Dynamic Element Specifications

Introduction

Dynamic elements serve as the basic building blocks of WebObjects applications by linking an application's scripted or compiled behavior to an HTML page. The linkage can be two-way, in that a dynamic element:

- Initially sets its attributes to values specified by scripted or compiled variables or methods.
- Represents itself as HTML when called upon to do so.
- Synchronizes the values of its attributes to those entered by the user, and passes these values back to your script or compiled code.

With WebObjects, most pages sent to the user's browser are composed of HTML from a static template combined with HTML that's dynamically generated by dynamic elements embedded (directly, or in the case of reusable components, indirectly) in that template.

Here are the dynamic elements that are defined in WebObjects Release 3.0:

WOActiveImage
WOApplet
WOBrowser
WOCheckBox
WOConditional
WOEmbeddedObject
WOForm
WOFrame
WOGenericContainer
WOGenericElement
WOHiddenField
WOHyperlink
WOImage
WOJavaScript
WOPasswordField
WOPopUpButton
WORadioButton
WORepetition
WOResetButton
WOStateStorage
WOString
WOSubmitButton
WOText
WOTextField
WOVBScript

See the WebObjects Developer's Guide for a more complete introduction to Dynamic Elements.

How to Use These Specifications

Each dynamic element specification that follows is divided into two sections: a synopsis and a description. 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 and each of its attributes.

The element synopses use several conventions that you should be aware of, for example:

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

- **Bold** denotes words or characters that are to be taken literally (typed as they appear). For example, the **action** and **value** attributes are to be take literally in the synopsis above.
- *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.
- Ellipsis (...) represents additional attributes and values that you might add but that aren't part of the element's specification. When a dynamic element is asked to produce its HTML representation, these additional attributes and values are simply copied into the HTML stream. The values for these additional attributes can be derived dynamically, just as with the built-in attributes.

Another point to note concerns the capitalization of attribute names (**action**, **value**, **disabled** above). In the specifications that follow, compound attribute names are shown with the first letter of each embedded word capitalized. For example, WOActiveImage has an **imageMapFile** attribute. You can capitalize attributes exactly as shown in these specifications, or you can use all lowercase letters (**imagemapfile**). No other capitalization is allowed.

WOActiveImage

Synopsis

WOActiveImage { src=*aPath* | value=*aMethod*; action=*aMethod* | href=*aURL*; [imageMapFile=*aString*;] [name=*aString*;] [x=*aNumber*; y=*aNumber*;] [target=*frameName*;] [disabled=YES|NO;] ... };

Description

A WOActiveImage displays an image within the HTML page. If the WOActiveImage is disabled, it simply displays its image as a passive element in the page. If enabled, the image is active, that is, clicking the image generates a request.

If located outside an HTML form, a WOActiveImage functions as a mapped, active image. When the user clicks such a WOActiveImage, the coordinates of the click are sent back to the server. Depending on where the user clicks, different actions can be invoked. An image map file associates actions with each of the defined areas of the image.

Within an HTML form, a WOActiveImage functions as a graphical submit button. You typically use WOActiveImages when you need more than one submit button within a form.

src
: Path to the file containing the image data. src can be statically specified in the declarations file, an object that responds to a description message by returning an NSString, or a method that returns an NSString.

value
: Image data in the form of a WOElement object. This data can come from a database, a file, or memory.

action
: Method to invoke when this element is clicked. If imageMapFile is specified, action is only invoked if the click is outside any mapped area. In other words, action defines the default action of the active image.

href
: URL to direct the browser to as a default when the image is clicked and no hot zones are hit.

imageMapFile
: Name of the image map file.

name
: If name is specified then the hit point is specified as name.x=*value*; name.y=*value*; in the form. This is useful when you need to use this element to submit a form to an external URL that expects the hit point to be expressed in a certain format.

x, y
: If specified, returns the coordinates of the user's click within the image.

target
: Frame in a frameset that will receive the page returned as a result of the user's click.

disabled
: If YES, a regular image element (<IMG>) is generated rather than an active image.

