



WebObjects Extensions

Component Reference



Apple Computer, Inc.
© 1999 Apple Computer, Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Computer, Inc., except to make a backup copy of any documentation provided on CD-ROM.

The Apple logo is a trademark of Apple Computer, Inc. Use of the “keyboard” Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this book. Apple retains all intellectual property rights associated with the technology described in this book. This book is intended to assist application developers to develop applications only for Apple-labeled or Apple-licensed computers.

Every effort has been made to ensure that the information in this manual is accurate. Apple is not responsible for typographical errors.

Apple Computer, Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Apple, the Apple logo, Macintosh, and WebObjects are trademarks of Apple Computer, Inc., registered in the United States and other countries. Enterprise Objects is a trademark of Apple Computer, Inc.

NeXT, the NeXT logo, OPENSTEP, Enterprise Objects Framework, Objective-C, and WEBSOCKET are trademarks of NeXT Software, Inc.

Adobe, Acrobat, and PostScript are trademarks of Adobe Systems Incorporated or its subsidiaries and may be registered in certain jurisdictions.

Helvetica and Palatino are registered trademarks of Linotype-Hell AG and/or its subsidiaries.

ITC Zapf Dingbats is a registered trademark of International Typeface Corporation.

ORACLE is a registered trademark of Oracle Corporation, Inc.

SYBASE is a registered trademark of Sybase, Inc.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

Windows NT is a trademark of Microsoft Corporation.

All other trademarks mentioned belong to their respective owners.

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this manual, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS MANUAL, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS MANUAL IS SOLD “AS IS,” AND YOU, THE PURCHASER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS MANUAL, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

WebObjects Extensions Component Specifications

Introduction

The WebObjects Extensions are non-synchronizing reusable components defined in the WebObjects Extensions Framework, which is included in every WebObjects Application project. Feel free to examine the source code for this framework available at `Developer/Examples/WebObjects/Source/WOExtensions` relative to the directory in which you installed WebObjects.

For more information about reusable components see "Creating Reusable Components" in the *WebObjects Developer's Guide*.

Here are the WebObjects Extensions defined in the WebObjects Extensions Framework:

[JSAlertPanel](#)

[JSConfirmPanel](#)

[JSImageFlyover](#)

[JSModalWindow](#)

[JSTextFlyover](#)

[JSValidatedField](#)

[WOAnyField](#)

[WOAppleScript](#)

[WOBatchNavigationBar](#)

[WOCheckboxMatrix](#)

[WOCollapsibleComponentContent](#)

[WOCompletionBar](#)

[WODictionaryRepetition](#)

[WOEventDisplayPage](#)

[WOEventSetupPage](#)

[WOIFrame](#)

[WOKeyValueConditional](#)

[WOMetaRefresh](#)

[WOPageRestorationErrorPage](#)

[WORadioButtonMatrix](#)

[WORedirect](#)

[WOSessionCreationErrorPage](#)

[WOSessionRestorationErrorPage](#)

[WOSimpleArrayDisplay](#)

[WOSimpleArrayDisplay2](#)

[WOSortOrder](#)

[WOSortOrderManyKey](#)

[WOStatsPage](#)

[WOTable](#)

[WOTabPanel](#)

[WOTHresholdColoredNumber](#)

[WOTOManyRelationship](#)

[WOTOOneRelationship](#)

How to Use These Specifications

Each component specification that follows is divided into three sections: a synopsis, a description, and a set of bindings. The synopsis is designed to give you ready reference to the element's attributes, showing which ones are mandatory and which ones optional. The description explains the purpose of the element. Finally, the bindings describe in detail each of the component's attributes.

The synopses use several conventions that you should be aware of. For example:

```
WOSubmitButton { action=submitForm; value=aString; [disabled=YES|NO;] [name=aName;] };
```

- *Italic* denotes words that represent something else or that can be varied. For example, *submitForm* represents a method in your script—the exact name of the method is your choice.
- Square brackets ([]) mean that the enclosed attribute or attributes are optional. The name attribute and its value are optional in the synopsis above.
- A vertical bar (|) separates two options that are mutually exclusive, as in disabled=YES|NO where the attribute's value must be either YES or NO.
- The remaining words or characters are to be taken literally (that is, they should be typed as they appear). For example, the action and value attributes are to be taken literally in the synopsis above.

JSAlertPanel

Component Description

This component appears as a hyperlink, which can contain an image, text, and the JSAlertPanel's component content. When the user clicks on it, a dialog box displaying an alert message appears. When the user clicks OK, the hyperlink is executed.

Synopsis

```
JSAlertPanel { action=anAction; | javaScriptFunction=javaScriptCodeString; |  
pageName=aPageName; alertMessage=message; [altTag=aTag;] [filename=imageFileName;]  
[targetWindow=windowName;] [string=aString;] };
```

Bindings

action

The action method performed when the user clicks OK.

javaScriptFunction

Java Script code executed when the user clicks OK.

pageName

The page displayed when the user clicks OK.

alertMessage

The message to display when the user clicks on the hyperlink.

altTag

The HTML alt attribute for the hyperlink's image if an image is specified. Browsers can display this attribute in place of the image.

filename

The name of the image file. Binding this causes the hyperlink to contain an image.

