
<hbj:content id="myContent">
  <hbj:document>
    ...
  </hbj:document>
</hbj:content>
```

documentBody

Renders <body> section of the document and attaches the appropriate style class. It is a plain HTML tag with no attributes.

Example

```
<hbj:content id="myContent">
  <hbj:document>
    <hbj:documentBody>
      ...
    </hbj:documentBody>
  </hbj:document>
</hbj:content>
```

documentHead

Renders <head> section of the document and includes the necessary style sheets and scripts. It is a plain HTML tag. In the documentHead a nested **META** element (standard HTML) can be used. With **META** element information about the document (name, content, scheme, http-equiv) can be specified.

- **title**
Set the title that is usually displayed in the title bar of the web client.

attribute	req.	values	default	case sens.	JSP taglib	classlib
title	no	String	none	no	title="SAPPortals"	see 'document'

Example

```
<hbj:content id="myContent">
  <hbj:document>
    <hbj:documentHead title="SAPPortals">
      <meta name="description" content="Introduction page">
      <meta name="author" content="SAPPortals">
      <meta name="date" content="Jan. 2002">
      ...
    </hbj:documentHead>
    <hbj:documentBody>
      ...
    </hbj:documentBody>
  </hbj:document>
</hbj:content>
```

Control API for Page (page)

Represents a complete HTML page consisting of tags <html>, <head> and <body> and includes the necessary style sheets and scripts. It is a plain HTML tag.

Important Note:

If JavaScripts are used (for 'onClick' events) the page tag is necessary for the renderer to place the JavaScripts at the end of the page.

- **title**
Set the title that is usually displayed in the title bar of the web client.

attribute	req.	values	default	case sens.	JSP taglib	classlib
title	no	String	none	no	title="SAPPortals"	

Example

```
<hbj:content id="myContent">  
  <hbj:page title="SAPPortals">  
    ...  
  </hbj:page>  
</hbj:content>
```

Control API for Form (form)

Is the outer shell of the document and encapsulates normal content, markup, controls and labels of those controls. Forms are essential for the event handling.

- **action**

Defines the form processing agent. For example, the value might be a HTTP URI to submit the form to a program or a mailto URI to email the form.

- **defaultButton**

Defines the default button for this document. This button will fire an event if the user presses the RETURN / ENTER key in the web agent. Usually the button should have the 'design' "EMPHASIZED" to graphically show that this button is the default button.

If you write an application for different web clients you should be aware of the fact that every web client has its own way to handle keyboard input. To achieve the right results on all web clients you should use the default button always together with an inputField and the inputField must have the focus (usually the inputField gets the focus because the user clicked on this field to do some input). In this case the onClick event of the default button will be fired when the user presses RETURN / ENTER in the inputField.

- **encodingType**

Defines the content type (MIME type) used to submit the form to the web server. Examples of content types can be found at <http://www.w3.org/TR/1998/REC-html40-19980424/interact/forms.html#h-17.3>

Important note:

If you use a fileUpload control in the JSP you must set the encodingType attribute to "multipart/form-data".

Example: `<hbj:form encodingType="multipart/form-data" >`

- **id**

Identification name of the form.

- **language**

Defines the language code for this document. The attribute defines the primary language and can define a series of sub languages. The language code is according to ISO 639. The language code and the description can be found at <http://www.w3.org/TR/1998/REC-HTML40-19980424/references.html#ref-RFC1766>

- **method**

Defines the HTTP method that will be used to submit the form data set. The form data set is a sequence of control name/current value pairs constructed from successful controls. A successful control is "valid" for submission. Every control has its control name paired with its current value as part of the submitted form data set. A successful control must be defined within a form and must have a control name.

Important:

The control name is generated by the HTML-Business for Java renderer. So you have no way to address the control via e.g. JavaScript.

- GET
The form data set is appended to the URI specified by the action attribute (with a question-mark (?) as separator) and this new URI is sent to the processing web agent.
- POST
The form data set is included in the body of the form and sent to the processing web agent.

The default method is POST and should not be altered (Limits of GET requests can cause problems).

- **scrollingToLastPosition**

A boolean value that controls the position in a form. By default the position in a form is always reset to "top of form" when the form is submitted (e.g. in case of an event). If the 'scrollingToLastPosition' attribute is set to true the last position in the form is saved and restored on a submit.

- **target**

Specifies the name of the frame where the document is to be opened. The following values refer to w3c HTML-standard.

- `_blank`
The web client should load the designated document in a new, unnamed window.
- `_self`
The web client should load the document in the same frame as the element that refers to the target.
- `_parent`
The web client should load the document into the immediate FRAMESET parent of the current frame. This value is equivalent to `_self` if the current frame has no parent.
- `_top`
The web client should load the document into the full, original window (thus canceling all other frames). This value is equivalent to `_self` if the current frame has no parent.

Attribute	Req.	Values	Default	Case sens.	JSP Taglib	Classlib
action	no	String	none	yes	action="/servlet/com.test.JspTest"	setAction("/servlet/com.test.JspTest")
defaultButton	no	String	none	yes		setDefaultButton(OKButton)
encodingType	no	String	none	no	encodingType="multipart/mime"	setEncodingType("multipart/mime")

id	yes	String	none	yes	id="SAPForm"	
language	no	String	none	no	language="de"	setLanguage("de")
method	no	GET POST	POST	no	method="POST"	setMethod("POST")
scrollingToLastPosition	yes	FALSE TRUE	FALSE	no		setScrollingToLastPosition(true)
target	no	_blank _self _parent _top	_self	no	target="_blank"	setTarget("_blank")

Example

This example also shows the definition of a default button. We define the OKbutton as default. The assignment is made by scripting (<% %>) and has to be done where the button is defined.

```
<hbj:form
  id="myFormId"
  method="post"
  target="_blank"
  encodingType="multipart/form-data"
  action="/htmlb/servlet/com.sapportals.htmlb.test.MyTestJsp1Test"
  >
```

This form submits to a new web client window
 because of 'target=_blank'.


```
<hbj:inputField
  id="myInputField1"
  type="String"
  invalid="false"
  width="310"
  value="After editing press <Enter> to submit"
  visible="true"
  disabled="false"
  required="true"
  maxlength="30"
  size="50"
  >
```

```
</hbj:inputField>
```

```
<br><br>
```

```
<hbj:button id="oKbutton"
  text="OK"
  onClick="onOKClick"
  design="EMPHASIZED"
  width="100"
  >
```

```
    <% myFormId.setDefaultButton(oKbutton); %>
</hbj:button>
```

```
<hbj:button
  id="Infobutton"
  text="Info"
  onClick="onInfoClick"
  width="100"
  />
```

```
<hbj:button
  id="Cancelbutton"
  text="Cancel"
  onClick="onCanClick"
  width="100"
  />
```

```
</hbj:form>
```


Result

This form submits to a new web client window because of 'target=_blank'.

After editing press <Enter> to submit

OK

Info

Cancel

Flow Layout

[Usage](#) | [Related Controls](#)



Figure 1: Example of the usage of flow layout



Figure 2: The controls within a flow layout.

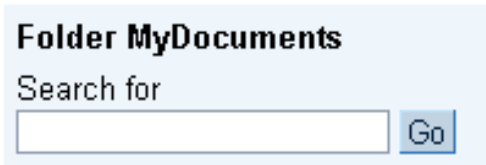


Figure 3: The controls will wrap to fit the size of its container (in this case a group).

The flow layout is an invisible control used to combine other controls one after another. It can be inserted into every container control.



[Top](#)

Usage

Use the flow layout if you do not need to align controls with other elements in your interface. Controls that are added to the flow layout are able to wrap if the available space for displaying all controls in one line does not suffice.

To separate controls within the flow layout you should currently use a [text view](#) control containing a simple space character.

Future Development: To simplify and standardize the separation of controls we will introduce a separator control that can be inserted instead of the text view control.

When use the Flow Layout - When Use the Form Layout

- Use the flow layout if you do not need to align controls with other elements in your interface; this will enhance performance because the flow layout does not have an overhead of table structures in the rendering
- Use the form layout to align controls with respect to other controls in the user interface



Related Controls

[Form Layout](#), [Grid Layout](#), [Text View](#)



More Info about Flow Layout

[Browser Compatibility](#) | [Editability in Style Editor](#) | [Accessibility – 508 Support](#)

Browser Compatibility

The flow layout control structures elements in every browser.

Editability in Style Editor

Currently, the flow layout is not changeable by the Style Editor.

Accessibility – 508 Support

The flow layout has no special accessibility enhancements. It can contain several controls that are by themselves in the accessible hierarchy and might have further descriptions for blind users.



[Top](#)

Control API for Flow Layout (flowLayout)

The flowLayout is a simple container that renders its contents without additions. It allows to align controls that do not need to be aligned with other elements in an interface. Controls that are added to the flowLayout are able to wrap if the available space for displaying all controls in one line does not suffice.

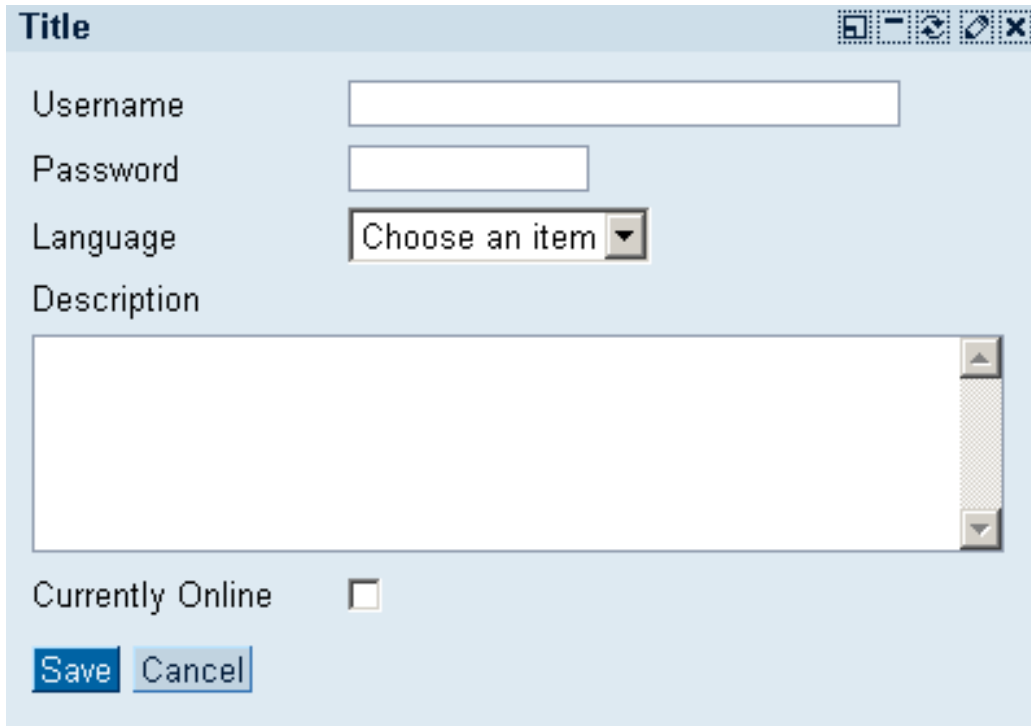
flowLayout has no tag. It has one method **getUI** which returns an identification string for the renderer that is unique for all supported components. The following attributes are inherited from **container**.

attribute	req.	description	case sens.	classlib
addComponent	no	Adds a component to the container. The component is added to the end of the already added text/components.	yes	addComponent ((component) component)
addRawText	no	Adds text - without encoding - to the container e.g. if HTML commands have to be added. The text is added to the end of the already added text/components.	-	addRawText(java.lang.String text)
addText	no	Adds text encoded to the container. The text is added to the end of the already added text/components.	-	addText(java.lang.String text)
removeComponent	no	Removes a component from the container	yes	removeComponent ((component) component)

 [Top](#)

Form Layout

[Usage](#) | [Design-relevant Attributes](#) | [Related Controls](#)



The image shows a window titled "Title" with a standard Windows-style title bar (minimize, maximize, close buttons). Inside the window, the following elements are arranged from top to bottom:

- A text input field labeled "Username".
- A text input field labeled "Password".
- A dropdown menu labeled "Language" with the text "Choose an item" and a downward arrow.
- A text area labeled "Description" with a vertical scrollbar on the right side.
- A checkbox labeled "Currently Online" which is currently unchecked.
- Two buttons: "Save" and "Cancel".

Figure 1: Three form layouts allow to arrange the above form elements and buttons in the manner shown; for details see the example

The form layout is an invisible control for arranging and aligning controls in an application container, group, or other container in a tabular manner. Elements can also be around wrapped within a cell.

The form layout replaces the previous [grid layout](#) control.



[Top](#)

Usage

Use the form layout to align controls within containers in a tabular fashion. Especially, use it in groups, tabstrips and trays (iViews). The grid defined by the form layout is composed of rows, which contain cells. Thus, rows and columns of the grid are defined implicitly. Cells can span multiple columns. In addition, elements in a cell can wrap around. Various controls can be added to the cells.

You can nest form layouts for arranging page elements on different levels.

When Use the Form Layout

- Use the form layout to align controls with respect to other controls in the user interface. The most common usage of the form layout is: (1) laying out forms inside containers, and (2) arranging different containers or form layouts. See the example [below](#) for details and both uses.
- You can also use the form layout if you do not need to align controls with other elements in your interface. In this case, insert only one cell into the respective rows and set the cell's width so that it exceeds the width of the form layout.

Note: The form layout is similar to the [grid layout](#) but has more features for adjusting the grid cells. You need not specify a fixed number of rows and columns but simply add rows and cells within rows. The form layout also can wrap around elements in a cell but the [flow layout](#) is more efficient for this purpose.



[Top](#)

Overview of the Elements and Spacing

The form layout consists of three elements, each of which has a spacing of its own:

- **Form:** `marginBottom`, `marginLeft`, `marginRight`, `marginTop` define the spacing in pixels between the border of the form layout and its content area; each margin is set to zero by default.
- **Row:** `paddingBottom`, `paddingTop` define the padding in pixels at the top and bottom of a row; set to zero by default.
- **Cell:** `paddingBottom`, `paddingLeft`, `paddingRight`, `paddingTop` define the spacing in pixels between the border of the cell and its content area; set to zero by default.

Figure 2 provides an overview of the different margins and paddings within a form layout:

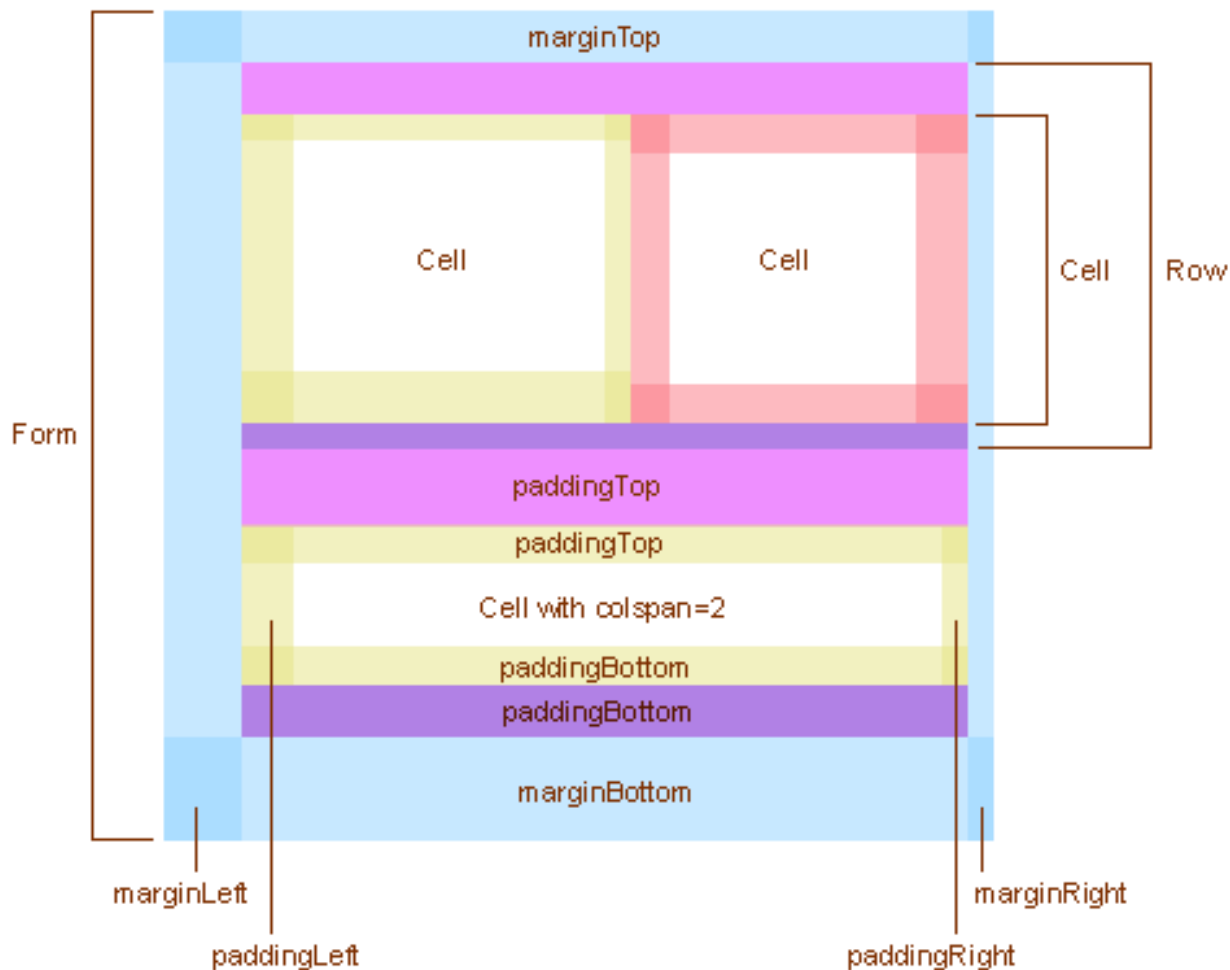


Figure 2: Overview of the different margins and paddings within a form layout

When adding elements to a form layout, take care that the margins and the padding at the different levels are set to the correct values. When nesting form layouts, the margins at the form level have to be left at the default value of zero in order to avoid additional padding. See the example [below](#) for details.



[Top](#)

Example

The following example demonstrates how to use the form layout. First, identify the different areas within your application that need to be aligned. In the example in figure 3 there are two areas, a form area containing labels and input elements, and a button row. Both areas are aligned using separate form layouts. The form layout at the top contains the form area, that is, the labels and the input elements. The second form layout includes the *Save* and *Cancel* buttons. Then both form layouts are arranged using a third form layout.

Top Form Layout

First create a form layout for the input elements. Set all margins (**marginBottom**, **marginLeft**, **marginRight**, **marginTop**) at the form level to 5 pixels. Note that the rules for [Spacing Between Grouped Controls](#) require a spacing of 10 pixels between the tray border and its content. As 5 pixels are already provided by the tray, the form layout has to provide the remaining 5 pixels at the appropriate borders.

Then add six rows to the top form layout and add two cells to each row except the fifth one for achieving a two column layout. In the fifth row, add only one cell and set `colspan=2`.

The rules in [Spacing Between Single Controls](#) require a spaces of 5 pixels between rows of screen elements. As the top and bottom spacings are already set correctly, set the bottom padding (**paddingBottom**) to 5 pixels for all but the last row.

Note: There are alternatives for achieving a 5 pixels spacing between rows but this approach seems to be the easiest way to do it.

For achieving the correct horizontal spacing between the labels and their corresponding input elements, do not specify a value for the width of the left cells. Specify a right padding (**paddingRight**) between 8 and 22 pixels for the left cells, instead. Leave all other cell paddings at their default values of zero.

Finally, add the labels and input elements to the respective cells.

Bottom Form Layout

Create a second form layout for the buttons and set all margins at the form level to 5 pixels. As there are 5 pixels at the bottom of the top form layout and 5 levels at the top of this form layout, you get automatically a spacing of 10 pixels between the bottom input element and the buttons.

Add only one row and two columns for the two buttons to the second form layout. Then add a button to each cell.

Leave the padding at the row and cell levels at their default values of zero, except for the right cell padding of the left cell. Set this padding (**paddingRight**) to 5 pixels. This ensures the correct horizontal spacing between the two buttons.

Title

Username	<input type="text"/>
Password	<input type="password"/>
Language	Choose an item ▾
Description	<input type="text"/>
Currently Online	<input type="checkbox"/>

Save Cancel

Form Layout for the form with margins of 5px

Row containing two cells - row borders not shown
Rowpadding paddingBottom set to 5 pixels, except for the bottom row

Form Layout for the button row with margins of 5px
Right padding of left cell set to 5 pixels

Figure 3: Two form layouts are used for arranging the controls

Arranging the Form Layouts

As the two form layouts are to be appear below each other, you need a third form layout that includes both. This outer form layout

needs two rows and one cell in each row, resulting in only one column (see figure 4). Simply add the two form layouts to the cells.

Note that this outer form layout must not have additional padding or margins. Therefore, let all four margins at the form level at their default values of zero. Also, leave all other padding values at their defaults.

Figure 4: A third form layout is used for arranging the two areas vertically

Hint: You can temporarily set the attribute **debugMode** to TRUE to render the form layout with frames. This makes it easier for you to achieve a proper layout (see [Control API for Form Layout](#) for details).



Top

Design-relevant Attributes

The form layout has design-relevant attributes on the form level, the row level, and the cell level. There are also some dependencies between the attribute values on the different levels. See figure 2 for an overview of the form layout, its elements and spacings.

Form Level

- **marginBottom**, **marginLeft**, **marginRight**, **marginTop**: Define the spacing in pixels between the border of the form layout and its content area; each is set to zero by default.
- **width**: The width can be specified in pixels or percent of the including container width. If the width of a cell is also specified in percent and it exceeds the form layout's width the cell content will be wrapped.

Row Level

- **paddingBottom**, **paddingTop**: Define the padding in pixels at the top and bottom of a row; set to zero by default.

Cell Level

- **align** (LEFT, RIGHT, CENTER, CHAR, JUSTIFY): Defines the horizontal alignment of elements within a cell.

- **valign** (BASELINE, BOTTOM, MIDDLE, TOP): Defines the vertical alignment of elements within a cell.
- **paddingBottom**, **paddingLeft**, **paddingRight**, **paddingTop**: Define the spacing in pixels between the border of the cell and its content area; set to zero by default.
- **Width**: Defines the width in pixels or percent of the form layout. Note that if different widths are specified for a column the last value is used in order to avoid conflicts. Also note that a value exceeding the width of the form layout will cause wrapping behavior.
- **colSpan**: Defines the horizontal expansion of a cell in columns.

You can also use the Boolean attribute **debugMode** as an aid for achieving a proper layout. If it is set to TRUE the cell borders are displayed.

For details see page [Control API for Form Layout](#).



[Top](#)

Related Controls

[Flow Layout](#), [Grid Layout](#)



[Top](#)

More Info about Form Layout

[Browser Compatibility](#) | [Editability in Style Editor](#) | [Accessibility – 508 Support](#)

Browser Compatibility

The form layout control structures elements in every browser.

Editability in Style Editor

Currently, the form layout is not changeable by the Style Editor.

Accessibility – 508 Support

The form layout has no special accessibility enhancements. It can contain several controls that are by themselves in the accessible hierarchy and might have further descriptions for blind users.



[Top](#)

Control API for Form Layout (formLayout)

A grid is a two dimensional arrangement of data in rows and columns. The control is similar to gridLayout but has more features in adjusting the cells of the grid. If no formLayoutCells are defined no formLayout is displayed.

Limitation:

Large formLayouts (large amount of rows and columns) can cause problems like:

- Compiler errors that are caused by a 64kB method length limit.
- Slow processing of page because of huge HTML-Code generated by the JSP-Compiler

- **debugMode**

A Boolean value. If set to "TRUE" the formLayoutCell is rendered with a frame. The frame size is defined by formLayoutCell 'width' and the padding. If a formLayoutCell is not defined or empty no frame is displayed.

If set to "FALSE" no frame is rendered.

By default the borders of the grid are invisible. To see the borders for layout and debug reasons set the debug attribute.

Note: Setting the debugMode attribute will add pixels to visualize borders. Therefore the sizes of the grid layout will change if you reset the attribute. The debugMode attribute, as indicated by the name, should only be used for debugging and not for "styling".

- **id**
Identification name of the formLayout. You have to specify an id if you want to access the control.

- **marginBottom**
Specifies the distance from the bottom of the control to the next control.

- **marginLeft**
Specifies the distance from the "actual" position to the left side of the control.

- **marginRight**
Specifies distance from the right side of the control to the next control.

- **marginTop**
Specifies distance from the "actual" position to the top of the control.

- **width**
Defines the width of the formLayout. If the 'width' in formLayoutCell is specified in percent, the percentage will be calculated from the width of the formLayout. If the formLayoutCell definition exceeds the 'width' of the formLayout the formLayoutCell content will be wrapped.

Attribute	Req.	Values	Default	Case sens.	JSP Taglib	Classlib
debugMode	no	FALSE TRUE	FALSE	yes	debugMode="TRUE"	setDebugMode(true)
id	no	String	none	yes	id="ZIPCode_form"	setId("ZIPCode_form")
marginBottom	no	Unit	0	no	marginBottom="5"	setMarginBottom("5")
marginLeft	no	Unit	0	no	marginLeft="5"	setMarginLeft("5")
marginRight	no	Unit	0	no	marginRight="5"	setMarginRight("5")
marginTop	no	Unit	0	no	marginTop="5"	setMarginTop("5")
width	no	Unit	100%	no	width="500"	setWidth("500")

formLayoutRow

Defines the rows in the formLayout. The cells (formLayoutCell) have to be nested in form layout rows.

- **id**
Identification name of the formLayoutRow. You have to specify an id if you want to access the control.

- **paddingBottom**
Specifies the bottom padding of each row in the form layout. The value of the paddingBottom attribute represents the distance from the bottom border of the cell to the

bottom of the content of each row specified in pixels.

- **paddingTop**

Specifies the top padding of each row in the form layout. The value of the paddingTop attribute represents the distance from the top border of the cell to the top of the content of each row specified in pixels.

Attribute	Req.	Values	Default	Case sens.	JSP Taglib	Classlib
id	no	String	none	yes	id="ZIPCode_row01"	setId("ZIPCode_row01")
padding	no	Numeric	0	-		setPadding(top, bottom) e.g. setPadding("5", "3")
paddingBottom	no	Numeric	0	-	paddingBottom="3"	setPaddingBottom("3")
paddingTop	no	Numeric	0	-	paddingTop="5"	setPaddingTop("5")

formLayoutCell

Defines the cells in the formLayoutRow.

- **align**

Defines the horizontal alignment of the cell content.

- LEFT
Left justifies the content of the cell.
- RIGHT
Right justifies the content of the cell.
- CENTER
Centers the content of the cell.
- CHAR
Aligns text around a specific character. Not supported by all web clients.
- JUSTIFY
Sets text in the cell left and right aligned. Not supported by all web clients.

- **colSpan**

Defines the horizontal expansion the cell in columns.

- **content**

Specifies the content for the cell.

- **id**

Identification name of the formLayoutCell.

- **paddingBottom**

Specifies the bottom padding of each cell in the form layout. The value of the paddingBottom attribute represents the distance from the bottom border of the cell to the bottom of the content of each cell specified in pixels.

- **paddingLeft**

Specifies the left padding of each cell in the form layout. The value of the paddingLeft attribute represents the distance from the left border of the cell to the left side of the content of each cell specified in pixels.

- **paddingRight**

Specifies the right padding of each cell in the form layout. The value of the paddingRight attribute represents the distance from the right border of the cell to the right side of the content of each cell specified in pixels.

- **paddingTop**

Specifies the top padding of each cell in the form layout. The value of the paddingTop attribute represents the distance from the top border of the cell to the top of the content of each cell specified in pixels.

- **valign**

Defines the vertical alignment of the cell content.

- BASELINE
The content of the cell is aligned on the baseline line of the cell (or bottom when no baseline exists).

Control API for Form Layout (formLayout)

- BOTTOM
The content of the cell is aligned to the bottom line of the cell.
- MIDDLE
The content of the cell is aligned to the middle of the cell height.
- TOP
The content of the cell is aligned to the top line of the cell.

• width

Defines the width of the formLayoutCell. One column can have only one width - when you specify different widths for the same column the width defined last is taken.

Attribute	Req.	Values	Default	Case sens.	JSP Taglib	Classlib
align	no	LEFT RIGHT CENTER CHAR JUSTIFY	LEFT	yes	align="LEFT"	setHorizontalAlignment(CellHAlign.LEFT)
colSpan	no	Numeric	1	-		setColSpan(2)
content	no	String	none	no		Text: setContent("A celltext") Component: setContent(ListBox)
id	no	String	none	yes	id="Cell01"	setId("Cell01")
paddingBottom	no	Numeric	0	-	paddingBottom="1"	setPaddingBottom("1")
paddingLeft	no	Numeric	0	-	paddingLeft="5"	setPaddingLeft("5")
paddingRight	no	Numeric	0	-	paddingRight="3"	setPaddingRight("3")
paddingTop	no	Numeric	0	-	paddingTop="2"	setPaddingTop("2")
valign	no	BASELINE BOTTOM MIDDLE TOP	MIDDLE	yes	valign="TOP"	setVerticalAlignment(CellVAlign.TOP)
width	no	Unit	-	-	width="20%"	setWidth("20%")

Example using the taglib

```
<hbj:formLayout id="myForm"
    marginTop="15px"
    marginRight="30px"
    marginBottom="5px"
    marginLeft="15px"
    width="500px" >

    <hbj:formLayoutRow id="Row1"
        paddingTop="10px"
        paddingBottom="5px" >

        <hbj:formLayoutCell id="Cell11"
            align="RIGHT"
            paddingLeft="3"
            paddingTop="5"
            paddingRight="10"
            paddingBottom="5"
            width="40%" >

            <hbj:button id="myButtonf11" text="Button" />
        </hbj:formLayoutCell>

        <hbj:formLayoutCell id="Cell12"
            align="LEFT"
            paddingLeft="3"
            paddingTop="5"
            paddingRight="10"
            paddingBottom="5" >

            <hbj:textView text="Celltext aligned left" />
        </hbj:formLayoutCell>
    </hbj:formLayoutRow>
```

```

<hbj:formLayoutRow id="Row2"
    paddingTop="10px"
    paddingBottom="5px" >

    <hbj:formLayoutCell id="Cell121"
        align="LEFT"
        paddingLeft="3"
        paddingTop="5"
        paddingRight="10"
        paddingBottom="5" >

        <hbj:button id="myButtonf21" text="Button" />
    </hbj:formLayoutCell>

    <hbj:formLayoutCell id="Cell122"
        align="RIGHT"
        paddingLeft="3"
        paddingTop="5"
        paddingRight="10"
        paddingBottom="5" >

        <hbj:textView encode="false" text="Celltext aligned right" />
    </hbj:formLayoutCell>
</hbj:formLayoutRow>
</hbj:formLayout>

```

Example using the classlib

```

Form form = (Form)this.getForm();
FormLayout fl = new FormLayout();
fl.setId("myForm");

fl.setMarginTop("15px");
fl.setMarginRight("30px");
fl.setMarginBottom("5px");
fl.setMarginLeft("15px");
fl.setWidth("500px");
fl.setDebugMode(true);

FormLayoutRow row1 = fl.addRow();
row1.setPaddingTop("10px");
row1.setPaddingBottom("5px");

Button button = new Button("button", "button");
FormLayoutCell cell11 = fl.addComponent(1,1, button);
cell11.setHorizontalAlignment(CellHAlign.RIGHT);
cell11.setPaddingLeft("3");
cell11.setPaddingTop("5");
cell11.setPaddingRight("10");
cell11.setPaddingBottom("5");
cell11.setWidth("40%");

TextView tv1 = new TextView("tv1");
tv1.setText("Celltext aligned left");

FormLayoutCell cell112 = fl.addComponent(1,2, tv1);
cell112.setHorizontalAlignment(CellHAlign.LEFT);
cell112.setPaddingLeft("3");
cell112.setPaddingTop("5");
cell112.setPaddingRight("10");
cell112.setPaddingBottom("5");
cell112.setWidth("40%");

FormLayoutRow row2 = fl.addRow();
row2.setPaddingTop("10px");
row2.setPaddingBottom("5px");

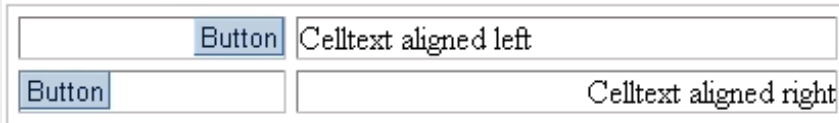
Button button2 = new Button("button2", "button");
FormLayoutCell cell121 = fl.addComponent(2,1, button2);
cell121.setHorizontalAlignment(CellHAlign.LEFT);
cell121.setPaddingLeft("3");
cell121.setPaddingTop("5");
cell121.setPaddingRight("10");
cell121.setPaddingBottom("5");
cell121.setWidth("40%");

```



```
TextView tv2 = new TextView("tv2");  
tv2.setText("Celltext aligned right");  
  