The Image Map File

If imageMapFile is specified, WebObjects searches for the file within the component bundle (Component.wo/). If it isn't found there, WebObjects searches the application directory (MyApplication.woa/).

Each line in the image map file has this format:

shape action coordinate-list

shape
: Either 'rect' or 'circle' (polygon not yet supported). For 'rect' shape, the coordinates x1,y1 specify the upper-left corner of the hot zone, and x2,y2 specify lower right corner. For 'circle' shape, the x1,y1 is the origin, and x2,y2 is a point on the circle.

action
: Name of the method to invoke.

coordinate-list
: x1, y1 x2, y2 ...

Here's an example of an image map file:

```
rect  home     0,0 135,56

rect  buy      135,0 270,56
```

Java Support: WOApplet and WOParam

Synopsis

WOApplet { code=*javaClassName*; width=*aWidth*; height=*aHeight*; [associationClass=*className*;] [codeBase=*aPath*;] ... };

WOParam { name=*aString*; value=*aString* | action=*aMethod*; ... };

Description

WOApplet is a dynamic element that generates HTML to specify a Java applet. The applet's parameters are passed by one or more WOParam elements.

WOApplet:

code
: Name of the Java class.

width
: Width, in pixels, of the area to allocate for the applet.

height
: Height, in pixels, of the area to allocate for the applet.

associationClass
: Name of Java class that aids in communication between client applet and the server.

codeBase
: Directory that contains the applet code. If this attribute is omitted, the applet code is assumed to be in the same directory as the template HTML file.

WOParam:

name
: Symbolic name associated with this element's value.

value
: Value of this parameter.

action
: Method that the applet will invoke.

WOBrowser

Synopsis

WOBrowser { list=*anArray*; [item=*anItem*; value=*displayedValue*;] [selections=*objectArray*;] [name=*fieldName*;] [disabled=YES|NO;] ... };

Description

WOBrowser displays itself as a selection list that allows the user to select multiple items at a time. The related element WOPopUpButton is similar to WOBrowser except that it restricts the user to selecting only one item at a time.

list
: Array of objects from which the browser derives its values. For example, colleges could name the list containing objects that represent individual schools.

item
: Identifier for the elements of the list. For example, aCollege could represent an object in the colleges array.

value
: Value to display in the selection list; for example, aCollege.name for each college object in the list.

selections
: Array of objects that the user chose from list. For the college example, selections would hold college objects.

name
: Name that uniquely identifies this element within the form. You can 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.

WOCheckBox

Synopsis

WOCheckBox {[value=*defaultValue*; [selection=*selectedValue*;]] [name=*fieldName*;] [disabled=YES|NO;] ... };

WOCheckBox {[checked=YES|NO;] [name=*fieldName*;] [disabled=YES|NO;] ... };

Description

A WOCheckBox object displays itself in the HTML page as its namesake, a check box user interface control. It corresponds to the HTML element <INPUT TYPE="CHECKBOX"...>.

value
: Value of this input element. If not specified, WebObjects provides a default value.

selection
: If selection and value are equal when the page is generated, the check box is checked. When the page is submitted, selection is assigned the value of the check box.

checked
: During page generation, if checked evaluates to YES, the check box appears in the checked state. During request handling, checked reflects the state the user left the check box in: YES if checked; NO if not.

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.

WOConditional

Synopsis

WOConditional { condition=YES|NO; [**negate**=YES|NO;] ... };

Description

A WOConditional object controls whether a portion of the HTML page will be generated, based on the evaluation of its assigned condition.

condition

Expression to evaluate. If the expression evaluates to YES (assuming **negate** is NO), the HTML code controlled by the WOConditional object is emitted; otherwise it is not.

negate

Inverts the sense of the condition. By default, **negate** is assumed to be NO.

