WebObjects Release Notes

For Release 3.0
Last Updated 1/2/97

This file contains notes about the WebObjects family of products: WebObjects, WebObjects Pro, WebObjects Enterprise, and WebObjects Lite. WebObjects release 3.0 is the current release for the Window NT, Solaris, and NEXTSTEP platforms.

Special Notes

For release notes pertaining to the Java Extensions product, see NeXTanswer #2492.

After you install WebObjects 3.0, Project Builder contains a new project type: WebObjectsApplication. To get full support for this project type, you must manually load a bundle into Project Builder. Follow these steps:

1. Open Project Builder's Preferences panel (Info->Preferences on Mach, Tools->Preferences on Windows NT).

2. Choose Bundles in the pop-up list.

3. Click the Add button.

4. Navigate to **NextDeveloper/PBBundles/WebObjectsSupport.bundle**, and click OK.

5. Quit and restart Project Builder.

Server Compatibility

WebObjects has been tested in these configurations. In addition, WebObjects should work with other servers, provided they follow the CGI or NSAPI specifications. The numbers in parentheses are references to bugs listed in the next section.

- Netscape + CGI Adaptor (Windows NT Workstation or Server and UNIX platforms)

- Netscape + NSAPI adaptor (Windows NT Workstation or Server and UNIX platforms)

- IIS Server + CGI Adaptor (Windows NT Workstation or Server) (install as 65135)

- IIS Server + ISAPI adaptor (Windows NT Workstation or Server) (install as 65135)

- Website 1.1D and greater (Windows NT Workstation or Server)

- Purveyor (Windows NT Workstation or Server)

- NCSA, CERN, Apache, + CGI (UNIX platforms)

Known Bugs

This section lists bugs that we are aware of with this release and suggests ways to avoid or work around these problems. Please refer to the bug reference number if you need to contact NeXT about a bug's status.

Installation

Reference: 72353

Problem: Cannot use forward slashes when specifying path names in Windows NT installer.

Description: When installing WebObjects on Windows NT, you are prompted for your server's cgi-bin and document root paths. If you choose to type in the path without going through the file system browser, you cannot type the path with forward slashes, as in **C:/cgipath**. If you do, the paths will become corrupted in the registry.

Workaround: Use backslashes when typing paths in the installer, as in **C:\cgipath**.

Reference: 74344

Problem: On Windows NT, there is no *NeXT_Root***/NextLibrary/WOApps** directory.

Description: The autostart mechanism searches /NextLibrary/WOApps for WebObjects executables, but this directory is not created on Windows NT.

Workaround: Create the *NeXT_Root***/NextLibrary/WOApps** directory yourself. This directory is simply a place where you can install your own WebObjects application executables for compiled applications. When you first install WebObjects, the directory is empty.

Reference: 73185

Problem: Windows NT Installer fails if cgi-bin or document root contain spaces

Description: When installing WebObjects on Windows NT 4.0, installation will fail if your cgi-bin directory or document root directory has a path containing any white space.

Workaround: Change your paths so that they do not contain any white spaces.

Reference: 73203

Problem: License file problem when adding a category to EOAccess in WebObjects Enterprise.

Description: If you have a framework that implements a category on a class in the EOAccess framework, NSBundle may incorrectly return your framework when **bundleForClass:** is invoked. If so, Enterprise Objects won't be able to find its license file, and the appplication is essentially downgraded to WebObjects Pro or WebObjects Lite.

Workaround: Copy the file from **EOAccess.framework/Resources/License.table** into your framework's **Resources** directory.

Reference: 73668

Problem: PDO Apache installation conflicts with PDO WOF installation.

Description: The Apache installation script wants to install Apache to **/usr/NextLibrary**. WOF doesn't install things in **/usr** anymore, and it creates a symbolic link in **/usr** to the real location of **NextLibrary**.

Workaround: Install the Apache server after installing WOF. Note that you must know what the cgi-bin

and document root will be ahead of time, as the WOF installation script prompts you for this information.

Reference: 74568

Problem: On Solaris, installer doesn't copy the CGI adaptor to cgi-bin.

Description: The Solaris WebObjects Deployment package installer doesn't copy the CGI adaptor to your server's cgi-bin directory.

Workaround: Copy the file *NeXTRoot*/**NextLibrary/WOAdaptors/CGI/WebObjects** to your server's cgi-bin directory. (You define *NeXTRoot* at installation time.)

Reference: 74723

Problem: Enterprise Objects examples wrongly included.

Description: The WebObjects package erroneously includes some examples based on the Application Kit and Enterprise Objects UI layer (Customer, AssociationExample, AssociationPalette, Inventory, PointOfSale, Studios). These examples will not compile unless you have **AppKit.framework** and **EOInterface.framework**, which are part of the OpenStep product.

Workaround: Ignore these examples.

Reference: 69158

Problem: A running WebObjects application may erroneously complain that a new page or component is missing

Description: If you run a WebObjects application, and then add a new component to the application, and then try to access the new component's page in the browser, you receive a message saying the application cannot find the HTML template for the new component, even though the **.wo** directory and the **.html** template file both exist.

Workaround: Restart the WebObjects application.

While you are building and debugging a WebObjects application, you shouldn't use the autostart capabilities of the WebObjects adaptor. Instead, run the application from a shell and then kill and restart the process as pages and components are added.

WebObjects Adaptors

Reference: 65135

Problem: Cannot autostart WebObjects applications with Microsoft Internet Information Server (IIS)

Description: The Microsoft IIS Server currently does not work properly on Windows NT Workstation. If you install WebObjects in a directory other than the root directory, the CGI adaptor won't autostart your applications correctly. The IIS Server creates subprocesses with a special user that has no privileges and that has no access to the **NEXT_ROOT** environment variable. An autostarted application won't have any privileges either and won't be able to

locate NeXT resource files (for example, time zone files).

