WebObjects Release 4.5 Post-Installation Instructions

This manual describes WebObjects 4.5

Writing: Greg Wilson
Production: Lorraine Findlay and Gerri Gray
With help from: Kory Hansen

About This Document

This short guide describes steps you should follow immediately after you've installed WebObjects. By following the instructions in this guide, you'll be assured that the WebObjects package was installed properly.

Before reading this guide, be sure you have followed all of the instructions in the printed *WebObjects Installation Guide* and that you've checked the *WebObjects Release Notes* and *Enterprise Objects Framework Release Notes*, both of which are available online. Updated documentation is maintained on Apple's web site; go to http://developer.apple.com/techpubs/webobjects/webobjects.html.

Chapter 1

Post-Installation Steps

Immediately after installation, depending on the platform onto which you've installed, and on the web server and database you are using, you may need to perform additional steps to fully enable your WebObjects installation. The sections within this chapter detail those steps, and are organized by platform.

Windows NT Post-Installation Steps

After you've finished installing on Windows NT, you may want to perform one or more of the following steps:

- Making Monitor Fail-safe
- Obtain and Install Database Client Libraries

Making Monitor Fail-safe

Because Monitor is a critical piece of any deployment, measures should be taken to make sure that it does not fail. Monitor is installed as a service (listed as "Apple WebObjects Monitor" in the Services control panel) and you can configure it to start automatically upon boot by changing its Startup mode to Automatic. Note that although this will cause Monitor to be started automatically, you'll have to manually start your web browser and connect to Monitor manually (or by putting a shortcut to your web browser in your Startup program group). The URL for Monitor can be verified by checking the Windows NT Event Viewer (Start > Programs > Administrative Tools > Event Viewer) and is similar to:

```
http://localhost:1027/cgi-bin/WebObjects.exe/Monitor
```

Obtain and Install Database Client Libraries

To use Enterprise Objects Framework on Windows NT, you must have the appropriate database client libraries. The Sybase client libraries are provided on the WebObjects Developer 4.5 CD as an optional package. To install the Sybase client libraries, you must do a custom installation and explicitly specify that you want to install the package. To use Enterprise Objects Framework with Oracle or Informix, you must purchase the appropriate client libraries from your database vendor.

Oracle

Phone: (800) 542-1170 or call your local sales representative

Ask for: Oracle 8 Client

The Oracle adaptor on NT requires the Oracle 8i, 8.0, 7.3, or 7.2 Client Library. It won't work with the 7.1 libraries.

On Windows NT, using the latest release of the Oracle client library requires you to use SQL*Net v2, which requires a `tnsnames.ora` file. `tnsnames.ora` is a file that you put on client machines, generally in the directory `c:\Oracle\Net80\Admin\`. The file contains information needed to connect to a server over the network. Entries in tnsnames.ora are keyed off of a server ID alias, and they include information such as the server ID, the host machine name, and the network protocol used by the client library to resolve the server ID alias. An entry in `tnsnames.ora` might resemble the following:

```
mySeverAlias = (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)
(HOST=myMachine) (PORT=1521))(CONNECT_DATA=(SID=eof)))
```

Oracle provides tools you can use to create `tnsnames.ora` files. Refer to your Oracle documentation for more information on `tnsnames.ora` files and the tools you can use to create them.

If you're using the 7.2 version of the Oracle client libraries on Windows NT, you can use either SQL*Net v1 or SQL*Net v2. To use SQL*Net v1, you should set your adaptor's connectionDictionary serverId entry to

```
"T:<host-machine>:<server-name>".
```

Informix

Phone: (800) 331-1763 or call your local sales representative

Ask for: ESQL/C version 7.23.TC9 for Win32

Informix Notes

If you get the error "INFORMIXSERVER not in sqlhosts file (25596)" but can connect to your database server using the Informix `ilogin` program, you may need to run SetNet32 to update the environment variables used by Informix.

The Informix client libraries appear to have redundant sources of server information. They use the `sqlhosts` file (`$INFORMIXDIR/etc/sqlhosts`) as well as a collection of environment variables managed by the Setnet32 program.

See your Informix documentation for more information on the `sqlhosts` file and the Setnet32 program.

Sybase

Phone: (800) 685-8225 or call your local sales representative

Ask for: OpenClient/C Version 11.1 or later

Mac OS X Server Post-Installation Steps

Using woservice on Mac OS X Server

On Mac OS X Server `wotaskd` is started automatically upon boot time under the control of `woservice`, which will restart `wotaskd` in the event that it is killed or crashes for any reason. This is done in `3100_WebObjects`, which is a script within `/etc/startup`.

You can use `woservice` to control Monitor in a similar fashion, thus ensuring that it is always running. You can enter the `woservice` tool on a shell command line (such as provided by `Terminal.app`), start it from a shell script, or configure it to launch Monitor automatically at boot time.

When invoking `woservice` from the command-line, pass as arguments the path to the application you want to be launched followed by any arguments you want to launch it with. So to start `woservice` for Monitor, you might give the following command:

```
woservice
/System/Library/WebObjects/Applications/Monitor.woa/Monitor
```

To have Monitor launched at system boot time, you must add a startup script to `/etc/startup`. The scripts in `/etc/startup` follow a naming convention whereby the first four characters of the script filename are numbers. These numbers signify the order in which the system runs the scripts in /etc/startup. You should start Monitor near the end of the boot cycle.

You could add the following script, named 3200_Monitor, to /etc/startup to start woservice when the system boots and have it keep Monitor running:

```
#!/bin/sh
#
# Start Monitor using woservice for WebObjects Deployment
#
. /etc/rc.common
# the following is one line:
/System/Library/WebObjects/Applications/Monitor.woa/woservice
/System/Library/WebObjects/Applications/Monitor.woa/Monitor &
```

Update Java CLASSPATH for old Swing/AWT Applications

Between Swing 1.0 and Swing 1.1, Sun changed the Swing package path from com.sun.java to javax. WebObjects 4.5 installs the new versions of of Swing.jar and AWT.jar in /System/Library/Frameworks/JavaVM.framework/Classes (AWT depends on Swing). Note that this is where the old .jar files were located; in order to allow applications that were using the older versions of the Swing and AWT packages to continue to function, the WebObjects 4.5 installer doesn't overwrite the old .jar files but instead renames them to Swing.old.jar and AWT.old.jar. Thus, to use the old .jar files you need only alter your runtime CLASSPATH to point to the new locations of the old versions of AWT and Swing.

This change needs to be made only for non-WebObjects projects that use the old version of Swing. WebObjects projects that use the old version of Swing should be modified to use the new version.

Solaris Post-Installation Steps

After you've finished installing on Solaris, perform the following steps:

- Set NEXT_ROOT

- Set PATH
- (optional) Enable and Start machd and nmserver
- Setting the LD_LIBRARY_PATH
- Set the Time Zone
- Install the Java Native Threads Pack
- Rebuild the Executable WODefaultApp
- Obtain and Install Database Client Libraries
- Build the Examples

Some of these steps are expanded upon in the sections below:

Setting the LD_LIBRARY_PATH

Set the LD_LIBRARY_PATH environment variable to an appropriate value for your installation, with the following prepended to it (enter the following all on one line):

```
$NEXT_ROOT/Local/Developer/Libraries:$NEXT_ROOT/
Developer/Libraries:$NEXT_ROOT/Local/Library/
Executables:$NEXT_ROOT/Library/Executables:$NEXT_ROOT/
Library/JDK/lib
```

Set the Time Zone

As part of the installation process, you're told how to manually set the time zone. If you haven't yet done so, you should do it now. The instructions for setting the time zone are repeated here:

Certain WebObjects APIs depend upon this system's local time zone being set. To set the local time zone:

1. 'su' to root.
2. 'cd' to `/usr/share/lib/zoneinfo`
3. Create the symbolic link `localtime` which points to the appropriate time zone. For example: `ln -s US/Pacific localtime`

Note that creating a symbol link isn't your only option for setting the system time zone. The algorithm that the `systemTimeZone` method (in the NSTimeZone class) uses to determine the time zone is as follows:

1. `systemTimeZone` first checks for an environment variable named "TZFILE". If it exists and its value is a path to a valid time zone file, `systemTimeZone` returns the indicated time zone. If the environment variable doesn't exist, or the path is nonexistent or otherwise invalid, proceed to the next step.