OTHER REFERENCE JSAlertPanel

targetWindow

The name of the window in which the page is displayed when the user clicks OK.

string

A string displayed in the hyperlink.

JSConfirmPanel

Component Description

This component appears as a hyperlink, which can contain an image, text, and the JSConfirmPanel's component content. When the user clicks on it, a dialog box displaying an confirmation message appears. The user can click OK, which executes the hyperlink, or Cancel.

Synopsis

```
JSConfirmPanel { action=anAction; | javascriptFunction=javaScriptCodeString; |  
pageName=aPageName; confirmMessage=message; [altTag=aTag;} [filename=imageFileName;}  
[targetWindow=windowName;} [string=aString];}  };
```

Bindings

action

The action method performed when the user clicks OK.

javascriptFunction

Java Script code executed when the user clicks OK.

pageName

The WComponent displayed when the user clicks OK.

confirmMessage

The message to display when the user clicks on the hyperlink.

altTag

The HTML alt attribute for the hyperlink's image if an image is specified. Browsers can display this attribute in place of the image.

filename

The name of the image file. Binding this causes the hyperlink to display as an image.

OTHER REFERENCE JSConfirmPanel

targetWindow

The name of the window in which the page is displayed when the user clicks OK.

string

A string displayed in the hyperlink

JSImageFlyover

Component Description

The JSImageFlyover appears in the browser as an active image which changes to another image when the mouse pointer hovers over it.

Synopsis

```
JSImageFlyover { action=anAction; | javascriptFunction=aURL; | pageName=pageName;  
selectedImage=selectedImageName; unselectedImage=unselectedImageName;  
[framework=frameworkName;] [targetWindow=windowName;] };
```

Bindings

action

The action method performed when the user clicks the image.

javascriptFunction

Java Script code executed when the user clicks the image.

pageName

The WComponent displayed when the user clicks the image.

selectedImage

The image displayed when the mouse pointer is hovering over it.

unselectedImage

The image displayed when the mouse pointer is not hovering over it.

framework

The framework containing the image. Defaults to the application.

OTHER REFERENCE JSImageFlyover

targetWindow

The name of the window in which the new page is displayed when the user clicks on the image.

JSModalWindow

Component Description

This component appears as a hyperlink in the browser. When the user clicks it, the result displays in a modal-like window.

Synopsis

```
JSModalWindow { action=anAction; | pageName=pageName; height=height; width=width;  
windowName=aString; [isResizable="YES"|"NO";] [showLocation="YES"|"NO";]  
[showMenuBar="YES"|"NO";] [showScrollbars="YES"|"NO";] [showStatus="YES"|"NO";]  
[showToolBar="YES"|"NO";] };
```

Bindings

action

Action method invoked when the user clicks the hyperlink that supplies the content for the modal-like window.

pageName

The WComponent displayed when the user clicks the hyperlink that appears in the modal-like window.

height

Height, in pixels, of the window.

width

Width, in pixels, of the window.

windowName

Specifies the title for the window.

OTHER REFERENCE JSModalWindow

isResizable

Controls whether the window can be resized.

showLocation

Controls whether the window displays the URL.

showMenubar

Controls whether the window has a menu bar.

showScrollbars

Controls whether the window has scroll bars.

showStatus

Controls whether the window has a status display.

showToolbar

Controls whether the window has a tool bar.

JSTextFlyover

Component Description

The `JSTextFlyover` appears in the browser as a hyperlink containing text that changes color when the mouse pointer hovers over it.

Synopsis

```
JSTextFlyover { [action=anAction; | pageName=pageName;] string=aString;  
selectedColor=selectedHexString; unselectedColor=unselectedHexString;  
[targetWindow=windowName;] };
```

Bindings

`action`

Action method invoked when the user clicks the hyperlink.

`pageName`

Name of the `WOComponent` displayed when the user clicks the hyperlink.

`selectedColor`

The color of the text when the mouse pointer is hovering over it.

`unselectedColor`

The color of the text when the mouse pointer is not hovering over it.

`targetWindow`

The name of the window in which the page is displayed when the user clicks on the hyperlink.

JSValidatedField

Component Description

This component is similar to a WOTextField and needs to be placed within a WOForm. When the user clicks on one of the form's submit buttons, the text in the field is tested according to the following rules:

- If `inputIsRequired` is set to YES, the text must not be the empty string.
- If `requiredText` is not the empty string, the text must contain it.

If the text conforms to these rules, JSValidatedField submits the form. Otherwise it displays a message and does not submit the form.

Synopsis

```
JSValidatedField { inputText=aVariable; errorMessage=aString; formName=aString;  
[fieldSize=fieldSize;] [inputIsRequired="YES"|"NO";] [requiredText=requiredText;] };
```

Bindings

`inputText`

The variable into which the entered data is stored.

`errorMessage`

The message to display if the validation fails.

`formName`

The name of the form that contains the JSValidatedField.

`fieldSize`

Specifies the width of the text field.

OTHER REFERENCE JSValidatedField

inputIsRequired

If YES, this field must be nonempty before JSValidate Field allows the user to submit the form.

requiredText

A string that the entered text must contain before JSValidateField allows the user to submit the form.

