WebObjects 3.0 Java Extensions Release Notes

The following release notes apply to the WebObjects Java Extensions. To access the on-line documentation, use the link provided on the WebObjects Home Page.

Installation Note

The Java Extensions examples that use the Enterprise Objects Framework (such as **MovieDemoJava**) may require a change to their EOModel files before they will operate properly. See Bug #75466 under "Enterprise Objects Framework," below, for more information.

New Features in WebObjects Builder

The WebObjects Java Extensions comes with a new release of WebObjects Builder. This WebObjects Builder contains the following improvements over the version on the WebObjects 3.0 CD-ROM:

- You can now write a component's logic in Java. To do this, you must set your language preference to Java before you create a new application. Go to the preferences panel, and choose "Java" in the Languages display. After your preference is set to Java, WebObjects Builder creates **.java** files where it used to create **.wos** files (for example, Application.java, Session.java, and Main.java). It also checks your Java syntax for you when you try to save the component.

- The ClientSideComponents palette (in *NEXT_ROOT*/NextDeveloper/Palettes/ClientSideComponents.wbpalette) has been improved. The palette now displays icons for each applet that look like the end-result applet (for example, a button applet looks like a button). In addition, WebObjects Builder now has a WOApplet inspector that lets you directly set

attributes for your applets.

Applets are now knob-resizable directly in the HTML and will look like the actual applet for all supported applets (those that are on the palette).

- You can now choose three different fonts in the Font Preferences panel. The first font (Document) is used to display most of the text inside of a component. The second font (fixed pitch) is used to display any fixed-pitch text inside of a component. The last font (Script) is used in the Script window to display your script file or Java file.

  The Font Preferences panel allows you to choose three different fonts: one for normal text in the component, one for fixed-pitch text in the component, and one for the script window. You can also set the tab width for the script window in this panel.

- WebObjects Builder now handles character encodings. When you use WebObjects Builder to create a component, it stores the encoding as one of the

component's attributes. If you don't explicitly specify the encoding, the Builder saves the component using the default character encoding. You can use the Set Save Encoding command on the File menu to change which encoding a component uses. You can also use the Set Save Encoding command to change the default encoding.

If you have a component that was created or edited outside of WebObjects Builder, WebObjects Builder might not know the correct encoding for that component. In such cases, use Open with Encoding to open the component. When you use this command, the first panel that appears is an open panel in which you select the component. After you click Open in the open panel, a separate panel appears in which you specify that component's encoding.

- You can now use the double-click shortcut to bind to a dynamic element to a method as well as to a variable. To use the shortcut, first select the dynamic element, then double-click the method you want to bind it to. For more information on binding dynamic elements, see the on-line book *Using WebObjects Builder*.

Troubleshooting

- If you're having trouble running a WebObjects application that's written in Java, first ensure that the HelloWorld example runs (the version that ships with WebObjects). Then, verify that the Java version, HelloWorldJava, runs. For instructions on how to run these examples, see the WebObjects and Java Extensions documentation.

- If you're having problems with a WebObjects application that uses the Enterprise Objects Framework, verify that you can access your database by using EOModeler to open the model and browse the database.

Known Bugs in This Release

Java Extensions

Reference: 73519

Problem: Constructor helper function doesn't work with floats on Solairs

Description: On Solaris, it isn't possible from Objective-C to instantiate a Java class using a constructor that contains one or more floats. This includes Java subclasses of Objective-C classes.

Workaround: None.

Reference: 74127

Problem: Spurious error messages displayed when Java loads libraries on Solaris

Description: **NextObject.loadLibrary()** tries to find the correct library by prepending a number of standard search paths to the supplied library name. For every

path that doesn't succeed, **ld.so** generates an error message similar to the following, even if the loading process eventually succeeds.:

```
ld.so.1: /opt/java/bin//../bin/sparc/java: fatal: JavaTest: can't
open file: errno=2 (JavaTest)
```

Assuming that the load process does eventually succeed, these spurious messages can be ignored.

Workaround: None.

Reference: 74941

Problem: Session.wos and Session.class crashes the Java bridge

Description: Having both a ".wos" and a ".class" file for the same class (such as Session.wos and Session.class) will cause your application to crash.

Workaround: Remove the ".wos" file.

Reference: 75000

Problem: You still need to know about Objective-C selectors when messaging from Java

Description: Some of the methods in the **next.util** package require that you know Objective-C selector names. The following code excerpt illustrates one such situation:

```
myVector = new MutableVector();
// fill the vector here
myVector.sortUsingMethod("compare:");
```

Workaround: None.

Reference: 75207

Problem: Cannot overload methods in hybrid Java subclass

Description: You cannot overload methods that are in "hybrid" Java classes (a "hybrid" Java class is a Java subclass of an Objective-C class). Thus, for instance,

the following code will cause your app to crash when the myMethod is invoked from Objective-C:

```
public class MyComponent extends Component {
    public void myMethod() {...}
    public void myMethod(int anint) {...}
}
```

Workaround: None.