FormLayoutCell cell122 = fl.addComponent(2,2, tv2);  
cell122.setHorizontalAlignment(CellHAlign.RIGHT);  
cell122.setPaddingLeft("3");  
cell122.setPaddingTop("5");  
cell122.setPaddingRight("10");  
cell122.setPaddingBottom("5");  
cell122.setWidth("40%");  
  
form.addComponent(fl);
```

Result



Grid Layout

[Usage](#) | [Design-relevant Attributes](#) | [Related Controls](#)

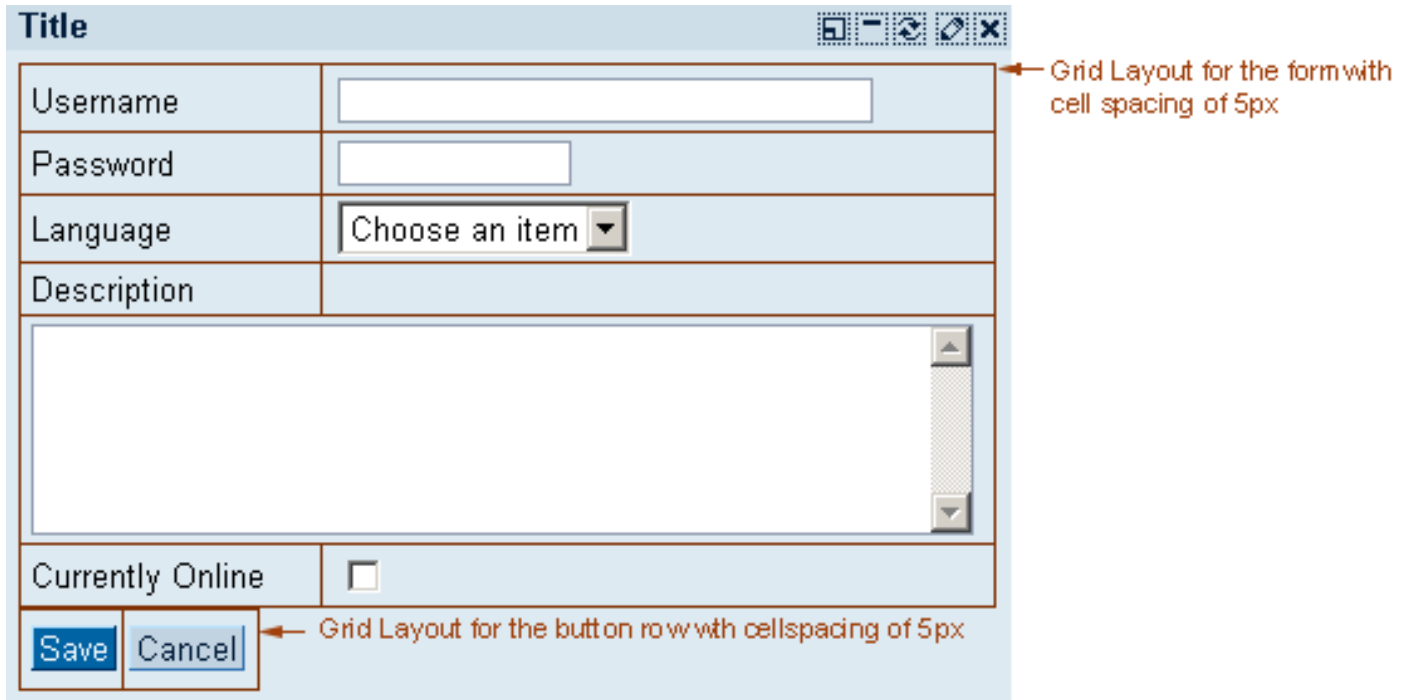


Figure 1: Grid layout arrangement of controls

The grid layout is an invisible control that helps you in arranging and aligning controls in an application, group or other container in a tabular manner.



[Top](#)

Usage

Note: The grid layout is obsolete - use the [form layout](#), instead.

Use the grid layout to align controls within containers in a tabular fashion. The grid consists of cells that are arranged in rows and columns. Various controls can be added to the cells. You can insert the grid layout into any container control. Especially, use it in groups, tabstrips and trays (iViews). You can also nest grid layouts for arranging page elements on different levels (see example [below](#)).

The most common usage of the grid layout is the layout of forms inside containers and the arrangement of different containers. There are two attributes for managing the spacing between rows and columns, **cellSpacing** and **cellPadding**. For both of the above cases a **cellSpacing** of 5 is recommended. There is no need to use **cellPadding** for these default layouts. See the example [below](#) for details.

Note: Currently, it is not possible to achieve the exact paddings and spacings as recommended in the "Layout" section. The suggestions given here are approximations of the optimal layout. Use these values until new controls are introduced, which can deal with the layout

issues of the grid and flow layout controls.

When Use the Grid Layout - When Use the Flow Layout

- Use the grid layout to align controls with respect to other controls in the user interface
Note: Use the [form layout](#) instead of the grid layout
- Use the flow layout if you do not need to align controls with other elements in your interface; this will enhance performance because the flow layout does not have an overhead of table structures in the rendering



[Top](#)

Example

The following example demonstrates how to use the grid layout. First, identify the different areas within your application that need to be aligned. In the example in figure 2 there are two areas, a form area containing the input elements that have to be aligned, and a button row, where buttons have to be added.

Both areas consist of separate grid layouts. The grid at the top contains the form area, the labels, and the input elements. The second grid includes the *Save* and *Cancel* buttons. As the spacing between the controls should always be 10px, set the **cellSpacing** to 5.

The screenshot shows a window titled "Title" with a standard Mac OS-style title bar (minimize, maximize, close buttons). The window content is a grid layout. The top part is a form with four rows: "Username" with a text field, "Password" with a text field, "Language" with a dropdown menu showing "Choose an item", and "Description" with a large text area. Below the form is a row with "Currently Online" and a checkbox. At the bottom is a row with "Save" and "Cancel" buttons. Two orange arrows with text point to the form area and the button row, indicating they are separate grid layouts with a cell spacing of 5px.

Figure 2: Step 1 - grid layout arrangement of inner controls

These two areas should appear below each other. Therefore, you need another grid consisting of one column only and two rows. In the cells you simply add the form area and the button row (see figure 3). This grid does not need additional padding or spacing because the two added areas already have the correct spacing.

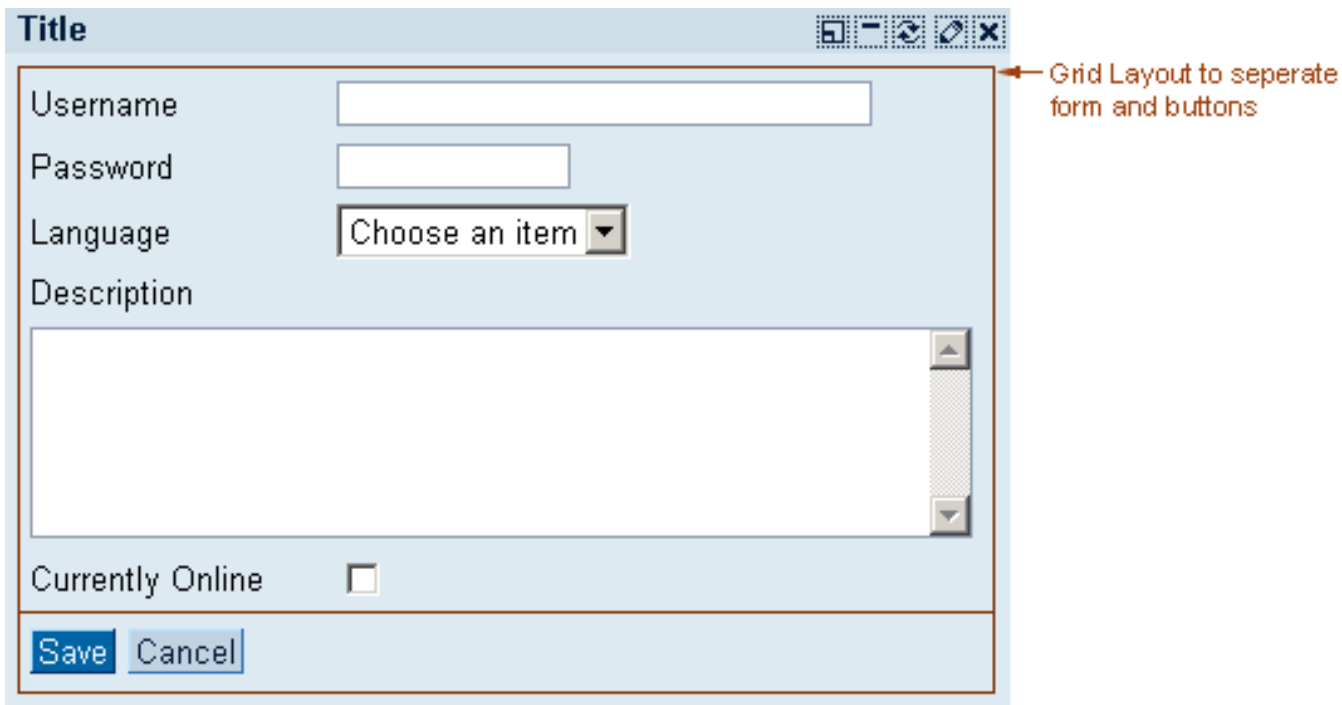


Figure 3: Step 2 - grid layout arrangement of areas

Hint: You can temporarily set the attribute **debugMode** to TRUE to render the grid layout with frames. This helps you in achieving a proper layout (see [Control API for Grid Layout](#) for details).



[Top](#)

Design-relevant Attributes

You can set the number of columns (**columnSize**), and the spacing within (**cellPadding**) and between cells (**cellSpacing**).

You can also use the Boolean attribute **debugMode** as an aid for achieving a proper layout. If it is set to TRUE the cell borders are displayed.

For details see page [Control API for Grid Layout](#).



[Top](#)

Related Controls

[Flow Layout](#), [Form Layout](#)



[Top](#)

More Info about Grid Layout

[Browser Compatibility](#) | [Editability in Style Editor](#) | [Accessibility – 508 Support](#)

Browser Compatibility

The grid layout control structures elements in every browser.

Editability in Style Editor

Currently, the grid layout is not changeable by the Style Editor.

Accessibility – 508 Support

The grid layout has no special accessibility enhancements. It can contain several controls that are by themselves in the accessible hierarchy and might have further descriptions for blind users.



[Top](#)

Control API for Grid Layout (gridLayout)

A grid is a two dimensional arrangement of data in rows and columns. To avoid unexpected results, the rows and columns should always be defined by gridLayoutCells instead of using the 'columnSize' and 'rowSize' attribute. With the gridLayoutCell you have control over the width of the gridLayoutCell while using the "columnSize" the renderer and web client take the control. Especially in combination with 'debugMode' attribute set to "FALSE", the layout of the grid is not displayed as expected. If no gridLayoutCells are defined no gridLayout is displayed.

Limitation:

Large gridLayouts (large amount of rows and columns) can cause problems like:

- Compiler errors that are caused by a 64kB method length limit.
- Slow processing of page because of huge HTML-Code generated by the JSP-Compiler

To avoid these problems you could use the gridLayout tag and combine it with <tr> & <td> tags.

- **cellPadding**

Defines the padding of each cell in the grid layout. The value of the cell padding attribute represents the distance from the border of the cell to the content of each cell specified in pixels.

Note: The cellPadding is applied to the top, left, right and bottom of the cell.

- **cellSpacing**

Specifies the space between the left side of the gridLayout and the left-hand side of the leftmost gridLayoutCell, the top of the gridLayout and the top side of the topmost row and so on for the right and bottom of the gridLayout. The attribute also specifies the amount of space to leave between the gridLayoutCells.

Defines the spacing between cells and the outer boundary in the grid layout in pixels.

- **columnSize**

Defines the number of columns for the gridLayout. The columns are defined with the gridLayoutCell control and 'ColumnSize' is overruled by the gridLayoutCell definition.

- **debugMode**

A Boolean value. If set to "TRUE" the gridLayoutCell is rendered with a frame. The frame size is defined by gridLayoutCell 'width' and the 'cellpadding'. If a gridLayoutCell is not defined or empty no frame is displayed.

If set to "FALSE" no frame is rendered. Please check the gridLayout description above for limitations.

By default the borders of the grid are invisible. To see the borders for layout and debug reasons set the debug attribute.

Note: Setting the debugMode attribute will add pixels to visualize borders. Therefore the sizes of the grid layout will change if you reset the attribute. The debugMode attribute, as indicated by the name, should only be used for debugging and not for "styling".

The image shows a web form titled "User Information" with a light blue border. It contains the following elements:

- Username:** A text input field.
- Password:** A text input field.
- Language:** A dropdown menu with the text "Choose an item" and a downward arrow.
- Description:** A large text area with a vertical scrollbar on the right side.
- Currently Online:** A checkbox that is currently unchecked.
- Buttons:** Two buttons at the bottom left, labeled "Save" and "Cancel".

- **id**

Identification name of the gridLayout.

- **rowSize**

Defines the number of rows for the gridLayout. The 'rowSize' is overruled when more rows are defined with the gridLayoutCell control then specified with the 'rowSize'

attribute. If 'rowSize' is higher than the rows defined by the gridLayoutCell, the frame height of the gridLayout is extended.

- **width**

Defines the width of the gridLayout. If the 'width' in gridLayoutCell is specified in percent, the percentage will be calculated from the width of the gridLayout and not from the width of the form.

If the gridLayoutCell definition exceeds the 'width' of the gridLayout the gridLayoutCell content will be wrapped.

Attribute	Req.	Values	Default	Case sens.	JSP Taglib	Classlib
cellPadding	no	Numeric	0	-	cellPadding="5"	setCellPadding(5)
cellSpacing	no	Numeric	0	-	cellSpacing="5"	setCellSpacing(5)
columnSize	no	Numeric	none	-	columnSize="3"	setColumnSize(3)
debugMode	no	FALSE TRUE	FALSE	yes	debugMode="TRUE"	setDebugMode(true)
id	yes	String	none	yes	id="ZIPCode_grid"	setId("ZIPCode_grid")
rowSize	no	Numeric	none	-	rowSize="5"	setRowSize(5)
width	no	Unit	100%	no	width="500"	setWidth("500")

gridLayoutCell

Defines the cells in the gridLayout.

- **columnIndex**

Defines the horizontal position of the cell.

- **colSpan**

Defines the horizontal expansion the cell in percent. If you specify e.g. 100, the cell uses the whole gridLayout width. Cells right of this cell are omitted. A possible application for this attribute is to display headlines in the gridLayout.

- **content**

Specifies the content for the cell.

- **horizontalAlignment**

Defines the horizontal alignment of the cell content.

- LEFT
Left justifies the content of the cell.
- RIGHT
Right justifies the content of the cell.
- CENTER
Centers the content of the cell.
- CHAR
Aligns text around a specific character. Not supported by all web clients.
- JUSTIFY
Sets text in the cell left and right aligned. Not supported by all web clients.

- **id**

Identification name of the gridLayoutCell.

- **rowIndex**

Defines the vertical position of the cell.

- **style**

Defines the stylesheet to be used to display the cell.

- **verticalAlignment**

Defines the vertical alignment of the cell content.

- BASELINE
The content of the cell is aligned on the baseline line of the cell (or bottom when no baseline exists).
- BOTTOM
The content of the cell is aligned to the bottom line of the cell.
- MIDDLE
The content of the cell is aligned to the middle of the cell height.
- TOP
The content of the cell is aligned to the top line of the cell.

- width**

Defines the width of the gridLayoutCell. One column can have only one width - when you specify different widths for the same column the width defined last is taken.

Attribute	Req.	Values	Default	Case sens.	JSP Taglib	Classlib
columnIndex	yes	Numeric	none	-	columnIndex="2"	setColumnIndex(2)
colSpan	no	Numeric	0	-		setColSpan(100)
content	no	String	none	no		Text: setContent("A celltext") Component:setContent(ListBox)
heightPercentage	no	Numeric	0	-		setHeightPercentage(20)
horizontalAlignment	no	LEFT RIGHT CENTER CHAR JUSTIFY	LEFT	yes	horizontalAlignment="LEFT"	setHAlignment(CellHAlign.LEFT)
id	no	String	none	yes	id="Cell01"	setId("Cell01")
rowIndex	yes	Numeric	none	-	rowIndex="1"	setRowIndex(1)
style	no	String	none	no	style="WildStyle"	setStyle("WildStyle")
verticalAlignment	no	BASELINE BOTTOM MIDDLE TOP	MIDDLE	yes	verticalAlignment="TOP"	setVAlignment(CellVAlign.TOP)
width	no	Unit	-	-	width="20%"	setWidth("20%")

Example using the taglib

```
<hbj:gridLayout
  id="myGridLayout1"
  debugMode="True"
  width="40%"
  cellSpacing="5"
  >

  <hbj:gridLayoutCell
    rowIndex="1"
    columnIndex="1"
    width="10%"
    horizontalAlignment="RIGHT"
  >

    <hbj:button
      id="myButton1"
      text="Button"
      tooltip="Button in the 1st row"    />

  </hbj:gridLayoutCell>

  <hbj:gridLayoutCell
    id="myGridLayoutCell2"
    rowIndex="1"
    columnIndex="2"
    width="40%"
    horizontalAlignment="LEFT"
    verticalAlignment="BOTTOM"
  >

    Celltext aligned left

  </hbj:gridLayoutCell>

  <hbj:gridLayoutCell
    rowIndex="2"
    columnIndex="1"
```



```

        width="20%"
    >

        <hbj:button
            id="myButton2"
            text="Button"
            tooltip="Button in the 2nd row"    />

    </hbj:gridLayoutCell>

    <hbj:gridLayoutCell
        rowIndex="2"
        columnIndex="2"
        horizontalAlignment="RIGHT"
    >

        Celltext aligned right

    </hbj:gridLayoutCell>

</hbj:gridLayout>

```

Example using the classlib

```

Form form = (Form)this.getForm();
GridLayout gl = new GridLayout();
gl.setId("myGrid");

gl.setCellSpacing(5);
gl.setWidth("40%");
gl.setDebugMode(true);

Button button = new Button("button", "button");
GridLayoutCell cell11 = new GridLayoutCell("cell11");
cell11.setHAlignment(CellHAlign.RIGHT);
cell11.setWidth("10%");
cell11.setContent(button);
gl.addCell(1, 1, cell11);

TextView tv1 = new TextView("tv1");
tv1.setText("Celltext aligned left");

GridLayoutCell cell12 = new GridLayoutCell("cell12");
cell12.setHAlignment(CellHAlign.LEFT);
cell12.setWidth("40%");
cell12.setContent(tv1);
gl.addCell(1, 2, cell12);

Button button2 = new Button("button2", "button");
GridLayoutCell cell21 = new GridLayoutCell("cell21");
cell21.setHAlignment(CellHAlign.LEFT);
cell21.setWidth("10%");
cell21.setContent(button2);
gl.addCell(2, 1, cell21);

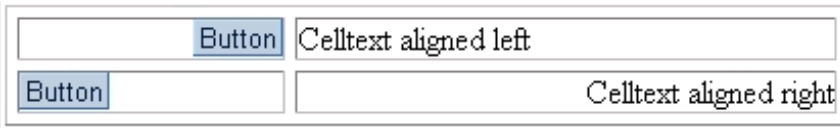
TextView tv2 = new TextView("tv2");
tv2.setText("Celltext aligned right");

GridLayoutCell cell22 = new GridLayoutCell("cell22");
cell22.setHAlignment(CellHAlign.RIGHT);
cell22.setWidth("40%");
cell22.setContent(tv2);
gl.addCell(2, 2, cell22);

form.addComponent(gl);

```

Result



Breadcrumb

[Usage](#) | [Types](#) | [Design-relevant Attributes](#) | [Related Controls](#)

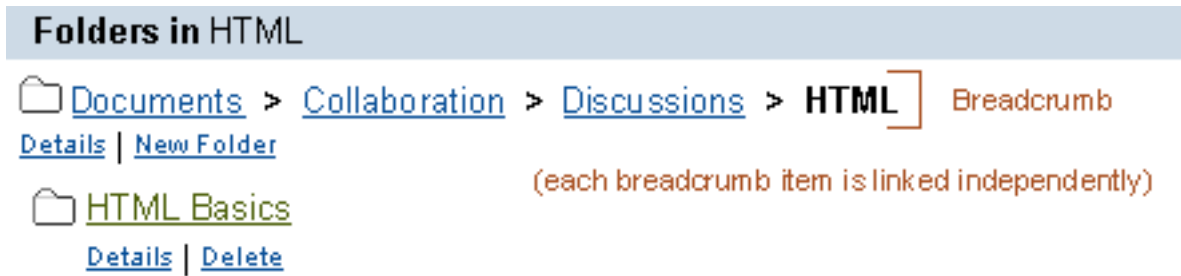


Figure 1: Search result using breadcrumbs

Breadcrumbs

- Inform users about their current position within a hierarchy, such as an application, a directory, a Website, or a document
- Allow for easy navigation back to the starting point, or to other levels within a hierarchy



Usage

Software applications and information in the Web are often organized hierarchically: General information may lead to more specific information, thus creating an information hierarchy. The breadcrumb control informs the user about the path to a specific content within such a hierarchy. For example, hit lists typically include a breadcrumb to inform users about the hierarchy level of a search result and therefore are guides to the list items. If the breadcrumb uses links in the path description, the user can move to a specific folder or topic.

An item in the breadcrumb chain is called breadcrumb item. Breadcrumb items can be defined by models or manually.



Figure 2: A wrapped breadcrumb

If the breadcrumb line becomes longer than the width of the browser window, the breadcrumb is word-wrapped like a text line (figure 2). The wrapping is done at word separators, such as blanks. If there is no word separator in the breadcrumb item string, the breadcrumb will wrap behind the breadcrumb item separator ">".



Types

Breadcrumbs can be displayed as simple path information (no link, figure 3 top), as a chain of clickable locations within the hierarchy (figure 3 center), or as **one** link that is described by the path information (figure 3 bottom).

FirstPathItem > SecondPathItem > ThirdPathItem

[FirstPathItem](#) > [SecondPathItem](#) > **ThirdPathItem**

[FirstPathItem > SecondPathItem > ThirdPathItem](#)

Figure 3: Simple path information (top), each breadcrumb item is linked independently (center), whole path is selectable (bottom)

The breadcrumb type is set through the attribute **behavior**: value SINGLELINK creates a breadcrumb, where the whole path is selectable; value DEFAULT creates a breadcrumb, where each item can be linked independently.

Usage - Types

Use the different breadcrumb types for the following purposes:

- Use **path information breadcrumbs** (figure 3 top) for indicating the location of files inside a hierarchy.
Example: A list of search results not only shows the hits themselves but also their paths.
- Use **independently linked breadcrumbs** (figure 3 center) if you want to allow users to move up and down within a hierarchy, or to jump to a certain category.
The last breadcrumb always shows the actual page and is no link.
- Use the **whole breadcrumb path as one single link** (figure 3 bottom) to inform users about the location of a link target (= the last breadcrumb item) inside a hierarchy.



Design-relevant Attributes

size

Breadcrumbs come in three different font sizes: *large*, *medium* (= default) and *small* (figure 4). Set the attribute **size** to the values LARGE, MEDIUM, or SMALL.

FirstPathItem > SecondPathItem > ThirdPathItem

FirstPathItem > SecondPathItem > ThirdPathItem

FirstPathItem > SecondPathItem > ThirdPathItem

Figure 4: Large, medium, and small size breadcrumbs

Use the text size that correspond to the size of the surrounding text. In case space requirements are tight, use smaller text sizes if available. These sizes may also be used for design and highlighting reasons.

 [Top](#)

Related Controls

[Links](#)

 [Top](#)

More Info about Breadcrumb

[Browser Compatibility](#) | [Editability in Style Editor](#) | [Accessibility – 508 Support](#)

Browser Compatibility

In Netscape 4.x and 6.x the breadcrumb path icons are not bold but have normal font weight.

[Text 1 > Text 2 > Text 3](#)

Figure 1: Netscape path icons have normal font weight



Editability in Style Editor

In the Style Editor, it is possible to modify the following attributes of the tree view control:

Group	Style	IE 5 and above	Netscape 4.7
Breadcrumb Styles	Font Color of Breadcrumb Path Icon	x	x
	Font Weight of Breadcrumb Path Icon	x	x
	Text Decoration of Active Entry	x	x
	Breadcrumb Padding	x	

Table 1: Editable styles for the tree view control

For common styles see section [HTMLB Controls and Style Editor](#) in *Customer Branding and Style Editor*.



Accessibility – 508 Support

- **Keyboard:** The breadcrumb control is inserted into the accessibility hierarchy by default if it contains links.
- **Default Description:** Is provided by the HTMLB rendering engine.
- **Application-specific Description:** Set an additional description using the *setTooltip* method if needed.

Control API for Breadcrumb (breadCrumb)

The breadcrumb represents the sequence of the visited pages (remember the story of Hansel + Gretel). It informs the user about his actual position in your application and allows easy navigation back to start page. An item in the breadcrumb chain is called breadcrumbItem. BreadcrumbItems can be defined with models or manually. If the breadcrumb line becomes longer than the web client window it is word wrapped like a text line - if there is no word separator in the breadcrumbItem value string, the line is not wrapped.

- **behavior**

The breadcrumb can behave as a

- **DEFAULT**
Each breadcrumbItem can be linked independently.
- **SINGLELINK**
The entire breadcrumb string is a single link.

- **id**

Identification name of the breadcrumb.

- **model**

Defines the model or bean which provides the breadcrumb with data.

- **nameOfKeyColumn**

Specifies the name of the column that contains the key. This is used when you use an underlying table in the model.

- **onClick**

Defines the event handling method that will be processed when the user clicks on the breadcrumb.

- **tooltip**

Defines the hint of the button which is displayed as the mouse cursor passes over the button, or as the mouse button is pressed but not released.

- **size**

Defines the text size of the breadcrumb. Possible values are:

- **LARGE**
Double the standard textsize.
- **MEDIUM**
Standard textsize.
- **SMALL**
Half of the standard textsize.

Attribute	Req.	Values	Default	Case sens.	JSP Taglib	Classlib
behavior	no	DEFAULT SINGLELINK	DEFAULT	yes	behavior="DEFAULT"	setBehavior (BreadcrumbBehavior.DEFAULT)
id	yes	String	none	yes	id="OrderConfirm"	setId("OrderConfirm")
model	no	String	none	yes	model="bean.model"	setModel((ListModel) model)
nameOfKeyColumn	no	String	none	no	nameOfKeyColumn="col1"	setNameOfKeyColumn("col1")
tooltip	no	String	none	no	tooltip="Confirm order"	setTooltip("Confirm order")
size	no	LARGE MEDIUM SMALL	none	yes	size="MEDIUM"	setSize(BreadcrumbSize.MEDIUM)

Events	Req.	Values	Default	Case sens.	JSP Taglib	Classlib
onClick	no	String	none	yes	onClick="ProcessCrumb"	setOnClick("ProcessCrumb")

breadcrumbItem

Defines the items in the breadcrumb instead of the model.

Recommendation:

We strongly recommend models to supply the breadcrumb with data.

- **key**

A string which is passed on to the event handling routine when the event occurs. A key string has to be defined and must not be empty. If the attribute 'behavior' is set to "SINGLELINK" the 'key' is set to "null" when passed on to the event handling routine.

- **value**

Defines the text string displayed in the breadcrumb. A value string has to be defined and must not be empty.

Attribute	Req.	Values	Default	Case sens.	JSP Taglib	Classlib
key	yes	String	none	no	key="EVK1"	addItem("EVK1","1stVisitedPage")
value	yes	String	none	no	value="1stVisitedPage"	see line above

Example using the taglib

```
<hbj:breadcrumb
  id="myNavigation"
  tooltip="Navigation and orientation in the application"
  onClick="ProcessbreadcrumbClick"
  size="SMALL"
  >

  <hbj:breadcrumbItem key ="EVK1" value="MainLevel" />
  <hbj:breadcrumbItem key ="EVK2" value="1stLevel" />
  <hbj:breadcrumbItem key ="EVK3" value="2ndLevel" />
  <hbj:breadcrumbItem key ="EVK4" value="3rdLevel" />

</hbj:breadcrumb>
```

Example using the classlib

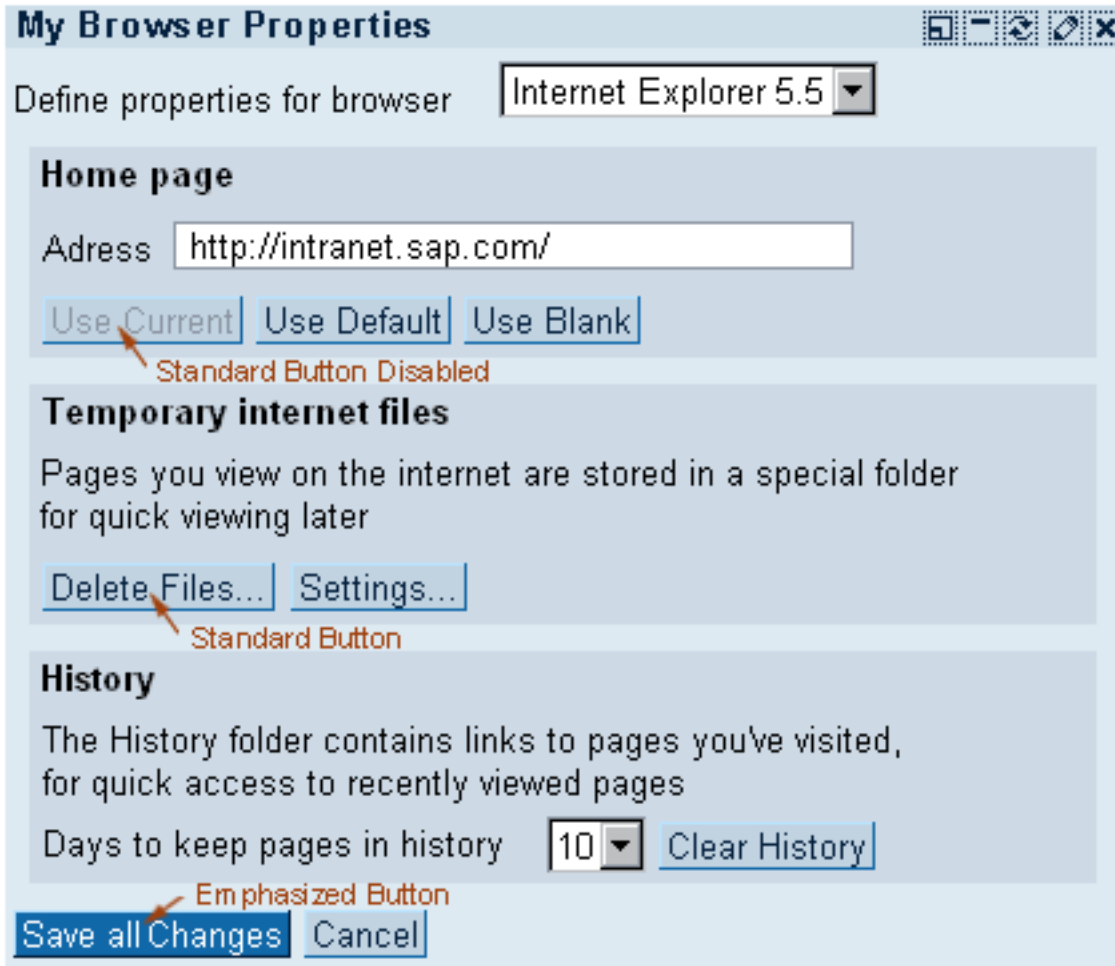
```
Form form = (Form)this.getForm();
Breadcrumb bc = new Breadcrumb("myNavigation");
bc.addItem("EVK1", "MainLevel");
bc.addItem("EVK2", "1stLevel");
bc.addItem("EVK3", "2ndLevel");
bc.addItem("EVK4", "3rdLevel");
bc.setSize(BreadcrumbSize.MEDIUM);
bc.setTooltip("Navigation and orientation in the application");
bc.setOnClick("ProcessbreadcrumbClick");
form.addComponent(bc);
```

Result

[MainLevel](#) > [1stLevel](#) > [2ndLevel](#) > [3rdLevel](#)

Button

[Usage](#) | [Types](#) | [Design-relevant Attributes](#) | [Related Controls](#)



Buttons are used for explicit functions that refer to a given object or serve for navigational purposes

Figure 1: Example of an iView containing groups with buttons and two buttons belonging to the iView itself



[Top](#)

Usage

Use buttons only for few and very important functions. A lot of buttons make a screen look heavy and complex. Therefore, when in doubt about whether to use a link or a button, go with a link. Because of their optical weight and their visual dominance, buttons qualify for this purpose.

Note: For a detailed discussion of when to use buttons and when to use links, see [Link](#).

Pressing the *Enter* key activates the default function (mostly, but not necessarily, identical to the emphasized button's function).

Usage guidelines for the different button types and sizes are presented [below](#).

Labeling

Use title case for button labels. Use the ellipsis character ("...") on button labels to indicate that the command needs further information to execute. Typically, the user is presented a dialog to fill in missing information.

Note: Title case means that the first letter of each word is capitalized, except for certain small words, such as articles and short prepositions.

Choose the button's function description carefully; try to be as explicit as possible. For complex interactions, use verb-noun combinations, e.g. "Search Database". If the context is clear, i.e. if the action can only be applied to one object, it is sufficient to use a single verb as the button's label ("Search"). For shufflers and comparable elements, you can also use a simple "Go".

Positioning and Design Alternatives

For detailed button positioning rules see [Button Positioning](#).

Buttons are similar in function to links. For a discussion of when to use buttons and when to use links, see [Link](#).



[Top](#)

Types

HTMLB offers three different button types (attribute **design**, values STANDARD, SMALL, EMPHASIZED):

-  Standard Button
-  Small Button
-  Emphasized Button

Figure 2: Standard, small, and emphasized button

In the following, we list usage guidelines for these types.

Usage - Emphasized Buttons vs. Standard Button

For functions that complete a task, always use an **emphasized** button. In all other cases use a standard button, or a small button (see below).

Rationale: Users need to realize that a certain function completes a task and know about - possibly negative - consequences.

The emphasized button is always the leftmost button if it is a member of a button group.

Usage - Standard Button vs. Small Button

- Use standard size buttons for frequently-used functions
- Use small size buttons for seldom-used functions
- Use small buttons in (exceptional) cases where space is scarce
- Do not mix both sizes within groups of elements



[Top](#)

Design-relevant Attributes

All buttons are available in an **enabled** and **disabled** state (Boolean attribute **disabled**).



Figure 3: Disabled buttons

Usage - Disabled Buttons vs. Hidden Buttons

Disabled buttons indicate that a function is not available. Therefore, use disabled buttons for functions that are temporarily disabled. For example, a certain system state, such as an error, may prevent that the user can execute a function.

Hide buttons that are permanently not available for a user. For example, a user may not have the permission to perform certain actions.



[Top](#)

Related Controls

[Link](#), [Input Field](#), [Group](#), [Table View](#)



[Top](#)

More Info about Button

[Browser Compatibility](#) | [Editability in Style Editor](#) | [Accessibility – 508 Support](#)

Browser Compatibility

In Netscape 4.X buttons will be displayed as standard HTML buttons



Figure 1: Netscape button



Editability in Style Editor

In the Style Editor, it is possible to modify the following attributes of the button control in Internet Explorer 5 and above (Netscape 4.7 uses standard HTML buttons):

Group	Style	IE 5 and above
Button Styles	Text Padding	x
	Text Decoration	x
	Font Weight	x
	Border Width and Style	x
Standard Buttons	Standard Background Color	x
	Standard Border Color	x
	Standard Font Color	x
	Standard Hover Color	x
	Disabled Standard Background Color	x
Emphasized Buttons	Emphasized Background Color	x
	Emphasized Border Color	x
	Emphasized Font Color	x

	Emphasized Hover Color	x
	Disabled Emphasized Background Color	x
	Disabled Emphasized Font Color	x
Standard-Sized Buttons	Height	x
Small-Sized Buttons	Small Height	x
Text Wrap	White Space	x

Table 1: Editable styles for the button control

For common styles see section [HTMLB Controls and Style Editor](#) in Customer *Branding and Style Editor*.



[Top](#)

Accessibility – 508 Support

- **Keyboard:** The button inserted into the accessibility hierarchy by default - including the button state (e.g. disabled) and type (e.g. emphasized).
- **Default Description:** Is provided by the HTMLB rendering engine.
- **Application-specific Description:** Set an additional description using the *setTooltip* method if needed.
An additional description is needed if users need more specific information or instructions. In general, the description has to be extended if a button introduces an interaction that cannot be recognized by a blind user. For example, the descriptions needs to be extended if the button opens a new window.



[Top](#)

Control API for Button (button)

Provides any type of functionality in your application at the touch of the button. Hints can be displayed as the mouse cursor passes over the button, or as the mouse button is pressed but not released.

- **design**

Defines the size and highlighting of the button.

- **STANDARD**
Displays the button with the standard background and text color.
- **SMALL**
Displays the button with the standard background and text color and half of the STANDARD size.
- **EMPHASIZED**
Displays the button with the highlighted background and text color. You can also refer to the emphasized button as default button. Therefore only one emphasized button per form can be defined. If you use more than one "EMPHASIZED" button, the last button defined becomes "EMPHASIZED".

- **disabled**

A boolean value that defines if the button is clickable. If the button is disabled it sends no event when you press a mouse button on the button. A disabled button has a different text color to show the user that it is disabled.

- **encode**

A Boolean value that defines how the text in the button is interpreted. HTML text formatting commands (e.g. <h1>, <i> etc.) can be used to change the display of the text. If there are no formatting commands in the text string, the encode attribute has no effect.

Example

```
text="<h1><i>Important</i></h1>
```

Important

```
encode = "false"
```

the text string is rendered by interpreting the formatting commands.

