WebObjects Extension Specifications

The WebObjects Extensions framework defines dynamic elements and components that you can use in any application that links to the framework. By default, when you create a WebObjectsApplication or WebObjectsFramework project in Project Builder, you are linked to the WOExtensions framework. In addition, WODefaultApp is already linked to the WOExtensions framework. Thus, you can use the elements and components defined in this framework in virtually all of your applications.

Here are the dynamic elements defined in the WOExtensions Framework:

WOCheckBoxList
WONestedList
WORadioButtonList

In addition to dynamic elements, the WebObjects Extensions framework defines shared components. WebObjects has the ability to share components across applications. All you have to do is define a component, place it in a framework, place the framework in *NeXT_ROOT*/**NextLibrary/Frameworks** and you can use that component in any WebObjects application as long as it links to that framework.

Some shared components define attributes, similar to the way dynamic elements define attributes. To use such components, you must bind their attributes to values and methods from your component's script or code file.

Here are the shared components defined in the WOExtensions Framework:

WORedirect
WOSimpleArrayDisplay
WOSortOrder
WOStats
WOToManyRelationship
WOToOneRelationship

See the *WebObjects Developer's Guide* for a more complete introduction to shared components. See the *Dynamic Elements Reference* to learn how to use the specifications in this guide.

WOCheckBoxList

Synopsis

WOCheckBoxList { list=*anObjectList*; **item**=*anIteratedObject*; **value**=*displayedValue*; [**index**=*aNumber*;] [**prefix**=*prefixString*;] [**suffix**=*suffixString*;] [**selections**=*selectedValues*;] [**name**=*fieldName*;] [**disabled**=YES|NO;] ... **};**

Description

WOCheckBoxList displays a list of check boxes. The user may select several of the objects in the list, and this sublist is returned as **selections**.

list
: Array of objects that the WOCheckBoxList will iterate through.

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

value
: String to display beside the check box for the current item.

index
: Index of the current iteration of the WOCheckBoxList.

prefix
: An arbitrary HTML string inserted before each value.

suffix
: An arbitrary HTML string inserted after each value.

selections
: An array of objects that the user chose from the list.

name
: Name that uniquely identifies this element within the form. You may specify a name or let WebObjects automatically assign one at runtime.

disabled
: If **disabled** evaluates to YES, this element appears in the page but is not active.

WONestedList

Synopsis

WONestedList { list=*anObjectList*; **item**=*anIteratedObject*; **value**=*displayedValue*; **sublist** = *aSubarray*; **action**=*aMethod*; **selection**=*selectedValue*; [**index**=*aCurrentIndex*;] [**level**=*aCurrentLevel*;] [**isOrdered**=YES|NO;] [**prefix**=*prefixString*;] [**suffix**=*suffixString*;] ... **};**

Description

WONestedList recursively displays a hierarchical, ordered (numbered) or unordered (bulleted) list of hyperlinks. This element is useful when you want to display hierarchical lists. When the user clicks one of the objects in the list, it is returned in **selection** and the **action** method is invoked.

At any point during iteration of the list, the method specified by the **sublist** attribute returns the current list's sublist (if any), **level** specifies the current nesting level (where the topmost level is zero), **index** gives index of the current item within that nesting level (**item** returns the actual item), and **isOrdered** specifies whether the current sublist should be a numbered list or a bulleted list.

list
: Hierarchical array of objects that the WONestedList will iterate through.

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

value
: String to display as a hyperlink for the current item.

sublist
: Method that returns the sublist of the current item or **nil** if the current item is a leaf.

action
: Action method to invoke when the element is activated. This method must return a WOElement.

selection
: When the page is submitted, **selection** contains the item that the user clicked.

index
: Index of the current iteration of the WONestedList. The index is unique to each level—that is, it starts at 0 for each sublist.

level
: Nesting level of the current iteration of the WONestedList. The topmost level is level 0.

isOrdered
: If **isOrdered** evaluates to YES, the current sublist is rendered as an ordered list. The default is to render as an unordered list.

prefix
: An arbitrary HTML string inserted before each value.

suffix
: An arbitrary HTML string inserted after each value.

WORadioButtonList

Synopsis

WORadioButtonList { list=*anObjectList*; **item**=*anIteratedObject*; **value**=*displayedValue*; [**index**=*aNumber*;] [**prefix**=*prefixString*;] [**suffix**=*suffixString*;] [**selection**=*selectedValue*;] [**name**=*fieldName*;] [**disabled**=YES|NO;] ... **};**

Description

WORadioButtonList displays a list of radio buttons. The user may select one of the objects in the list, and this object is returned as **selection**.

list
: Array of objects that the WORadioButtonList will iterate through.

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

value
: String to display beside the radio button for the current item.

index
: Index of the current iteration of the WORadioButtonList.

prefix
: An arbitrary HTML string inserted before each value.

suffix
: An arbitrary HTML string inserted after each value.

selection
: An object that the user chose from the list.

name
: Name that uniquely identifies this element within the form. You may specify a name or let WebObjects automatically assign one at runtime.

disabled
: If **disabled** evaluates to YES, this element appears in the page but is not active.