WOAnyField

Component Description

The WOAnyField component provides an interface for the user to qualify a WODisplayGroup's enterprise objects based on a single attribute of the objects. The user can choose the attribute, an operator (less than, greater than, equal to, or not equal to), and a value for the attribute. The attribute can be an attribute of the displayed objects, or an attribute of another object obtained by traversing a relationship. The component sets a WODisplayGroup's `queryMatch` dictionary according to the user's choices but does not redisplay the objects.

This component must be embedded within a WOForm. If you want to redisplay the objects with the new qualifier, bind the `value` attribute of the WOForm's submit button to the WODisplayGroup's `qualifyDataSource` method.

where is

Synopsis

```
WOAnyField { displayGroup=aDisplayGroup; displayKey=aString;] [formatter=formatterObj;  
key=aString; keyList=anArray; [relationshipKey=aString;] [selectedKey=aString;]  
sourceEntityName=aString; [value=anObject;] };
```

Bindings

displayGroup

The display group for which the WOAnyField component sets the `queryMatch` dictionary.

OTHER REFERENCE WOAnyField

displayKey

The string corresponding to `key` that the WOAnyField component displays. Can be bound to the same String (NSString in Objective-C) as `key`.

formatter

An instance of an NSStringFormatter subclass for the attribute corresponding to `key` used to format the attribute's values for display as strings and format user entered strings back into the attribute's values. The `formatter` attribute should specify a variable containing (or method returning) a preconfigured formatter object.

key

The key corresponding to the current iteration through the key list.

keyList

An array containing keys corresponding to the attributes with which the user can qualify the displayed objects.

relationshipKey

The key corresponding to one of the source entity's relationships. If this binding is specified, the WOAnyField component builds the `queryMatch` dictionary based on attributes from the destination object. This binding allows you to query with a single level of indirection. For example, you can query for all movies produced by studios starting with 'P'. If this binding is omitted, the source entity's attributes are used.

selectedKey

The key that is selected when the WOAnyField component is first displayed.

sourceEntityName

The name of entity displayed by the display group.

value

The value that appears in the value text field when the WOAnyField component is first displayed.

WOAppleScript

Component Description

The WOAppleScript component provides the ability to include client-side AppleScript in web pages, allowing WebObjects to control Macintosh computers that have the appropriate browser plug-in.

Synopsis

```
WOAppleScript { scripttext=aString; [controller=aString;] [height=aNumber;]  
[width=aNumber;] [scriptcomment=aString;] [scripttitle=aString;] };
```

Bindings

scripttext

A string identifying the AppleScript to be executed on the client. This attribute is required.

controller

A string containing either "True" or "False" that determines whether or not the controller panel should appear.

height

The height of the AppleScript component in the client browser.

width

The width of the AppleScript component in the client browser.

scriptcomment

A comment for the AppleScript plug-in.

scripttitle

A title for the AppleScript.

OTHER REFERENCE WOAppleScript

WOBatchNavigationBar

Component Description

The WOBatchNavigationBar component provides the ability to navigate through a WODisplayGroup in batches. The component has buttons that allow the user to navigate to the next batch and to the previous batch. It also displays the number of batches, which batch the user is currently viewing, and how many objects are in each batch.



Synopsis

```
WOBatchNavigationBar { displayGroup=aDisplayGroup; sortKeyList=anArray;  
objectName=aString; [width=aNumber;] [textColor=hexString;] [border=aString;]  
[bgcolor=hexString;] };
```

Bindings

displayGroup

The display group that the WONavigation bar displays in batches.

sortKeyList

Array of keys for the attributes by which the displayed objects can be sorted. The user chooses one of these attributes and a sort ordering (ascending or descending), and the navigation bar displays the batches accordingly.

OTHER REFERENCE WOBatchNavigationBar

objectName

The name of the object displayed by the display group. The navigation bar displays this name.

width

Width of the navigation bar. This attribute is passed to the HTML TABLE that makes up the navigation bar.

textColor

Color of the text within the bar.

border

Width of the navigation bar's border. This attribute is passed to the TABLE that makes up the navigation bar.

bgcolor

Background color of the navigation bar. This attribute is passed to the TABLE that makes up the navigation bar.

WOCheckboxMatrix

Component Description

The WOCheckboxMatrix component displays a multi-column array of checkboxes based on a list of objects and allows the user to select any combination of these objects. This component displays its content (everything between the `<WEBOBJECT...>` and `</WEBOBJECT...>` tags in the template file) for each of the items in `list` in the same order as [WOTable](#). This component must be embedded within a WOForm.

Synopsis

```
WOCheckBoxMatrix { list=anArray; item=anObject; selections=anArray; maxColumns=aNumber;
};
```

Bindings

`list`

Array of objects from which the checkboxes derive their values. For example, the array could be named `movieArray` and contain `Movie` objects.

`item`

Identifier for the elements of the list. This attribute is updated for each iteration through `list`. For example, `currentMovie` could represent an object in `movieArray`.

`selections`

An array of objects the user chooses from the list. This attribute is updated when the user submits the form containing the WOCheckboxMatrix. For the movie example, `selections` array would hold `Movie` objects.

`maxColumns`

The number of columns of checkboxes displayed.