```
Encode = "true" <h1><i>Important</i></h1>
```

the formatting commands are displayed and not interpreted.

- **id**

Identification name of the button.

- **onClick**

Defines the event handling method that will be processed when the user clicks on the enabled button. If you do not define a 'onClick' event the button can be clicked but no event is generated.

- **onClientClick**

Defines the JavaScript fragment that is executed when the user clicks on the button. If both events ('onClick' and 'onClientClick') are specified, the 'onClientClick' event handling method is activated first. By default the 'onClick' event handling method is activated afterwards. In the JavaScript fragment you can cancel the activation of the 'onClick' event handling method with the command

```
htmlbevent.cancelSubmit=true;
```

The 'onClientClick' event is useful to preprocess the form and only send the form to client if the preprocessing was successful (e.g. date validation, valid number format etc.) to save client/server interaction.

Example

A button click usually activates the client/server interaction. If an input field has to be filled out for further processing, the JavaScript fragment can check the necessary input on the client side and display a message if the necessary input is missing, without server interaction.

Note

to use JavaScript the JSP has to use the **pagetag** (set [pagetag](#)).

- **text**

Defines the string of text placed centered on the button. If no text should be displayed in the button an empty string (null) can be used. The width of the button is automatically adjusted to the length of the text.

- **width**

Defines the width of the button. The width of the button is automatically adjusted to the length of the 'text'. To see an effect of the 'width' attribute, 'width' has to be set higher as the width defined through the length of the 'text' string. The text string of the button is always placed centered on the button. If an empty (null) 'text' string is set no 'text' attribute is defined the width of the button is set according to the 'width' attribute.

- tooltip**
 Defines the hint of the button which is displayed as the mouse cursor passes over the button, or as the mouse button is pressed but not released.

Attribute	Req.	Values	Default	Case sens.	JSP Taglib	Classlib
design	no	STANDARD SMALL EMPHASIZED	STANDARD	yes	design="STANDARD"	setDesign(ButtonDesign.STANDARD)
disabled	no	TRUE FALSE	FALSE	yes	disabled="FALSE"	setDisabled(true)
encode	no	TRUE FALSE	TRUE	yes		setEncode(false)
id	yes	String	none	yes	id="OrderConfirm"	setId("OrderConfirm")
text	no	String	none	no	text="Confirm"	setText("Confirm")
width	no	Unit	10	-	width="125px"	setWidth("125px")
tooltip	no	String	none	no	tooltip="Confirm order"	setTooltip("Confirm order")

Events	Req.	Values	Default	Case sens.	JSP Taglib	Classlib
onClick	no	String	none	yes	onClick="ProcessConfirm"	setOnClick("ProcessConfirm")
onClientClick	no	String	none	yes	onClientClick="JavaScript"	setOnClientClick("JavaScript")

Example using the taglib

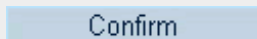
```
<hbj:button
  id="OrderConfirm"
  text="Confirm"
  width="125px"
  tooltip="Click here to confirm order"
  onClick="ProcessConfirm"
  disabled="false"
  design="STANDARD"

/>
```

Example using the classlib

```
Form form = (Form)this.getForm();
Button button = new Button("button", "button");
button.setText("Confirm");
button.setWidth("125px");
button.setTooltip("Click here to confirm order");
button.setOnClick("ProcessConfirm");
button.setDisabled(false);
button.setDesign(ButtonDesign.STANDARD);
form.addComponent(button);
```

Result



Chart

[Usage](#) | [Types](#) | [Design-relevant Attributes](#) | [Related Controls](#)



The chart control displays a chart; it offers a variety of different chart types.

Figure 1: A stacked bar chart as an example of a chart control



[Top](#)

Usage

A chart displays data that are relevant for the user in a graphical representation so that the characteristics of the data and their relations are easy to capture for the user.

In cases where it is important for users to know the exact values behind the data, an alternate view may present the data in a [table](#) as numbers or texts. A button should allow users to toggle between the diagram and the table view.

Note: For thorough information on charts and their uses see *Recommendations for Charts and Graphics* in the *SAP Design Guild*.

Positioning

A chart can be presented in an iView; in this case, it should comprise the main part of the iView. A chart can be combined with other screen elements to allow for interaction with the chart.

Legend

A legend explains the colors used in a chart. For the chart control the legend is generated automatically. It can be placed to the right (preferable) or below the graph; other positions are also possible, but should not be used. The position of the legend is set by the attribute **legendPosition** using the values EAST, NORTH, SOUTH, WEST, or NONE for no legend.

Order of Screen Elements

If there are further interaction elements, obey the following order:

- A *filter of shuffler* can be placed *above* the chart, if data can be selected from several sets or if the amount of data has to be reduced.
- Then follows the *chart*.
- Place *legends* or other text right to the image or below it, depending on the format of the chart (see above)

Chart

- Place pushbuttons for chart-related functionality and status information (e.g. zoom factor) below the chart and left align them.
- If there exists an *alternative table view*, a button *below* the table allows to toggle between diagram and table view.

Functionality

Typical functionality, which charts may offer, are:

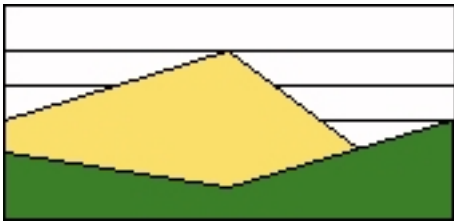
- Switch between chart view and table view
- Zooming and panning (nor available for the chart control)
- Drill-down



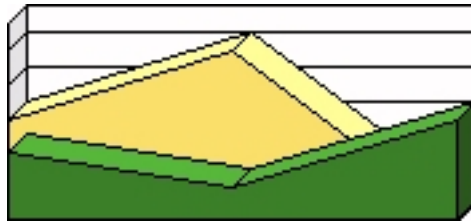
[Top](#)

Types

Charts are available in a number of types. These are selected using the attribute **chartType**. Below you find an overview of the available chart types and the respective values for attribute **chartType**.



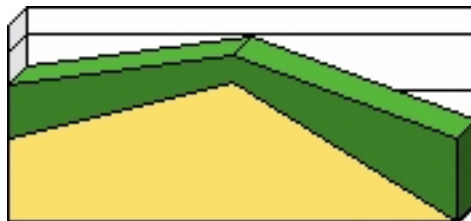
Area chart (**chartType** = AREA)



3D Area chart (**chartType** = AREA_3D)



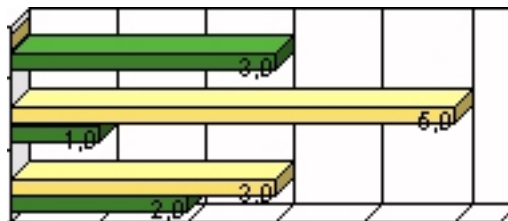
Stacked area chart (**chartType** = AREA_STACKED)



Stacked 3D area chart (**chartType** = AREA_STACKED_3D)

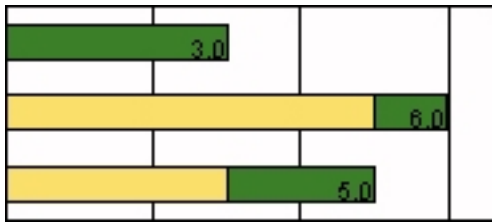


Bar chart (**chartType** = BARS)

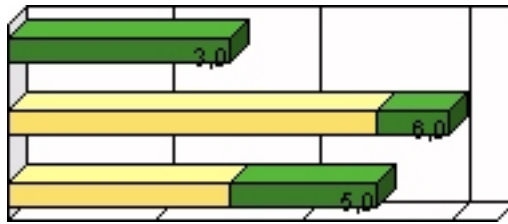


3D bar chart (**chartType** = BARS_3D)

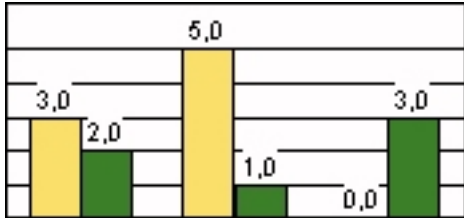
Chart



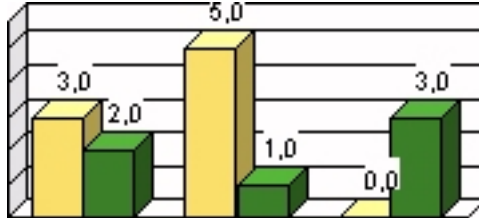
Stacked bar chart (**chartType** = BARS_STACKED)



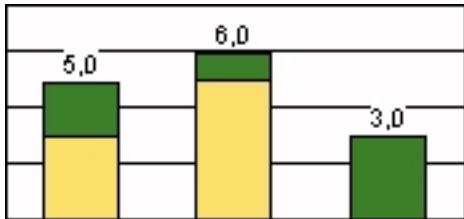
Stacked 3D bar chart (**chartType** = BARS_STACKED_3D)



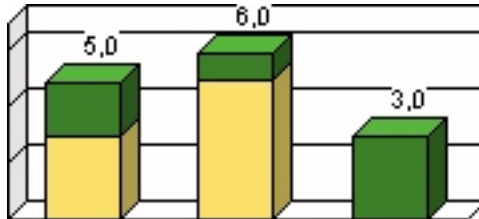
Column chart (**chartType** = COLUMNS)



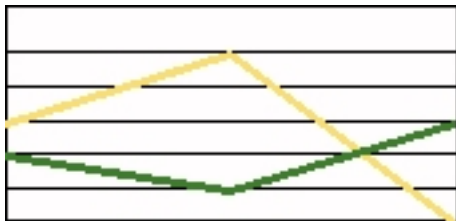
3D column chart (**chartType** = COLUMNS_3D)



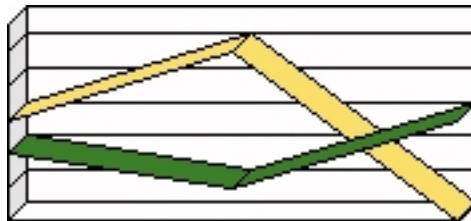
Stacked column chart (**chartType** = COLUMNS_STACKED)



Stacked 3D column chart (**chartType** = COLUMNS_STACKED_3D)



Line chart (**chartType** = LINES)



3D line chart (**chartType** = LINES_3D)



Pie chart (**chartType** = PIE)



3D pie chart (**chartType** = PIE_3D)

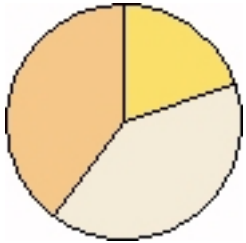
Chart



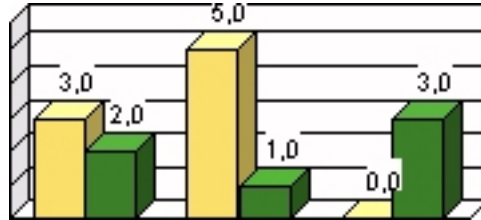
Extruded pie chart (**chartType** = PIE_EX)



Extruded 3D pie chart (**chartType** = PIE_EX_3D)



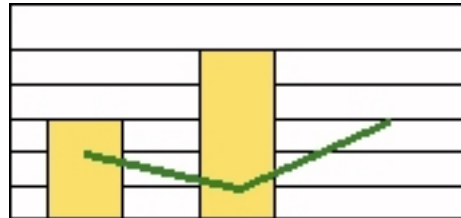
Split pie chart (**chartType** = PIE_SPLIT)



Bitmap chart (**chartType** = BITMAP) - arbitrary bitmap



Pyramid chart (**chartType** = PYRAMID)



Trend chart (**chartType** = TREND)

Figure 2: Overview of the available chart types

Using Chart Types - Overview

The following table overview presents usage hints for the available chart types.

Chart Type	Typical Applications	Variants, Remarks
Area	Cumulated totals (numbers or percentages) over time	Percentage, Cumulative
Column/Bar	Observations over time or under different conditions; data sets must be small	Vertical (columns), horizontal (bars); multiple columns/bars, columns/bars centered at zero
Segmented Column/Bar	Proportional relationships over time	May be scaled to 100%

Line, Curve	Trends, functional relations	Data point connected by lines or higher order curves
Pie	Proportional relationships at a point in time	Segments may be pulled out of the the pie for emphasis (exploded pie chart)

Table 1: Chart types and their applications and variants

Note that there are chart types that may better fit the intended purpose than the available ones. For more information, consult *Recommendations for Charts and Graphics* in the *SAP Design Guild*.



[Top](#)

Design-relevant Attributes

Look and behavior of the chart control can be controlled by a number of attributes:

- **Position and Visibility of Legend:** Attribute **legendPosition** allows to hide or show and position the legend (values NONE, EAST, NORTH, SOUTH, WEST).
- **Color Order:** Use attribute **colorOrder** to control the sequence of colors (values are DEFAULT, STRAIGHT, REVERSE, and SNAKE).
- **Height and Width:** Attributes **height** and **width** allow to set the size of the chart.
- **Display of Values and Titles:** A number of attributes is at your disposal to control the look and position of labels for values and categories. For more information see page [Control API for Chart](#).

For detailed information on attributes see page [Control API for Chart](#).



[Top](#)

Related Controls

[Image, Table View](#)



[Top](#)

More Info about Chart

[Browser Compatibility](#) | [Editability in Style Editor](#) | [Accessibility – 508 Support](#)

Browser Compatibility

No known issues - charts are like images



Editability in Style Editor

Customers cannot customize charts via the Style Editor. The tool offers no editable styles related to charts placed as portal content.



Accessibility – 508 Support

Charts are like images; therefore, the same measures apply.

- **Keyboard:** Charts are **not** inserted into the accessibility hierarchy by default.
- **Default Description:** Is provided by the HTMLB rendering engine
- **Application-specific Description:** Set an additional description using the *setTooltip* method if needed. Do **not** use the *setAlt* method that sets the alternate text (*alt* attribute).



Control API for Chart (chart)

A control to visualize data in annotated diagrams.

- **axisMaxVal**

Used to calculate the annotation and scaling of the chart. 'axisMaxVal' specifies the maximum value the axis is annotated with. If 'axisMaxVal' is not specified or a value is specified that is less than the maximum value provided by the model, 'axisMaxVal' is set to the maximum value of the model.

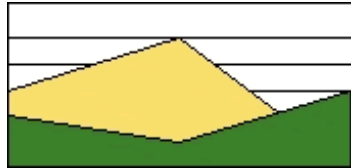
- **axisMinVal**

Used to calculate the annotation and scaling of the chart. 'axisMinVal' specifies the minimum value of the axis. If 'axisMinVal' is not specified or a value is specified that is greater than the minimum value provided by the model, 'axisMinVal' is set to 0.

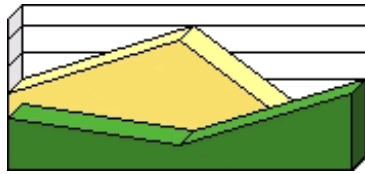
- **chartType**

Controls the style in which the data is displayed.

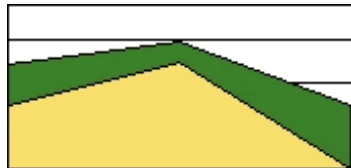
- AREA



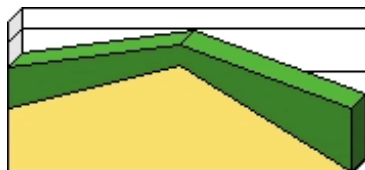
- AREA_3D



- AREA_STACKED



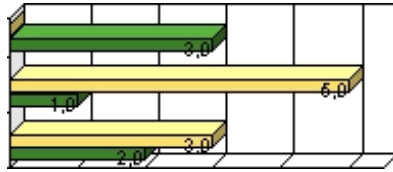
- AREA_STACKED_3D



- BARS



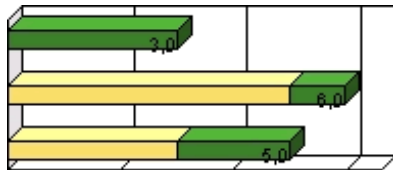
o BARS_3D



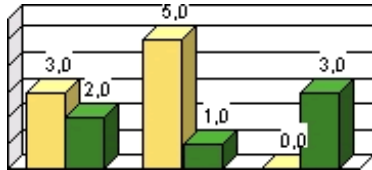
o BARS_STACKED



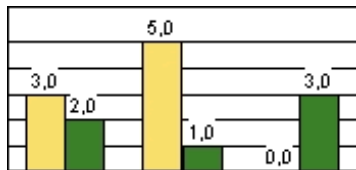
o BARS_STACKED_3D



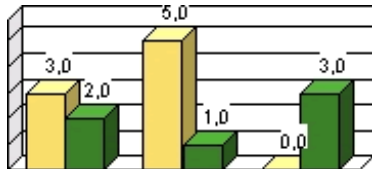
o BITMAP



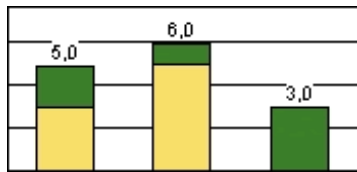
o COLUMNS



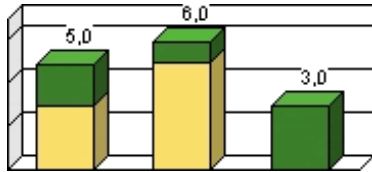
o COLUMNS_3D



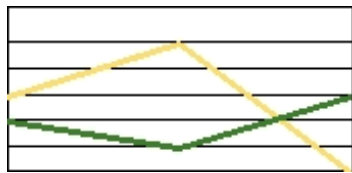
o COLUMNS_STACKED



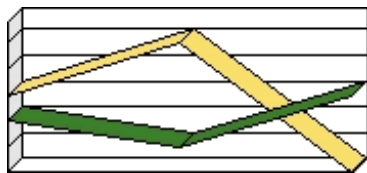
o COLUMNS_STACKED_3D



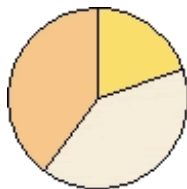
o LINES



o LINES_3D



o PIE



o PIE_3D



o PIE_EX



- PIE_EX_3D



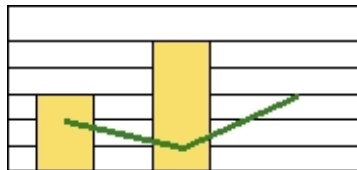
- PIE_SPLIT



- PYRAMID



- TREND



- **colorOrder**

The various types of the chart control all use the same set of colors to visualize the values of a data set, but explore the space of possible colors on different paths. The following pictures show the three predefined color schemes and the chart types using them.

- STRAIGHT

This color scheme is used by the various area, column and bar chart graphs.

	1	2	3	4	5	6	7	8
	255 255 255		150 81 54		245 244 231	222 222 200		255 0 255
9	255 248 163	250 225 107	248 215 83	243 192 28	240 180 0	245 242 226	247 179 87	79 85 106
17	169 204 143	130 177 106	92 151 70	61 129 40	30 108 11	245 238 217	247 181 91	163 204 75
25	178 200 217	119 157 191	62 117 167	32 95 154	0 72 140	245 233 208	247 186 102	255 235 106
33	190 163 122	144 122 82	122 101 62	99 82 43	51 38 0	245 228 195	247 190 111	255 173 136
41	243 170 121	235 137 83	225 102 42	220 83 19	216 64 0	246 222 171	246 196 124	77 166 25
49	181 181 169	138 141 130	116 121 111	93 100 90	67 76 67	246 217 171	246 202 137	255 235 0
57	230 165 164	214 112 123	196 56 79	188 28 57	179 0 35	246 212 161	246 207 149	230 0 0

o SNAKE

This color scheme is used by the pie chart graphs.

	1	2	3	4	5	6	7	8
	255 255 255		150 81 54		245 244 231	222 222 200		255 0 255
9	255 248 163	250 225 107	248 215 83	243 192 28	240 180 0	245 242 226	247 179 87	79 85 106
17	169 204 143	130 177 106	92 151 70	61 129 40	30 108 11	245 238 217	247 181 91	163 204 75
25	178 200 217	119 157 191	62 117 167	32 95 154	0 72 140	245 233 208	247 186 102	255 235 106
33	190 163 122	144 122 82	122 101 62	99 82 43	51 38 0	245 228 195	247 190 111	255 173 136
41	243 170 121	235 137 83	225 102 42	220 83 19	216 64 0	246 222 171	246 196 124	77 166 25
49	181 181 169	138 141 130	116 121 111	93 100 90	67 76 67	246 217 171	246 202 137	255 235 0
57	230 165 164	214 112 123	196 56 79	188 28 57	179 0 35	246 212 161	246 207 149	230 0 0

- REVERSE

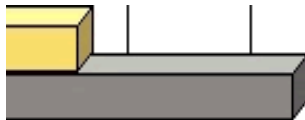
This color scheme is used by the line chart graphs.

	1	2	3	4	5	6	7	8
	255 255 255		150 81 54		246 244 231	222 222 200		255 0 255
9	255 248 163	250 225 107	248 215 83	243 192 28	240 180 0	245 242 226	247 179 87	79 85 106
17	169 204 143	130 177 106	92 151 70	61 129 40	30 108 11	245 238 217	247 181 91	163 204 75
25	178 200 217	119 157 191	62 117 167	32 90 154	0 72 140	245 233 208	247 186 102	255 235 106
33	190 163 122	144 122 82	122 101 62	99 82 43	51 38 0	245 228 195	247 190 111	255 173 136
41	243 170 121	235 137 83	225 102 42	220 83 19	216 64 0	246 222 171	246 196 124	77 166 25
49	181 181 169	138 141 130	116 121 111	93 100 90	37 76 67	246 217 171	246 202 137	255 235 0
57	230 165 164	214 112 123	196 56 79	188 28 57	179 0 35	246 212 161	246 207 149	230 0 0

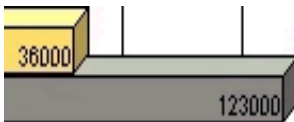
- **displayObjectValues**

A boolean value that controls if the values is displayed with the object .

displayObjectValues = "false"



displayObjectValues = "true"



Note: Not all 'ChartType' settings support the display of values. The example pictures in the 'chartType' attribute description show which types support the display of values.

- **height**

Defines the overall height of the chart. The height includes the 'title', 'titleValues' and 'legendPosition'.

- **id**

Identification name of the chart.

- **legendPosition**

Controls the position of the legend.

- EAST
Places the legend on the right side of the chart.
- NONE
The legend will be suppressed.
- NORTH
Places the legend on top of the chart.

Control API for Chart (chart)

- SOUTH
Places the legend under the chart.
- WEST
Places the legend left of the chart.

- **model**

Defines the model which provides the chart with data.

- **title**

Specifies the headline of the chart.

- **titleCategories**

Specifies the axis title for the categories.

- **titleValues**

Specifies the axis title for the values.

- **visible**

A boolean value that defines if the chart is visible.

- **width**

Defines the width of the chart. The width include 'titleCategories' and the 'legendPosition'.

Attribute	Req.	Values	Default	case sens	JSP Taglib	Classlib
axisMaxVal	no	Numeric	defined by model	-	axisMaxVal="2000"	setAxisMaxVal(2000)
axisMinVal	no	Numeric	defined by model	-	axisMinVal="100"	setAxisMinVal(100)
chartType	no	AREA AREA_3D AREA_STACKED AREA_STACKED_3D BARS BARS_3D BARS_STACKED BARS_STACKED_3D BITMAP COLUMNS COLUMNS_3D COLUMNS_STACKED COLUMNS_STACKED_3D LINES LINES_3D PIE PIE_3D PIE_EX PIE_EX_3D PIE_SPLIT PYRAMID TREND	BARS_3D	yes	chartType="PIE"	setChartType(ChartType.PIE)
colorOrder	no	DEFAULT STRAIGHT REVERSE SNAKE	DEFAULT	yes	colorOrder="SNAKE"	setColorOrder (ChartColorOrder.SNAKE)
displayObjectValues	no	FALSE TRUE	FALSE	yes	displayObjectValues="TRUE"	setDisplayObjectValues(true)
height	no	Unit	200	-	height="300"	setHeight("300")
id	yes	String	none	yes	id="VacationPlanner"	setId("VacationPlanner")

legendPosition	no	EAST NONE NORTH SOUTH WEST	EAST	yes	legendPosition="SOUTH"	setLegendPosition (ChartLegendPosition.SOUTH)
model	yes	String	none	yes	model="myBean.model"	setModel ((IChartModel) model)
title	no	String	none	no	title="Bill board chart"	setTitle("Bill board chart")
titleCategories	no	String	none	no	titleCategories="Brand"	setTitleCategories("Brand")
titleValues	no	String	none	no	titleValues("Overview")	setTitleValues("Overview")
visible	no	FALSE TRUE	TRUE	yes	visible="FALSE"	setVisible(false)
width	no	Unit	500	-	width="400"	setWidth("400")

Example using the taglib

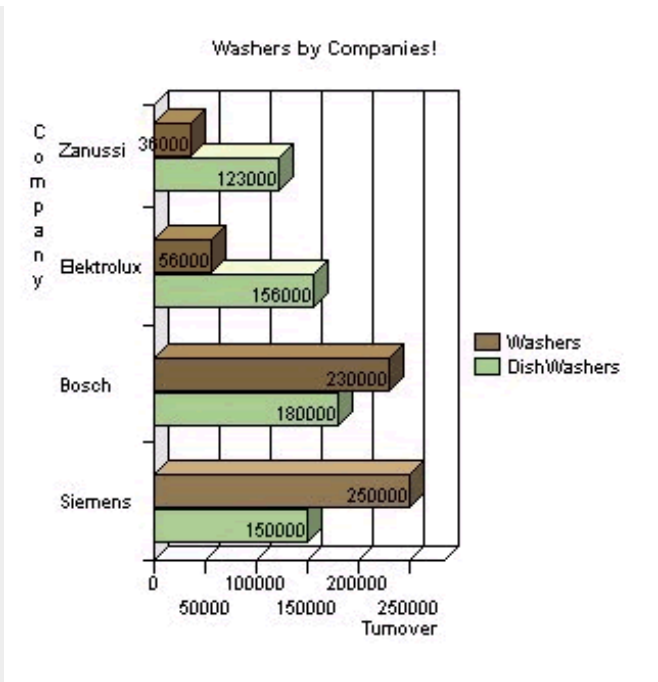
```
<hbj:chart
    id="myChart1"
    model="myChartBean.model"
    visible="true"
    displayObjectValues="true"
    titleCategories="Company"
    titleValues="Turnover"
    title="Washers by Companies!"
    chartType="BARS_3D"
    legendPosition="EAST"
    colorOrder="STRAIGHT"
/>
```

Example using the classlib.

```
Form form = (Form)this.getForm();
Chart myChart = new Chart();
myChart.setVisible(true);
myChart.setDisplayObjectValues(true);
myChart.setTitleCategories("Company");
myChart.setTitleValues("Turnover");
myChart.setTitle("Washers by Companies!");
myChart.setChartType(ChartType.BARS_3D);
myChart.setLegendPosition(ChartLegendPosition.EAST);
myChart.setColorOrder(ChartColorOrder.STRAIGHT);

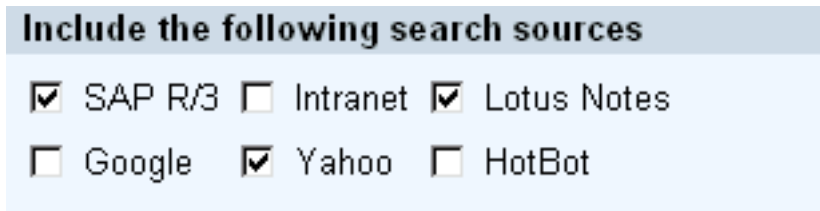
MyVecBean myVecBean = new MyVecBean();
IChartModel chartModel = myVecBean.getModel();
myChart.setModel(chartModel);
form.addComponent(myChart);
```

Result



Checkbox

[Usage](#) | [Design-relevant Attributes](#) | [Related Controls](#)



Checkboxes offer one or multiple choices to the user. The user can select none, one, or as many options as desired in a group of checkboxes.

Figure 1: A checkbox group



Usage

Checkboxes are the appropriate elements when users can choose between multiple options. They can appear as a single checkbox or grouped.

In a checkbox group the choices are not exclusive, that is, a user can check several options in a group. If you need single-selection use radio buttons or a dropdown list box, instead.

Checkbox Group

For groups of checkboxes use the checkbox group control if applicable. This control allows to arrange checkboxes in one column, one row, or in a matrix-like fashion.

Note: It is not possible to determine the horizontal spacing within a checkbox group. If you need a different spacing than that supplied by the checkbox group control, use single checkboxes and a [grid layout](#) control if applicable.

Arrangement and Design Alternatives

For details on the arrangement of checkboxes as well as for design alternatives see [Forms - Using Checkboxes](#).



Design-relevant Attributes

Checkboxes have the **disabled** and **checked** attributes. Set **disabled** to TRUE if a checkbox cannot be checked or unchecked by a user temporarily. Set **checked** to TRUE to preset a checkbox to the checked state. Use attribute **text** to set the descriptive label text for a checkbox.

You can also set the column count for checkbox groups (attribute **columnCount**).



[Top](#)

Related Controls

[Radio Button](#), [Dropdown List Box](#), [List Box](#), [Label](#), [Grid Layout](#)



[Top](#)

More Info about Checkbox

[Browser Compatibility](#) | [Editability in Style Editor](#) | [Accessibility – 508 Support](#)

Browser Compatibility

The checkbox renders in every supported browser.



Editability in Style Editor

The checkbox itself renders as the standard browser control. Style Editor changes can be made to the corresponding label.

Checkbox Groups

There is no editability for checkbox groups in the style editor.



Accessibility – 508 Support

Checkboxes have to be used in combination with the label control, which points to the assigned checkbox if they are used with a label to the left of the checkbox. This ensures, that screenreaders are aware of the relationship between the both elements and can read the correct label to the according checkbox.

- **Keyboard:** Checkboxes are inserted into the accessibility hierarchy by default.
- **Default Description:** Is provided by the HTMLB rendering engine.
- **Application-specific Description:** Set an additional description using the *setTooltip* method if needed.
- **Label:** Has to be connected to a label control for left-hand labels (use method *setLabelFor* for identifying the corresponding checkbox or checkbox group).



Control API for Checkbox (checkbox)

A control, consisting of a graphic and associated text, that a user clicks to select or deselect an option. A check mark in the checkbox graphic indicates that the option is selected.

- checked**
 A boolean value that indicates if a checkbox is selected. "True" shows a check mark in a checkbox and indicates that the checkbox is selected, "false" leaves the checkbox empty and indicates that the checkbox is not selected.
- disabled**
 A boolean value that defines if the checkbox is clickable. If the checkbox is disabled it is not selectable. A disabled checkbox has a different background color for the checkbox graphic and if the checkbox is checked the a different color for the check mark.
- encode**
 A boolean value that defines how the checkbox text is interpreted. HTML text formatting commands (e.g. <h1>, <i> etc.) can be used to change the display of the checkbox text. If there are no formatting commands in the checkbox text string, the encode attribute has no effect.

Example

```
text="<h1><i>Important</i></h1>"
```



encode = "false"
the text string is rendered by interpreting the formatting commands.

Encode = "true" <h1><i>Important</i></h1>
the formatting commands are displayed and not interpreted

- id**
 Identification name of the checkbox.
- key**
 A string which is assigned to the checkbox when the form is sent to the server. A key string has be defined and must not be empty.
- onClick**
 Defines the action that will be processed when the user clicks on the enabled checkbox. If you do not define a 'onClick' Event the checkbox can be clicked but no event is generated.
- text**
 Defines the string of text placed right of the check box graphic. If no text should be displayed an empty string (null) can be used. See 'encode' for a formatting example with embedded HTML commands.
- tooltip**
 Defines the hint of the checkbox which is displayed as the mouse cursor passes over the checkbox, or as the mouse button is pressed but not released.

attribute	req.	values	default	case sens.	JSP taglib	classlib
checked	no	TRUE FALSE	FALSE	yes	checked="TRUE"	setChecked(true)
disabled	no	TRUE FALSE	FALSE	yes	disabled="TRUE"	setDisabled(true)
encode	no	TRUE FALSE	TRUE	yes	encode="FALSE"	setEncode(false)
id	yes	String	none	yes	id="CheckCPU"	
key	yes	String	none	no	key="chk_k1"	setKey("chk_k1")
text	no	String	none	no	text="CPU status"	setText("CPU status")
tooltip	no	String	none	no	tooltip="Check CPU status"	setTooltip("Check CPU status")

events	req.	values	default	case sens.	JSP taglib	classlib
onClick	no	String	none	yes	onClick="process_checkbox"	setOnClick("process_checkbox")

Example

```
<hbj:checkbox  
  id="CheckCPU"  
  text="CPU status"  
  key="chk_k1"  
  tooltip="Check CPU status"  
  disabled="false"  
  checked="true"  
  
/>
```

Result

CPU status

Control API for Checkbox Group (checkboxGroup)

Places several checkboxes in tabular form. The setting of an individual checkbox is independent of other checkboxes - that is, more than one checkbox in a set can be checked at any given time.

- **columnCount**

Defines the amount of columns in which the checkbox items are divided.

Example

If the columnCount is set to 2 and you define 7 checkbox items the result is

```

 Option 1  Option 2
 Option 3  Option 4
 Option 5  Option 6
 Option 7

```

- **id**

Identification name of the checkboxGroup.

attribute	req.	values	default	case sens.	JSP taglib	classlib
columnCount	no	String	1	-	columnCount="2"	setColumnCount(2)
id	yes	String	none	yes	id="CPUCheckGroup"	

Example

```

<hbj:checkboxGroup
  id="CPUCheckGroup"
  columnCount="2"
  >

  <hbj:checkbox
    id="CheckCPUStat"
    text="CPU status"
    key="chk_Stat"
    tooltip="Check CPU status"
    disabled="false"
    checked="true"
  />

  <hbj:checkbox
    id="CheckCPUUsage"
    text="CPU usage"
    key="chk_Use"
    tooltip="Actual CPU usage"
    disabled="false"
    checked="false"
  />

  <hbj:checkbox
    id="CheckProc"
    text="Processes"
    key="chk_Process"
    tooltip="Show active processes"
    disabled="false"
    checked="false"
  />

  <hbj:checkbox
    id="CheckProc"
    text="Resources"
    key="chk_Res"
    tooltip="Available resources"
    disabled="false"
  />

```

```
        checked="true"  
    />  
</hbj:checkboxGroup>
```

Result

CPU status Processes
 CPU usage Resources

Date Navigator

[Usage](#) | [Design-relevant Attributes](#) | [Related Controls](#)

««		January 2002							»»	
	Su	Mo	Tu	We	Th	Fr	Sa			
1	30	31	1	2	3	4	5			
2	6	7	8	9	10	11	12			
3	13	14	15	16	17	18	19			
4	20	21	22	23	24	25	26			
5	27	28	29	30	31	1	2			

Figure 1: Example of the date navigator displaying one month

««		January 2002							February 2002							March 2002							April 2002							»»				
	Su	Mo	Tu	We	Th	Fr	Sa		Su	Mo	Tu	We	Th	Fr	Sa		Su	Mo	Tu	We	Th	Fr	Sa		Su	Mo	Tu	We	Th	Fr	Sa			
1	30	31	1	2	3	4	5	5	27	28	29	30	31	1	2	9	24	25	26	27	28	1	2	14	31	1	2	3	4	5	6			
2	6	7	8	9	10	11	12	6	3	4	5	6	7	8	9	10	3	4	5	6	7	8	9	15	7	8	9	10	11	12	13			
3	13	14	15	16	17	18	19	7	10	11	12	13	14	15	16	7	10	11	12	13	14	15	16	16	14	15	16	17	18	19	20			
4	20	21	22	23	24	25	26	8	17	18	19	20	21	22	23	12	17	18	19	20	21	22	23	17	21	22	23	24	25	26	27			
5	27	28	29	30	31	1	2	9	24	25	26	27	28	1	2	13	24	25	26	27	28	29	30	18	28	29	30	1	2	3	4			
																	14	31	1	2	3	4	5	6										
		May 2002							June 2002							July 2002							August 2002											
	Su	Mo	Tu	We	Th	Fr	Sa		Su	Mo	Tu	We	Th	Fr	Sa		Su	Mo	Tu	We	Th	Fr	Sa		Su	Mo	Tu	We	Th	Fr	Sa			
18	28	29	30	1	2	3	4	22	26	27	28	29	30	31	1	27	30	1	2	3	4	5	6	31	28	29	30	31	1	2	3			
19	5	6	7	8	9	10	11	23	2	3	4	5	6	7	8	28	7	8	9	10	11	12	13	32	4	5	6	7	8	9	10			
20	12	13	14	15	16	17	18	24	9	10	11	12	13	14	15	29	14	15	16	17	18	19	20	33	11	12	13	14	15	16	17			
21	19	20	21	22	23	24	25	25	16	17	18	19	20	21	22	30	21	22	23	24	25	26	27	34	18	19	20	21	22	23	24			
22	26	27	28	29	30	31	1	26	23	24	25	26	27	28	29	31	28	29	30	31	1	2	3	35	25	26	27	28	29	30	31			
									27	30	1	2	3	4	5	6																		
		September 2002							October 2002							November 2002							December 2002											
	Su	Mo	Tu	We	Th	Fr	Sa		Su	Mo	Tu	We	Th	Fr	Sa		Su	Mo	Tu	We	Th	Fr	Sa		Su	Mo	Tu	We	Th	Fr	Sa			
36	1	2	3	4	5	6	7	40	29	30	1	2	3	4	5	44	27	28	29	30	31	1	2	49	1	2	3	4	5	6	7			
37	8	9	10	11	12	13	14	41	6	7	8	9	10	11	12	45	3	4	5	6	7	8	9	50	8	9	10	11	12	13	14			
38	15	16	17	18	19	20	21	42	13	14	15	16	17	18	19	46	10	11	12	13	14	15	16	51	15	16	17	18	19	20	21			
39	22	23	24	25	26	27	28	43	20	21	22	23	24	25	26	47	17	18	19	20	21	22	23	52	22	23	24	25	26	27	28			
40	29	30	1	2	3	4	5	44	27	28	29	30	31	1	2	48	24	25	26	27	28	29	30	1	29	30	31	1	2	3	4			

Figure 2: Example of the date navigator displaying 12 months

[Top](#)

Usage

The date navigator is a control for advanced handling of all actions, which require a date input and to visualize a date. Thus, the main purpose of the date navigator control is to aid users in inputting a date. It also ensures that the date is entered in an appropriate format. In such cases,

it is highly recommended that users also be allowed the option to manually input the date as well.

Note: If the date must be entered in a particular format, an example should be given next to the entry field.

The date navigator can also be used to visualize the Western calendar.



[Top](#)

Design-relevant Attributes

The date navigator allows to set the number of months per row (**monthsPerRow**) and per column (**monthsPerColumn**).



[Top](#)

Related Controls

There are currently no related controls.



[Top](#)

More Info about Date Navigator

[Browser Compatibility](#) | [Editability in Style Editor](#) | [Accessibility – 508 Support](#)

Browser Compatibility

Netscape Navigator 4 cannot display certain visual aspects of the standard date navigator control.



Figure 1: Example of the date navigator in Netscape Navigator 4

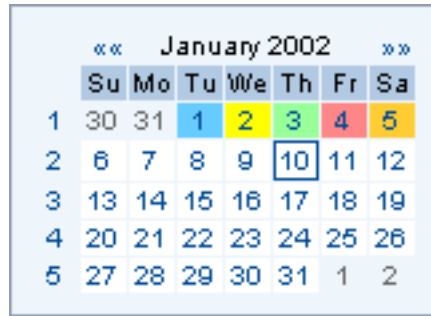


Figure 2: Example of the standard date navigator



[Top](#)

Editability in Style Editor

In the Style Editor for release 5.0 the date navigator is called "calendar." In the Style Editor, one can change the background colors, text attributes, padding and the type of cursor that appears over clickable elements. Here is a list of the styles that can be changed:

Group	Style	IE 5 and above	Netscape 4.7
Day Names	Background Color for Days of the Week	x	x
Day Numbers	Entry Width	x	
	Alignment of Entry Text	x	
	Decoration of Entry Text	x	
	Selection Background Color 1	x	x
	Selection Background Color 2	x	x

	Selection Background Color 3	x	x
	Selection Background Color 4	x	x
	Selection Background Color 5	x	x
Present Day	Background Color	x	x
	Border	x	
Other Days of this Month	Background Color	x	x
Days of Previous or Next Month	Inactive Font Color	x	
Container	Background Color of Container Body	x	x
	Container Border	x	
	Padding of Container Content	x	

Table 1: Editable styles for the date navigator control

For common styles see section [HTMLB Controls and Style Editor](#) in Customer *Branding and Style Editor*.



[Top](#)

Accessibility – 508 Support

Currently the date navigator has not been adapted to support screen readers. For more information about accessibility, see the *SAP Portals Accessibility Guidelines*.



[Top](#)

Control API for Date Navigator (dateNavigator)

A control for advanced handling of all actions which require a date input and to visualize a date.

- id**
 Identification name of the dateNavigator.
- model**
 Defines the model which provides the dateNavigator with data.
- monthPerColumn**
 The dateNavigator can display several month. The months are arranged in matrix form. This attribute defines the number of columns of the matrix.
- monthPerRow**
 The dateNavigator can display several month. The months are arranged in matrix form. This attribute defines the number of rows of the matrix.
- onNavigate**
 The navigation fields are located left and right of the displayed month «« November 2001 »». The << and >> fields can be used to select the previous and next month. If a dateNavigator has more columns the previous month navigator is located at the first column and the next month navigator at the last column. The 'onNavigate' attribute defines the action that will be processed when the user clicks on the navigation fields.
- onDayClick**
 Defines the action that will be processed when the user clicks on a day.
- onWeekClick**
 Defines the action that will be processed when the user clicks on a week. The week is the first column of the dateNavigator grid.
- onMonthClick**
 Defines the action that will be processed when the user clicks on the header text string representing the month displayed.

attribute	req.	values	default	case sens.	JSP taglib	classlib
id	yes	String	none	yes	id="VacationPlanner"	
model	yes	String	none	yes	model="myBean.model"	setModel ((DateNavigatorModel) model)
monthPerColumn	no	Numeric	1	-	monthPerColumn="3"	setMonthPerColumn(3)
monthPerRow	no	Numeric	1	-	monthPerRow="4"	setMonthPerRow(4)

events	req.	values	default	case sens.	JSP taglib	classlib
onNavigate	no	String	none	yes	onNavigate="ProcNav"	setOnNavigate("ProcNav")
onDayClick	no	String	none	yes	onDayClick="DaySel"	setOnDayClick("DaySel")
onWeekClick	no	String	none	yes	onWeekClick="WeekSel"	setOnWeekClick("WeekSel")
onMonthClick	no	String	none	yes	onMonthClick="MonSel"	setOnMonthClick("MonSel")

Example