2. `systemTimeZone` next checks the "TZ" environment variable. If it exists and its value, when used as a path relative to `/usr/share/lib/zoneinfo`, indicates a valid time zone file, `systemTimeZone` returns the indicated time zone. If the "TZ" environment variable doesn't exist or doesn't identify a valid time zone file, proceed to the next step.

3. `systemTimeZone` finally checks for the existence of a "localtime" symbolic link within `/usr/share/lib/zoneinfo`. If the link doesn't exist or doesn't point to a valid time zone file, NSTimeZone throws an exception and sets the time zone to GMT. Otherwise, the indicated time zone is returned from `systemTimeZone`.

Install the Java Native Threads Pack

If you are running Solaris 2.5.1 (with patches 103566, 103600, and 103640), you must install the Java Native Threads Pack. You can download the Native Threads Pack from JavaSoft's Java Development Kit website.

Be aware that the Native Threads patch for JDK 1.1.8 uses the green threads implementation by default, even if the native threads patch is in place. To get native threads you must do at least one of the following:

- Have an environment variable named THREADS_FLAG that is set to "native".
- Pass "-native" to each executable in the JDK whenever you call it (for instance, "javac -native myProg.java"). This is for *all executables* in the JDK, including the compiler, the JVM and AppletViewer.
- Modify java_wrapper, changing the script so that DEFAULT_THREADS_FGLAG is set to "native".

Rebuild the Executable WODefaultApp

On Solaris, the Enterprise Objects Framework cannot automatically load your database's client library and its adaptor, as it can on other platforms. Because of this, you must rebuild the `WODefaultApp` executable, which is installed with WebObjects and is used to run purely scripted applications. If you don't rebuild this executable, any purely scripted applications you run with `WODefaultApp` won't be able to access a database.

If you answered "y" to all questions you were asked during installation, the `WODefaultApp` executable has already been rebuilt by the installation process. If you answered "n" to the question about building `WODefaultApp` or you have installed new client libraries afterwards, you should rebuild `WODefaultApp` before testing your installation.

To rebuild `WODefaultApp`, run the `RebuildWODefaultApp` script located in `$NEXT_ROOT/Developer/Examples/WebObjects/Source/WODefaultApp`.

Note: Each time you create a new project, you'll need to set it up so that it statically links the database's client library and adaptor. To do so, add the appropriate adaptor framework to the FRAMEWORKS makefile variable definition, and uncomment this line in the `Makefile.preamble`:

```
include $(MAKEFILEDIR)/pdo-eoadaptor-linking.make
```

Obtain and Install Database Client Libraries

To use Enterprise Objects Framework on Solaris, you must have the appropriate database client libraries.

Here's what you need:

Oracle

Phone: (800) 542-1170 or call your local sales representative

Ask for: 8.0 SQLNet V2 TCP/IP Client libraries

The Oracle adaptor on Solaris requires the Oracle 8.0 or 7.3 Client Library. The makefiles are configured for 7.3 or 8.0 (they default to 7.3); with some modification of the makefiles, you can also link with 7.2.

Informix

Phone: (800) 331-1763 or call your local sales representative

Ask for: ESQL/C Version 7.23.UC9

If you get the error "INFORMIXSERVER not in sqlhosts file (25596)" but can connect to your database server using the Informix ilogin program, you may need to run SetNet32 to update the environment variables used by Informix.

The Informix client libraries appear to have redundant sources of server information. They use the sqlhosts file ($INFORMIXDIR/etc/sqlhosts) as well as a collection of environment variables managed by the Setnet32 program.

See your Informix documentation for more information on the sqlhosts file and the Setnet32 program.

Sybase

Phone: (800) 685-8225 or call your local sales representative

Ask for: OpenClient/C Version 11.1

Linking Against the Adaptor

On Solaris applications must explicitly link against the adaptor framework and the client libraries. New makefiles look for adaptor frameworks and automatically add in the right linker arguments. Simply add the adaptor framework to your project, make sure that LD_LIBRARY_PATH contains the client library directory, and set the requisite environment variable specifying where the client libraries are installed. For Oracle set ORACLE_HOME and, optionally, ORACLE_REL in the makefile preamble. (The ORACLE_REL flag controls which set of libraries are used. It uses the Oracle 7.3 static link libraries by default, but you can also specify “8.0-static” or “7.3-dynamic.”) For Sybase set SYBASE_HOME. For Informix set INFORMIX_HOME, INFORMIXDIR, and INFORMIXSERVER.

Build the Examples

The section “Verifying the Installation” (page 31) describes how to verify that your installation is working properly by running the examples, some of which are compiled examples. On Solaris, you must build the examples before you can test your system. If you want to use the WebObjects examples to test your system, build the examples found in `$NEXT_ROOT/Developer/Examples/WebObjects`.

HP-UX Post-Installation Steps

After you've finished installing on HP-UX, perform the following steps:

- Set NEXT_ROOT
- Set PATH
- (optional) Enable and Start machd and nmserver
- Setting the LPATH and SHLIB_PATH Environment Variables
- Set the Time Zone
- Change the UID for User nobody
- Rebuild the Executable WODefaultApp
- Obtain and Install Database Client Libraries
- Build the Examples

Some of these steps are expanded upon in the sections below:

Setting the LPATH and SHLIB_PATH Environment Variables

Set the LPATH environment variable to an appropriate value for your installation, with the following prepended to it (enter the following all on one line):

```
$NEXT_ROOT/Local/Developer/Libraries:$NEXT_ROOT/
Developer/Libraries:$NEXT_ROOT/Local/Library/
Executables:$NEXT_ROOT/Library/Executables:$NEXT_ROOT/
Library/JDK/lib
```

Also set the SHLIB_PATH to the value of LPATH with:

```
setenv SHLIB_PATH "$LPATH"
```

Set the Time Zone

As part of the installation process, you're told how to manually set the time zone. If you haven't yet done so, you should do it now. The instructions for setting the time zone are repeated here:

Certain WebObjects APIs depend upon this system's local time zone being set. To set the local time zone:

1. 'su' to root.
2. 'cd' to `/etc/zoneinfo`
3. Create the symbolic link `localtime` which points to the appropriate time zone. For example: `ln -s US/Pacific localtime`

Note that creating a symbol link isn't your only option for setting the system time zone. The algorithm that the `systemTimeZone` method (in the NSTimeZone class) uses to determine the time zone is as follows:

1. `systemTimeZone` first checks for an environment variable named "TZFILE". If it exists and its value is a path to a valid time zone file, `systemTimeZone` returns the indicated time zone. If the environment variable doesn't exist, or the path is nonexistent or otherwise invalid, proceed to the next step.
2. `systemTimeZone` next checks the "TZ" environment variable. If it exists and its value, when used as a path relative to `/etc/zoneinfo`, indicates a valid time zone file, `systemTimeZone` returns the indicated time zone. If the "TZ" environment variable doesn't exist or doesn't identify a valid time zone file, proceed to the next step.
3. `systemTimeZone` finally checks for the existence of a "localtime" symbolic link within `/etc/zoneinfo`. If the link doesn't exist or

doesn't point to a valid time zone file, NSTimeZone throws an exception and sets the time zone to GMT. Otherwise, the indicated time zone is returned from `systemTimeZone`.