OTHER REFERENCE WOCheckboxMatrix

WOCollapsibleComponentContent

Component Description

The WOCollapsibleComponentContent component allows the user to display (expand) or hide (collapse) the content (that is, everything between the `<WEBOBJECT...>` and `</WEBOBJECT...>` tags in the template file). When the content is collapsed, it is replaced with an image, which defaults to . Clicking the image expands the content. When the content is expanded, it is displayed below an image, which defaults to . Clicking the image collapses the content.

 [Click here for the Gettysburg Address](#)

 [The Gettysburg Address](#)

Four score and seven years ago our fathers
brought forth on this continent a new
nation conceived in liberty and dedicated
to the proposition...

This component can be embedded within a WOForm. When the WOCollapsibleComponentContent contains form fields that are bound to an enterprise object's attributes, the HTML element the user clicks to collapse the content must be a submit button. This ensures that the values the user types into these fields get stored in the enterprise object upon collapse. If the HTML element the user clicks is a hyperlink, the enterprise object's attributes are not updated when the user collapses the content and the form field values will not reappear when the user expands the content again. The `submitActionName` binding determines the whether the HTML element that collapses the content is a hyperlink or a submit button.

Synopsis

```
WOCollapsibleComponentContent { [condition=aBoolean;] [visibility=aBoolean;]  
[openedImageFileName=aPath;] [closedImageFileName=aPath;] [framework=aString;]  
[openedLabel=aString;] [closedLabel=aString;] [submitActionName=anActionName; ];
```

Bindings

condition

A flag to indicate whether the component is initially expanded (YES) or collapsed (NO).

visibility

The current state of expansion. This attributed is updated each time the component is expanded or collapsed.

openedImageFileName

The file name for an active image displayed above the expanded content. The user clicks this image to collapse the content. WebObjects assumes the image resides in the `WebServerResources` directory of the framework containing the image (see the `framework` attribute.) If this attribute is not defined, `WOCollapsibleComponentContent` displays ▼.

closedImageFileName

The filename for an active image displayed when the content is collapsed. The user clicks this image to expand the content. WebObjects assumes the image resides in the `WebServerResources` directory of the framework containing the image (see the `framework` attribute.) If this attribute is not defined, `WOCollapsibleComponentContent` displays ▶.

framework

The `framework` `openedImageFileName` and `closedImageFileName` come from. Defaults to the application.

openedLabel

A label displayed above the expanded content.

closedLabel

A label displayed next to the collapsed content.

submitActionName

The name of an action that is called upon collapse. This attribute functions only when the `WOCollapsibleComponentContent` component is embedded within a `WOForm`. If `submitActionName` is not defined or is set to `null`, the HTML element that expands

OTHER REFERENCE WO CollapsibleComponentContent

the content is a hyperlink. Otherwise the it is a submit button. If you want to expand the content with a submit button, but don't want it to invoke an action, set this attribute to the empty string ("").

WOCompletionBar

Component Description

The `WOCompletionBar` component displays a progress bar on a page. You might use `WOCompletionBar` in the status page of your long-running action (see the `WOLongResponsePage` Class).

Synopsis

```
WOCompletionBar { valueMin=aNumber; valueMax=aNumber; [value=aNumber;]  
[numberformat=numberFormatString;] [progressColor=hexString;]  
[backgroundColor=hexString;] [width=aNumber;] [border=aString;] [align=aString;] };
```

Bindings

`valueMin`

Minimum value for the bar or value at which the bar begins.

`valueMax`

Maximum value for the bar or value at which the bar ends.

`value`

The current amount completed. The bar sizes to this number and displays the number inside itself. The value must be between `valueMin` and `valueMax`.

`numberformat`

A format string that specifies how `value` should be formatted as a number. If a number format is used, `value` must be assigned an `NSNumber` object. If the value can't be interpreted according to the format you specify, the value is set to `nil`. See the `NSNumberFormatter` class specification for a description of the number format syntax.

OTHER REFERENCE W O Completion Bar

progressColor

Color for showing the value. That is, this color shows the amount completed.

backgroundColor

Color for the uncompleted portion.

width

Width of the bar. This attribute is passed directly to the HTML `TABLE` that makes up the bar.

border

Thickness of the bar's border in pixels. This attribute is passed directly to the HTML `TABLE` that makes up the bar.

align

Alignment of the displayed number within the completed portion of the bar. This attribute is passed directly to the HTML `FONT` specification for the number.

WODictionaryRepetition

Component Description

A WODictionaryRepetition is a container element that repeats its contents (that is, everything between the `<WEBOBJECT...>` and `</WEBOBJECT...>` tags in the template file) for each entry in a dictionary. You can use a WODictionaryRepetition to create dynamically generated unordered lists or banks of check boxes or radio buttons.

Synopsis

```
WODictionaryRepetition {dictionary=aDictionary; item=anObject; key=aString; };
```

Bindings

dictionary	Dictionary of key/value pairs through which the WODictionaryRepetition iterates.
key	Current key in the dictionary. This attribute's value is updated with each iteration.
item	Current object corresponding to the key in the dictionary. This attribute's value is updated with each iteration.