The negate attribute lets you use the same test to display mutually exclusive information; for example:

HTML file

```
<HTML>
<WEBOBJECTS NAME="PAYING_CUSTOMER">Thank you for your order!</WEBOBJECTS>
<WEBOBJECTS NAME="WINDOW_SHOPPER">Thanks for visiting!</WEBOBJECTS>
</HTML>
```

Declarations File

```
PAYING_CUSTOMER: WOConditional {condition=payingCustomer;};
WINDOW_SHOPPER: WOConditional {condition=payingCustomer; negate=YES;};
```

Script File

```
- payingCustomer {
   if (/* ordered something */) {
      return YES;
   }
   return NO;
}
```

WOEmbeddedObject

Synopsis

WOEmbeddedObject {value=*aMethod*; | src=*aURL*; ... };

Description

A WOEmbeddedObject provides support for Netscape plug-ins. It corresponds to the HTML element <EMBED SRC = >. If the embedded object's content comes from outside the WebObjects application, use the **src** attribute. If the embedded object's content is returned by a method within the WebObjects application, use the **value** attribute.

value
: Method that will supply the content for this embedded object.

src
: External source that will supply the content for this embedded object.

WOForm

Synopsis

WOForm { [**action**=*aMethod*; | **href**=*aURL*;] [**multipleSubmit**=YES|NO;] ... };

Description

A WOForm is a container element that generates a fill-in form. It gathers the input from the input elements it contains and sends it to the server for processing. WOForm corresponds to the HTML element <FORM ... > ... </FORM>.

href

URL specifying where the form will be submitted.

action

Action method that's invoked when the form is submitted. If the form contains a dynamic element that has its own action (such as a WOSubmitButton or a WOActiveImage), that action is invoked instead of the WOForm's.

multipleSubmit

If **multipleSubmit** evaluates to YES, the form can have more than one WOSubmitButton, each with its own action. By default, WOForm supports only a single WOSubmitButton. **Note:** Some older browsers support only a single submit button in a form.

WOFrame

Synopsis

WOFrame { value=*aMethod*; | src=*aURL*; | pageName=*aString*; ... };

Description

WOFrame represents itself as a dynamically generated Netscape Frame element.

value
: Method that will supply the content for this frame.

pageName
: Name of WebObjects page that will supply the content for this frame.

src
: External source that will supply the content for this frame.

WOGenericContainer

Synopsis

WOGenericContainer { elementName = *aConstantString*; ... };

Description

WOGenericContainer provides a way for WebObjects to accommodate custom HTML container elements, that is, elements that affect a range of text. Since the HTML language is evolving rapidly, it's convenient to have a way to dynamically generate elements which are not explicitly supported by WebObjects.

In HTML, a container element (for example, <A ... > ... </A>) has opening and closing tags that delimit the text or graphic affected by the element. In contrast, an empty element (for example <HR> or <BR>) is represented by a single tag and so can't enclose any text or graphics. (See the related element WOGenericElement for information about the support of empty elements.)

elementName
: Name of the HTML element to generate. This name (for example "TEXTAREA") will be used to generate the container's opening and closing tags (<TEXTAREA>...</TEXTAREA>).

elementName must be statically defined, that is, it must be a constant. It can't be something returned by a script method, for example. Please note that for elements with URL attributes, the URLs specified will appear as is in the HTML document.

This approach works for many elements, but has one limitation. Some HTML elements have an href attribute that associates the element with a URL. In WebObjects, the corresponding dynamic element generally has two mutually exclusive attributes, href and action, which make use of the HTML element's href attribute. (See WOHyperlink for an element that can have either an href or an action attribute.) The dynamic element's href attribute simply returns a URL, but action invokes a WebObjects method, which returns a URL. This overloading of the HTML href attribute is not supported by WOGenericContainer. If your custom element requires this functionality, you will have to create your own subclass of WODynamicElement.

WOGenericElement

Synopsis

WOGenericElement { elementName = *aConstantString*; ... };

Description

WOGenericElement provides a way for WebObjects to accommodate custom HTML elements that are empty, that is, that don't affect a range of text. Since the HTML language is evolving rapidly, it's convenient to have a way to dynamically generate elements which are not explicitly supported by WebObjects.