Change the UID for User nobody

If you're using the Apache web server you'll need to change the UID of the user nobody, which is used to launch CGI processes. By default, the UID is -2, which causes setuid to complain about an invalid argument.

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

Rebuild the Executable WODefaultApp

On HP-UX, the Enterprise Objects Framework cannot automatically load your database's client library and its adaptor, as it can on other platforms. Because of this, you must rebuild the `WODefaultApp` executable, which is installed with WebObjects and is used to run purely scripted applications. If you don't rebuild this executable, any purely scripted applications you run with `WODefaultApp` won't be able to access the database.

If you answered "y" to all questions you were asked during installation, the `WODefaultApp` executable has already been rebuilt by the installation process. If you answered "n" to the question about building `WODefaultApp` or you have installed new client libraries afterwards, you should rebuild `WODefaultApp` before testing your installation.

To rebuild `WODefaultApp`, run the `RebuildWODefaultApp` script located in `$NEXT_ROOT/Developer/Examples/WebObjects/Source/WODefaultApp`.

Note: Each time you create a new project, you'll need to set it up so that it statically links the database's client library and adaptor. To do so, add the appropriate adaptor framework to the FRAMEWORKS makefile variable definition, and uncomment this line in the `Makefile.preamble`.

```
include $(MAKEFILEDIR)/pdo-eoadaptor-linking.make
```

Obtain and Install Database Client Libraries

To use Enterprise Objects Framework on HP-UX, you must have the appropriate database client libraries.

Oracle

Phone: (800) 542-1170 or call your local sales representative

Ask for: 8.0 SQLNet V2 TCP/IP Client libraries

The Oracle adaptor on HP-UX requires the Oracle 8.0 or 7.3 Client Library. The makefiles are configured for 7.3 or 8.0 (they default to 7.3); with some modification of the makefiles, you could also link with 7.2.

Informix

Phone: (800) 331-1763 or call your local sales representative

Ask for: ESQL/C Version 7.23.UC6

If you get the error "INFORMIXSERVER not in sqlhosts file (25596)" but can connect to your database server using the Informix ilogin program, you may need to run SetNet32 to update the environment variables used by Informix.

The Informix client libraries appear to have redundant sources of server information. They use the sqlhosts file ($INFORMIXDIR/etc/sqlhosts) as well as a collection of environment variables managed by the Setnet32 program.

See your Informix documentation for more information on the sqlhosts file and the Setnet32 program.

Sybase

Phone: (800) 685-8225 or call your local sales representative

Ask for: OpenClient/C Version 11.1

Linking Against the Adaptor

On HP-UX applications must explicitly link against the adaptor framework and the client libraries. New makefiles look for adaptor frameworks and automatically add in the right linker arguments. Simply add the adaptor framework to your project, make sure that LD_LIBRARY_PATH contains the client library directory, and set the requisite environment variable specifying where the client libraries are installed. For Oracle set ORACLE_HOME and, optionally, ORACLE_REL in the makefile preamble. (The ORACLE_REL flag controls which set of libraries are used. It uses the Oracle 7.3 static link libraries by default, but you can also specify "8.0-static" or "7.3-dynamic".) For Sybase set SYBASE_HOME. For Informix set INFORMIX_HOME, INFORMIXDIR, and INFORMIXSERVER.

Build the Examples

The section "Verifying the Installation" (page 31) describes how to verify that your installation is working properly by running the examples, some of which are compiled examples. On HP-UX, you must build the examples before you can test your system. If you want to use the WebObjects examples to test your system, build the examples found in `$NEXT_ROOT/Developer/Examples/WebObjects`.

Using Microsoft Access

Read this section if you want to use Enterprise Objects Framework to connect to a Microsoft Access database. The information in this section describes what you need to do before you can install the sample databases or create your own custom databases.

What You Need

Enterprise Objects Framework uses ODBC (a standard API developed by Microsoft for accessing database management systems) to interact with Access databases, so you'll need an Access-specific ODBC driver if you don't have one already.

To determine if you've already got an ODBC driver installed, open the Control panel. If there isn't an ODBC option listed, then you don't have any ODBC drivers installed. The latest ODBC manager and drivers for Windows NT are available from Microsoft. See `http://www.microsoft.com/data/`.

Some bugs in old versions of the ODBCJT32.DLL (used by the Enterprise Objects Framework to communicate with Microsoft Access) can cause problems with primary key generation in the ODBC Adaptor (the ODBC manager reports an invalid SQL statement when updating the EO_PK_TABLE). ODBCJT32.DLL version 3.40.2728 causes this problem, but the more recent ODBCJT32.DLL version 3.51.1029 works correctly. To verify that you don't have the older version of the ODBC drivers, open the ODBC Control Panel and click on the tab labeled "ODBC Drivers".

Creating Data Sources and Databases

After you've installed an ODBC driver for Access, you'll need to create databases and corresponding data sources for them. A data source simply stores the information needed to connect to your database. The easiest way to create an Access database and a corresponding data source from scratch is to create them both at the same time with the ODBC Data Source Administrator:

1. Choose the ODBC option in the Control Panel.

   An ODBC Data Source Administrator panel opens.

2. Click the System DSN tab.

   System data sources are accessible to all users, including NT services. It's important to create system data sources instead of user data sources, especially for use with WebObjects, because autolaunched applications aren't able to access user data sources.

3. Click Add.

   The Create New Data Source panel opens.

4. Select the Microsoft Access Driver.

5. Click Finish.

   The ODBC Microsoft Access Setup panel opens.

6. Type a name for your data source in the Data Source Name field (Movies, for example).

Remember the name you provide because later you'll it to login to your database.

7. In the Database section of the panel, click Create.

   A New Database panel opens. You'll use this panel to create a new, empty database.

8. Type a name for your database in the Database Name field (Movies.mdb, for example).

9. Choose a directory where you want the database located.

10. Click OK.

    A panel opens telling you that the database was successfully created.

11. In the ODBC Microsoft Access Setup panel, click OK.

Your database and data source are now ready to use. If you want to populate the new database with one of the sample databases that come with WebObjects, you've completed the first step: setting up the database accounts. To find out what to do next, see "Setting Up the Sample Databases."

Setting Up the Sample Databases

WebObjects includes Enterprise Objects Framework database integration technology. If you've installed WebObjects Developer, several examples demonstrate Enterprise Objects Framework programming techniques. In particular, if you work through the tutorials in the book *Getting Started With WebObjects*, you'll need two sample databases: Movies, which contains background information on all movies available form a store's distributor, and Rentals, which contains the inventory, customer list, and rental transaction records for the store. In addition, WebObjects ships with an example application named Movies, which uses a database of Movies, their ratings, and principal actors. The section "Verifying the Installation" (page 31) uses Movies to verify that the installation is working properly.

Warning: None of the data in the sample databases is intended to be accurate.

All of the examples supplied with this release come pre-built, and are runnable directly from their distribution directories. Those that use the Enterprise Objects Framework run "out of the box" against the supplied OpenBaseLite databases. On Mac OS X Server and Windows NT systems, you can invoke the Setup Wizard to reconfigure the example application models to use another supported database, such as Oracle or Sybase, and to optionally load the database for you. The Setup Wizard copies the examples into a directory of your choosing before modifying them.