OTHER REFERENCE WODictionaryRepetition

WOEventDisplayPage

Component Description

The WOEventDisplayPage component is a page that WebObjects displays in the browser when the user accesses the WebObjects profiling tool. This component is a complete page and is not designed to be embedded within another component. You can modify this page if you want to customize the profiling tool's display. See *Deploying WebObjects Applications* for more information.

WOEventSetupPage

Component Description

The WOEventSetupPage component is a page that WebObjects displays in the browser when the user configures the WebObjects profiling tool. This component is a complete page and is not designed to be embedded within another component. You can modify this page if you want to customize the profiling tool's setup page. See *Deploying WebObjects Applications* for more information.

WOIFrame

Component Description

The WOIFrame component inserts an IFRAME tag into your page. This tag is a container to create an in-line or floating frame: a frame in which the contents of another HTML document can be seen. The difference between an IFRAME and a normal frame is that the floating frame can be seen inside a document and is treated as a part of the document. This means that when you scroll through the page the frame will scroll with it. IFRAME tags are supported by Microsoft's Internet Explorer browser. If you want to insert a FRAME tags, as used by Netscape's browser, use the WOFrame dynamic element documented in the *Dynamic Elements Reference*.

The HTML content displayed within the WOIFrame can come from a method, a URL, or a WebObjects page.

Synopsis

```
WOIFrame { value=aMethod; | src=aURL; | pageName=aString; [frameborder=aString];
[height=aString]; [marginheight=aString]; [marginwidth=aString]; [name=aString];
scrolling=aString; width=aString; };
```

Bindings

value

Method that supplies the content for this frame.

src

External source that supplies the content for this frame.

pageName

Name of WebObjects page that supplies the content for this frame.

OTHER REFERENCE WOIFrame

frameborder

Specifies whether or not a border should be displayed for the frame. This attribute is passed through to the `IFRAME` HTML tag.

height

Specifies the height of the frame to the browser, either as a number of pixels or as a percentage of the current screen height.

marginheight

Controls the vertical margins for the frame (margins are specified in pixels). This attribute is passed through to the `IFRAME` HTML tag.

marginwidth

Controls the horizontal margins for the frame (margins are specified in pixels). This attribute is passed through to the `IFRAME` HTML tag.

name

Assigns a name to the frame so it can be targeted by links in other documents or, more commonly, from other frames in the same document. This attribute is passed through to the `IFRAME` HTML tag.

scrolling

Specifies whether or not the frame should have scrollbar. This attribute is passed through to the `IFRAME` HTML tag.

width

Specifies the width of the frame to the browser, either as a number of pixels or as a percentage of the current screen width. This attribute is passed through to the `IFRAME` HTML tag.

WOKeyValueConditional

Component Description

A WOKeyValueConditional component displays its contents (that is, everything between the `<WEBOBJECT...>` and `</WEBOBJECT...>` tags in the template file) if the result of the parent component's `valueForKey` method matches a particular value. Raises an exception if the parent component's dictionary does not contain `key`. This component is very similar to the WOConditional dynamic element and is usually used in conjunction with the [WOTabPanel](#) component.

Synopsis

```
WOKeyValueConditional { key=aString; value=anObject; };
```

Bindings

`key`

The key whose value is compared.

`value`

The value that must match the result of the parent's `valueForKey` method.

OTHER REFERENCE WOKeyValueConditional

WOMetaRefresh

Component Description

The WOMetaRefresh component inserts an HTML `<META REFRESH=...>` tag into your page, which instructs the user's browser to display a new page after a specified time interval. You can use the WOMetaRefresh component to create pages that refresh themselves regularly (for example, a stock price page), or display a message and jump to a new page (for example, a page that jumps to a website from an outdated URL).

Note that not all browsers support page refreshing.

Synopsis

```
WOMetaRefresh { seconds=aNumber; pageName=aString; | action=aMethod; };
```

Bindings

seconds

Number of seconds before the page is refreshed.

pageName

Component to navigate to after the page is refreshed.

action

Action method to invoke after the page is refreshed.

WOPageRestorationErrorPage

Component Description

The WOPageRestorationErrorPage component is displayed in the browser when the user has backtracked too far. This component is a complete page and is not designed to be embedded within another component. You can modify this page if you want to customize the error message.

WORadioButtonMatrix

Component Description

The WORadioButtonMatrix displays a multi-column array of radio buttons based on a list of objects and allows the user to select one of these objects. This component displays its content (everything between the `<WEBOBJECT...>` and `</WEBOBJECT...>` tags in the template file) for each of the items in `list` from left to right, and wraps around to the next line when the number of columns reaches `maxColumns`. This component must be embedded within a WOForm.

Synopsis

```
WORadioButtonMatrix { list=anArray; item=anObject; selection=theSelection;
maxColumns=aNumber; };
```

Bindings

`list`

Array of objects from which the radio buttons derive their values. For example, the array could be named `movieArray` and contain `Movie` objects.

`item`

Identifier for the elements of the list. This attribute is updated for each iteration through `list`. For example, `currentMovie` could represent an object in `movieArray`.

`selection`

Object that the user chooses from the selection list. This attribute is updated when the user submits the form containing the WORadioButtonMatrix. For the movie example, `selection` would be a `Movie` object.

`maxColumns`

The number of columns of radio buttons displayed.