Also, sometimes the IIS Server does not recognize the WebObjects adaptor unless you specify the **.exe** extension. This means that an URL to start a WebObjects application might have to be:

http://localhost/Scripts/WebObjects.exe/*MyApp*

Workaround: Do one of the following:

- Install NeXT software under the root directory. When the installer asks you for a directory in which to install NeXT software, delete the **C:/NeXT/** path from the field.

- Go to your WWW Server configuration panel and change the anonymous user to a user with root privileges. This will allow you to autostart web applications from the browser.

- If you want to keep the anonymous user for security reasons, start your WebObjects applications by hand.

Reference: 73392

Problem: Microsoft's IIS server: Autostarted applications can't be killed.

Description: You can't use TaskManager to kill autostarted applications on Windows NT with the IIS server. By default, the IIS server runs autostarted applications as a special user with no privileges. Consequently, you don't have permission to kill these processes.

Workaround: Do one of the following:

- Go to your WWW server configuration panel and change the anonymous user to your user name and password. Be careful to type the password correctly as you are not warned if you enter the wrong password. Reboot your computer for the change to take effect.

- If you want to keep the anonymous user for security reasons, start your WebObjects application by hand. During application development starting by hand is generally the better course, so you can observe debugging messages in the terminal window.

Reference: 72826

Problem: ISAPI adaptor doesn't behave correctly when URL is missing WOApp name.

Description: The ISAPI adaptor will not come back with a "missing application name from URL" error message when you pass it an incomplete URL like:

http://host/Scripts/WebObjects-ISAPI.dll

You will just get a blank page.

Workaround: None. Inoffensive.

Reference: 67984

Problem: Apache Server on HP-UX: Can't change user to "nobody"

Description: By default, the Apache server sets the user to nobody when launching cgi-bin processes. However, the UID of nobody is -2, which causes setuid to complain about an invalid argument.

Workaround: Change the nobody UID and nogroup group ID in **/etc/passwd** and **/etc/group** to positive numbers.

Reference: 69715

Problem: Cannot disable autostarting functionality.

Description: There is no way to turn off the autostarting functionality of a WebObjects application stored under the document root. That means that outside users may be able to run undesired applications, such as installed examples.

Workaround: Before deploying an application, clean the site of any undesired application. To disallow autostarting of an application, store it in *NeXT_Root*/**NextLibrary/WOApps**, and rename the application executable. If the application is scripted, rename **WODefaultApp** to any other name. If the application is compiled, rename the executable so that it's name is different from the **.woa** name. (See also 73208)

Reference: 72343

Problem: Application is inaccessible if incorrect application side adaptor is specified.

Description: By default, WebObjects applications use WODefaultAdaptor as the adaptor class. If you have created your own adaptor class, you can specify it on the command line using the

-a option. If the application executable cannot find the specified adaptor class at runtime, the WebObjects application is unreachable. This means if you don't capitalize the class name properly or if you misspell it, your application will be inaccessible and will provide no reason for failure.

Workaround: Check the name you specified with the **-a** option and verify that it is correct.

Reference: 72341

Problem: Adaptor mode no longer supported

Description: WebObjects 2.0 had a server HTTP adaptor mode that allowed you to contact a specific instance of an application using URLs of the following format:

`http://.../WebObjects/`*AppName*`:`*instanceNumber*`@`*hostName*

In 3.0, the instance number is an optional attribute that the user can no longer specify.

Workaround: None.

Reference: 73208

Problem: Clients can browse source code within *Document_Root* **/WebObjects** unless you take steps to prevent them.

Description: Users could view the source code of your WebObjects applications unless you deny them access or move your application. For example, if someone enters this URL

```
http://your_server/WebObjects
```

Workaround: Either of these two approaches will protect your source from prying eyes:

1. Configure your web server to disallow read access to WebObjects-related files (*.wos, *.wod, *.plist, etc.). This won't affect the operation of WebObjects applications, which access the file system directly, but will prevent the server from dispensing these files to browsers.

2. For WebObjects 3.0, move your WebObjects applications from the server's document root to *NeXT_ROOT***/NextLibrary/WOApps**.

If you use the second approach, you must move the entire application (the ".woa" directory) to *NeXT_ROOT***/NextLibrary/WOApps**. However, if you have any statically linked resources such as sound or image files (for example, <IMG src="../Images/logo.gif">, you must leave a sparse copy of your application under the document root. In this case "sparse" means that the application's directory structure is reproduced in the document root, but the only files it contains are the static resources that the server must dispense to a client's browser.

If you choose to move a scripted application to *NeXT_ROOT***/NextLibrary/WOApps**, you must take one further step. Each application in the **WOApps** directory must contain an executable, but a scripted application located in the server's document root relies on the default application *NeXT_ROOT***/NextLibrary/Executables/WODefaultApp** (or **WODefaultApp.exe** on Windows NT) as its executable. So, when you move the scripted application into WOApps, copy the default application into your scripted application, and then rename this copy with the name of your application.

WebObjects Builder

Reference: 69669

Problem: WebObjects Builder does not support languages other than WebScript.

Description: WebObjects Builder automatically generates a **.wos** file even if the component's code is being written in another language.

Workaround: Manually delete the **.wos** file within the **.wo** directory, then add the code file you need.

Reference: 73162

Problem: Consistency Check assumes unknown names in WebScript are Class references.

Description: The Consistency Check command assumes any unknown name to be a reference to a Class name. For example, in

```
[MyClassName someMethod];
```

since MyClassName is not known, it is assumed to be a Class reference. Class references are resolved at run time, so the Consistency Check command does not inform you of the unknown reference.

Workaround: None.

Reference: 73881

Problem: Double-clicking to bind to a Client-Side Java Component overwrites code attribute.

Description: If you add a Java component from the Client-Side Component palette to a .wo, and then double-click on something in the object browser, the code attribute will be replaced, and the applet won't know what type it is any more.

Workaround: Either undo the double-click (using Edit->Undo), or remove the WOApplet from the .wo and re-drag the component from the palette.