In HTML, an empty element (for example <HR> or <BR>) is represented by a single tag and so can't enclose any text or graphics. In contrast, a container element (for example, <A ... > ... </A>) has opening and closing tags that delimit the text or graphic affected by the element. (See the related element WOGenericContainer for information about the support of container elements.)

elementName
Name of the HTML element to generate. This name (for example "HR") will be used to generate the element's tag (<HR>).

elementName must be statically defined, that is, it must be a constant. It can't be something returned by a script method, for example. Please note that for elements with URL attributes, the URLs specified will appear as is in the HTML document.

This approach works for many elements, but has one limitation. Some HTML elements have an **href** attribute that associates the element with a URL. In WebObjects, the corresponding dynamic element generally has two mutually exclusive attributes, **href** and **action**, which make use of the HTML element's **href** attribute. (See WOHyperlink for an element that can have either an **href** or an **action** attribute.) The dynamic element's **href** attribute simply returns a URL, but **action** invokes a WebObjects method, which returns a URL. This overloading of the HTML **href** attribute is not supported by WOGenericElement. If your custom element requires this functionality, you will have to create your own subclass of WODynamicElement.

WOHiddenField

Synopsis

WOHiddenField { value=*defaultValue*; [name=*fieldName*;] [disabled=YES|NO;] ... };

Description

A WOHiddenField adds hidden text to the HTML page. It corresponds to the HTML element <INPUT TYPE="HIDDEN"...>. Hidden fields are sometimes used to store application state data in the HTML page. In WebObjects, the WOStateStorage element is designed expressly for this purpose.

value

Value for the hidden text field.

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, the element appears in the page but is not active.

WOHyperlink

Synopsis

WOHyperlink { action=*aMethod* | href=*aURL* | pageName=*aString*; [fragmentIdentifier=*anchorFragment*;] [string=*aString*;] [target=*frameName*;] [disabled=YES|NO;] ... };

Description

WOHyperlink generates a hypertext link in an HTML document.

action
: Action method to invoke when this element is activated. The method must return a WOElement.

href
: URL to direct the browser to when the image is clicked.

pageName
: Name of WebObjects page to display when the link is clicked.

fragmentIdentifier
: Named location to display in the destination page.

string
: Text displayed to the user as the link. If you include any text between the <WEBOBJECTS ...> and </ WEBOBJECT> tags for this element, the contents of string is appended to that text.

target
: Frame in a frameset that will receive the page returned as a result of the user's click.

disabled
: If evaluates to YES, the content string is displayed, but the hyperlink is not active.

WOImage

Synopsis

WOImage { src=*aPath* | value=*imageData*; ... };

Description

A WOImage displays an image in the HTML. It corresponds to the HTML element <IMG SRC="URL">.

src

Path to the file containing the image data. The source can be statically specified in the declaration file or it can be an NSString, an object that responds to a description message by returning an NSString, or a method that returns an NSString.

value

Image data in the form of a WOElement object. This data can come from a database, a file, or memory.

WOJavaScript

Synopsis

WOJavaScript { **scriptFile**=*aPath* | **scriptString**=*aString* | **scriptSource**=*aURL*; [**hideInComment**=*aBOOL*;] ... };

Description

WOJavaScript lets you embed a script written in JavaScript in a dynamically generated page.

scriptFile
: Path to the file containing the script. The path can be statically specified in the declaration file or it can be an NSString, an object that responds to a **description** message by returning an NSString, or a method that returns an NSString.

scriptString
: String containing the script. Typically, **scriptString** is an NSString object, an object that responds to a **description** message by returning an NSString, or a method that returns an NSString.

scriptSource
: URL specifying the location of the script.

hideInComment
: If **hideInComment** evaluates to YES, the script will be enclosed in an HTML comment (<!-- *script* //-->). Since scripts can generate errors in some older browsers that weren't designed to execute them, you may want to enclose your script in an HTML comment. Browsers designed to run these scripts will still be able to execute them despite the surrounding comment tags.