```

<hbj:dateNavigator
  id="myDateNavigator1"
  model="myBean.model"
  monthsPerColumn="2"
  monthsPerRow="3"
  onNavigate="myOnNavigate"
  onDayClick="myOnDayClick"
  onWeekClick="myOnWeekClick"
  onMonthClick="myOnMonthClick"
/>

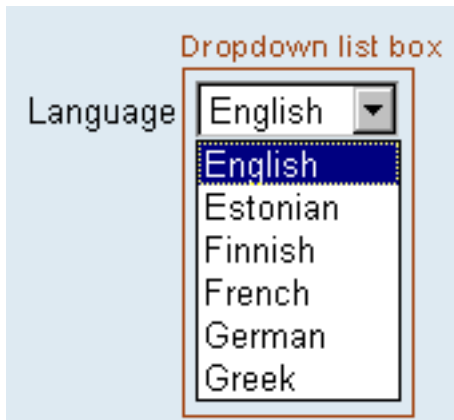
```

Result

« September 2001							Oktober 2001							November 2001 »»									
Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So			
35	27	28	29	30	31	1	2	40	1	2	3	4	5	6	7	44	29	30	31	1	2	3	4
36	3	4	5	6	7	8	9	41	8	9	10	11	12	13	14	45	5	6	7	8	9	10	11
37	10	11	12	13	14	15	16	42	15	16	17	18	19	20	21	46	12	13	14	15	16	17	18
38	17	18	19	20	21	22	23	43	22	23	24	25	26	27	28	47	19	20	21	22	23	24	25
39	24	25	26	27	28	29	30	44	29	30	31	1	2	3	4	48	26	27	28	29	30	1	2
Dezember 2001							Januar 2002							Februar 2002									
Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So	Mo	Di	Mi	Do	Fr	Sa	So			
48	26	27	28	29	30	1	2	1	31	1	2	3	4	5	6	5	28	29	30	31	1	2	3
49	3	4	5	6	7	8	9	2	7	8	9	10	11	12	13	6	4	5	6	7	8	9	10
50	10	11	12	13	14	15	16	3	14	15	16	17	18	19	20	7	11	12	13	14	15	16	17
51	17	18	19	20	21	22	23	4	21	22	23	24	25	26	27	8	18	19	20	21	22	23	24
52	24	25	26	27	28	29	30	5	28	29	30	31	1	2	3	9	25	26	27	28	1	2	3
1	31	1	2	3	4	5	6																

Dropdown List Box

[Usage](#) | [Design-relevant Attributes](#) | [Related Controls](#)



The dropdown list box is a field with an arrow icon on the right side. Clicking on this icon drops down a list immediately below the field and shows the user which values can be chosen. An entry in the list is called list box item. The dropdown list box is read-only.

Figure 1: A dropdown list box with six language items



Usage

Use dropdown list boxes:

- To support the **selection of a value** from a limited quantity. The number of items should not exceed 20.
- To **switch between views** on large amounts of data. Especially in iViews, this is a good method to save space. With dropdown list boxes, more views are possible than, for example, in a tabstrip, because the number of views is not limited by space. However, the list should not be longer than about 12 items.
- In a shuffler for **filtering** a larger data set, in order to get simplified and reduced views of the data. The shuffler mimics natural language statements for formulating the query, but can also be used with query statements consisting of words only. The query statement is typically assembled by combining static texts with dynamic elements like dropdown lists, edit fields and selection elements.

Note: The dropdown list box control does not render a descriptive label automatically. Use the [label](#) control to add a description.

The following table shows examples for the usage described above:

Street Address *

City *

Country

Email address *

Selection of a value

Contents of documents sorted by

[Name](#)
[Modified](#)
[Size](#)

documents
[Details](#) | [New Folder](#) | [New File](#) | [New Text File](#) | [New Link](#)

Name	Modified	Size
discussions	1/7/02 4:08:06 PM	Details Delete
faq	1/7/02 4:07:19 PM	Details Delete
Links	1/7/02 4:04:27 PM	Details Delete
news	1/7/02 4:13:21 PM	Details Delete
Public Documents	1/15/02 10:14:17 AM	Details Delete

View selection

Search in

Groups		
	ID	Descript
	administrators	Group th
	Developers	
	EngServices	

Dropdown list box used in a shuffler

Table 1: Usage examples for the dropdown list box

Choosing the Appropriate Selection Control

A dropdown list box is similar in function to a list box - both offer a list of items where users can select one item from, that is, both are single-selection lists.

See [Forms - Using Different List Types](#) for guidelines on choosing the appropriate selection control.

Note: For very small item numbers (2-6) and if the users should immediately see all alternatives, use [radio buttons](#) for single-

selection choices.



Design-relevant Attributes

The dropdown list box can be set to an **enabled** or **disabled** state. Set attribute **disabled** to FALSE to enable a checkbox, set **disabled** to TRUE to disable it. To see an example of an enabled dropdown list box, [click here](#).

A disabled dropdown list box is not clickable, no item is selectable.



Figure 2: Disabled dropdown list box

The dropdown list box does not have a **width** attribute. Note, that this control takes the width from the widest list box item.

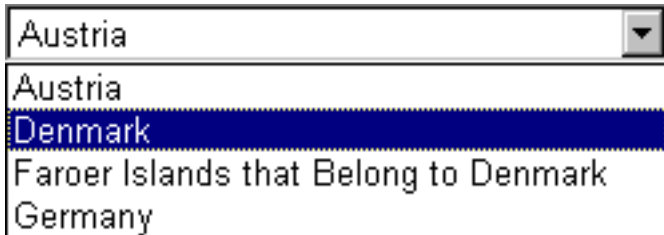


Figure 3: Dropdown list box with a very wide item

Usage - Disabled State

Set the disabled state if the user is not allowed to change the value of a dropdown list box or if a larger group of input elements including a dropdown list box is disabled.

Example: A set of fields including a dropdown list box is disabled because the user unchecked an option (see figure 4).

First name *

Last name *

City

Street

Payment method Invoice Credit Card

Card

Number

Expired

Figure 4: Disabled dropdown list box - the fields are disabled because the user checked the Invoice option



Related Controls

[Input Field](#), [Item List](#), [Label](#), [List Box](#), [Radio Button](#), [Tree View](#)



More Info about Dropdown List Box

[Browser Compatibility](#) | [Editability in Style Editor](#) | [Accessibility – 508 Support](#)

Browser Compatibility

Netscape 4.7

The disabled version of the dropdown list box is not available for Netscape 4.7.

Netscape 6.1 and 6.2

In Netscape Version 6.1 and 6.2, the dropdown list box looks slightly different than the standard control.

To see examples of standard dropdown list boxes [click here](#).

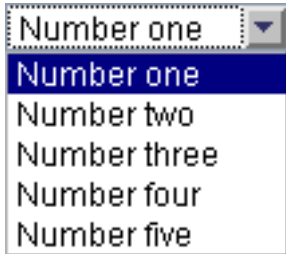


Figure 1: Netscape 6.1/6.2 example of an enabled dropdown list box



Figure 2: Netscape 6.1/6.2 example of a disabled dropdown list box



[Top](#)

Editability in Style Editor

In the Style Editor, the dropdown list box does not appear in the list of customizable elements directly. No control-specific styles exist for this element, only common styles are used.



[Top](#)

Accessibility – 508 Support

Dropdown list boxes have to be used in combination with the label element which points to the assigned listbox. This ensures, that

screenreaders are aware of the relationship between the both elements and can read the correct label to the according dropdown list box.

- **Keyboard:** The dropdown list box inserted into the accessibility hierarchy by default.
- **Default Description:** Is provided by the HTMLB rendering engine.
- **Application-specific Description:** Set an additional description using the *setTooltip* method if needed.
- **Label:** Has to be connected to a label control (use method *setLabelFor* for identifying the corresponding dropdown list box).



[Top](#)

Control API for Dropdown List Box (dropdownListBox)

A control with a dropdown arrow that the user clicks to display a list of options. An item in the dropdownListBox is called listBoxItem.

- disabled**
 A boolean value that defines if the dropdownListBox is clickable. If the dropdownListBox is disabled it is not selectable. A disabled dropdownListBox has a different color for the displayed listBoxItem.
- id**
 Identification name of the dropdownListBox.
- model**
 Defines the model which provides the dropdownListBox with data.
- nameOfKeyColumn**
 Specifies the name of the column that contains the key. This is used when you use an underlying table in the model.
- nameOfValueColumn**
 Specifies the name of the column that contains the visible text. This is used when you use an underlying table in the model.
- onSelect**
 Defines the action that will be processed when the user clicks on the enabled dropdownListbox. If you do not define a onClick event the dropdownListbox can be clicked but no event is generated.

- onClientSelect**
 Defines the JavaScript fragment that is executed when the user clicks on the dropdownListbox. If both events ('onClick' and 'onClientSelect') are specified, the 'onClientSelect' event is activated first. By default the 'onClick' event is activated afterwards. In the JavaScript fragment you can cancel the activation of the 'onClick' event with the command

```
htmlbevent.cancelSubmit=true;
```

The 'onClientSelect' event is very useful to save client/server interaction.

Example

A dropdownListbox click usually activates the client/server interaction. If an input field has to be filled out for further processing, the JavaScript fragment can check the necessary input on the client side and display a message if the necessary input is missing, without server interaction.

Note

To use JavaScript the JSP has to use the **page** tag (see [page](#) tag).

- selection**
 Specifies the key of the listBoxItem which is displayed in the dropdownListBox.
- tooltip**
 Defines the hint of the dropdownListBox which is displayed as the mouse cursor passes over the dropdownListBox, or as the mouse button is pressed but not released.
- width**
 Defines the width of the dropdownListBox in pixel or percent.

attribute	req.	values	default	case sens.	JSP taglib	classlib
disabled	no	TRUE FALSE	FALSE	yes	disabled="True"	setDisabled(true)
id	yes	String	none	yes	id="listbox_te"	
model	no	String	none	yes	model="myBean.model"	setModel((IListModel) model)
nameOfKeyColumn	no	String	none	no	nameOfKeyColumn("k1")	setNameOfKeyColumn("k1")
nameOfValueColumn	no	String	none	no	nameOfValueColumn("v1")	setNameOfValueColumn("v1")
selection	no	String	none	yes	selection("HD")	setSelection("HD")
tooltip	no	String	none	no	tooltip="select a item"	setTooltip("select a item")
width	no	Unit	150	-	width="200"	setWidth("200")

events	req.	values	default	case sens.	JSP taglib	classlib
onClientSelect	no	String	none	yes	onClientSelect="JavaScript"	setOnClientSelect("JavaScript")
onSelect	no	String	none	yes	onSelect="proc_listbox"	setOnSelect("proc_listbox")

listBoxItem

Defines the items in a dropdownListBox or listBox instead of the model.

- key**

A string which is passed on to the event handling routine when the event occurs. A key string has be defined and must not be empty. Each listBoxItem must have an unique key.

- selected**

A boolean value.
selected="false": no effect

selected:="true"

dropdownListBox:

The item is displayed in the dropdownListBox. It overrules the "selection" attribute of the dropdownListBox. If several listBoxItems are selected the last defined listBoxItem is displayed in the dropdownListBox.

listBox:

selected="true": The item is displayed as selected in the listBox.

- value**

Defines the text string displayed in the dropdownListBox or listBox. A 'value' string has be defined and must not be empty.

attribute	req.	values	default	case sens.	JSP taglib	classlib
key	yes	string	none	yes	key="WD"	addItem("WD","Walldorf")
selected	no	FALSE TRUE	FALSE	no	selected="true"	addItem("WD","Walldorf") addSelection("WD")
value	yes	String	none	no	value="Walldorf"	see "key"

Example

```
<hbj:dropdownListBox
  id="DDCitiesNearby"
  tooltip="Cities surrounding SAP"
  selection="WD"
  disabled="false"
  nameOfKeyColumn="KeyCol"
  nameOfValueColumn="KeyVal"
  onSelect="ProcessCity"
  onClientSelect="PreprocessCity"
  >

  <hbj:listBoxItem key="HD" value="Heidelberg" />
  <hbj:listBoxItem key="HK" value="Hockenheim" />
  <hbj:listBoxItem key="WD" value="Walldorf" />
  <hbj:listBoxItem key="WL" value="Wiesloch" />

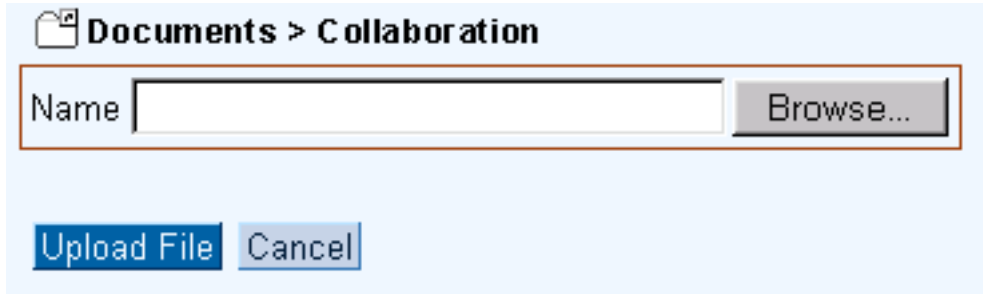
</hbj:dropdownListBox>
```

Result



File Upload

[Usage](#) | [Related Controls](#)



File upload is a control that allows to access files on the client for uploading them to the server

Figure 1: Example of a file upload control in a dialog window

 [Top](#)

Usage

Use the file upload control in case you want to provide the capability to pass files to the server.

 [Top](#)

Related Controls

[Breadcrumb](#), [Button](#)

 [Top](#)

More Info for File Upload

[Browser Compatibility](#) | [Editability in Style Editor](#) | [Accessibility – 508 Support](#)

Browser Compatibility

File upload is a very security sensitive control, since it allows to access the client's harddisk. For this reason, the original browser control with no design modification is used. The *Browse* button appears as a platform-specific standard button.



[Top](#)

Editability in Style Editor

No editability



[Top](#)

Accessibility – 508 Support

To be provided



[Top](#)

Control API for File Upload (fileUpload)

A control to select a file for upload. The control generates a input field and a "Browse" button. The "Browse" button activates the file browser and allows to select the file interactively. To use the selected file in the fileUpload control you need another control, e.g. a button named "Start upload", to finally start the upload.

Important note:

If you use a fileUpload control in the JSP you must set the encodingType attribute of the form control to "multipart/form-data".

Example: `<hbj:form encodingType="multipart/form-data" >`

- **id**
Identification name of the fileUpload.
- **accept**
Defines the accepted MIME type.
- **maxLength**
Defines the maximum file size in byte allowed for the upload. By default there is no limit.
- **size**
Defines the width of the fileUpload inputField in characters. The frame of the inputField is adjusted accordingly considering the actual text font and the design attribute.

attribute	req.	values	default	case sens.	JSP taglib	classlib
id	yes	String	none	yes	id="chooseInputFile"	
accept	no	String	none	-	accept="text/rtf"	setAccept("text/rtf")
maxLength	no	Numeric	-1	-	maxLength="125000"	setMaxlength(125000)
size	no	Numeric	20	-	size="30"	setSize(30)

Example

```
<hbj:form encodingType="multipart/form-data" >
  <hbj:fileUpload id="myfileupload"
    maxLength="125000"
    size="50"
  />
</hbj:form>
```

Result



Programming Tip

In an application you usually have an additional control, usually a button, to start the upload once the file has been selected with the "Browse..." button.

Here we show what the server programm has to do when the user starts the upload. In our example we define a button with an "onClick" event and specified as "onClick" event handling method "onLoadFile". The "onLoadFile" method does the upload handling.

Example

```
public void onLoadFile(Event event) {
    FileUpload fu = (FileUpload) this.getComponentByName("myfileupload");

// this is the temporary file
    if (fu != null) {
// Output to the console to see size and UI.
        System.out.println(fu.getSize());
        System.out.println(fu.getUI());
// Get file parameters and write it to the console
        IFileParam fileParam = fu.getFile();
        System.out.println(fileParam);
// Get the temporary file name
        File f = fileParam.getFile();
        String fileName = fileParam.getFileName();

// Get the selected file name and write it to the console
        ivSelectedFileName = fu.getFile().getSelectedFileName();
        System.out.println("selected filename: "+ivSelectedFileName);
    }
}
```


Group

[Usage](#) | [Types](#) | [Design-relevant Attributes](#) | [Related Controls](#)

Data Sources - General Settings

ID

Name

Description

URL

Data source is an SAP Portals Unifier project

Mapping Administration

Notification: You must restart the IIS in order for your changes to take effect

Authentication method

Authorization level

Teach URL

Always teach

User Mapping

Update Delete

A group control clusters a set of controls or information: it demonstrates which parts belong together, and separates them from other parts of content. Figure 1 (to the left)

The primary and secondary group types, consisting only of a background color, can be used to highlight a part of the content. Figure 2 (below)

In full-page applications, the primary and secondary group types may be used to create an area into which other controls can be placed. Figure 3 (bottom)

Figure 1: Example of a group used in an iView

Corporate News Group

Latest News:
2/5/2004
Due to unexpected oil findings on company grounds in Alaska, SAP announces extremely high revenue growth. Company to enter new business field.

General Availability of SAP SEM 3.0A
3/28/2001
The SAP SEM function of mySAP Financials has been significantly enhanced for 3.0A, specifically in the areas of business planning and simulation and the Corporate Performance Monitor.

Europe to Leapfrog U.S. in the E-Business Arena
idg.net, 3/28/2001
Less than a year ago, the skepticism of European businesses towards electronic business was much criticized. Now, with the dot-com crash in the U.S., Europeans will be able to benefit from their conservatism, according to market researcher Gartner Group "Europe is not suffering from the U.S. dot-com hangover. U.S. companies have gone the wrong road and need to back out. That will take them nine months to a year," said Alexander Drobik, vice president at Gartner EMEA (Europe, Middle East, and Africa) in interview here at the Gartner Europe Spring Symposium/ITxpo 2001.

Figure 2: Example of a group used to highlight a part of the content

Data Sources - General Settings Group used as an Area

ID Description

Name URL

Data source is an SAP Portals Unifier project

Mapping Administration Group

Notification: You must restart the IIS in order for your changes to take effect

Authentication method

Authorization level

Teach URL

Always teach

[User Mapping](#)

[Update](#) [Delete](#)

Figure 3: Example of groups used in a full-page application

Usage

In full-page applications, use the group control to

- Group each coherent set of fields or information and separate one set from another
- Define an area where text or controls can be placed
- Highlight a certain part of the application or information

Within an iView, there should normally be no need to highlight or separate different groups, since an iView is per definition small and simple. However, there are certain cases where it makes sense to use a group control to

- Highlight or separate a certain part of an iView to better demonstrate it's structure.
For example, to show whether a certain button relates to the whole application or only to a part of it
- Highlight a certain portion of textual information within a large body of text.

General Usage Tips

Use groups only if other ways of separating information or field groups do not suffice. Group boxes look similar to the tray container of iViews and may clutter the interface visually. Preferably, use white space or vertical dividing lines to group elements, relying on the Gestalt laws.

Tip: Sometimes you don't really need groups in your iView, but simply want to create a better visual structure. Instead of misusing the group control, use the text view control to give users a better overview of your content: Create a text label for each part and add a blank line between parts to separate them.

Try not to nest groups; separate subgroups within groups by lines or white space. If you need to nest groups, consider nesting different group types (see [below](#)).

Positioning

Tabstrips, table views and tree views are **only** allowed to be included in a group control if they appear together with other elements (see [Layout Hierarchy](#) for details). Placed as a single element, these controls do not need any further separation from their surroundings, as they already have distinct borders and a dominant shape.



[Top](#)

Types

Depending on the items the group will contain, you can choose one of the offered styles. Currently, there are five group designs available, which are set using the attribute **design**.

Primary and Secondary Groups

The **primary** (**design** = PRIMARYCOLOR) and **secondary groups** (**design** = SECONDARYCOLOR) allow to visualize groups through a simple, flat background color. Both are suitable for textual content. Controls with a white background color, such as input fields and checkboxes, stand out well on both group designs.

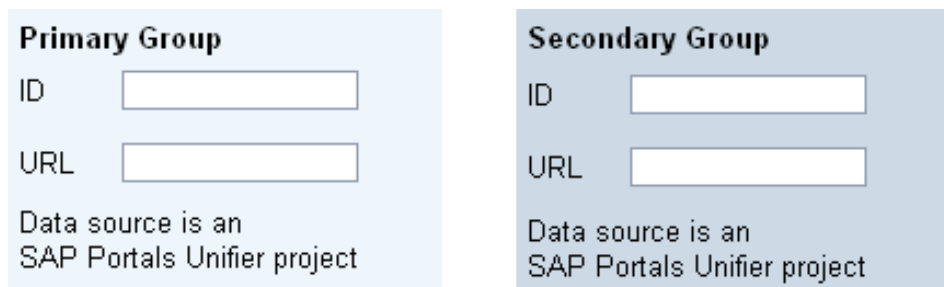


Figure 1: Primary (left) and secondary (right) groups

Use primary and secondary groups (figure 1) to

- Define an area in a full-page application

Group

Note: It is recommended to use the secondary (darker) group as an area background. You may then place the primary group on top of it to highlight or group parts of the area's content.

- Highlight a part of the content
- Group a set of coherent elements

Group Box

The **group box** design (**design** = SAPCOLOR) has a transparent body background. Because of its border and header bar, it has a quite dominant appearance.

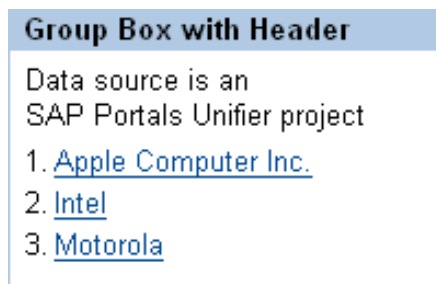


Figure 2: Group box

Use the group box (figure 2) to group a set of coherent elements.

Do not nest any further groups inside a group box.

Avoid putting more than one or two of this group type adjacent to one another. They create a grid-like visual effect, which makes it hard for users to determine, which border belongs to which group.

Header Group 1

Header group 1 (figure 3, left - **design** = SECONDARYBOXCOLOR) is best suited for forms; its light body background color lets white input fields stand out well.

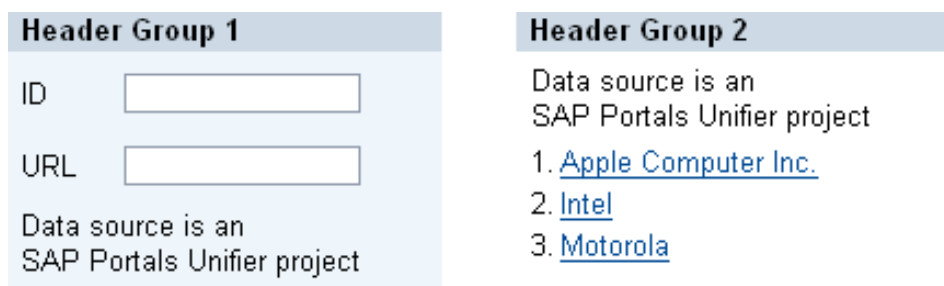


Figure 3: Header group 1 (left) and header group 2 (right)

Header Group 2

Header group 2 (figure 3, right - **design** = SECONDARYBOX) has a white body background color, which makes it unsuitable for forms. Textual content and lists work best with this group style.



Design-Relevant Attributes

The look of groups can be determined by three attributes: **design** selects the group type (values PRIMARYCOLOR, SAPCOLOR, SECONDARYBOX, SECONDARYBOXCOLOR, SECONDARYCOLOR), **width** sets the **width** of the group, and **title** sets the title text.

For details refer to page [Control API for Group](#).

Group

 [Top](#)

Related Controls

[Tabstrip](#), [Table View](#), [Tree View](#), [Text View](#) (for headers of subgroups)

 [Top](#)

More Info about Group

[Browser Compatibility](#) | [Editability in Style Editor](#) | [Accessibility – 508 Support](#)

Browser Compatibility

Netscape 4.7 doesn't render the padding correctly. Title and body text begin immediately at the left edge of a group. Exception: Body text padding in the group box with header is correct.

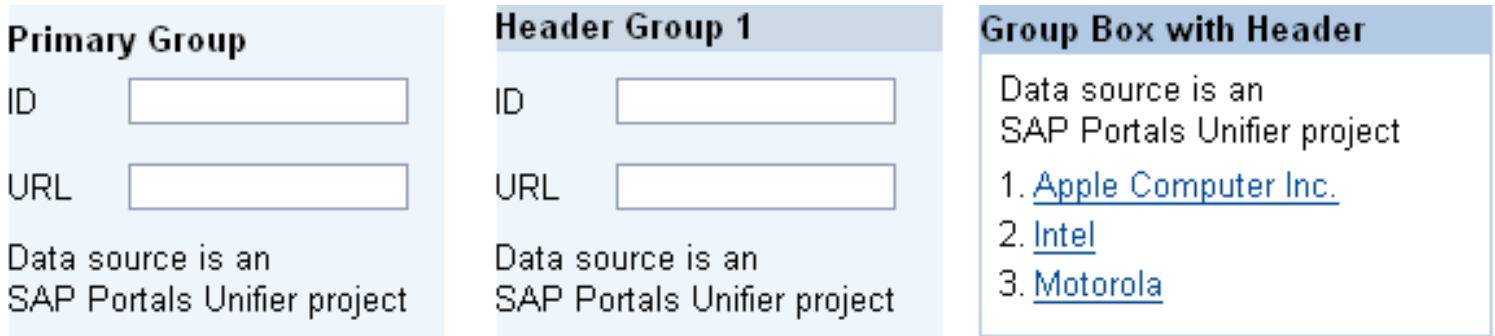


Figure 1: Examples of how groups look in some Netscape versions.



[Top](#)

Editability in Style Editor

In the Style Editor, it is possible to modify the following attributes of the group control:

Group	Style	IE 5 and above	Netscape 4.7
Fonts	Font Weight of Title	x	x
Background Colors	First Background Color	x	x
	Second Background Color	x	x
	Third Background Color	x	x
Borders	Border Width, Style and Color	x	x
Layout	Title Padding for Groups with Header Strip	x	
	Body Padding	x	
	Title Padding for Groups without Header Strip	x	

	Padding of Body Content	x	
Background	Height of Container Title	x	
	Background Color of Container Title	x	x
	Background Color of Container Body	x	x

Table 1: Editable styles for the group control

For common styles see section [HTMLB Controls and Style Editor](#) in Customer *Branding and Style Editor*.



[Top](#)

Accessibility – 508 Support

- **Keyboard:** The group is **not** inserted into the accessibility hierarchy by default. Elements inside a group have to be handled separately, depending on the respective controls included in the group.
- **Default Description:** Is provided by the HTMLB rendering engine.
- **Application-specific Description:** Set an additional description using the *setTooltip* method if needed. Elements inside a group have to be handled separately, depending on the controls used.

Users with low vision can use the portal personalization link to select the portal's **high contrast** design, which was developed to offer maximum contrast. It renders the screen without using many colors or shades, which leaves us with little more than black and white. Though the results may look highly unaesthetic to people with normal vision, sharp contrast is what users with low vision need, in order to be able to read text at all.

As a developer, you needn't do anything to enable the high contrast scheme, but you should have a feeling for what happens to your application when it is viewed in high contrast.

This is how groups look in high contrast. (Figure 2, left side). On the right hand side you can see some possible variations of group designs, which can be achieved using the Style Editor.

Primary Group

Standard Text

High Contrast
Default Design

Primary Group

Standard Text

Possible
Variations

Secondary Group

Standard Text

Secondary Group

Standard Text

Group Box with Header

Standard Text

Group Box with Header

Standard Text

Header Group 1

Standard Text

Header Group 1

Standard Text

Header Group 2

Standard Text

Header Group 2

Standard Text

Figure 2: Examples of groups in high contrast (left) and possible Style Editor variations (right).



[Top](#)

Control API for Group (group)

A framed panel to visually group controls. See also 'tray'.

- **design**

The design of the group that refers to CSS. You can select

- PRIMARYCOLOR
The panel of the group is filled with the same background as the title bar (primary color).
- SAPCOLOR
The title bar and the frame around the panel is in SAP blue and the panel has a white background.
- SECONDARYBOX
No frame around the panel. Title bar with background color (primary color), panel has a white background.
- SECONDARYBOXCOLOR
The panel is filled with a background color that is different from the title background color (primary color). No frame around the panel.
- SECONDARYCOLOR
The panel is filled with a background color that is same as the title background color (secondary color).

- **id**

Identification name of the group.

- **title**

Defines the string of text placed left aligned in the title bar. If no title should be displayed an empty string (null) can be used. The width of the group is automatically adjusted to the length of the text when the 'width' attribute is set smaller than the title text width.

- **tooltip**

Defines the hint of the group which is displayed as the mouse cursor passes over the group, or as the mouse button is pressed but not released.

- **width**

Defines the width of the group. The width of the group is automatically adjusted to the length of the 'title'. To see an effect of the 'width' attribute 'width' has to be set higher as the width defined through the length of the 'title' string. If an empty (null) 'title' string is set no 'title' attribute is defined the width of the group is set according to the 'width' attribute.

attribute	req.	values	default	case sens.	JSP taglib	classlib
design	yes	PRIMARYCOLOR SAPCOLOR SECONDARYBOX SECONDARYBOXCOLOR SECONDARYCOLOR	none	yes	design="SAPCOLOR"	setDesign(GroupDesign.SAPCOLOR)
id	yes	String	none	yes	id="Intro_Text"	
title	no	String	none	no	title="Headlines"	setTitle("Headlines")
tooltip	no	String	none	no	tooltip="latest news"	setTooltip("latest news")
width	no	Unit	50%	-	width="300"	setWidth("300")

groupBody

Defines the items in the group. A group can be filled with any control (checkbox, image, textView etc.).

Example

```
<hbj:group
  id="HeadlineNewsGroup"
  design="PRIMARYCOLOR"
  title="latest headlines"
  tooltip="all the news you need"
  width="50%"
  >

  <hbj:groupBody>
    <hbj:textView
      encode="false"
      text="The NASDAQ on an upswing<br>Good news for homeowners"
    />
  </hbj:groupBody>