Reference: 75022

Problem: Attempting to launch an uncompiled Java application can disable your server

Description: If you attempt to launch a WebObjects Java application before it has been compiled, it will cause errors on your web server and will leave the server in a state from which you'll be unable to launch any other WebObjects applications. Essentially, the WebObjects process that's trying to contact the uncompiled application holds on to the WebObjects.autostart file and

waits for five minutes, to prevent another concurrent autostart.

Workaround: If this happens, you'll need to kill the WebObjects process as you won't be able to remove the WebObjects.autostart file. Note that you may have to restart the web server in order to unlock the WebObjects.autostart file.

Reference: 75264

Problem: You can't specify the **.woa** extension when invoking an app from the command line

Description: When starting an application from the command line, the Java Extensions append a **.woa** extension to the application name you specify, whether or not you've supplied an extension. So for instance, if you try to invoke your app as follows:

```
$ ./Zowee -d c:/netscape/server/docs Zowee.woa
```

It won't start because WebObjects won't be able to find an app directory named "Zowee.woa.woa".

Workaround: Don't specify the **.woa** extension when starting WebObjects applications from the command line.

Reference: 75467

Problem: Subclasses of **next.wo.Component** shouldn't be given a package name

Description: Subclasses of **next.wo.Component** shouldn't be given a package name, or the **pageWithName()** method won't be able to find them. This is because **pageWithName()** creates the page by looking up and instantiating the component class with the same name as the page. If the Java class is given a package name, the class lookup will fail.

Workaround: None.

Reference: 75537

Problem: Java Browser comes up empty

Description: If you installed the latest release of the Java Extensions on top of the

prerelease, the JavaBrowser may come up empty. This is because some old versions of the loadable bundles used by the browser will still be present.

Workaround: Traverse to the browser resource directory (*NEXT_ROOT*/**NextDemos/JavaBrowser.app/Resources**) and remove the **Decompiler.beandle** and **InputSources.beandle** directories.

Reference: 75691

Problem: Setting JAVA_HOME produces better error messages during compilation

Description: The JDK environment that's set up by the Java Extensions for compilation needs to have JAVA_HOME set so that the compiler can find the properties file that contains the English error messages. Otherwise, **javac** complains and prints an error message that's only mildly descriptive

Workaround: Using the Windows NT System control panel, create a JAVA_HOME environment variable and set it to *NEXT_ROOT*/**NextLibrary/JDK-1_0_2**.

Reference: None

Problem: Control-C doesn't always kill **java.exe**

Description: If you use Control-C to kill a running Java application, the **java.exe** process isn't always killed properly. This will cause you to have multiple Java VMs running at the same time.

Workaround: Use the Windows NT TaskManager to ensure that the extra VM processes have been killed off.

Enterprise Objects Framework

Reference: 75466

Problem: EOF client libraries can't be automatically loaded into Java on Solaris systems

Description: On Solaris, the client libraries and adaptor frameworks cannot be

automatically loaded by EOF; you must explicitly load a library that is linked against the adaptor frameworks and the client libraries in your Java program. See the **main()** method and Makefile.postamble in the DodgeDemoJava example for the recommended way to do this. Note that some demos--such as MovieDemoJava--won't work unless this change is made.

Workaround: None.

Examples

Reference: 75464

Problem: DodgeLiteJava may throw an exception when selecting a car color.

Description: When running the DodgeLiteJava example, if you are selecting a car color and you click outside the active areas of the map (these areas do not perfectly correspond to the color samples, the example will throw an

exception.

Workaround: None.

Foundation Framework

Reference: 75519

Problem: **NSClassFromString()** is very slow

Description: Code that makes heavy use of **NSClassFromString()** will run slower once the Java extensions are installed.

Workaround: None.

Bridget

Reference: 75284

Problem: #import includes files multiple times in Bridget on Windows NT

Description: If you have a header file that imports the same file more than once using different names, such as #import "Foo.h" and #import "d:/Headers/Foo.h", it will be parsed by Bridget multiple times, and methods declared in the imported files will be treated as multiple declarations. This results in duplicate methods being emitted in the generated Java classes.

Workaround: None.

WebObjects Builder

Reference: 75485

Problem: WebObjects Builder generates "protected Object foo[]" for an array

Description: When creating a new array variable in WebObjects Builder, you get an

array of type Object instead of an ImmutabbleVector. You should get an ImmutableVector, because an array is usually created to work with WORepetitions.

Workaround: None.

WebObjects Framework

Reference: 74753

Problem: **appendContentHTMLString:** header file comments are 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: None.

Reference: 74498

Problem: Run-time URL completion doesn't work with some dynamic elements when the attribute namess are lowercase.

Description: At times the WebObjects Framework is unable to complete a path or 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 both 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**, 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"
};
```

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.

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.