WORedirect

Synopsis

WORedirect { url = *aURL*; **};**

Description

WORedirect is a component that may be returned to force the user's browser to redirect to another URL. You should only return this component as a response to an action method and never use it in an declarations file directly. This component can be useful, for example, if you have an image map with both static and dynamic actions.

```
- (WOComponent *)someAction

{

    WOComponent *aRedirect = [[self application] pageWithName:@"WORedirect"];

    [aRedirect setURL:@"http://enterprise.apple.com"];

    return aRedirect;

}
```

WOSimpleArrayDisplay

Synopsis

WOSimpleArrayDisplay { list = *anObjectArray*; [**numberToDisplay =** *anInt*;] [**itemDisplayKey =** *aKey*;] [**listAction =** *methodName*;] [**listActionString =** *actionString*;]**};**

Description

The WOSimpleArrayDisplay component is intended to display a to-many relationship of an Enterprise Object. It could also be used to display an array of objects.

The WOSimpleArrayDisplay is designed to only provide an idea of what is in the relationship (or list). It only displays the first *n* items in the list, where *n* is specified by the **numberToDisplay** attribute. (The default is 5.) If you want, you can have a hyperlink displayed at the bottom of the list by specifying the **listAction** and **listString** attributes. You could use this hyperlink to provide more information about the relationship or, for example, to list all of the elements in the relationship.

If you use Direct to Web to create your application, you may have a page that uses WOSimpleArrayDisplay.

list
: Array of items to display.

numberToDisplay
: The number of items to display at one time. The default is 5 items.

itemDisplayKey
: Key to the value that should be displayed for each item. For example, if you were displaying a movies entity object, you'd want to display the value of the movieName key.

listAction
: Method to perform when the hyperlink is clicked.

listActionString
: The string that should appear on the hyperlink. The default is "Inspect."

WOSortOrder

Synopsis

WOSortOrder { displayGroup = *aDisplayGroup*; **key =** *aKey*; **};**

Description

The WOSortOrder component displays an active image that shows the current sort ordering for a WODisplayGroup and allows the user to change it. A WODisplayGroup's sort ordering specifies the order in which items it displays should be displayed (ascending or descending).

If you use Direct to Web to create your application, you may have a page that uses WOSortOrder.

displayGroup
: WODisplayGroup to be sorted.

key
: Key on which the WODisplayGroup should be sorted.

WOStats

Synopsis

WOStats is a full-paged component. To access it, use a URL such as this one:

```
http://localhost/cgi-bin/WebObjects/MyAppName.woa/-/WOStats
```

Description

The WOStats component is a page that displays statistics about the application. These statistics are recorded by all WebObjects applications. If you need to find out about transaction processing times or how many users are accessing your application, you can access WOStats while the application is running.

The statistics described here are actually recorded by a WOStatisticsStore object. You can use WOStatisticsStore to do things like change the moving average sample size. For more information, see its class specification in the *WebObjects Class Reference*.

The WOStats page provides this information:

Transactions
: Total number of transactions processed by this application instance. A *transaction* is defined as one cycle of the WebObjects request-response loop. That is, a transaction begins when the user sends an HTTP request and ends when the user receives a response page. (Starting up the application is a transaction and accessing the WOStats page is a transaction as well.)

Avg. Transaction Time
: Average length of time it took to process a transaction.

Avg. Idle Time
: Average length of time the application sat idle between transactions.

Moving Avg. Transaction Time
: Average length of time it took to process the last *n* transactions, where *n* is the moving average sample size.

Move Avg. Idle Time
: Average length of time the application sat idle between the last *n* transactions, where *n* is the moving average sample size.

Sample Size for Moving Avg.
: The number of transactions used to compute moving average statistics, such as the Moving Avg. Transaction Time and the Moving Avg. Idle Time. (The sample size is set through the WOStatisticsStore class.)

Started at
: Time at which this application instance started running.

Running time
: Length of time this instance has been running.

Avg. Transactions Per Session
: Average number of transactions each user performed in a session.

Total Sessions Created
: The number of sessions this application has created in its lifetime. A session is created each time a new user accesses an application.

Current Active Sessions
: Number of the created sessions that are still alive.

Peak Active Sessions
: The maximum number of sessions that have been active at the same time.

Avg. Session Life

Average length of time a session lasted.

Peak Concurrent Sessions At

The date and time at which the peak number of active sessions was reached.

Memory Usage

The amount of memory allocated to the application. Resident Set Size is the number of physical memory pages. Virtual is the number of virtual memory pages.

Avg. Memory Usage Per Session

Average amount of memory each session took.

The memory display differs depending on the operating system. On Windows NT, you see the amount of memory reserved for this application and the amount committed. On all other platforms, you see the Resident Set Size (the number of physical memory pages) and the Virtual memory size (number of virtual memory pages).