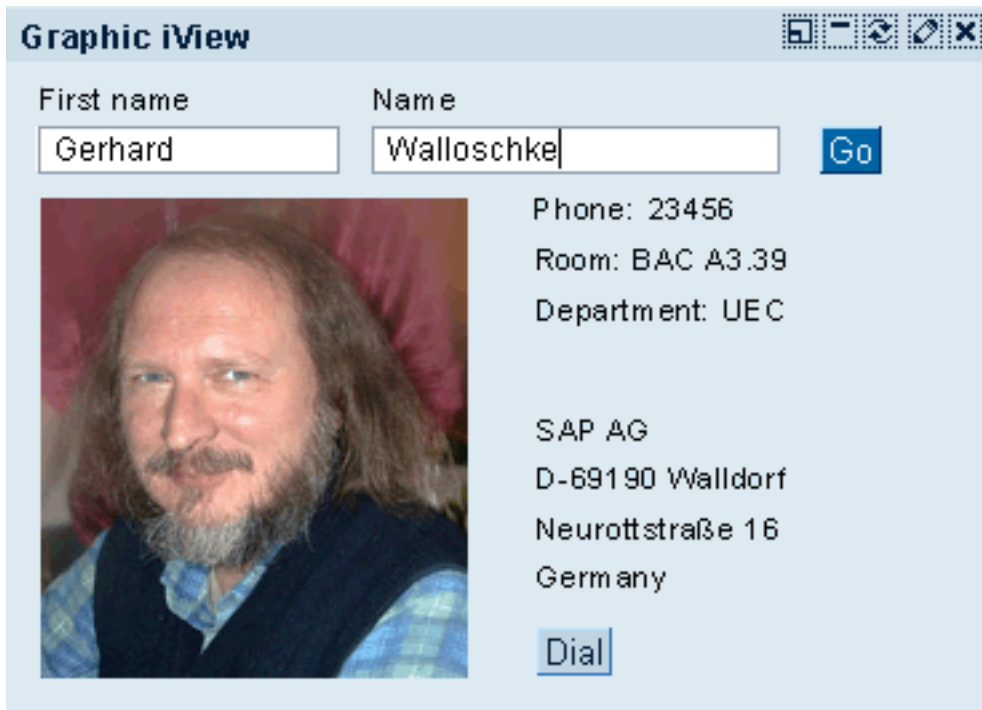
</hbj:group>
```

Result**latest headlines**

The NASDAQ on an upswing
Good news for homeowners

Image

[Usage](#) | [Types](#) | [Design-relevant Attributes](#) | [Related Controls](#)



The image control displays a bitmap GIF or JPEG format. The width and height of the image can be specified.

Figure 1: Example of an image in an iView. The image appears only after the user makes a selection via the shuffler placed above the image.

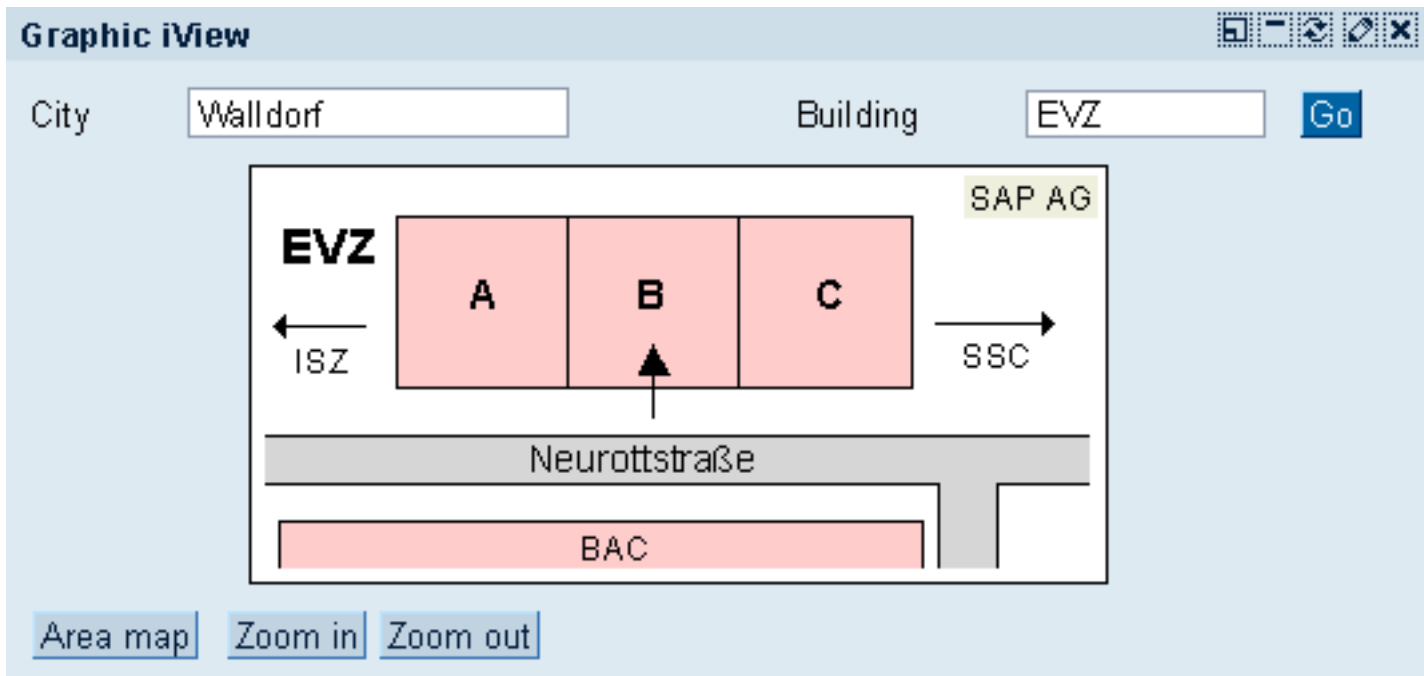


Figure 2: Example of an image with a row of image-related function buttons



Usage

The image control displays a bitmap in GIF or JPEG format. The width and height of the image can be specified.

Photographs, graphics, charts and diagrams, maps, sketches, animated graphics and video (movies) may be used in the portal. If used properly, they carry great amounts of information, which would take up much more time and screen-space, if they were explained in words.

Note: Although icons are also images, they are not allowed in iViews, except for displaying status information. That is, there are no function icons on buttons or tabs. For more information on **status icons** in the portal, see the *SAP Reference Lists* in the *SAP Design Guild*.

Interacting with Images

- If graphics and data can be selected from several sets, or if the amount of data has to be reduced, place a filter or shuffler *above* the image. (See figure 1, above)
- Place buttons for image-related functionality and status information (e.g. zoom factor) below the image and left align them. (See figure 2, above)
- Place buttons related to the whole iView in the lower left corner; these buttons may reside in the same row as the table-related buttons.
- If there is an emphasized button, it is the leftmost button of the respective button group (image-related or iView-related). There must not be more than one emphasized button in an iView.

Legend

Always provide an appropriate legend. Place legends or other text below the image or to its right, depending on the format of the image and the iView or page layout.

Tips for Using Images

- Align graphics so that their main contents points towards the text, not away from it.
- Crop graphics to the relevant section; make them as small as possible and avoid irrelevant and distracting elements.
Example: Do not show a US map if you want to illustrate data in Michigan - use a Michigan map instead.
- Use high quality graphics.
Example: Do not draw graphics by yourself, involve graphic designers.
- Care for the correct format of images:
JPEG for photos and images with many colors and gradations.
GIF for images with flat-colored areas and bold lines, like diagrams or cartoon, and images with (less than) 256 colors. Typically, screen dumps work better in GIF format.
Sharp-edged graphics work well as transparent GIF, on any background. Using transparent GIF format for round, smooth forms and large-sized text, as used in many logos, may cause problems if you don't know what color their background will be.



Types

Charts

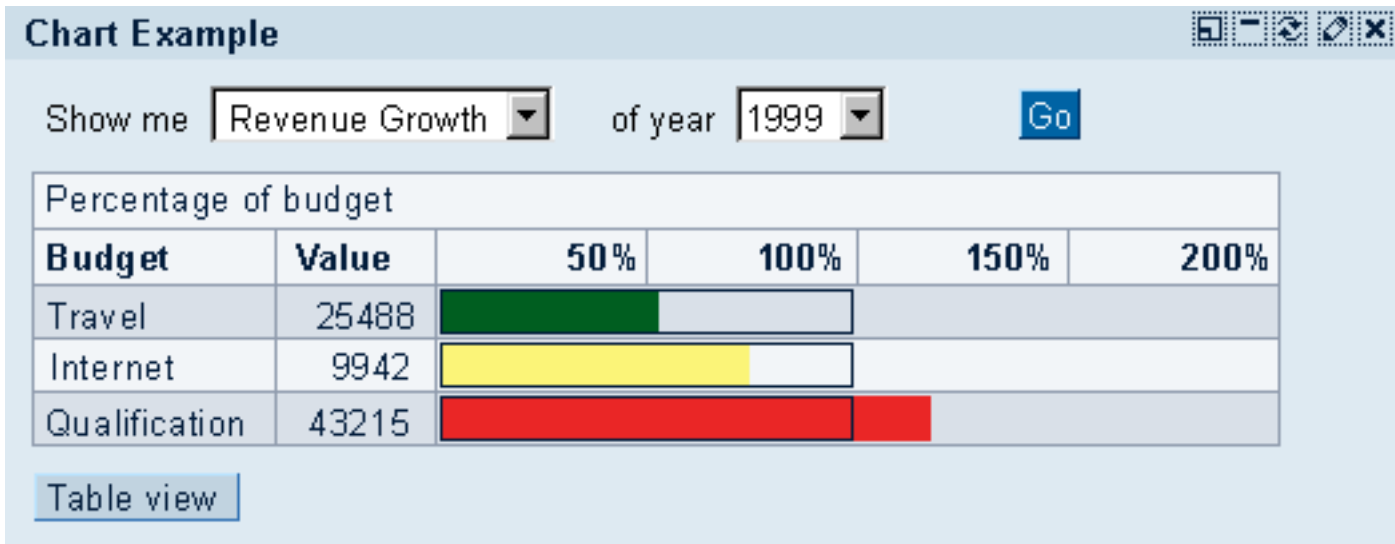


Figure 3: A horizontal bar chart table; for exact comparisons, the values have been added to the chart.

See [Chart](#) for details and *Recommendations for Charts and Graphics* in the *SAP Design Guild* for thorough information on the usage of charts.

Animated Graphics, Video, or Movie



Figure 4: Animation can explain procedures or be just fun to watch...

Note: The image control currently supports only animated GIFs.

Sketches

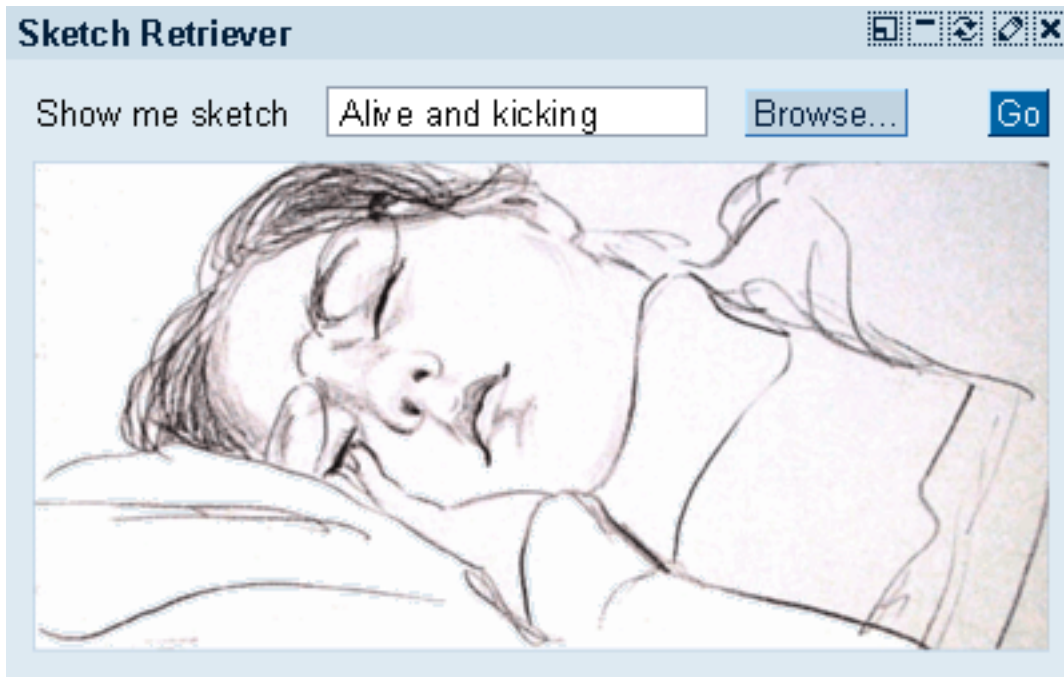


Figure 5: A sketch may be effective for fast communication or serve as a preliminary version of a diagram



[Top](#)

Design-relevant Attributes

You can set the height (**height**) and width (**width**) of an image, also the tooltip text (**tooltip**), which is displayed as the mouse cursor passes over the image, or as the mouse button is pressed but not released.

For details see page [Control API for Image](#).

Usage - Height and Width

Note: Do not scale images in the browser by changing the values for height and width. This results in poor image quality.



[Top](#)

Related Controls

[Table View](#)



[Top](#)

More Info about Image

[Browser Compatibility](#) | [Editability in Style Editor](#) | [Accessibility – 508 Support](#)

Browser Compatibility

No known issues



Editability in Style Editor

Customers can customize portal images used within the portal's outer frame and control-rendering (iView function images, table buttons, etc.) quite easily via Style Editor. The tool offers no editable styles related to images placed as portal content.



Accessibility – 508 Support

- **Keyboard:** Images are **not** inserted into the accessibility hierarchy by default.
- **Default Description:** Is provided by the HTMLB rendering engine
- **Application-specific Description:** Set an additional description using the *setTooltip* method if needed. Do **not** use the *setAlt* method that sets the alternate text (*alt* attribute).



Control API for Image (image)

Displays a bitmap in GIF or JPEG format. The width and height of the image can be specified.

- **alt**
Defines an alternative text for the 'src' attribute. If the 'src' bitmap cannot be found or opened (e.g. unrecognized graphic format) the 'alt' text is displayed with a red X in front of it and surrounded by a frame indicating the bitmap size.
- **height**
Defines the height of the bitmap. If 'height' is omitted the height of the image is determined by the bitmap itself.
- **id**
Identification name of the image.
- **src**
Name of bitmap (=source). The name of the bitmap is case sensitive. The images are usually stored in the public resource path of the component in the subfolder `/images`. The public resource path can be determined with the JSP command:

```
<% String PublicURL = componentRequest.getPublicResourcePath(); %>
You can also get the complete path to image include the subfolder /image with the command:
<% String ImageURL = componentRequest.getPublicResourcePath() + "/images/"; %>
```

Hint: We use the string variable ImageURL in following table to demonstrate the setting of the src attribute.

Another possibility is to get the complete URL of the image and set the src attribute in a scriptlet.

```
<%
IResource rs = componentRequest.getResource(IResource.IMAGE, "images/mypicture.gif");
image.setSrc(rs.getResourceInformation().getURL(componentRequest));
%>
```

- **tooltip**
Defines the hint of the image which is displayed as the mouse cursor passes over the image, or as the mouse button is pressed but not released.
- **width**
Defines the width of the bitmap. If 'width' is omitted the width of the image is determined by the bitmap itself.

attribute	req.	values	default	case sens.	JSP taglib	classlib
alt	yes	String	none	no	alt="picture not found"	setAlt("picture not found")
height	no	Unit	bitmap height	no	height="150"	setHeight("150")
id	yes	String	none	yes	id="Hometown"	
src	yes	String	none	yes	src="<%= ImageURL + "walldorf.jpg" %>"	setSrc(ImageUri + "walldorf.jpg")
tooltip	no	String	none	no	tooltip="center of ebiz"	setTooltip("center of ebiz")
width	no	Unit	bitmap width	-	width="300"	setWidth("300")

Example

```
<hbj:image
  id="Logo"
  tooltip="center of ebiz"
  width="70"
  height="35"
  src="<%= ImageURL +\"saplogo.gif\" %>"
  alt="picture saplogo.gif not found"

/>
```

Result



Input Field

[Usage](#) | [Types](#) | [Design-related Attributes](#) | [Related Controls](#)

Input fields are used for entering and displaying data in forms. The data can be of various types, such as Date, Integer, or String.

Input fields can have different behaviors, such as password, read-only, or required, and different states, such as normal and error.

Figure 1: Example of grouped input fields with and without required inputs



[Top](#)

Usage

In most cases, input fields appear in combination with the label control and sometimes with additional elements, such as descriptions or buttons.

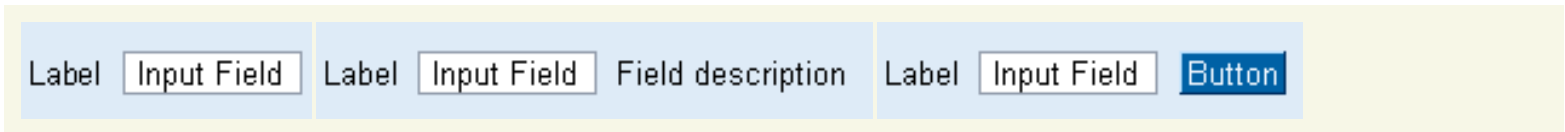


Figure 2: Input field with label (left) and additional elements

Typically, the label is placed left to the input field, while the description follows the field. There is one exception to this rule: You may use small labels if you place the labels above input fields to achieve a more compact design (figure 3).

Figure 3: Use small labels for labels that are placed above their associated input fields

In addition, help texts can be placed right to the field below it.

Help Texts

Help texts are special descriptions that are placed behind the input field or - if space is limited - below the input field and left-aligned with it (figure

4). Do **not** use the [label](#) control for help texts. Use the [text view control](#), instead. Set the help text size to small (style **Legend**).

Occupation

Example: Developer

Figure 4: A small label used in conjunction with a small help text below the associated input fields - do not use the label control for the help text (below the input field)

Width and Alignment

Often input fields are grouped to form a semantic block of input data, such as address data, or bank data. In that case, input fields should indicate how many characters the user has to enter. Therefore, it is not appropriate to set all fields within a group to the same size. That is important because input fields are often used in combination with other input types, such as checkboxes and radio buttons.

First name *

Last name *

Gender

City

Street

List my mail address for free

Figure 5: Example of grouped input fields with the width attribute set to appropriate values

Use the [grid layout](#) control for aligning fields and labels. Both fields and labels are left-aligned within the grid. The number of characters of the longest label determines the offset between the labels and the input fields.

For the usage of the different input field types and sizes, see [below](#).



Top

Types

Input fields come in two sizes: standard size and small size. They are set using the attribute **size** to STANDARD (default) or SMALL.

Usage - Sizes

Usually, only the standard size is used. If the screen real estate is limited, the small size might be appropriate. Do **not** mix small and standard input fields within **one** field group.



Figure 6: Standard input field (left) and small input field (right)



[Top](#)

Design-relevant Attributes

A number of attributes allow to display several different states and behaviors of an input field, such as read-only, error, password field and required field.

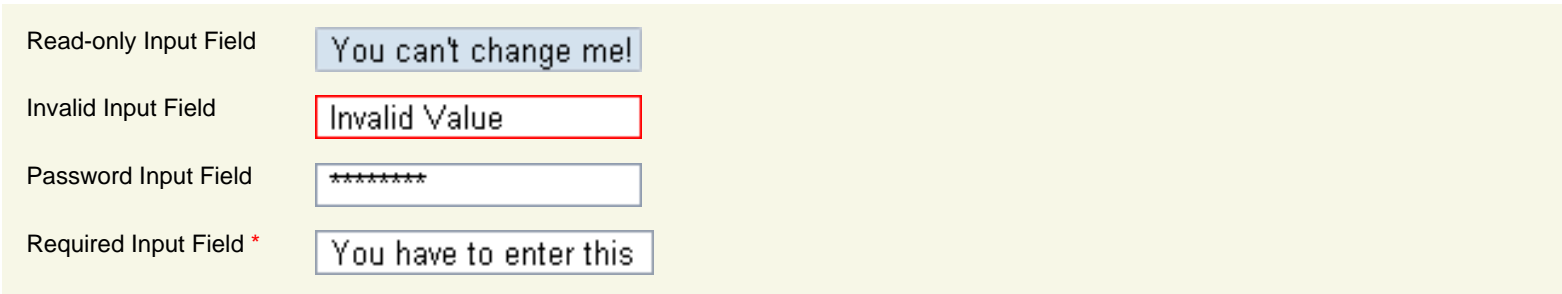


Figure 7: Different states of input fields

These states and behaviors are set by assigning the value TRUE to the Boolean attributes **disabled**, **invalid**, **password**, and **required**.

The data type of input fields, such as Integer, String, etc., is set using the attribute **type**. For possible values and further attributes, see page [Control API for Input Field](#).

In addition, there are specialized input fields, which may also be accompanied by a value help. Below, we present the date field as example (this is currently the only type of value help that is supported - see page [Control API for Input Field](#) for details).

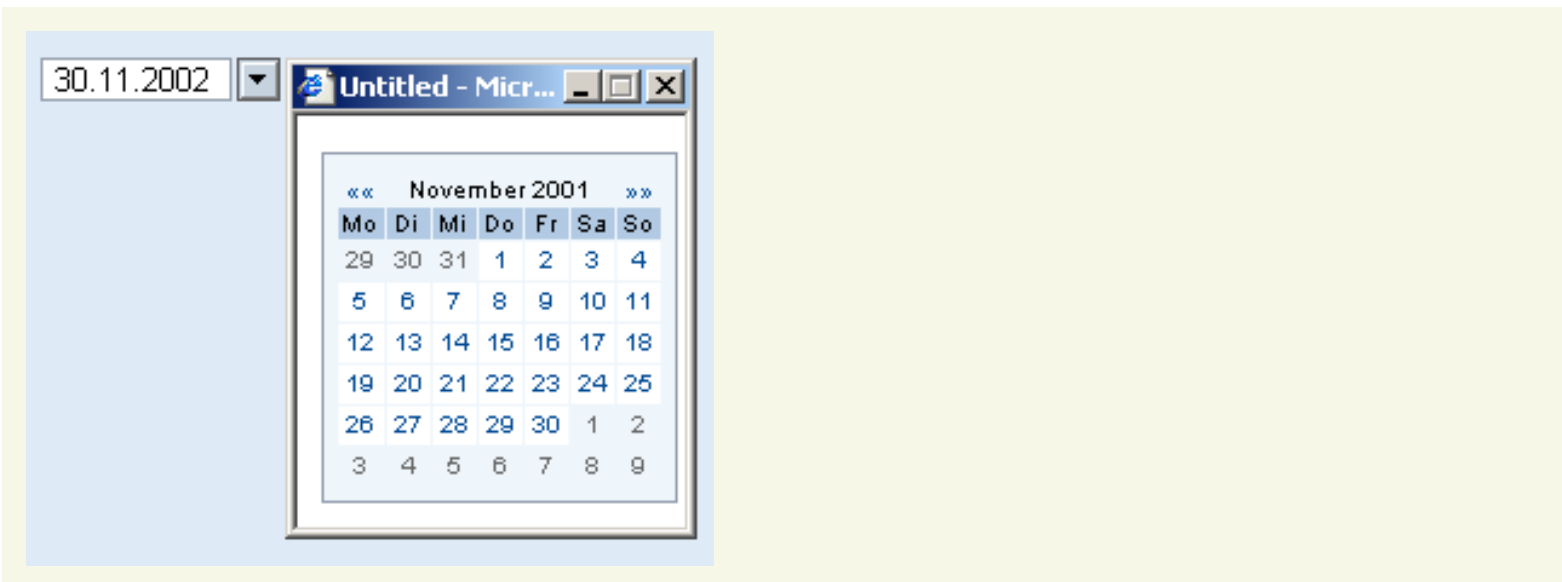


Figure 8: Date input field with date picker

Usage - Required Fields

Some fields require that users enter a value before they can continue, for example, before they can save data. Set **required** = TRUE for making an input field a required field; its corresponding [label](#) has also to be set to **required**.

Usage - Read-only Fields

Read-only fields (**disabled** = TRUE) are input fields that do not allow users to enter data.

Use read-only fields for data that have previously been entered by the user, for example, on a preceding page or during a previous session, or by the system, and that currently cannot be changed by the user. Often, the data that the user is allowed to enter depend on the protected data that are displayed in a read-only field.

Usage - Password Fields

Use password fields (**password** = TRUE) when users have to enter a password, for example, in a login dialog.

Usage - Invalid Fields (Error State)

Set the error state for an input field (**invalid** = TRUE), whenever the system detects an input error that the user committed. Depending on the system behavior, an additional error message should appear immediately, or after a certain "stable" system state has been reached, when the system performs a more thorough error check.

For details on error fields see [Error Handling](#).



[Top](#)

Related Controls

[Label](#), [Checkbox](#), [Radio Button](#), [Grid Layout](#), [Flow Layout](#), [Group](#)



[Top](#)

More Info about Input Field

[Browser Compatibility](#) | [Editability in Style Editor](#) | [Accessibility – 508 Support](#)

Browser Compatibility

Renders in every supported browser.



[Top](#)

Editability in Style Editor

In the Style Editor, it is possible to modify the following attributes of the input field control:

Group	Style	IE 5 and above	Netscape 4.7
Field Styles	Border Width, Style and Color	x	
	Padding	x	
Standard-Sized Field	Font Size	x	
	Height	x	x
Small-Sized Field	Small Font Size	x	
	Small Height	x	x
Invalid Field	Border for Invalid Input	x	
Required Field	Font Color of "Required" Indicator	x	x
Background	Background Color of Editable Fields	x	
	Background Color of Non-Editable Fields	x	

Table 1: Editable styles for the input field control

For common styles see section [HTMLB Controls and Style Editor](#) in *Customer Branding and Style Editor*.



[Top](#)

Accessibility – 508 Support

Input fields have to be used in combination with the label element which points to the assigned input field. This ensures, that screenreaders are aware of the relationship between the both elements and can read the correct label to the according field.

- **Keyboard:** The input field is inserted into the accessibility hierarchy by default.
- **Default Description:** Is provided by the HTMLB rendering engine.
- **Application-specific Description:** Set an additional description using the *setTooltip* method if needed.
- **Label:** Has to be connected to a label control (use method *setLabelFor* for identifying the corresponding input field)



[Top](#)

Control API for Input Field (inputField)

An framed area that allows user input. The inputField can be displayed with default input. A user can type new text or edit the existing text.

- design**
 Defines the size of the input field. The value for this attribute can be "STANDARD" or "SMALL".
- disabled**
 A boolean value that defines if the inputField allows input. A disabled inputField has a different background color.
- firstDayOfWeek**
 Controls the date navigator. The value for the firstDayOfWeek has to be between 1 (=Monday) and 7 (=Sunday). If the value is set e.g. to 3 the datenavigator starts with Wednesday when activated.
 To work with the firstDayOfWeek attribute the attribute showHelp must be set to true and the attribute type must be set to date.
- id**
 Identification name of the inputField.
- invalid**
 A boolean value that controls the frame of the inputField.


```
invalid="true": InputField is displayed with a red frame.
invalid="false": InputField is displayed with the regular frame color
```
- maxlength**
 Defines the maximum amount of characters allowed for the inputField. If the type attribute is set e.g. to date or time the 'maxlength' has to take care of the characters delivered by this format and local settings.
- password**
 A boolean value that controls the echo of the inputField. If set to "true" the typed in characters are echoed - displayed as asterisks (*). A common use for this attribute is to inquire passwords.
- required**
 This attribute sets a different style sheet class for a required inputField (that is a inputField that has to be filled out by the user). That gives you the opportunity do create a complete different look for a required inputfield (e.g. light blue background). Use the 'label' control to indicate with a acharacter that the inputField is required (e.g. an asterisk).
- showHelp**
 A boolean value that activates a help button when set to "true". 'showHelp' shows effect only when 'type' is set to "date". The help button pops up the date navigator allowing selection of the date by clicking on to the required day.
 If you define a textView before the inputfield (to explain the meaning of the inputField) and than an inputField with enabled showHelp it is recommended to place the textView and the inputField in a grid or a tableView for better formatting.
 If textView is not placed in a grid or tableView a line wrap between textView and inputField will occur.
- size**
 Defines the width of the inputField in characters. The frame of the inputField is adjusted accordingly considering the actual text font and the design attribute. The inputField width can also be set by the attribute 'width'. If 'size' and 'width' are set the 'width' attribute has priority and overwrites the 'size' setting.
- type**
 If 'type' is set to date a help button to call the dateNavigator can be generated (see 'showHelp'). Other than that this attribute has no further effect on the client side. It can be used later on when the form gets processed.
 Note: The type INTEGER is not null save and will therefor cause an exception when the field is empty. We recommend to use type STRING instead.
- value**
 Default string that is displayed in the inputField frame. The 'maxlength' attribute has no effect on the 'value' attribute. The 'value' string is not truncated to 'maxlength'.
- visible**
 A boolean value that defines if the inputField is visible or invisible.
- width**
 Defines the width of the inputField in pixel or percent. This attribute allows better adjustment of the inputField in a form.
 The inputField width can also be set by the attribute 'width'. If 'size' and 'width' are set the 'width' attribute has priority and overwrites the 'size' setting.

attribute	req.	values	default	case sens.	JSP taglib	classlib
-----------	------	--------	---------	------------	------------	----------

design	no	STANDARD SMALL	STANDARD	yes	design="SMALL"	setDesign(InputFieldDesign.SMALL)
disabled	no	FALSE TRUE	FALSE	yes	disabled="TRUE"	setDisabled(true)
firstDayOfWeek	no	range 1 to 7	1	-	firstDayOfWeek="3"	setFirstDayOfWeek(3)
id	yes	String	none	yes	id="GetInput"	
invalid	no	FALSE TRUE	FALSE	no	invalid="TRUE"	setInvalid(true)
maxlength	no	Numeric	25	no	maxlength="25"	setMaxlength(25)
password	no	FALSE TRUE	FALSE	no	password="TRUE"	setPassword(true)
required	no	FALSE TRUE	FALSE	yes	required="FALSE"	setRequired(false)
showHelp	no	FALSE TRUE	FALSE	yes	showHelp="TRUE" only effective if type is set to "date"	setShowHelp(true) only effective if type is set to "date"
size	no	Numeric	30	-	size="30"	setSize(30)
type	no	BCD BOOLEAN DATE INTEGER STRING TIME	STRING	yes	type="INTEGER"	setType(DataType.INTEGER)
value	no	String	none	no	value="Your name here"	setValue("Your name here")
visible	no	FALSE TRUE	TRUE	yes	visible="FALSE"	setVisible(false)
width	no	Unit	150	-	width="200"	setWidth("200")

Example

```
<hbj:inputField
  id="InputName"
  type="string"
  maxlength="100"
  value="Your name here"

/>
```

Result

Item List

[Usage](#) | [Types](#) | [Related Controls](#)



Figure 1a-b: Ordered item list (left) and unordered item list (right)

Use an ordered item list if you want to represent items in a certain predefined sequence, e.g. a ranking. Use an unordered list if there is no predefined item order.



[Top](#)

Usage

Use item lists whenever you want to present a list of items in an unobtrusive way, where reading is the primary usage and where there is - apart from links - no interaction on the list elements.

Lists have the following characteristics:

- List items are read-only.
- List items may contain links.
- Item lists do not scroll. Therefore make sure that all list items fit the iView!
- Item lists consist of one column only; for multiple columns include the lists in an HTML table with one row and several columns (2-3) or use a [table view](#).
- Lists may be ordered (numbers) or unordered (bullets).
- Lists may be nested; do not use more than 2-3 levels!

Description - Label or Heading

Item lists may have a label or heading. A descriptive label may be placed above or to the left of the item list, a heading should typically be placed above the item list.

Use the [label](#) control for both labels and headings. Note that the label control allows to set font attributes in order to emphasize headings.



Figure 2: Item list with a heading



Types

There are two types of item lists, an unordered or bullet list, and an ordered or numbered list. Both types are set using the Boolean attribute **ordered**: **ordered** = TRUE renders a numbered list, **ordered** = FALSE a bullet list.

Usage - Types

Use an ordered item list if you want to present items in a certain predefined sequence, e.g. a ranking. Use an unordered list if there is no predefined item order.



Related Controls

[Table View](#), [Text View](#), [Link](#), [Listbox](#), [Label](#)

More Info about Item List

[Editability in Style Editor](#) | [Accessibility – 508 Support](#)

Editability in Style Editor

The following characteristics are editable in the Style Editor:

- List Style Image (List Bullet)
- List Margin

In the Style Editor, it is possible to modify the following attributes of the item list control:

Style	IE 5 and above	Netscape 4.7
URL to List Style Image	x	x
List Margin	x	

Table 1: Editable styles for the item list control

For common styles see section [HTMLB Controls and Style Editor](#) in Customer *Branding and Style Editor*.



[Top](#)

Accessibility – 508 Support

- **Keyboard:** The item list is **not** inserted into the accessibility hierarchy by default.
- **Default Description:** Not needed.
- **Application-specific Description:** Not needed.



[Top](#)

Control API for Item List (itemList)

A static list that appears in the interface. The list can be displayed as bulleted or numbered (ordered) list. An element in an itemList is called listItem.

- **id**
Identification name of the itemList.
- **ordered**
A boolean value that defines if the list is displayed as bulleted (ordered="false") or numbered list (ordered="true").

attribute	req.	values	default	case sens.	JSP taglib	classlib
id	yes	String	none	yes	id="ImportantItems"	
ordered	no	FALSE TRUE	FALSE	yes	ordered="TRUE"	setOrdered(true)

listItem

Defines the items in the itemList. A listItem can be built with any control (checkbox, image, textView etc.) and one listItem can contain more than one control.

Example

```
<hbj:itemList
  id="ImportantItems"
  ordered="true"
  >

  <hbj:listItem>
    <hbj:textView text="Introduction" />
  </hbj:listItem>

  <hbj:listItem>
    <hbj:textView text="Definitions" />
  </hbj:listItem>

  <hbj:listItem>
    <hbj:textView text="Main Part" />
  </hbj:listItem>

  <hbj:listItem>
    <hbj:textView text="Conclusion" />
  </hbj:listItem>

</hbj:itemList>
```

Result

```
1. Introduction
2. Definitions
3. Main Part
4. Conclusion
```

Label

[Usage](#) | [Design-relevant Attributes](#) | [Related Controls](#)

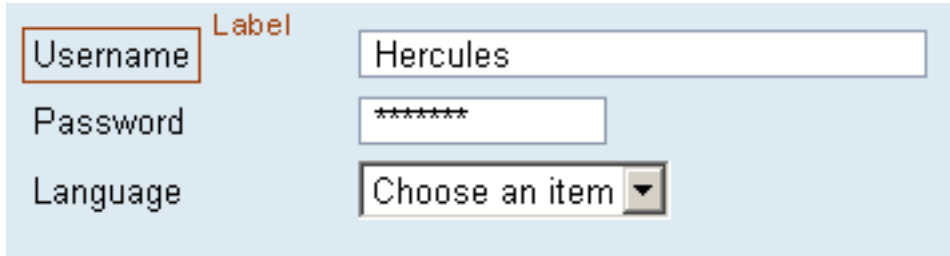


Figure 1 shows a form with three rows. The first row has a label 'Username' (highlighted with a red box) and a text input field containing 'Hercules'. The second row has a label 'Password' and a text input field containing '*****'. The third row has a label 'Language' and a dropdown menu with the text 'Choose an item' and a downward arrow. The word 'Label' is written in red above the 'Username' label.

Figure 1: Example of a label with its corresponding input element

Labels are texts that describe associated input elements. The label control creates a firm connection between the descriptive text and the respective element.



[Top](#)

Usage

Labels are simple text elements that can be used for descriptive texts. Labels can wrap and spread over multiple lines. Use labels to describe input elements, such as, [dropdown list boxes](#), [input fields](#), [list boxes](#), and [item lists](#).

Note: [Checkbox](#) and [radio button](#) controls already include their own labels.

Required Fields

Set the label to required if the input field is required.

Positioning

Typically, labels are placed in front of the input element they describe (figure 1). In some cases, labels may be placed above the corresponding input element (figure 2). See also the special case for input fields [below](#) (figure 3).

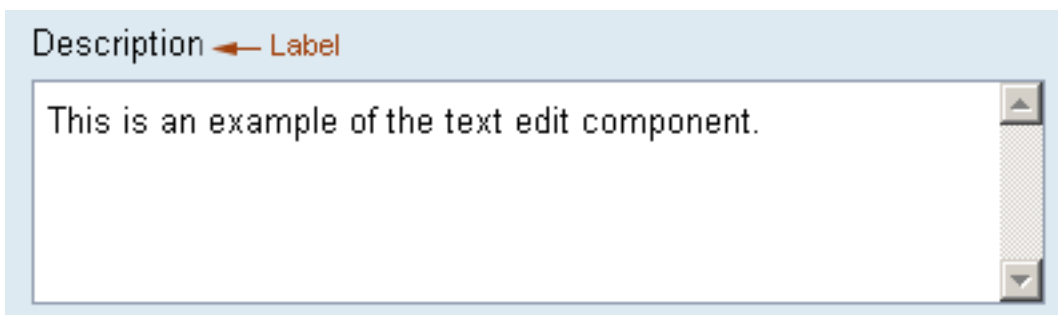


Figure 2 shows a form with a label 'Description' (highlighted with a red box) positioned above a text area. The text area contains the text 'This is an example of the text edit component.' The word 'Label' is written in red above the label 'Description'.

Figure 2: Label above a text edit control

Size

The label control can be set to two sizes, standard and small. Use the sizes depending on the context of the label, that is, use the standard size if the surrounding fields are standard size, use small if the surrounding fields are small.

There is one exception to this rule: You may use small labels if you place the labels above input fields to achieve a more compact design (figure 3).

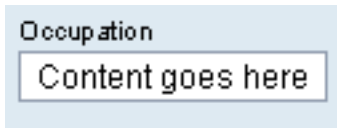


Figure 3: Use small labels for labels that are placed above their associated input fields

Relation to Text View

Do **not** use the [text view control](#) as a label because only the label establishes a connection between the description and the corresponding input element. This connection is mandatory for achieving accessibility.

Help Texts

Help texts are special descriptions that are placed behind or, if space is limited, below and left-aligned with input fields (figure 4). Do **not** use the label control for help texts. Use the [text view control](#), instead. Set the help text size to small (style **Legend**).

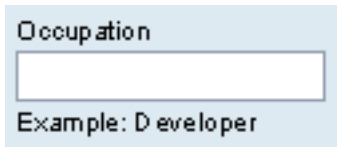


Figure 4: A small label used in conjunction with a small help text below the associated input fields - do not use the label control for the help text (below the input field)



[Top](#)

Design-relevant Attributes

The appearance of the label control can be determined through several attributes. The attribute **design** allows to set the size of the label: **design** = LABEL sets the standard size, **design** = LABELSMALL sets the small size. The attribute **labelFor** established the connection to the associated input control. Further attributes set the width, wrapping behavior (Boolean attribute **wrapping**), text, and tooltip text of the label. The Boolean attribute **required** must be set to TRUE for labels that describe required fields

For details see page [Control API for Label](#).

Label



[Top](#)

Related Controls

[Dropdown List Box](#), [List Box](#), [Input Field](#), [Text Edit](#), [Text View](#), [Item List](#)



[Top](#)

More Info about Label

[Editability in Style Editor](#) | [Accessibility – 508 Support](#)

Editability in Style Editor

The label control is editable in the Style Editor via "Text". Common styles used for labels (changes affect other elements) are:

- Font Size
- Font Color
- Font Family
- Font Weight
- Font Style

For an overview of all common styles see section [HTMLB Controls and Style Editor](#) in Customer *Branding and Style Editor*.



[Top](#)

Accessibility – 508 Support

- **Keyboard:** The label is **not** inserted into the accessibility hierarchy by default.
- **Default Description:** Is provided by the HTMLB rendering engine.
- **Application-specific Description:** Typically, an additional description is not needed. Set a description using the *setTooltip* method only if the label text is not sufficient for explaining the meaning, function, or purpose of the corresponding control.
- **Corresponding Input Elements:** Checkbox, dropdown list box, input field, radio button, and text edit control



[Top](#)

Control API for Label (label)

A multiline region for displaying text. Text in the component is restricted to a single font, size and style unless set with HTML commands. label works similar to textView. In addition a label has an associated component like an inputField or listBox.

- **design**

Defines the appearance of the label. The CSS controls how the different options are rendered. The following description is based on the standard CSS delivered.

- LABEL
Text size and attributes STANDARD
- LABELSMALL
Text size -2 in comparison to STANDARD

- **encode**

A boolean value that defines how the label text is interpreted. HTML text formatting commands (e.g. <h1>, <i> etc.) can be used to change the display of the text. If there are no formatting commands in the text string, the encode attribute has no effect.

Example

```
text="<h1><i>Important</i></h1>
```

Important

```
encode = "false"
the text string is rendered by interpreting the formatting commands.
```

```
Encode = "true"    <h1><i>Important</i></h1>
the formatting commands are displayed and not interpreted
```

- **id**

Identification name of the label.

- **labelFor**

Identification name of the next component, which is associated with the label.

- **required**

A boolean value. If set to "true" a character (e.g. an asterisk (*) in red color) defined by the style sheet is placed at the end of the text string. This is a common method to indicate that input is required. See also inputField.

- **text**

Defines the string of text displayed. See 'encode' for a formatting example with embedded HTML commands.

- **tooltip**

Defines the hint of the textView which is displayed as the mouse cursor passes over the textView, or as the mouse button is pressed but not released.

- **width**

Defines the width of the textView. The width shows only effect when the 'wrapping' attribute is set to "true". Otherwise the width and layout follows the HTML commands in the text string.

- **wrapping**

A boolean value. If set to "true" the text is word wrapped at the set 'width' or - if no 'width' is set - at the form width.

attribute	req.	values	default	case sens.	JSP taglib	classlib
design	no	LABEL LABELSMALL	LABEL	yes	design="LABEL"	setDesign(TextViewDesign.LABEL)
encode	no	TRUE FALSE	TRUE	yes	encode="FALSE"	setEncode(false)
id	yes	String	none	yes	id="Intro_Text"	
labelFor	yes	String	none	yes	labelFor="id_inputField"	setLabelFor(id_inputField_component)
required	no	TRUE FALSE	FALSE	yes	required="TRUE"	setRequired(true)
text	no	String	none	no	text="Your name please"	setText("Your name please")
tooltip	no	String	none	no	tooltip="Name required"	setTooltip("Name required")
width	no	Unit	100%	no	width="300"	setWidth("300")

wrapping	no	TRUE FALSE	FALSE	yes	wrapping="TRUE"	setWrapping(true)
----------	----	---------------	-------	-----	-----------------	-------------------

Example