For WebObjects 4.5, the Setup Wizard doesn't completely perform all operations necessary to get the examples running. In particular, it doesn't convert the models in the BusinessLogicInheritance framework, and it doesn't load the inheritance model data into a database. However, it does work correctly on the BusinessLogic framework; thus, the simpler examples that don't use inheritance will work correctly with your database after you run the SetupWizard and perform a few additional manual steps.

Running the Setup Wizard

SetupWizard.app is located in `/System/Developer/Examples/EnterpriseObjects` (on Windows NT systems, `*NEXT_ROOT*\Developer\Examples\EnterpriseObjects`). After you invoke the wizard, you'll be prompted for a directory into which the examples should be installed. *This directory must already exist in order for the wizard to work properly.*

Warning: The wizard will delete any existing contents of the directory you specify.

After advancing to the next screen, follow the prompts through the rest of the wizard. Note that you'll need to select a database adaptor and login to the database twice: once for Movies, and once for Rentals. Finally, the wizard will ask you whether or not you want it to populate the selected databases for you.

The Setup Wizard doesn't set up the inheritance databases; for this, you'll need to perform the steps listed in `ReadMe-Inheritance.html`, which is located in the same directory in which the Setup Wizard is located (`.../Examples/EnterpriseObjects`).

Chapter 2

Troubleshooting

Verifying the Installation

At this point, you should have installed WebObjects as described in the *WebObjects Installation Guide* (included with the WebObjects 4.5 CD), perform the post-installation steps for your operating system as described in the first part of this document, and rebooted your computer.

After you have completed the installation and the post-installation, verify the installation by performing the following steps:

Note: These steps assume you've installed WebObjects Developer. The examples mentioned below are not installed with WebObjects Deployment. To verify a Deployment installation, you might complete these same steps running other applications.

1. Verify that the WebObjects processes are running.

   On Windows NT, open the Services control panel and verify that "Apple Mach Daemon", "Apple NetName Server", and "Apple WebObjects Task Daemon" are all running (their startup mode should be set to "Automatic").

   On Mac OS X Server, click the icon in the upper left-hand corner of the screen and choose "Show All Processes". Verify that `nmserver`, `woservice`, and `wotaskd` are all running.

   On Solaris and HP-UX, type "`ps -eaf`" in a shell window and verify that `machd`, `nmserver`, and `wotaskd` are all running.

2. Try to run a simple scripted application.

   Open a command-shell window (on Windows NT, open a Bourne shell window) and enter the following commands:

   ```
   > cd
   NEXT_ROOT/Developer/Examples/WebObjects/WebScript/
   HelloWorld
   ```

   ```
   > NEXT_ROOT/Library/WebObjects/Executables/
   WODefaultApp[.exe]
   ```

   where:

NEXT_ROOT is the directory in which you installed WebObjects software (*NEXT_ROOT* isn't applicable on Mac OS X systems).

These commands should run the HelloWorld example application, launch a web browser, and enter HelloWorld's URL in the browser. If this doesn't work, go to the topic *Troubleshooting*.

After you have verified that HelloWorld runs, type Control-C to shut it down.

3. If scripted applications work, try running compiled applications. Enter the following commands:

```
> cd ../../Java/Movies/Movies.woa
> ./Movies[.exe]
```

If you have trouble running compiled applications, go to the topic *Troubleshooting*.

After you have verified that Movies or HelloWorldCompiled runs, type Control-C to shut it down.

Troubleshooting

WebObjects operates on a number of hardware platforms, running various operating systems and supporting many different types of HTTP servers. Taking these variables together means that WebObjects finds itself in a large number of distinctly different environments, a situation that can lead to problems affecting the installation of WebObjects or the running of WebObjects applications.

This short guide will help you find solutions to the problems most commonly encountered. It's divided into these major sections:

- Checking the Installation
  - Windows NT
  - Solaris or HP-UX
  - Mac OS X Server

- Problems With Scripted Applications
- Problems With Compiled Applications

Start by reading the section for your operating system in “Checking the Installation.” If you don’t see the solution there, continue with “Problems With Scripted Applications” and then “Problems With Compiled Applications.”

Remember that the WebObjects release notes, available online (on your disk and on Apple’s web site), have the latest information about bugs and workarounds.

Checking the Installation

Windows NT

On Windows NT, these locations should contain the following files or directories:

- In your HTTP server’s *cgi-bin* directory:
  - `WebObjects.exe`: The WebObjects adaptor
  - `WebObjects`: This file is simply a copy of the WebObjects.exe file above minus the .exe extension. Some HTTP servers disallow the extension.
- Your HTTP server’s *document root* directory
  - This directory should contain the following (in addition to any WebObjects applications you may have installed):

```
HTTP_Server_Doc_Root\
    WebObjects\
        Frameworks\
            DirectToJavaClient.framework\
                WebServerResources\
            DirectToWeb.framework\
                WebServerResources\
            EOJavaClient.framework\
                WebServerResources\
            WOExtensions.framework\
                WebServerResources\
        Java\
            directtoweb.jar
            eojavaclient.jar
            woextensions.jar
            com\
                apple\
                    client\
                        directtoweb\
                        eoapplication\
                        eocontrol\
                        eodistribution\
                        eogeneration\
                        eointerface\
                        foundation\
                        playback\
                        webobjects\
```

- *NEXT_ROOT* directory

  This is the location where you installed the WebObjects software. Check for these files and directories:

  - `Library\WebObjects\Executables`: Contains `WODefaultApp.exe`, the default application executable for scripted WebObjects applications

  - `Library\Frameworks\WebObjects.framework`: WebObjects library of classes, plus header files (Developer installations only)

  - `Library\Frameworks\WOExtensions.framework`: WebObjects Extensions framework, which contains extra dynamic elements and shared components

- `Library\Java`: The Java interfaces to WebObjects classes.
- `Library\WebObjects\Adaptors`: Contains WebObjects configuration files and adaptors.

Corrective actions:

If you are missing any of the files from your server's *cgi-bin* directory, you can copy them from `Library\WebObjects\Adaptors\CGI` to your HTTP server's *cgi-bin* directory.

If you are missing any of the contents of your server's *document root* directory, copy the missing files from the following locations, or simply reinstall WebObjects.

If you are missing...	Copy it from...
DirectToJavaClient.framework/ WebServerResources	`$NEXT_ROOT`/Library/Frameworks/DirectToJavaClient.framework/
DirectToWeb.framework/ WebServerResources	`$NEXT_ROOT`/Library/Frameworks/DirectToWeb.framework/
EOJavaClient.framework/ WebServerResources	`$NEXT_ROOT`/Library/Frameworks/EOJavaClient.framework/
WOExtensions.framework/ WebServerResources	`$NEXT_ROOT`/Library/Frameworks/WOExtensions.framework/
Java/directtoweb.jar	`$NEXT_ROOT`/Library/Frameworks/DirectToWeb.framework/ WebServerResources/Java/
Java/eojavaclient.jar	`$NEXT_ROOT`/Library/Frameworks/EOJavaClient.framework/ WebServerResources/Java/
Java/woextensions.jar	`$NEXT_ROOT`/Library/Frameworks/WOExtensions.framework/ WebServerResources/Java/
Java/com/apple/client/*	Obtain the contents of the `client` directory by combining the `WebServerResources/Java/com/apple/client/` directories of the DirectToWeb, EOJavaClient, DirectToJavaClient, and WOExtensions frameworks (all in `$NEXT_ROOT/Library/Frameworks`)

If you are missing any of the contents of the `$NEXT_ROOT` directory, reinstall WebObjects.