WOPasswordField

Synopsis

WOPasswordField { value=*defaultValue*; [name=*fieldName*;] [disabled =YES|NO;] ... };

Description

A WOPasswordField represents itself as a text field that doesn't echo the characters that a user enters. It corresponds to the HTML element <INPUT TYPE="PASSWORD"...>.

value

During page generation, value sets the default value of the text field. This value is not displayed to the user. During request handling, value holds the value the user entered into the field, or the default value if the user left the field untouched.

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, the element appears in the page but is not active.

WOPopUpButton

Synopsis

WOPopUpButton { list=*anArray*; [item=*anItem*; value=*displayedValue*;] [selection=*objectArray*;] [name=*fieldName*;] [disabled=YES|NO;] ... };

Description

WOPopUpButton displays itself as a selection list that allows the user to select only one item at a time. The related element WOBrowser is similar to WOPopUpButton except that it allows the user to select more than one item at a time.

list

Array of objects from which the WOPopUpButton derives its values. For example, **colleges** could name the array containing objects that represent individual schools.

item

Identifier for the elements of the list. For example, **aCollege** could represent an object in the colleges array.

value

Value to display in the selection list; for example, **aCollege.name** for each college object in the list.

selection

Array of objects that the user chose from the selection list. For the college example, selection would hold college objects. Since a WOPopUpButton lets the user select only one item at a time, this array holds no more than one item.

name

Name that uniquely identifies this element within the form. You can 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.

WORadioButton

Synopsis

WORadioButton {[value=*defaultValue*; [selection=*selectedValue*]]; [name=*fieldName*;] [disabled=YES|NO;] ... };

WORadioButton {[checked=YES|NO;] [name=*fieldName*;] [disabled=YES|NO;] ... };

Description

WORadioButton represents itself as an on-off switch. Radio buttons are normally grouped, since the most important aspect of their behavior is that they allow the user to select no more than one of several choices. If the user selects one button, the previously selected button (if any) becomes deselected.

Since radio buttons normally appear as a group, WORadioButton is commonly found within a WORepetition.

checked
: During page generation, if checked evaluates to YES, the radio button appears in the selected state. During request handling, checked reflects the state the user left the radio button in: YES if checked; NO if not.

value
: Value of this input element. If not specified, WebObjects provides a default value.

selection
: If selection and value are equal when the page is generated, the radio button is selected. When the page is submitted, selection is assigned the value of the radio button.

name
: Name that identifies the radio button's group. Only one radio button at a time can be selected within a group.

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

Note that either checked or value is required in a WORadioButton declaration, but that they are mutually exclusive.

WORepetition

Synopsis

WORepetition {list=*anObjectList*; item=*anIteratedObject*; [index=*aNumber*;] [identifier=*aString*;] ... };

WORepetition {count=*aNumber*; [index=*aNumber*;] ... };

Description

A WORepetition is a container element that repeats its contents (that is, everything between the <WEBOBJECT...> and </WEBOBJECT...> tags in the template file) a given number of times. You can use a WORepetition to create dynamically generated ordered and unordered lists or banks of check boxes or radio buttons.

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

item
: Current item in the list array.

index
: Index of the current iteration of the WORepetition.

identifier
: Value used to uniquely identify this item in the list array. Typically it is the primary key of an enterprise object.

count
: Number of times this element will repeat its contents.

WOResetButton

Synopsis

WOResetButton { value=*aString*; ... };

Description

A WOResetButton element generates a reset button in an HTML page. This element is used within HTML forms.

value

Title of the button.

WOStateStorage

Synopsis

WOStateStorage { [size=*numBytes*;] ... };

Description

A WOStateStorage element provides a simple mechanism for storing application state in an HTML page. If you include a WOStateStorage element in a form, any session and persistent data will be stored in the page rather than on the server.(For a detailed discussion of state management in WebObjects applications, see the chapter “Managing State” in the *WebObjects Developer’s Guide*).

WOStateStorage uses HTML hidden fields (<INPUT TYPE="HIDDEN"...>) to store state data. It will use as many hidden field as needed to store the data, but no field will be larger than the size specified by the **size** attribute. The default size setting is designed to work with most browsers.

size

Maximum size for each of the hidden fields used to store the state data. This attribute is optional; if **size** is not specified, the maximum size for hidden fields will be 1000 bytes.