Reference: 74096

Problem: Can't double-click a method to bind to it.

Description: Select a dynamic element that ought to bind methods (WOSubmitButton, for example) and double-click a method. The system beeps, and the message "No suitable default binding" appears. This only happens if the method you double-clicked is in the first column of the object browser; for example, you can correctly bind to methods within display groups.

Workaround: Bind methods using the bindings inspector.

Reference: 72229

Problem: Browse button in Preference panel doesn't select directories on NT.

Description: If you click on the Browse button in the Preferences panel to specify the document root, the Open panel doesn't select directories.

Workaround: In the Open panel, select a file inside the directory you want to select then click OK. After the Open panel is dismissed, edit the text field to remove the file name and leave only the directory name.

Reference: 69976

Problem: There is no way to use a WOString for a title

Description: Because the document's title is set in the inspector, there is no way to use a WOString for the title.

Workaround: Open the **.html** file directly, create a WOString within the document's title, then set its bindings appropriately in the **.wod** file.

Reference: 70765

Problem: Can't add an empty list.

Description: If no text is selected, the Format->List->New List menu item doesn't do anything.

Workaround: Type a few characters, select them, then choose Format->List->New List; the characters appear as part of a list. You can now select the characters and delete or type over them. Alternatively, drag a list from the Static Elements palette to produce an empty list.

Reference: 71070

Problem: Selecting inserted HTML element and typing replaces element.

Description: Drag a heading off the Static Elements palette into a component, and then go to the component window and start typing. The whole element is deleted, rather than just the contents.

Workaround: Before typing, make sure only the text within the heading is selected; if the selection goes beyond the heading (including either of the new-lines before and after the heading), the heading will be deleted.

Reference: 74314

Problem: Adding a dynamic element while the binding text field of another dynamic element ruins the component's display.

Description: If the selection is inside the binding text field of an abstract element (within a WOString, for instance), using any of the Format->Abstract Element menu commands will destroy the component's display.

Workaround: Select Edit->Undo until the component's display is restored. Now select the entire element by clicking one of its icons; the Format->Abstract Element menu items should behave as expected now.

WebObjects Builder: HTML Support

Reference: 68067

Problem: Pasting RTF does not work very well

Description: Sometime, spaces and attributes are lost when you paste RTF text into WebObjects Builder.

Workaround: None.

Reference: 68068

Problem: Pasting RTFD does not work

Description: Attachment (images) are not preserved.

Workaround: None.

Reference: 68816

Problem: Implementation of anchors is incomplete.

Description: WebObjects Builder supports local anchors but does not provide any way to edit them besides custom markers and generic inspectors.

Workaround: Use the custom HTML marker to create the anchor and edit it in that marker's inspector.

Reference: 69495

Problem: It is difficult to see a form's boundaries.

Description: It is not always obvious if you are editing inside or outside of a form.

Workaround: To see the form's boundaries, click near one of the controls, and a dashed line will appear. At this point, you can be sure that the editing will take place inside the form.

Reference: 71374

Problem: Adding a form after setting text attributes occasionally creates bad display.

Description: Enter some text, select it, and increase the font size. Now drag in the form from the palette. The fields will initially be in the big font from before. Select the text and make it smaller. The display of the whole form is wrong now.

Workaround: When you first encounter the bad display of the form, choose Edit->Undo from the menus until the form displays properly. Now select the entire form and choose Format->Text->Plain Text from the menus. You should now be able to set the font attributes within the form as you wish.

Reference: 72504

Problem: Horizontal line appears inside a heading.

Description: Drag a heading from the Static Elements palette, edit it, then drag a horizontal line from the palette. The line ends up inside the title.

Workaround: Delete the horizontal line, and then move the text selection to the beginning of the line following the heading. (The dotted line around the heading should disappear.) Now drag the horizontal line from the palette; it will appear outside of the heading.

Reference: 72550

Problem: Cut and paste from part of a list does not work

Description: If you copy part of a large list, then paste outside of a list, the structure of the list is lost. Instead, you get only the text from the list without bullets or carriage returns.

Workaround: Select the entire list, copy and paste it, and then remove the pieces you do not want from the new copy.

Reference: 73282

Problem: Cannot create a reusable component with no form in it.

Description: In WebObjects Builder, it's impossible to create a component that has form elements in it but does not have a surrounding <FORM> tag.

Workaround: Edit the **.html** manually to remove the <FORM> tags.

Reference: 73541

Problem: Existing components that have two dynamic elements with the same name confuse WebObjects Builder.

Description: Some legacy WebObjects applications have components that contain references to two dynamic elements with the same name; the **.wod** file then had a single entry for that name, which both dynamic elements shared. While this worked under WebObjects 2.0,

it was never supported; all dynamic elements within a document should have unique names. If such a document is opened and saved inside of WebObjects Builder, WebObjects Builder produces an unusable document.

Workaround: None. Edit the component's HTML file manually to remove the duplicate reference.

Reference: 73653

Problem: Cannot change the top-level marker from <BODY> to another marker.

Description: The HTML expert inspector does not allow you to change the top-level marker (the <BODY> marker). This is particularly important when trying to create a page with frames, where the top-level should be <FRAMESET>.

Workaround: None. A <FRAMESET> page must be created manually.

Reference: 73726

Problem: Dragging a color onto selected text does not work well if the selected text has mixed fonts.

Description: If the selected text has mixed fonts (different font sizes or different font attributes like bold and italics), dragging a color from the Color Panel to the selection does not always work; sometimes only part of the selected text gets the color, instead of all of it.

Workaround: Drag to the color well in the toolbar instead of to the selection.

WebObjects Builder: Image Support

Reference: 69412

Problem: Cannot create image maps

Description: There is no way to create image maps in WebObjects Builder.

Workaround: For client-side image maps, create the map description using the generic HTML tag on the Static Elements palette. Then select the image, open the inspector, and click HTML Expert in the inspector. In HTML Expert mode, you can add the attribute necessary to reference your map. For server-side image maps, edit the attributes of the image as above, then use a text editor to create the map file.