Solaris or HP-UX

On Solaris and HP-UX, these locations should contain the following files or directories:

- Your server's *cgi-bin* directory.
  - `WebObjects`: The WebObjects adaptor
- Your server's *document root* directory
  - This directory should contain the following (in addition to any WebObjects applications you may have installed):

```
HTTP_Server_Doc_Root/
    WebObjects/
        Frameworks/
            DirectToJavaClient.framework/
                WebServerResources/
            DirectToWeb.framework/
                WebServerResources/
            EOJavaClient.framework/
                WebServerResources/
            WOExtensions.framework/
                WebServerResources/
        Java/
            directtoweb.jar
            eojavaclient.jar
            woextensions.jar
            com/
                apple/
                    client/
                        directtoweb/
                        eoapplication/
                        eocontrol/
                        eodistribution/
                        eogeneration/
                        eointerface/
                        foundation/
                        playback/
                        webobjects/
```

- *NEXT_ROOT* directory

  This is the location where you installed the WebObjects software. Check for these files and directories:

- `Library/WebObjects/Executables`: Contains `WODefaultApp`, the default application executable for scripted WebObjects applications
- `Library/Frameworks/WebObjects.framework`: WebObjects library of classes, plus header files (Developer installations only)
- `Library/Frameworks/WOExtensions.framework`: WebObjects Extensions framework, which contains extra dynamic elements and shared components
- `Library/Java`: The Java interfaces to WebObjects classes.
- `Library/WebObjects/Adaptors`: Contains WebObjects configuration files and adaptors.

Corrective action:

If you are missing any of the files from your server's *cgi-bin* directory, you can copy them from `$NEXT_ROOT/Library/WebObjects/Adaptors/CGI` to your HTTP server's *cgi-bin* directory.

If you are missing any of the contents of your server's *document root* directory, copy the missing files from the following locations, or simply reinstall WebObjects.

If you are missing...	Copy it from...
DirectToJavaClient.framework/ WebServerResources	`$NEXT_ROOT`/Library/Frameworks/DirectToJavaClient.framework/
DirectToWeb.framework/ WebServerResources	`$NEXT_ROOT`/Library/Frameworks/DirectToWeb.framework/
EOJavaClient.framework/ WebServerResources	`$NEXT_ROOT`/Library/Frameworks/EOJavaClient.framework/
WOExtensions.framework/ WebServerResources	`$NEXT_ROOT`/Library/Frameworks/WOExtensions.framework/
Java/directtoweb.jar	`$NEXT_ROOT`/Library/Frameworks/DirectToWeb.framework/ WebServerResources/Java/
Java/eojavaclient.jar	`$NEXT_ROOT`/Library/Frameworks/EOJavaClient.framework/ WebServerResources/Java/

If you are missing...	**Copy it from...**
Java/woextensions.jar	`$NEXT_ROOT`/Library/Frameworks/WOExtensions.framework/ WebServerResources/Java/
Java/com/apple/client/*	Obtain the contents of the `client` directory by combining the `WebServerResources/Java/com/apple/client/` directories of the DirectToWeb, EOJavaClient, DirectToJavaClient, and WOExtensions frameworks (all in `$NEXT_ROOT/Library/Frameworks`)

If you are missing any of the contents of the `$NEXT_ROOT` directory, reinstall WebObjects.

Mac OS X Server

On Mac OS X Server, these locations should contain the following files or directories:

- Your Apache server's *cgi-bin* directory (`/Local/Library/WebServer/CGI-Executables`).
  - `WebObjects`: The WebObjects adaptor
- Your server's *document root* directory (`/Local/Library/WebServer/Documents`)
  - This directory should contain the following (in addition to any WebObjects applications you may have installed):

```
HTTP_Server_Doc_Root/
    WebObjects/
        Frameworks/
            DirectToJavaClient.framework/
                WebServerResources/
            DirectToWeb.framework/
                WebServerResources/
            EOJavaClient.framework/
                WebServerResources/
            WOExtensions.framework/
                WebServerResources/
        Java/
            directtoweb.jar
            eojavaclient.jar
            woextensions.jar
            com/
                apple/
                    client/
                        directtoweb/
                        eoapplication/
                        eocontrol/
                        eodistribution/
                        eogeneration/
                        eointerface/
                        foundation/
                        playback/
                        webobjects/
```

- `/System/Library` directory

  Check for these files and directories:

  - `Frameworks/WebObjects.framework`: WebObjects library of classes, plus header files (Developer installations only)
  - `Frameworks/WOExtensions.framework`: WebObjects Extensions framework, which contains extra dynamic elements and shared components
  - `Java`: The Java interfaces to WebObjects classes.
  - `WebObjects/Adaptors`: Contains WebObjects configuration files and adaptors.

- `WebObjects/Executables`: Contains `WODefaultApp`, the default application executable for scripted WebObjects applications

Corrective action:

If the WebObjects adaptor is missing from your Apache web server's *cgi-bin* directory, reinstall it using the instructions found in `/System/Library/WebObjects/Adaptors/Apache/Installation.html`.

If you are missing any of the contents of your server's *document root* directory, copy the missing files from the following locations, or simply reinstall WebObjects.

If you are missing...	Copy it from...
DirectToJavaClient.framework/ WebServerResources	/System/Library/Frameworks/DirectToJavaClient.framework/
DirectToWeb.framework/ WebServerResources	/System/Library/Frameworks/DirectToWeb.framework/
EOJavaClient.framework/ WebServerResources	/System/Library/Frameworks/EOJavaClient.framework/
WOExtensions.framework/ WebServerResources	/System/Library/Frameworks/WOExtensions.framework/
Java/directtoweb.jar	/System/Library/Frameworks/DirectToWeb.framework/ WebServerResources/Java/
Java/eojavaclient.jar	/System/Library/Frameworks/EOJavaClient.framework/ WebServerResources/Java/
Java/woextensions.jar	/System/Library/Frameworks/WOExtensions.framework/ WebServerResources/Java/
Java/com/apple/client/*	Obtain the contents of the `client` directory by combining the `WebServerResources/Java/com/apple/client/` directories of the DirectToWeb, EOJavaClient, DirectToJavaClient, and WOExtensions frameworks (all in `/System/Library/Frameworks`)

If you are missing any of the contents of your `/System/Library` directory, reinstall WebObjects.

Problems With Scripted Applications

Scripted example applications (HelloWorld, TimeOff, and so on) are the simplest ones and don't contain compiled code.

Problem

The web browser does not launch or launches the incorrect URL

Checklist

1. Check the debugging statements printed in the command-shell window.

   When you launch a WebObjects application from the command line, the application computes its own URL, launches the web browser, and enters the URL in the browser. It prints messages about the values it computes to standard output.

   Check the standard output (the command-shell window) for these messages (among others):

   ```
   The applicationPath:
   /System/Developer/Examples/WebObjects/WebScript/HelloWor
   ld
   The applicationBaseURL: /WebObjects/HelloWorld
   Opening application’s URL in Browser: url
   ```

   Corrective action:

   If you see these messages but your web browser doesn’t launch, you might not have a browser installed on your system, or WebObjects cannot find the browser. This is always true on Solaris and HP-UX. If the URL looks correct (as described below), open your browser and type that URL into it.