Since WOStateStorage elements are implemented using hidden fields–which in HTML must be located within a form–they too must be located within a form. If a page has more than one form, you must declare a WOStateStorage element within each form.

WOString

Synopsis

WOString { **value**=*aString*; [**escapeHTML**=YES|NO;] [**dateformat**=*dateFormatString*;] [**numberformat**=*numberFormatString*;] ... };

Description

A WOString represents itself in the HTML page as a dynamically generated string.

value

Text to display in the HTML page. **value** is typically assigned an NSString object, an object that responds to a **description** message by returning an NSString, or a method that returns an NSString.

The NSString's contents are substituted into the HTML in the place occupied by this dynamic element.

dateformat

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

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 NSDecimalNumber object. If the element's value can't be interpreted according to the format you specify, **value** is set to **nil**. See the NSNumberFormatter class specification for a description of the number format syntax.

escapeHTML

If **escapeHTML** is YES, HTML tags in WOString's contents are protected from being interpreted by the browser; otherwise, they are not.

By default, WebObjects tries to ensure that the contents of a WOString appears in the client browser just as it appears in the WebObjects application source code. Thus, if a WOString's value is "<B>a bold idea</B>" (and **escapeHTML** is YES or not specified), the string will be passed to the browser as "&lt;B&gt;a bold idea&lt;/B&gt;" and it will appear in the browser as "<B>a bold idea</B>". If **escapeHTML** is NO, WebObjects simply passes the string to the browser without protecting HTML tags from being interpreted as commands. In this case, the string will appear in the browser as "**a bold idea**".

WOSubmitButton

Synopsis

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

Description

A WOSubmitButton element generates a submit button in an HTML page. This element is used within HTML forms.

action
: Action method to invoke when the form is submitted.

value
: Title of the button.

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

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

WOText

Synopsis

WOText { value=*defaultValue*; [name=*fieldName*;] [disabled=YES|NO;] ... };

Description

WOText generates a multi-line field for text input and display. It corresponds to the HTML element <TEXTAREA>.

value

During page generation, value specifies the text that is displayed in the text field. During request handling, value contains the text as the user left it.

name

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

disabled

If disabled evaluates to YES, the text area appears in the page but no input is allowed.

WOTextField

Synopsis

WOTextField { value=*aValue*; [dateformat=*dateFormatString*;] [numberformat=*numberFormatString*;] [name=*fieldName*;] [disabled=YES|NO;] ... };

Description

A WOTextField represents itself as a text input field. It corresponds to the HTML element <INPUT TYPE="TEXT"...>.

value
: During page generation, value sets the default value displayed in the single-line text field. During request handling, it holds the value the user entered into the field, or the default value if the user left the field untouched.

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

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 NSDecimalNumber object. If the element's value can't be interpreted according to the format you specify, value is set to nil. See the NSNumberFormatter class specification for a description of the number format syntax.

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, the element appears in the page but is not active.

WOVBScript

Synopsis

WOVBScript { scriptFile=*aPath* **| scriptString=***aString* **| scriptSource=***aURL*; [**hideInComment=***aBOOL*;] ... };

Description

WOVBScript lets you embed a script written in Visual Basic in a dynamically generated page.

scriptFile
Path to the file containing the script. The path can be statically specified in the declaration file or it can be an NSString, an object that responds to a **description** message by returning an NSString, or a method that returns an NSString.

scriptString
String containing the script. Typically, **scriptString** is an NSString object, an object that responds to a **description** message by returning an NSString, or a method that returns an NSString.

scriptSource
URL specifying the location of the script.

hideInComment
If **hideInComment** evaluates to YES, the script will be enclosed in an HTML comment (<!-- *script* -->). Since scripts can generate errors in some older browsers that weren't designed to execute them, you may want to enclose your script in an HTML comment. Browsers designed to run these scripts will still be able to execute them despite the surrounding comment tags.]