Reference: 73574

Problem: Resources in untitled documents don't work very well.

Description: If you have an Untitled document (that is, a document that has never been saved), manipulating resources from WebObjects Builder does not work properly. For example, if you create an active image and then set the button title in the inspector, the image in the document is not updated.

Workaround: Save the document before adding resources so that there is a location in the file system where resources can be manipulated.

WebObjects Builder: Inspectors

Reference: 70071

Problem: Sometimes the structure selection in the text is out of sync with the inspector.

Description: When you type, the smallest structure around the insertion point is selected (gets the dotted rectangle), which is not necessarily the same as the structure selected in the inspector.

Workaround: In the inspector, click the icon of the structure you want to select (even if it is already highlighted); the selection drawn in the main window will update to match the inspector.

Reference: 70117

Problem: Undo doesn't work while editing classes.

Description: If you are changing a type in the Classes window and select Undo without pressing return to end editing in the inspector, the Classes window ends up in an invalid state, and potentially the underlying types are set incorrectly as well.

Workaround: Press return before selecting Undo.

Reference: 73370

Problem: Bindings occasionally go to the wrong element when double-clicking.

Description: If you have a dynamic element imbedded in another dynamic element and you try to bind

a variable to the outer element by selecting the outer element and double-clicking the variable, sometimes the binding erroneously goes to the inner element. This can occur when the outer dynamic element contains nothing but the inner element. That is, there are no characters between the start of the outer element and the start of the inner element or between the end of the inner element and the end of the outer element.

Workaround: Open the inspector and select the bindings inspector for the inner web object. Delete the erroneous binding. Now select the bindings inspector for the outer web object and create the desired binding. Alternatively, put a character between the beginning of the outer web object and the beginning of the inner one; double-clicking will work as expected. Later, delete the extra character.

Reference: 73485

Problem: HTML Expert inspector is easily confused

Description: If the inspector is left in HTML Expert mode, it occassionally loses track of the current document selection. Once this has occurred, the display of the expert inspector can become wildly inaccurate, and the expert inspector is no longer usebable.

Workaround: Switch the inspector to its normal mode then back to the HTML Expert mode; this should synchronize it with the current document selection. However, the HTML Expert mode should only be used briefly, to make a small change directly to the HTML.

Reference: 73569

Problem: WOBrowser inspector is misleading.

Description: WOBrowsers have inspectors to allow you to set the options within the browser, but this is only relevant if you are creating a purely static SELECTPA WOBrowser will ignore them.

Workaround: None.

Reference: 73805

Problem: Binding the **string** attribute of a WOHyperlink does not remove the body of the WOHyperlink in the HTML

Description: Add a WOHyperlink to a component, and then bind its **string** attribute to anything. The word "Hyperlink" is not removed from within the WOHyperlink, and when the application is run, the link shows up with the word "Hyperlink" followed by the value of the **string** attribute.

Workaround: Delete the word "Hyperlink" between the WOHyperlink's icons in the document.

Reference: 73891

Problem: Binding inspector occasionally does not display all bindings.

Description: If the binding inspector is open, and new bindings are created without using the inspector (for instance, by double-clicking a variable), the binding inspector occasionally does not display the newly-created binding.

Workaround: Click another button in the inspector's path, then go back to the binding inspector; this should update the binding inspector's display. If this does not work, try closing and

reopening the inspector.

WebObjects Builder: Palettes

Reference: 68866

Problem: Placing form objects outside of a form generates a new form.

Description: Currently, if you place controls individually on a page without placing a form object first, you get a separate <FORM>...</FORM> around each control.

Workaround: None.

Reference: 69491

Problem: Form Elements palette is not complete.

Description: There are many form functions that cannot be accessed from the palette, including adding a submit or reset button, creating a completely empty form, or combining several form elements into a single form.

Workaround: Use the entries in the Format->Form menu, which provides all of the above functionality.

WebObjects Builder: Application Windows

Reference: 72429

Problem: Save As is not enabled for the application window

Description: You cannot use File->Save As to rename an application.

Workaround: Save the application and all components within it. Then copy it in the file system.

Reference: 74247

Problem: WebObjects Builder has no direct interface to Project Builder.

Description: Unlike Interface Builder, WebObjects Builder currently does not notify Project Builder when application documents are edited. In addition, WebObjects Builder does not currently provide users with a way to add resources to Project Builder's PB.project file.

Workaround: Use Project Builder to add resources to web-based projects.

WebObjects Builder: Script Window

Reference: 73056

Problem: Cannot set tab width for script window

Description: The tab width used in the script window is fixed. You cannot set it yourself.

Workaround: Perform this command in a shell or terminal window:

`defaults write WebObjectsBuilder TabWidth` *characters*

to set the tab width to *characters* number of characters.

Reference: 73647

Problem: Find panel only works in HTML display of component window

Description: The Find panel only searches the main component window's HTML display. You cannot use the Find panel in the script window.

Workaround: To perform searches in the script window, use the keyboard equivalents (Command-e to enter the selection, Command-g to Find Next, Command-d to Find Previous). Note that on Windows NT, the key equivalents are Control-e, Control-g, and Control-d, respectively. There is no workaround for doing Find/Replace operations in the script window.

WebObjects Builder: Dynamic Elements Support

Reference: 70141

Problem: WOStrings should pick up font attributes.

Description: If you make a WOString bold, the visible contents of the WOString do not display in bold. This makes it difficult to see the font attributes on a WOString.

Workaround: Click within the WOString and look at the toolbar. The highlighted buttons correctly

portray the font attributes of the string.

Reference: 72313

Problem: Attributes get lowercased when copied or when put on a palette.

Description: When you copy and paste or drag an element from the palette, all of the attributes that appear in the binding inspector will be forced to lowercase letters. In certain circumstances, this may cause duplicate keys to appear in the binding inspector.