   If you see a message that says “No Adaptor URL in WebServerConfig.plist,” either the `WebServerConfig.plist` file is missing, or the `WOAdaptorURL` key is missing from it. The file should look something like this:

   ```
   {
      DocumentRoot = "/Apple/Library/WebServer/Documents";
      WOAdaptorURL = "http://localhost/cgi-bin/WebObjects";
   }
   ```

If `WOAdaptorURL` is missing, the web browser does not launch when you launch a WebObjects application. You can enter `WOAdaptorURL` or you can type the URL in the browser and connect to the running application that way.

This base URL value of `WOAdaptorURL` is of the form:

```
http://localhost/cgi-bin/WebObjects
```

cgi-bin is the name of your HTTP server's cgi-bin directory. You specify this name when you configure your HTTP server. The *cgi-bin* directory name is often `cgi-bin`, but it may have a different name. For example, the Microsoft Internet Information Server uses the name `Scripts`.

WebObjects is the name of the WebObjects CGI adaptor as you see it in your HTTP server's cgi-bin directory. Usually, the name is WebObjects. If you're using Windows NT, the adaptor name might be `WebObjects.exe` (however, some older Netscape servers don't use the `.exe` extension.)

If the base URL's cgi-bin and WebObjects adaptor names look correct, consider the `localhost` value. On most sites, `localhost` accesses the server on the local host. However, some sites require a domain name as well (`http://localhost.apple.com`). If your HTTP server isn't running on your local machine, use the host name of the machine running the server in place of "localhost" in the URL above, and make sure a WebObjects adaptor is installed on that machine.

Problem

A simple scripted application won't run properly.

Checklist

1. Try using direct-connect to access your WebObjects application.
2. Check that you can load a static page.

Corrective action:

If your browser displays a message saying that it was unable to connect or that the connection was refused, your HTTP server is probably not running. Check that your server is running. Otherwise, see “Checking the Installation” (page 33) for information on how to fix your installation of WebObjects.

3. Check that the WebObjects adaptor is functioning.

   Check that the WebObjects adaptor is installed correctly and can run. Use your browser to open this URL (which specifies the WebObjects adaptor, but fails to specify an application name):

   ```
   http://localhost/cgi-bin/WebObjects
   ```

   (You may need to replace “localhost” with the name of the host running your HTTP server. You may also need to replace “cgi-bin” with the actual name of the directory that contains scripts and CGI programs on your server.) If the WebObjects adaptor is installed correctly, it displays a list of WebObjects applications running on the local machine.

   If the adaptor is installed incorrectly or can’t run, the browser will instead display a message indicating that the requested object cannot be located. The message may look like this:

   ```
   404 Not Found
   The requested URL /cgi-bin/WebObjects was not found on
   this server.
   ```

   Corrective action:

   Make sure you've supplied the right names in the URL for the host (“localhost” in the example above) and for the cgi-bin directory (sometimes named “Scripts” or “cgiPrograms” rather than “cgi-bin”). Otherwise, see “Checking the Installation” (page 33) for information on how to fix your installation of WebObjects.

Problems With Compiled Applications

Running compiled applications exercises more features of the WebObjects framework and development environment. Before attempting to troubleshoot a problem with running a compiled application, make sure you can run a simple scripted application.

Problem

A simple compiled application won't run properly.

Checklist

1. Make sure the executable has been built.

   On the HP-UX and Solaris platforms, examples are not automatically built as part of installation. You must build them yourself before you can run them. To compile an example, `cd` to the example's project directory and type `make`. For example:

   ```
   > cd /Apple/Developer.Examples/WebObjects/ObjectiveC/
   HelloWorldCompiled
   > make
   ```

Problem

The Movies application won't run.

Checklist

1. Make sure the application was correctly installed and compiled.

   The Movies application must be compiled before you can run it. In addition, you must create the Movies database (scripts are provided) and install the database model file that is compatible with your database server as described in "Setting Up the Sample Databases" (page 25).

   Check the Movies directory for an directory named `Movies.woa`. This is the WebObjects application wrapper. Check the wrapper for an executable file. If the wrapper or the executable doesn't exist, build the Movies application.

   On Solaris and HP-UX, you need to build Movies with the correct client libraries and adaptor. Before you build, add the appropriate adaptor framework to the `FRAMEWORKS` makefile variable. Then uncomment the following line in the `Makefile.preamble` to link the appropriate client libraries:

```
include $(MAKEFILEDIR)/pdo-eoadaptor-linking.make
```

If Movies compiles and runs but can't access data about the various movies, it's probably because the application can't communicate with the database server.

Problem

A WebObjects application won't connect to the database server.

Checklist

1. Check that your database server itself is operating correctly.

   Check that the client libraries for your database server are correctly installed on your machine. If so, you can, for example, use the tools supplied with the database server (`isql` for Sybase, `sqlplus` for Oracle, and `dbaccess` for Informix) to test that you can connect to the server and execute simple SQL commands.

2. Make sure that the database model file is accessible to your application

   The model file should be in the `Resources` directory under the application's `.woa` directory. Taking the Movies application for example, the directory structure would look like this:

```
Movies.woa/
    Movies (the executable file)
    Resources/Movies.eomodeld   (the model file)
```

Chapter 3

Converting Projects From Earlier Releases

WebObjects applications from a prior release cannot simply be recompiled using WebObjects 4.5 Developer; you must first make a number of changes to your project's source code. To aid in this process, WebObjects 4.5 Developer includes a set of "tops" scripts that automatically make many of the changes for you.

If you are converting an application created with WebObjects 4.0 (or 4.01), you need only rebuild your app using WebObjects 4.5—unless your app uses Direct To Web, in which case you'll need to read "Converting Projects From WebObjects 4," below. If your WebObjects applications were created with WebObjects 3.5, you'll first need to perform those steps outlined in the sections beginning with "Converting Projects from WebObjects 3.5 to 4.0" (page 50). Then, review the instructions in "Converting Projects From WebObjects 4," below, to make your applications build under WebObjects 4.5.

Converting Projects From WebObjects 4

Except for Direct To Web applications, no conversion is necessary for WebObjects 4 applications. Simply make any fixes as necessary and recompile your application. Consult the *What's New in WebObjects* document for a detailed list of differences between WebObjects 4.0 and WebObjects 4.5, and note that deprecated API for each framework is listed along with that framework's reference documentation in a separate "Deprecated API" document.

Converting Direct To Web Projects

Since the D2W components are now public, the Java source files, the `Main.wod` file, and the rule files have changed in this release. Two tops scripts are provided to ease the conversion process.

The first script, located in `$(NEXT_ROOT)/Developer/Java/Conversion/WebObjects/D2W4_5codechanges.tops`, modifies the code to conform to Direct to

Web's API changes (see the *What's New in WebObjects* document for more information). You need to execute it on

- all code generated by releases of Direct to Web earlier than 4.5
- the project's `Main.wod` file

A ReadMe file in the script's directory explains how to execute the script.

The second script, located in `$(NEXT_ROOT)/Developer/Java/Conversion/WebObjects/D2W4_5modelchanges.tops`, modifies the rule files to use renamed property-level components. You need to execute it on all files in your project ending in `.d2wmodel`. See the ReadMe file in the script's directory to see how to execute the script.

Converting Projects from WebObjects 3.5 to 4.0

Prior to conversion, make sure you have the latest patches for WebObjects 4.0. Check the Tech Info Library at http://til.info.apple.com/techinfo.nsf/artnum/n70037 for the most up-to-date list. In particular, on Windows NT systems make sure you've obtained and installed the NSTimeZone bug workaround.