Response Description for Last User

A list of the pages accessed by the last session that timed out. The pages are listed from first accessed to last accessed. This list is the same as the list that appears in the application log, if the application log is enabled.

Page Statistics

A table that shows page generation times for each component in the application. (Only parent components, components that represent a full page, are listed; nested components that represent a portion of a page are not listed.)

The page generation time is the amount of time it took for the request-response loop to receive the request for that page, process the request, invoke the appropriate action in the request component, and generate the page. Often, the bulk of the time it takes to generate a page happens in the action invocation and response generation phases of the request-response loop. The initial processing of the request takes a minimal amount of time.

For example, suppose the user clicks a button in Page A that fetches items from a database and displays those items in Page B. The total amount of time it takes to handle the request on Page A, invoke the action, fetch items from the database, and generate Page B is recorded as the amount of time it took to generate Page B.

Detailed Statistics

If the application supports more detailed statistics (which it does if you've overridden **descriptionForResponse:inContext:** in the application's components), this table shows how many times a particular instance of a component was accessed. (An instance of a component might have different values for the component's instance variables.)

WOToManyRelationship

Synopsis

WOToManyRelationship { sourceEntityName = *anEntity*; **relationshipKey =** *aKey*; **sourceObject =** *anObject*; [**isMandatory =** YES|NO;] [**destinationDisplayKey =** *aKey*;] [**dataSource =** *aDataSource*;] [**uiStyle = "browser"** | **"checkbox"** ;] **};**

Description

The WOToManyRelationship component displays items from a to-many relationship in either a browser or a check-box list. Users can select one or more items from this list to learn more about those items.

For example, suppose you have a database of Movies where each movie is a table (or entity). Each movie has one or more roles, which means that the Movies table would have a to-many relationship with the MovieRoles table. You could use this component on a page that displays information about a movie, using this component to display the roles associated with this movie.

This component is used only in applications that access a database using the Enterprise Objects Framework. In particular, it is used if you use Direct to Web to create your application. Because WOToManyRelationship returns a form element, it must be used within an HTML form.

sourceEntityName
: The name of the entity that contains the relationship. In the Movies example, this would be "Movies."

relationshipKey
: The key for the relationship that you want to display. In the Movies example, you'd want to display the **roles** key.

sourceObject
: An object that contains the actual relationship value. This object can be an enterprise object, a mutable dictionary, or anything else that can contain the relationship. Upon return, this object contains the user's selection.

isMandatory
: If YES, the relationship must exist. If the relationship must exist, the element must have at least one item selected when the form is submitted, so the WOToManyRelationship selects the first item if the list if the user has not selected any. If NO, the relationship is optional.

destinationDisplayKey
: Property to display in the list. For example, with the MovieRoles entity, you would want to display the **roleName** property.

dataSource
: EODatabaseDatasource containing the items in the to-many relationship.

uiStyle
: If **"browser"**, displays as a browser. If **"checkbox"**, displays as a list of check boxes. The default depends on the number of items in the list. If there are less than 5 items, the default is a check-box list. If there are 5 items or more, the default is a browser.

WOToOneRelationship

Synopsis

WOToOneRelationship { sourceEntityName = *anEntity*; **relationshipKey =** *aKey*; **sourceObject =** *anObject*; [**dataSource =** *aDataSource*;] [**destinationDisplayKey =** *aKey*;] [**isMandatory =** YES|NO;] [**uiStyle = "radio"|"popup"|"browser";**] **};**

Description

The WOToOneRelationship component displays items from a to-one relationship in a pop-up list, a radio button list, or a browser. Users can select an item from this list to learn more about that item.

For example, suppose you have a database of Movies where each movie is a table (or entity). Each movie has only one studio, which means that the Movies table would have a to-one relationship with the Studios table. You could use this component on a page that displays information about a movie, using this component to display the studio associated with the movie.

This component is used only in applications that access a database using the Enterprise Objects Framework. In particular, it is used if you use Direct to Web to create your application. Because WOToOneRelationship returns a form element, it must be used within an HTML form.

sourceEntityName
: The name of the entity that contains the relationship. In the Movies example, this would be "Movies."

relationshipKey
: The key for the relationship that you want to display. In the Movies example, you'd want to display the **studios** key.

sourceObject
: An object that contains the actual relationship value. This object can be an enterprise object, a mutable dictionary, or anything else that can contain the relationship. Upon return, this object contains the user's selection.

dataSource
: EODatabaseDataSource containing the items in the to-one relationship.

destinationDisplayKey
: Property to display in the list. For example, with the Studio entity, you would want to display the **name** property.

isMandatory
: If YES, the relationship must exist. If NO, the relationship is optional.

uiStyle
: Specifies how to display the list: as a browser, a radio button list, or a pop-up list. The default depends on the number of items in the list. If there are less than 5 items, the default is a radio button list. If between 5 and 20 items, the default is a pop-up list. If there are more than 20 items, a browser is used.