Workaround: Choose attribute names that are all lowercase.

Reference: 73707

Problem: Components can't access Application or Session data if the application window is closed.

Description: If you open a component that is inside of a **.woa**, the corresponding application window is automatically opened. At this point, application and session variables and methods are accessible through the components Object Browser. If, however, you close the application window, the application and session information will no longer be accessible by the component.

Workaround: Leave the application window open.

Reference: 74059

Problem: WebObjects Builder adds quotes around constant values.

Description: If you use the bindings inspector to set an attribute's value to a constant, save the application, and then close and reopen it, WebObjects Builder adds quotation marks around your constants. If you save the component again, the quotation marks are written out to the component's **.wod** file.

In many cases, this is harmless. For example, if you bind the **multiplesubmit** attribute of WOForm to the constant 0, it will eventually be written to the **.wod** file as

```
multiplesubmit = "0"
```

When the framework evaluates **multiplesubmit**, it will send **intValue** to the string "0", which returns the integer value 0.

However, this is a problem if you use defined constants such as YES and NO. If you set **multiplesubmit** to NO in the Builder and it is written to the **.wod** file as

```
multiplesubmit = "NO"
```

The framework sends **intValue** to the string "NO", which does not return 0.

Workaround: Always use 1 and 0 in places of YES and NO.

WebObjects Builder: Database Integration

Reference: 73566

Problem: Dragging and EOModel from a **.woa** onto itself deletes the EOModel.

Description: If you have previously added a WODisplayGroup by dropping an EOModel file on your **.woa**, you will have a copy of that EOModel file in your **.woa**. If you subsequently drop that EOModel file onto the same **.woa**, WebObjects Builder tries to recopy the EOModel but deletes the file in the process.

Workaround: You can move the file outside of the **.woa** first before copying or copy from the original place that the EOModel file came from.

Reference: 74399

Problem: If eomodel is a link, WOBuilder copies the link, not the original file.

Description: This is only a problem on UNIX systems. If the user selects a symbolic link to an eomodel file, rather than the original file itself, the symbolic link is copied into the component. This is particularly troublesome when working with a relative link, since the link will not be valid in the new location.

Workaround: Find and select the original file, rather than the link.

WebObjects Builder: Language Support

Reference: 75922

Problem: If your language preference (as set in the Preferences application) is set to something other than Japanese, it's not possible to enter Japanese characters.

Description: On OPENSTEP for Mach, most applications automatically switch fonts if you type a character that is unrepresentable in the current font (for example, typing a Japanese character into TextEdit when Helvetica is the current font). WebObjects Builder doesn't do this, however.

Workaround: When you run WebObjects Builder with a language preference set to something other than Japanese and want to type Japanese characters, you have to manually set the font in Builder's Preferences panel to a Japanese font before entering the Japanese text.

WebObjects Framework

Reference: 66089

Problem: WebObjects applications on Solaris cannot access environment variables

Description: If you autostart a WebObjects application on Solaris, the process is owned by user nobody. nobody has no user environment, and thus does not have access to the **NEXT_ROOT** environment variable, meaning it won't be able to locate NeXT resources files (for example, time zone files).

Workaround: Do one of the following:

- Install NeXT software under the root directory.

- Change your web server's configuration so that CGI process are launched by a user with the appropriate environment set up rather than user nobody. This will allow you to autostart web applications from the browser.

- If you want to keep the nobody user for security reasons, start your WebObjects applications by hand.

Reference: 73419

Problem: Project Builder does not add **.woa** extension

Description: When you create a project of type WebObjectsApplication in Project Builder, Project Builder does not append the extension **.woa** to the name you specify. WebObjects cannot recognize a WebObjects application without the **.woa** extension.

Workaround: Load **WebObjectsSupport.bundle** as described in "Special Notes" at the beginning of this file.

Reference: 73644

Problem: Can't backtrack in applications that access a database.

Description: If you try to backtrack using the browser in a database application, it doesn't work because the WODisplayGroup object is out of sync with the display.

Workaround: Disable browser backtracking using WOApplication's **setBacktrackingEnabled:** method.

Reference: 73745

Problem: Externally editing **.woo**, **.wod**, or **.wos** files created by WebObjects Builder can cause conflicts

Description: If you rename an instance variable using an editor other than WebObjects Builder on a component created by WebObjects Builder, it can cause your application to become non-functional. At run time, you receive a message that the variable does not exist. The problem is that the information in the **.woo** file (a file maintained internally by WebObjects Builder and the WebObjects Framework) is now out of sync with the file you edited.

Workaround: Always use WebObjects Builder to edit the script file. Do not edit the **.wod** file. If you encounter the error message, delete the **.woo** file.

Reference: 74753

Problem: **appendContentHTMLString:** header file comments incorrect.

Description: In the file WOResponse.h, the descriptions of **appendContentString:** and **appendContentHTMLString:** are swapped. That is, if you have the string @"<b>" and you want it shown in the browser literally, you should use **appendContentHTMLString:**, which converts it to <b>. If you want the string interpreted as a bold tag, use **appendContentString:**.

Workaround: Use the API as described above.

WebObjects Framework: Request Handling

Reference: 71990

Problem: Cannot cancel a **WODefaultApp** using the ORACLE adaptor

Description: If you run have a WebObjects application that uses the ORACLE adaptor and you run it from the command line, you cannot use Control-C or Control-Z to kill the executable. The Oracle client library traps all signals, including Control-C and Control-Z, so the executable never receives the signal.

Workaround: Use **kill -9** to kill the executable.

Dynamic Elements

Reference: 64906

Problem: WOCheckbox **checked** attribute doesn't work as expected

Description: WOCheckBox's **checked** attribute should return YES or NO as specified in the API, but it actually returns **self** or **nil**.

Workaround: Test the checked attribute for **nil** or non-**nil** values instead of NO or YES. You may also want to write your own checkbox component that returns YES or NO instead of **self** or **nil**.