```
<hbj:label
  id="label_InputName"
  required="TRUE"
  text="ZIP Code"
  design="LABEL"
  labelFor="InputName"

/>

<hbj:inputField
  id="InputName"
  type="string"
  maxlength="100"

/>
```

Result

ZIP Code *

Link

[Usage](#) | [Types](#) | [Links vs. Buttons](#) | [States](#) | [Design-relevant Attributes](#) | [Related Controls](#)

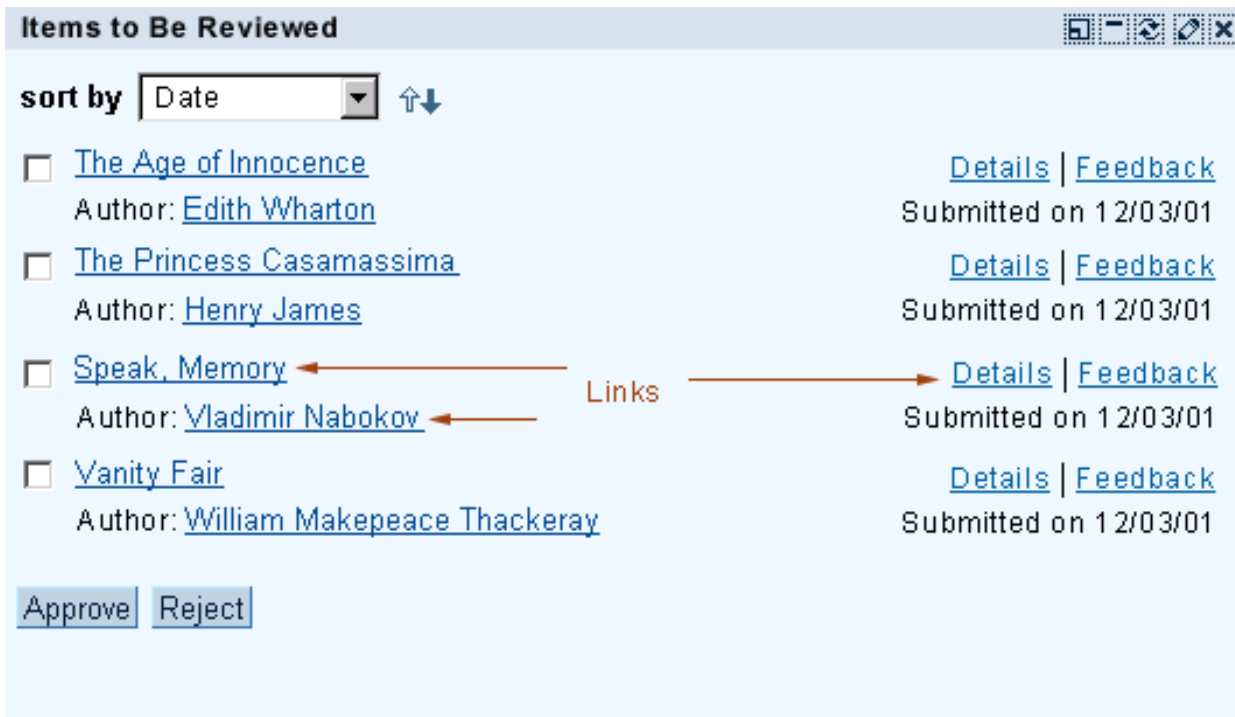


Figure 1: Example of links in the content area



Usage

The link control is not the simple topic it may seem to be. Not all links should be handled equally. There are hypertext links in the navigation area, header area, and content area as well as on buttons and in tables and applications. All of these links can have different appearances and behaviors.

For aesthetic and usability reasons, the links in the navigation area (not to be confused with content area navigation links!) have a different appearance from those in the content area. It's important that the user recognizes these links as pertaining to the overall navigational structure and not just a way to drill-down to detail information or jump to a page in a different context.

The HTMLB control called "link," which is described here, refers to hypertext links (referred to here as "links") in the **content area**.

Positioning

Links may either appear

- as part of a larger text, (for example, an article about customer service might contain a context link to the mySAP CRM homepage) or
- standalone (for example, a list of links to articles about CRM, a "More" link placed at the end of a abstract to navigate to additional information, etc.)

Standalone links should be grouped together separately from buttons and vice versa. When possible, functions and links should be grouped together and displayed in the same way (either all as links or all as buttons), as in figure 1. A mixture of links and buttons in the same grouping context should

Link

be avoided.

View switching links are most often displayed above the content to which they refer, as in figure 2.

Capitalization

It is not necessary to make any special capitalization considerations for links that are part of a larger context or that are automatically generated (such as contact names, document titles, etc.). In the same way that buttons require title case capitalization (i.e. the first word always capitalized, all significant words are capitalized, prepositions and articles are not capitalized), so do function, navigation, toggle and view switching links.

Market Analysis.pdf

Title case → [Show Contents of All Messages](#)

	Subject	Author	Date
Sentence case	▼ Your opinion on our market analysis	Mary Scott	May 25th 2001 , 09:05
	A great challenge	Daniel Jackson	May 25th 2001 , 09:09
	I agree	Charles Brown	May 25th 2001 , 09:15
	Let's go for it	Mary Scott	May 25th 2001 , 09:24
	Consider the risks	Daniel Jackson	May 25th 2001 , 09:12
	My opinion	Charles Brown	May 25th 2001 , 09:11
	▼ The numbers look suspicious	Jena Watson	May 25th 2001 , 09:11
	Accounting says they'll pass	Albert Finkel	May 25th 2001 , 09:11

[Create New Topic](#) [Subscribe to this Discussion](#)

Figure 2: Example of an iView where links have both title and sentence case depending on the usage

[Top](#)

Types

Although there is only one link type from a technical point of view, we must make a distinction between the various kinds of links in the content area. This helps to establish usage rules for links and to make a distinction between buttons and links.

Note:

Based on what purpose the links serve, we can establish five different types of content area links: view switch, toggle, drill-down, function, and navigation.

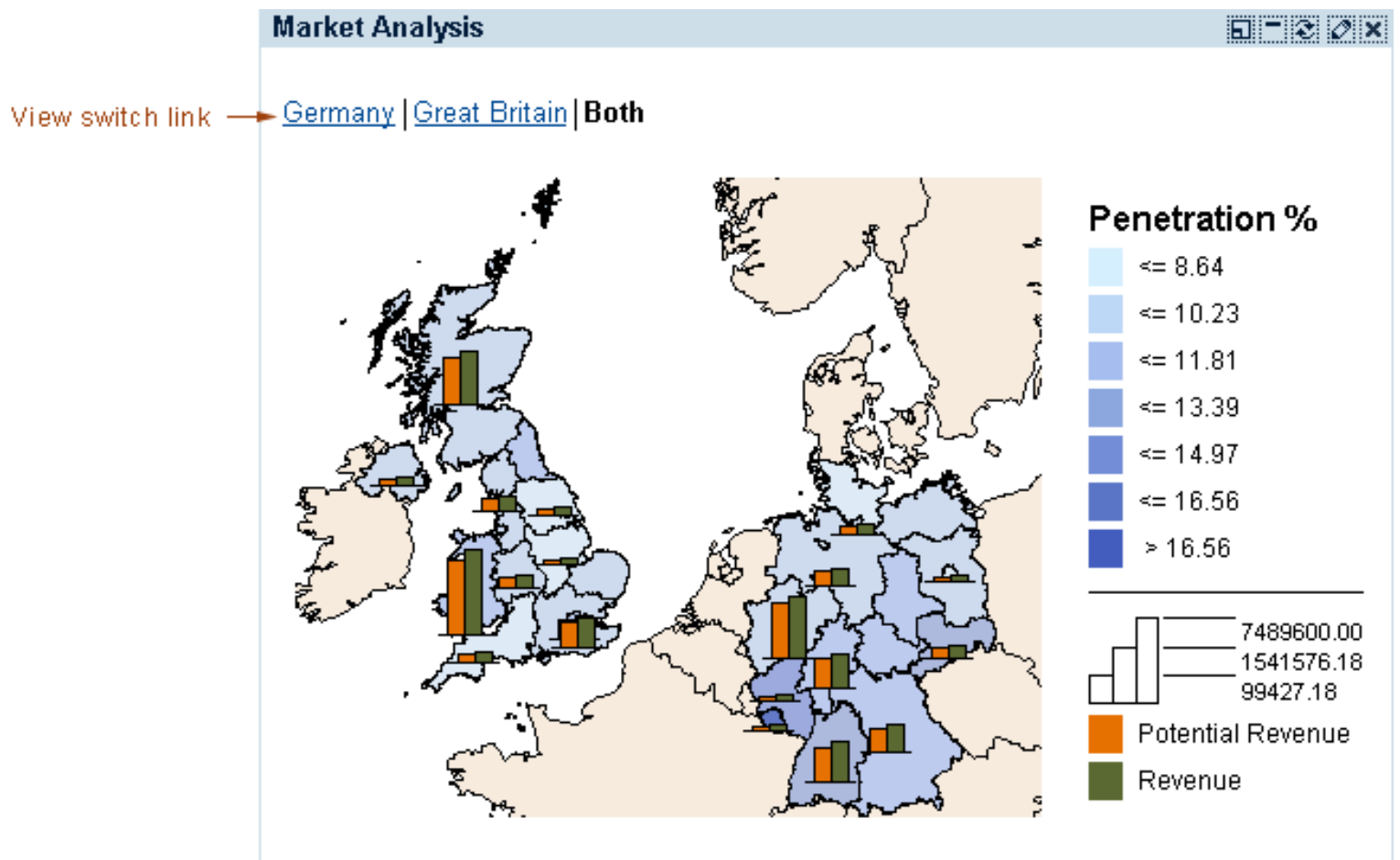


Figure 3: Example of an iView with view switch links.

View Switch Links

View switch links are similar in function to toggle links, but are different in that they are always visible. The view switch links are an alternative to the tabstrip control. The advantage that view switch links have over the tabstrip is that they take up less space and can be used vertically, if this makes more sense in the application (for example in mobile applications where the format of the device allows more vertical than horizontal space). This type of link is similar to the navigation link. As opposed to navigation links, view switch links all refer to the same context and must be persistent in all the views. The currently selected view should not be presented as a link, but as bold text (see figure 3 above). View switch links should be separated from one another by a vertical line, like this one |.

Use title case (see [Capitalization](#)) for view switch links.

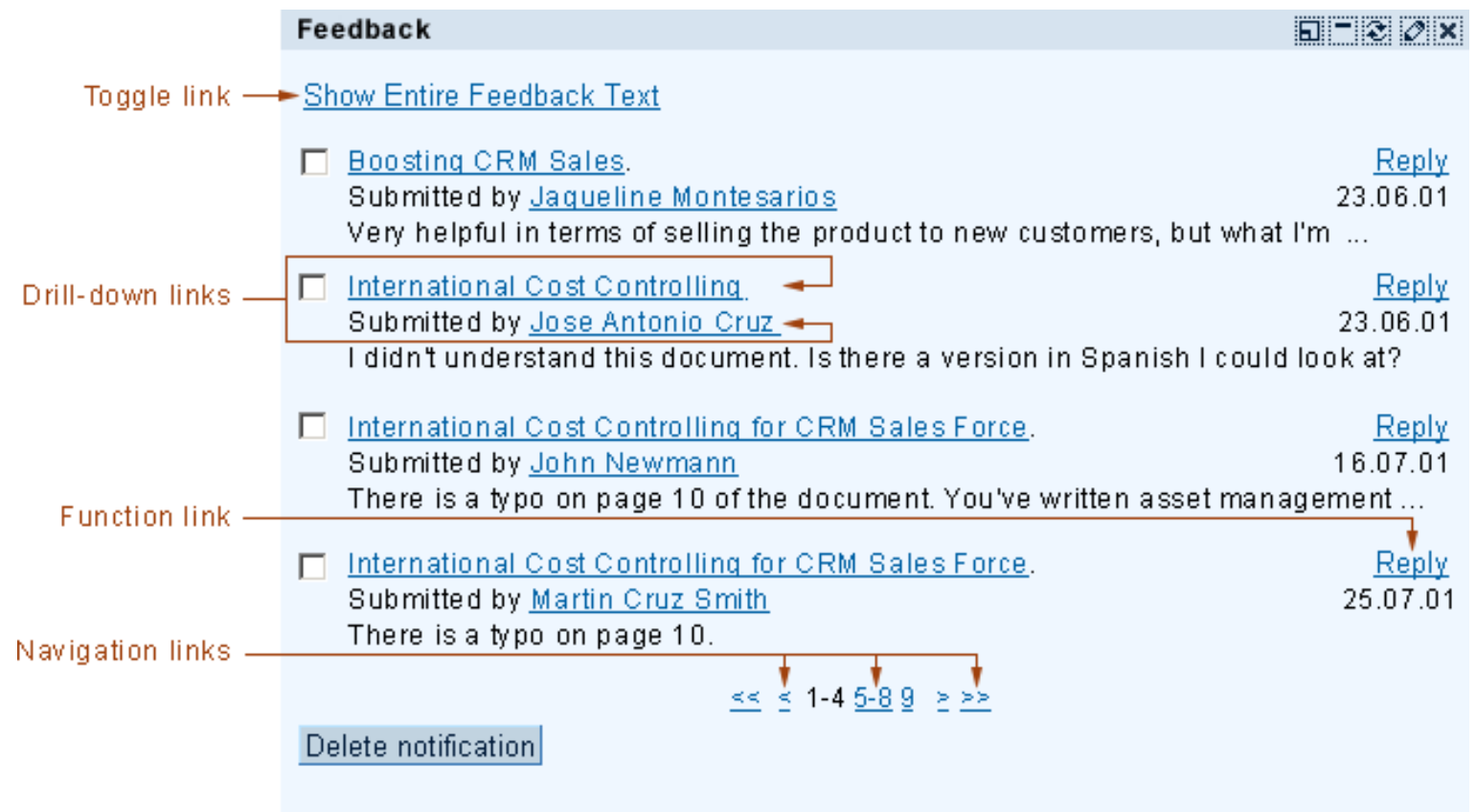


Figure 4: Example of an iView with toggle, drill-down, function and navigation links.

Toggle Links

Toggle links are used when there are two alternative views of the current data. Toggle links are always pairs of links, but only one is visible at a time. In figure 4 the toggle link reads, "Show Entire Feedback Text." If the user were to click on that link, the alternative view would show the feedback in its entirety and the text of the link would change to "Show Only Feedback Preview."

Some common examples of toggle links are:

- Expand All / Collapse All
- Show Chart View / Show Table View
- Hide Help / Show Help
- etc.

Use title case (see [Capitalization](#) below) for toggle links.

Drill-Down Links

A drill-down link allows a user to see more detailed or specific information. For example, in the overview of an address book, the contact names are drill-down links that allow the user to access details on that person. In a mail inbox, the title of each message would automatically be a link to the contents of the message.

Some common drill-down links are:

- Contact name
- Customer name
- Document title
- Message title
- Report title
- Revenue

Link

- etc.

These links are most often automatically generated and the developer should make no attempt to influence the capitalization. (See figure 2.)

Function Links

A function link allows the user to carry out an action.

Although in general buttons should be reserved for functions (see [Links vs. Buttons](#) for more information), there may be cases where a link would be preferable or where the distinction between a function and a link to another view might be very blurry.

For example, when the user wants to subscribe to an object (e.g. a folder), the link may just be a link to a new screen where the user has to fill in some more information and then can submit the information to the server. This whole act of subscribing, however, could be thought of as carrying out a function and in some cases might make more sense as a button than as a link, or vice versa depending on the context.

Sometimes you may have a collection of items, for example a list of documents. There are some actions that you can perform all at once on a number of documents whereas there are other actions which make more sense when they are performed on each item one at a time. If the action requires a new screen or additional information (details, feedback, edit, reply, etc.), chances are that you can only perform the action on each item one at a time. In such cases, a link is generally preferable to a button.

Taking the iView in figure 1 as an example, we can see that the user can select a number of documents and approve or reject them all at once by using the checkboxes and buttons. However, if the user were to want to see the details of a number of documents, it makes more sense to have him choose a link next to each document. Otherwise, the user would have a number of detail screens and would likely be confused about which details belong to which document.

If your application has a list of items, and each one requires it's own function (see figures 1 and 4 for examples), it is preferable to use links as opposed to buttons for functions in this case. This is mainly due to aesthetics, but it is also a usability factor. An application with a wall of buttons makes the application look heavy and complicated.

Some common function links are:

- Details
- Feedback
- Add
- Subscribe
- Reply
- Edit
- etc.

Use title case (see [Capitalization](#)) for function links.

Navigation Links

Many times, there will be navigation within an iView or application which is independent of the main navigation of the Portal. End users might not even perceive these links as "navigation" per se. Navigation links allow users to move backwards or forwards through a data set or process. Sometimes the difference between a drill-down link and a navigation link might be difficult to assess, for example with a "more..." link.

Some common navigation links are:

- More...
- Next or Next >
- Back or < Back
- Backward "<" and forward ">" as well as back to the beginning "<<" and forward to the end ">>" arrows in text form (as in figure 4 above)
- Numbers to navigate to a set of entries (as in figure 4 above)

Use title case (see [Capitalization](#)) for navigation links.



Links vs. Buttons

In general, use **links** to indicate **navigation** to another HTML page or to a different view of the current information as well as to link to further or more detailed information. Links commonly appear within the context of the application (within trees, tables or text).

In general, use **buttons** to indicate that a **function** can be carried out (save, print, close, delete, etc.) or that a process can be started (subscribe, etc.). Buttons generally appear at the bottom left of a grouped area to indicate a function that can either be performed on selected items (if checkboxes appear as well) or that apply to the whole screen. (See the section of these guidelines called [Buttons](#) for further information.)

For more information about cases where links may be used to indicate a function, see [function links](#) above.



Figure 5: Example of an iView where links and buttons are both used.



States

The link control has four states: link (i.e. unvisited, normal state), hover, visited and active.

This is the link control	Example of a link in its normal state
This is the link control	Example of a link in the hover state
This is the link control	Example of a link in the visited state
This is the link control	Example of a link in the active state

Table 1: Examples of the different states the link control can have



Design-relevant Attributes

Attribute **reference** allows to assign an URL to the link, whereas attribute **target** allows to set a link target. The latter follows the HTML conventions for specifying link targets. Attribute **text** sets the link text, and attribute **tooltip** the tooltip text for the link.

For details see page [Control API for Link](#).



[Top](#)

Related Controls

[Button](#)



[Top](#)

More Info about Link

[Browser Compatibility](#) | [Editability in Style Editor](#) | [Accessibility – 508 Support](#)

Browser Compatibility

Netscape does not recognize the hover state.



[Top](#)

Editability in Style Editor

In the Style Editor it is possible to edit the following styles:

- Font Color for Unvisited ("link"), Active ("active"), Visited ("visited") and Mouseover ("hover")
- Text Decoration for Unvisited ("link"), Active ("active"), Visited ("visited") and Mouseover ("hover")

For an overview of all common styles see section [HTMLB Controls and Style Editor](#) in Customer *Branding and Style Editor*.



[Top](#)

Accessibility – 508 Support

- **Keyboard:** The link is inserted into the accessibility hierarchy by default.
- **Default Description:** Is provided by the HTMLB rendering engine.
- **Application-specific Description:** Set an additional description using the *setTooltip* method if needed.
An additional description is needed if users need more specific information or instructions. In general, the description has to be extended if a link introduces an interaction that cannot be recognized by a blind user. For example, the descriptions needs to be extended if the link opens a new window.



[Top](#)

Control API for Link (link)

Defines a link to another page. The text of the link becomes an underline and is displayed in a different color. An image can be defined as link as well - see the example for details.

- **id**
Identification name of the link.
- **onClick**
Defines the action that will be processed when the user clicks on the link. If 'onClick' is specified, the event handling routine is called.
- **reference**
Specifies the address of the page/document to be opened. If the 'onClick' attribute is defined the event handling routine is started and the 'reference' string is handed to the event handling routine. The referenced document is not opened - the event handling routine has to do that.
If the 'onClick' attribute is not defined, the link is opening the referenced document.
- **target**
Specifies the name of the frame where the document is to be opened. The following values refer to w3c HTML-standard.
 - **_blank**
The web client should load the designated document in a new, unnamed window.
 - **_self**
The web client should load the document in the same frame as the element that refers to the target.
 - **_parent**
The web client should load the document into the immediate FRAMESET parent of the current frame. This value is equivalent to **_self** if the current frame has no parent.
 - **_top**
The web client should load the document into the full, original window (thus canceling all other frames). This value is equivalent to **_self** if the current frame has no parent.
- **text**
A text string that is displayed underlined and in different color. If no 'text' attribute is provided or a the 'text' attribute is set to an empty string, the link is not displayed.
- **tooltip**
Defines the hint of the link which is displayed as the mouse cursor passes over the link, or as the mouse button is pressed but not released.

attribute	req.	values	default	case sens.	JSP taglib	classlib
id	yes	String	none	yes	id="ImportantItems"	
reference	no	String	none	no	reference="http://www.sap.com"	setReference("http://www.sap.com")
target	no	_blank _self _parent _top	_self	no	target="_TOP"	setTarget("_TOP")
text	no	String	none	no	text="To the beach"	addText("To the beach")
tooltip	no	String	none	no	tooltip="Enjoy and relax"	setTooltip("Enjoy and relax")

events	req.	values	default	case sens.	JSP taglib	classlib
onClick	no	String	none	yes	onClick("ProcessLink")	setOnClick("ProcessLink")

Example - Text as link

```
<hbj:link
    id="link1"
    text="Link to google"
    reference="http://www.google.com"
    target="_TOP"
    tooltip="this takes you to: http://www.google.com"
    onClick="LinkClick"
/>
```

Result

[Link to google](http://www.google.com)

Example - Image as link (and getting the resource path once - at the beginning of the JSP. This is useful when the JSP uses several images)

```
<%-- Get resource url of component --%>
<% String imageURL = componentRequest.getPublicResourcePath() + "/images/"; %>

<hbj:link
    id="link1"
    text=""
    reference="http://www.sap.com"
    target="_TOP"
    tooltip="this takes you to: http://www.sap.com"
    onClick="LinkClick"
    <hbj:image
        src="<%= imageURL+"sap.gif\" %>"
        alt="Image not available" />
</hbj:link>
```

Result**Example - Text and Image as link (and getting the resource path at the control)**

```
<%@ page import="com.sapportals.portal.prt.resource.IResource" %>
.
.

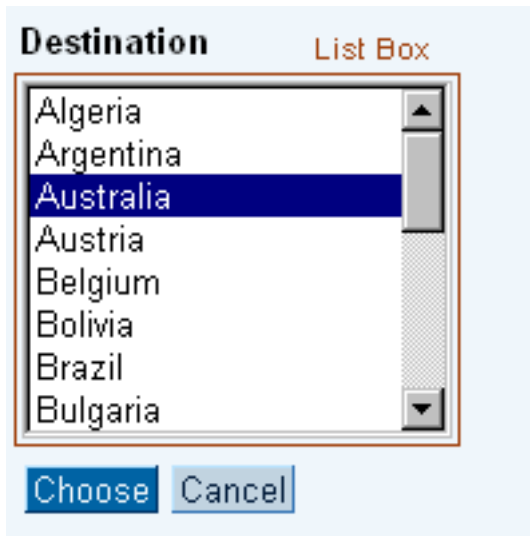
<hbj:link
    id="link1"
    text="Link to SAP"
    reference="http://www.sap.com"
    target="_TOP"
    tooltip="this takes you to: http://www.sap.com"
    onClick="LinkClick" >
    <hbj:image id="image_logo"
        alt="Image not available"
        src="" >
    <%
        IResource rs = componentRequest.getResource(IResource.IMAGE, "images/sap.gif");
        image_logo.setSrc(rs.getResourceInformation().getURL(componentRequest));
    %>
    </hbj:image>
</hbj:link>
```

Result



List Box

[Usage](#) | [Design-relevant Attributes](#) | [Related Controls](#)



The list box is a box that displays a list of items where users can select one item from. If the number of items exceeds the box size, a vertical scrollbar is activated. The list box is read-only.

Figure 1: Example of the list box control.



Usage

A list box offers a set of choices from which a user can select one item. If the number of items exceeds the control size, a vertical scrollbar is activated. An item in the list box is called list box item. The list box is read-only.

Note: The list box control does not render a descriptive label automatically. Use the [label](#) control to add a description. See there, how you can change text attributes if you need to highlight the label, for example, make it bold (see figure 1).

Choosing the Appropriate Selection Control

A list box is similar in function to a dropdown list box - both offer a list of items where users can select one item from, that is, both are single-selection lists.

See [Forms - Using Different List Types](#) for guidelines on choosing the appropriate selection control.

Note: For very small item numbers (2-6) and if the users should see all alternatives, use [radio buttons](#).



Design-Relevant Attributes

You can set the number of displayed lines of a list box (**size**), its width (**width**), and whether it is enabled or disabled (Boolean attribute **disabled**).

See page [Control API for List Box](#) for details.



[Top](#)

Related Controls

[Dropdown List Box](#), [Item List](#), [Radio Button](#), [Table View](#), [Tree View](#)



[Top](#)

More Info about List Box

[Browser Compatibility](#) | [Editability in Style Editor](#) | [Accessibility – 508 Support](#)

Browser Compatibility

To be provided



Editability in Style Editor

The list box itself renders as the standard browser control. Style Editor changes can be made to the corresponding label.



Accessibility – 508 Support

List boxes have to be used in combination with the label element which points to the assigned list box. This ensures, that screenreaders are aware of the relationship between the both elements and can read the correct label to the according list box.

- **Keyboard:** The listbox inserted into the accessibility hierarchy by default.
- **Default Description:** Is provided by the HTMLB rendering engine.
- **Application-specific Description:** Set an additional description using the *setTooltip* method if needed.
- **Label:** Has to be connected to a label control (use method *setLabelFor* for identifying the corresponding list box).



Control API for List Box (listBox)

A set of choices from which a user can select one items. If the number of text lines exceeds the control size, a vertical scrollbar is activated. An item in the listBox is called listBoxItem. listBoxItems are explained above - after the listBox description.

- disabled**
 A boolean value that defines if the listBox is clickable. If the listBox is disabled it is not selectable. A disabled listBox has a different color for the displayed listBoxItem.
 - id**
 Identification name of the listBox.
 - model**
 Defines the model which provides the listBox with data.
 - nameOfKeyColumn**
 Specifies the name of the column that contains the key. This is used when you use an underlying table in the model.
 - nameOfValueColumn**
 Specifies the name of the column that contains the visible text. This is used when you use an underlying table in the model.
 - onSelect**
 Defines the action that will be processed when the user clicks on the enabled listBox. If you do not define a 'onSelect' event the listBox can be clicked but no event is generated.
 - onClientSelect**
 Defines the JavaScript fragment that is executed when the user clicks on the listBox. If both events ('onClick' and 'onClientSelect') are specified, the 'onClientSelect' event is activated first. By default the 'onClick' event is activated afterwards. In the JavaScript fragment you can cancel the activation of the 'onClick' event with the command

```
htmlbevent.cancelSubmit=true;
```

 The 'onClientSelect' event is very useful to save client/server interaction.
- Example**
 A listBox click usually activates the client/server interaction. If an input field has to be filled out for further processing, the JavaScript fragment can check the necessary input on the client side and display a message if the necessary input is missing, without server interaction.
- Note**
 To use JavaScript the JSP has to use the **page** tag (see [page](#) tag).
- selection**
 Specifies the key of the listBoxItem which is displayed in the listBox.
 - Size**
 Sets the number of lines displayed for the listBox. If the number of text lines for listBox is higher then the size attribute a vertical scrollbar is activated - the width of the listBox is not changed, the text display window becomes smaller.
 - tooltip**
 Defines the hint of the listBox which is displayed as the mouse cursor passes over the listBox, or as the mouse button is pressed but not released.
 - width**
 Defines the width of the listBox. Text lines are truncated if the length of the string extends the width.

attribute	req.	values	default	case sens.	JSP taglib	classlib
disabled	no	TRUE FALSE	FALSE	yes	disabled="TRUE"	setDisabled(true)
id	yes	String	none	yes	id="listbox_te"	
model	no	String	none	yes	model="myBean.model"	setModel((IlistModel) model)
nameOfKeyColumn	no	String	none	no	nameOfKeyColumn("k1")	setNameOfKeyColumn("k1")
nameOfValueColumn	no	String	none	no	nameOfValueColumn("v1")	setNameOfValueColumn("v1")
selection	no	String	none	yes	selection("HD")	setSelection("HD")
size	no	Numeric	4	-	size="10"	setSize(10)
tooltip	no	String	none	no	tooltip="select a item"	setTooltip("select a item")
width	no	Unit	max. item length	-	width="100"	setWidth("100")

events	req.	values	default	case sens.	JSP taglib	classlib
onClientSelect	no	String	none	yes	onClientSelect="JavaScript"	setOnClientSelect("JavaScript")
onSelect	no	String	none	yes	onSelect="proc_listbox"	setOnSelect("proc_listbox")

listBoxItem

Defines the items in a dropdownListBox or listBox instead of the model. [See dropdownListBox.](#)

Example

```
<hbj:listBox
  id="LB_CitiesNearby"
  tooltip="Cities surrounding SAP"
  selection="WD"
  disabled="false"
  nameOfKeyColumn="KeyCol"
  nameOfValueColumn="KeyVal"
  onSelect="ProcessCity"
  onClientSelect="PreprocessCity"
>

  <hbj:listBoxItem
    key="HD"
    value="Heidelberg"
    selected="true"
  />

  <hbj:listBoxItem
    key="HK"
    value="Hockenheim"
  />

  <hbj:listBoxItem
    key="WD"
    value="Walldorf"
    selected="true"
  />

  <hbj:listBoxItem
    key="WL"
    value="Wiesloch"
  />

</hbj:listBox>
```

Result



Radio Button

[Usage](#) | [Design-relevant Attributes](#) | [Related Controls](#)

Select your payment method

- Invoice
- Credit Card
- Personal Check

Radio buttons provide users with a single choice from a set of alternative options

Figure 1: A radio button group



[Top](#)

Usage

Radio buttons provide users with a single choice from a set of alternative options. They always appear in a group of at least two radio buttons. Therefore, you should define radio buttons only within the radio button group control, not as single elements.

A click on one choice selects the current choice and deselects the previous choice. Usually, always one radio button is checked. The Internet introduced one exception to this rule. In some cases, a radio button group can initially show up with no radio button checked.

Note: It is not possible to determine the horizontal spacing within a radio button group. If you need a different spacing than that supplied by the radio button group control, use single radio buttons and a [grid layout](#) control if applicable.

Arrangement and Design Alternatives

For details on the arrangement of radio buttons as well as design alternatives see [Forms - Using Radio Buttons](#).



[Top](#)

Design-relevant Attributes

Radio buttons have the **disabled** attribute. Set **disabled** to TRUE if users are not allowed to change their state temporarily. Attribute **text** sets the descriptive label text for a radio button.

For radio button groups there are two relevant attributes: You can determine which radio button is "on" in a group; set attribute

selection to the **id** of the respective radio button. You can also set the column count for radio button groups (attribute **columnCount**).



[Top](#)

Related Controls

[Dropdown List Box](#), [Checkbox](#), [List Box](#), [Label](#), [Grid Layout](#)



[Top](#)

More Info about Radio Button

[Browser Compatibility](#) | [Editability in Style Editor](#) | [Accessibility – 508 Support](#)

Browser Compatibility

The radio button renders in every supported browser.



Editability in Style Editor

The radio button itself renders as the standard browser control. Style Editor changes can be made to the corresponding label.

Radio Button Groups

There is no editability for radio button groups in the style editor.



Accessibility – 508 Support

If radio buttons are used with a label to the left, they have to be used in combination with the label control which points to the assigned radio button. This ensures, that screenreaders are aware of the relationship between the both elements and can read the correct label to the according radio button.

- **Keyboard:** Radio buttons are inserted into the accessibility hierarchy by default.
- **Default Description:** Is provided by the HTMLB rendering engine.
- **Application-specific Description:** Set an additional description using the *setTooltip* method if needed.
- **Label:** Has to be connected to a label control for left-hand labels (use method *setLabelFor* for identifying the corresponding radio button or radio button group).



Control API for Radio Button (radioButton)

A button that a user clicks to set an option. Unlike checkboxes, radio buttons are mutually exclusive - selecting one radio button menu item deselects all others in that group. That is also the reason why you cannot define a radioButton by itself - it always has to be defined within a radioButtonGroup.

- disabled**
 A boolean value that defines if the radioButton is clickable. If the radioButton is disabled it is not selectable. A disabled radioButton has a different background color for the radioButton graphic and if the radioButton is checked the a different color for the button mark.
- encode**
 A boolean value that defines how the radioButton text is interpreted. HTML text formatting commands (e.g. <h1>, <i> etc.) can be used to change the display of the radioButton text. If there are no formatting commands in the radioButton text string, the encode attribute has no effect.

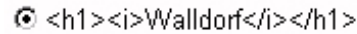
Example

```
text="<h1><i>Important</i></h1>
```



```
encode = "false"
```

the text string is rendered by interpreting the formatting commands.