WOREdirect

Component Description

WOREdirect immediately sends the browser to another URL.

Synopsis

```
WOREdirect { setURL= aURL; };
```

Bindings

setURL

URL that the browser is redirected to.

WOSessionCreationErrorPage

Component Description

The `WOSessionCreationErrorPage` component is displayed in the user's browser when it cannot create a new session for the user. This component is a complete page and is not designed to be embedded within another component. You can modify this page if you want to customize the error message.

WOSessionRestorationErrorPage

Component Description

The `WOSessionRestorationErrorPage` component is displayed in the browser when the user tries to access a session that has timed out. This component is a complete page and is not designed to be embedded within another component. You can modify this page if you want to customize the error message.

WOSimpleArrayDisplay

Component Description

The `WOSimpleArrayDisplay` component displays some or all of an array's objects in a single-column table. If the `WOSimpleArrayDisplay` component does not display all of the objects in the array, it displays a hyperlink with the text `More... (x items)`, which can be linked to a page that displays all of the objects.

first
second
third
fourth
fifth
More... (5 items)

Synopsis

```
WOSimpleArrayDisplay {list=anArray; itemDisplayKey=aString; [numberToDisplay=aNumber];  
listAction=aMethod; [listActionString=aString]; };
```

Bindings

`list`

Array of objects to display.

`itemDisplayKey`

The key for the displayed attribute of the array's objects. For example, `roleName`. If the objects are strings, use `description`.

OTHER REFERENCE WOSimpleArrayDisplay

numberToDisplay

The maximum number of objects to be displayed (defaults to 5.) If the number of objects exceeds this number, a hyperlink is displayed.

listAction

The action method that is called when the user clicks the hyperlink that the component displays when the number of objects exceeds `numberToDisplay`.

listActionString

This string is appended to the `More (x items)` hyperlink text that the component displays when the number of objects exceeds `numberToDisplay`.

WOSimpleArrayDisplay2

Component Description

The WOSimpleArrayDisplay2 component displays some or all of an array's objects in a single-column table. Each object has a hyperlink that can be used to jump to an edit page for the object. If the WOSimpleArrayDisplay2 component does not display all of the objects in the array, it displays an inspect image hyperlink, which can be linked to a page that displays all of the objects.



Synopsis

```
WOSimpleArrayDisplay2 {list=anArray; [itemDisplayKey=aString;]
[numberToDisplay=aNumber;] listAction=aMethod; [listTarget=aString;] item=anObject;
[itemTarget=aString;] displayItemAction=aMethod; };
```

Bindings

list

Array of objects to display.

itemDisplayKey

The key for the displayed attribute of the array's objects. For example, `roleName`. Use `description` if the objects are strings.

OTHER REFERENCE WOSimpleArrayDisplay2

numberToDisplay

The maximum number of objects to be displayed (defaults to 5.) If the number of objects exceeds this number, a hyperlink is displayed.

listAction

The action method invoked when the user clicks the hyperlink that the component displays when the number of objects exceeds `numberToDisplay`.

item

The selected object. This attribute is updated when the user clicks one of the object hyperlinks.

listTarget

The target frame for the hyperlink that the component displays when the number of objects exceeds `numberToDisplay`.

itemTarget

The target frame for the object hyperlinks.

displayItemAction

The action invoked when the user clicks an object hyperlink. The `item` attribute contains the object.

WOSortOrder

Component Description

A `WOSortOrder` component enables the user to sort the objects displayed by a `WODisplayGroup`. It displays an icon representing the current sorting order (☰ unsorted, ≡ ascending, or ≡ descending.) When the user clicks the icon, the component modifies the display group's sort orderings and redisplay the display group's objects.

Synopsis

```
WOSortOrder { displayGroup=aDisplayGroup; key=aString; [displayKey=aString]; }
```

Bindings

`displayGroup`

The display group that receives the new sort order specification.

`key`

The key corresponding to the attribute to sort by.

`displayKey`

A user presentable string corresponding to `key`. The user's browser displays a tooltip above the sort order icon: "Push to toggle sorting order according to *displayKey*."
Defaults to `key`.

WOSortOrderManyKey

Component Description

The WOSortOrderManyKey component provides a user interface to specify the sort ordering of the objects displayed by a WODisplayGroup. The user specifies the key on which to sort using a popup button and the order (ascending or descending). The WOSortOrderManyKey component updates the display group's displayed objects accordingly. This component must be placed in a WOForm.



Synopsis

```
WOSortOrderManyKey { displayGroup=aDisplayGroup; keyList=anArray; };
```

Bindings

displayGroup

The display group that receives the new sorting order specification.

keyList

Array of keys for the attribute that can be used to sort the objects. This array appears in the popup button.

WOStatsPage

Component Description

The WOStatsPage component is displayed in the browser when the user accesses the WOStats direct action. This component is a complete page and is not designed to be embedded within another component. You can modify this page if you want to customize the information that on this page. See *Deploying WebObjects Applications* for more information.