Reference: 66845

Problem: Imagemap files can only reside in the **.wo** directory

Description: Image map files used with WOActiveImage are assumed to reside in the **.wo** in which they are referenced. No indication is given by WOActiveImage when it cannot find this image map file.

Workaround: Make sure the imagmap file is saved in the **.wo** in which it is referenced.

Reference: 74208

Problem: WOBrowsers and WOPopUpButtons cannot have contents with trailing spaces.

Description: WOBrowser and WOPopUpButton do not work properly if their content strings contain trailing spaces.

Workaround: Filter out the trailing spaces before you send the string to the dynamic element.

Reference: 73112

Problem: WebObjects Builder includes <HTML>, <HEADER>, and <BODY> tags in reusable components

Description: WebObjects Builder inserts <HTML>, <HEADER> and <BODY> tags in all HTML templates. If you use WebObjects Builder to create a reusable component, WebObjects will generate them the <HTML>, <HEADER>, and <BODY> tags into the parent page at the reusable component's location, meaning that a page might have any number of

<HTML>, <HEADER>, and <BODY> tags. Netscape Navigator and Microsoft Internet Explorer browsers ignore the redundant tags. However because this is illegal HTML, there is no guarantee that all browsers will behave properly.

Workaround: Outside of WebObjects Builder, remove the redundant tags from the reusable component's HTML template file.

Reference: 74498

Problem: URL completion at run time is broken on a few dynamic elements if your attributes are lowercase.

Description: Sometimes, the WebObjects Framework is not able to complete a path or an URL if you supply a relative path. This affects only a few dynamic elements and only if the attribute names are provided in lowercase letters. See the workarounds below for specific instances of this bug and the ways to work around each instance.

Workaround: WOGenericContainer/WOGenericElement:
If you use an attribute in a WOGenericContainer or WOGenericElement whose value should be completed, the value for the **elementname** attribute and the keys for the attributes that need to be completed must be in uppercase. For example, a WOGenericContainer representing a hyperlink should be:

```
WOGenericContainer {elementname="A"; HREF="foo.html"};
```

WOApplet:
WOApplet cannot complete the paths for the attributes **code** and **codebase**. There are three possible solutions :

- Hard code the complete paths for both attributes.

- Use a WOGenericContainer (with uppercase elementname "APPLET" and uppercase keys **CODE** and **CODEBASE**). If you use WOGenericContainer, you won't be able to use the client-side component features of WOApplet.

- Subclass WOApplet into MYWOApplet. In MYWOApplet, override the following method as shown:

```
- (NSString *)elementName {
     return @"APPLET";
}
```

and then provide the attribute names in uppercase letters (**CODE** and **CODEBASE**).

WOImage:
Completion works fine for WOImage's **src** attribute, but not for **dynsrc** and **usemap**. If you use these **dynsrc** or **usemap**, you should use a WOGenericElement with the **elementname** value in uppercase and the attributes keys in uppercase. For example:

```
WOGenericElement {
     elementname="IMG";  SRC="foo";
     DYNSRC="foo1";
     USEMAP="foo2"
};
```

Client-Side Components

Reference: 73905

Problem: Client-Side Components may not work with some browsers

Description: Java Client-Side Components in WOF 3.0 have been tested with both Netscape Navigator 3.0 and Internet Explorer 3.0. Earlier versions or prereleases of these browsers are not guaranteed to work with Client-Side Components. In particular Beta releases 5 and 6 of Netscape Navigator 3.0 had Java run-time bugs that prevent the proper functioning of Client-Side Components.

Workaround: None.

Reference: 73877

Problem: Values changed in **awake** won't get noticed as changed when synchronizing client-side components

Description: Server-side snapshotting of client-side component attribute values is done after the parent component's **awake** method. If you have a variable bound to a client-side component attribute and you modify the value of that variable in your page's **awake** method, the server-side snapshot will reflect new value. The original value of the attribute will not be preserved in the snapshot, and thus the change will not be noticed when synchronization occurs. In fact, the only changes you can really be sure will be noticed are changes you make to the variables during an action.

Workaround: Don't modify variables bound to client-side component attributes except in your action methods.

Reference: 74080

Problem: Server side snapshotting of client-side components captures values, not copies

Description: When the server-side snapshot is taken of a client-side component, the actual values of the variables are stored. If one of those variables is a mutable object and you make a change to that object's value (for example, if you add or change a key in an NSMutableDictionary), it will change the value in the snapshot as well. The original value is not preserved in the snapshot, and future synchronization will not notice that the value is changed.

Workaround: Always make a copy of the variable before modifying it. For example if **paramDict** is a mutable dictionary, make changes to it in this way:

```
        paramDict = [paramDict mutableCopy];
[paramDict setObject:@"Some new val" forKey:@"myKey"];
```

WebObjects Framework: HTML Generation

Reference: 74772

Problem: WebObjects treats text inside <script></script> tags as HTML.

Description: When dynamically generating HTML, WebObjects tries to display the text inside of a <script> tag as HTML. If you use a "<" or ">" operator in the script, WebObjects converts

the operator to < or >, respectively, causing the script to fail on the client.

Workaround: To include script in a component, use the WOJavaScript or WOVBScript dynamic element.

WebObjects Framework: State Management

Reference: 66834

Problem: WebObjects applications that store state in the server are not indexed well by internet indexing robots.

Description: When indexed by the robots of internet indices like AltaVista, WebCrawler, or Yahoo, WebObject applicationss that keep state in the server will have many entries (sometimes hundreds, depending on the robot's crawling algorithm). For example, the "Surfshops" page of the CyberWind examples would show up under URLs such as these:

http://XXX/cgi-bin/WebObjects/Examples/CyberWind@xxx/Main.wo:1.395502469.0$Main.2.3.8.0.1

http://XXX/cgi-bin/WebObjects/Examples/CyberWind@xxx/Main.wo:1.395502469.2$Main.2.3.8.0.1

http://XXX/cgi-bin/WebObjects/Examples/CyberWind@xxx/Main.wo:1.395502469.4$Main.2.3.8.0.1

: : : : : :

Clicking on the links corresponding to these URLs will typically not generate useful results, since the server state the URLs refer to has usually timed out by the time the link is clicked. If server state is not being timed out in a particular application, the many hits

an indexing robot can generate may lead to that application being overwhelmed by the state it has to keep.