To convert a project to WebObjects 4 from an earlier version of WebObjects, do the following:

1. Make a backup copy of your application, and save that backup copy somewhere other than in the base directory.

2. Convert your project's makefiles as described in the document `*$NEXT_ROOT*/Developer/Makefiles/Conversion/DirectoryLayout/ConvertMakefilesReadMe.rtf.`

   (*`$NEXT_ROOT`* is the root installation directory specified when WebObjects was installed.) The project files must change to point to new locations for the build tools. `ConvertMakefilesReadMe.rtf` tells you how to make those changes.

3. If your project contains Java code files, convert them as described in "Converting Java Code From WebObjects 3.5" (page 52).

Note: The Java APIs changed considerably between WebObjects 3.5 and 4. If you've written Java code, you must convert it before you can compile.

4. Open your project in Project Builder, and click "Upgrade Now" when prompted.

   Among other things, upgrading your project in this fashion will:

   - Upgrade your `PB.project` file to the latest version.
   - Upgrade your makefiles to the latest versions.
   - Convert your WebObject project suitcases to the new format.
   - Convert your project so that it conforms to the WebObjects 4 localization scheme. As a result of this, script (`.wos`) and `.api` files are moved outside of their component (`.wo`) directories. Script files now appear in the Classes suitcase.

5. Build. If errors occur during the build, fix them and re-build the project.

   The WebObjects Framework contains many optimizations that should greatly improve your application's performance. However, you may find your code relied on some part of the request-response loop or template parsing code that is no longer always performed.

6. Run the project. If your application doesn't run as expected, see the following sections:

   - "Troubleshooting WebObjects 4 Template Parsing" (page 54)
   - "Troubleshooting WebObjects 4 Request Handling" (page 55)
   - "WebScript Changes" (page 59)

   Also check the *What's New in WebObjects* document for a list of differences between the current release and the previous one.

7. At this stage, if you want, you can remove all usage of deprecated API. WebObjects has been rewritten to be thread-safe, which required deprecating some of the existing APIs. You can still use deprecated API as long as you do not enable multithreading in your application. You'll receive warnings about deprecated API at run-time. For a complete list of what's deprecated, see the "Deprecated API" file that's included as part of each framework's reference documentation.

8. Make the following changes as appropriate for your system:

   - Convert dynamic elements to use `escapeHTML` and `displayString` (WOBrowser, WOPopUpButton, WOCheckBoxList, WONestedList).

   - Replace EOFetchSpecification hints with the appropriate new methods (see the EOFetchSpecification class reference).

Changes in both WebObjects template parsing and request handling may require you to make additional, manual adjustments to your WebObjects 3.5 applications. Refer to "Troubleshooting WebObjects 4 Template Parsing" (page 54) and "Troubleshooting WebObjects 4 Request Handling" (page 55) for details.

Converting Java Code From WebObjects 3.5

This section covers the details of converting any Java code you may have in an existing WebObjects application (see step 3 above). In WebObjects 4, the Java APIs changed considerably. These changes are summarized here:

- A two-letter prefix was added to each Java class name so that class names are unique without the package names. The Java class name is now identical to its Objective-C counterpart in almost all cases. For example, Component is now WOComponent, and WebApplication is now WOApplication.

- The Java package names changed to the following:

```
com.apple.yellow.eoaccess
com.apple.yellow.eocontrol
com.apple.yellow.foundation
com.apple.yellow.webobjects
```

Notice that the `next.eo` package was split into two packages: `eoaccess` and `eocontrol`.

- The basic classes (for arrays, dictionaries, and data) became more like their Foundation counterparts than their Java counterparts. For example, ImmutableVector is now named NSArray and responds to `count` instead of `size`. MutableHashtable is now named NSMutableDictionary and responds to `setObjectForKey` instead of `put`.

  Note that for numbers and strings, you still use the classes `java.lang.Number` and `java.lang.String`.

Warning: Changing to Foundation-style methods for the dictionary class introduces a subtle change. The Java Hashtable classes take the arguments in the key-value order. For example, the `put` method takes the key and then the value. NSDictionary takes the value and then the key. The conversion scripts change the order of the arguments for you. Unfortunately, these scripts incorrectly convert uses of `get()` and `put()` on java.util.Hashtable objects as well as on objects of other Foundation classes.

- DecimalNumber is no longer available. Use `java.math.BigDecimal` instead.
- CalendarDate is now named NSGregorianDate.
- The root object is now `com.apple.yellow.foundation.NSObject`.
- Delegate interfaces are now declared as inner interfaces of the appropriate class. For example, the DisplayGroupDelegates interface is now `WODisplayGroup.Delegates`.

Scripts are provided with the release to help you convert Java code to the new APIs. They are located in `/System/Developer/Java/Conversion/WebObjects` or, on NT, in

`$NEXT_ROOT/Developer/Java/Conversion/WebObjects`. Descriptions of these scripts and instructions for their use can be found in the ReadMe.html file, which is located in the same directory as the script files.

Troubleshooting WebObjects 4 Template Parsing

The WebObjects template parser parses the HTML that is to be included in a response. In WebObjects 4, the template parser preserves all of the static HTML that you provide in a component's template. Previously, the parser recognized many HTML tags and performed special processing based on the type of tag. The WebObjects 4 template parser ignores all tags besides `<WEBOBJECT>` and HTML comment tags.

The new parser has several advantages:

- It solves the problem many have encountered where WebObjects attempts to "fix" your HTML. For example, it previously was difficult to split a container element, such as a form, across two components because WebObjects would insert a closing tag for you.
- It improves your application's performance because it tends to treat larger parts of a file as a single chunk than the previous parser did.
- It allows you to suppress the copying of comments to the outgoing response. This speeds up response generation and shortens download times.

A WebObjects application may unknowingly depend upon the previous behavior of the template parser. For this reason, a compatibility flag is available on WOApplication to revert to the previous behavior.

Usually when WebObjects 4 template parsing produces an error, it is because you have included a WebObjects dynamic form element inside of a static HTML `FORM` element. Change the `FORM` element to a WOForm, and your component should operate normally again. An error may also arise if your HTML pages contain `BODY` or `IMG` tags with `src` parameters containing relative pathnames (absolute pathnames aren't a problem). Change the affected tags to WOBody and WOImage, respectively.

If you want, you can go back to the previous parser by implementing this method in your application class (shown in Java and WebScript):

```
public boolean requiresWOF35TemplateParser() {
  return true;
}
- requiresWOF35TemplateParser {
  return YES;
}
```

If you use the WebObjects 4 template parser, you might want to suppress the inclusion of HTML comments. Use WOApplication's `setIncludeCommentsInResponses:` method, or use the `WOIncludeCommentsInResponses` option described in "Starting Up Applications From the Command Line" in *Deploying WebObjects Applications*.

Troubleshooting WebObjects 4 Request Handling

In previous WebObjects releases, the application, session, request component, and all of the dynamic elements in the request component got a chance to perform the `takeValuesFromRequest:inContext:` and `invokeActionForRequest:inContext:` methods during each cycle of the request-response loop. In WebObjects 4, there are some performance enhancements to this request-handling scheme:

- The take values phase is not always performed.

  If the request has no form values to use as input, the take values phase of the request-response loop (in which the application, the session, the request component, and the component's dynamic elements are sent `takeValuesFromRequest:inContext:`) is not performed.