WOTable

Component Description

A WOTable is a container element similar to a WORepetition that repeats its contents (that is, everything between the `<WEBOBJECT...>` and `</WEBOBJECT...>` tags in the template file) a given number of times. It differs from a WORepetition in that it displays the contents as a multi-column table. You can use a WOTable to create dynamically generated banks of check boxes or radio buttons. The WOTable displays the items in `list` from left to right and wraps around to the next line when the number of columns reaches `maxColumns`.

first	second
third	fourth
fifth	sixth

Synopsis

```
WOTable {list=anObjectList; [item=anIteratedObject;} maxColumns=aNumber;
[index=aNumber;} [col=aNumber;} [row=aNumber;} [tableBackgroundColor=hexString;}
[border=aNumber;} [cellpadding=aString;} [cellspacing=aString;}
[rowBackgroundColor=hexString;} [cellBackgroundColor=hexString;} [cellAlign=aString;}
[cellVAlign=aString;} };
```

Bindings

list

Array of objects through which the WOTable iterates.

item

Current item in the list array. (This attribute's value is updated with each iteration.)

maxColumns

Number of columns in the table.

index

Index of the current iteration of the WOTable. (This attribute's value is updated with each iteration.)

col

Current column in the table. (This attribute's value is updated with each iteration.)

row

Current row in the table. (This attribute's value is updated with each iteration.)

tableBackgroundColor

Specifies the background color of the table. This attribute is passed to the `TABLE` HTML tag.

border

Specifies the width, in pixels, of the table border. This attribute is passed to the `TABLE` HTML tag.

cellpadding

Specifies the spacing between cells of the table. This attribute is passed to the `TABLE` HTML tag.

cellspacing

Specifies the spacing within cells of the table. This attribute is passed to the `TABLE` HTML tag.

rowBackgroundColor

Specifies the background color for each row. This attribute is passed to each `TR` HTML tag in the table.

cellBackgroundColor

Specifies the background color for the cells. This attribute is passed to each `TD` HTML tag in the table.

OTHER REFERENCE WOTable

cellAlign

Specifies the horizontal alignment of each cell. This attribute is passed to each TD HTML tag in the table.

cellVAlign

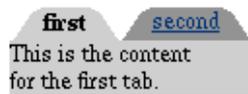
Specifies the vertical alignment of each cell. This attribute is passed to each TD HTML tag in the table.

OTHER REFERENCE WOTable

WOTabPanel

Component Description

The WOTabPanel component displays a tab panel. When the user selects a tab the corresponding panel appears.



This component can be embedded within a WOForm. When the WOTabPanel contains form fields that are bound to an enterprise object's attributes, the HTML element the user clicks to change tabs must be a submit button. This ensures that the values the user types into these fields get stored in the enterprise object. If the HTML element the user clicks is a hyperlink, the enterprise object's attributes are not updated when the user switches tabs and the form field values are lost. The `submitActionName` binding determines the whether the HTML element that switches tabs is a hyperlink or a submit button.

Synopsis

```
WOTabPanel { tabs=anArray; selectedTab=aString; [tabNameKey=aString];
[nonSelectedBgColor=hexString]; [bgcolor=hexString]; [textColor=hexString];
[submitActionName=anActionName]; };
```

Bindings

tabs

An array of objects representing the tabs. For example, the array could be named `movieArray` and contain `Movie` objects.

selectedTab

The object corresponding to the currently selected tab. For example, `currentMovie` could represent the currently selected object in `movieArray`. This attribute gets updated each time the tab panel is redisplayed. In addition, this attribute defines the initially selected object and defaults to the first object in the list.

tabNameKey

A key representing the tab object's name string displayed in the tab. For the movie example, the name key is set to "title" and the `WOTabPanel` will display `currentMovie.title`. If your tab objects are `Strings` (`NSStrings` in Objective-C), do not set this attribute.

nonSelectedBgColor

Color of the tabs that are not selected.

bgcolor

Color of the selected tab and the main panel area.

textColor

Color of the text within the tab.

submitActionName

The name of an action that is called when a new tab is selected. This attribute functions only when the tab panel is embedded within a `WOForm`. If `submitActionName` is not defined or is set to `null`, the tab selectors are hyperlinks. Otherwise the tab selectors are submit buttons. If you want the tab selector to be a submit button, but don't want to call an action, set this attribute to the empty string ("").

Example

The `WOTabPanel` component is used in conjunction with the `WOKeyValueConditional` which conditionally displays the tab panel contents depending on the tab the user selects. The following example shows the HTML, template, and Java files for the parent component containing a tab panel.

```
<WEBOBJECT NAME=TabPanel1>
```

OTHER REFERENCE WOTabPanel

```
<WEBOBJECT NAME=KeyValueConditional1>
    This is the content<BR>for the first tab.<BR>
</WEBOBJECT>
<WEBOBJECT NAME=KeyValueConditional2>
    This is the content<BR>for the second tab.<BR>
</WEBOBJECT>
</WEBOBJECT>
```

```
TabPanel1: WOTabPanel {
    tabs = keyList;
    selectedTab = selection;
}
KeyValueConditional1: WOKeyValueConditional {
    key = "selection";
    value = "first";
}
KeyValueConditional2: WOKeyValueConditional {
    key = "selection";
    value = "second";
}

protected NSArray keyList;
protected String selection;

public Main() {
    keyList = new NSArray(new Object[] {"first","second"});
}
```

The parent component's constructor initializes the key list. The `selection` string holds the current selection which the `WOKeyValueConditional` components query to decide if they should display their contents.