This problem only occurs with applications that keep state in the server. Also, for an application to be found by an indexing robot, some URL in that application has to be referenced, directly or through intervening pages, by a page that is submitted for indexing.

Workaround: To prevent indexing of all WebObject applications on a site, create a file called **robots.txt** in the document root of the site's web server, and put lines like the following in the file:

```
# Disallow robots from indexing all WebObject apps.
User-agent: *
# All WebObjects URLs on this site start with this.
Disallow: /cgi-bin/WebObjects
```

To prevent indexing of particular apps, use a **robots.txt** file similar to this one:

```
# Disallow robots from indexing certain WebObject apps.
User-agent: *
# Disallow indexing of WebObject examples.
Disallow: /cgi-bin/WebObjects/Examples
# Disallow indexing of one app that has server state.
Disallow: /cgi-bin/WebObjects/MyApp
```

In your version of **robots.txt**, replace "/cgi-bin" by the path of your cgi-bin directory (for example, "/Scripts").

If an indexing robot indexes your WebObjects apps in spite of a **robots.txt** file, or for more information, see **http://info.webcrawler.com/mak/projects/robots/robots.html**.

Reference: 72951

Problem: CookieSessionStore size limitations

Description: Due to the size limitation on cookies and the limit on environment variables for the CGI interface on certain systems, the cookie session store is only useable for cookie session state that has less than 2K in size. Note that the developer is still able to use cookies programatically -- this limitation is only on CookieSessionStore.

Workaround: None.

WebScript

Reference: 67852

Problem: WebScript cannot talk to NSProxies

Description: If you obtain a proxy in WebScript and try to send messages to the proxy, you will get the error message "Unknown type 16."

Workaround: Use Objective-C instead of WebScript.

Reference: 73843

Problem: Boolean expressions evaluate incorrectly if both sides do not evaluate to a number.

Description: If you have a Boolean expression such as the following:

```
id order = [[self session] order];
if (order && [order count] > 0) {
  /* do something here */
}
```

where one side of the expression evaluates to a number (0) and the other side evaluates to **self** or **nil**, the expression will always evaluate to true.

Workaround: Rewrite as follows:

```
id order = [[self session] order];
if (order) {
  if ([order count] > 0) {
    /* do something here */
  }
}
```

Reference: 73867

Problem: Variable argument lists don't work in modern WebScript syntax.

Description: You can't use methods that have a variable number of arguments (such as **logWithFormat:** or NSString's **stringWithFormat:**) in the modern syntax.

Workaround: Use the Objective-C syntax style. That is, instead of

```
self.logWithFormat(@"I am %@", self);
```

use this:

```
[self logWithFormat:@"I am %@", self];
```

Reference: 74070

Problem: WebScript sends **retain** too early if you use **alloc**/**init**.

Description: If you have the following script:

```
- awake {
    id var = [[MyClass alloc] init];
    // something
    [var release];
}
```

the WebScript engine sends the retain message to the object **var** prior to sending **init**. This causes no problems in most cases, but it might cause problems for some classes (for example, Foundation collection classes such as NSArray and NSDictionary).

Workaround: Avoid using **alloc** in WebScript wherever possible.

Reference: 74112

Problem: Unicode characters are not handled in WebObjects Builder.

Description: If a component contains characters that can't be represented in the standard encoding, WebObjects Builder refuses to save it.

Workaround: Use 8-bit ASCII characters only.

Reference: 74422

Problem: **dealloc** cannot be overriden in WebScript

Description: Overriding **dealloc** in WebScript causes two problems. First, all of the script's variables will be leaked. Second, some random/intermitent exceptions can occur due to the fact that a new scripted object instance may inherit the leaked instance variable

Workaround: If you need to override **dealloc**, you will need to rewrite your scripted object in Java or Objective-C.

WebObjects Examples

Reference: 74015

Problem: On Solaris, the Performance example requires Solaris Developer

Description: The Performance example requires files (for example, **stdio.h**, **stdlib.h**, **unistd.h**) that are provided on Solaris only if you have Solaris Developer installed. If you do not have Solaris Developer, compiling the Performance example will fail.

Workaround: Install Solaris Developer before installing WebObjects 3.0.

Reference: 74479

Problem: On PDO, database-aware examples won't work out of box with new database client libs.

Description: On PDO platforms, you must modify the DodgeDemo example's makefile for it to build successfully.

Workaround: Update your **Makefile.preamble-solaris-oracle** to be the following:

```
#
# Oracle Setup for solaris
#
FRAMEWORKS += -framework OracleEOAdaptor
#
# Path for the libs
#
OTHER_LDFLAGS += -L$(ORACLE_HOME)/lib

#
# Static linking
#
OTHER_LIBS += -lclient -lsqlnet -lncr -lsqlnet -lclient -lcommon \
              -lgeneric -lsqlnet -lncr -lsqlnet -lclient -lcommon \
              -lgeneric -lepc -lnlsrtl -lcv6 -lcore3 -lnlsrtl3 \
              -lcore3 -lnlsrtl3

#
# Dynamic linking
#
#OTHER_LIBS += -lclntsh
```

There are two environment variables that you should know about for linking and running applications with the Oracle Client Libraries on Solaris:

ORACLE_HOME
This environment variable is used in **Makefile.preamble-solaris-oracle**, so if you don't

have it defined, the linker won't find the oracle libraries. It is also used to find the **tnsnames.ora** file and similar files.