If you have overridden `takeValuesFromRequest:inContext:` at the application, session, or component level and your method needs to be invoked even when there are no input values, you must either change your logic or disable WebObjects 4 request handling at the application level. To disable WebObjects 4 request handling, implement the following method in your application class:

```
//WARNING! Put this method in Application class, not component.
//Java implementation
public boolean requiresWOF35RequestHandling() {
  return true;
}
//WebScript implementation
- requiresWOF35RequestHandling {
  return YES;
}
```

It is the application object that makes the decision to perform the take values phase of the request-response loop; therefore, you must disable WebObjects 4 request handling in the application class if you want to ensure that the take values phase always occurs.

- The take values phase does not iterate through WOBrowser and WOPopUpButton lists.

  In previous releases, WebObjects would iterate through the `list` attribute of the WOBrowser and WOPopUpButton looking for the item that the user selected. As of release 4 this is no longer necessary because WebObjects can directly access the selected item without iterating. WebObjects is able to do this because the use of the `value` attribute has changed so that by default it is set to the index of the item.

  - Use of the `item` attribute as the selection.

    The `item` attribute is intended to point to the current item, and it is updated upon each iteration through the list. Because WebObjects used to iterate through the list until it found a selection, the `item` attribute ended up pointing to the selected item.

If you need to refer to the selected item, use the `selection` attribute instead of `item`. Make sure `selection` is bound to a variable in your component's code and then use that variable instead of the one bound to `item`.

- Use of the `value` attribute.

  The `value` attribute was previously used as the string displayed in the browser or pop-up button. It also set the HTML value attribute for the `<OPTION>` tag. In WebObjects 4, this attribute still sets the value in HTML, but it no longer specifies the display string. By default, it is set to an index value, which allows WebObjects to find the selection without iterating through the list.

  If you have a binding for the `value` attribute, change it to `displayString`, which is a new attribute that specifies the display string. Change this:

  ```
  value = aCollege.name;
  ```

  to this:

  ```
  displayString = aCollege.name;
  ```

  Use `value` only if you really want to set the HTML value in the `<OPTION>` tag.

- An `item` attribute bound to a method.

  If you bound the `item` attribute to a method, your method used to be invoked several times during the take values phase, and now it is invoked only once (for the selected item). If your component depends upon the previous behavior, you either need to change your logic or use WOApplication's request-handling compatibility flag as described above.

• The invoke action phase does not iterate through a WORepetition's list.

When a repetition's list is iterated upon, the `item` and `index` attribute values are updated at each iteration. In previous releases, list iteration occurred during the take values phase and during the invoke action phase of the request-response loop. In WebObjects 4, WORepetition list iteration occurs during take values only if the request has input values, and it doesn't occur during the invoke action phase. (WebObjects is able to forgo iterating during the invoke action phase because by default it sets the `identifier` attribute to the item's element ID so that it is able to navigate directly to the list item that responds to the requested action. If you already declare a binding for the `identifier` attribute, your binding is used instead of the element ID, and the invoke action phase does iterate through the list.)

If you've bound the `item` or `index` attribute to a method, your method used to be invoked several times during the take values phase, and then again several more times during the invoke action phase. In WebObjects 4, your method will only be invoked during the take values phase if there are input values in the request, and it won't be invoked during the invoke action phase (unless you specify a non-default binding for the `identifier` attribute).

If your component depends upon the previous behavior, you either need to change your logic or use the component's request-handling compatibility flag. To set the component's request handling compatibility flag, implement this method in the component:

```
// Java implementation
public boolean requiresWOF35RequestHandling() {
  return true;
}
// WebScript implementation
- requiresWOF35RequestHandling {
  return YES;
}
```

When you implement this method at the component level, WebObjects uses the old behavior for invoke action on that component only. All other components use the new behavior for the invoke action phase.

WebScript Changes

In WebObjects 3.5, WebScript would always evaluate both sides of an "&&" or "||" expression. In WebObjects 4, these expressions are short-circuited, so that only the left side is evaluated unless evaluation of the right side is necessary in order to determine the result. For example:

```
(YES || <this will NOT evaluate>)
(NO  || <this will evaluate>)
(YES && <this will evaluate>)
(NO  && <this will NOT evaluate>)
```

To aid in the debugging process, WebObjects 4 has a WebScript 3.5 compatibility mode. This mode is controlled by a method in WOApplication named `requiresWOF35Scripting`. By default, this method returns NO; override it to return YES to get backward compatibility.

Chapter 4

Uninstalling WebObjects

Uninstalling on Windows NT

To uninstall WebObjects Developer or Deployment on a Windows NT system, stop any running WebObjects applications and then select Programs > WebObjects > Uninstall WebObjects Developer from the Start menu. You'll be presented with a dialog asking you to confirm the deletion of all WebObjects programs and files; click Yes to uninstall WebObjects.

The uninstall process removes WebObjects itself (and cleans up the Windows NT registry), but doesn't remove any applications you may have created with WebObjects. Thus, after uninstalling the WebObjects product you'll need to manually delete any WebObjects applications you created or installed on your Windows NT computer.

Cleaning Up After a Failed Installation

If a problem occurred during installation and WebObjects isn't listed under the Start menu, you can remove the partial installation using software provided on the WebObjects CD. Within the Install directory on the WebObjects CD is a WebObjects 4.5 directory; this directory contains the uninstallation software. Follow the instructions in the `readme.txt` file to both remove any installed files and clean up the Windows NT registry. Note that you can safely skip the first step, which tells you to run the uninstaller.

Uninstalling on Mac OS X Server

Uninstalling WebObjects on a computer running Mac OS X Server is simply a matter of double-clicking the installed packages in the `/Local/Library/Receipts` directory (which opens the packages in the Installer application) and clicking the Delete button (note that you must be logged in as `root` in order to uninstall WebObjects). If you did a Custom install you'll need to remove those packages that you selected. If you did a Complete install, you'll need to remove the following:

WebObjects Developer

- WebObjectsLicense.pkg
- DeveloperDoc.pkg
- DeveloperTools.pkg
- ProfileLibs.pkg
- Emaces.pkg
- Source.pkg
- EOUser.pkg
- WebObjectsAdaptors.pkg
- WebObjectsDeployment.pkg
- EODeveloper.pkg
- LDAPEOAdaptor.pkg
- OracleEOAdaptor.pkg
- OpenBaseLiteAdaptor.pkg
- WebObjectsExamples.pkg
- WebObjectsDeveloper.pkg

WebObjects Deployment

- WebObjectsLicense.pkg
- EOUser.pkg
- WebObjectsAdaptors.pkg
- WebObjectsDeployment.pkg
- LDAPEOAdaptor.pkg
- OracleEOAdaptor.pkg

Note that a Complete install of either WebObjects Developer or Deployment installs the MacOSXServer1.0-2.pkg; this package is a system software patch for Mac OS X Server and shouldn't be removed along with the WebObjects packages listed above.

Uninstalling on Solaris and HP-UX

On both Solaris and HP-UX, a shell script—named `uninstall.sh`—is placed in the `Library/Executables` directory (relative to the directory in which you installed WebObjects). To uninstall WebObjects, you need only log in as `root` and execute this script. You'll be asked whether you really mean to uninstall WebObjects; type ‘y’ to proceed with the uninstallation, or any other character to abort the process.

Cleaning Up After a Failed Uninstall

If for some reason the uninstall fails or is prematurely aborted, you can perform the following steps to uninstall WebObjects manually.

1. Log in as `root`.
2. Remove the contents of the `NEXT_ROOT` directory with:

   ```
   rm -rf NEXT_ROOT
   ```

3. Remove the `ApplePDOstartup` scripts from `/etc` with:

   ```
   cd /etc
   find . -name ‘*Apple*’ -print | xargs rm -f
   ```