```
Encode = "true"
```

the formatting commands are displayed and not interpreted

- id**
 Identification name of the radioButton.
- key**
 A string which is assigned to the radioButton when the form is sent to the server. A key string has be defined and must not be empty.
- text**
 Defines the string of text placed right of the radiobutton graphic. If no text should be displayed an empty string (null) can be used. See 'encode' for a formatting example with embedded HTML commands.
- tooltip**
 Defines the hint of the radioButton which is displayed as the mouse cursor passes over the radioButton, or as the mouse button is pressed but not released.

attribute	req.	values	default	case sens.	JSP taglib	classlib
disabled	no	TRUE FALSE	FALSE	yes	disabled="TRUE"	setDisabled(true)
encode	no	TRUE FALSE	TRUE	yes	encode="FALSE"	setEncode(false)
id	yes	String	none	yes	id="GenderInfo"	
key	yes	String	none	yes	key="rb_k1"	setKey("rb_k1")
text	no	String	none	no	text="female"	setText("female")
tooltip	no	String	none	no	tooltip="I am female"	setTooltip("I am female")

Example

A radioButton has to be defined in a radioButtonGroup.

Control API for Radio Button Group (radioButtonGroup)

Places several radiobuttons in tabular form. Only one radiobutton can be on at any given time.

- **columnCount**

Defines the amount of columns in which the radiobuttons are devided.

- **Example**

If the columnCount is set to 3 and you define 7 radiobuttons the result is

Choice 1
 Choice 2
 Choice 3
 Choice 4
 Choice 5
 Choice 6
 Choice 7

- **id**

Identification name of the radioButtonGroup.

- **selection**

Specifies the key of the radioButton that is on in the radioButtonGroup.

attribute	req.	values	default	case sens.	JSP taglib	classlib
columncount	no	Numeric	1	-	columnCount="3"	setColumnCount(3)
id	yes	String	none	yes	id="Genderselect"	
selection	no	String	none	yes	selection="rb_k1"	setSelection("rb_k1")

Example

```

<hbj:radioButtonGroup
  id="Genderselect"
  columnCount="2"
  selection="rb_fem"

  <hbj:radioButton
    id="RBGenderFemale"
    text="female"
    key="rb_fem"
    tooltip="I am female"
    disabled="false"
  />

  <hbj:radioButton
    id="RBGenderMale"
    text="male"
    key="rb_male"
    tooltip="I am male"
    disabled="false"
  />

</hbj:radioButtonGroup>

```

Result

female
 male

Table View

[Usage](#) | [Types](#) | [Design-relevant Attributes](#) | [Related Controls](#)


Table View					
	Course	Course #	Location	Training Facility	Date
<input type="radio"/>	HTML Basics I	50000484	Walldorf	Training Center Walldorf	12/01/2001
<input type="radio"/>	HTML Basics II	50000485	Zürich	Training Center Zürich	12/01/2001
<input type="radio"/>	Web Design Beginners	50000486	Wien	Training Center Wien	12/01/2001
<input type="radio"/>	Web Design Advanced	50000733	Los Angeles	Training Center Los Angeles	12/01/2001
<input checked="" type="radio"/>	Java Basics	50000734	Philadelphia	Training Center Philadelphia	12/01/2001
<input type="radio"/>	Javascript Basics	50000736	Atlanta	Training Center Atlanta	12/01/2001
					1/3

Figure 1: Example of a table view with different column types and an erroneous input field

The table view allows to arrange data - text, images, links, other tables etc. - into rows and columns, that is, in a tabular fashion. Table view rows may be grouped into a header, body and footer section. The table view supplies navigation buttons for scrolling the table. In addition, the table view offers mechanisms for single and multiple selection of rows.



[Top](#)

Usage

Table views are primarily used as data tables for displaying numeric or non-numeric tabular data. Table views can be read-only or used for data entry. Depending on the usage of the table view, different looks and behaviors can be chosen.

General Usage Tips

Tables are relatively complex screen elements that lead developers to squeezing in lots of information. Keep tables small with respect to the number of columns and rows.

For long tables consider effective filtering methods like the shuffler: These tools effect that only a few rows are displayed and that users need not scroll, or need to scroll only a little bit.

Also, consider alternative presentations, such as charts or graphs - they may reveal relevant information faster.

Look

The table view can be presented in three alternative looks:

- Grid and background - either with alternating row colors or with a uniformly colored background
- Transparent, that is, without grid and background

Selection rules for the different table view presentations:

- **Grid and uniform background:** Use this look preferably for entry tables and for numeric display tables. Use the uniformly colored background also for narrow to medium wide display tables.
- **Stripe pattern background:** Use the alternative stripe pattern preferably for data entry tables and for wide display tables.
- **Transparent background:** Use this look preferably for non-numeric display tables, that is, for tables, which display text and/or images.

Table Title and Table Parts

A table view consists of three main parts: a header row, the table view body, and a footer row.

- The table view header contains the table view's **title**. A table view should have a title if the table is not described elsewhere (e.g. by a group or tray title).
- The table view body contains the actual **data**.
- The table view footer is located in the bottom row; it contains the **scroll buttons** to the left and an optional **text**. The footer text may, for example, offer paging information.

The header row as well as the footer row can be hidden. Note that hiding the footer also hides the scroll buttons.

Row and Column Headers

Row and column headers describe data columns and rows. Typically, a table view has only column headers; these describe the different attributes of items that are listed in rows. Row headers can be used, for example, in matrix-like tables, which have both row and column titles.

Cell Content

Table view cells can contain text, images or icons, links, buttons, input fields and dropdown list boxes.

Like stand-alone input fields, input fields in tables can have different attributes, such as *required*, *read-only*, or *error state*.

Row Selection

The table view can be used in two selection modes, if needed:

- single-selection (radio buttons in the first table column)
- multiple-selection (checkboxes in the first table column)

These mode are set by assigning the values SINGLESELECT or MULTISELECT to the attribute **selectionMode**.

In the default mode of the table view (**selectionMode** = NONE) users cannot select any rows. Use this mode if there is no need for users to select rows/items. Use single-selection if users shall only select one row, that is, one item, at a time. Use multiple-selection

if users can select several options or items in parallel.

Scrolling

The table view offers up to six buttons for scroll functions: *First page*, *Page up*, *Line up*, *Line down*, *Page down*, and *Last page*. The scroll buttons are invisible if the footer is hidden.

Note: Scrollbars are currently not supported in table views.

Technical Info: The *Line up/down* buttons appear only if the **selectionMode** attribute has been set to NONE or SINGLESELECT.

Table View Size

Recommendations for the table view size:

Vertical Size

Table views should vertically fit the window or iView they are placed into. As table views can be scrolled through buttons, there should not be no need to use the window's scrollbars.

Make the height of the table view as large as possible with respect to the surrounding container. The larger the table view, the less scrolling is needed (scrolling through buttons is extremely cumbersome..).

Table views should have at least three visible lines - five lines are even better.

Horizontal Size

Table Views should also horizontally fit the window or iView they are placed into. Avoid horizontal scrolling at any price.

Matrix Tables

Matrix tables should have at least 2x2 data cells.

Initial Size and Appearance

Empty tables are not displayed in iViews. In addition, no space is reserved for the table or for a sentence, such as "No entries found".

Exception: There is one exception to this rule: Tables where users can immediately enter data should appear in the intended size and with empty lines.

Note: Do not use the transparent design (see below) in this case.



[Top](#)

Types

Global Table View Look

The attribute **design** defines the global look of the table. It can have one of the following values:

- ALTERNATING: The rows of the table entries are colored alternating.
- STANDARD: The background of Table View is uniformly colored.
- TRANSPARENT: The Table View has no background (grid).

Header			Header		
Column 1	Column 2	Column 3	Column 1	Column 2	Column 3
<input type="checkbox"/>	cell 5	cell 6	<input type="checkbox"/>	cell 5	cell 6
<input type="checkbox"/>	cell 9	cell 9	<input type="checkbox"/>	cell 8	cell 9
		Footer			Footer

Figure 2: Table View with grid and patterned background (left) vs. transparent table

Cell Style

Individual cells can be given different styles as shown in figure 3; most of them are used for numeric values.

Style	Value
STANDARD	Text
NEGATIVE	-2.000.000,35
POSITIVE	2.000.000,35
TOTAL	2.000.000,35
SUBTOTAL	0
SUBTOTAL_LIGHT	0
BADVALUE_DARK	-2.000.000,35
BADVALUE_MEDUIM	-2.000,35
BADVALUE_LIGHT	-1,35
CRITICALVALUE_DARK	-0,35
CRITICALVALUE_MEDIUM	-0,15
CRITICALVALUE_LIGHT	-0,01
GOODVALUE_DARK	2.000.000,35
POSITIVE	2.000,35
GOODVALUE_LIGHT	20,35
GROUP_HIGHLIGHTED	Use to emphasize group data, never together with critical colors
GROUP_HIGHLIGHTED_LIGHT	Use to emphasize group level 2 data, never together with critical colors
KEY_MEDIUM	Use for Key Values Unique Identifiers
GROUP_LEVEL1	Use to group like a tree level 1
GROUP_LEVEL2	Use to group like a tree level 2
GROUP_LEVEL3	Use to group like a tree level 3
MARKED	Use to show that a ROW is selected

Figure 3: Example table showing different cell styles - click [image](#) for larger view

Note: The "GROUP_HIGHLIGHTED" colors should not be used in conjunction with the "CRITICALVALUE" colors.

Cell Content Types

Typically, table cells contain numeric or alphanumeric text. However, there are more cell types available:

- **Text:** Text cell - the text cannot be edited
- **Image:** Cell displays an icon or image
- **Link:** Cell contains a link (reference)
- **Button:** Cell contains a button
- **Input:** The cell can be edited
- **User:** The cell contains a dropdown list box

Different TableColumnType settings					
Text	Image	Link	Button	Input	User
Hello World		Apple Computer Inc.	Click Me	<input type="text" value="Enter something"/>	Midnight ▼
Good morning		IBM	Click Me	<input type="text" value="Enter something"/>	Midnight ▼
Good morning		IBM	Click Me	<input type="text" value="Enter something"/>	Midnight ▼

Figure 4: Example table showing different cell types

[Top](#)

Design-relevant Attributes

The appearance and behavior of tables can be affected by various attributes.

- **Header and Footer:** The header text can be defined (**headerText**) and the header as well as the footer be hidden (Boolean attributes **headerVisible**, **footerVisible**).
Note: The footer must be visible if the table has to be scrolled, that is, if the table contains more lines than are visible.
- **Size, Number of Lines, Initial Appearance:** Another set of attributes determines the width of the table view (**width**), the number of visible rows (**visibleRowCount**), the first visible row (**firstVisibleRow**), and the initial appearance of empty rows (Boolean attribute **fillUpEmptyRows**).
- **Selection Mode:** Further attributes define how the table entries can be selected: single, multiple, or none (attribute **selectionMode**, values SINGLESELECT, MULTISELECT, or NONE).
Note: The selection mode also influences the display of scroll buttons, provided the footer is set to visible.

For details see page [Control API for Table View](#).



[Top](#)

Related Controls

[Tree View](#), [Item List](#), [List Box](#)



[Top](#)

Style	Value
STANDARD	Text
NEGATIVE	-2.000.000,35
POSITIVE	2.000.000,35
TOTAL	2.000.000,35
SUBTOTAL	0
SUBTOTAL_LIGHT	0
BADVALUE_DARK	-2.000.000,35
BADVALUE_MEDIUM	-2.000,35
BADVALUE_LIGHT	-1,35
CRITICALVALUE_DARK	-0,35
CRITICALVALUE_MEDIUM	-0,15
CRITICALVALUE_LIGHT	-0,01
GOODVALUE_DARK	2.000.000,35
POSITIVE	2.000,35
GOODVALUE_LIGHT	20,35
GROUP_HIGHLIGHTED	Use to emphasize group data, never together with critical colors
GROUP_HIGHLIGHTED_LIGHT	Use to emphasize group level 2 data, never together with critical colors
KEY_MEDIUM	Use for Key Values Unique Identifiers
GROUP_LEVEL1	Use to group like a tree level 1
GROUP_LEVEL2	Use to group like a tree level 2
GROUP_LEVEL3	Use to group like a tree level 3
MARKED	Use to show that a ROW is selected

More Info about Table View

[Browser Compatibility](#) | [Editability in Style Editor](#) | [Accessibility – 508 Support](#)

Browser Compatibility

To be provided



[Top](#)

Editability in Style Editor

In the Style Editor, it is possible to modify the following attributes of the table view control:

Group	Style	IE 5 and above	Netscape 4.7
Table Styles	Background Color of Standard Table Cell	x	x
	Background Color of Alternating Table Cell	x	x
	Grid Color	x	
	Cell Height	x	
	Cell Padding	x	
	Background Color of Selected Cell	x	x
	Background Color 1 of Grouping Cell	x	x
	Background Color 2 of Grouping Cell	x	x
	Background Color 3 of Grouping Cell	x	x
Table Icons	Background Position	x	
	Height	x	
	Width	x	
	Padding	x	
	URL to "Top" Icon	x	x
	URL to Inactive "Top" Icon	x	x
	URL to "Page Up" Icon	x	x

	URL to Inactive "Page Up" Icon	x	x
	URL to "Up" Icon	x	x
	URL to Inactive "Up" Icon	x	x
	URL to "Down" Icon	x	x
	URL to Inactive "Down" Icon	x	x
	URL to "Page Down" Icon	x	x
	URL to Inactive "Page Down" Icon	x	x
	URL to "Bottom" Icon	x	x
	URL to Inactive "Bottom" Icon	x	x
Container	Font Color of Container Title	x	x
	Background Color of Container Body	x	x

Table 1: Editable styles for the table view control

For common styles see section [HTMLB Controls and Style Editor](#) in Customer *Branding and Style Editor*.



[Top](#)

Accessibility – 508 Support

- **Keyboard:** Table views are **not** inserted into the accessibility hierarchy by default.
- **Default Description:** Is provided by the HTMLB rendering engine for the title, the navigation buttons, and certain elements inside the table.
- **Application-specific Description:** Set the table title using the *SetHeaderText* method. Set the summary text using the *setSummary* method (should display a tooltip). Note that there is no *setTooltip* method for table views, use *setSummary*, instead.



[Top](#)

Control API for Table View (tableView)

Arrangement of data - text, images, links, other tables etc. - into rows and columns of cells. TableView rows may be grouped into a head, foot and body section. The tableView supplies also navigation buttons which allow browsing thru the table.

TableView example 1						Header
	PLVAR	LOCTP	LOCID	LOCSH	LOCTX	
<input type="checkbox"/>	01	F	50000484	Walldorf	training center Walldorf	
<input type="checkbox"/>	01	F	50000485	Zürich	training center Zürich	
<input type="checkbox"/>	01	F	50000486	Vienna	training center Vienna	
<input type="checkbox"/>	01	F	50000733	Los Angeles	training center Los Angeles	
<input type="checkbox"/>	01	F	50000734	Philadelphia	training center Philadelphia	
						1/16
						Footer

- **design**
Defines the look of the table
 - ALTERNATING
The rows of the table entries are colored alternating.
 - STANDARD
The background of tableView is uniformly colored.
 - TRANSPARENT
The tableView has no background.
- **fillUpEmptyRows**
A boolean value. If set to "TRUE" the tableView has always the height set by the 'visibleRowCount' attribute, regardless of the available table entries. The not available table entries will be filled up with empty lines and according the 'design' attribute.
If set to "FALSE" the tableView height is adjusted to the available table entries.
- **footerVisible**
A boolean value that controls the footer row. If set to "FALSE" the footer row including the navigation buttons is invisible.
- **headerText**
Defines the headline of the tableView.
- **headerVisible**
A boolean value that controls the header line. If set to "FALSE" the header row is invisible.
- **id**
Identification name of the tableView.
- **model**
Defines the model which provides the tableView with data.
- **navigationMode**
Controls the navigation buttons in the footer row.
 - BYPAGE
Four navigation buttons are displayed that allow browsing page up, page down, first and last table entry
 - BYLINE
Two additional buttons are displayed allowing single row up and down.
- **onCellClick**
Defines the action that will be processed when the user clicks on one cell of a column.
- **onHeaderClick**
Defines the action that will be processed when the user clicks on the header of the table.
- **onNavigate**
Defines the action that will be processed when the user clicks on the navigation buttons.

- **onRowSelection**

Defines the action that will be processed when the user clicks on the radiobutton button in the first column. The radiobutton is visible when the 'selectionMode' is set to "SINGLESELECT". The method `com.sapportals.htmlb.event.TableSelectionEvent.getRowIndex()` can be used to retrieve the index of the row that initiated the event.

- **rowCount**

Defines the maximum record that is displayed together with the actual position (e.g. 3/16 - meaning: actual position 3, maximum records 16) on the right side of the footer. If 'rowCount' is not specified the number of records in the model define the value.

- **selectionMode**

Defines how the table entries can be selected

- MULTISELECT

A checkbox is displayed on every row at the first column of the table. According to the nature of the checkbox multiple columns can be selected at a time.

- NONE

No selection possible (no checkbox and no radiobutton).

- SINGLESELECT

A radiobutton is displayed on every row at the first column of the table. According to the nature of the radiobutton one column can be selected at a time. Together with this attribute 'onRowSelection' attribute can be set so that an event is fired, when the user clicks on the radiobutton.

- **visibleFirstRow**

Defines the number of the table entry that is displayed in the first row of the tableView. All subsequent entry are displayed accordingly.

- **visibleRowCount**

Defines the visible rows of the tableView. If 'fillUpEmptyRows' is set to "TRUE", all rows specified with 'visibleRowCount' are displayed. The default for 'visibleRowCount' is the number of table entries supplied by the model.

- **width**

Defines the width of tableView

attribute	req.	values	default	case sens.	JSP taglib	classlib
design	no	STANDARD ALTERNATING TRANSPARENT	STANDARD	yes	design="ALTERNATING"	setDesign (TableViewDesign.ALTERNATING)
fillUpEmptyRows	no	FALSE TRUE	TRUE	yes	fillUpEmptyRows="FALSE"	setFillUpEmptyRows(false)
footerVisible	no	FALSE TRUE	TRUE	yes	footerVisible="FALSE"	setFooterVisible(false)
headerText	no	String	TRUE	no	headerText="SAP training"	setHeaderText("SAP training")
headerVisible	no	FALSE TRUE	TRUE	yes	headerVisible("FALSE")	setHeaderVisible(false)
id	yes	String	none	yes	id="Trainingscenter"	
model	no	String	none	no	model="myBean.model"	setModel((TableViewModel) model)
navigationMode	no	BYPAGE BYLINE	BYPAGE	yes	navigationMode="BYLINE"	setNavigationMode (TableNavigationMode.BYLINE)
rowCount	no	Numeric	defined by model	-	rowCount="5"	setRowCount(5)
selectionMode	no	MULTISELECT NONE SINGLESELECT	MULTISELECT	yes	selectionMode="NONE"	setSelectionMode (TableSelectionMode.NONE)
visibleFirstRow	no	Numeric	1	-	visibleFirstRow="5"	setVisibleFirstRow(5)
visibleRowCount	no	String	defined by model	-	visibleRowCount="20"	setVisibleRowCount(20)
width	no	Unit	none	-	width="500"	setWidth("500")

events	req.	values	default	case sens.	JSP taglib	classlib
onCellClick	no	String	none	yes		setOnCellClick("Pr_Cell")
onHeaderClick	no	String	none	yes		setOnHeaderClick("Pr_Header")
onNavigate	no	String	none	yes	onNavigate="Pr_NavClick"	setOnNavigate("Pr_NavClick")
onRowSelection	no	String	none	yes		setOnRowSelection("Pr_Row")

Example

```

<hbj:tableView
    id="myTableView1"
    model="myTableViewBean.model"
    design="ALTERNATING"
    headerVisible="true"
    footerVisible="true"
    fillUpEmptyRows="true"
    navigationMode="BYLINE"
    selectionMode="MULTISELECT"
    headerText="TableView example 1"
    onNavigate="myOnNavigate"
    visibleFirstRow="1"
    visibleRowCount="5"
    rowCount="16"
    width="500 px"
/>

```

Result

TableView example 1					
	PLVAR	LOCTP	LOCID	LOCSH	LOCTX
<input type="checkbox"/>	01	F	50000484	Walldorf	training center Walldorf
<input type="checkbox"/>	01	F	50000485	Zürich	training center Zürich
<input type="checkbox"/>	01	F	50000486	Vienna	training center Vienna
<input type="checkbox"/>	01	F	50000733	Los Angeles	training center Los Angeles
<input type="checkbox"/>	01	F	50000734	Philadelphia	training center Philadelphia
					1/16

Tabstrip

[Usage](#) | [Design-relevant Attributes](#) | [Related Controls](#)

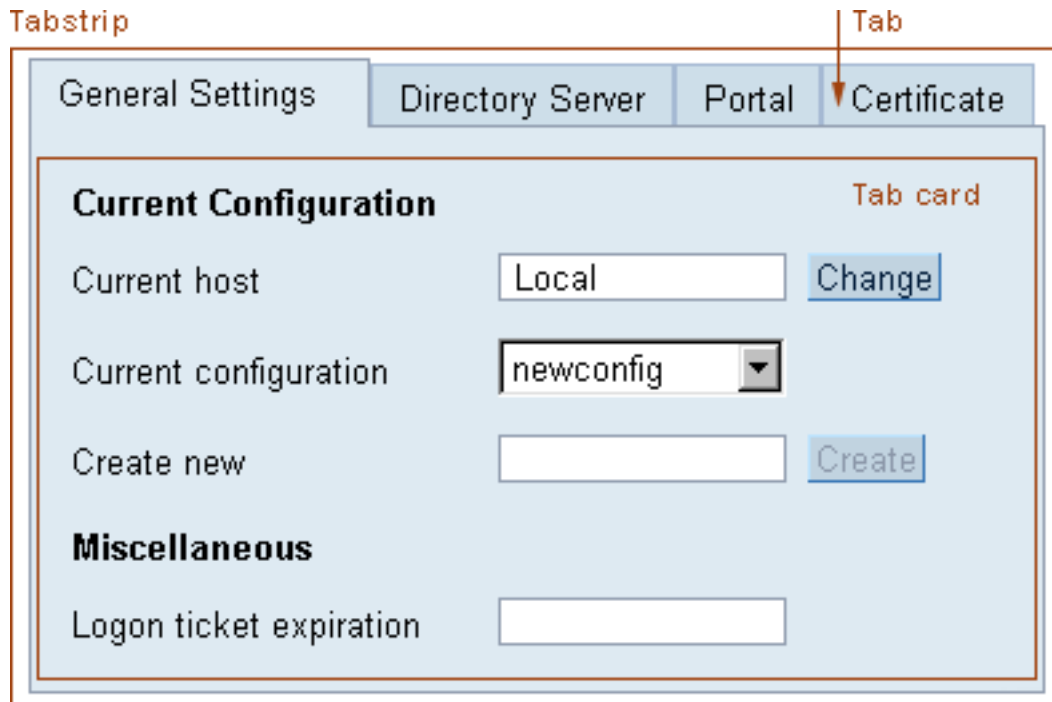


Figure 1: Example of a tabstrip control with an individual tab and the tab card indicated

The tabstrip is a container that allows the user to switch between several views by clicking the tabs. The views appear to share the same space on the screen. The user can access a particular view by clicking its tab.



[Top](#)

Usage

Advantages

- The tabstrip has the advantage that users can see all the alternative views at once. Thus users have a stable context and can navigate easily between the views.
- Tabstrips are also the ideal choice for presenting multiple views of information when the views look very different from one another and a different form of presentation would cause an unstable environment.

Disadvantages

- The disadvantage of tabstrips is that they consume a lot of space compared to other view switching alternatives (e.g. radio buttons or dropdown list box shufflers). For views and alternatives to tabstrips that consume less space, see [Related Controls](#).
- Another disadvantage of tabstrips is that the number of views is limited by the space for the tabs.

Do's

- Tabstrips may contain dynamic information so that users get a quick overview of important data, events or changes within the views.
- The tab card may contain tables and group boxes.

Don'ts

- Avoid using long names in the tab labels and using too many tabs, as this will cause the control to be very wide and may cause problems such as scrolling or excessively wide iViews.
- Tabs may not contain icons.
- Tabstrips indicate to the user that views can be accessed in any order; if this is not the case, then avoid using tabstrips.
- Although space is limited in the tab card, it should not be scrolled.
- Tabstrips may not be nested inside one another!

General Usage Tips

Use tabstrips for selecting views only if other alternatives lead to an unstable interface that might confuse users: Tabstrips appear rather massive, and they take a lot of screen real estate. Furthermore tabstrips should only be used for tasks without a prescribed order of steps as they communicate freedom of choice of interaction sequence.



[Top](#)

Design-relevant Attributes

The appearance and behavior of tabstrips can be affected by various attributes.

- **Height and Width:** Attribute **bodyHeight** sets the vertical size of the tabstrip panel, attribute **width** the overall width of the tabstrip.
- **Horizontal and Vertical Alignment of Tabs:** Use attributes **horizontalAlignment** (values CENTER, CHAR, JUSTIFY, LEFT, RIGHT) and **verticalAlignment** (values BASELINE, BOTTOM, MIDDLE, TOP) to align the tabs.
- **Selected Tab:** Attribute **selection** selects a tab.
- **Tooltip Text:** Use attribute **tooltip** to set the tooltip text for a tabstrip as a whole.

In addition for each view (or tabstrip item) several attributes can be set individually:

- **Height and Width:** Attribute **height** sets the vertical size of the tabstrip view, attribute **width** its width.
- **Tab Text:** Attribute **title** sets the label for the tab.
- **Tooltip Text:** Use attribute **tooltip** to set the tooltip text for a tabstrip view.

For details see page [Control API for Tabstrip](#).



[Top](#)

Related Controls

[Radio Button](#), [Dropdown List Box](#) (shufflers)

More Info about Tabstrip

[Browser Compatibility](#) | [Editability in Style Editor](#) | [Accessibility – 508 Support](#)

Browser Compatibility

Some versions of Netscape Navigator cannot display certain visual nuances of the standard tabstrip control.

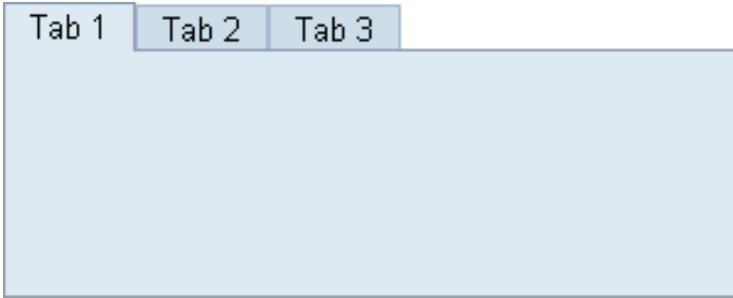


Figure 1: Example of the standard tabstrip

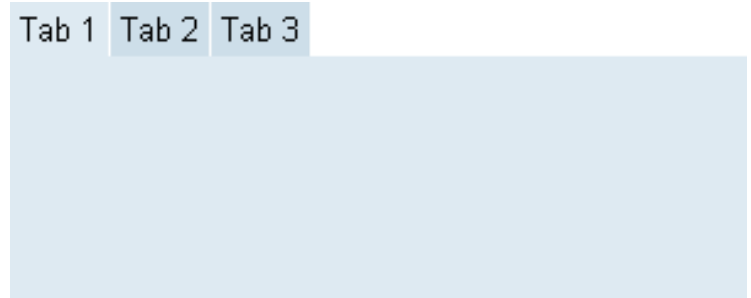


Figure 2: Example of the tabstrip in Netscape Navigator 4

This tabstrip is much less sophisticated visually than the standard tabstrip (figure 1) — the tabs and the tab card have no border, there is a space between the tabs, and the height of the active tab is the same as the height of the inactive tabs.

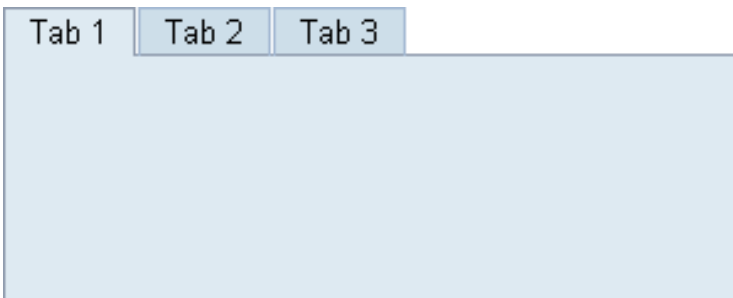


Figure 3: Example of the tabstrip in Netscape Navigator 6.1

This tabstrip is different from the standard tabstrip (figure 1) in that there is a space between the tabs. Additionally, the active tab is the same height as the inactive tabs.

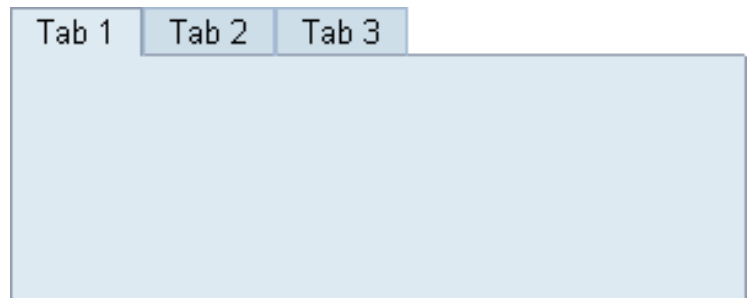


Figure 4: Example of the tabstrip in Netscape Navigator 6.2

This tabstrip is different from the standard tabstrip (figure 1) in that the tabs are all the same height.

Tabstrip Items

Tabstrip items cannot be stored on the Web client. The application has to manage tabstrip items. Therefore, changing the tabs always generates the event `tabSelectionChange`. It is recommended to at least declare the method so that no exception will be thrown if the application is opened in a Netscape Navigator 4 Web client.



[Top](#)

Editability in Style Editor

In the Style Editor, it is possible to change all the background and border colors, as well as the padding and all the text attributes. Here is a list of the styles you can influence:

Group	Style	IE 5 and above	Netscape 4.7
Tabstrip Styles	Background Color of Selected Tab	x	x
	Background Color of Inactive Tab	x	
	Left Border of Inactive Tabs	x	
	Right Border of Inactive Tabs	x	
	Top Border of Inactive Tabs	x	
	Tab Padding	x	
	Tabstrip Border Color		x
	Tab Height		x
Container	Container Border	x	
	Top Border of Container	x	
	Right Border of Container	x	
	Left Border of Container	x	

Table 1: Editable styles for the tabstrip control

For common styles see section [HTMLB Controls and Style Editor](#) in Customer *Branding and Style Editor*.



[Top](#)

Accessibility – 508 Support

- **Keyboard:** Each tab of a tabstrip is inserted into the accessibility hierarchy by default.
- **Default Description:** Is provided by the HTMLB rendering engine for each tab.
- **Application-specific Description:** Set an additional description using the *setTooltip* method for each tab if needed.



[Top](#)

Control API for Tabstrip (tabStrip)

A container that enables the user to switch between several panels -by clicking on the tab - that appear to share the same space on the screen. The user can view a particular panel by clicking its tab. Use `tabStripItem` to define the panel size and title. Use `tabStripItemBody` to define the layout of the `tabStripItem`. Use `tabStripItemHeader` to change the settings of the title (specified through `tabStripItem`).

- **bodyHeight**

Defines the height of panel. The tabs are added on top of panel. The height of the tabs is defined by used text font.

- **horizontalAlignment**

Defines the horizontal alignment of the `tabStripItems`.

- LEFT
Left justifies the content of the cell.
- RIGHT
Right justifies the content of the cell.
- CENTER
Centers the content of the cell.
- CHAR
Aligns text around a specific character. Not supported by all web clients.
- JUSTIFY
Sets text in the cell left and right aligned. Not supported by all web clients.

- **id**

Identification name of the `tabStrip`.

- **selection**

Defines which tab is the active/displayed panel.

- **tooltip**

Defines the hint of the tab which is displayed as the mouse cursor passes over the panel of the `tabStrip`, or as the mouse button is pressed but not released.

- **verticalAlignment**

Defines the vertical alignment of the `tabStripItems`.

- BASELINE
The content of the cell is aligned on the baseline line of the cell (or bottom when no baseline exists).
- BOTTOM
The content of the cell is aligned to the bottom line of the cell.
- MIDDLE
The content of the cell is aligned to the middle of the cell height.
- TOP
The content of the cell is aligned to the top line of the cell.

- **width**

Defines the overall width of the `tabStrip` control.

attribute	req.	values	default	case sens.	JSP taglib	classlib
bodyHeight	no	Unit	100	-	bodyHeight="100"	setBodyHeight("100")
horizontalAlignment	no	CENTER CHAR JUSTIFY LEFT RIGHT	CENTER	yes	horizontalAlignment="LEFT"	setHAlign(CellHAlign.LEFT)
id	yes	String	none	yes	id="TabbedNotebook"	
selection	no	Numeric	1	-	selection="3"	setSelection(3)
tooltip	no	String	none	no	tooltip="select a tab"	setTooltip("select a tab")
verticalAlignment	no	BASELINE BOTTOM MIDDLE TOP	TOP	yes	verticalAlignment="MIDDLE"	setVAlign(CellHAlign.MIDDLE)
width	no	Unit	400	-	width="200"	setWidth("200")

tabStripItem

Specifies the panel size and the tab of a tabStrip. Use `tabStripItemBody` to define the layout of the `tabStripItem`. Use `tabStripItemHeader` to change the settings of the title later on. A `tabStripItem` must have a unique 'id' and 'index' attribute and can call a specific event handler that is activated when this tab is clicked.

- header**
 The tab can have text (set by the 'title' attribute) or any other control. 'header' specifies the component. Common use would be to display icons in the tabs (instead of text).
- height**
 Defines the height of the `tabStripItem`.
- id**
 Identification name of the `tabStripItem`.
- index**
 Defines the index of the `tabStripItem`. The 'selection' attribute of the `tabStrip` refers to the 'index'. The 'index' is mandatory and can be alphanumeric.
- onSelect**
 Defines the action that will be processed when the user clicks on the tab. The string for the event name is not case sensitive - the reference however has to be spelled exactly the same way as the definition of the 'onSelect' event.
 If you do not define a 'onSelect' event the tab can be clicked but no event is generated.
- title**
 Defines the text that is displayed in the tab itself.
- tooltip**
 Defines the hint of the tab which is displayed as the mouse cursor passes over the panel of the `tabStrip`, or as the mouse button is pressed but not released.
- width**
 Defines the overall width of the `tabStripItem`.

attribute	req.	values	default	case sens.	JSP taglib	classlib
header	no	Component	none	no		setHeader(htmlb.Image("Icon.gif", "Texttitle"))
height	no	Unit	100	-	height="80"	setHeight("80")
id	yes	String	none	yes	id="TabbedNotebook"	
index	yes	String	none	-	index="I3"	setIndex("I3")
title	no	String	none	no	title="Settings"	setTitle("Settings")
tooltip	no	String	none	no	tooltip="Desktop settings"	setTooltip("Desktop settings")
width	no	Unit	400	-	width="200"	setWidth("200")

events	req.	values	default	case sens.	JSP taglib	classlib
onSelect	no	String	none	yes	onSelect="proc_tab3"	setOnSelect("proc_tab3")

tabStripItemBody

Specifies the layout of the `tabStripItem`.

tabStripItemHeader

The `tabStripItem` attributes 'title' and 'header' can be altered or set by `tabStripItemHeader` (see following example definition of "tab 4").

Example

```

<hbj:tabStrip
    id="myTabStrip1"
    bodyHeight="100"
    width="200"
    horizontalAlignment="CENTER"
    verticalAlignment="TOP"
    selection="3"
    tooltip="Tooltip for myTabStrip1"
>

    <hbj:tabStripItem
        id="myTabStripItem1"
        index="1"
        height="80"
        width="160"
        onSelect="myTabStripItem1OnSelect"
        title="Tab 1"
        tooltip="My Tooltip for Tab 1"
    >

        <hbj:tabStripItemBody>
            <hbj:textView text="TextView on Tab 1" />
        </hbj:tabStripItemBody>
    </hbj:tabStripItem>

    <hbj:tabStripItem
        id="myTabStripItem2"
        index="2"
        height="80"
        width="160"
        onSelect="myTabStripItem2OnSelect"
        title="Tab 2"
        tooltip="My Tooltip for Tab 2"
    >

        <hbj:tabStripItemBody>
            <hbj:textView text="TextView on Tab 2" />
        </hbj:tabStripItemBody>
    </hbj:tabStripItem>

    <hbj:tabStripItem
        id="myTabStripItem3"
        index="4"
        height="80"
        width="160"
        onSelect="myTabStripItem3OnSelect"
        tooltip="My Tooltip for Tab 3"
    >

        <hbj:tabStripItemBody>
            <%
                myTabStripItem3.setHeader(new com.sapportals.htmlb.Image
                    ("/icons/bottom.gif",
                     "Image not available" ) );
            %>
            <hbj:textView text="TextView on Tab 3" />
        </hbj:tabStripItemBody>
    </hbj:tabStripItem>

    <hbj:tabStripItem
        id="myTabStripItem4"
        index="3"
        height="80"
        width="160"
        onSelect="myTabStripItem4OnSelect"
        tooltip="My Tooltip for Tab 4"
    >

```

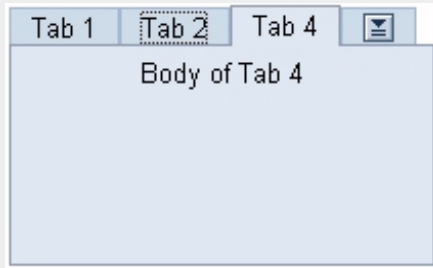
Control API for Tabstrip (tabStrip)

```
<hbj:tabStripItemHeader>
  <hbj:textView text="Tab 4" />
</hbj:tabStripItemHeader>

<hbj:tabStripItemBody>
  <hbj:textView text="Body of Tab 4" />
</hbj:tabStripItemBody>
</hbj:tabStripItem>