LD_LIBRARY_PATH = $ORACLE_HOME/lib
This environment variable is used by Solaris when you run an application that is linked against shared libraries. It only needs to be set if you link your WebObjects application against the Oracle Client libraries dynamically (The second option in **Makefile.preamble-solaris-oracle**).

The following are sample command lines for static linking and dynamic linking (with $ORACLE_HOME = /NeXT/app/oracle/product/7.3.2/)

Static:

```
host% make
== Making DodgeDemo for sparc-nextpdo-solaris2 ==
Pre-build setup...
Building...
/NextDeveloper/bin/gcc -o
/NeXT/WOF/NextDeveloper/Examples/WebObjects/DodgeDemo.woa/DodgeDemo
-L/NeXT/WOF/NextDeveloper/Examples/WebObjects/DodgeDemo.woa/obj-sparc-nextpdo-solaris2-opt
-F/NextLibrary/PrivateFrameworks  -L/NeXT/app/oracle/product/7.3.2/lib -arch sparc-nextpdo-
solaris2 /NeXT/WOF/NextDeveloper/Examples/WebObjects/DodgeDemo.woa/obj-sparc-nextpdo-
solaris2-opt/Car.o /NeXT/WOF/NextDeveloper/Examples/WebObjects/DodgeDemo.woa/obj-sparc-
nextpdo-solaris2-opt/CarLease.o
/NeXT/WOF/NextDeveloper/Examples/WebObjects/DodgeDemo.woa/obj-sparc-nextpdo-solaris2-opt/
CarQualifier.o /NeXT/WOF/NextDeveloper/Examples/WebObjects/DodgeDemo.woa/obj-sparc-nextpdo-
solaris2-opt/ShoppingCart.o /NeXT/WOF/NextDeveloper/Examples/WebObjects/DodgeDemo.woa/obj-
sparc-nextpdo-solaris2-opt/main.o             -framework WebObjects -framework EOAccess
-framework EOControl -framework Foundation -framework MultiScript -framework OracleEOAdaptor
-lclient -lsqlnet -lncr -lsqlnet -lclient -lcommon -lgeneric -lsqlnet -lncr -lsqlnet
-lclient -lcommon -lgeneric -lepc -lnlsrtl -lcv6 -lcore3 -lnlsrtl3 -lcore3 -lnlsrtl3
```

Dynamic:

```
host% make
/usr/ucb/echo == Making DodgeDemo for sparc-nextpdo-solaris2 ==
```

```
/usr/ucb/echo Pre-build setup...
/usr/ucb/echo Building...
/NextDeveloper/bin/gcc -o
/NeXT/WOF/NextDeveloper/Examples/WebObjects/DodgeDemo.woa/DodgeDemo
-L/NeXT/WOF/NextDeveloper/Examples/WebObjects/DodgeDemo.woa/obj-sparc-nextpdo-solaris2-opt
-F/NextLibrary/PrivateFrameworks  -L/NeXT/app/oracle/product/7.3.2/lib -arch sparc-nextpdo-
solaris2 /NeXT/WOF/NextDeveloper/Examples/WebObjects/DodgeDemo.woa/obj-sparc-nextpdo-
solaris2-opt/Car.o /NeXT/WOF/NextDeveloper/Examples/WebObjects/DodgeDemo.woa/obj-sparc-
nextpdo-solaris2-opt/CarLease.o
/NeXT/WOF/NextDeveloper/Examples/WebObjects/DodgeDemo.woa/obj-sparc-nextpdo-solaris2-opt/
CarQualifier.o /NeXT/WOF/NextDeveloper/Examples/WebObjects/DodgeDemo.woa/obj-sparc-nextpdo-
solaris2-opt/ShoppingCart.o /NeXT/WOF/NextDeveloper/Examples/WebObjects/DodgeDemo.woa/obj-
sparc-nextpdo-solaris2-opt/main.o             -framework WebObjects -framework EOAccess
-framework EOControl -framework Foundation -framework MultiScript -framework OracleEOAdaptor
-lclntsh
```

Documentation

Reference: 74966

Problem: Problems with WebObjects PDF files on Mach

Description: Some of the PDF files provided with WebObjects 3.0 documentation don't show up properly in Mach-based PDF viewers. In particular, the graphics might not show up or might be too dark to be legible. PDFView might not be able to open the file at all.

Workaround: Currently there is no known workaround on Mach. The best solution is probably to view the PDF files on another platform using the Acrobat PDF viewer. Adobe provides free PDF viewers for all other platforms supported by WebObjects. In the meantime, NeXT's Publications department is working to find solutions to this problem.

Reference: 75079

Problem: The **fullName** method in the Movies Database Tutorial application is broken.

Description: Binding the Movies Detail browser's **value** attribute to the **fullName** method as described in the "Creating a Web-Based Database Application With WebObjects Builder" chapter of the *Getting Started* book causes problems. When a Talent object doesn't have a last name, **fullName** returns a string that ends with a space (such as "Vampira "). The browser strips any trailing white space, so the browser entry doesn't match the corresponding array entry.

Workaround: Change the **fullName** method as follows:

```
- fullName {
    if (![talent.lastName length])
        return talent.firstName;

    return [NSString stringWithFormat:@"%@ %@", talent.firstName,
talent.lastName];
}
```

Reference: 75066

Problem: Model file for the Movies tutorial application is incomplete.

Description: If you follow the steps in the Movies Database Tutorial (the "Creating a Web-Based Database Application With WebObjects Builder" chapter in *Getting Started*), the resulting application has a bug: saving a newly inserted MovieRole fails. A new MovieRole object starts out with a **nil** primary key. Enterprise Objects Framework automatically attempts to generate a primary key but fails because the MovieRole entity has a compound

primary key. (Automatic primary key generation only works for primary keys that consist of exactly one attribute.)

Workaround: Open the model file and select the Movie entity's to-many relationship, **movieRoles**. In the Advanced Relationship Inspector, check Propagate Primary Key and Owns Destination. Also select the Cascade delete rule.