WThresholdColoredNumber

Component Description

The `WThresholdColoredNumber` component displays a number in one of two colors depending on whether the number is above or below a threshold.

Synopsis

```
WThresholdColoredNumber { lowColor=hexString; highColor=hexString; threshold=aNumber;  
value=aNumber; [numberformat=numberFormatString;] };
```

Bindings

`lowColor`

The color in which the number is rendered when the number is below the threshold. Defaults to "#FF0000".

`highColor`

The color in which the number is rendered when the number is equal to or above the threshold. Defaults to "#00FF00".

`threshold`

The threshold at which the number's rendered color is changed.

`value`

The displayed number.

`numberformat`

A format string that specifies how `value` should be formatted as a number. If the value can't be interpreted according to the format you specify, the value is set to `nil`. See the `NSNumberFormatter` class specification for a description of the number format syntax.

OTHER REFERENCE W O Threshold Colored Number

WOToManyRelationship

Component Description

The WOToManyRelationship component displays a set of possible destination objects of an enterprise object's to-many relationship, allows the user to select a subset of the destination objects, and sets the enterprise object's relationship accordingly. It displays the possible destination objects of the relationship in a browser or as a set of checkboxes. This component must be embedded within a WOForm.

Synopsis

```
WOToManyRelationship { [uiStyle="checkbox"|"browser";] sourceObject=anObject;  
sourceEntityName=aString; relationshipKey=aString; [destinationDisplayKey=aString;  
[isMandatory=boolean]; [maxColumns=aNumber]; [size=aNumber]; [dataSource=aDataSource;  
};
```

Bindings

uiStyle

The type of user interface. Defaults to "checkbox" when the WOToManyRelationship component displays five or fewer objects. Otherwise defaults to "browser".

sourceObject

The enterprise object whose relationship is modified. A display group's `queryMatch` dictionary is also a valid source object. This allows you to query for objects having a particular to-many relationship.

sourceEntityName

The name of the entity that is modified.

relationshipKey

The name of the relationship that is modified.

OTHER REFERENCE WOToManyRelationship

destinationDisplayKey

A displayable attribute of the relationship's destination objects. Defaults to "userPresentableDescription".

isMandatory

A flag to indicate that a selection is necessary. Defaults to NO.

maxColumns

The maximum number of columns in the checkbox array. Used only when the user interface style is "checkbox".

size

The maximum number of rows in the browser. Used only when the user interface style is "browser".

dataSource

A data source for the relationship's destination objects. By default, the WOToManyRelationship component creates an EODatabaseDataSource containing all possible destination objects. However, if you want to limit the number of destination objects the user can choose, you can create your own EODatabaseDataSource that has a subset of the possible destination objects and bind it to this attribute.

WOToOneRelationship

Component Description

The WOToOneRelationship component displays a set of possible destination objects of an enterprise object's to-one relationship, allows the user to select one of the destination objects, and sets the enterprise object's relationship accordingly. It displays the possible destination objects of the relationship in a browser, a popup list, or as a set of radio buttons. This component must be embedded within a WOForm.

Synopsis

```
WOToOneRelationship { [uiStyle="radio"|"popup"|"browser";] sourceObject=anObject;  
sourceEntityName=aString; relationshipKey=aString; [destinationDisplayKey=aString;  
[isMandatory=aBoolean]; [maxColumns=aNumber]; [size=aNumber]; [dataSource=aDataSource];  
};
```

Bindings

uiStyle

The type of user interface (radio buttons, popup list, or browser.) Defaults to "radio" when the WOToOneRelationship component displays fewer than five objects. Defaults to "popup" when the component displays between five and twenty objects. Defaults to "browser" when the component displays more than twenty objects.

sourceObject

The enterprise object whose relationship is edited. A display group's queryMatch dictionary is also a valid source object. This allows you to query for objects having a particular to-one relationship.

sourceEntityName

The name of the entity that is modified.

OTHER REFERENCE `WOToOneRelationship`

`relationshipKey`

The name of the relationship that is modified.

`destinationDisplayKey`

A displayable attribute of the relationship's destination objects. Defaults to `"userPresentableDescription"`.

`isMandatory`

A flag to indicate that a selection is necessary. Defaults to `NO`.

`maxColumns`

The maximum number of columns in the radio button array. Used only when the user interface style is `"radio"`.

`size`

The maximum number of rows in the browser. Used only when the user interface style is `"browser"`.

`dataSource`

A data source for the relationship's destination objects. By default, the `WOToOneRelationship` component creates an `EODatabaseDataSource` containing all possible destination objects. However, if you want to limit the number of destination objects the user can choose, you can create your own `EODatabaseDataSource` that has a subset of the possible destination objects and bind it to this attribute.

This Apple manual was written, edited, and composed on a desktop publishing system using Apple Macintosh computers and FrameMaker software.

Line art was created using Adobe™ Illustrator and Adobe Photoshop.

Text type is Palatino® and display type is Helvetica®. Bullets are ITC Zapf Dingbats®. Some elements, such as program listings, are set in Adobe Letter Gothic.

WRITER

Clif Liu

PRODUCTION EDITOR

Gerri Gray