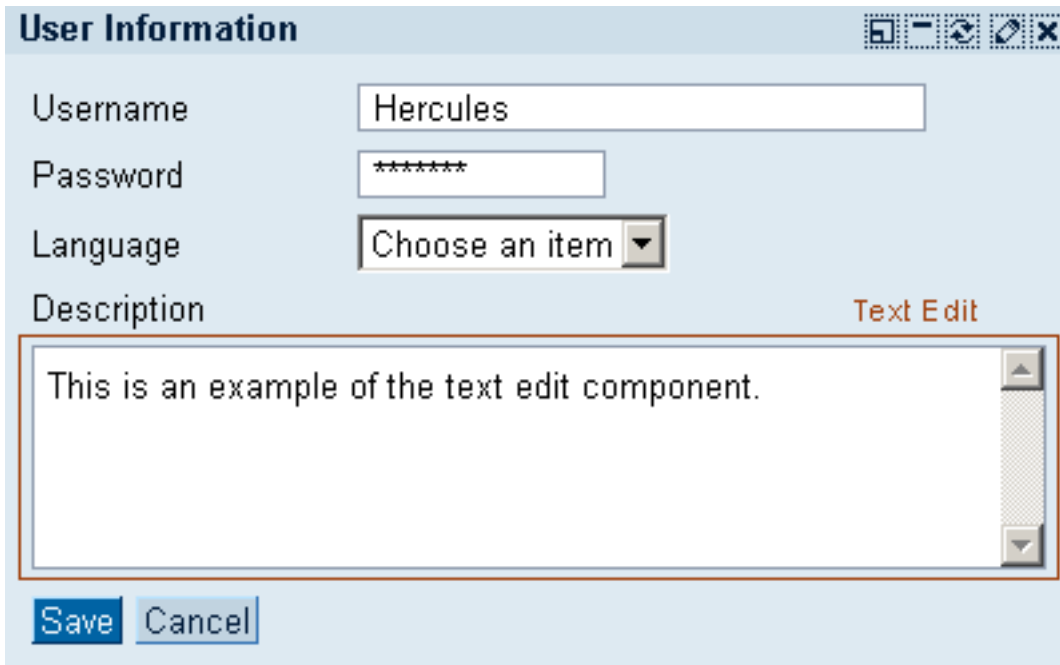
</hbj:tabStrip>
```

Result



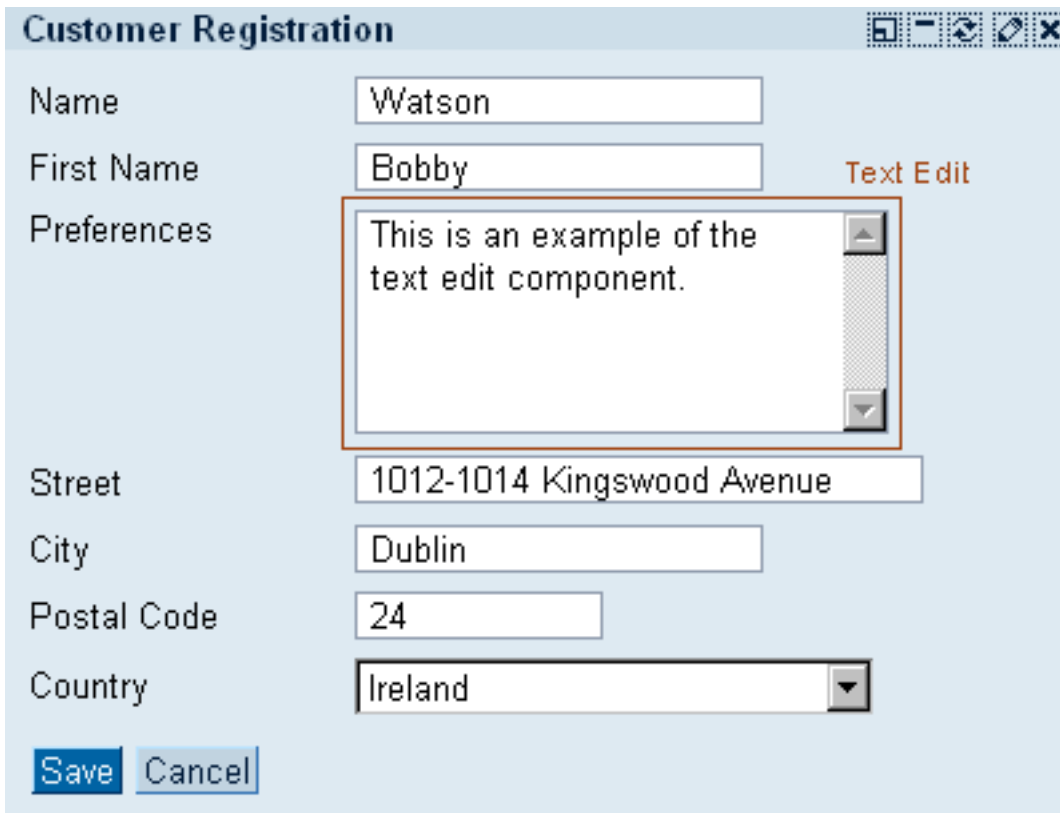
Text Edit

[Usage](#) | [Design-relevant Attributes](#) | [Related Controls](#)



The screenshot shows a dialog box titled "User Information" with a standard window control bar (minimize, maximize, refresh, edit, close). It contains several input fields: "Username" with the text "Hercules", "Password" with "*****", and "Language" with a dropdown menu showing "Choose an item". Below these is a "Description" field, which is a text edit control containing the text "This is an example of the text edit component." and is highlighted with a red border. The label "Text Edit" is positioned to the right of the text area. At the bottom are "Save" and "Cancel" buttons.

Figure 1: Example of text edit control in an iView



The screenshot shows a dialog box titled "Customer Registration" with a standard window control bar. It contains several input fields: "Name" with "Watson", "First Name" with "Bobby", "Street" with "1012-1014 Kingswood Avenue", "City" with "Dublin", "Postal Code" with "24", and "Country" with a dropdown menu showing "Ireland". The "Preferences" field is a text edit control containing the text "This is an example of the text edit component." and is highlighted with a red border. The label "Text Edit" is positioned to the right of the text area. At the bottom are "Save" and "Cancel" buttons.

Figure 2: Example of text edit control in an iView

The text edit control provides an area for multiple-row text editing.



[Top](#)

Usage

Use the text edit control to allow users to edit multiple line of text.

The text is restricted to a single font, size and style unless set with HTML commands. The text edit control has a frame. The size of the frame is defined by the **rows** and **cols** attributes. A vertical scrollbar is displayed permanently. The scrollbar is enabled when the number of text lines exceeds the number of visible lines.

Alignment

There are two possible ways to align the text edit control:

- Below a group of fields with a descriptive label above the text edit control (figure 1)
- In line with other fields within a field group with a label to the left (figure 2)

Place the text edit below the field group if it is used as the main information, whereas the fields above it provide only the context for the information in the text edit control.

Example: A problem description when sending an problem message to a service center

Place the text edit field within the field group if the information is just one piece of information among other information, and the text edit control is used as a freeform multiple-line input field.

Example: A customer enters his or her preferences when registering for an online shop



[Top](#)

Design-relevant Attributes

The text edit control can be influenced through a number of attributes:

- The text and a tooltip text can bet set (attributes **text** and **tooltip**)
- The number of rows and columns can be set (attributes **rows** and **cols**)
- The wrapping behavior can be determined (attribute **wrapping**, values HARD, SOFT and OFF)

For details see the [Control API page for the text edit control](#).

Text Edit



Top

Related Controls

[Text View](#)



Top

More Info about Text Edit

[Browser Compatibility](#) | [Editability in Style Editor](#) | [Accessibility – 508 Support](#)

Browser Compatibility

In Netscape 4.7 the border will be displayed in 3D; the background color is always white; disabled looks like enabled.



[Top](#)

Editability in Style Editor

In the Style Editor, it is possible to modify the following attributes of the text edit control:

Group	Style	IE 5 and above	Netscape 4.7
Text Edit Styles	Padding	x	
Container	Container Border	x	

Table 1: Editable styles for the text edit control

For common styles see section [HTMLB Controls and Style Editor](#) in *Customer Branding and Style Editor*.



[Top](#)

Accessibility – 508 Support

Text edit controls have to be used in combination with the label element which points to the assigned text edit control. This ensures, that screenreaders are aware of the relationship between the both elements and can read the correct label to the according text edit control.

- **Keyboard:** Text edit controls are inserted into the accessibility hierarchy by default.
- **Default Description:** Is provided by the HTMLB rendering engine.
- **Application-specific Description:** Set an additional description using the `setTooltip` method if needed.
- **Label:** Has to be connected to a label control (use method `setLabelFor` for identifying the corresponding text edit control).



[Top](#)

Control API for Text Edit (textEdit)

A multiline region for displaying and editing text. Text in the control is restricted to a single font, size and style unless set with HTML commands.

The textEdit control has a frame. The size of the frame is defined by the 'rows' and 'cols' attribute. A vertical scrollbar is displayed permanently. The scroll bar is enabled when the number of lines exceed the 'rows' attribute.

If the 'wrapping' attribute is not "OFF" the text is wrapped according to the 'cols' attribute - no horizontal scrollbar. If 'wrapping' is set to "OFF" a horizontal scrollbar is activated if the text line length exceeds the width set by the 'cols' attribute.

- **cols**

Defines width of the textEdit control in characters. If the text line exceeds the width defined with the 'cols' attribute and the 'wrapping' attribute if "OFF" a horizontal scrollbar is activated. The scrollbar is appended to the textEdit frame, so that the 'rows' attribute stays unchanged.

If 'wrapping' is set to "HARD" or "SOFT" the text line is wrapped and no horizontal scrollbar is activated.

Be aware that the definition of 'cols' by characters is only an approach and varies by the used character font. A character font with unequal spacing shows different results. A i character fits more often into the textEdit field than a m character.

- **id**

Identification name of the textEdit.

- **text**

Defines the string of text displayed. This text can be edited and/or new text can be added.

- **tooltip**

Defines the hint of the textEdit which is displayed as the mouse cursor passes over the textEdit, or as the mouse button is pressed but not released.

- **rows**

Defines the height of the textEdit control in lines. If the text lines exceed the 'rows' attribute the vertical scrollbar becomes active.

- **wrapping**

Controls the text flow. "HARD" and "SOFT" are passed on to the HTML-Output and control how the carriage return is handled. Web clients handle text wrapping differently. Therefore the following description cannot be guaranteed on all web clients.

- HARD

Wraps the text at the width set by the 'cols' attribute. A carriage control is transmitted at every line break.

No horizontal scrollbar is displayed.

- SOFT

Wraps the text at the width set by the 'cols' attribute. No carriage control is transmitted.

No horizontal scrollbar is displayed.

- OFF

The text line is not wrapped. If the text line length exceeds the width set by the 'cols' attribute a horizontal scrollbar is displayed.

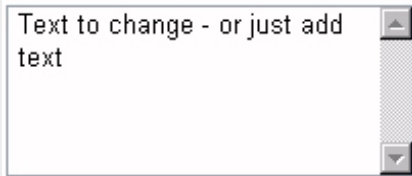
attribute	req.	values	default	case sens.	JSP taglib	classlib
cols	no	Numeric	35	-	cols="20"	setCols(20)
id	yes	String	none	yes	id="Edit_Text"	
text	no	String	none	no	text="editable Text"	setText("editable Text")
tooltip	no	String	none	no	tooltip="PDK Document"	setTooltip("PDK Document")
rows	no	Numeric	5	-	rows="10"	setRows(10)
wrapping	no	HARD SOFT OFF	HARD	yes	wrapping="SOFT"	setWrapping(TextWrapping.SOFT)

Example

```
<hbj:textEdit
  id="Edit_Text
  text="Text to change - or just add text"
  wrapping="SOFT"
  tooltip="Edit and/or add text"
  rows="10"
  cols="30"

/>
```

Result



Text View

[Usage](#) | [Types](#) | [Design-relevant Attributes](#) | [Related Controls](#)

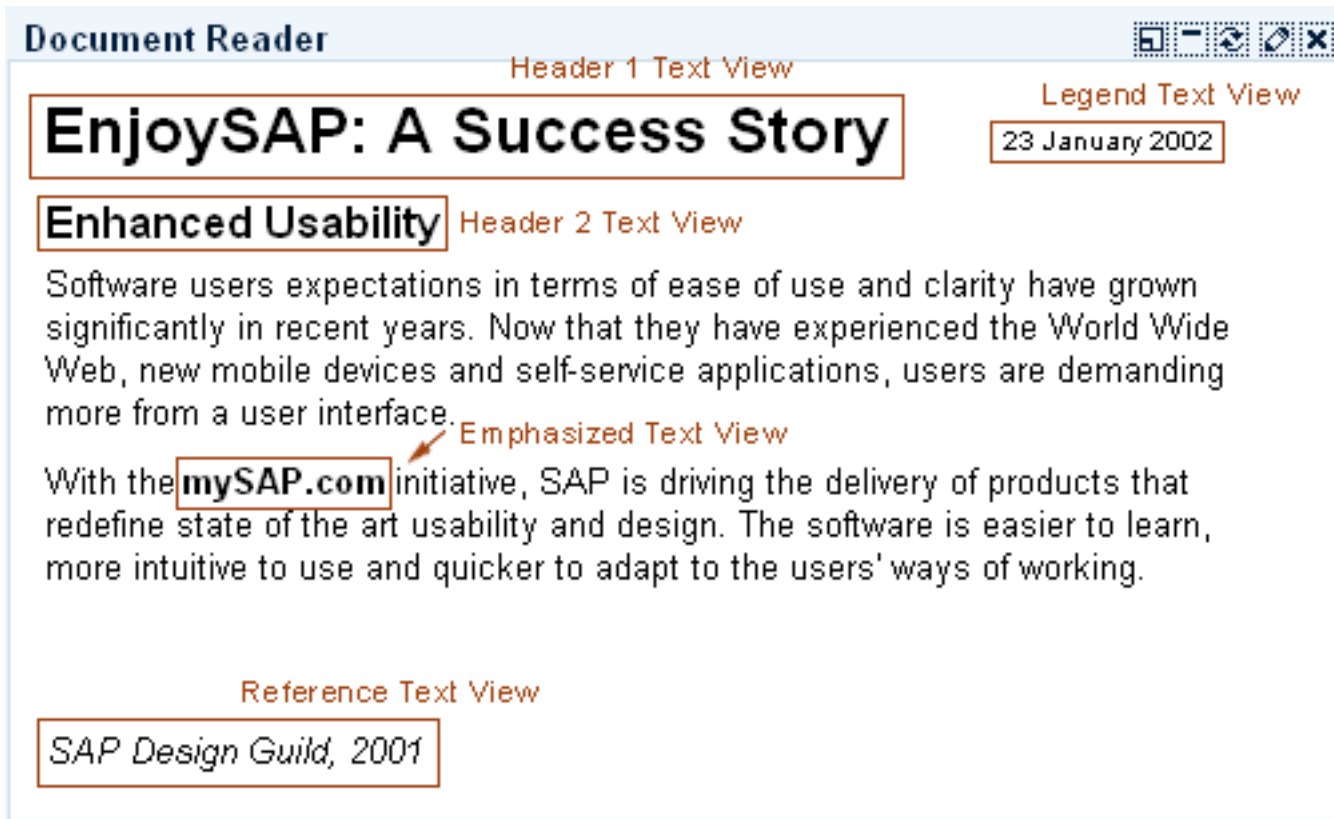


Figure 1: Example of a text view control in an iView.

The text view control offers a multiline region for displaying text; several text attributes can be defined.



[Top](#)

Usage

The text view control is used to display plain text. The text in the control is restricted to a single font, size and style unless set with HTML commands. The text size can be set using different styles (see "Types of Text Views")

Note: The text View control must **not** be used to create a label for input fields; use the [label](#) control instead.

Also note that if you occupy a certain area on the screen for a text view control you should reserve enough space for the translation to other languages. Text in other languages may use up to 30% more space than needed in English.



Types

The text view control is available in several text styles, which are set by the attribute **design** (values STANDARD, EMPHASIZED, REFERENCE, LEGEND, HEADER1, HEADER2, HEADER3). The following description is based on the standard CSS delivered:

Text Style	Use
Standard	used to display body text
Emphasized	used to display emphasized text e.g. phrases, single terms; not to be used for complete text areas; text size "Standard"
<i>Reference</i>	Style for Text Used as a Reference; text size "Standard"
Legend	used to display a legend or small-size help text; text size -2 in comparison to "Standard"
Header 1	used to display a headline or page title; text size +4 in comparison to "Standard"
Header 2	used to display a page subtitle; text size +2
Header 3	used to display a subtitle; text size "Standard"

Table 1: Text styles and their use



Design-relevant Attributes

While the different text types are set using the attribute **design** (values STANDARD, EMPHASIZED, REFERENCE, LEGEND, HEADER1, HEADER2, HEADER3), the appearance of the different text types can be determined by a style sheet (CSS).

The text itself is set by the attribute **text**, an accompanying tooltip text by the attribute **tooltip**.

In addition, alignment (attribute **layout**, values BLOCK, NATIVE, PARAGRAPH), wrapping behavior (Boolean attribute **wrapping**), and width (attribute **width**) can be defined for the text view control.

For details see page [Control API for Text View](#).

Text View



[Top](#)

Related Controls

[Text Edit, Label](#)



[Top](#)

More Info about Text View

[Editability in Style Editor](#) | [Accessibility – 508 Support](#)

Editability in Style Editor

For the text view control, only common styles can be changed. As these styles are important for the text view control, we list them here, too.

Style Group	Style	IE 5 and above	Netscape 4.7
Text Styles	Standard Font Family	x	x
Standard Text	Standard Font Size	x	x
	Standard Font Color	x	x
	Standard Font Style	x	x
	Standard Font Weight	x	x
Non-Standard Text	Font Size for Small Text	x	x
	Font Size for Large Text	x	x
	Font Size for Extra Large Text	x	x
	Font Style for Text Used as a Reference	x	x
	Font Color for Headlines	x	x
	Font Weight for Headlines	x	x
	Font Weight for Emphasized Text	x	x

Table 1: Common styles for the text view control

For an overview of all common styles see section [HTMLB Controls and Style Editor](#) in Customer *Branding and Style Editor*.



[Top](#)

Accessibility – 508 Support

- **Keyboard:** Text view controls are **not** inserted into the accessibility hierarchy by default.
- **Default Description:** Not needed
- **Application-specific Description:** Not needed

Control API for Text View (textView)

A multiline region for displaying text. Text in the control is restricted to a single font, size and style unless set with HTML commands.

- **design**

Defines the appearance of the text. Design can be set to "HEADER1", "EMPHASIZED", "LABEL" etc. . The CSS controls how the different options get rendered. The following description is based on the standard CSS delivered.

- **Emphasized**

- Bold text, text size STANDARD

- **Header 1**

- Bold text, text size +4 in comparison to STANDARD

- **Header 2**

- Bold text, text size +2 in comparison to STANDARD

- **Header 3**

- Bold text, text size STANDARD

- **Standard**

- Text size and attributes STANDARD

- **Legend**

- Text size -2 in comparison to STANDARD

- **Legend**

- Text size -2 in comparison to STANDARD

- **Reference**

- Italic text, text size STANDARD

- **Standard**

- No text attributes and standard text size

- **encode**

A boolean value that defines how the text is interpreted. HTML text formatting commands (e.g. <h1>, <i> etc.) can be used to change the display of the text. If there are no formatting commands in the text string, the encode attribute has no effect.

- **Example**

```
text="<h1><i>Important</i></h1>
```

Important

```
encode = "false"
```

the text string is rendered by interpreting the formatting commands.

```
Encode = "true"
```

```
<h1><i>Important</i></h1>
```

the formatting commands are displayed and not interpreted

- **id**

Identification name of the text.

- **layout**

Defines the alignment of the text.

- BLOCK

- Renders the textView with a <div> HTML tag.

- NATIVE

- Renders the textView with a HTML tag.

- PARAGRAPH

- Renders the textView with a <p> HTML tag

- required**
 Deprecated - the control 'label' should be used instead to label required input fields.
 A boolean value. If set to "true" an asterisks (*) in red color is placed at the end of the text string. This is a common method to indicate that input is required. See also `inputField` and `label`.
- text**
 Defines the string of text displayed. See 'encode' for a formatting example with embedded HTML commands.
- tooltip**
 Defines the hint of the textView which is displayed as the mouse cursor passes over the textView, or as the mouse button is pressed but not released.
- width**
 Defines the width of the textView. The width shows only effect when the 'wrapping' attribute is set to "true". Otherwise the width and layout follows the HTML commands in the text string.
- wrapping**
 A boolean value. If set to "true" the text is word wrapped at the set 'width' or - if no 'width' is set - at the form width.

attribute	req.	values	default	case sens.	JSP taglib	classlib
design	no	EMPHASIZED HEADER1 HEADER2 HEADER3 LABEL LABELSMALL LEGEND REFERENCE STANDARD	STANDARD	yes	design="HEADER1"	setDesign(TextViewDesign.HEADER1)
encode	no	TRUE FALSE	TRUE	yes	encode="FALSE"	setEncode(false)
id	yes	String	none	yes	id="Intro_Text"	
layout	no	BLOCK NATIVE PARAGRAPH	NATIVE	yes	layout="BLOCK"	setLayout(TextViewLayout.BLOCK)
required	no	TRUE FALSE	FALSE	yes	required="TRUE"	setRequired(true)
text	no	String	none	no	text="PDK Introduction"	setText("PDK Introduction")
tooltip	no	String	none	no	tooltip="PDK Document"	setTooltip("PDK Document")
width	no	Unit	100%	no	width="300"	setWidth("300")
wrapping	no	TRUE FALSE	FALSE	yes	wrapping="TRUE"	setWrapping(true)

Example

```
<hbj:textView
  id="Text_ZIP"
  text="ZIP Code"
  design="EMPHASIZED"

/>
```

Result

ZIP Code

Control API for Tray (tray)

Similar to [group](#) the tray allows grouping of controls. The 'tray' allows additional functionality like collapsing/expanding - similar to the behavior of windows on your Microsoft Windows desktop.

Be aware that portal components (components that will run in the SAP portal) are placed in a tray by the portal.

- **design**


The design of the tray can be

- **BORDER**
The tray has a title bar and the panel has a frame that defines the size.
- **BORDERLESS**
The tray has only a title bar.
- **FORM**
The tray has a title bar. The panel is filled with a background color. The color is different from the title background color.
- **TEXT**
The tray has a title bar. The panel is filled with the same background color as the title bar.


- **id**

Identification name of the tray.


- **isCollapsed**

A boolean value that if "true" shows only the title bar. As indicator that the tray is collapsed, the collapsed symbol is displayed.  When clicking on this symbol the 'onExpand' event is fired.


- **onCollapse**

Defines the action that will be processed when the user clicks on the collapse symbol .
If the attribute is set to a `<%=null %>` string or the attribute is omitted the symbol is not displayed in the title bar.


- **onEdit**

Defines the action that will be processed when the user clicks on the collapse symbol .
If the attribute is set to a `<%=null %>` string or the attribute is omitted the symbol is not displayed in the title bar.

- **onExpand**

Defines the action that will be processed when the user clicks on the expand symbol .
This symbol can be actives only when 'isCollapsed' is set to "true". If the attribute is set to a `<%=null %>` string or the attribute is omitted the symbol is not displayed in the title bar.

- **onRemove**

Defines the action that will be processed when the user clicks on the expand symbol .
If the attribute is set to a `<%=null %>` string or the attribute is omitted the symbol is not displayed in the title bar.

- **title**

Defines the string of text placed left aligned in the title bar. If no title should be displayed an empty string (null) can be used. The width of the tray is automatically adjusted to the length of the text when the 'width' attribute is set smaller than the title text width.

- **tooltip**

Defines the hint of the tray which is displayed as the mouse cursor passes over the tray, or as the mouse button is pressed but not released.

- **width**

Defines the width of the tray. The width of the button is automatically adjusted to the length of the 'title'. To see an effect of the 'width' attribute 'width' has to be set higher as the width defined thru the length of the 'title' string. If an empty (null) 'title' string is set no 'title' attribute is defined the width of the tray is set according to the 'width' attribute.

Attribute	req.	values	default	case sens.	JSP taglib	classlib
design	no	BORDER BORDERLESS FORM TEXT	BORDER	yes	design="BORDERLESS"	setDesign(TrayDesign.BORDERLESS)
id	yes	String	none	yes	id="Intro_Text"	
isCollapsed	no	FALSE TRUE	FALSE	yes	isCollapsed="TRUE"	setCollapsed(true)

title	no	String	none	no	title="Headlines"	setTitle("Headlines")
tooltip	no	String	none	no	tooltip="latest news"	setTooltip("latest news")
width	no	Unit	50%	no	width="300"	setWidth("300")

events	req.	values	default	case sens.	JSP taglib	classlib
onCollapse	no	String	none	yes	onCollapse("ev_Col")	setOnCollapse("ev_Col")
onEdit	no	String	none	yes	onEdit("ev_Ed")	setOnEdit("ev_ed")
onExpand	no	String	none	yes	onExpand("ev_Ex")	setOnExpand("ev_Ex")
onRemove	no	String	none	yes	onRemove("ev_Re")	setOnRemove("ev_Re")

trayBody

Defines the items in the tray. A tray can be filled with any any control (checkbox, image, textView etc.).

Example

```
<hbj:tray
  id="HeadlineNews"
  design="BORDER"
  title="latest Headlines"
  tooltip="all the news you need"
  onEdit="ev_hd_edit"
  onRemove="ev_hd_rem"
  width="25%"
  >

  <hbj:trayBody>
    <hbj:textView
      encode="true"
      text="The NASDAQ on an upswing<br>Good news for homeowners" />
  </hbj:trayBody>

</hbj:tray>
```

Result



Tree View

[Usage](#) | [Design-relevant Attributes](#) | [Related Controls](#)



The tree view control is used to display hierarchical data or text. The hierarchy levels may be expanded and collapsed. Every tree node contains a text and an arrow icon that expands and collapses the node. If a node has no child elements in the hierarchy, there is no arrow icon. The node text might also link to a function that displays the connected data.

The first four levels have different colors. From the 5th level on the color stays the same like in the 4th level.

Figure 1: Example of a tree with three levels



[Top](#)

Usage

Trees contain complex information and are cumbersome to use. If possible, do not use trees and consider other alternatives, especially in iViews. Trees with hierarchies more than 2-3 levels deep should be avoided altogether!

How to Avoid Trees

If the number of tree elements is small, hierarchies can be flattened to lists, and the items may follow some other ordering like by alphabet or relevance.

Consider using [dropdown list boxes](#), the shuffler (filter) or [tabstrips](#) in combination with [tables](#) in order to select partial data sets. This leads to a far less complex user interface than large trees that have to be scrolled or paged through.



[Top](#)

Design-relevant Attributes

The tree view control does not have a **width** attribute. To set the width, place the tree inside a [grid layout](#) control.

Use attribute **title** to set a title for the tree.

For tree items, the item text and a corresponding tooltip text can be defined (attributes **text** and **tooltip**).



[Top](#)

Related Controls

[Item List](#), [Dropdown List Box](#), [Table View](#), [Tabstrip](#), [Grid Layout](#) (for sizing)



[Top](#)

More Info about Tree View

[Browser Compatibility](#) | [Editability in Style Editor](#) | [Accessibility – 508 Support](#)

Browser Compatibility

Netscape 4.7 cannot display certain visual nuances of the standard tree control. This tree has no borders and the title height differs from the standard tree (figure 1).

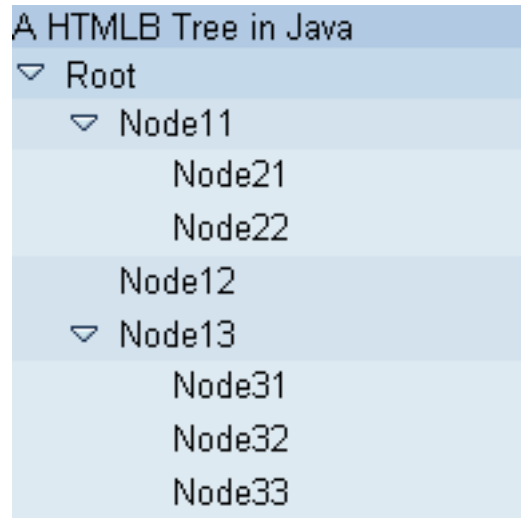
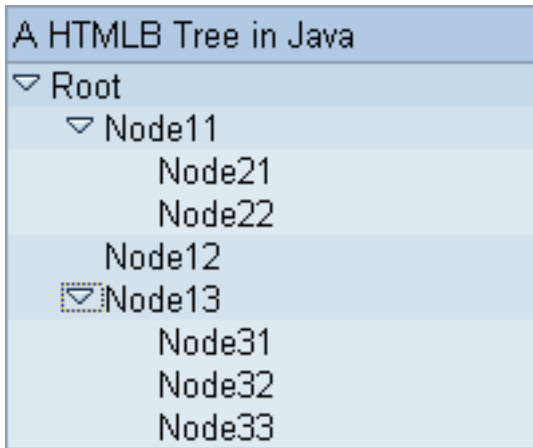


Figure 1: Example of the Standard Tree **Figure 2:** Example of the Tree in Netscape Navigator 4.7

The tree view is always opened completely in Netscape 4.7. It is not possible to expand and collapse nodes locally. The application has to handle these operations (requires server round-trip).



[Top](#)

Editability in Style Editor

In the Style Editor, it is possible to modify the following attributes of the tree view control:

Group	Style	IE 5 and above	Netscape 4.7
Level Background Colors	Background Color of 1st Level	x	x
	Background Color of 2nd Level	x	x
	Background Color of 3rd Level	x	x

	Background Color of 4th Level	x	
Tree Icons	URL to "Expand" Icon	x	
	URL to "Collapse" Icon	x	x
	URL to "Node" Icon	x	x
	Height of Tree Icon	x	
	Width of Tree Icon	x	
Container	Background Color of Container Title	x	x
	Font Color of Container Title	x	x
	Height of Container Title	x	
	Container Border	x	
	Bottom Border of Container	x	
	Cell Padding	x	

Table 1: Editable styles for the tree view control

For common styles see section [HTMLB Controls and Style Editor](#) in Customer *Branding and Style Editor*.



[Top](#)

Accessibility – 508 Support

- **Keyboard:** Each tree node is inserted into the accessibility hierarchy by default.
- **Default Description:** Is provided by the HTMLB rendering engine for each tree node.
- **Application-specific Description:** Set an additional description using the *setTooltip* method for each tree node if needed.



[Top](#)

Control API for Tree View (treeView)



A representation of hierarchical data (for example, directory and file names) as a graphical outline. Clicking expands or collapses elements of the outline. The item in a tree is called `TreeNode`. The nesting depth of `treeNodes` define the hierarchy level. The tree has no width attribute. Place the tree in a grid layout control if a certain width is required.

- **id**
Identification name of the tree.
- **offsetForTreeNode**
Defines the distance in pixel used by the control to indent the sub nodes.
- **rootNode**
Defines the root node of tree. This attribute is used when the tree structure is defined in a bean. The tree node in the bean is created with the command line:
`TreeNode root = new TreeNode("root", "RootNode");`
- **rootNodesVisible**
A Boolean value that indicates if the rootNodesVisible.
- **title**
Defines the headline of the tree.
- **tooltip**
Defines the hint of the tree which is displayed as the mouse cursor passes over the tree, or as the mouse button is pressed but not released.



Attribute	Req.	Values	Default	Case sens.	JSP Taglib	Classlib
id	yes	String	none	yes	id="Intro_Text"	setId("Intro_Text")
offsetForTreeNode	no	Value		-	offsetForTreeNode="20"	setOffsetForTreeNode(20)
rootNode	no*	String	none	no	rootNode="TreeNode"	setRootNode("TreeNode")
rootNodesVisible	no	TRUE FALSE	TRUE	no	rootNodesVisible="TRUE"	setRootNodesVisible(true)
title	no	String	none	no	title="Family tree"	setTitle("Family tree")
tooltip	no	String	none	no	tooltip="Our family"	setTooltip("Our family")

* 'rootNode' is required when the `TreeNode` definition is not made immediately after the tree definition. In this case an error message - indicating that a tree without `treeNodes` is invalid - is generated.

TreeNode

Defines the items in the tree. The level of the tree is defined by the nesting depth. A `TreeNode` with sub nodes has an indicator. The indicator is a triangle that shows if the node is expanded  or collapsed .


- **hoverMenuId**
Defines which hover menu is displayed for this tree node. You can define different trigger methods to display the hover menu. For more details, see [hover menu](#).
- **id**
Identification name of the tree.
- **isOpen**
A Boolean value that indicates if the node is expanded or collapsed. This attribute only has an effect when the node has at least one sub node. If a node is expanded all sub nodes of the node are displayed. Symbols to indicate the node status:

Status	Symbol
Node expanded	
Node collapsed	


- **onNodeClick**

Defines the event handling method that will be processed when the user clicks on the text of the node.

- **onNodeClose**

Defines the event handling method that will be processed when the user clicks on the node symbol. The node has to be initially defined in expanded mode (`isOpen=true`) in order to create the event. The event will then always occur when the user clicks on the  symbol to expand the node. No event occurs when the tree node expands. When the `onNodeClose` attribute is set, the tree node does not collapse on the web client. The event is sent to the server and the application has to take care about the further processing (e.g. define the sub nodes of the tree node and set the tree node collapsed (`isOpen=false`)).

- **onNodeExpand**

Defines the event handling method that will be processed when the user clicks on the node symbol. The node has to be collapsed initially (`isOpen=false`) in order to create the event. The event will then always occur when the user clicks on the  symbol to expand the node. No event occurs when the tree node collapses. When the `onNodeExpand` attribute is set, the tree node does not expand on the web client. The event is sent to the server and the application has to take care about the further processing (e.g. define the sub nodes of the tree node and set the tree node expanded (`isOpen=true`)).



Hint:

The attributes `onNodeExpand` and `onNodeClose` are useful for trees with a lot of entries (transmission problems possible since the page could become pretty big) or if you want to have full control over the tree nodes and build the sub nodes dynamically. The server application has to take care about the modes of the node itself. If you have set an `onNodeExpand` attribute initially, you have to take care about following steps yourself when the event is fired:

- Create the sub nodes.
- Set the node status (`isOpen=true`).
- Set the `onNodeClose` event to receive an event when the user closes the tree node again.

This works vice versa if you have set an `onNodeClose` attribute initially.

- **text**

Defines the string of text displayed for the `treeNode`. HTML commands for text formatting (e.g. `` for bold characters) can be used..

- **tooltip**

Defines the hint of the `treeNode` which is displayed as the mouse cursor passes over the `treeNode`, or as the mouse button is pressed but not released.

Attribute	Req.	Values	Default	Case sens.	JSP Taglib	Classlib
<code>hoverMenuId</code>	no	String	none	yes	<code>hoverMenuId="helpHover1"</code>	<code>setHoverMenu(helpHover1)</code>
<code>id</code>	yes	String	none	yes	<code>id="Intro_Text"</code>	<code>setId("Intro_Text")</code>
<code>isOpen</code>	no	FALSE TRUE	TRUE	yes	<code>isOpen="False"</code>	<code>setIsOpen(false)</code>
<code>text</code>	no	String	none	no	<code>text="Smith"</code>	<code>setTitle("Smith")</code>
<code>tooltip</code>	no	String	none	no	<code>tooltip="1st familyname"</code>	<code>setTooltip("1st familyname")</code>

Events	Req.	Values	Default	Case sens.	JSP Taglib	Classlib
<code>onNodeClick</code>	no	String	none	yes		<code>setOnNodeClick("no1_textclick")</code>
<code>onNodeClose</code>	no	String	none	yes		<code>setOnNodeClose("no1_close")</code>
<code>onNodeExpand</code>	no	String	none	yes		<code>setOnNodeExpand("no1_ex")</code>

Example using the taglib

```

<hbj:tree
  id="S_Tree"
  title="e-enviroment"
  tooltip="enviroment of my computer"
>

  <hbj:treeNode
    id="e_root"
    text="Desk"
    isOpen="true"
    tooltip="My desk"
  >

    <hbj:treeNode
      id="e_comp"
      text="Computer"
      isOpen="true"
    >

      <hbj:treeNode
        id="e_comp_fl"
        text="Floppy"
      />
      <hbj:treeNode
        id="e_comp_hd"
        text="Harddisk"
      />
      <hbj:treeNode
        id="e_comp_dvd"
        text="DVD"
      />
    </hbj:treeNode>

    <hbj:treeNode
      id="e_net"
      text="Network"
      isOpen="true"
      tooltip="Company network"
    >

      <hbj:treeNode
        id="n_lan"
        text="LAN"
        tooltip="Local Area Network"
      />
      <hbj:treeNode
        id="n_wan"
        text="WAN"
        tooltip="Wide Area Network"
      />
      <hbj:treeNode
        id="n_infra"
        text="Infrared"
        tooltip="Infrared connection"
      />
    </hbj:treeNode>

  </hbj:treeNode>
</hbj:tree>

```

Example using the classlib

```

Form form = (Form)this.getForm();
Tree tree = new Tree("S_Tree", "e-enviroment");
tree.setTooltip("enviroment of my computer");

TreeNode root = new TreeNode("e_root", "Desk");
root.setOpen(true);
root.setTooltip("My desk");

// Tags at the second level - the entries are defined with the event "onName"
// which is fired when the event is clicked.
TreeNode name1 = new TreeNode("e_comp", "Computer", root);
name1.setOnNodeClick("onName");
TreeNode name2 = new TreeNode("e_net", "Network", root);
name2.setOnNodeClick("onName");

```

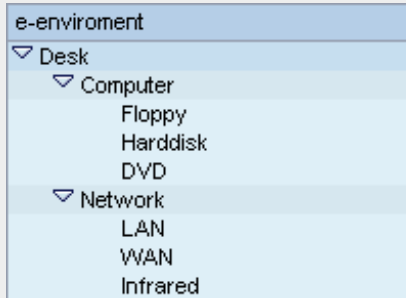
```

TreeNode name11 = new TreeNode("e_comp_fl", "Floppy", name1);
name11.setOnNodeClick("onName");
TreeNode name12 = new TreeNode("e_comp_hd", "Harddisk", name1);
name12.setOnNodeClick("onName");
TreeNode name13 = new TreeNode("e_comp_dvd", "DVD", name1);
name13.setOnNodeClick("onName");
TreeNode name21 = new TreeNode("n_lan", "LAN", name2);
name21.setOnNodeClick("onName");
TreeNode name22 = new TreeNode("n_wan", "WAN", name2);
name22.setOnNodeClick("onName");
TreeNode name23 = new TreeNode("n_infra", "Infrared", name2);
name23.setOnNodeClick("onName");

tree.setRootNode(root);
form.addComponent(tree);

```

Result



Programming Tip

Usually the root node is visible and all sub nodes are displayed on the second level. If you make the root node invisible all sub nodes are displayed on first level.

Example

```

<hbj:tree
  id="S_Tree"
  title="e-environment"
  tooltip="environment of my computer"
  >
  <% S_Tree.setRootNodeIsVisible(false); %>

  <hbj:treeNode
    id="e_root"
    text="Desk"
    isOpen="true"
    tooltip="My desk"
  >
    <hbj:treeNode
      id="e_comp"
      text="Computer"
      isOpen="true"
    >
      <hbj:treeNode
        id="e_comp_fl"
        text="Floppy"
      />
      <hbj:treeNode
        id="e_comp_hd"
        text="Harddisk"
      />
      <hbj:treeNode
        id="e_comp_dvd"
        text="DVD"
      />
    </hbj:treeNode>
    <hbj:treeNode
      id="e_net"
      text="Network"
    >

```

```
        isOpen="true"
        tooltip="Company network"
    >
    <hbj:treeNode
        id="n_lan"
        text="LAN"
        tooltip="Local Area Network"
    />
    <hbj:treeNode
        id="n_wan"
        text="WAN"
        tooltip="Wide Area Network"
    />
    <hbj:treeNode
        id="n_infra"
        text="Infrared"
        tooltip="Infrared connection"
    />
    </hbj:treeNode>
</hbj:treeNode>
</hbj:tree>
```

Result

